

000 AI-SEARCHPLANNER: MODULAR AGENTIC SEARCH 001 VIA PARETO-OPTIMAL MULTI-OBJECTIVE REIN- 002 FORCEMENT LEARNING 003 004

006 **Anonymous authors**

007 Paper under double-blind review

010 ABSTRACT

013 Recent studies have explored integrating Large Language Models (LLMs) with
014 search engines to leverage both the LLMs' internal pre-trained knowledge and
015 external information. Specially, reinforcement learning (RL) has emerged as a
016 promising paradigm for enhancing LLM reasoning through multi-turn interactions
017 with search engines. However, existing RL-based search agents rely on a single
018 LLM to handle both search planning and question-answering (QA) tasks in an
019 end-to-end manner, which limits their ability to optimize both capabilities simulta-
020 neously. In practice, sophisticated AI search systems often employ a large, frozen
021 LLM (e.g., GPT-4, DeepSeek-R1) to ensure high-quality QA. Thus, a more effec-
022 tive and efficient approach is to utilize a small, trainable LLM dedicated to search
023 planning. In this paper, we propose **AI-SearchPlanner**, a novel reinforcement
024 learning framework designed to enhance the performance of frozen QA models
025 by focusing on search planning. Specifically, our approach introduces three key
026 innovations: 1) Decoupling the Architecture of the Search Planner and Generator,
027 2) Dual-Reward Alignment for Search Planning, and 3) Pareto Optimization of
028 Planning Utility and Cost, to achieve the objectives. Extensive experiments on
029 real-world datasets demonstrate that AI SearchPlanner outperforms existing RL-
030 based search agents in both effectiveness and efficiency, while exhibiting strong
031 generalization capabilities across diverse frozen QA models and data domains.

032 1 INTRODUCTION

034 Recent research has investigated combining Large Language Models (LLMs) with search engines
035 to harness both the LLMs' internal knowledge and external information retrieval (IR), thereby en-
036 hancing performance on complex reasoning tasks. The early representative paradigm, Retrieval-
037 Augmented Generation (RAG), retrieves relevant passages based on input queries and incorporates
038 them into the LLM's context for generation. While this approach facilitates the utilization of exter-
039 nal knowledge, most RAG systems rely on static retrieval mechanisms, lacking adaptive planning
040 capabilities for complex reasoning. Consequently, their performance remains highly sensitive to
041 retrieval accuracy. With advancements in the general capabilities of LLMs, recent research has
042 explored leveraging search engines as tools — primarily through prompt-based methods or self-
043 supervised fine-tuning (SFT). These approaches enable LLMs to autonomously determine whether
044 and how to invoke search engines based on the problem-solving context. However, prompt-based
045 methods often exhibit limited generalization, particularly for complex reasoning tasks that fall out-
046 side the LLM's pretraining distribution. Although SFT-based methods offer greater adaptability,
047 they require large-scale, high-quality annotated trajectories of search-reasoning interactions, posing
048 scalability challenges. Furthermore, recent studies indicate that SFT methods tend to overfit to mem-
049 orized reasoning paths, further constraining their generalization to out-of-distribution scenarios.

050 Recently, reinforcement learning (RL) has emerged as a promising paradigm for improving the rea-
051 soning capabilities of LLMs. Models such as OpenAI-o1¹ and DeepSeek-R1 Guo et al. (2025)
052 employ RL techniques (e.g., PPO Schulman et al. (2017), GRPO Shao et al. (2024)) to enhance log-
053 ical reasoning and problem-solving through iterative experiential learning. Research like Search-R1

¹<https://openai.com/zh-Hans-CN/index/learning-to-reason-with-lmms/>

Jin et al. (2025) further highlights the potential of RL by leveraging question-answering rewards in multi-turn reasoning to train LLMs for search engine interactions. This approach yields significant performance improvements, demonstrating RL’s effectiveness in enhancing LLM-based search utilization. Although RL-based search agents have demonstrated superior performance in complex reasoning tasks, existing approaches typically rely on a single LLM to handle both search planning and QA tasks in an end-to-end manner, which poses significant challenges in optimizing both capabilities simultaneously. In particular, in practical scenarios, sophisticated AI search systems such as Baidu² and Tencent Yuanbao³ commonly utilize a large frozen LLM (e.g., GPT-4 Achiam et al. (2023), DeepSeek-R1 Guo et al. (2025)) to ensure high-quality QA. Thus, a more effective and efficient solution is to employ a small, trainable LLM dedicated to search planning, thereby improving QA performance while maintaining low computational latency.

To address this challenge, in this paper, we propose **AI-SearchPlanner**, a novel reinforcement learning framework designed to enhance end-to-end QA performance by focusing on search planning. Specifically, our approach introduces three innovations to achieve the objectives:

- 1) **Decoupling the Architecture of the Search Planner and Generator:** We offload QA functionality to a large, frozen generator LLM, and make another small, trainable search planner LLM focus on search planning, thereby ensuring flexibility for real-world applications.
- 2) **Dual-Reward Alignment for Search Planning:** We design a dual-reward mechanism to align search planning capabilities at two levels. At the outcome level, an outcome reward quantifies the performance gain of search planning over non-planning baselines (e.g., direct inference or naive RAG). At the process level, a process reward evaluates the rationality of the search planning trajectory. Together, these rewards ensure precise alignment of search planning abilities.
- 3) **Pareto Optimization of Planning Utility and Cost:** In real-world scenarios, the trade-off between search planning effectiveness (e.g., end-to-end QA accuracy) and computational overhead (e.g., reasoning turns) significantly impacts user experience. We formalize search planning as a pareto optimization problem to jointly maximize planning utility while minimizing planning cost.

To demonstrate the effectiveness of our proposed AI-SearchPlanner, we conduct extensive experiments across multiple search reasoning datasets, and observe that AI-SearchPlanner outperforms existing RL-based search agents in both effectiveness and efficiency, while exhibiting strong generalization capabilities across diverse frozen QA models and data domains.

2 RELATED WORK

Although large language models (LLMs) Vaswani et al. (2017); Achiam et al. (2023); Zhao et al. (2023) exhibit remarkable general question-answering capabilities, they still lack the ability to leverage external domain-specific knowledge to solve complex reasoning problems. To mitigate these limitations, search engines are commonly employed to augment LLMs with external information. There are two primary approaches to integrating search engines with LLMs: (1) retrieval-augmented generation (RAG) Lewis et al. (2020); Gao et al. (2023); Mei et al. (2025) and (2) treating the search engine as a tool within an agent framework Schick et al. (2023); Jin et al. (2025); Qu et al. (2025); Song et al. (2025); Wu et al. (2025a); Li et al. (2025a). Retrieval-Augmented Generation (RAG) Lewis et al. (2020); Gao et al. (2023); Mei et al. (2025) enhances LLMs by integrating retrieved external knowledge into the input, thereby providing access to up-to-date or domain-specific information. However, most existing RAG systems Ma et al. (2023); Jiang et al. (2023b;a) follow rigid retrieval-generation pipelines, which lack adaptive planning mechanisms for complex reasoning. As a result, their performance remains heavily dependent on retrieval accuracy, limiting their robustness in dynamic or knowledge-intensive scenarios.

With the advancement of the general capabilities of LLMs, recent research Qu et al. (2025); Trivedi et al. (2022a); Yao et al. (2023); Schick et al. (2023) has explored integrating search engines as tools to enhance problem-solving. Specially, prompt-based methods Trivedi et al. (2022a); Yao et al. (2023) and self-supervised fine-tuning (SFT) approaches Schick et al. (2023) have been proposed to enable LLMs to autonomously determine when and how to invoke search engines during reasoning. For instance, IRCoT Trivedi et al. (2022a) and ReAct Yao et al. (2023) utilize iterative prompting strategies to interleave reasoning with search engine calls, while Toolformer Schick et al. (2023)

²<https://chat.baidu.com/>

³<https://yuanbao.tencent.com/chat/>

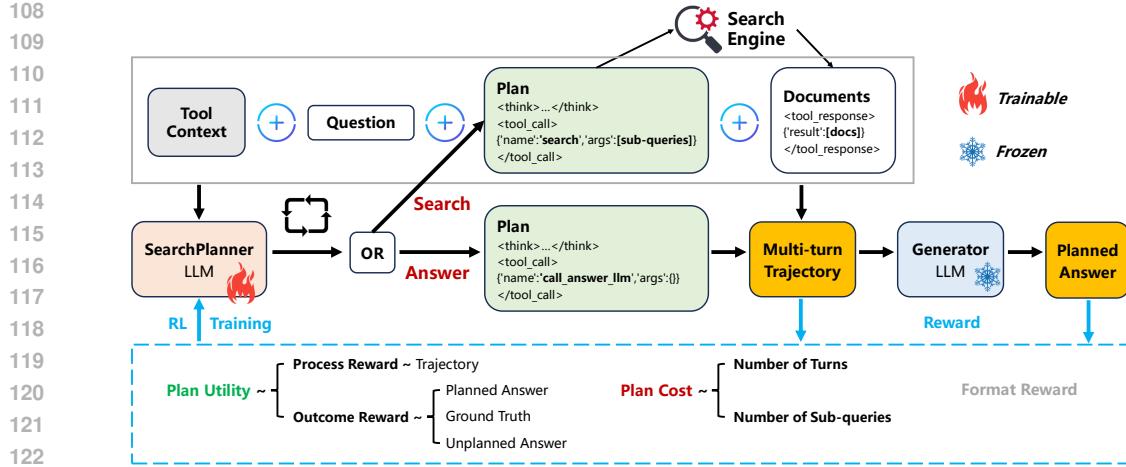


Figure 1: The overview of AI-SearchPlanner framework.

leverages supervised fine-tuning to improve search-related capabilities. However, prompt-based methods often exhibit limited generalization, especially for complex reasoning tasks that lie outside the pretraining distribution of LLMs. SFT-based methods often depend on large-scale, high-quality annotated trajectories of search-reasoning interactions, which limits their scalability. Moreover, recent studies suggest that SFT-based approaches tend to overfit to memorized reasoning paths, further constraining their generalization to novel or out-of-distribution scenarios. Recent studies Jin et al. (2025); Song et al. (2025); Wu et al. (2025a); Li et al. (2025a;c) demonstrate that reinforcement learning (RL) can enable LLMs to develop advanced reasoning capabilities using only outcome-based rewards. For example, Search-R1-like approaches Jin et al. (2025); Song et al. (2025) have investigated the potential of RL methods such as PPO Schulman et al. (2017) and GRPO Shao et al. (2024) in agentic search scenarios. By employing outcome-based reward mechanisms and end-to-end training, these methods enable LLMs to jointly optimize multi-turn search planning and answer generation, achieving strong QA performance. A related concurrent work is s3 Jiang et al. (2025), which also decouples the planner from the generator. Our approach differs from s3 as follows: 1) Objective: s3 aims to train a high-performing planner with minimal data, while we focus on industrial applications, optimizing the planner for the best utility-cost trade-off using state-of-the-art LLMs. 2) Framework: s3 provides the generator with key documents from the final reasoning step; we instead supply the entire planning trajectory. 3) Method Design: s3 introduces a Gain Beyond RAG reward and filters out training examples solvable by naive RAG, thereby improving end-to-end QA performance on complex questions. We directly optimize planning ability with an outcome-process dual reward and incorporates Pareto optimization of utility and cost, significantly reducing cost while maintaining performance. Additionally, another similar line of recent work Xiao et al. (2024); Shao et al. (2025), ReasoningIR, enhance IR models with intrinsic reasoning ability, by using LLM reasoning to generate query-document pairs for training. In contrast, we leverages query–search–document planning trajectories to directly produce high-quality answers.

3 AI-SEARCHPLANNER

In this section, we introduce AI-SearchPlanner, a novel reinforcement learning framework designed to enhance end-to-end QA performance by focusing on search planning. As illustrated in Figure 1, we demonstrate AI-SearchPlanner framework. Given a question q and its ground truth answer gt , a trainable search planner LLM_{plan} iteratively interacts with a search engine $S(\cdot)$: At each turn t , LLM_{plan} first generates reasoning context, and then decides to either: 1) Generates sub-queries $\{sq\}^t$ within search tool to retrieve relevant documents via $S(\cdot)$, or 2) Terminates reasoning by generating `call_answer_llm` tool, which feeds the accumulated knowledge from the reasoning trajectory into a frozen generator LLM_{gen} to produce an answer a . The final multi-turn reasoning trajectory is represented as $T = \{q, r_1^S, S(\{sq\}^1), \dots, r_L^S, S(\{sq\}^L), r^A\}$, where r_*^S and r^A refer to the planning contexts of searching sub-queries and invoking a frozen generator, respectively. In the following, we introduce three key innovations in AI-SearchPlanner to achieve the objectives.

162 3.1 DECOUPLING THE ARCHITECTURE OF THE SEARCH PLANNER AND GENERATOR
163

164 In practical applications, advanced AI search systems often rely on large, frozen LLMs to ensure
165 high-quality QA. Thus, to balance effectiveness and efficiency, a more practical approach is to em-
166 ploy a small, trainable LLM dedicated to search planning — enhancing QA performance while
167 maintaining low latency. As shown in Figure 1, our framework offloads QA functionality to a large,
168 frozen generator LLM_{gen} , and make another small, trainable search planner LLM_{plan} focus on
169 search planning, thereby ensuring flexibility for real-world applications. Specifically, LLM_{plan} it-
170 eratively interacts with the search engine, dynamically deciding whether to generate sub-queries
171 and retrieve relevant documents, or feed the accumulated context from the reasoning trajectory into
172 LLM_{gen} to generate the answer.

173 3.2 DUAL-REWARD ALIGNMENT FOR SEARCH PLANNING
174

175 Unlike existing methods that rely on a single LLM to jointly handle both search planning and ques-
176 tion answering (QA) — aligning end-to-end QA capabilities, we propose a dual-reward mechanism
177 to explicitly align search planning abilities.

178 • **Outcome Reward** measures the performance gain achieved by search planning compared to
179 non-planning baselines (e.g., direct inference or naive RAG). Let a , a_I , and a_R denote the an-
180 swers produced by search planning, direct inference, and naive RAG, respectively. The function
181 $Score(\cdot)$ is used to evaluate the consistency between an LLM-generated answer and the corre-
182 sponding ground truth answer gt , with a value within $\in \{0, 1\}$. Specifically, the performance
183 gains of search planning vary across different problems, presenting four different pairs of states,
184 each associated with a specific reward ranking.

$$185 \quad 186 \quad Rank \left(\left\langle Score(a, gt), \max \left\{ \begin{array}{l} Score(a_I, gt) \\ Score(a_R, gt) \end{array} \right\} \right\rangle \right) \Rightarrow \langle 1, 0 \rangle > \langle 1, 1 \rangle > \langle 0, 0 \rangle > \langle 0, 1 \rangle \quad (1)$$

188 These above reward rankings incorporates both answer quality and the inherent planning com-
189 plexity of the questions. The final reward metric can be formalized as follows:

$$191 \quad R_{outcome} = \frac{1}{2} + Score(a, gt) - \frac{1}{2} * \max \left\{ \begin{array}{l} Score(a_I, gt) \\ Score(a_R, gt) \end{array} \right\} \in [0, 1.5] \quad (2)$$

193 • **Process Reward** evaluates the rationality of the search planning trajectory T through the frozen
194 generator LLM_{gen} and corresponding prompt P_T .

$$196 \quad R_{process} = LLM_{gen}(T, P_T) \in [0, 0.5] \quad (3)$$

197 **Dual Reward** combines outcome and process rewards to constitute the utility of search planning,
198 ensuring precise alignment with search planning capabilities.

$$200 \quad R_{utility} = R_{outcome} + R_{process} \quad (4)$$

202 3.3 PARETO OPTIMIZATION OF PLANNING UTILITY AND COST
203

204 In real-world applications, the trade-off between search planning effectiveness (e.g., end-to-end QA
205 performance) and computational overhead (e.g., search frequency, reasoning turns) critically influ-
206 ences user experience. We formalize search planning as a pareto optimization problem to sim-
207 taneously maximize utility while minimizing cost. The overall cost of the planning trajectory T
208 comprises two key components: 1) **The number of planning turns L** : Directly impacts inference
209 latency during planning. 2) **The number of sub-queries $\sum_i^L |\{sq\}^i|$** : Determines the frequency of
210 search engine invocations. We formalize the final planning cost as:

$$212 \quad R_{cost} = R_{cost}^{turn} + R_{cost}^{query} \quad (5)$$

$$214 \quad R_{cost}^{turn} = \max \left(0, 1 - \frac{L}{M_t} \right), \quad R_{cost}^{query} = \max \left(0, 1 - \frac{\sum_i^L |\{sq\}^i|}{M_q} \right) \quad (6)$$

216 Where M_t and M_q denote the maximum thresholds for the number of planning turns and sub-
 217 queries, respectively. To jointly optimize utility maximization and cost minimization in search plan-
 218 ning, we formulate the following Pareto optimization problem.

$$219 \quad R_{\text{pareto}} = R_{\text{utility}} + \alpha * R_{\text{cost}} + R_{\text{format}} \quad (7)$$

221 Where α is a non-negative coefficient. By adjusting α , we guide the search planning towards differ-
 222 ent trade-offs between utility and cost, enabling exploration of the Pareto frontier. R_{format} denotes
 223 the reward for format correctness.

225 3.4 REINFORCEMENT LEARNING TRAINING

227 We implement a reinforcement learning approach to optimize the search planner’s decision-making
 228 on when to invoke the search engine or the frozen generator.

229 • **Reward Design.** We train the search planner to integrate search during inference, enhancing
 230 problem-solving accuracy while reducing planning costs. Let n_S and n_A denote the number of
 231 search engine and frozen generator invocations, respectively.

$$232 \quad R = R_{\text{pareto}}, \quad s.t. \quad R_{\text{format}} = \begin{cases} 0, & \text{if format} = \checkmark \text{ & if } n_S \geq 1 \text{ & } n_A \geq 1 \\ -1, & \text{else} \end{cases} \quad (8)$$

235 • **Loss Masking for Retrieved Tokens.** During reinforcement learning training, the rollout se-
 236 quence consists of LLM-generated tokens and retrieved tokens. Since retrieved documents serve
 237 as environmental observations rather than LLM-generated content, we apply a loss masking strat-
 238 egic to exclude retrieved tokens from gradient computation. This ensures that the policy gradient
 239 objective is computed solely over LLM-generated tokens, preventing interference from retrieved
 240 content and preserving the model’s intrinsic reasoning and generation capabilities.

241 • **Optimization of Search Planner.** We optimize the search planning policy using reinforcement
 242 learning with a dual reward formulation. Each search planning trajectory comprises an input
 243 question, a sequence of retrieved documents, and a termination decision (i.e., call generator to
 244 answer). Upon constructing the final context, a frozen generator generates the answer, and the
 245 dual reward is computed. In this work, we employ Proximal Policy Optimization (PPO) Schulman
 246 et al. (2017). The corresponding PPO objective is defined as:

$$247 \quad \mathbb{L}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \min \left(\frac{\pi_\theta(a_t | s_t)}{\pi_{\text{old}}(a_t | s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t | s_t)}{\pi_{\text{old}}(a_t | s_t)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right] \quad (9)$$

249 where π_θ and π_{old} denote the current and reference policy models, respectively. A_t corresponds to
 250 the estimated advantage. ϵ is the clipping threshold introduced in PPO to ensure training stability.

252 4 EXPERIMENT

254 We conduct extensive experiments on multiple benchmark datasets, and validate the effectiveness of
 255 our proposed AI-SearchPlanner.

257 4.1 DATASETS

259 To evaluate AI-SearchPlanner, we use seven Wikipedia-based datasets and two Web-based datasets:
 260 1) **Wiki-based:** NQ Kwiatkowski et al. (2019), TriviaQA Joshi et al. (2017), PopQA Mallen et al.
 261 (2022), HotpotQA Yang et al. (2018), 2WikiMultiHopQA Ho et al. (2020), Musique Trivedi et al.
 262 (2022b), and Bamboogle Press et al. (2022). Specially, the first three are general QA datasets,
 263 and the last four are multi-hop QA datasets. 2) **Web-based:** WebShaper Tao et al. (2025) and
 264 WebWalkerQA Wu et al. (2025b).

266 4.2 BASELINES

268 To evaluate the effectiveness of AI SearchPlanner, we compare it with the following baseline meth-
 269 ods that do not decouple the planner and generator: 1) **Inference without Retrieval:** Direct in-
 ference and Chain-of-Thought (CoT) reasoning Wei et al. (2022). 2) **Inference with Retrieval:**

Table 1: Performance comparison of all methods on Wikipedia-based datasets. We highlight the best performance among the trainable or frozen models in the same series.

Method	General QA			Multi-Hop QA				Avg.	
	NQ	TriviaQA	PopQA	HotpotQA	2wiki	Musique	Bamboogle*		
Non-decoupled Planner-Generator									
⇒ Qwen2.5-7b-Instruct									
↓ Planner (Frozen) & Generator (Frozen)									
Direct Inference	0.325	0.507	0.196	0.301	0.268	0.135	0.192	0.275	
Naive RAG	0.585	0.753	0.489	0.478	0.325	0.162	0.408	0.457	
CoT	0.370	0.583	0.178	0.299	0.278	0.109	0.432	0.321	
IRCoT	0.537	0.703	0.442	0.394	0.194	0.105	0.336	0.387	
Search-ol	0.462	0.652	0.350	0.441	0.400	0.198	0.472	0.425	
↓ Planner (Trainable) & Generator (Trainable)									
SFT	0.556	0.553	0.177	0.384	0.269	0.166	0.400	0.358	
Search-R1	0.630	0.771	0.503	0.571	0.402	0.215	0.544	0.519	
Generator Only (Frozen)									
⇒ Qwen3-32b									
Direct Inference	0.486	0.737	0.276	0.451	0.364	0.218	0.648	0.454	
Naive RAG	0.652	0.824	0.517	0.587	0.397	0.229	0.568	0.539	
⇒ Deepseek-V3									
Direct Inference	0.604	0.870	0.447	0.519	0.425	0.254	0.432	0.507	
Naive RAG	0.670	0.840	0.527	0.606	0.438	0.241	0.424	0.535	
⇒ Deepseek-R1									
Direct Inference	0.657	0.903	0.499	0.655	0.594	0.365	0.632	0.615	
Naive RAG	0.668	0.903	0.495	0.661	0.599	0.369	0.648	0.620	
Decoupled Planner-Generator									
⇒ AI-SearchPlanner									
↓ Planner (Trainable): Qwen2.5-7b-Instruct & Generator (Frozen): Qwen3-32b									
α = 0	0.674	0.800	0.540	0.678	0.565	0.355	0.568	0.597(+10.76%)	
Generator Transferability									
Qwen2.5-7b-Instruct	0.628	0.764	0.517	0.638	0.462	0.320	0.552	0.554(+6.74%)	
Deepseek-V3	0.678	0.833	0.534	0.671	0.562	0.349	0.640	0.610(+14.02%)	
Deepseek-R1	0.719	0.880	0.549	0.741	0.617	0.373	0.656	0.648(+4.52%)	

Naive Retrieval-Augmented Generation (RAG) Lewis et al. (2020), IRCoT Trivedi et al. (2022a), and Search-o1 Li et al. (2025b). 3) **Training-based Methods**: Supervised fine-tuning (SFT) Chung et al. (2024) and Search-R1 Jin et al. (2025).

4.3 EXPERIMENTAL SETTINGS

For training AI-SearchPlanner, We merge the training sets of NQ and HotpotQA to form a unified dataset. Qwen-2.5-7b-Instruct⁴ and Qwen3-32b⁵ were used as the trainable planner and frozen generator, respectively. The maximum thresholds for the number of reasoning turns and sub-queries M_t and M_q is set to 5 and 10, respectively. The prompts for evaluating process reward and answer accuracy are provided in Appendix B. Since different frozen QA models may produce answers with divergent content and stylistic variations, the Exact Match (EM) metric is inadequate for measuring answer correctness. To address this limitation, we introduce an LLM-based scoring function $Score(\cdot)$ that evaluates the answer accuracy of model responses based on the question and the ground truth answer. On the Wikipedia-based datasets, we follow the settings in Jin et al. (2025) to conduct the experiments. We use the 2018 Wikipedia dump Karpukhin et al. (2020) as the knowledge source and E5 Wang et al. (2022) as the retriever. The number of retrieved passages is set to 3. The maximum number of reasoning turns is set to 5. On the Web-based datasets, we use Google search API⁶ as the knowledge retriever. The number of retrieved titles and snippets is set to 10. The maximum number of reasoning turns is set to 10. We provide the case study in Appendix A.

⁴<https://huggingface.co/Qwen/Qwen2.5-7B-Instruct>

⁵<https://huggingface.co/Qwen/Qwen3-32B>

⁶<https://serper.dev/>

324 Table 2: Performance Comparison of AI-Searchplanner transferred to Web-based data domains.
325

326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377	326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
	Method	Web QA
	WebShaper	WebWalker
⇒ Qwen2.5-7b-Instruct		
Direct Inference	0.110	0.054
Naive RAG	0.188	0.297
CoT	0.104	0.022
IRCoT	0.140	0.228
Search-ol	0.186	0.231
⇒ Qwen3-32b		
Direct Inference	0.170	0.057
Naive RAG	0.204	0.299
⇒ AI-SearchPlanner		
↓ Planner (Trainable): Qwen2.5-7b-Instruct & Generator (Frozen): Qwen3-32b		
α = 0	0.366(+79.41%)	0.375(+25.42%)
Generator Transferability		
Qwen2.5-7b-Instruct	0.322(+71.28%)	0.306(+3.03%)

341 4.4 RESULTS

342 In this section, we demonstrate our experimental results, and provide detailed analysis.

343 4.4.1 OVERALL PERFORMANCE

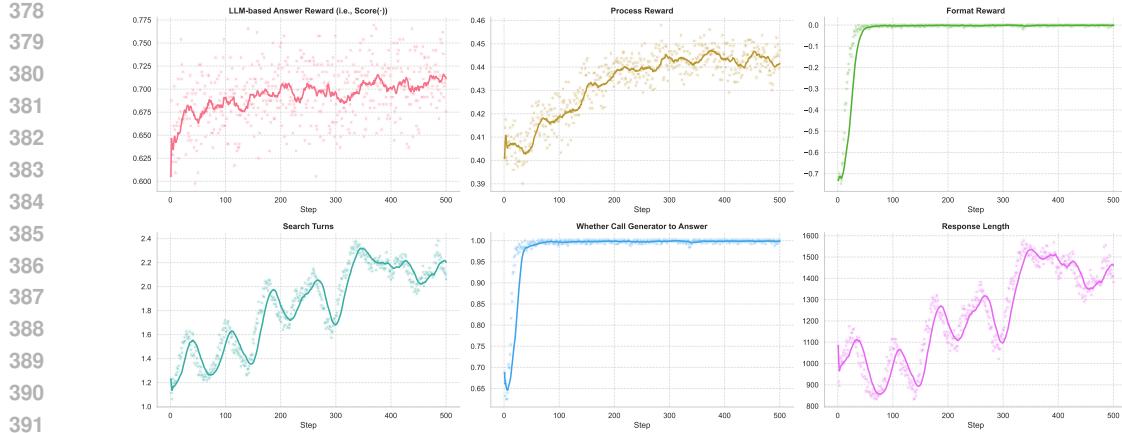
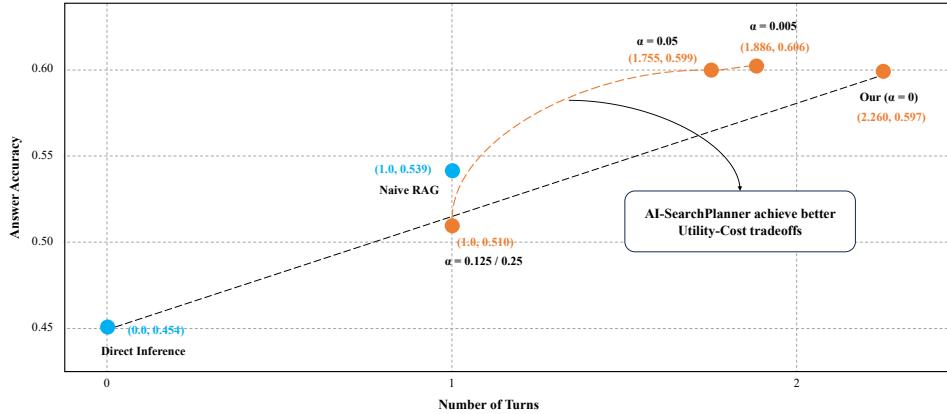
344 The experimental results on Wikipedia-based datasets are given in Table 1. For the overall performance, AI-SearchPlanner state-of-the-art performance, substantially outperforming all baseline methods. In detail, in terms of **Search Planning Ability**, the results indicate that: (1) When employing the same frozen generator, AI-SearchPlanner yields significant performance gains over both non-planning baselines (e.g., direct inference and naive RAG) and existing planning-based methods; (2) Notably, the model exhibits more pronounced improvements on multi-hop questions compared to single-hop questions. This indicates the capability of dual reward-aligned search planning mechanism optimized via reinforcement learning, highlighting its superior reasoning and retrieval strategies in addressing complex questions. In terms of **Architecture Decoupling**, we can find that: When transferring to a larger frozen generator, AI-SearchPlanner maintains consistent performance improvements over both generator-only baselines and non-decoupled architectures. This result validates the advantage of decoupling the planner module from the generator, as it allows the planner to specialize explicitly in search planning while facilitating seamless integration with more powerful generators. Consequently, this flexibility leads to robust performance enhancements without requiring model retraining. Additionally, on the Wiki dataset, we observe that the performance gain of naive RAG over direct inference diminishes with more powerful generators. This suggests that stronger generators may have memorized more comprehensive wiki knowledge during pre-training.

345 4.4.2 DOMAIN TRANSFERABILITY

346 In Section 4.4.1, we have demonstrated the transferability of AI-SearchPlanner across different frozen generators. In this section, we discuss the transferability of the wiki-trained AI-SearchPlanner on web-based datasets and search engines. As shown in Table 2, AI-SearchPlanner consistently outperforms baseline methods in overall answer accuracy, even when adapted to out-of-domain datasets and search engines. These results demonstrate the robustness and strong transferability of the learned search planning capability. Moreover, on more complex web-based datasets, untrained planning baselines performs even worse than no-planning baselines. This indicates that oversimplified search planning strategies can be detrimental to solving challenging reasoning problems.

347 4.4.3 TRAINING DYNAMICS

348 In Figure 2, we present the key dynamics during the training process of AI-SearchPlanner with cost coefficient $\alpha = 0$. For **Reward-related Dynamics**, the first row of Figure 2 illustrates three key rewards influencing training performance: LLM-based answer reward, process reward, and format reward. As training progresses, all three rewards exhibit consistent improvement, with the format

Figure 2: Training dynamics of AI-SearchPlanner with cost coefficient $\alpha = 0$.Figure 3: Utility-Cost tradeoffs on Wikipedia-based datasets. Blue points represent non-planning baselines. Orange points represent AI-SearchPlanner with different cost coefficient α .

reward converging significantly faster than the other two. This suggests that AI-SearchPlanner effectively enhances both the quality of generated answers and the rationality of reasoning trajectories, while format correctness is learned earlier in the training process. For **Action-related Dynamics**, the second row of Figure 2 presents three important actions during the training process: number of turns calling the search engine, whether the frozen generator is invoked, and response length. We observe that, as training steps increase: 1) AI-SearchPlanner learns to call the search engine more frequently to solve problems, resulting in longer responses; 2) After obtaining sufficient external knowledge from multi-turn searches, AI-SearchPlanner quickly learns to terminate reasoning and invoke the frozen generator to produce answers. This demonstrates that during training, AI-SearchPlanner learns how to utilize the search engine and the frozen generator to address complex reasoning problems.

4.4.4 PARETO FRONTIER ANALYSIS OF UTILITY-COST

We trained four variants of AI-SearchPlanner by varying the cost coefficient $\alpha \in \{0.005, 0.05, 0.125, 0.25\}$. The experimental results are given in Figure 3 and Table 3, we can find that, AI-SearchPlanner achieves a compelling Pareto frontier compared to non-planning baselines. Specifically, as the cost coefficient α increases, the number of search reasoning turns employed by AI-SearchPlanner continuously decreases, eventually converging to 1, while the answer performance initially exhibits a slight improvement before undergoing a consistent decline. This highlights AI-SearchPlanner’s effectiveness in navigating the trade-off between performance and cost.

432 Table 3: Performance and cost comparison of AI-SearchPlanner with different cost coefficient α on
 433 Wikipedia-based Datasets. The numbers in [] represent the search planning turns. We highlight the
 434 best performance among these models.

435

436 Method	437 General QA			438 Multi-Hop QA			439 Avg.
	440 NQ	441 TriviaQA	442 PopQA	443 HotpotQA	444 2wiki	445 Musique	
\Rightarrow AI-SearchPlanner							
$\alpha = 0$	0.674 [1.856]	0.800 [1.910]	0.540 [1.749]	0.678 [2.352]	0.565 [2.681]	0.355 [2.867]	0.568 [2.408] 0.597 [2.260]
$\alpha = 0.005$	0.669 [1.388]	0.809 [1.304]	0.546 [1.546]	0.677 [1.961]	0.570 [2.452]	0.368 [2.476]	0.600 [2.072] 0.606(+1.51%) [1.886](-16.55%)
$\alpha = 0.05$	0.671 [1.135]	0.798 [1.157]	0.543 [1.248]	0.657 [1.858]	0.535 [2.313]	0.346 [2.596]	0.640 [1.976] 0.599(+0.34%) [1.755](-22.35%)
$\alpha = 0.125$	0.662 [1.000]	0.795 [1.000]	0.542 [1.000]	0.564 [1.000]	0.385 [1.000]	0.219 [1.000]	0.400 [1.000] 0.510(-14.57%) [1.000](-55.75%)
$\alpha = 0.25$	0.650 [1.000]	0.797 [1.000]	0.564 [1.000]	0.560 [1.000]	0.380 [1.000]	0.204 [1.000]	0.416 [1.000] 0.510(-14.57%) [1.000](-55.75%)

447

448 Table 4: Performance comparison of all ablation models on Wikipedia-based datasets. We highlight
 449 the best performance among the ablation models.

450

451 Method	452 General QA			453 Multi-Hop QA			454 Avg.
	455 NQ	456 TriviaQA	457 PopQA	458 HotpotQA	459 2wiki	460 Musique	
\Rightarrow AI-SearchPlanner							
$\alpha = 0$	0.674	0.800	0.540	0.678	0.565	0.355	0.568 0.597
w/o Outcome Gain	0.644	0.796	0.557	0.557	0.375	0.200	0.416 0.506(-15.24%)
w/o Process Reward	0.686	0.813	0.557	0.674	0.529	0.327	0.528 0.588(-1.51%)
w/o RL Training	0.616	0.772	0.447	0.623	0.528	0.325	0.520 0.547(-8.38%)

462

463

464 4.4.5 ABLATION STUDY

465

466 In Table 4, we analyze the effects of the dual reward mechanism and whether to train the planner.
 467 For **Dual-Reward Mechanism**, we remove one of them once a time to analyze its contribution.
 468 As shown in Table 4, we can observe that, 1) Removing outcome gain (i.e., $Score(\cdot)$ only) or
 469 process reward in AI-SearchPlanner will bring a decline in performance; 2) Compared to the ablation
 470 baselines, AI-SearchPlanner demonstrates significant performance improvements on complex multi-
 471 hop problems, which proves the effectiveness of dual-reward alignment. For **Trainable vs. Frozen**
 472 **Planner**, we can find that: Compared to the frozen planner, the trainable planner demonstrates
 473 a significant improvement in answer quality, which demonstrates the capability of reinforcement
 474 learning training.

475

476

477 5 CONCLUSION & FUTURE WORK

478

479

480 In this paper, we propose AI-SearchPlanner, a novel reinforcement learning (RL) framework de-
 481 signed to enhance the effectiveness of frozen QA models through optimized search planning. By
 482 decoupling the search planner from the generator, our approach enables a small, trainable LLM to
 483 specialize in search planning while leveraging a large, frozen LLM for high-quality QA — an
 484 architecture that aligns with real-world AI search system constraints. The framework introduces
 485 three key innovations: (1) Decoupling the architecture of the search planner and generator, (2)
 486 Dual-reward alignment to refine search planning capabilities, and (3) Pareto optimization to bal-
 487 ance planning utility and cost. Extensive experiments on real-world datasets demonstrate that AI-
 488 SearchPlanner achieves superior or comparable effectiveness to existing RL-based search agents
 489 while maintaining higher efficiency. Our findings underscore the advantages of separating search
 490 planning from QA generation, enabling more scalable and adaptive search-reasoning systems. Fu-
 491 ture work may explore extending this framework to multi-modal search tasks or integrating dynamic
 492 reward mechanisms for broader generalization. Ultimately, AI-SearchPlanner offers a practical and
 493 efficient solution for enhancing LLM-powered search systems, advancing the development of more
 494 intelligent and resource-efficient AI assistants.

486 REFERENCES
487

488 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Ale-
489 man, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical
490 report. *arXiv preprint arXiv:2303.08774*, 2023.

491 Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li,
492 Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned lan-
493 guage models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

494

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

498 Daya Guo, Dejian Yang, Huawei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
499 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
500 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

501

502 Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop
503 qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*,
504 2020.

505 Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing
506 prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*,
507 2023a.

508

509 Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Ji-
510 awei Han. s3: You don't need that much data to train a search agent via rl. *arXiv preprint
arXiv:2505.14146*, 2025.

512

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

515

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

519

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

522

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

525

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

529

530 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal,
531 Heinrich Kütller, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented gener-
532 ation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:
533 9459–9474, 2020.

534

535 Kuan Li, Zhongwang Zhang, Hufeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baix-
536 uan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web
537 agent. *arXiv preprint arXiv:2507.02592*, 2025a.

538

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

540 Yuchen Li, Hengyi Cai, Rui Kong, Xinran Chen, Jiamin Chen, Jun Yang, Haojie Zhang, Jiayi Li,
 541 Jiayi Wu, Yiqun Chen, et al. Towards ai search paradigm. *arXiv preprint arXiv:2506.17188*,
 542 2025c.

543 Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-
 544 augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods
 545 in Natural Language Processing*, pp. 5303–5315, 2023.

546 Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi.
 547 When not to trust language models: Investigating effectiveness and limitations of parametric and
 548 non-parametric memories. *arXiv preprint arXiv:2212.10511*, 7, 2022.

549 Lang Mei, Siyu Mo, Zhihan Yang, and Chong Chen. A survey of multimodal retrieval-augmented
 550 generation. *arXiv preprint arXiv:2504.08748*, 2025.

551 Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring
 552 and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*,
 553 2022.

554 Chang Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-
 555 Rong Wen. Tool learning with large language models: A survey. *Frontiers of Computer Science*,
 556 19(8):198343, 2025.

557 Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro,
 558 Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can
 559 teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–
 560 68551, 2023.

561 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy
 562 optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

563 Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan
 564 Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, et al. Reasonir: Training retrievers
 565 for reasoning tasks. *arXiv preprint arXiv:2504.20595*, 2025.

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

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

572 Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li,
 573 Liwen Zhang, Xinyu Wang, Yong Jiang, et al. Webshaper: Agentically data synthesizing via
 574 information-seeking formalization. *arXiv preprint arXiv:2507.15061*, 2025.

575 Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving re-
 576 trieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv
 577 preprint arXiv:2212.10509*, 2022a.

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

581 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,
 582 Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural informa-
 583 tion processing systems*, 30, 2017.

584 Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Dixin Jiang, Rangan Ma-
 585 jumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv
 586 preprint arXiv:2212.03533*, 2022.

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

597

598 Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang,
 599 Zekun Xi, Gang Fu, Yong Jiang, et al. Webdancer: Towards autonomous information seeking
 600 agency. *arXiv preprint arXiv:2505.22648*, 2025a.

601 Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang,
 602 Yulan He, Deyu Zhou, Pengjun Xie, et al. Webwalker: Benchmarking llms in web traversals.
 603 *arXiv preprint arXiv:2501.07572*, 2025b.

604

605 Chenghao Xiao, G Thomas Hudson, and Noura Al Moubayed. Rar-b: Reasoning as retrieval bench-
 606 mark. *arXiv preprint arXiv:2404.06347*, 2024.

607

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

611

612 Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan
 613 Cao. React: Synergizing reasoning and acting in language models, 2023. *URL https://arxiv.*
614 org/abs/2210.03629, 2023.

615

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

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648
649

A CASE STUDY

650
651
652
653

We present two case studies to demonstrate the performance of AI-Searchplanner in solving complex problems. In both cases, AI-Searchplanner arrives at the correct solution after multiple (≥ 5) rounds of search and planning. In contrast, non-planning baselines fails to reach a correct solution.

654

<Question>: In what county is the city that shares a border with the capital of the state where Levi Casey was born?
<Ground Truth>: ['Richland County', 'Richland County, South Carolina']

655
656

Qwen3-32B: Direct Inference ✗

657
658
659

<Response>: Grundy County

660
661

Qwen3-32B: Naive RAG ✗

662
663
664
665

<Context>: Doc 1: Idaho Idaho () is a state in the northwestern region of the United States. It borders the state of Montana to the east and northeast, Wyoming to the east, Nevada and Utah to the south, and Washington and Oregon to the west. To the north, it shares a small portion of the Canadian border with the province of British Columbia. With a population of approximately 1.7 million and an area of , Idaho is the 14th largest, the 12th least populous and the 7th least densely populated of the 50 U.S. states. The state's capital and largest city is Boise.
 Doc 2: Lyndon, Kentucky Lyndon is a home rule-class city in Jefferson County, Kentucky, United States. The population was 11,002 at the 2010 census, up from 9,369 at the 2000 census. Incorporated on May 10, 1965, Lyndon became part of the new Louisville Metro government in 2003. It remains an independent city with its own mayor and emergency services and is not counted in Louisville's population although its citizens can vote for the mayor of Louisville and Metro Council members. The area is the home of Oxmoor Farm, the residence of Alexander Scott Bullitt, one of the drafters of Kentucky's first constitution.
 Doc 3: Roman Catholic Diocese of Davenport The Roman Catholic Diocese of Davenport () is a diocese of the Catholic Church for the southeastern quarter of the U.S. state of Iowa. There are within the diocese. The diocese's eastern border is at the Mississippi River; the northern border comprises the counties of Jasper, Poweshiek, Iowa, Johnson, Cedar, and Clinton; the western border is made up of the counties of Jasper, Marion, Monroe, and Appanoose; and the southern border is the Iowa-Missouri border. The current bishop of the diocese is Bishop Thomas Zinkula. It is a suffragan see of the Archdiocese of Dubuque.
<Response>: Jefferson County

666

AI-SearchPlanner ✓

667
668
669

<Planning Trajectory>: To answer the question, I need to find out the following information step by step:
 1. Find out where Levi Casey was born.
 2. Determine the state where Levi Casey was born.
 3. Find out the capital of that state.
 4. Find out the city that shares a border with the capital of that state.
 5. Find out the county where that city is located.
 I will start with finding out where Levi Casey was born.
<tool.call>
 {"name": "search", "arguments": {"query_list": ["where was Levi Casey born"]}}
</tool.call>
<tool.response>
 {"result": "Search query:where was Levi Casey born.
 Doc 1 (Title: \"Levi Casey (politician)\") Levi Casey (politician) General Levi Casey (c. 1752 (February 3, 1807) was a United States Representative from South Carolina. He was born in South Carolina and served in the South Carolina militia during the American Revolutionary War. Son of Abner Casey, he served in the American Revolutionary War along with his step brothers, brother(s) and nephews. One of his stepbrothers, Benjamin Casey was killed in action at Camp Middlebrook, New Jersey. He married Elizabeth Duckett in 1775. He reached the rank of brigadier general in the South Carolina militia. After the war, he served as justice of Newberry County.
 Doc 2 (Title: \"Lyman R. Casey\") a State into the Union, Casey was elected as a Republican to the U.S. Senate and served from November 25, 1889, to March 4, 1893. He was an unsuccessful candidate for renomination in 1892. While in the Senate, he was chairman of the Committee on Railroads (Fifty-second Congress). He moved to New York City. In 1861, Casey married Harriett Mary Platt. One of her siblings was Jane Platt, wife of the drinking straw inventor Marvin C. Stone. Casey returned to Washington, D.C.. On January 25, 1914, Casey died in Washington DC. Casey is buried at Greenmount Cemetery in Baltimore, Maryland.
 Doc 3 (Title: \"Levi Greenwood\") 2012 ! scope="row" style="text-align:center" | 2013 ! scope="row" style="text-align:center" | 2014 ! scope="row" style="text-align:center" | 2015 ! scope="row" style="text-align:center" | 2016 ! scope="row" style="text-align:center" | 2017 ! scope="row" style="text-align:center" | 2018 ! colspan=3| Career ! 135 ! 56 ! 63 ! 1322 ! 1028 ! 2350 ! 444 ! 614 ! 0.4 ! 0.5 ! 9.8 ! 7.6 ! 17.4 ! 3.3 ! 4.6 Levi Greenwood Levi Greenwood (born 19 February 1989) is an Australian rules footballer, currently playing for the Collingwood Football Club in the Australian Football League (AFL). Greenwood grew up in the small coastal town of Port. }
</tool.response>
 Doc 1 states that Levi Casey (the politician) was born in South Carolina. I will proceed with the next step.
<tool.call>
 {"name": "search", "arguments": {"query_list": ["capital of the state where Levi Casey was born"]}}
</tool.call>
<tool.response>
 {"result": "Search query:capital of the state where Levi Casey was born."}

696
697
698
699700
701

702 Doc 1 (Title: "Levi Casey (politician)") Levi Casey (politician) General Levi Casey (c. 1752 (February 3, 1807) was a United States Representative from South Carolina. He was born in South Carolina and served in the South Carolina militia during the American Revolutionary War. Son of Abner Casey, he served in the American Revolutionary War along with his step brothers, brother(s) and nephews. One of his stepbrothers, Benjamin Casey was killed in action at Camp Middlebrook, New Jersey. He married Elizabeth Duckett in 1775. He reached the rank of brigadier general in the South Carolina militia. After the war, he served as justice of Newberry County. Doc 2 (Title: "Zadok Casey") in the Illinois House from 1848 to 1852, serving as speaker in 1852, and in the State Senate from 1860 to 1862. Casey Creek, a tributary of the Big Muddy River, is named in honor of Zadok Casey. Casey Middle School and Casey Avenue in Mount Vernon are also named after him. Zadok Casey was born in Greene County, Georgia. Not much is known about his early life. One story is that, as a young man, he witnessed a murder. Because he did not wish to testify, he fled to the frontier. He died in Caseville, Illinois at age 66,

703 Doc 3 (Title: "Levi Casey (politician)") Court in 1785. Casey was a member of the South Carolina Senate in 1781 and 1782 and 1800(1802 and a member of the South Carolina House of Representatives 1786(1788, 1792(1795 and 1798(1799. He was elected as a Republican to the Eighth and Ninth Congresses and served from March 4, 1803, until his death, before the close of the Ninth Congress. Prior to dying, he had been reelected to the Tenth Congress. He died in Washington, D.C., February 3, 1807 and was buried in the Congressional Cemetery.

704 Levi Casey (politician) General Levi Casey (c. 1752 (February 3, 1807) was a "}

705 </tool_response>

706 Doc 1 confirms that Levi Casey (the politician) was born in South Carolina. Now I will proceed with finding the capital of South Carolina.

707 <tool_call>

708 {"name": "search", "arguments": {"query_list": ["capital of South Carolina"]}}

709 </tool_call>

710 <tool_response>

711 {"result": "Search query:capital of South Carolina.

712 Doc 1 (Title: "South Carolina") 40th most extensive and 23rd most populous U.S. state. Its GDP as of 2013 was \$183.6 billion, with an annual growth rate of 3.13%. South Carolina is composed of 46 counties. The capital is Columbia with a 2016 population of 134,309; while its largest city is Charleston with a 2016 population of 134,385. The Greenville-Anderson-Mauldin metropolitan area is the largest in the state, with a 2016 population estimate of 884,975. South Carolina is named in honor of King Charles I of England, who first formed the English colony, with "Carolus" being Latin for "Charles". South Carolina is known for its.

713 Doc 2 (Title: "Columbia, South Carolina") Columbia, South Carolina Columbia is the capital and second largest city of the U.S. state of South Carolina, with a population estimate of 134,309 as of 2016. The city serves as the county seat of Richland County, and a portion of the city extends into neighboring Lexington County. It is the center of the Columbia metropolitan statistical area, which had a population of 767,598 as of the 2010 United States Census, growing to 817,488 by July 1, 2016, according to 2015 U.S. Census estimates. The name "Columbia" is a poetic term used for the United States, originating from the name.

714 Doc 3 (Title: "South Carolina State House") South Carolina State House The South Carolina State House is the building housing the government of the U.S. state of South Carolina. The building houses the South Carolina General Assembly and the offices of the Governor and Lieutenant Governor of South Carolina. Until 1971, it also housed the Supreme Court. It is located in the capital city of Columbia near the corner of Gervais and Assembly Streets. The State House is in the Greek Revival style; it is approximately tall, long, wide. It weighs more than and has of space. The old State House was constructed between 1786 and 1790."

715 </tool_response>

716 Doc 1 states that the capital of South Carolina is Columbia. Now I will proceed with finding the city that shares a border with the capital of South Carolina.

717 <tool_call>

718 {"name": "search", "arguments": {"query_list": ["city that shares a border with Columbia, South Carolina"]}}

719 </tool_call>

720 <tool_response>

721 {"result": "Search query:city that shares a border with Columbia, South Carolina.

722 Doc 1 (Title: "Columbia, South Carolina") The city of Columbia has 5 sister cities: Columbia, South Carolina Columbia is the capital and second largest city of the U.S. state of South Carolina, with a population estimate of 134,309 as of 2016. The city serves as the county seat of Richland County, and a portion of the city extends into neighboring Lexington County. It is the center of the Columbia metropolitan statistical area, which had a population of 767,598 as of the 2010 United States Census, growing to 817,488 by July 1, 2016, according to 2015 U.S. Census estimates. The name "Columbia" is a poetic term used.

723 Doc 2 (Title: "Columbia, South Carolina") Columbia, South Carolina Columbia is the capital and second largest city of the U.S. state of South Carolina, with a population estimate of 134,309 as of 2016. The city serves as the county seat of Richland County, and a portion of the city extends into neighboring Lexington County. It is the center of the Columbia metropolitan statistical area, which had a population of 767,598 as of the 2010 United States Census, growing to 817,488 by July 1, 2016, according to 2015 U.S. Census estimates. The name "Columbia" is a poetic term used for the United States, originating from the name.

724 Doc 3 (Title: "Columbia, South Carolina") of Christopher Columbus. The city is located approximately northwest of the geographic center of South Carolina, and is the primary city of the Midlands region of the state. It lies at the confluence of the Saluda River and the Broad River, which merge at Columbia to form the Congaree River. Columbia is home to the University of South Carolina, the state's flagship university and the largest in the state, and is also the site of Fort Jackson, the largest United States Army installation for Basic Combat Training. Columbia is also located 20 miles west of the site of McEntire Joint"}

725 </tool_response>

726 Doc 1 states that Columbia, South Carolina, is the capital and serves as the county seat of Richland County. Therefore, the city that shares a border with Columbia, South Carolina, is in Richland County. Now I will proceed with the final step to find out the county where the city that shares a border with the capital of the state where Levi Casey was born is located.

727 <tool_call>

728 {"name": "search", "arguments": {"query_list": ["county that contains Columbia, South Carolina"]}}

729 </tool_call>

730 <tool_response>

731 {"result": "Search query:county that contains Columbia, South Carolina.

732 Doc 1 (Title: "Columbia, South Carolina") Columbia, South Carolina Columbia is the capital and second largest city of the U.S. state of South Carolina, with a population estimate of 134,309 as of 2016. The city serves as the county seat of Richland County, and a portion of the city extends into neighboring Lexington County. It is the center of the Columbia metropolitan statistical area, which had a population of 767,598 as of the 2010 United States Census, growing to 817,488 by July 1, 2016, according to 2015 U.S. Census estimates. The name "Columbia" is a poetic term used for the United States, originating from the name.

733

756 Doc 2 (Title: "Richland County, South Carolina") Richland County, South Carolina Richland County is located
 757 in the U.S. state of South Carolina. As of the 2016 census estimate, the population was 409,549, making
 758 it the second-most populous county in South Carolina, behind only Greenville County. The county seat and
 759 largest city is Columbia, the state capital. The county was founded in 1785. Richland County is part of the
 760 Columbia, SC Metropolitan Statistical Area. In 2010, the center of population of South Carolina was located
 761 in Richland County, in the city of Columbia. Richland County was probably named for its "rich land." The
 762 county was formed in 1785.
 763 Doc 3 (Title: "Columbia, South Carolina") The city of Columbia, South Carolina Columbia is the capital and second largest city of the U.S. state of South Carolina, with a population
 764 estimate of 134,309 as of 2016. The city serves as the county seat of Richland County, and a portion of the
 765 city extends into neighboring Lexington County. It is the center of the Columbia metropolitan statistical
 766 area, which had a population of 767,598 as of the 2010 United States Census, growing to 817,488 by July 1,
 767 2016, according to 2015 U.S. Census estimates. The name "Columbia" is a poetic term used"}
 768 </tool.response>
 769 Doc 1 states that Columbia, South Carolina, is the county seat of Richland County. Now I have enough
 770 information to call answer expert llm to generate the final answer.
 771 <tool.call>
 772 {"name": "callanswer_llm", "arguments": {}}
 773 </tool.call>
 774 <Response>: Richland County

775

776

777 <Question>: Storm of the Century filmed in state A based on the book by the author who wrote a
 778 story featuring the author of The Hope. When do alcohol sales start in state A?
 779 <Ground Truth>: ['Sam']

780 Qwen3-32B: Direct Inference ✗

781 <Response>: 9:00 AM

782 Qwen3-32B: Naive RAG ✗

783 <Context>: Doc 1: Miracle (novel) Miracle is a novel written by Danielle Steel and published by Random House
 784 in June 2005. The book is Steel's sixty-sixth novel. It is New Year's Eve when the storm of the century hits
 785 northern California. In a quiet neighborhood in San Francisco, amid the chaos of fallen trees and damaged
 786 homes, the lives of three strangers are about to collide. For Quinn Thompson, what happens in the storm's
 787 wake will bring down a barrier he has built around himself since his wife's death. For neighbor Maggie
 788 Dartman, it will spark friendship at a time when she
 789 Doc 2: Dry state A dry state is a state in the United States in which the manufacture, distribution,
 790 importation, and sale of alcoholic beverages are prohibited or tightly restricted. While some states,
 791 such as North Dakota, entered the United States as dry states, others went dry after passage of prohibition
 792 legislation. Currently, no state in the United States remains completely dry, but some states contain dry
 793 counties. Prior to the adoption of nationwide prohibition in 1920, state legislatures in the United States
 794 passed local option laws that allowed a county or township to go dry if it chose to do so.
 795 Doc 3: Rick Moody Hiram Frederick ""Rick"" Moody III (born October 18, 1961) is an American novelist and
 796 short story writer best known for the 1994 novel ""The Ice Storm"", a chronicle of the dissolution of two
 797 suburban Connecticut families over Thanksgiving weekend in 1973, which brought him widespread acclaim, became
 798 a bestseller, and was made into the film ""The Ice Storm"". Many of his works have been praised by fellow
 799 writers and critics alike, and in 1999 ""The New Yorker"" chose him as one of America's most talented young
 800 writers, placing him on their ""20 Writers for the 21st Century"" list.
 801 <Response>: The question cannot be answered with the given context.

802 AI-SearchPlanner ✓

803 <Planning Trajectory>: To answer the question, I need to find out the state A where the movie "Storm of the
 804 Century" was filmed, the author of the book it was based on, and the author of "The Hope". Then I will find
 805 out when alcohol sales start in state A. I will start with finding out the state A where the movie "Storm of
 806 the Century" was filmed.
 807 <tool.call>
 808 {"name": "search", "arguments": {"query_list": ["state where the movie Storm of the Century was filmed"]}}
 809 </tool.call>
 810 <tool.response>
 811 {"result": "Search query:state where the movie Storm of the Century was filmed.
 812 Doc 1 (Title: "Storm of the Century") Storm of the Century Storm of the Century, alternatively known as
 813 Stephen King's Storm of the Century, is a 1999 horror TV miniseries written by Stephen King and directed by
 814 Craig R. Baxley. Unlike many other King mini-series, "Storm of the Century" was not based upon a Stephen King
 815 novel|King wrote it as a screenplay from the beginning. The screenplay was published in February 1999. A
 816 very powerful blizzard hits the fictional small town of Little Tall Island (also the setting of King's novel
 817 "Dolores Claiborne") off the coast of Maine. The storm is so powerful that all access off
 818 Doc 2 (Title: "1993 Storm of the Century") north-west Florida early on the morning of March 13, with a
 819 significant storm surge in the northwestern Florida peninsula that drowned several people. Barometric
 820 pressures recorded during the storm were low. Readings of were recorded in Tallahassee, Florida, and even
 821 lower readings of were observed in New England. Low pressure records for March were set in areas of twelve
 822 states along the Eastern Seaboard, with all-time low pressure records set between Tallahassee and Washington,
 823 D.C. Snow began to spread over the eastern United States, and a large squall line moved from the Gulf of
 824 Mexico into Florida and Cuba."}

810 Doc 3 (Title: "1993 Storm of the Century") Heavy snow was first reported in highland areas as far south
 811 as Alabama and northern Georgia, with Union County, Georgia reporting up to of snow in the north Georgia
 812 mountains. Birmingham, Alabama, reported a rare of snow. The Florida Panhandle reported up to of snow,
 813 with hurricane-force wind gusts and record low barometric pressures. Between Louisiana and Cuba, the
 814 hurricane-force winds produced high storm surges across the big bend of Florida which, in combination with
 scattered tornadoes, killed dozens of people. Record cold temperatures were seen across portions of the south
 and east of the US in the wake of"}
 815 </tool.response>
 816 Doc 1 states that "Storm of the Century" is a 1999 horror TV miniseries written by Stephen King and directed
 817 by Craig R. Baxley. It was filmed in Maine, as it is based on a fictional small town of Little Tall Island
 off the coast of Maine. Now I will proceed with finding out the author of the book it was based on.
 818 <tool.call>
 819 {"name": "search", "arguments": {"query.list": ["author of the book Storm of the Century by Stephen
 King"]}}
 820 </tool.call>
 821 <tool.response>
 822 {"result": "Search query:author of the book Storm of the Century by Stephen King.
 Doc 1 (Title: "Storm of the Century") Storm of the Century Storm of the Century, alternatively known as
 823 Stephen King's Storm of the Century, is a 1999 horror TV miniseries written by Stephen King and directed by
 Craig R. Baxley. Unlike many other King mini-series, "Storm of the Century" was not based upon a Stephen King
 novel; King wrote it as a screenplay from the beginning. The screenplay was published in February 1999. A
 824 very powerful blizzard hits the fictional small town of Little Tall Island (also the setting of King's novel
 "Dolores Claiborne") off the coast of Maine. The storm is so powerful that all access off.
 Doc 2 (Title: "Storm of the Century") the format the story should take, either a novel or a screenplay,
 825 ending up writing a "novel for television". "Storm of the Century" aired on ABC on February 14(18, 1999.
 It was released on DVD on June 22 the same year. The miniseries was well received by critics. Review
 826 aggregator Rotten Tomatoes counts six positive out of eight reviews. U.S. Television Ratings: Storm of the
 827 Century Storm of the Century, alternatively known as Stephen King's Storm of the Century, is a 1999 horror TV
 miniseries written by Stephen King and directed by Craig R. Baxley. Unlike many other King mini-series.
 Doc 3 (Title: "Storm of the Century") the wrong decision, "but in daylight, I know better." The screenplay
 828 for the miniseries was written by Stephen King expressly for television, and was published by Pocket Books
 just prior to the initial airing of "Storm of the Century" on ABC, while the mass market edition of the
 829 screenplay was published only as a trade paperback. A hardcover edition was published concurrently by the
 830 Book of the Month Club. The book contains an introduction in which King describes the genesis of the idea as
 it occurred to him in late 1996, beginning to write it in December 1996, and debating"}
 831 </tool.response>
 832 Doc 1 states that "Storm of the Century" is a 1999 horror TV miniseries written by Stephen King. Now I will
 833 proceed with finding out the author of the book "The Hope".
 834 <tool.call>
 835 {"name": "search", "arguments": {"query.list": ["author of the book The Hope"]}}
 836 </tool.call>
 837 <tool.response>
 838 {"result": "Search query:author of the book The Hope.
 Doc 1 (Title: "John Hope Bryant") John Hope Bryant John Hope Bryant (born February 6, 1966) is an American
 839 financial literacy entrepreneur and businessman. Bryant is the founder, chairman and chief executive officer
 840 of nonprofit Operation HOPE, chief executive officer of Bryant Group Ventures and The Promise Homes Company,
 841 co-founder of Global Dignity, advisor to business and government and author of bestselling books on economics
 842 and leadership. He served as a member of the President's Advisory Council on Financial Capability, the vice
 843 chair on the U.S. President's Advisory Council on Financial Literacy and as the chairman of the Committee on
 844 the Underserved. Bryant was born on.
 Doc 2 (Title: "The Life of Hope") The Life of Hope The Life of Hope is a novel by Paul Quarrington, published
 845 in 1985 by Doubleday Canada. It is part of an unofficial trilogy with Quarrington's later novels "King
 846 Leary" and "Logan in Overtime"; although none of the novels centre on the same protagonists, they all feature
 847 some background interrelationships of character and setting. The novel's central character, essentially an
 848 authorial self-insertion, is a novelist named Paul who is suffering from writer's block after the publication
 849 of his baseball-themed novel "Home Game". Attending a writer's retreat in the small Southern Ontario town of
 850 Hope, he learns about.
 Doc 3 (Title: "The Hope (novel)") The Hope (novel) The Hope is a historical novel by Herman Wouk about
 851 pivotal events in the history of the State of Israel from 1948 to 1967. These include Israel's War of
 852 Independence, the 1956 Sinai War (known in Israel as "Operation Kadesh"), and the Six-Day War. The narrative
 853 is continued in the sequel "The Glory". In a loose way it can be considered a sequel to Wouk's Second World
 854 War books - "The Winds of War" and "War and Remembrance". Though the characters are different, the earlier
 855 books gave a major part to the Holocaust and several to their"}
 856 </tool.response>
 857 Doc 1 states that "The Hope" is a book by Herman Wouk. Now I will proceed with finding out the author of the
 858 book "Dolores Claiborne" by Stephen King, as it is mentioned that "Storm of the Century" is based on a novel
 859 by Stephen King.
 860 <tool.call>
 861 {"name": "search", "arguments": {"query.list": ["author of the book Dolores Claiborne by Stephen King"]}}
 862 </tool.call>
 863 <tool.response>
 864 {"result": "Search query:author of the book Dolores Claiborne by Stephen King.
 Doc 1 (Title: "Dolores Claiborne") Dolores Claiborne Dolores Claiborne () is a 1992 psychological thriller
 865 novel by Stephen King. The novel is narrated by the title character. Atypically for a King novel, it has
 866 no chapters, double-spacing between paragraphs, or other section breaks; thus the text is a single continuous
 867 narrative which reads like the transcription of a spoken monologue. It was the best-selling novel of 1992 in
 868 the United States. The book is dedicated to King's mother: "For my mother, Ruth Pillsbury King." Dolores
 869 Claiborne, an opinionated 65-year-old widow living on the tiny Maine community of Little Tall Island, is
 870 suspected of murdering her.
 Doc 2 (Title: "Dolores Claiborne") Dolores, with Jennifer Jason Leigh as her daughter Selena, and Judy
 871 Parfitt as Vera Donovan. "Dolores Claiborne", the operatic adaptation of the novel composed by Tobias Picker
 872 to a libretto by J. D. McClatchy, premiered at San Francisco Opera on September 18, 2013, with Patricia
 873 Racette in the title role. Dolores Claiborne Dolores Claiborne () is a 1992 psychological thriller novel by
 874 Stephen King. The novel is narrated by the title character. Atypically for a King novel, it has no chapters,
 875 double-spacing between paragraphs, or other section breaks; thus the text is a single continuous narrative
 876 which reads like the.
 Doc 3 (Title: "Dolores Claiborne (film)") Dolores Claiborne (film) Dolores Claiborne is a 1995 American
 877 psychological thriller drama film directed by Taylor Hackford and starring Kathy Bates, Jennifer Jason Leigh,
 878 and David Strathairn. It is based on the novel of the same name by Stephen King. The plot focuses on the
 879 strained relationship between a mother and her daughter, largely told through flashbacks, after her daughter
 880 arrives to her remote hometown on a Maine island where her mother has been accused of murdering the elderly
 881 woman for whom she cared. The screenplay for "Dolores Claiborne" was adapted by Tony Gilroy, and the film was
 882 shot in"}
 883 </tool.response>

```

864 Doc 1 states that "Dolores Claiborne" is a 1992 psychological thriller novel by Stephen King. Now I have
865 enough information to proceed with finding out when alcohol sales start in state A, which is Maine.
866 <tool.call>
867 {"name": "search", "arguments": {"query.list": ["when do alcohol sales start in state Maine"]}}
868 </tool.call>
869 <tool.response>
870 {"result": "Search query:when do alcohol sales start in state Maine.
871 Doc 1 (Title: \"Alcohol laws of Maine\") week. On New Year's Day, alcohol may be sold until 2 a.m. It may
872 be consumed in establishments licensed for on-premises consumption until 1:15 a.m., with the exception of New
873 Year's Day when it is 2:15 a.m. Alcohol sales were previously prohibited until 9 a.m. on Sundays. In 2013
874 an exception was made to allow sales starting at 6 a.m. when St. Patrick's Day falls on a Sunday. But in
875 2015 the legislature revised the law again to allow sales on any Sunday starting at 5 a.m., the same as every
876 other day of the week. It is illegal to.
877 Doc 2 (Title: \"Alcohol laws of Maine\") popular spirits in an attempt to prevent the loss of sales to New
878 Hampshire, which has lower prices and no sales tax. The Maine Department of Public Safety, through the Maine
879 State Police's Liquor and Licensing Division, is responsible for licensing the manufacture, importation,
880 storage, transportation and sale of all liquor. They also administer those laws relating to licensing and
881 the collection of taxes on malt liquor and wine. Title 28-A of the Maine Revised Statutes contains the laws
882 and rules regulating liquor. Alcohol may be sold between the hours of 5 a.m. and 1 a.m. each day of the.
883 Doc 3 (Title: \"Alcohol laws of Maine\") it is prohibited to use an image of the Maine State House in liquor
884 advertising. The legal age to purchase and consume alcohol in Maine is 21, which was instituted in 1987.
885 Minors may consume alcohol at home in the presence of the minor's parents. Minors may not transport liquor
886 unless required due to the minor's employment, or at the request of their parent, guardian, or custodian.
887 Taxes on liquor at store level are included within the price and calculated by BABLO in order to raise enough
888 money to cover their expenses and generate money for the State General Fund"}
889 </tool.response>
890 Doc 1 states that alcohol sales in Maine start at 5 a.m. on weekdays and on Sundays. Now I will call answer
891 expert_llm to generate the final answer based on the conversation context.
892 <tool.call>
893 {"name": "call_answer.llm", "arguments": {}}
894 </tool.call>
895 <Response>: 5 a.m.
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

```

918 **B PROMPTS USED IN EXPERIMENT**
919920 **Prompt for Process Reward**
921

923 You are a professional evaluation expert for multi-round search planning in RAG systems. Please provide a comprehensive score for the entire search planning process.

924 **## Input Format**
925 You will receive a complete message conversation flow, including:
926 - User's previous conversation content (optional)
- User question
- Assistant's multi-round search planning (reflected through tool_calls)
- Search results from each round (reflected through tool_returns)
- Final call to call_answer_llm to end the planning process

927 **## Scoring Dimensions and Deduction Standards**
928 **### [Reference Resolution Quality] Key Assessment**
929 - **Severe issues (deduct 3 points)**: Any round of search queries contains unclear references
- ✗ "its price" "how to apply for this policy" "the conditions mentioned above"
- ✗ Search queries in multi-round conversations still contain reference words like "just mentioned", "previous", "this"
- ✗ Second round searching "what are its competitors" without clarifying what "it" refers to
- **Medium issues (deduct 2 points)**: Inconsistent reference resolution across rounds
- **Good performance (no deduction)**: All rounds of search queries can be independently understood and executed

930 **### [Multi-round Search Strategy Coherence] Key Assessment**
931 - **Severe issues (deduct 3 points)**: Multi-round search lacks logical coherence
- ✗ First round search results are sufficient, but still repeatedly searching for the same information
- ✗ Ignoring previous search results, subsequent searches have no correlation
- ✗ Over 50% of queries are highly repetitive across different rounds
932 - **Medium issues (deduct 2 points)**:
- 2-3 rounds of obvious repetitive searching
- Search strategy is too jumpy, lacking progressive logic
- **Minor issues (deduct 1 point)**: Individual rounds slightly repetitive but overall strategy is reasonable
- **Good performance (no deduction)**:
- ✓ Adjusting subsequent query strategies based on search results
- ✓ Different rounds cover different angles or delve into different aspects
- ✓ Reflects information progression or supplementary logic

933 **### [Search Precision]**
934 - **Too broad (deduct 1 points per round)**:
- ✗ "Shenzhen residency" (should add specific conditions/time)
- ✗ "phone recommendations" (should add budget/requirements)
935 - **Too specific (deduct 1 points per round)**:
- ✗ "December 2024 Shenzhen Nanshan District residency policy article 3"
936 **### [Information Utilization and Completion Efficiency]**
937 - **Severe issues (deduct 2 points)**:
- Found key information but did not effectively utilize it in subsequent rounds
- Obviously need supplementary information but did not conduct targeted searches
938 - **Medium issues (deduct 1 point)**:
- Insufficient information utilization, redundant searching exists
- Completion strategy not precise enough
- **Good performance (no deduction)**:
- ✓ Discovering information gaps based on search results and precisely supplementing
- ✓ Effectively utilizing obtained information to guide subsequent searches

939 **### [Coverage Completeness and Termination Timing]**
940 - **Missing core requirements (deduct 2 points)**:
- The entire planning process did not cover user's core requirements
- Far from user's intent
- **Inappropriate termination timing (deduct 1 point)**:
- Terminating prematurely when information is insufficient
- Continuing ineffective searches when information is already sufficient
- **Single perspective (deduct 1 points)**:
- Multi-round searches still only query from a single perspective

941 **### [Timeliness Consideration]**
942 - **Missing time constraints (deduct 0.5 points per round)**:
- For time-sensitive queries, should combine current time and add time constraints
- **Unnecessary time constraints (deduct 0.5 points per round)**:
- Mistakenly adding time constraints for non-time-sensitive requirements

943 **## Scoring Levels**
944 - **5 points**: Excellent performance in all dimensions, efficient and coherent search strategy, sufficient information utilization, no obvious defects
- **4 points**: Overall good, basically reasonable search strategy, good information utilization (total deduction ≤ 1 point)
- **3 points**: Basically usable, search has certain strategic nature but average efficiency (total deduction around 2 points)
- **2 points**: Obvious problems exist, insufficient strategic nature or low efficiency (total deduction around 3 points)
- **1 point**: Seriously non-compliant, lacking effective search strategy (total deduction ≥ 4 points)

945 **## Output Format**
946 **#### [Score]**
947 Strictly follow the deduction standards, focus on the coherence of multi-round search strategies, information utilization efficiency, and entity decomposition completeness. Do not output any explanations and analysis, only output the score according to the format.

948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

972
973**Prompt for Answer Accuracy**

974

You are a seasoned Q&A expert tasked with evaluating the quality of a response based on the user's question, the standard answer, and the provided response.

975

User Question: <question>

976

Standard Answer: <ground_truth>

977

Provided Response: <solution_str>

978

Criteria: Based on the standard answer, you will determine whether the meaning of the provided response aligns with the standard answer. If they are consistent, the response is yes; otherwise, it is no.

979

Please directly answer with "yes" or "no". Do not include any explanation.

980

981

982

C LLM USAGE DISCLOSURE

983

In this work, the LLM is used only for two tasks: (1) generating the final answer based on the search planning trajectory; (2) scoring the consistency between the model-generated answer and the ground-truth answer.

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025