

# PragmaticSearch: Learning to Efficient Retrieval via Advantage Shaping

Anonymous ACL submission

## Abstract

Retrieval-Augmented Generation (RAG) is essential for grounding Large Language Models, yet its deployment is hindered by the high latency and financial costs of frequent retrieval. Existing reinforcement learning approaches primarily maximize answer accuracy, inadvertently encouraging excessive search behavior and ignoring the trade-off between performance and efficiency. To resolve this, we introduce *PragmaticSearch*, a framework that learns efficient retrieval. Unlike standard RL, *PragmaticSearch* employs a Two-Stage Advantage Shaping (TSAS) curriculum that explicitly decouples capability learning from cost calibration. We further introduce a gating mechanism, theoretically grounded in a Bayesian estimate of retrieval necessity, that dynamically neutralizes cost penalties when retrieval provides high utility. Optimized via our stabilized *PS-GRPO* algorithm, this approach prevents policy collapse. Experiments across seven benchmarks show that *PragmaticSearch* reduces retrieval calls by up to 76.2% while maintaining competitive performance.

## 1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding and generation (Achiam et al., 2023). However, their knowledge is static, confined to the data they were trained on, leading to factual inaccuracies (“hallucinations”) and an inability to access real-time information. Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm to mitigate these limitations, dynamically retrieving relevant documents from external knowledge sources to ground the generation process (Fan et al., 2024; Lewis et al., 2020; Guu et al., 2020). This approach has achieved state-of-the-art performance on a wide array of knowledge-intensive tasks.

Despite its success, the practical deployment of RAG is severely constrained by a critical factor:

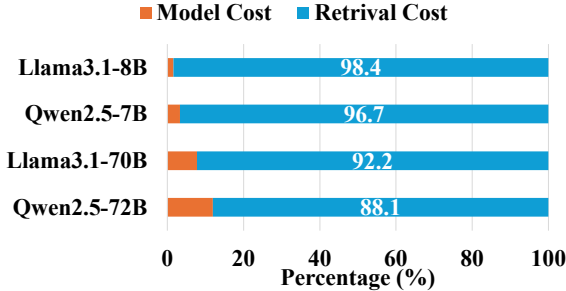


Figure 1: Breakdown of average inference cost per query for RAG systems. Retrieval costs (orange) dominate the cost compared to model inference (blue), highlighting the need for efficient retrieval strategies.

the inference cost of the retrieval step. Each retrieval operation introduces significant latency and incurs direct financial costs when using commercial search APIs (Xu and Peng, 2025) as shown in Figure 1. Beyond the API fees, the retrieval process creates a synchronous bottleneck where the model must pause reasoning to await external feedback. This issue is significantly exacerbated as model size increases and context length grows. In a real-world, high-throughput environment, these costs accumulate rapidly, rendering many sophisticated RAG systems economically and operationally infeasible. The key is not just to make RAG *effective*, but to make it *efficient*.

A promising direction for optimizing this process is to make the retrieval decision adaptive. Reinforcement Learning (RL) presents a natural framework for this, modeling the LLM as an agent that learns a sequential policy on when to search for information versus when to generate an answer directly (Jiang et al., 2023; Asai et al., 2024). However, prevailing RL-based approaches are *myopic* in their formulation. They typically design reward functions that exclusively target the maximization of final answer accuracy. This inadvertently trains the agent to adopt a “search-at-all-costs” strategy, as more evidence often correlates with a higher chance of a correct answer. While this may im-

071 prove task *capability*, it fundamentally fails to ad- 121  
072 dress the underlying efficiency problem. 122

073 In this work, we argue that an effective RAG 123  
074 agent must learn to explicitly navigate the trade- 124  
075 off between task performance and computational 125  
076 cost. To this end, we introduce *PragmaticSearch*, 126  
077 a systematic RL framework designed to train cost- 127  
078 aware RAG agents. A naive solution might involve 128  
079 simply adding a fixed penalty for each search ac- 129  
080 tion to the reward function. However, our initial 130  
081 investigations reveal that this approach is highly 131  
082 unstable, often leading to training collapse or sub- 132  
083 optimal “lazy” policies where the agent learns to 133  
084 avoid searching altogether.

085 To overcome this instability, *PragmaticSearch* 134  
086 employs a carefully designed *two-stage curriculum* 135  
087 with *Two-Stage Advantage Shaping (TSAS)*. The 136  
088 first stage, a *Capability-Building* phase, focuses 137  
089 on developing search competence. Here, the ad- 138  
090 vantage function is shaped to be cost-agnostic, re- 139  
091 warding exploration and reasoning to build a robust 140  
092 understanding of how to effectively use the retrieval 141  
093 tool. The second stage, an *Efficiency-Cultivating* 142  
094 phase, transitions the objective to cost-awareness. 143  
095 It introduces a novel advantage function where a 144  
096 soft performance gate, conditioned on the group- 145  
097 relative quality of the generated answer, dynam- 146  
098 ically modulates the reward. This gate is derived 147  
099 from a Bayesian estimate of retrieval utility and 148  
100 serves to refine the optimization process by neu- 149  
101 tralizing penalties for high-utility searches while 150  
102 simultaneously punishing redundant ones. This 151  
103 entire curriculum is optimized using our *Pragmat-* 152  
104 *icSearch Group-Relative Policy Optimization (PS-* 153  
105 *GRPO)* algorithm, which robustly teaches the agent 154  
106 a judicious policy on when to search. Extensive 155  
107 experiments across diverse single-hop and multi- 156  
108 hop question-answering benchmarks (Kwiatkowski 157  
109 et al., 2019; Joshi et al., 2017; Yang et al., 2018; 158  
110 Ho et al., 2020) demonstrate that this framework 159  
111 achieves significant efficiency gains, reducing re- 160  
112 trieval calls by up to 76.2% with negligible impact 161  
113 on task performance. 162

114 Our contributions are as follows:

- 115 • We identify the critical flaw in existing RL- 160  
116 based RAG frameworks: their myopic focus 161  
117 on accuracy leads to inefficient, costly search 162  
118 policies unsuited for practical deployment. 163
- 119 • We propose a systematic two-stage curriculum 164  
120 learning framework that decouples capability-

building from efficiency-tuning, proving es- 121  
sential for stable training. 122

- We design a novel adaptive policy op- 123  
timization mechanism that synergizes a 124  
performance-gated reward function with a sta- 125  
bilized group-based algorithm, enabling the 126  
model to robustly learn a cost-efficient policy. 127
- We provide comprehensive empirical valida- 128  
tion across seven diverse QA benchmarks, 129  
demonstrating that *PragmaticSearch* drasti- 130  
cally reduces retrieval costs while maintaining 131  
competitive performance. 132

## 2 Methodology 133

### 2.1 Problem Formulation 134

We formulate the task of controlling a RAG 135  
agent as a finite-horizon Markov Decision Process 136  
(MDP), defined by the following components: 137

- **State ( $s_t$ ):** The state at timestep  $t$  is a tuple 138  
 $s_t = (\mathbf{q}, h_{t-1})$ , where  $\mathbf{q}$  is the initial user 139  
query and  $h_{t-1}$  is the history of past actions 140  
and their corresponding observations. 141
- **Action ( $a_t$ ):** At each step, the agent 142  
chooses an action from a discrete set  $\mathcal{A} =$  143  
 $\{\text{SEARCH}(\text{QUERY}), \text{GENERATE}(\text{ANSWER})\}$ . 144
- **Policy ( $\pi_\theta$ ):** The agent’s policy  $\pi_\theta(a_t|s_t)$  is 145  
represented by an LLM with parameters  $\theta$ . 146
- **Trajectory ( $\tau$ ):** A trajectory is a full 147  
sequence of states and actions,  $\tau =$  148  
 $(s_0, a_0, \dots, s_T, a_T)$ , generated by the policy. 149
- **Reward and Advantage:** The policy  $\pi_\theta$  150  
aims to maximize the expected reward 151  
 $\mathbb{E}_\tau \sim \pi_\theta[R(\tau)]$ . To mitigate the high vari- 152  
ance of raw rewards  $R(\tau)$ , we utilize the ad- 153  
vantage  $A(\tau)$ , which measures performance 154  
relative to a baseline. Our core contribution 155  
involves shaping this advantage  $A(\tau)$  through 156  
a curriculum to stabilize training. 157

A central challenge in training cost-aware agents 158  
is the cold start policy collapse problem. A naive 159  
approach might use a single, composite objective 160  
from the outset, aiming to maximize a reward like 161  
 $R(\tau) - \lambda \cdot C(\tau)$ , where  $C(\tau)$  is the trajectory cost. 162  
However, for a weakly initialized agent that has yet 163  
to acquire effective search skills, simultaneously 164  
maximizing  $R(\tau)$  and minimizing  $C(\tau)$  is noisy 165

and challenging; reward hacking often emerges, the gradient is soon dominated by the cost penalty, and the agent collapses into a “lazy” policy that trivially maximizes its objective. This leads to a suboptimal local minimum where the agent never acquires task-solving skills. To circumvent this, we propose a *two-stage curriculum* that decouples capability acquisition from efficiency cultivation.

### 2.1.1 Stage 1: Capability-Building (Warm-up)

The exclusive objective of this stage is to develop a competent agent. Conceptually, we aim to find a “capable policy,”  $\pi_{\text{capable}}$ , that maximizes a composite reward encouraging both task success and exploration. This can be formally described as:

$$\pi_{\text{capable}} = \arg \max_{\pi_{\theta}} \mathbb{E}_{\tau \sim \pi_{\theta}} [R_1(\tau)] \quad (1)$$

where  $R_1(\tau) = R(\tau) + R_{\text{int}}(\tau)$ . Here,  $R(\tau)$  is the external task reward, and  $R_{\text{int}}(\tau)$  is an intrinsic reward function that encourages exploratory behaviors causally linked to success, such as using tools and generating detailed reasoning. It is a function of the search count  $S(\tau)$  and reasoning length  $T_{\text{len}}(\tau)$ .

### 2.1.2 Stage 2: Efficiency-Cultivating (Annealing)

This stage commences after the agent has developed a baseline competence. The objective shifts to finding an “efficient policy,”  $\pi_{\text{efficient}}$ , that refines the strategies learned in Stage 1. This policy should maximize task reward while penalizing cost *only when it is not justified by performance*. We formalize this as:

$$\pi_{\text{efficient}} = \arg \max_{\pi_{\theta}} \mathbb{E}_{\tau \sim \pi_{\theta}} [R_2(\tau)] \quad (2)$$

where  $R_2(\tau) = R(\tau) - g(R(\tau)) \cdot C(\tau)$ . In this formulation,  $C(\tau)$  is the trajectory cost (a function of  $S(\tau)$  and  $T_{\text{len}}(\tau)$ ), and  $g(R(\tau))$  is a crucial *performance gating function*. This gate is designed to be near-zero for failing trajectories (where  $R(\tau)$  is low), thus ignoring their cost, but becomes positive for successful trajectories, creating pressure to find more cost-effective solutions.

The transition between these two stages is triggered after a fixed number of training iterations,  $M_1$ . This curriculum design directly motivates our specific advantage shaping mechanism.

## 2.2 Two-Stage Advantage Shaping (TSAS)

TSAS implements the conceptual goals of our curriculum by defining two different advantage functions,  $\hat{A}_i$ , one for each stage. Let’s denote the  $i$ -th trajectory in a batch of size  $G$  as  $(\mathbf{q}, \mathbf{o}_i)$ .

**Stage 1: Capability-Building Advantage.** In the first stage, the advantage function  $\hat{A}_{i,t}$  for trajectory  $i$  at timestep  $t$  is defined as:

$$\begin{aligned} \hat{A}_{i,t} \equiv & R(\mathbf{q}, \mathbf{o}_i) - \text{mean}\{R(\mathbf{q}, \mathbf{o}_j)\}_{j=1}^G \\ & + \log(S(\mathbf{q}, \mathbf{o}_i) + 1) + \alpha \log(T_{\text{len}}(\mathbf{q}, \mathbf{o}_i)) \end{aligned} \quad (3)$$

This function combines the group-normalized task reward with intrinsic rewards for search ( $S$ ) and reasoning ( $T_{\text{len}}$ ), directly reflecting the goal of the capability-building stage. Specifically, this advantage function combines the group-relative reward  $R(\mathbf{q}, \mathbf{o}_i)$  with auxiliary rewards:  $S(\mathbf{q}, \mathbf{o}_i)$  counts the number of search operations performed, and  $T_{\text{len}}(\mathbf{q}, \mathbf{o}_i)$  measures the total length of reasoning traces. The hyperparameter  $\alpha$  controls the relative importance of reasoning depth, while  $\epsilon$  defines the clipping range for stable optimization.

**Stage 2: Efficiency-Cultivating Advantage.** In the second stage, the advantage function is reformulated to implement the performance-gated efficiency objective:

$$\hat{A}_i \equiv A_i^+ + \sigma_i \cdot S_i^+ + \sigma_i \cdot T_i^+ \quad (4)$$

The components are defined as follows:

- $A_i^+$ : The group-normalized task reward, defined as  $R(\mathbf{q}, \mathbf{o}_i) - \text{mean} R(\mathbf{q}, \mathbf{o}_j)_{j=1}^G$ .
- $S_i^+$ : A normalized reward for searching, calculated as the logarithm of the search count minus the batch’s standard deviation of search counts:  $\log(S(\mathbf{q}, \mathbf{o}_i) + 1) - \text{std}$ .
- $T_i^+$ : A reward for reasoning length, given by the logarithm of the trajectory’s token length:  $\log(T_{\text{len}}(\mathbf{q}, \mathbf{o}_i) + 1)$ .
- $\sigma_i$ : The critical *soft performance gate*,  $\text{sigmoid}(A_i^+) - 0.5$ . When a trajectory is successful ( $A_i^+ > 0$ ),  $\sigma_i$  is positive, preserving the intrinsic rewards from  $S^+i$  and  $T_i^+$ . When it fails ( $A_i^+ < 0$ ),  $\sigma_i$  becomes negative, effectively penalizing costly exploration that did not lead to success.

## 2.3 Theoretical Justification

The design of the soft performance gate  $\sigma_i$  addresses the credit assignment problem: was a retrieval action necessary? We model the utility of retrieval as a latent variable estimation problem.

Let  $Z \in \{0, 1\}$  be a latent variable where  $Z = 1$  denotes a *necessary* retrieval and  $Z = 0$  denotes a *redundant* one. We assume the observed performance advantage  $A$  follows a Gaussian distribution conditioned on utility:

$$P(A|Z = z) = \begin{cases} \mathcal{N}(A; \mu_{\text{pos}}, \tau^2) & \text{if } z = 1 \\ \mathcal{N}(A; \mu_{\text{neg}}, \tau^2) & \text{if } z = 0 \end{cases} \quad (5)$$

Assuming a uniform prior  $P(Z = 1) = 0.5$ , the posterior probability that a retrieval was useful is given by Bayes' rule:

$$P(Z = 1|A) = \frac{1}{1 + \exp\left(-\frac{\mu_{\text{pos}} - \mu_{\text{neg}}}{\tau^2}(A - c)\right)} \quad (6)$$

where  $c$  is the decision boundary. This simplifies to a logistic function:  $P(Z = 1|A) = \text{sigmoid}(k(A - c))$ .

In *PragmaticSearch*, we translate this probability into value shaping. We center the posterior to range  $[-0.5, 0.5]$  as:

$$\text{Gate}(A_i) = P(Z = 1|A_i) - 0.5 \quad (7)$$

This theoretical derivation justifies Eq. (4): the cost penalty is dynamically modulated by the probability that the search actually contributed to the model's success.

## 2.4 PS-GRPO

To optimize the policy  $\pi_\theta$  using the advantages defined by TSAS, we employ a policy gradient algorithm based on GRPO (Shao et al., 2024), which we refer to as *PragmaticSearch* Group-Relative Policy Optimization (PS-GRPO). The objective function for Stage 1 is:

$$\begin{aligned} \mathcal{J}_{\text{LS-GRPO}}(\pi_\theta) = & \mathbb{E}_{\mathbf{q} \sim p_{\mathcal{Q}}, \{\mathbf{o}_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|\mathbf{q})} \frac{1}{G} \\ & \sum_{i=1}^G \sum_{t=1}^{|\mathbf{o}_i|} \left\{ \min\left[\rho_{i,t} \hat{A}_{i,t}, \text{clip}(\rho_{i,t}, 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t}\right] \right\}, \end{aligned} \quad (8)$$

where  $\mathbf{q} \sim p_{\mathcal{Q}}$  represents queries sampled from the task distribution,  $\{\mathbf{o}_i\}_{i=1}^G$  denotes a group of  $G$  responses generated by the reference policy  $\pi_{\theta_{\text{old}}}$ ,

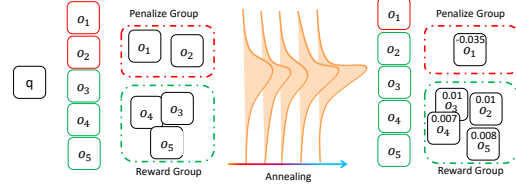


Figure 2: Evolution of the Group Relative Score distribution for  $A_i^+$ .  $q$  denotes the query;  $o_1 \sim o_5$  are the model-sampled group responses.

$\hat{A}_{i,t}$  is the advantage from Eq. 3. The objective for Stage 2 uses the same structure, but substitutes  $\hat{A}_{i,t}$  with the annealing advantage  $\hat{A}_i$  from Eq. 4. and  $\rho_{i,t}$  is the importance sampling ratio between the current and reference policies.

$$\rho_{i,t} = \frac{\pi_\theta(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|\mathbf{q}, \mathbf{o}_{i,<t})} \quad (9)$$

The clipping term, controlled by  $\epsilon$ , ensures stable training. This combination of a clear curriculum, precisely shaped advantages, and a stable optimization algorithm is key to our framework.

## 2.5 Mechanism Analysis

The effectiveness of TSAS lies in the dynamic transition of optimization focus driven by the evolution of the reward distribution. In the early warm-up stage (Stage 1), the broad score distribution allows the group-relative advantage  $A_i^+$  to dominate the gradient, prioritizing the acquisition of basic reasoning and search skills as defined in Eq. (8). As the policy matures and the reward distribution becomes more concentrated, the magnitude of  $A_i^+$  naturally diminishes (Figure 2). This concentration causes the gradient dominance to smoothly shift toward the efficiency-oriented terms  $\sigma_i \cdot S_i^+$  and  $\sigma_i \cdot T_i^+$  in the Eq. (4). This data-driven transition avoids the "goal interference" common in multi-objective optimization. Furthermore, as the distribution develops a "low-score long-tail" as training matures (see Figure 2), the soft performance gate  $\sigma_i$  (defined as  $\text{sigmoid}(A_i^+) - 0.5$  in Eq. (4)) leverages the non-linear nature of RL gradients. In this late phase, trajectories with redundant search yield large negative advantages, exerting a disproportionately strong corrective force that effectively prunes unnecessary search and reasoning steps.

## 3 Experiment

### 3.1 Experiment Setup

Our framework is built upon the *verl* (Sheng et al., 2024), which is optimized for distributed reinforce-

Method	Single-Hop QA									Averages	
	NQ			TriviaQA			PopQA			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
<b><i>Qwen-2.5-7B (Base)</i></b>											
Direct Answer	12.40	-	-	21.80	-	-	7.20	-	-	13.80	-
CoT	21.40	-	-	34.60	-	-	13.00	-	-	23.00	-
RAG	20.60	<u>0.68</u>	0.81	31.60	<u>0.67</u>	0.79	22.20	<u>0.52</u>	0.67	24.80	<u>0.62</u>
Search-o1	20.40	1.62	1.89	41.00	2.03	2.72	10.00	1.78	1.60	23.80	1.81
Search-R1	42.20	1.22	<u>0.46</u>	<u>62.20</u>	1.32	<u>0.59</u>	49.20	1.15	<u>0.37</u>	51.20	1.23
ZeroSearch	<u>43.40</u>	1.12	0.66	61.40	1.13	0.65	<u>57.00</u>	1.18	0.64	<u>53.93</u>	1.15
<i>PragmaticSearch</i>	<b>49.60</b>	<b>0.53</b>	<b>0.23</b>	<b>71.60</b>	<b>0.47</b>	<b>0.52</b>	<b>63.40</b>	<b>0.05</b>	<b>0.23</b>	<b>61.53</b>	<b>0.35</b>
<b><i>Qwen-2.5-3B (Base)</i></b>											
Direct Answer	7.00	-	-	14.40	-	-	4.00	-	-	8.47	-
CoT	9.00	-	-	13.60	-	-	6.00	-	-	9.53	-
RAG	10.40	<u>0.50</u>	0.63	16.20	0.57	<u>0.69</u>	11.40	0.59	0.67	12.67	0.55
Search-o1	16.40	0.75	0.81	31.00	0.88	0.99	8.40	1.01	0.98	18.60	0.97
Search-R1	40.40	1.28	0.72	59.40	0.94	0.78	41.40	1.23	<u>0.46</u>	47.07	1.15
ZeroSearch	<u>44.60</u>	<u>0.50</u>	<u>0.50</u>	<u>64.60</u>	<u>0.21</u>	<u>0.41</u>	<u>64.60</u>	<u>0.30</u>	<u>0.46</u>	<u>57.93</u>	<u>0.34</u>
<i>PragmaticSearch</i>	<b>48.00</b>	<b>0.23</b>	<b>0.43</b>	<b>65.80</b>	<b>0.02</b>	<b>0.15</b>	<b>66.20</b>	<b>0.00</b>	<b>0.00</b>	<b>60.00</b>	<b>0.08</b>
<b><i>LLaMA-3.2-3B (Base)</i></b>											
Direct Answer	7.40	-	-	11.40	-	-	6.80	-	-	8.53	-
CoT	10.40	-	-	24.40	-	-	0.80	-	-	11.87	-
RAG	28.80	0.78	0.74	45.60	<u>0.68</u>	0.68	35.80	<u>0.62</u>	0.54	36.73	0.69
Search-o1	26.00	1.39	1.04	48.40	0.79	0.64	8.80	0.75	0.70	27.73	1.01
Search-R1	39.80	1.33	0.82	52.10	0.91	0.89	46.00	1.20	<b>0.39</b>	45.97	1.15
ZeroSearch	<u>44.60</u>	<u>0.50</u>	<u>0.50</u>	<u>64.60</u>	<b>0.21</b>	<u>0.41</u>	<b>64.60</b>	<b>0.30</b>	<u>0.46</u>	<u>57.93</u>	<b>0.34</b>
<i>PragmaticSearch</i>	<b>46.40</b>	<b>0.10</b>	<b>0.13</b>	<b>68.60</b>	1.00	<b>0.00</b>	<u>63.40</u>	1.01	0.77	<b>59.47</b>	0.70

Table 1: Main results for Single-Hop QA tasks using Base LLMs as the backbone. The best performance is set in bold. Acc: Accuracy (%), ST: Search Times, SD: Search Standard Deviation.

ment learning with large models. All experiments are conducted on a single node equipped with 8 NVIDIA H800 80GB GPUs. For complete hyperparameter configurations and other implementation details, please refer to appendix C.

**Datasets and Evaluation.** Following the setting of ZeroSearch (Sun et al., 2025a), we use the ZeroSearch dataset for training our models. For evaluation, we assess performance on a diverse suite of seven question-answering benchmarks, which are divided into two categories: single-hop and multi-hop. A total of seven datasets are used (Kwiatkowski et al., 2019; Joshi et al., 2017; Mallen et al., 2022; Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022; Press et al., 2022); the details are reported in the appendix D.1. This allows us to measure both in-domain and out-of-domain generalization. Across all benchmarks, the F1 score is used as the performance reward for each answer during training. At the evaluation stage, Exact Match (EM) is used as the primary evaluation metric. To assess search cost and the stability of search behavior, we introduce two additional metrics: the number and standard deviation of searches (ST and SD).

**Models and Baselines.** To evaluate robustness and generalizability, we utilize *Qwen2.5 (3B, 7B)* (Qwen et al., 2025) and *LLaMA-3.2 (3B)* (Dubey et al., 2024) as backbones. We compare *PragmaticSearch* against several categories of baselines: (1) **Direct Answer** and **CoT** (Wei et al., 2022) for closed-book reasoning; (2) **Standard RAG** (Lewis et al., 2020) and **Search-o1** (Li et al., 2025a), an agentic framework that performs on-demand knowledge supplementation; (3) **Search-R1** (Jin et al., 2025b), an end-to-end RL framework that interleaves search into reasoning trajectories; and (4) **ZeroSearch** (Sun et al., 2025a), a curriculum RL framework integrated with external search to facilitate policy learning. For a comprehensive evaluation, we conduct experiments across both Base and Instruct variants for all models and baselines. This allows us to assess the effectiveness of *PragmaticSearch* and the comparative methods across different initialization states and varying levels of prior instruction alignment.

### 3.2 Main Results

The results in Table 1 and Table 2 show that *PragmaticSearch* establishes a more favorable performance-efficiency frontier. Specifically, it

Method	Multi-Hop QA												Averages	
	HotpotQA			2Wiki			Musique			Bamboogle			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
<i>Qwen-2.5-7B (Base)</i>														
Direct Answer	11.40	-	-	14.20	-	-	2.60	-	-	6.94	-	-	8.79	-
CoT	14.80	-	-	21.40	-	-	6.80	-	-	13.89	-	-	14.22	-
RAG	23.00	<u>0.76</u>	0.88	19.40	<u>0.92</u>	0.96	8.00	<u>0.85</u>	0.91	18.06	0.81	0.95	17.12	<u>0.84</u>
Search-o1	16.20	3.37	4.03	28.60	3.97	3.93	10.00	3.79	4.28	28.80	3.40	3.65	20.90	3.63
Search-R1	30.20	2.01	0.83	38.40	2.31	<u>0.90</u>	18.80	2.09	<u>0.74</u>	35.20	1.91	<u>0.63</u>	30.65	2.08
ZeroSearch	<u>35.20</u>	1.43	<u>0.83</u>	<u>38.40</u>	1.58	0.97	<u>20.80</u>	1.49	0.95	<b>44.00</b>	<u>0.74</u>	0.89	<u>34.60</u>	1.31
<i>PragmaticSearch</i>	<b>43.40</b>	<b>0.24</b>	<b>0.71</b>	<b>47.40</b>	<b>0.57</b>	<b>0.82</b>	<b>26.60</b>	<b>0.13</b>	<b>0.37</b>	<b>44.00</b>	<b>0.57</b>	<b>0.58</b>	<b>40.35</b>	<b>0.38</b>
<i>Qwen-2.5-3B (Base)</i>														
Direct Answer	7.40	-	-	8.40	-	-	0.80	-	-	4.17	-	-	5.19	-
CoT	6.40	-	-	9.40	-	-	1.00	-	-	2.78	-	-	4.90	-
RAG	7.80	0.68	0.74	9.80	<u>0.68</u>	0.78	1.20	<u>0.65</u>	0.78	5.56	<u>0.64</u>	<u>0.63</u>	6.09	0.66
Search-o1	14.60	0.91	0.63	22.40	1.02	0.70	5.20	0.95	0.87	<b>22.40</b>	0.96	0.84	16.15	0.96
Search-R1	29.80	0.97	0.76	31.20	1.37	0.94	12.00	0.92	0.77	12.80	0.99	0.68	21.45	1.06
ZeroSearch	<b>37.80</b>	<b>0.26</b>	<b>0.45</b>	<u>34.20</u>	0.93	<u>0.36</u>	<u>18.20</u>	0.89	<b>0.32</b>	<u>22.22</u>	<b>0.00</b>	<b>0.00</b>	<b>28.11</b>	<u>0.52</u>
<i>PragmaticSearch</i>	<u>37.40</u>	<u>0.51</u>	<u>0.50</u>	<b>37.20</b>	<b>0.00</b>	<b>0.00</b>	<b>18.40</b>	<b>0.26</b>	<u>0.44</u>	16.00	<b>0.00</b>	<b>0.00</b>	<u>27.25</u>	<b>0.19</b>
<i>LLaMA-3.2-3B (Base)</i>														
Direct Answer	8.90	-	-	6.60	-	-	4.00	-	-	6.94	-	-	6.61	-
CoT	9.00	-	-	8.20	-	-	4.80	-	-	17.60	-	-	9.90	-
RAG	12.40	<u>0.60</u>	0.86	9.60	<u>0.77</u>	<u>0.83</u>	6.40	<u>0.83</u>	0.77	16.40	0.83	<u>0.85</u>	11.20	<u>0.76</u>
Search-o1	19.40	1.30	0.87	17.40	1.53	0.99	6.00	1.24	0.91	<b>32.00</b>	1.45	0.81	18.70	1.38
Search-R1	<u>29.80</u>	1.01	<u>0.83</u>	<u>30.80</u>	1.52	1.12	<u>13.40</u>	0.85	<u>0.77</u>	18.44	0.91	0.89	23.11	1.07
ZeroSearch	27.80	<b>0.58</b>	0.96	27.50	<b>0.63</b>	0.87	12.00	<b>0.66</b>	0.95	<b>32.20</b>	<u>0.64</u>	0.92	<u>24.88</u>	<b>0.63</b>
<i>PragmaticSearch</i>	<b>35.00</b>	1.00	<b>0.06</b>	<b>37.40</b>	1.01	<b>0.13</b>	<b>19.40</b>	1.00	<b>0.04</b>	16.00	<b>0.58</b>	<b>0.49</b>	<b>26.95</b>	0.90

Table 2: Main results for Multi-Hop QA tasks using Base LLMs as the backbone. The best performance is set in bold. Acc: Accuracy (%), ST: Search Times, SD: Search Standard Deviation.

Method	TriviaQA			HotpotQA			Musique			Average	
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD	Avg. Acc	Avg. ST
<i>PragmaticSearch (Full)</i>	<b>68.60</b>	<b>1.00</b>	<b>0.00</b>	<b>35.00</b>	<b>1.00</b>	<b>0.06</b>	<b>19.40</b>	<b>1.00</b>	<b>0.04</b>	<b>41.00</b>	<b>1.00</b>
w/o Stage 2 (Annealing)	52.20	1.38	0.74	23.80	1.48	0.83	8.40	1.59	0.83	28.13	1.48
w/o Stage 1 (Warm-up)	61.80	1.22	0.51	30.60	1.28	0.53	12.60	1.37	0.59	35.00	1.29

Table 3: Ablation study on *PS-GRPO* using LLaMA-3.2-3B-Base. We evaluate the impact of removing key training stages: the annealing stage (Stage 2) and the warm-up stage (Stage 1). The results demonstrate that both stages are crucial for achieving optimal performance and search efficiency. Acc: Accuracy (%), ST: Average Search Times per query, SD: Search Standard Deviation.

shows superior performance-cost trade-off, its generalizability, and its enhanced policy stability.

***PragmaticSearch* Establishing a Superior Performance-Efficiency Frontier.** *PragmaticSearch* establishes a superior trade-off between performance and efficiency across both single- and multi-hop tasks. This is demonstrated on Qwen models, where *PragmaticSearch* often improves accuracy while simultaneously reducing search cost; e.g., with Qwen-2.5-7B on single-hop tasks, it achieves higher accuracy (61.53% vs. 53.93%) with fewer searches (0.352 vs. 1.147). The results with Llama-3.2-3B highlight a more nuanced policy, where a marginal increase in search cost is traded for substantial accuracy gains (+12.34

points on single-hop, +8.08 on multi-hop). This indicates that our curriculum fosters a policy that optimizes for the marginal utility of each search, making strategic investments for disproportionate performance returns rather than defaulting to a naive cost-minimization strategy.

***PragmaticSearch* Generalizability Across Diverse Models and Task Complexities.** The advantages of *PragmaticSearch* generalize across model architectures and scales. With Qwen-2.5-7B on single-hop tasks, *PragmaticSearch* achieves 61.53% accuracy with 0.352 searches, outperforming ZeroSearch, which scores 53.93% with a slightly higher cost of 1.147. This demonstrates an instance of achieving higher accuracy with

Method	Tokens		Time (ms)	
	Base	Instruct	Base	Instruct
<b>Qwen-2.5-3B</b>				
<i>PragmaticSearch</i>	293.96	356.78	1740.42	2114.64
ZeroSearch	249.75	345.33	1479.06	2049.30
Search-o1	683.22	694.91	4052.83	4129.95
Search-R1	754.39	952.58	4475.00	5661.28
<b>Qwen-2.5-7B</b>				
<i>PragmaticSearch</i>	489.65	240.60	3319.58	1629.24
ZeroSearch	473.74	287.32	3210.96	1948.30
Search-o1	929.31	162.66	3515.88	2098.95
Search-R1	264.85	349.50	1808.54	2358.57

Table 4: Average response length and inference time comparison between Ours and ZeroSearch.

lower computational overhead. The trend holds for the Qwen-2.5-3B model, where *PragmaticSearch* maintains a performance lead on single-hop tasks and is competitive on multi-hop tasks while reducing search frequency by over 76% (0.08 vs. 0.337 Avg ST). This consistent behavior across different models and task complexities validates the robustness of our training framework.

***PragmaticSearch* Enhanced Policy Stability and Operational Reliability.** Beyond aggregate efficiency, *PragmaticSearch* induces a more stable and reliable policy, which manifests in two ways. First, it exhibits lower variance in its search behavior. *PragmaticSearch* consistently reduces the search standard deviation; for Llama-3.2-3B, the SD is reduced to 0.00 from ZeroSearch’s 0.63. This indicates a more predictable agent that applies a consistent strategy to similar problems.

Meanwhile, *PragmaticSearch* demonstrates superior operational reliability by mitigating the *format collapse* issue observed in the baseline, where the agent’s outputs progressively degrade and fail to adhere to the required action format. Our *format reward design* directly addresses this by explicitly rewarding correctly formatted actions. As illustrated in Figure 3, this design leads to a more stable, valid action ratio and smaller fluctuations in response length throughout training.

The Instruct models yield similar results, which can be found in Appendix E.

### 3.3 Ablation Study

To investigate the necessity of our two-stage curriculum, we conducted an ablation study with the results presented in Table 3. The experiments confirm that both stages are indispensable for achieving optimal results. Removing the final *annealing stage* cripples accuracy (e.g., from 35.87 to 28.13)

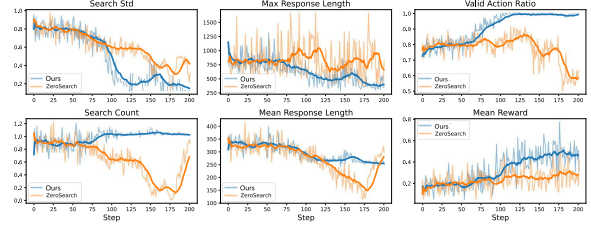


Figure 3: RL fine-tuning dynamics of *PragmaticSearch* and *ZeroSearch*. Solid lines are moving averages over 15 steps.

and causes erratic, excessive searches. This occurs because the agent learns *how* to search but is never taught *when* to do so efficiently, as it is never exposed to a cost-aware objective. Conversely, omitting the initial *warm-up stage* also degrades performance by increasing search cost and variance. The premature cost penalty stifles exploration, preventing the agent from developing a robust base policy for subsequent optimization. Ultimately, the results demonstrate that the two stages are complementary: the warm-up is essential for building a capable foundation, while the annealing stage is critical for refining it into a cost-efficient policy.

### 3.4 Complementary Study

To investigate the generalizability and modularity of our two-stage curriculum, we conducted a complementary study. In this experiment, we integrated our Stage 1 (Warm-up) into the existing *ZeroSearch* framework, denoted as *ZeroSearch*<sup>+</sup>. The results, presented in Table 5 and Table 6, show a significant performance improvement. *ZeroSearch*<sup>+</sup> consistently outperforms the original *ZeroSearch* across all tested single-hop and multi-hop datasets, with the average accuracy increasing from 50.33 to 61.13 on single-hop tasks and from 18.13 to 23.60 on multi-hop tasks. This demonstrates that our warm-up strategy is not only effective within our own framework but can also serve as a transferable module to enhance other RL-based methods. Yet, lacking an explicit annealing phase, *ZeroSearch*<sup>+</sup> remains sub-optimal in search budget and stability.

### 3.5 Case Study

To qualitatively analyze the model’s behavior, we visualize the search trajectories in Table 9 in the Appendix F. The results show that *PragmaticSearch* successfully balances between utilizing internal knowledge and executing complex, multi-step search strategies.

Method	Single-Hop QA									Averages	
	NQ			TriviaQA			PopQA			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
<i>LLaMA-3.2-3B-Instruct</i>											
ZeroSearch	38.00	<b>1.04</b>	0.72	54.40	<b>0.91</b>	0.74	57.40	<b>0.77</b>	0.57	49.93	<b>0.91</b>
<i>ZeroSearch</i> <sup>+</sup>	<b>47.20</b>	1.07	<b>0.33</b>	<b>63.80</b>	1.04	<b>0.29</b>	<b>72.40</b>	1.00	<b>0.17</b>	<b>61.13</b>	1.04

Table 5: Supplemental study (Single-Hop QA) comparing our method with the original ZeroSearch. The best performance is set in bold. Acc: Accuracy (%), ST: Search Times, SD: Search Standard Deviation.

Method	Multi-Hop QA												Averages	
	HotpotQA			2Wiki			Musique			Bamboogle			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
<i>LLaMA-3.2-3B-Instruct</i>														
ZeroSearch	23.00	<b>1.06</b>	0.83	22.00	1.31	0.95	10.60	1.18	0.84	28.80	<b>0.98</b>	0.78	21.10	1.13
<i>ZeroSearch</i> <sup>+</sup>	<b>29.40</b>	1.11	<b>0.41</b>	<b>23.80</b>	<b>1.19</b>	<b>0.50</b>	<b>14.80</b>	<b>1.10</b>	<b>0.38</b>	<b>26.40</b>	1.06	<b>0.26</b>	<b>23.60</b>	<b>1.12</b>

Table 6: Supplemental study (Multi-Hop QA) comparing our method with the original ZeroSearch. The best performance is set in bold. Acc: Accuracy (%), ST: Search Times, SD: Search Standard Deviation.

## 4 Related Work

Retrieval-Augmented Generation (RAG) integrates external knowledge to mitigate LLM hallucinations and provide real-time information (Lewis et al., 2020; Guu et al., 2020). While early systems utilized static "retrieve-then-read" pipelines (Izacard and Grave, 2021), the necessity of iterative search for complex queries (Mallen et al., 2022) catalyzed a shift toward adaptive retrieval. Initial adaptive strategies leveraged prompt engineering and Chain-of-Thought (Wei et al., 2022) to manually structure search trajectories (Press et al., 2022; Yoran et al., 2023). Subsequent learned approaches, such as Self-RAG (Asai et al., 2024), employed supervised fine-tuning (SFT) and reflection tokens to automate retrieval decisions. Other advancements include retriever-LLM co-optimization (Shi et al., 2023) and generation-based retrieval (Yu et al., 2022). However, these methods remain constrained by their reliance on expensive, expert-annotated demonstration data. In contrast, our framework employs reinforcement learning to autonomously learn a dynamic retrieval policy, eliminating the need for static rules or expert trajectories.

### 4.1 Learning to Search with RL

RL provides a powerful framework for training LLMs as autonomous agents capable of learning complex behaviors through trial and error (Hou et al., 2025). Recently, a surge of research has applied RL to teach LLMs how to use tools, particularly web search, to solve complex problems. For

instance, Active RAG (Jiang et al., 2023) explored using RL to learn a policy that decides whether to perform another retrieval during the generation process. After the release of OpenAI Deep Research (OpenAI, 2025), a series of works including Search-R1 (Jin et al., 2025a), R1-Searcher (Song et al., 2025), WebThinker (Li et al., 2025b), and DeepResearcher (Zheng et al., 2025) have demonstrated that RL can enable agents to discover sophisticated, multi-hop search and reasoning strategies that outperform SFT-based methods on knowledge-intensive tasks (Xu and Peng, 2025).

However, a critical and unifying limitation of these capability-focused RL methods is that they are designed almost exclusively to maximize final answer accuracy. In contrast, our work is the first to systematically incorporate cost into the RL objective, training an agent to explicitly balance task performance with search efficiency.

## 5 Conclusion

We introduced *PragmaticSearch*, a systematic framework that trains cost-aware agents by explicitly balancing task performance with search efficiency. Extensive experiments demonstrate that *PragmaticSearch* drastically reduces retrieval calls across a diverse suite of question-answering benchmarks, all while maintaining competitive task accuracy. By successfully navigating the trade-off between effectiveness and efficiency, our work provides a practical and principled path toward developing powerful, yet economically viable, LLM agents for real-world deployment.

## 547 Limitations

548 While *PragmaticSearch* significantly enhances inference efficiency, the training phase remains computationally intensive due to the group-sampling requirements of the *PS-GRPO* algorithm. Additionally, the transition between the warm-up and annealing stages currently relies on a fixed iteration threshold, which may require manual tuning for significantly different model architectures or domains. Finally, our evaluation focused on fact-centric question answering; further research is needed to determine how the Bayesian utility estimate generalizes to more subjective tasks where the necessity of external information is less clearly defined by binary correctness.

## 562 References

563 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

568 Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

573 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

578 Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6491–6501.

585 Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

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

594 Yuexiang Hou and 1 others. 2025. Reinforcement learning for large language models: A survey. *arXiv preprint arXiv:2501.00001*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Xiaohan Jin, Yongqi Mei, Tongxuan Zhou, Yifan Wang, Mengfei Liu, Jiawei Chen, Dayiheng Liu, Haojie Pan, Bowen Li, Tianyu Yang, and 1 others. 2025a. Search-r1: Searching for better reasoning steps in test time. *arXiv preprint arXiv:2501.14438*.

Ziniu Jin and 1 others. 2025b. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

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

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP '23*, New York, NY, USA. Association for Computing Machinery.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Zekun Li and 1 others. 2025a. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Zhongxiang Li, Ming Zhang, and Shuai Chen. 2025b. Webthinker: Learning to reason with web search. *arXiv preprint arXiv:2501.10555*.

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

654	OpenAI. 2025. <a href="#">Deep research system card</a>   openai.	709
655	Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt,	710
656	Noah A Smith, and Mike Lewis. 2022. Measuring	711
657	and narrowing the compositionality gap in language	
658	models. <i>arXiv preprint arXiv:2210.03350</i> .	
659	Qwen, :, An Yang, Baosong Yang, Beichen Zhang,	
660	Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan	
661	Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan	
662	Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin	
663	Yang, Jiayi Yang, Jingren Zhou, and 25 oth-	
664	ers. 2025. <a href="#">Qwen2.5 technical report</a> . <i>Preprint</i> ,	
665	arXiv:2412.15115.	
666	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu,	
667	Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and	
668	Daya Guo. 2024. Deepseekmath: Pushing the limits	
669	of mathematical reasoning in open language models.	
670	<i>arXiv preprint arXiv:2402.03300</i> .	
671	Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin	
672	Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin	
673	Lin, and Chuan Wu. 2024. Hybridflow: A flexible	
674	and efficient rlhf framework. <i>arXiv preprint arXiv:</i>	
675	<i>2409.19256</i> .	
676	Weijia Shi, Sewon Min, Michihiro Yasunaga, Min-	
677	joon Seo, Rich James, Mike Lewis, Luke Zettle-	
678	moyer, and Wen-tau Yih. 2023. Replug: Retrieval-	
679	augmented black-box language models. <i>arXiv</i>	
680	<i>preprint arXiv:2301.12652</i> .	
681	Yuhang Song, Chen Wang, and Zihan Xu. 2025. R1-	
682	searcher: Leveraging reinforcement learning for au-	
683	tomated search in reasoning tasks. <i>arXiv preprint</i>	
684	<i>arXiv:2501.14623</i> .	
685	Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan	
686	Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei	
687	Huang, and Jingren Zhou. 2025a. Zerosearch: Incent-	
688	ivize the search capability of llms without searching.	
689	<i>arXiv preprint arXiv:2505.04588</i> .	
690	Yutao Sun, Zhenyu Li, Yike Zhang, Tengyu Pan, Bowen	
691	Dong, Yuyi Guo, and Jianyong Wang. 2025b. Effi-	
692	cient attention mechanisms for large language mod-	
693	els: A survey. <i>arXiv preprint arXiv:2507.19595</i> .	
694	Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot,	
695	and Ashish Sabharwal. 2022. Musique: Multi-	
696	hop questions via single-hop question composition.	
697	<i>Transactions of the Association for Computational</i>	
698	<i>Linguistics</i> , 10:539–554.	
699	Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam,	
700	Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan,	
701	Yi Zhu, Quanlu Zhang, and 1 others. 2023. Efficient	
702	large language models: A survey. <i>arXiv preprint</i>	
703	<i>arXiv:2312.03863</i> .	
704	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten	
705	Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and	
706	Denny Zhou. 2022. Chain of thought prompting	
707	elicits reasoning in large language models. <i>arXiv</i>	
708	<i>preprint arXiv:2201.11903</i> .	
	Renjun Xu and Jingwen Peng. 2025. A comprehensive	709
	survey of deep research: Systems, methodologies,	710
	and applications. <i>arXiv preprint arXiv:2506.12594</i> .	711
	Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen,	712
	Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao,	713
	and Tianyi Zhou. 2024. A survey on knowledge dis-	714
	tillation of large language models. <i>arXiv preprint</i>	715
	<i>arXiv:2402.13116</i> .	716
	An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui,	717
	Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu,	718
	Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.	719
	5 technical report. <i>arXiv preprint arXiv:2412.15115</i> .	720
	Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Ben-	721
	gio, William W Cohen, Ruslan Salakhutdinov, and	722
	Christopher D Manning. 2018. Hotpotqa: A dataset	723
	for diverse, explainable multi-hop question answer-	724
	ing. <i>arXiv preprint arXiv:1809.09600</i> .	725
	Ori Yoran, Tomer Wolfson, Yoav Ziser, and Jonathan Be-	726
	rant. 2023. Answering questions by meta-reasoning	727
	over multiple chains of thought. <i>arXiv preprint</i>	728
	<i>arXiv:2304.13007</i> .	729
	Wenhao Yu, Dan Iter, Shuohang Wang, Yichong	730
	Xu, Mingxuan Ju, Soumya Sanyal, Chenguang	731
	Zhu, Michael Zeng, and Meng Jiang. 2022. Gen-	732
	erate rather than retrieve: Large language mod-	733
	els are strong context generators. <i>arXiv preprint</i>	734
	<i>arXiv:2209.10063</i> .	735
	Xinyue Zheng, Lei Wang, and Yuting Zhang. 2025.	736
	Deepresearcher: Reasoning through long con-	737
	texts with reinforcement learning. <i>arXiv preprint</i>	738
	<i>arXiv:2501.08889</i> .	739
	Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping	740
	Wang. 2024. A survey on model compression for	741
	large language models. <i>Transactions of the Associa-</i>	742
	<i>tion for Computational Linguistics</i> , 12:1556–1577.	743

## A Declaration of LLM Usage

Throughout the preparation of this manuscript, large language models served only as linguistic aids. They were invoked solely to (1) enhance sentence clarity and fluency, (2) correct grammar and adjust style for better readability, and (3) propose alternative wordings that preserved the intended technical meaning. LLMs played no role in study design, data gathering, algorithm creation, experimental execution, or outcome interpretation. All methodological insights, implementation choices, and scientific conclusions were developed independently by the authors, who affirm that no new ideas, data, or claims were generated by these tools and that the intellectual substance of the work remains exclusively their own.

## B Additional Related Works

### B.1 Efficient Large Language Models

Given the substantial computational demands of LLMs, a vibrant research area is dedicated to improving their inference efficiency (Wan et al., 2023). These efforts primarily focus on reducing the model’s intrinsic computational cost. Key techniques include model compression, such as quantization (reducing numerical precision), pruning (removing redundant weights), and knowledge distillation (training a smaller model to mimic a larger one) (Zhu et al., 2024; Xu et al., 2024). Another major direction is the development of efficient architectures, most notably through novel attention mechanisms that approximate the standard quadratic-complexity self-attention with more scalable, linear-time alternatives (Sun et al., 2025b). The common goal of these methods is to lower the latency, memory, and energy consumption of a single forward pass of the model. Our work is *orthogonal* to these model-centric optimizations. While they reduce the computational cost of a single forward pass, we focus on improving the *strategic efficiency* of an agent’s policy to minimize the number of costly external actions (e.g., API calls).

## C Implementation Details

### C.1 Training Infrastructure and Framework

We implement our *PragmaticSearch* framework using the verl training infrastructure (Sheng et al., 2024), which provides efficient distributed training capabilities for reinforcement learning with large language models. All experiments are conducted

on a single node with 8 NVIDIA A100-SXM4-40GB GPUs interconnected via NVLink. The compute node is equipped with dual AMD EPYC 7742 64-Core Processors (256 CPU cores in total) and 512 MiB L3 cache, ensuring sufficient computational resources for both model training and search simulation.

### C.2 Model Configuration

We conduct experiments with multiple base models to validate the generalizability of our approach.

**Primary Models:** Qwen2.5-3B (Base/Instruct) (Yang et al., 2024), Qwen2.5-7B (Base/Instruct), and Llama-3.2-3B-Instruct (Dubey et al., 2024).

**Context Length:** Maximum prompt length of 4,096 tokens and maximum response length of 512 tokens.

**Generation Settings:** During rollout, we employ  $n = 5$  parallel agents with temperature  $T = 1.0$  for diverse response generation.

### C.3 Two-Stage Curriculum Training

**Stage 1: Learning to Search (Warm-up).** In the first stage, we focus on developing the model’s search and reasoning capabilities.

**Training Steps:** 150 steps.

**Reward Configuration:** Set  $\alpha = 0.01$  for search and thinking length rewards (Equation 8).

**Stage 2: Learning When Not to Search (Annealing).** The annealing stage refines the model’s selective search behavior.

**Training Steps:** 52 steps for efficiency optimization.

**Dynamic Rewards:** Sigmoid activation with performance-based adjustment (Equation 4).

### C.4 Reward Design

The reward signal serves as the primary supervision in the reinforcement learning process. We adopt a rule-based reward function that focuses solely on answer accuracy.

Following ZeroSearch (Sun et al., 2025a), we use F1 score instead of Exact Match (EM) as the reward metric. During preliminary experiments, we observed that using EM as the reward metric often led to reward hacking: the policy model tended to produce excessively long answers to increase the chance of including the correct answer. To mitigate this issue, we adopt an F1 score-based reward, which balances precision and recall and is

840 calculated as:

$$841 r_{F1}(y, \hat{y}) = \frac{2 \times |y \cap \hat{y}|}{|y| + |\hat{y}|},$$

842 where  $y$  is the ground truth answer and  $\hat{y}$  is the  
843 model’s prediction. Both  $y$  and  $\hat{y}$  are tokenized  
844 into sets of words for the computation.

### 845 C.5 Optimization Hyperparameters

846 We employ the following optimization settings  
847 across both training stages.

848 **Learning Rate:**  $1 \times 10^{-6}$  with cosine decay  
849 schedule.

850 **Warm-up:** 95% of total steps for learning rate  
851 warm-up.

852 **Batch Sizes:** Training batch size of 12, valida-  
853 tion batch size of 12.

854 **PPO Configuration:** Mini-batch size of 192,  
855 micro-batch size of 48.

856 **KL Penalty:** Coefficient  $\beta = 0.001$  with low-  
857 variance KL loss formulation.

858 **Memory Optimization:** FSDP with parameter,  
859 gradient, and optimizer offloading enabled.

### 860 C.6 Search Simulation and Retrieval

861 **Training-Time Search Simulation.** Following Ze-  
862 roSearch (Sun et al., 2025a), we employ a 14B  
863 parameter simulation LLM to generate search re-  
864 sults during training, eliminating dependency on  
865 external APIs.

866 **Simulation Model:** A fine-tuned LLM  
867 (Simulation\_LLM\_google\_14B\_V2) deployed via  
868 vLLM (Kwon et al., 2023).

869 **Deployment:** Tensor parallelism across 2 GPUs  
870 with 90% GPU memory utilization.

871 **Throughput:** Maximum 1,024 sequences with  
872 optimized batching.

873 **Document Generation:** Controlled quality  
874 through prompt engineering with adjustable noise  
875 injection.

876 **Test-Time Real Search.** During evaluation, we  
877 use real Google Search API via SerpAPI for au-  
878 thentic retrieval.

879 **Search Engine:** Google Search with top-5 re-  
880 sults retrieval.

881 **API Configuration:** Rate-limited queries to  
882 avoid throttling.

883 **Result Processing:** Extract and concatenate rel-  
884 evant snippets up to 2,048 tokens.

### 885 C.7 Dataset Configuration

886 We utilize the ZeroSearch dataset (Sun et al.,  
887 2025a) organized as follows.

888 **Training Data:** Questions from diverse QA  
889 benchmarks stored in Parquet format.

890 **Validation Data:** Held-out test split for monitor-  
891 ing training progress.

892 **Data Loading:** Shuffled training dataloader with  
893 drop\_last=True for consistent batch sizes.

894 **Prompt Processing:** Maximum prompt length  
895 of 4,096 tokens with truncation at word boundaries.

### 896 C.8 Evaluation Protocol

897 **Validation Frequency:** Every 600 training steps.

898 **Checkpoint Saving:** Every 50 steps with best  
899 model selection based on validation performance.

900 **Evaluation Metrics:** Accuracy, average search  
901 counts (ST), and search standard deviation (SD).

902 **Reward Function:** F1-score based verification  
903 for answer correctness.

### 904 C.9 Training Infrastructure and 905 Computational Cost

906 All experiments were conducted on a single server  
907 node equipped with eight NVIDIA A100 40GB  
908 GPUs.

909 **Hardware Configuration:** The workload was  
910 distributed as follows. Simulation Environment: 2  
911 GPUs were dedicated to running the vLLM-based  
912 search simulator. Model Training: 6 GPUs were  
913 used for the main training loop.

914 **Memory Utilization:** The two simulation GPUs  
915 each operated at approximately 90% memory ca-  
916 pacity. The six training GPUs each maintained an  
917 average memory utilization of approximately 90%  
918 throughout the training process.

919 **Training Duration:** The full two-stage train-  
920 ing required approximately 8 hours to complete.  
921 Stage 1 (Warmup): ~4 hours (150 steps). Stage 2  
922 (Annealing): ~4 hours (100-150 steps).

923 **Total Computational Cost:** The total compute  
924 for a complete training run is estimated at **64 GPU-**  
925 **hours**, derived from (6 training GPUs  $\times$  8 hours) +  
926 (2 simulation GPUs  $\times$  8 hours).

### 927 C.10 Reproducibility

928 To ensure reproducibility of our results. **Random**  
929 **Seeds:** Fixed random seed (42) for model initial-  
930 ization. **Code Release:** Full training and evalua-  
931 tion code will be made available upon publication.  
932 **Model Checkpoints:** Trained model weights for

933	both Stage 1 and Stage 2 will be released. <b>Logging:</b>	of-the-art RL-based framework for training RAG	982
934	Comprehensive tracking via Weights & Biases for	models. It introduces a search simulator to avoid	983
935	all experiments. <b>Environment:</b> Docker container	expensive real-time API calls during training and	984
936	with exact package versions provided.	uses a curriculum learning approach. Unlike our	985
937		proposed <i>PragmaticSearch</i> , its reward mechanism	986
	<b>D Experiment Setup</b>	does not explicitly optimize for search efficiency.	987
938		This serves as our primary RL baseline. <b>Search-</b>	988
	<b>D.1 Benchmarks</b>	<b>o1</b> (Li et al., 2025a): An agentic RAG framework	989
939	We evaluate our framework on a diverse set of ques-	that employs an iterative search-and-reasoning loop	990
940	tion answering benchmarks to assess its search and	to autonomously decide when to search. Unlike	991
941	reasoning capabilities across varying complexity.	<i>PragmaticSearch</i> , it focuses on agentic autonomy	992
942	<b>Single-Hop Question Answering:</b> These bench-	rather than specialized efficiency. <b>Search-R1</b> (Jin	993
943	marks require retrieving a single piece of informa-	et al., 2025b): A framework that uses reinforce-	994
944	tion to answer the question. <b>Natural Questions</b>	ment learning to train models to integrate search	995
945	<b>(NQ)</b> (Kwiatkowski et al., 2019): Questions posed	actions into their reasoning trajectories. It serves	996
946	by real users to Google search. <b>TriviaQA</b> (Joshi	as a strong RL-based baseline for reasoning-heavy	997
947	et al., 2017): A challenging dataset of trivia ques-	search tasks.	998
948	tions. <b>PopQA</b> (Mallen et al., 2022): A dataset of		
949	popular questions about entities.	<b>E Results on Instruct Models</b>	999
950	<b>Multi-Hop Question Answering:</b> These bench-	The results of Instruct models are shown in Table 7	1000
951	marks require finding and reasoning over multiple	and Table 8.	1001
952	pieces of information to construct the answer. <b>Hot-</b>		
953	<b>potQA</b> (Yang et al., 2018): A standard benchmark	<b>F Case Study Analysis</b>	1002
954	for multi-hop reasoning. <b>2WikiMultiHopQA</b> (Ho	We contrast the trajectories of <i>PragmaticSearch</i>	1003
955	et al., 2020): A more complex multi-hop dataset	and ZeroSearch in Table 9 to highlight two critical	1004
956	derived from Wikipedia. <b>Musique</b> (Trivedi et al.,	advantages of our framework:	1005
957	2022): A dataset focusing on questions that require		
958	reasoning over multiple paragraphs. <b>Bamboogle</b>	<b>1. Efficiency: Avoiding Redundant Searches.</b>	1006
959	(Press et al., 2022): A dataset of challenging ques-	In Case 1, <i>PragmaticSearch</i> achieves the correct	1007
960	tions designed to be difficult for standard search	answer with <b>zero search calls</b> , whereas ZeroSearch	1008
961	engines. For all benchmarks, we follow standard	triggers two unnecessary retrievals. This demon-	1009
962	practice and use Exact Match (EM) as the primary	strates that <i>PragmaticSearch</i> effectively calibrates	1010
963	evaluation metric.	its internal knowledge, recognizing when a ques-	1011
964		tion can be solved via internal reasoning, thus sig-	1012
	<b>D.2 Baselines</b>	nificantly reducing redundant search API calls and	1013
965	We compare <i>PragmaticSearch</i> against a compre-	associated financial costs.	1014
966	hensive set of baselines to evaluate its effectiveness		
967	and efficiency. <b>Direct Answer:</b> This is a zero-shot	<b>2. Quality: Deep Reasoning without Halluci-</b>	1015
968	baseline where the model is prompted to answer	<b>nations.</b> In Case 2, <i>PragmaticSearch</i> shows that	1016
969	the question directly without any explicit reason-	its cost-awareness does not lead to a “lazy” pol-	1017
970	ing steps or external information. It measures the	icy. Instead, it executes <b>four deliberate searches</b>	1018
971	model’s inherent knowledge. <b>Chain-of-Thought</b>	to resolve a complex temporal constraint. Un-	1019
972	<b>(CoT)</b> (Wei et al., 2022): We prompt the model	like ZeroSearch, which exhibits factual hallucina-	1020
973	to generate a step-by-step reasoning process be-	tions (e.g., misattributing the author of <i>The Color</i>	1021
974	fore providing the final answer. This baseline tests	<i>Purple</i> ), <i>PragmaticSearch</i> maintains high logical	1022
975	the model’s reasoning capabilities without exter-	rigor—iteratively switching between director and	1023
976	nal retrieval. <b>Standard RAG</b> (Lewis et al., 2020):	actor hypotheses until it identifies the definitive ev-	1024
977	A standard retrieval-augmented generation setup.	idence. Even when both models reach the same	1025
978	For each question, we perform a one-time retrieval	answer, <i>PragmaticSearch</i> provides a high-fidelity	1026
979	using a search engine and provide the retrieved	reasoning trace free from the substantial factual	1027
980	documents as context to the model for answer gen-	errors seen in the baseline.	1028
981	eration. <b>ZeroSearch</b> (Sun et al., 2025a): A state-		

Method	Single-Hop QA									Averages	
	NQ			TriviaQA			PopQA			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
<i>Qwen-2.5-7B (Instruct)</i>											
Direct Answer	11.60	-	-	35.60	-	-	1.20	-	-	16.13	-
CoT	27.00	-	-	45.20	-	-	15.00	-	-	29.07	-
RAG	20.20	<u>0.07</u>	<u>0.25</u>	28.20	<u>0.11</u>	<u>0.32</u>	22.20	<u>0.03</u>	<u>0.18</u>	23.53	<u>0.07</u>
Search-o1	18.60	0.99	0.33	42.40	1.00	0.42	12.60	1.00	0.16	24.53	1.00
Search-R1	40.80	2.35	1.11	58.10	2.41	1.15	10.40	2.54	1.10	36.43	2.43
ZeroSearch	<u>43.60</u>	0.10	0.33	<u>65.40</u>	0.16	0.44	<u>47.40</u>	0.10	0.31	52.13	0.12
<i>PragmaticSearch</i>	<b>49.60</b>	<b>0.000</b>	<b>0.000</b>	<b>70.80</b>	<b>0.000</b>	<b>0.000</b>	<b>56.80</b>	<b>0.000</b>	<b>0.000</b>	<b>59.07</b>	<b>0.000</b>
<i>Qwen-2.5-3B (Instruct)</i>											
Direct Answer	16.20	-	-	26.60	-	-	14.40	-	-	19.07	-
CoT	19.40	-	-	35.60	-	-	8.20	-	-	21.07	-
RAG	16.20	<b>0.18</b>	<u>0.41</u>	28.20	<b>0.24</b>	<u>0.46</u>	25.60	<b>0.36</b>	<u>0.49</u>	23.33	<b>0.26</b>
Search-o1	16.60	0.82	0.88	31.00	1.10	1.33	8.20	0.74	0.66	18.60	0.00
Search-R1	33.80	0.95	0.77	51.50	0.90	0.81	<u>41.20</u>	0.97	0.86	42.17	0.94
ZeroSearch	<u>47.80</u>	0.99	<b>0.28</b>	<u>64.40</u>	1.00	<b>0.35</b>	<b>64.60</b>	1.01	<b>0.08</b>	<u>58.93</u>	1.00
<i>PragmaticSearch</i>	<b>48.60</b>	<u>0.56</u>	<u>0.74</u>	<b>71.80</b>	<u>0.60</u>	<u>0.67</u>	<b>64.60</b>	<u>0.56</u>	<u>0.64</u>	<b>61.67</b>	<u>0.57</u>
<i>LLaMA-3.2-3B (Instruct)</i>											
Direct Answer	16.20	-	-	29.60	-	-	7.40	-	-	17.73	-
CoT	26.20	-	-	44.40	-	-	2.80	-	-	24.47	-
RAG	28.80	<u>0.78</u>	0.74	45.60	<u>0.68</u>	<u>0.68</u>	35.80	<u>0.62</u>	<u>0.54</u>	36.73	<u>0.69</u>
Search-o1	28.20	0.93	0.88	48.40	0.75	<u>0.68</u>	8.80	0.74	0.86	28.47	0.81
Search-R1	<u>39.40</u>	1.07	0.68	<u>57.20</u>	1.30	0.94	43.60	0.89	0.94	46.73	1.09
ZeroSearch	38.00	1.04	0.72	54.40	0.91	0.74	<b>57.40</b>	0.77	0.57	49.93	0.91
<i>PragmaticSearch</i>	<b>41.40</b>	<b>0.00</b>	<b>0.00</b>	<b>60.60</b>	<b>0.00</b>	<b>0.00</b>	<u>48.60</u>	<b>0.00</b>	<b>0.00</b>	<b>50.20</b>	<b>0.00</b>

Table 7: Main results for Single-Hop QA tasks using Instruct LLMs as the backbone. The best performance is set in bold. Acc: Accuracy (%), ST: Search Times, SD: Search Standard Deviation.

Method	Multi-Hop QA												Averages	
	HotpotQA			2Wiki			Musique			Bamboogle			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
<i>Qwen-2.5-7B (Instruct)</i>														
Direct Answer	16.40	-	-	22.20	-	-	4.80	-	-	14.40	-	-	14.45	-
CoT	21.00	-	-	24.80	-	-	8.00	-	-	26.39	-	-	20.05	-
RAG	17.40	<b>0.28</b>	<b>0.51</b>	19.20	<b>0.63</b>	<b>0.80</b>	7.40	<b>0.26</b>	<u>0.55</u>	26.39	<b>0.07</b>	<b>0.25</b>	17.60	<b>0.31</b>
Search-o1	17.60	1.58	0.66	29.60	2.05	0.92	9.80	1.64	0.80	30.20	1.42	0.65	21.80	1.67
Search-R1	<u>33.40</u>	2.87	1.12	<u>37.40</u>	3.12	1.18	<u>19.80</u>	2.92	1.11	<b>35.20</b>	2.63	0.89	<u>31.45</u>	2.89
ZeroSearch	23.00	0.52	0.67	27.80	1.01	1.08	8.80	0.54	0.69	<u>33.60</u>	0.34	0.52	23.30	0.60
<i>PragmaticSearch</i>	<b>40.00</b>	0.52	0.52	<b>38.60</b>	1.31	1.18	<b>20.20</b>	1.00	0.48	<b>35.20</b>	1.00	0.89	<b>33.50</b>	0.96
<i>Qwen-2.5-3B (Instruct)</i>														
Direct Answer	17.00	-	-	19.00	-	-	4.20	-	-	2.78	-	-	10.75	-
CoT	15.60	-	-	21.00	-	-	4.80	-	-	19.44	-	-	15.21	-
RAG	16.60	<b>0.50</b>	<u>0.65</u>	21.80	<b>0.68</b>	<u>0.73</u>	8.00	<b>0.54</b>	<u>0.69</u>	11.11	<b>0.44</b>	<u>0.62</u>	14.38	<b>0.54</b>
Search-o1	14.80	<u>0.58</u>	<b>0.45</b>	22.40	<u>1.00</u>	0.74	5.20	<u>0.96</u>	0.88	22.40	1.01	0.82	16.20	0.89
Search-R1	32.20	1.12	0.95	28.80	1.48	1.04	11.60	1.15	1.01	18.40	1.12	1.07	22.75	1.22
ZeroSearch	<u>34.80</u>	1.33	<u>0.53</u>	<u>42.80</u>	1.49	<b>0.57</b>	<u>20.00</u>	1.30	<b>0.53</b>	<u>40.00</u>	1.32	<b>0.48</b>	<u>34.40</u>	1.36
<i>PragmaticSearch</i>	<b>41.80</b>	1.32	0.80	<b>54.80</b>	1.78	0.98	<b>30.40</b>	1.58	0.74	<b>40.80</b>	<u>0.78</u>	0.78	<b>41.95</b>	1.36
<i>LLaMA-3.2-3B (Instruct)</i>														
Direct Answer	12.60	-	-	9.20	-	-	2.00	-	-	8.00	-	-	7.95	-
CoT	16.00	-	-	10.20	-	-	5.80	-	-	8.60	-	-	10.15	-
RAG	18.60	<u>0.95</u>	0.82	14.80	<u>1.18</u>	0.87	7.20	<u>0.95</u>	0.86	19.44	<u>0.81</u>	<u>0.70</u>	15.01	<u>0.97</u>
Search-o1	19.40	1.14	0.80	17.40	1.22	0.95	6.00	<u>0.87</u>	<u>0.66</u>	<u>32.00</u>	1.11	0.94	18.70	1.08
Search-R1	<u>21.20</u>	1.09	0.74	20.80	1.60	0.86	10.40	1.22	0.84	25.60	1.34	0.99	19.50	1.31
ZeroSearch	<b>23.00</b>	1.06	0.83	<u>22.00</u>	1.31	0.95	<u>10.60</u>	1.18	<u>0.84</u>	28.80	0.98	0.78	<u>21.10</u>	1.13
<i>PragmaticSearch</i>	<b>23.00</b>	<b>0.00</b>	<b>0.00</b>	<b>30.40</b>	<b>0.00</b>	<b>0.00</b>	<b>12.40</b>	<b>0.23</b>	<b>0.49</b>	<b>34.40</b>	<b>0.49</b>	<b>0.50</b>	<b>25.05</b>	<b>0.18</b>

Table 8: Main results for Multi-Hop QA tasks using Instruct LLMs as the backbone. The best performance is set in bold. Acc: Accuracy (%), ST: Search Times, SD: Search Standard Deviation.

---

### Case Study: Search Behavior Comparison

---

*Question: How old would Elvis Presley have been had he lived to the end of the 20th century? (GT: 65)*

---

**PragmaticSearch:** Correct  
**Search Count: 0**

<think> To answer this question, let's first determine how old Elvis Presley was when he died and then calculate how old he would have been if he lived to the end of the 20th century. ... 1. Born Jan 8, 1935. 2. Died Aug 16, 1977. 3. Age at death: 42. 4. End of 20th century: Dec 31, 2000. 5. Years to add: 23. 6. 42 + 23 = 65. </think>  
 <answer> 65 </answer>

**ZeroSearch:** Wrong  
**Search Count: 2**

<think> Elvis Presley born Jan 8, 1935. End of 20th century was Dec 31, 1999. I can figure out age by subtracting 1935 from 1999. </think>  
 <search> Elvis Presley died in 1977 </search>  
 <information> Doc 1: Born 1935, Died 1977 (aged 42). </information> <think> Assuming Elvis Presley would have reached the age calculation to September 29th... </think> <search> elvis presley age chart 1935-present. </search>  
 <information> Doc 4: If Presley were alive in 2025, he would be 90. </information>  
 <answer> 42 </answer>

*Question: Beloved in 1999 was whose first movie since The Color Purple in 1985? (GT: Oprah Winfrey)*

---

**PragmaticSearch:** Correct  
**Search Count: 4**

<think> To answer this, I need to find who was in both films and had a gap. Let's check the director first. </think> <search> First movie after Color Purple for director of Beloved </search>  
 <information> Doc 3: Spielberg directed Color Purple (1985)... Beloved (1998) directed by Jonathan Demme. </information> <think> Spielberg isn't the one. Check the lead actress, Oprah Winfrey. </think> <search> Who was the lead actress in Beloved and her first movie after Color Purple </search> <information> Doc 3: Beloved (1998) - Back on the big screen for the first time since The Color Purple... </information> <think> Verify other movies between 1985 and 1998. </think> <search> Oprah Winfrey filmography 1985-1998 </search> <information> Doc 5: Oprah made her acting debut in 1985... In 1998 produced Beloved. </information> <think> Confirmed. </think> <answer> Oprah Winfrey </answer>

**ZeroSearch:** Correct  
**Search Count: 2**

<think> The film is "Beloved" (1998). One lead was Toni Morrison, the author. Wait, the timeline suggests searching for an actor or director. </think> <search> Who starred in Beloved released in 1985's The Color Purple </search>  
 <information> Doc 1: The Color Purple stars Whoopi Goldberg, Danny Glover, Oprah Winfrey. </information> <think> I need to see who matches the 1999 Beloved movie. </think> <search> Who starred in 1999 film Beloved after being a star since 1985 movie The Color Purple </search> <information> Doc 2: Oprah Winfrey as Sethe. </information> <think> Oprah was in both. Even though I thought Morrison wrote Color Purple (it was Alice Walker), Oprah matches the timeline. </think> <answer> Oprah Winfrey </answer>

Table 9: Comparison of search trajectories. Case 1 (Top) demonstrates **efficiency**: *PragmaticSearch* avoids redundant searches using internal knowledge (0 vs. 2 calls). Case 2 (Bottom) demonstrates **quality**: *PragmaticSearch* employs a rigorous 4-step search to ensure logical integrity and factual accuracy, whereas the baseline suffers from factual hallucinations.