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: 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*.

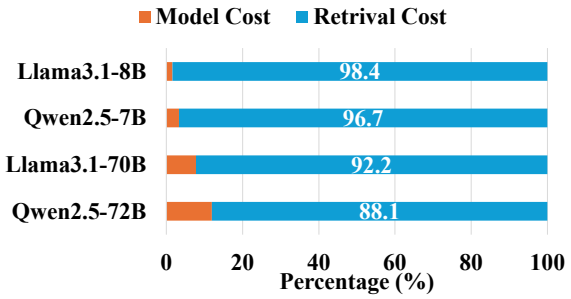

Figure 1: Breakdown of average inference cost per query for RAG systems. Retrieval costs (orange) dominate the cost compared to model inference (blue), highlighting the need for efficient retrieval strategies.

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-

prove task *capability*, it fundamentally fails to address the underlying efficiency problem.

In this work, we argue that an effective RAG agent must learn to explicitly navigate the trade-off between task performance and computational cost. To this end, we introduce *PragmaticSearch*, a systematic RL framework designed to train cost-aware RAG agents. A naive solution might involve simply adding a fixed penalty for each search action to the reward function. However, our initial investigations reveal that this approach is highly unstable, often leading to training collapse or sub-optimal "lazy" policies where the agent learns to avoid searching altogether.

To overcome this instability, *PragmaticSearch* employs a carefully designed *two-stage curriculum* with *Two-Stage Advantage Shaping (TSAS)*. The first stage, a Capability-Building phase, focuses on developing search competence. Here, the advantage function is shaped to be cost-agnostic, rewarding exploration and reasoning to build a robust understanding of how to effectively use the retrieval tool. The second stage, an Efficiency-Cultivating phase, transitions the objective to cost-awareness. It introduces a novel advantage function where a soft performance gate, conditioned on the group-relative quality of the generated answer, dynamically modulates the reward. This gate is derived from a Bayesian estimate of retrieval utility and serves to refine the optimization process by neutralizing penalties for high-utility searches while simultaneously punishing redundant ones. This entire curriculum is optimized using our *PragmaticSearch Group-Relative Policy Optimization* (*PS-GRPO*) algorithm, which robustly teaches the agent a judicious policy on when to search. Extensive experiments across diverse single-hop and multi-hop question-answering benchmarks (Kwiatkowski et al., 2019; Joshi et al., 2017; Yang et al., 2018; Ho et al., 2020) demonstrate that this framework achieves significant efficiency gains, reducing retrieval calls by up to 76.2% with negligible impact on task performance.

Our contributions are as follows:

- We identify the critical flaw in existing RL-based RAG frameworks: their myopic focus on accuracy leads to inefficient, costly search policies unsuited for practical deployment.
- We propose a systematic two-stage curriculum learning framework that decouples capability-building from efficiency-tuning, proving essential for stable training.
- We design a novel adaptive policy optimization mechanism that synergizes a performance-gated reward function with a stabilized group-based algorithm, enabling the model to robustly learn a cost-efficient policy.
- We provide comprehensive empirical validation across seven diverse QA benchmarks, demonstrating that *PragmaticSearch* drastically reduces retrieval costs while maintaining competitive performance.

2 Methodology

2.1 Problem Formulation

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

- **State (s_t):** The state at timestep t is a tuple $s_t = (\mathbf{q}, h_{t-1})$, where $\mathbf{q}$ is the initial user query and h_{t-1} is the history of past actions and their corresponding observations.
- **Action (a_t):** At each step, the agent chooses an action from a discrete set $\mathcal{A} = \{\text{SEARCH(QUERY)}, \text{GENERATE(ANSWER)}\}$.
- **Policy (π_θ):** The agent's policy $\pi_\theta(a_t|s_t)$ is represented by an LLM with parameters θ.
- **Trajectory (τ):** A trajectory is a full sequence of states and actions, $\tau = (s_0, a_0, \ldots, s_T, a_T)$, generated by the policy.
- **Reward and Advantage:** The policy π_θ aims to maximize the expected reward $\mathbb{E}\tau \sim \pi\theta[R(\tau)]$. To mitigate the high variance of raw rewards $R(\tau)$, we utilize the advantage $A(\tau)$, which measures performance relative to a baseline. Our core contribution involves shaping this advantage $A(\tau)$ through a curriculum to stabilize training.

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

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," π_{capable}, that maximizes a composite reward encouraging both task success and exploration. This can be formally described as:

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

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:

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

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:

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

This function combines the group-normalized task reward with intrinsic rewards for search (S) and reasoning (T_{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 α controls the relative importance of reasoning depth, while ϵ 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:

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

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: $\log(S(\mathbf{q}, \mathbf{o}_i) + 1) - \text{std} \cdot$.
- T_i^+: A reward for reasoning length, given by the logarithm of the trajectory's token length: $\log(T_{\text{len}}(\mathbf{q}, \mathbf{o}_i) + 1)$.
- σ_i: The critical *soft performance gate*, $\text{sigmoid}(A_i^+) - 0.5$. When a trajectory is successful ($A_i^+ > 0$), σ_i is positive, preserving the intrinsic rewards from S_i^+ and T_i^+. When it fails ($A_i^+ < 0$), σ_i becomes negative, effectively penalizing costly exploration that did

not lead to success. Importantly, this gating mechanism does not rely on the LLM's internal confidence or uncertainty, which can be flawed during hallucinations. Instead, it is strictly conditioned on the objective, external ground-truth reward (e.g., F1 score). Even if the model confidently hallucinates an incorrect answer, the external reward will be low, resulting in a negative advantage and a strict penalty for the trajectory.

2.3 Theoretical Justification

The design of the soft performance gate σ_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: $P(Z = 1|A) = \text{sigmoid}(k(A - c))$.

Crucially, the Gaussian assumption is not an arbitrary modeling choice, but a property enforced by our optimization design. In *PS-GRPO*, the group-relative advantages (A_i^+) are explicitly normalized within the batch to have a mean of 0 and a standard deviation of 1. This normalization naturally causes the empirical distribution of advantages to approximate the Gaussian properties assumed in our derivation, making the theoretical foundation practically sound.

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.

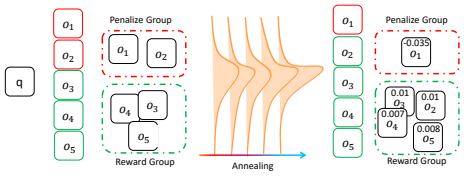

Figure 2: Evolution of the Group Relative Score distribution for A_i^+. q denotes the query; $o_1 \sim o_5$ are the model-sampled group responses.

2.4 PS-GRPO

To optimize the policy π_θ 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:

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

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 $\pi_{\theta_{\text{old}}}$, $\hat{A}_{i,t}$ 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 ϵ, 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

Method	Single-Hop QA									Averages	
	NQ			TriviaQA			PopQA			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
Qwen-2.5-7B *(Base)*											
Direct Answer	12.40	-	-	21.80	-	-	7.20	-	-	13.80	-
CoT	21.40	-	-	34.60	-	-	13.00	-	-	23.00	-
RAG	20.60	0.68	0.81	31.60	0.67	0.79	22.20	0.52	0.67	24.80	0.62
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	0.46	62.20	1.32	0.59	49.20	1.15	0.37	51.20	1.23
ZeroSearch	43.40	1.12	0.66	61.40	1.13	0.65	57.00	1.18	0.64	53.93	1.15
PragmaticSearch	**49.60**	**0.53**	**0.23**	**71.60**	**0.47**	**0.52**	**63.40**	**0.05**	**0.23**	**61.53**	**0.35**
Qwen-2.5-3B *(Base)*											
Direct Answer	7.00	-	-	14.40	-	-	4.00	-	-	8.47	-
CoT	9.00	-	-	13.60	-	-	6.00	-	-	9.53	-
RAG	10.40	0.50	0.63	16.20	0.57	0.69	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	0.46	47.07	1.15
ZeroSearch	44.60	0.50	0.50	64.60	0.21	0.41	64.60	0.30	0.46	57.93	0.34
PragmaticSearch	**48.00**	**0.23**	**0.43**	**65.80**	**0.02**	**0.15**	**66.20**	**0.00**	**0.00**	**60.00**	**0.08**
LLaMA-3.2-3B *(Base)*											
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	0.68	0.68	35.80	0.62	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	**0.39**	45.97	1.15
ZeroSearch	44.60	0.50	0.50	64.60	**0.21**	0.41	**64.60**	**0.30**	0.46	57.93	**0.34**
PragmaticSearch	**46.40**	**0.10**	**0.13**	**68.60**	1.00	**0.00**	63.40	1.01	0.77	**59.47**	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.

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 σ_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 reinforcement 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.

Setup. We train on the ZeroSearch dataset (Sun et al., 2025a) and evaluate on seven single- and multi-hop QA benchmarks (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) (details in Appendix D.1). Training uses F1 score as the reward, while evaluation uses Exact Match (EM). To measure cost and stability, we report the average (ST) and standard deviation (SD) of searches. For models, we use *Qwen2.5 (3B, 7B)* (Qwen et al., 2025) and *LLaMA-3.2 (3B)* (Dubey et al., 2024) in both `Base` and `Instruct` variants. Baselines include closed-book approaches (Direct Answer, CoT (Wei et al., 2022)), standard RAG (Lewis et al., 2020), and advanced agentic/RL frameworks (Search-o1 (Li et al., 2025a), Search-R1 (Jin et al., 2025b), and ZeroSearch (Sun et al., 2025a)).

3.2 Main Results

The results in Table 1 and Table 2 show that *PragmaticSearch* establishes a more favorable performance-efficiency frontier. Specifically, it shows superior performance-cost trade-off, its generalizability, and its enhanced policy stability.

***PragmaticSearch* Establishing a Superior Performance-Efficiency Frontier.** *Pragmatic-*

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		
Qwen-2.5-7B *(Base)*														
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	0.76	0.88	19.40	0.92	0.96	8.00	0.85	0.91	18.06	0.81	0.95	17.12	0.84
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	0.90	18.80	2.09	0.74	35.20	1.91	0.63	30.65	2.08
ZeroSearch	35.20	1.43	0.83	38.40	1.58	0.97	20.80	1.49	0.95	**44.00**	0.74	0.89	34.60	1.31
PragmaticSearch	**43.40**	**0.24**	**0.71**	**47.40**	**0.57**	**0.82**	**26.60**	**0.13**	**0.37**	**44.00**	**0.57**	**0.58**	**40.35**	**0.38**
Qwen-2.5-3B *(Base)*														
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	0.68	0.78	1.20	0.65	0.78	5.56	0.64	0.63	6.09	0.66
Search-o1	14.60	0.91	0.63	22.40	1.02	0.70	5.20	0.95	0.87	**22.40**	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	**37.80**	**0.26**	**0.45**	34.20	0.93	0.36	18.20	0.89	**0.32**	22.22	**0.00**	**0.00**	**28.11**	0.52
PragmaticSearch	37.40	0.51	0.50	**37.20**	**0.00**	**0.00**	**18.40**	**0.26**	0.44	16.00	**0.00**	**0.00**	27.25	**0.19**
LLaMA-3.2-3B *(Base)*														
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	0.60	0.86	9.60	0.77	0.83	6.40	0.83	0.77	16.40	0.83	0.85	11.20	0.76
Search-o1	19.40	1.30	0.87	17.40	1.53	0.99	6.00	1.24	0.91	32.00	1.45	0.81	18.70	1.38
Search-R1	29.80	1.01	0.83	30.80	1.52	1.12	13.40	0.85	0.77	18.44	0.91	0.89	23.11	1.07
ZeroSearch	27.80	**0.58**	0.96	27.50	**0.63**	0.87	12.00	**0.66**	0.95	**32.20**	0.64	0.92	24.88	**0.63**
PragmaticSearch	**35.00**	1.00	**0.06**	**37.40**	1.01	**0.13**	**19.40**	1.00	**0.04**	16.00	**0.58**	**0.49**	**26.95**	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
***PragmaticSearch* (Full)**	**68.60**	**1.00**	**0.00**	**35.00**	**1.00**	**0.06**	**19.40**	**1.00**	**0.04**	**41.00**	**1.00**
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.

Search 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 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

Method	Tokens		Time (ms)	
	Base	Instruct	Base	Instruct
Qwen-2.5-3B				
PragmaticSearch	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
Qwen-2.5-7B				
PragmaticSearch	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.

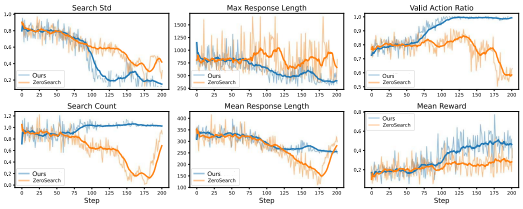

Figure 3: RL fine-tuning dynamics of *PragmaticSearch* and *ZeroSearch*. Solid lines are moving averages over 15 steps.

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) 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. We also conducted detailed sensitivity analyses on the two-stage transition threshold and the Soft Performance Gate. Due to space constraints, we refer the reader to Appendix F for these extended ablation studies, which analyze transition-timing sensitivity and show that the Soft Performance Gate significantly improves search efficiency without requiring heavy manual tuning.

3.4 Complementary Study

To investigate the generalizability and modularity of our curriculum, we integrated our Stage 1 (Warm-up) into $ZeroSearch$ (denoted $ZeroSearch^{+}$). Results in Table 5 and Table 6 show $ZeroSearch^{+}$ outperforms $ZeroSearch$ on all datasets, raising average single-hop accuracy from 50.33 to 61.13 and multi-hop from 18.13 to 23.60. This demonstrates our warm-up strategy is an effective, transferable module. Yet, lacking explicit annealing, $ZeroSearch^{+}$ remains suboptimal in search budget and stability.

3.5 Case Study

To qualitatively analyze behavior, we visualize search trajectories in Table 11 (Appendix G). *PragmaticSearch* successfully balances internal knowledge usage and complex search executions.

4 Related Work

Retrieval-Augmented Generation (RAG) integrates external knowledge to mitigate LLM hallucinations

Method	Single-Hop QA									Averages	
	NQ			TriviaQA			PopQA			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
LLaMA-3.2-3B-Instruct											
ZeroSearch	38.00	**1.04**	0.72	54.40	**0.91**	0.74	57.40	**0.77**	0.57	49.93	**0.91**
$ZeroSearch^{+}$	**47.20**	1.07	**0.33**	**63.80**	1.04	**0.29**	**72.40**	1.00	**0.17**	**61.13**	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		
LLaMA-3.2-3B-Instruct														
ZeroSearch	23.00	**1.06**	0.83	22.00	1.31	0.95	10.60	1.18	0.84	28.80	**0.98**	0.78	21.10	1.13
$ZeroSearch^{+}$	**29.40**	1.11	**0.41**	**23.80**	**1.19**	**0.50**	**14.80**	**1.10**	**0.38**	**26.40**	1.06	**0.26**	**23.60**	**1.12**

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.

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 presented *PragmaticSearch*, a framework that trains cost-aware agents by explicitly balancing task performance and search efficiency. Empirical results across diverse QA benchmarks demonstrate that *PragmaticSearch* significantly reduces retrieval calls while maintaining competitive accuracy, facilitating the deployment of economically viable LLM agents. Although our evaluation primarily focused on RAG settings, the core mechanism of dynamic advantage shaping based on trajectory utility is inherently tool-agnostic. Future work will extend this framework to multi-tool agents utilizing other costly operations, such as code execution or paid APIs.

Limitations

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.

References

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*.

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*.

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*.

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.

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.

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*.

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 Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 7.

OpenAI. 2025. Deep research system card | openai.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Yuhang Song, Chen Wang, and Zihan Xu. 2025. R1-searcher: Leveraging reinforcement learning for automated search in reasoning tasks. *arXiv preprint arXiv:2501.14623*.

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

Yutao Sun, Zhenyu Li, Yike Zhang, Tengyu Pan, Bowen Dong, Yuyi Guo, and Jianyong Wang. 2025b. Efficient attention mechanisms for large language models: A survey. *arXiv preprint arXiv:2507.19595*.

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

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, and 1 others. 2023. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

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

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

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

Ori Yoran, Tomer Wolfson, Yoav Ziser, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought. *arXiv preprint arXiv:2304.13007*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*.

Xinyue Zheng, Lei Wang, and Yuting Zhang. 2025. Deepresearcher: Reasoning through long contexts with reinforcement learning. *arXiv preprint arXiv:2501.08889*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

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

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).

validation: Qwen-2.5-7B (Base/Instruct), Qwen-2.5-3B (Base/Instruct) (Qwen et al., 2025), and Llama-3.2-3B-Instruct (Dubey et al., 2024). Our evaluation includes a comprehensive set of baselines: Direct Answer, Chain-of-Thought (CoT) (Wei et al., 2022), standard Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), and ZeroSearch (Sun et al., 2025a). For prompt-based baselines (Direct Answer and CoT), we utilize Instruct models, as Base models often struggle to follow specific task instructions. For the reinforcement learning-based methods (ZeroSearch and our own), we evaluate both Base and Instruct variants to assess the generalizability of the approach across different model types.

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.

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 H800 80GB 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: 150 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 calculated as:

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

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

C.5 Optimization Hyperparameters

We employ the following optimization settings across both training stages.

Learning Rate: 1×10^{-6} with cosine decay schedule.

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

Batch Sizes: Training batch size of 12, validation batch size of 12.

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

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

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

C.6 Search Simulation and Retrieval

Training-Time Search Simulation. Following ZeroSearch (Sun et al., 2025a), we employ a 14B parameter simulation LLM to generate search results during training, eliminating dependency on external APIs.

Simulation Model: A fine-tuned LLM (`Simulation_LLM_google_14B_V2`) deployed via vLLM (Kwon et al., 2023).

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

Throughput: Maximum 1,024 sequences with optimized batching.

Document Generation: Controlled quality through prompt engineering with adjustable noise injection. Specifically, we use two prompts for the search simulator: one generates plausible search results, and the other produces random or irrelevant "failed search" outputs. Each simulated search request is routed to the failure prompt with a 5% probability to inject noise. This noise-injection strategy successfully prevents the sim-to-real gap when transitioning to the live Google Search API.

Test-Time Real Search. During evaluation, we use real Google Search API via SerpAPI for authentic retrieval.

Search Engine: Google Search with top-5 results retrieval.

API Configuration: Rate-limited queries to avoid throttling.

Result Processing: Extract and concatenate relevant snippets up to 2,048 tokens.

C.7 Dataset Configuration

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

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

Validation Data: Held-out test split for monitoring training progress.

Data Loading: Shuffled training dataloader with drop_last=True for consistent batch sizes.

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

C.8 Evaluation Protocol

Validation Frequency: Every 600 training steps.

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

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

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

C.9 Training Infrastructure and Computational Cost

All experiments were conducted on a single server node equipped with eight NVIDIA H800 80GB GPUs.

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

Memory Utilization: The two simulation GPUs each operated at approximately 90% memory capacity. The six training GPUs each maintained an average memory utilization of approximately 90% throughout the training process.

Training Duration: The full two-stage training required approximately 8 hours to complete. Stage 1 (Warmup): ~4 hours (150 steps). Stage 2 (Annealing): ~4 hours (100-150 steps).

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

C.10 Reproducibility

To ensure reproducibility of our results. **Random Seeds**: Fixed random seed (42) for model initialization. **Code Release**: Full training and evaluation code will be made available upon publication. **Model Checkpoints**: Trained model weights for both Stage 1 and Stage 2 will be released. **Logging**: Comprehensive tracking via Weights & Biases for all experiments. **Environment**: Docker container with exact package versions provided.

D Experiment Setup

D.1 Benchmarks

We evaluate our framework on a diverse set of question answering benchmarks to assess its search and reasoning capabilities across varying complexity.

Single-Hop Question Answering: These benchmarks require retrieving a single piece of information to answer the question. **Natural Questions (NQ)** (Kwiatkowski et al., 2019): Questions posed by real users to Google search. **TriviaQA** (Joshi et al., 2017): A challenging dataset of trivia questions. **PopQA** (Mallen et al., 2022): A dataset of popular questions about entities.

Multi-Hop Question Answering: These benchmarks require finding and reasoning over multiple pieces of information to construct the answer. **HotpotQA** (Yang et al., 2018): A standard benchmark for multi-hop reasoning. **2WikiMultiHopQA** (Ho et al., 2020): A more complex multi-hop dataset derived from Wikipedia. **Musique** (Trivedi et al., 2022): A dataset focusing on questions that require reasoning over multiple paragraphs. **Bamboogle** (Press et al., 2022): A dataset of challenging questions designed to be difficult for standard search engines. For all benchmarks, we follow standard practice and use Exact Match (EM) as the primary evaluation metric.

D.2 Baselines

We compare *PragmaticSearch* against a comprehensive set of baselines to evaluate its effectiveness and efficiency. **Direct Answer**: This is a zero-shot baseline where the model is prompted to answer the question directly without any explicit reasoning steps or external information. It measures the model's inherent knowledge. **Chain-of-Thought (CoT)** (Wei et al., 2022): We prompt the model

Method	Single-Hop QA									Averages	
	NQ			TriviaQA			PopQA			Avg Acc	Avg ST
	Acc	ST	SD	Acc	ST	SD	Acc	ST	SD		
Qwen-2.5-7B *(Instruct)*											
Direct Answer	11.60	-	-	35.60	-	-	1.20	-	-	16.13	-
CoT	27.00	-	-	45.20	-	-	15.00	-	-	29.07	-
RAG	20.20	0.07	0.25	28.20	0.11	0.32	22.20	0.03	0.18	23.53	0.07
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	43.60	0.10	0.33	65.40	0.16	0.44	47.40	0.10	0.31	52.13	0.12
PragmaticSearch	**49.60**	**0.000**	**0.000**	**70.80**	**0.000**	**0.000**	**56.80**	**0.000**	**0.000**	**59.07**	**0.000**
Qwen-2.5-3B *(Instruct)*											
Direct Answer	16.20	-	-	26.60	-	-	14.40	-	-	19.07	-
CoT	19.40	-	-	35.60	-	-	8.20	-	-	21.07	-
RAG	16.20	**0.18**	0.41	28.20	**0.24**	0.46	25.60	**0.36**	0.49	23.33	**0.26**
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	41.20	0.97	0.86	42.17	0.94
ZeroSearch	47.80	0.99	**0.28**	64.40	1.00	**0.35**	**64.60**	1.01	**0.08**	58.93	1.00
PragmaticSearch	**48.60**	0.56	0.74	**71.80**	0.60	0.67	**64.60**	0.56	0.64	**61.67**	0.57
LLaMA-3.2-3B *(Instruct)*											
Direct Answer	16.20	-	-	29.60	-	-	7.40	-	-	17.73	-
CoT	26.20	-	-	44.40	-	-	2.80	-	-	24.47	-
RAG	28.80	0.78	0.74	45.60	0.68	0.68	35.80	0.62	0.54	36.73	0.69
Search-o1	28.20	0.93	0.88	48.40	0.75	0.68	8.80	0.74	0.86	28.47	0.81
Search-R1	39.40	1.07	0.68	57.20	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	**57.40**	0.77	0.57	49.93	0.91
PragmaticSearch	**41.40**	**0.00**	**0.00**	**60.60**	**0.00**	**0.00**	48.60	**0.00**	**0.00**	**50.20**	**0.00**

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.

to generate a step-by-step reasoning process before providing the final answer. This baseline tests the model's reasoning capabilities without external retrieval. **Standard RAG** (Lewis et al., 2020): A standard retrieval-augmented generation setup. For each question, we perform a one-time retrieval using a search engine and provide the retrieved documents as context to the model for answer generation. **ZeroSearch** (Sun et al., 2025a): A state-of-the-art RL-based framework for training RAG models. It introduces a search simulator to avoid expensive real-time API calls during training and uses a curriculum learning approach. Unlike our proposed *PragmaticSearch*, its reward mechanism does not explicitly optimize for search efficiency. This serves as our primary RL baseline. **Search-o1** (Li et al., 2025a): An agentic RAG framework that employs an iterative search-and-reasoning loop to autonomously decide when to search. Unlike *PragmaticSearch*, it focuses on agentic autonomy rather than specialized efficiency. **Search-R1** (Jin et al., 2025b): A framework that uses reinforcement learning to train models to integrate search actions into their reasoning trajectories. It serves as a strong RL-based baseline for reasoning-heavy search tasks.

E Results on Instruct Models

The results of Instruct models are shown in Table 7 and Table 8.

F Additional Ablation Studies

Robustness and Transition Threshold. The transition between stages is determined by monitoring the *valid action ratio* during training. As this ratio approaches 1, it indicates the model has achieved basic task competence. Table 9 compares transitioning at a 90% ratio versus a 99% ratio. The results confirm that delaying the transition until a 99% valid action ratio is reached reliably yields superior performance across multiple datasets (NQ, TriviaQA, PopQA) without degrading the policy.

Soft Performance Gate. We also evaluate the core contribution of the efficiency stage: the Soft Performance Gate. As shown in Table 10, incorporating the dynamic gate dramatically improves performance compared to an un-gated variant. Be-

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		
Qwen-2.5-7B *(Instruct)*														
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	**0.28**	**0.51**	19.20	**0.63**	**0.80**	7.40	**0.26**	0.55	26.39	**0.07**	**0.25**	17.60	**0.31**
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	33.40	2.87	1.12	37.40	3.12	1.18	19.80	2.92	1.11	**35.20**	2.63	0.89	31.45	2.89
ZeroSearch	23.00	0.52	0.67	27.80	1.01	1.08	8.80	0.54	0.69	33.60	0.34	0.52	23.30	0.60
PragmaticSearch	**40.00**	0.52	0.52	**38.60**	1.31	1.18	**20.20**	1.00	0.48	**35.20**	1.00	0.89	**33.50**	0.96
Qwen-2.5-3B *(Instruct)*														
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	**0.50**	0.65	21.80	**0.68**	0.73	8.00	**0.54**	0.69	11.11	**0.44**	0.62	14.38	**0.54**
Search-o1	14.80	0.58	**0.45**	22.40	1.00	0.74	5.20	0.96	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	34.80	1.33	0.53	42.80	1.49	**0.57**	20.00	1.30	**0.53**	40.00	1.32	**0.48**	34.40	1.36
PragmaticSearch	**41.80**	1.32	0.80	**54.80**	1.78	0.98	**30.40**	1.58	0.74	**40.80**	0.78	0.78	**41.95**	1.36
LLaMA-3.2-3B *(Instruct)*														
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	0.95	0.82	14.80	1.18	0.87	7.20	0.95	0.86	19.44	0.81	0.70	15.01	0.97
Search-o1	19.40	1.14	0.80	17.40	1.22	0.95	6.00	0.87	0.66	32.00	1.11	0.94	18.70	1.08
Search-R1	21.20	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	**23.00**	1.06	0.83	22.00	1.31	0.95	10.60	1.18	0.84	28.80	0.98	0.78	21.10	1.13
PragmaticSearch	**23.00**	**0.00**	**0.00**	**30.40**	**0.00**	**0.00**	**12.40**	**0.23**	**0.49**	**34.40**	**0.49**	**0.50**	**25.05**	**0.18**

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.

Annealing Trigger Criterion	NQ	TriviaQA	PopQA
Valid action ratio = 90%	42.20	62.40	48.00
Valid action ratio = 99%	**43.80**	**63.20**	**65.00**

Table 9: Ablation on transition threshold. Triggering the annealing stage at a 99% valid action ratio yields robust optimal performance across datasets.

cause the gate is conditioned on normalized, group-relative advantages (A_i^+) rather than raw absolute rewards, the mechanism inherently auto-scales, making our reward coefficients highly robust across different domains without the need for manual tuning.

Method	NQ	TriviaQA	PopQA
PragmaticSearch (without gate)	30.90	41.25	50.00
***PragmaticSearch* (with gate)**	**42.20**	**48.00**	**62.40**

Table 10: Ablation on the Soft Performance Gate. The presence of the dynamic gate significantly improves accuracy compared to standard advantage shaping.

G Case Study Analysis

We contrast the trajectories of *PragmaticSearch* and ZeroSearch in Table 11 to highlight two critical advantages of our framework:

1. Efficiency: Avoiding Redundant Searches. In Case 1, *PragmaticSearch* achieves the correct answer with **zero search calls**, whereas ZeroSearch triggers two unnecessary retrievals. This demonstrates that *PragmaticSearch* effectively calibrates its internal knowledge, recognizing when a question can be solved via internal reasoning, thus significantly reducing redundant search API calls and associated financial costs.

2. Quality: Deep Reasoning without Hallucinations. In Case 2, *PragmaticSearch* shows that its cost-awareness does not lead to a "lazy" policy. Instead, it executes **four deliberate searches** to resolve a complex temporal constraint. Unlike ZeroSearch, which exhibits factual hallucinations (e.g., misattributing the author of *The Color Purple*), *PragmaticSearch* maintains high logical rigor—iteratively switching between director and actor hypotheses until it identifies the definitive evidence. Even when both models reach the same answer, *PragmaticSearch* provides a high-fidelity reasoning trace free from the substantial factual errors seen in the baseline.

G.1 Artifact Licenses and Intended Use

We use existing scientific artifacts only for research training, evaluation, and comparison purposes, following their original intended use where specified. The ZeroSearch dataset and search-simulation setting are used for training search-capable question-answering agents, while the public QA benchmarks are used only for question-answering evaluation. Pretrained Qwen2.5 and LLaMA-3.2 models are used as backbone models under their respective model licenses and access terms. The ZeroSearch-provided `Simulation_LLM_google_14B_V2` artifact is used only as the training-time search simulator. Baseline methods, including Standard RAG, Search-o1, Search-R1, and ZeroSearch, are used only for research comparison. The verl and vLLM frameworks are used as software infrastructure under their original open-source licenses. Google Search access through SerpAPI is used only for test-time retrieval evaluation under the service's access terms. We do not redistribute third-party datasets, model weights, or API outputs; released code and checkpoints will require users to obtain any external artifacts under the corresponding original licenses and terms.

G.2 Artifact Documentation

We document the artifacts used in our experiments throughout Appendix C. The training artifact is the ZeroSearch dataset, used for training search-capable question-answering agents. The evaluation artifacts are seven English question-answering benchmarks, covering both single-hop factual questions and multi-hop reasoning questions; their domains and task characteristics are listed in Appendix D.1. The model artifacts are Qwen2.5 and LLaMA-3.2 backbones, for which we report model family, scale, and Base/Instruct variants. Following ZeroSearch, we use its released `Simulation_LLM_google_14B_V2` artifact to generate training-time simulated Google Search results. The baseline artifacts include Direct Answer, CoT, Standard RAG, Search-o1, Search-R1, and ZeroSearch; their roles are documented in the baseline description. The software and service artifacts include verl for reinforcement-learning training, vLLM for serving the search simulator, Weights & Biases for experiment logging, and SerpAPI-backed Google Search for test-time retrieval, with their roles and configurations described above. We do not use demographic labels or demographic-group annotations in training or evaluation, and the benchmarks are not selected to represent demographic groups.

G.3 Package and Implementation Parameters

We report the main package-level implementation parameters used in our experiments. Training is implemented with `verl` using the GRPO advantage estimator, FSDP training, gradient checkpointing, remove-padding optimization, and FSDP parameter, gradient, and optimizer offloading for the actor. The search simulator is served with `vLLM` using the fine-tuned `Simulation_LLM_google_14B_V2` model. In our launch scripts, the simulator uses tensor parallelism over 2–4 GPUs, 80–90% GPU memory utilization, `max_model_len`=2,048, and `max_num_seqs`=128–256, depending on the run. For policy rollouts, `verl` uses the `vLLM` rollout backend with rollout tensor parallel size 1, five search agents (`n_agent`=5), sampling temperature $T = 1.0$, top-$p = 0.95$, top-$k = -1$, maximum prompt length 4,096 tokens, maximum response length 512 tokens at every tool-calling iteration, maximum start length 2,048 tokens, maximum observation length 2,048 tokens, and at most 5 search turns. Optimization uses learning rate 1×10^{-6} with a 95% warm-up ratio under the configured constant schedule, KL loss coefficient $\beta = 0.001$ with `low_var_kl`, entropy coefficient 0.001, clip ratio 0.2, and gradient clipping at 1.0. The default PS-GRPO script sets PPO mini-batch size 256 and PPO micro-batch size 64. The training and validation batch sizes are 64 or 128. Checkpointing is every 50 steps in the rebuttal launch scripts that set `TRAIN_SAVE_FREQ=50`; otherwise the default in `train_grpo.sh` is 100. Validation is every 100 steps by default. Evaluation reports Exact Match as the primary metric, while training rewards use F1 score. During training, search is enabled with `search_mode=simulate_prompt`, start and end thresholds set to 0, and the simulator temperature set to 0.8 for search-document generation. For Google-based test-time retrieval, the SerpAPI branch requests the top 5 organic results and formats each result using its title and snippet; observations are truncated to at most 2,048 tokens.

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**	**ZeroSearch**: Wrong **Search Count: 2**
<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> **[Decision Node: Pruning Search]** <answer> 65 </answer>	<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**	**ZeroSearch**: Correct **Search Count: 2**
<think> To answer this, I need to find who was in both films and had a gap. Let's check the director first. </think> **[Decision Node: Executing Search]** <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> **[Decision Node: Executing Search]** <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> **[Decision Node: Executing Search]** <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> **[Decision Node: Pruning Search]** <answer> Oprah Winfrey </answer>	<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 11: 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.