SAAS: Self-Aware Reinforcement Learning for Over-Search Mitigation in Agentic Search

Anonymous ACL submission

Abstract

Agentic search enables LLMs to solve complex multi-hop questions through iterative reasoning and external search. Despite the effectiveness, these systems often suffer from a critical limitation in practice: agents fail to recognize their own knowledge boundaries, blindly triggering searches when internal knowledge suffices and failing to terminate search even when adequate evidence has been collected. The lack of self-awareness leads to severe **over-search**, incurring substantial inference latency and prohibitive computational cost. To this end, we propose SAAS, a novel RL framework designed to cultivate dynamic self-awareness that precisely regulates search behavior without compromising accuracy. SAAS introduces three key components: (i) a search boundary modeling mechanism, which identifies the search boundary under the evolving policy by contrasting search-disabled and search-enabled rollouts; (ii) a boundary-aware reward module, which translates this boundary awareness into trajectory-level penalties, suppressing unnecessary and redundant searches; and (iii) a stage-wise optimization strategy, which leverages a sequential curriculum to prioritize reasoning over search regularization, thereby avoiding reward hacking. Extensive experiments demonstrate that SAAS substantially reduces over-search, while maintaining accuracy. Our code is anonymously released at `https://anonymous.4open.science/r/SAAS-50B0`.

1 Introduction

Large language models (LLMs) (Guo et al., 2025; Yang et al., 2025; Jaech et al., 2024) exhibit strong reasoning capabilities across complex tasks, but their reliance on static parametric knowledge can lead to hallucinations in knowledge-intensive settings (Ji et al., 2023; Kandpal et al., 2023). Agentic search addresses this limitation by coupling reasoning with iterative retrieval, allowing the model to decide when and what to search (Singh et al., 2026; Li et al., 2025a; Zhang et al., 2026a). Through this reasoning-retrieval cycle, the model decomposes complex questions, retrieves external evidence, and integrates it into subsequent reasoning.

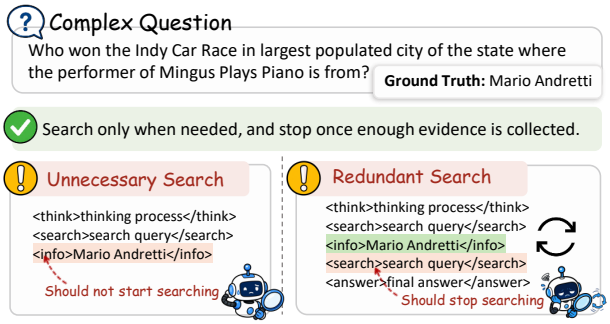

Figure 1: Illustration of two types of over-search in agentic search. `Unnecessary search` denotes triggering search despite sufficient parametric knowledge, while `redundant search` denotes continuing search after sufficient external evidence has already been collected.

Despite its strengths, agentic search often suffers from critical over-search problem: As shown in Figure 1, the model may trigger unnecessary searches or continue searching even after sufficient evidence is collected (Wu et al., 2025a, 2026). The former increases unnecessary dependence on external tools, while the latter leads to redundant multi-turn retrieval. Both cases increase computation cost and inference latency, and may introduce noisy evidence that distracts the final answer.

Existing work on over-search can be broadly categorized into two classes: (i) Prompt-based methods that regulate search behavior through prompting strategies or external routing mechanisms without modifying model parameters. Typically, DRAGIN (Su et al., 2024) optimizes query formulation to prevent irrelevant exploration, while Adaptive-RAG (Jeong et al., 2024) employs a trained lightweight classifier to assess query complexity and dynamically route requests to the suitable retrieval strategy (ii) RL-based methods which apply reinforcement learning to constrain search

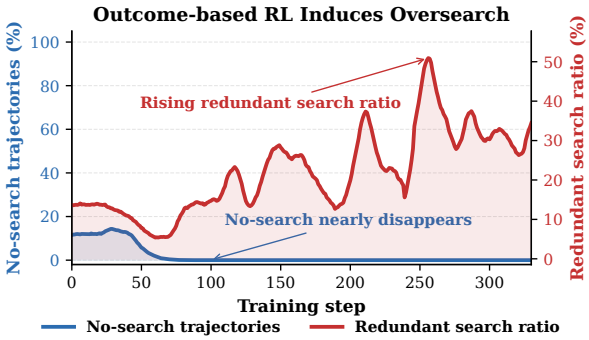

Figure 2: Outcome-based RL induces over-search during training. The ratio of no-search trajectories (blue) quickly drops to nearly zero, while the redundant search ratio (red) continues to rise, indicating that RL encourages frequent search and increases redundant search.

depth and restrict excessive tool usage. Typically, StepSearch (Wang et al., 2025b) mitigates redundant exploration through planning, decomposing complex question answering into step-wise reasoning and training the model to dynamically schedule search actions at each step, while HiPRAG (Wu et al., 2026) assigns fine-grained process rewards based on the informativeness of each retrieved passage, penalizing unnecessary search steps.

However, both paradigms share a common limitation: they rely on static heuristics or fixed thresholds that fail to capture the agent's evolving capability, making them vulnerable to reward hacking, where agents learn to game the penalty rather than develop genuine search boundary awareness. Specifically, existing methods suffer from three critical limitations in practice: (i) outcome-only rewards provide no signal on search awareness, leaving the agent blind to when to initiate and terminate search properly; (ii) naively adding search penalties triggers reward hacking, as static penalties cannot distinguish disciplined restraint from lazy guessing; and (iii) the agent's knowledge distribution changes throughout training, resulting in dynamic shifts in search boundary, yet existing methods provide no mechanism to track these shifts. Without this dynamic self-awareness, the agent is prone to severe over-search, incurring substantial inference latency and prohibitive computational cost.

To this end, we propose SAAS, a **S**elf-**A**ware Reinforcement learning framework for **A**gentic **S**earch, designed to reduce over-search without compromising answer accuracy. It aims to address two fundamental questions: (i) how to dynamically model agent's search boundary as its capability evolves, and (ii) how to translate this awareness into trajectory-level penalties for RL optimization, suppressing unnecessary search without triggering reward hacking. Specifically, SAAS consists of three key components: (i) a search boundary modeling mechanism that contrasts search-disabled and search-enabled rollouts to track the evolving knowledge boundary; (ii) a boundary-aware reward module that translates this boundary awareness into trajectory-level adaptive penalties, suppressing unnecessary search without triggering reward hacking; (iii) a stage-wise optimization strategy that prioritizes deep exploration before search regularization to further stabilize training. Our major contributions are listed as follows:

- We identify the key limitation of existing agentic search models and propose SAAS, a novel reinforcement learning framework that dynamically models the agent's evolving search boundary to suppress unnecessary and redundant searches.
- SAAS establishes search boundary awareness over the agent's evolving capability and leverages this awareness to penalizes over-search behavior, combined with a stage-wise policy optimization to prevent reward hacking.
- Extensive experiments on seven benchmarks show that SAAS consistently mitigate over-search without compromising model accuracy.

2 Preliminary Analysis

In this study, we present a comprehensive analysis of the over-search problem in agentic search from the perspective of optimization dynamics. First, we analyze how standard outcome-based RL optimization drives excessive reliance on external evidence (§ 2.1). We then investigate whether naive search constraints can mitigate this issue and find that fixed penalties fail to reliably define a proper search boundary under an evolving policy (§ 2.2). Finally, we discuss the empirical results in § 2.3 and conclude that *a boundary-aware training framework is needed to overcome this over-search issue*.

2.1 Over-search Issue in Agentic Search

We first examine how over-search emerges in agentic search. Specifically, we train an agentic search model from scratch with the commonly used outcome-based reward. As shown in Figure 2, the model becomes increasingly reliant on search as training proceeds. Since external search often improves final-answer accuracy on complex questions, outcome-based optimization encourages search use

but provides no signal about whether a search is necessary or still useful. Specifically, we observe two forms of over-search:

Question-level over-search. At the question level, the ratio of no-search trajectories steadily decreases during training and becomes nearly zero by step 50. This indicates that outcome-based reward optimization quickly encourages the model to rely on search, making search as its default behavior. Even when internal parametric knowledge is sufficient, the model still performs unnecessary searches. *This suggests that the model fails to learn when search is actually needed.*

Step-level redundancy. While question-level over-search concerns whether a question requires search, step-level redundancy concerns whether the model can stop searching during reasoning. We measure this using the redundant search ratio, defined as the proportion of searches issued after sufficient evidence has been searched. As shown in Figure 2, this ratio rises throughout training and eventually approaches about 50%, indicating that the model increasingly continues searching even when additional search is no longer needed. These extra searches add little useful information while increasing inference cost and latency. *This suggests that the model fails to judge evidence sufficiency, leading to substantial redundant searches.*

2.2 Challenges in Learning Search Boundary

The above observations show that outcome-based rewards can induce severe over-search. A straightforward remedy is to impose explicit constraints on search actions, such as directly penalize search calls beyond a predefined threshold. We therefore further conduct a preliminary study on whether such strategy can equip the model with reliable search boundary awareness. Our analysis shows that this remains challenging: not only does the agent's internal knowledge distribution evolve during training, leading to a shifting search boundary, but fixed constraints may also induce reward hacking by discouraging necessary searches. The results are summarized as follows.

Search boundary shifts with the model's evolving capability. Figure 3 shows that the ratio of questions answerable without search steadily increases from 12.7% at step 100 to 24.3% at step 300. This trend indicates that the search boundary is not an intrinsic property of a question, but is conditioned on the model's current capability. As training progresses, questions that initially depend on external search may become solvable with parametric knowledge alone. Therefore, predefined search constraints can become misaligned with the evolving policy, providing unstable or even misleading optimization signals.

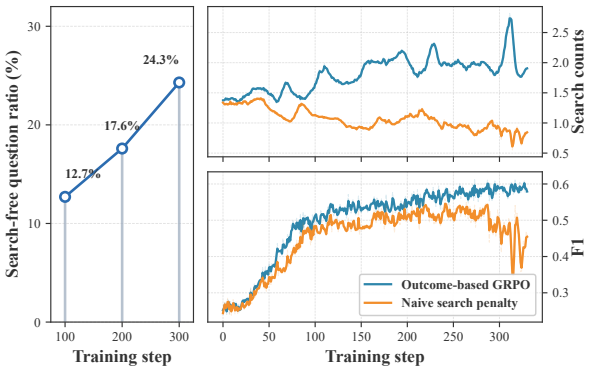

Figure 3: Limitations of naive search penalization. As training progresses, more questions become answerable without search, indicating a shifting search boundary. A naive search penalty reduces search actions but degrades performance, causing late-stage optimization collapse.

Fixed penalty can induce reward hacking. As shown in Figure 3, applying a fixed penalty to search calls yields lower accuracy than outcome-based training, with training collapse around step 250. The sharp decline in search calls further suggests that the performance drop mainly stems from excessive search suppression. This indicates that naive search penalization does not simply remove redundant searches, but can instead distort the optimization objective. Under a uniform penalty, the model may avoid necessary searches regardless of whether external evidence is still required, resulting in reward hacking and degraded task performance.

2.3 Discussion

Overall, these preliminary results reveal two limitations of vanilla RL. First, naive search penalties provide no explicit signal for modeling the agent's search boundary, making it hard to decide when search is necessary or existing evidence is sufficient. Second, fixed constraints struggle to balance search efficiency and task performance, as uniform penalties may suppress necessary searches and destabilize optimization. These findings motivate a framework that dynamically models search boundaries for RL optimization to reduce over-search without compromising accuracy.

3 Method

In this section, we present SAAS, a training framework that reduces over-search by teaching a search-

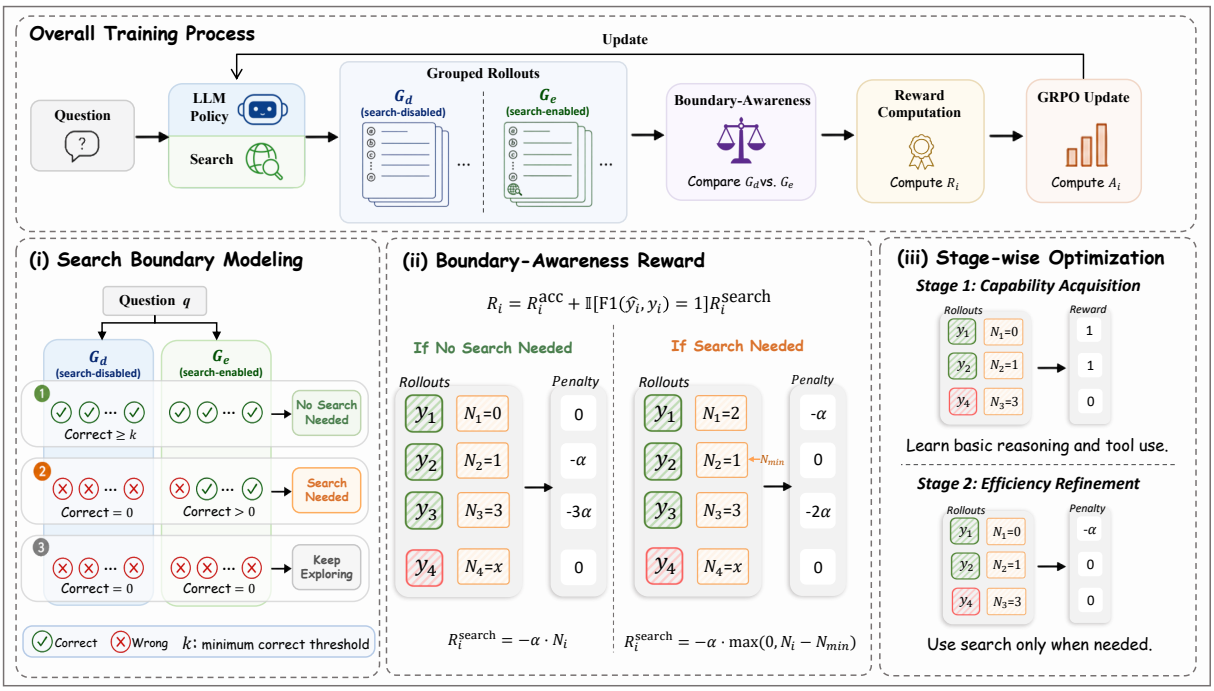

Figure 4: The Overall Pipeline of SAAS. SAAS reduces over-search by training a agent to recognize when search is needed and when further search should stop, organizing optimization around search boundary awareness. It consists of three components: (i) **Search Boundary Modeling** that tracks evolving search boundary by contrasting search-disabled and search-enabled rollouts; (ii) **Boundary-aware Reward** that translates this boundary awareness into trajectory-level penalties; (iii) **Stage-wise Optimization** which leverages a sequential curriculum to prioritize deep exploration over search regularization, thereby preventing reward hacking caused by excessive search suppression.

augmented agent both when to search and when to terminate further searches. Specifically, SAAS consists of three key components as shown in Figure 4: (i) **Search boundary modeling**, which contrasts search-disabled and search-enabled rollouts under the evolving policy to identify each question's search boundary; (ii) **Boundary-aware reward**, which translates this boundary awareness into trajectory-level penalties, guiding the agent to discriminatively suppress unnecessary and redundant search steps; and (iii) **Stage-wise optimization**, which leverages a sequential curriculum to prioritize reasoning over search regularization, thereby preventing reward hacking caused by excessive search suppression.

3.1 Search Boundary Modeling

For each question, the agent's search boundary is dynamic: as the policy improves during RL training, questions that initially require external evidence may later become solvable using the model's parametric knowledge alone. Motivated by this observation, SAAS models the search boundary under the evolving policy rather than relying on static annotations or predefined heuristic rules.

Specifically, SAAS identifies the search boundary for each question q by comparing a search-disabled rollout group $G_{\text{d}}(q)$ with a search-enabled rollout group $G_{\text{e}}(q)$ under the evolving policy:

$$\begin{aligned} G_{\text{d}}(q) &= \{\tau_i \sim \pi_\theta(\cdot \mid q)\}_{i=1}^{N_d}, \\ G_{\text{e}}(q) &= \{\tau_i \sim \pi_\theta(\cdot \mid q, \mathcal{C})\}_{i=1}^{N_e}. \end{aligned} \tag{1}$$

where $\mathcal{C}$ denotes the external evidence searched from the knowledge base during rollout phase. In the search-disabled group G_{d}, the agent is restricted to reasoning solely with its own parametric knowledge, whereas in the search-enabled group G_{e}, it is allowed to interact with the search engine to acquire additional evidence for reasoning. We then compare the two groups by evaluating the success number of trajectories that produce correct final answers. This comparison reveals whether q lies within the search boundary of the evolving policy. Formally:

$$\begin{aligned} n_{\text{d}}(q) &= \sum_{\tau \in G_{\text{d}}(q)} \mathbb{I}[r_{\text{ans}}(\tau) = 1], \\ n_{\text{e}}(q) &= \sum_{\tau \in G_{\text{e}}(q)} \mathbb{I}[r_{\text{ans}}(\tau) = 1], \end{aligned} \tag{2}$$

where $n_{\text{d}}(q)$ and $n_{\text{e}}(q)$ denote the numbers of correct trajectories in the search-disabled and search-enabled groups. $r_{\text{ans}}(\tau)$ indicates whether trajec-

tory τ produces the correct final answer. We then define the search boundary of this question as:

$$\mathcal{S}(q) = \begin{cases} \textsc{NoSearch}, & n_{\mathrm{d}}(q) \geq \delta, \\ \textsc{NeedSearch}, & n_{\mathrm{d}}(q) = 0,\ n_{\mathrm{e}}(q) > 0, \\ \textsc{Undetermined}, & \text{otherwise}. \end{cases} \tag{3}$$

where δ is a threshold that determines whether parametric knowledge is enough to answer the question.

This categorization determines how search behavior should be regulated for q. NoSearch indicates that the current policy can reliably solve q without external evidence, while NeedSearch indicates questions that require further searches for successful reasoning. Undetermined covers cases where grouped rollouts provide insufficient evidence to determine the current search boundary.

3.2 Boundary-Aware Reward

We next translate the identified search boundary into trajectory-level rewards. Because a uniform penalty over all searches fails to account for the varying search reliance across questions, we design discriminative, boundary-aware rewards that selectively penalize unnecessary and redundant search.

For trajectory τ_i, we define the total reward as:

$$R_i = R_i^{\mathrm{acc}} + \mathbb{I}[\mathrm{F1}(\hat{y}_i, y_i) = 1] R_i^{\mathrm{search}}, \tag{4}$$

where R_i^{acc} measures answer quality and R_i^{search} guides search behavior. The indicator function $\mathbb{I}$ ensures that the search reward is activated only when the trajectory produces a fully correct answer. We use the F1 score as the accuracy reward:

$$R_i^{\mathrm{acc}} = \mathrm{F1}(\hat{y}_i, y_i), \tag{5}$$

where $\hat{y}_i$ and y_i denote the predicted and golden answers, respectively. Compared with binary correctness metrics, the F1 score provides a smoother accuracy signal. This indicator further prevents excessive suppression of tool use before the model has learned to leverage search effectively.

We instantiate the search reward R_i^{search} according to the boundary-aware category $\mathcal{S}(q)$. Let N_i denote the number of search actions in trajectory τ_i. For NoSearch questions, we apply a zero-tolerance penalty:

$$R_i^{\mathrm{search}} = -\alpha N_i. \tag{6}$$

For NeedSearch questions, search itself should not be penalized. Instead, to target only redundant searches, we introduce $N_{\min}$ as the minimum sufficient search count, estimated as the fewest search actions among correct trajectories in the search-enabled rollout group:

$$N_{\min} = \min_{\tau_j \in G_{\mathrm{ena}},\, r_a(\tau_j)=1} N_j, \tag{7}$$

where $r_a(\tau_j)$ indicates whether τ_j produces the correct final answer. We then penalize only the search actions that exceed $N_{\min}$:

$$R_i^{\mathrm{search}} = -\alpha \max(0, N_i - N_{\min}). \tag{8}$$

For Undetermined questions, the boundary is unclear, so we impose no additional search constraints to avoid restricting reasoning and tool use.

To handle the heterogeneous reward scales between search-disabled and search-enabled rollout groups, we further employ group-wise advantage normalization in GRPO: within each group, advantages are computed and normalized independently. This avoids gradient contamination from distributional mismatch and greatly stabilizes training.

3.3 Stage-Wise Optimization

Although the boundary-aware reward regulates search behavior, activating it too early may induce reward hacking, where the agent avoids necessary searches before acquiring sufficient tool-use competence. We therefore adopt a stage-wise optimization strategy to protect early tool-use learning.

Specifically, the optimization is divided into two stages: capability acquisition and efficiency refinement. In Stage I, the agent is trained only with the outcome-based reward, so it can first learn basic reasoning and tool use without search constraints. When validation performance stops improving, we switch to Stage II and activate the boundary-aware reward to reduce unnecessary and redundant search while preserving answer quality. The trajectory reward is defined as:

$$R_i = \begin{cases} R_i^{\mathrm{acc}}, & \text{Stage I}, \\ R_i^{\mathrm{acc}} + \mathbb{I}[\mathrm{F1}(\hat{y}_i, y_i) = 1] R_i^{\mathrm{search}}, & \text{Stage II}. \end{cases} \tag{9}$$

In essence, stage-wise optimization leverages a sequential curriculum to prioritize deep exploration over search regularization, ensuring that the agent first masters when and how to search before it is taught when to refrain. This sequential curriculum mitigates reward hacking and yields a stable training procedure that reduces over-search without compromising agent performance.

Method	TriviaQA		PopQA		NQ		HotpotQA		2wiki.		Musique		Bamboogle		AVG	
	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC
Qwen2.5-3B-Instruct																
Direct Inference	44.9	-	14.4	-	21.6	-	23.6	-	22.0	-	6.1	-	28.8	-	23.1	-
RFT (Ahn et al., 2024)	59.8	0.89	42.3	1.81	40.6	1.20	45.0	1.65	38.6	1.70	17.0	2.76	40.8	1.40	40.6	1.63
Search-R1 (Jin et al., 2025)	65.7	1.22	42.4	1.23	45.0	1.20	43.6	1.53	36.5	1.76	15.2	1.79	40.8	1.53	41.2	1.47
StepSearch (Wang et al., 2025b)	60.0	1.44	39.6	1.37	41.2	1.37	45.0	1.83	41.1	2.04	16.8	2.10	27.2	1.70	38.7	1.69
HiPRAG (Wu et al., 2026)	68.8	1.59	43.8	1.63	**52.5**	1.39	48.6	1.96	38.0	2.22	17.0	2.37	36.8	2.08	43.6	1.89
Ours	**69.2**	**0.66**	**45.1**	**1.01**	43.6	**0.72**	**52.9**	**1.16**	**43.9**	**1.44**	**20.9**	**1.62**	**44.8**	**1.23**	**45.8**	**1.13**
Qwen2.5-7B-Instruct																
Direct Inference	55.7	-	16.9	-	28.2	-	28.3	-	25.7	-	8.8	-	40.0	-	29.1	-
RFT (Ahn et al., 2024)	67.5	0.92	47.3	1.46	46.2	1.19	50.1	1.64	43.2	1.75	23.1	2.50	42.4	1.36	45.7	1.56
Search-R1 (Jin et al., 2025)	68.3	1.11	44.9	1.26	45.6	1.24	45.8	1.19	38.7	1.43	16.8	1.38	40.0	1.15	42.9	1.25
StepSearch (Wang et al., 2025b)	67.8	1.28	43.6	1.17	47.7	1.17	53.2	1.80	45.1	2.27	**26.2**	2.27	**49.6**	1.84	47.6	1.69
HiPRAG (Wu et al., 2026)	**74.0**	2.04	46.8	2.03	**56.0**	1.96	**57.3**	2.16	43.5	2.39	23.8	2.55	47.2	2.23	**49.8**	2.19
Ours	**74.0**	**0.56**	**47.4**	**1.01**	47.8	**0.74**	53.6	**0.96**	**45.9**	**1.18**	22.6	**1.30**	**49.6**	**1.02**	48.7	**0.97**

Table 1: Performance comparison on seven QA benchmarks. ACC denotes answer accuracy (%), and SC denotes average search count per question. Best results are in **bold**, and second-best results are underlined.

4 Experiments

In this section, we aim to answer the following questions. **Q1** (Main Results): How does SAAS compare with existing baselines in terms of answer accuracy and search efficiency? **Q2** (Model Analysis): How does SAAS reduce question-level and step-level over-search, and how do accuracy and search count evolve across training stages? **Q3** (Ablation Study): How does each component affect accuracy and search efficiency? **Q4** (Hyperparameter Analysis): How do key hyperparameters affect the performance of SAAS? Please refer to Case Study (**Q5**) and Efficiency Analysis (**Q6**) in Appendix B.

4.1 Experimental Setup

Datasets. We evaluate SAAS on seven open-domain QA benchmarks, covering both single-hop and multi-hop scenarios. The single-hop setting includes TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023) and NQ (Kwiatkowski et al., 2019). The multi-hop setting includes HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). More details on datasets are provided in Appendix C.1.

Baselines. We compare SAAS against a comprehensive set of baselines that represent different paradigms in agentic search: (1) **Direct Inference**: direct generation without any search mechanism. (2) **Rejection Sampling Fine Tuning (RFT)** (Ahn et al., 2024): fine-tuning the model on trajectories generated through rejection sampling. (3) **RL-Based Agentic Search**: use reinforcement learning to train agentic search capacity, including Search-R1 (Jin et al., 2025), StepSearch (Wang et al., 2025b), and HiPRAG (Wu et al., 2026). Please refer to more details in Appendix C.2.

Evaluation Metrics. We evaluate each method from two aspects: answer quality and search behavior. For answer quality, we report **Accuracy (Acc)**, measured by an LLM judge following prior work (Chen et al., 2025). For search behavior, we report **Search Count (SC)**, **Question-level Over-search Ratio (QOR)**, and **Step-level Over-search Ratio (SOR)**. SC measures the average number of search calls, QOR measures the ratio of unnecessary searches on questions answerable from parametric knowledge, while SOR measures the ratio of redundant search actions. Detailed metric definitions are provided in Appendix C.3.

Implementation Details. We construct the training corpus by combining the training splits of NQ and HotpotQA, and employ E5 (Wang et al., 2024) as the dense encoder to embed queries for evidence search over the 2018 Wikipedia dump (Karpukhin et al., 2020). We use GPT-4 as the LLM judge for answer evaluation. Various LLMs serve as backbone models for the experiment. More implementation details are included in Appendix B.1.

4.2 Main Results (Q1)

To address Q1, we evaluate SAAS with direct inference, RFT, and other RL-based agentic search baselines across seven open-domain benchmarks. According to Table 1 which reports accuracy and

Method	TriviaQA		PopQA		NQ		HotpotQA		2wiki.		Musique		Bamboogle		AVG	
	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR
RFT (Ahn et al., 2024)	22.8	40.9	7.6	59.3	15.6	61.2	12.6	**40.0**	5.8	42.0	12.7	79.7	26.5	44.4	14.8	52.5
Search-R1 (Jin et al., 2025)	37.8	91.2	12.6	94.2	24.1	97.9	14.8	81.2	4.8	75.5	12.7	**75.0**	25.2	64.3	18.9	82.8
StepSearch (Wang et al., 2025b)	37.2	99.8	15.5	99.9	26.8	99.8	20.0	99.9	6.8	100.0	19.9	100.0	43.7	100.0	24.3	99.9
HiPRAG (Wu et al., 2026)	30.8	100.0	12.4	100.0	21.2	100.0	15.6	100.0	4.2	100.0	15.1	100.0	36.9	100.0	19.5	100.0
Ours	**11.5**	**29.4**	**2.1**	**45.8**	**8.7**	**52.6**	**4.1**	40.7	**1.7**	**36.9**	**2.7**	78.0	**13.3**	**38.1**	**6.3**	**45.9**

Table 2: Over-search analysis on Qwen2.5-7B-Instruct. SOR and QOR denote step-level and question-level over-search ratios (%), respectively. A lower SOR indicates that the model performs fewer redundant searches, while a lower QOR indicates that the model better leverages its parametric knowledge to avoid unnecessary search.

search count on Qwen2.5 series model, we summarize the key observations below.

SAAS consistently outperforms baselines. SAAS achieves the best average accuracy on Qwen2.5-3B-Instruct (45.8%), surpassing the strongest baseline HiPRAG by 2.2%, and remains competitive on Qwen2.5-7B-Instruct (48.7%). The gain is especially clear on multi-hop tasks. Specifically, SAAS improves over HiPRAG by 8.0% on Bamboogle. This suggests that modeling search boundaries does not compromise reasoning quality, but helps focus search on questions where external evidence is truly needed.

SAAS substantially reduces redundant search. SAAS consistently uses fewer searches than baselines, reducing the average search count to 1.13 on Qwen2.5-3B-Instruct and 0.97 on Qwen2.5-7B-Instruct. In contrast, strong baselines such as StepSearch and HiPRAG require 1.69 and 2.19 searches on average, respectively. This shows that SAAS avoids default search behavior while preserving necessary retrieval for reasoning.

SAAS achieves a better accuracy-efficiency trade-off. SAAS achieves a favorable balance between answer accuracy and search cost. Specifically, on Qwen2.5-3B-Instruct, it improves accuracy over the strongest baseline from 43.6% to 45.8% while using 40.2% fewer search calls. This indicates that SAAS does not simply reduce search frequency, but learns a clearer search boundary that better aligns retrieval with actual evidence needs.

4.3 Model Analysis (Q2)

4.3.1 Over-search Behavior Analysis

To verify whether SAAS directly mitigates over-search, we further measure two over-search-related metrics: QOR and SOR. QOR captures whether the model triggers search on questions that can be answered without search, while SOR captures whether it issues redundant searches during the reasoning process. The results on Qwen2.5-7B-Instruct are reported in Table 2.

Figure 5: Training dynamics of F1 and search count across the two-stage RL training process.

SAAS effectively reduces question-level over-search. As shown in Table 2, SAAS achieves the lowest average QOR (45.9%), clearly lower than RFT (52.5%) and much lower than search-heavy baselines such as StepSearch (99.9%) and HiPRAG (100.0%). This shows that SAAS better identifies when the model's parametric knowledge is sufficient and avoids unnecessary searches.

SAAS also suppresses step-level redundant search. SAAS achieves the lowest SOR on every benchmark, reducing the average SOR to 6.3%, far below RFT (14.8%) and StepSearch (24.3%). The reduction is also clear on multi-hop datasets, such as 2WikiMultiHopQA, where SOR drops to 1.7%. These results show that SAAS learns to stop once sufficient external evidence has been collected, encouraged by the $N_{\min}$-based reward that penalizes searches beyond the sufficient-evidence boundary.

4.3.2 Training Dynamics

To better understand how our Boundary-Awareness Guided Reward affects search-augmented agent training, we track the training dynamics of average F1 and search count across two stages on Qwen2.5-3B-Instruct. The results are shown in Figure 5.

Figure 5 shows two distinct training stages. In Stage I, before the boundary-aware reward is introduced, both average F1 and search count increase, indicating that the model first learns to leverage the search tool to improve answer quality. In Stage II,

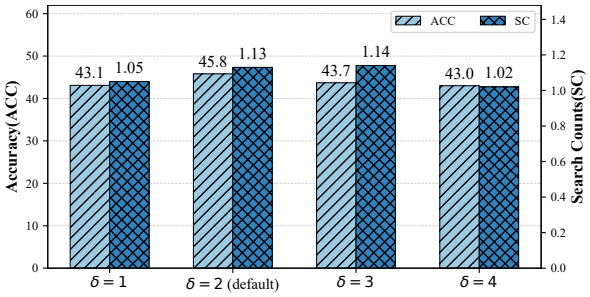

Figure 6: Sensitivity analysis of the key hyperparameter δ, the number of GRPO groups used for evidence-demand estimation. ACC and RC are averaged across seven QA benchmarks on Qwen2.5-3B-Instruct.

after activating the boundary-aware reward, the average search count drops sharply from about 2.0 to below 1.0, while F1 shows only a mild temporary decrease and then remains stable. Unlike the naive penalty setting in Section 2.2, this transition does not cause training collapse. Instead, SAAS recalibrates the search boundary, reducing over-search behavior while preserving answer quality.

4.4 Ablation Study (Q3)

To analyze the contribution of each component in SAAS, we conduct ablation studies on Qwen2.5-3B-Instruct. As shown in Table 3, SAAS achieves the best balance between accuracy and search count, indicating that stage-wise optimization and on-policy boundary modeling jointly contribute to stable and boundary-aware search behavior.

Stage-wise optimization promotes stable search learning. As shown in Table 3, removing stage-wise optimization further reduces the average search count, but substantially lowers accuracy from 45.8% to 40.9%. This suggests that optimizing for search efficiency from the beginning can over-constrain the policy before it acquires reliable tool-use behavior. In contrast, stage-wise optimization first develops the agent's tool-use capability and then refines search efficiency, yielding a better accuracy-efficiency trade-off.

On-policy boundary modeling better matches the evolving policy. Without on-policy boundary modeling, the search boundary is derived from the base model before training and then kept fixed during optimization. This variant achieves only 42.8% accuracy with 1.07 search count, below the full method. This shows that the search boundary shifts as the model improves. A boundary fixed before training can become misaligned with the current policy, while on-policy boundary modeling keeps the search-regulation signal actively updated.

Method Variant	ACC	SC
SAAS	45.8	1.13
w/o Stage-wise optimization	40.9	0.95
w/o On-policy estimation	42.8	1.07

Table 3: Ablation study on SAAS components using Qwen2.5-3B-Instruct as the backbone language model.

4.5 Parameter Sensitivity Analysis (Q4)

We further conduct a sensitivity analysis on the thershold hyperparameter δ used in search boundary modeling. Specifically, we vary δ on Qwen2.5-3B-Instruct and evaluate the resulting average answer accuracy and search count across the seven QA benchmarks. This experiment examines whether the default setting provides a stable search-boundary modeling for reward assignment. More detailed analysis are provided in Appendix B.4.

The sensitivity analysis confirms that $\delta = 2$ provides the most reliable search-boundary signal for downstream policy optimization. Under this setting, SAAS achieves the highest average ACC of 45.8% with a low average SC of 1.13, indicating a favorable accuracy-efficiency balance. Reducing the threshold to $\delta = 1$ slightly lowers SC to 1.05, but ACC drops to 43.1%, showing that the lower search count comes from over-suppressing search behavior rather than yielding more reliable search-boundary awareness. Increasing the threshold to $\delta = 3$ or $\delta = 4$ also degrades ACC to 43.7% and 43.0%, respectively, suggesting that stricter grouping introduces less stable boundary modeling and weakens the reward signal for necessary evidence acquisition. To summarize, overly aggressive compression of search actions leads to accuracy degradation, whereas $\delta = 2$ preserves sufficient evidence acquisition while discouraging unnecessary and redundant search steps.

5 Conclusion

In this paper, we introduce SAAS, a self-aware reinforcement learning framework for mitigating over-search in agentic search. SAAS models the evolving search boundary of the RL policy and integrates boundary-aware rewards with stage-wise optimization to regulate search behavior. Experiments across 7 open-domain benchmarks show that SAAS effectively reduces both question-level and step-level over-search while maintaining strong performance. These results underscore the importance of dynamic search boundary awareness for more efficient and effective search-augmented reasoning.

Limitation

While SAAS effectively regulates search behavior within text-based agentic systems, our current evaluation focuses on unimodal textual retrieval. In practice, knowledge-intensive tasks may also involve multimodal evidence, such as images, tables, and structured databases. Although textual serialization can capture the essential semantics of these sources, directly incorporating raw multimodal signals may provide richer contextual information. Importantly, the core formulation of SAAS is inherently agnostic to input modalities, as its fundamental mechanism of modeling the boundary of required information to prevent over-search does not rely on text-specific features. Consequently, generalizing this framework to multimodal reasoning constitutes a natural progression for future research. Such an extension would enable the model to perform cross-modal reasoning, grounding textual information in visual, structural, or other multimodal evidence, thereby moving toward more comprehensive search-augmented reasoning systems.

Ethics Statement

We confirm that this study follows the Ethics Policy. All data and models used are publicly available and contain no private user information.

Data and Model Bias Our experiments use publicly available open-domain QA benchmarks and a Wikipedia-based corpus. We do not collect private user data. These datasets are well-established and widely adopted in the research community, with no known systematic biases that would affect the validity of our conclusions. The pre-trained language models used in our experiments are standard off-the-shelf models, while our method, SARS, focuses solely on regulating search behavior based on the model's evolving knowledge boundary, and does not rely on or amplify any sensitive attributes.

Intended Use SAAS is designed as a research framework to support researchers and developers in studying efficient search-augmented reasoning. By reducing redundant search, SAAS aims to improve the deployability of search-augmented agents in practical scenarios where inference latency and retrieval cost are important considerations.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *Preprint*, arXiv:2402.00157.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

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

Alan Dao and Thinh Le. 2025. Rezero: Enhancing llm search ability by trying one-more-time. *Preprint*, arXiv:2504.11001.

Kaustubh D. Dhole. 2025. To retrieve or not to retrieve? uncertainty detection for dynamic retrieval augmented generation. *Preprint*, arXiv:2501.09292.

Yuchen Fan, Kaiyan Zhang, Heng Zhou, Yuxin Zuo, Yanxu Chen, Yu Fu, Xinwei Long, Xuekai Zhu, Che Jiang, Yuchen Zhang, Li Kang, Gang Chen, Cheng Huang, Zhizhou He, Bingning Wang, Lei Bai, Ning Ding, and Bowen Zhou. 2025. Ssrl: Self-search reinforcement learning. *Preprint*, arXiv:2508.10874.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *Preprint*, arXiv:2504.11536.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Ziyang Huang, Xiaowei Yuan, Yiming Ju, Jun Zhao, and Kang Liu. 2025. Reinforced internal-external knowledge synergistic reasoning for efficient adaptive search agent. *Preprint*, arXiv:2505.07596.

Liu Huanshuo, Hao Zhang, Zhijiang Guo, Jing Wang, Kuicai Dong, Xiangyang Li, Yi Quan Lee, Cong Zhang, and Yong Liu. 2025. Ctrla: Adaptive retrieval-augmented generation via inherent control. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12592–12618.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar,

Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

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

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025a. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *Preprint*, arXiv:2503.00223.

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

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. *Preprint*, arXiv:2305.06983.

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

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

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International conference on machine learning*, pages 15696–15707. PMLR.

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

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.

Jian Li, Xiaoxi Li, Yan Zheng, Yizhang Jin, Shuo Wang, Jiafu Wu, Yabiao Wang, Chengjie Wang, and Xiaotong Yuan. 2025a. A survey on ai search with large language models.

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

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025c. Webthinker: Empowering large reasoning models with deep research capability. *Preprint*, arXiv:2504.21776.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025d. Torl: Scaling tool-integrated rl. *Preprint*, arXiv:2503.23383.

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

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

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *Preprint*, arXiv:2504.13958.

Zeyang Sha, Shiwen Cui, and Weiqiang Wang. 2025. Sem: Reinforcement learning for search-efficient large language models. *Preprint*, arXiv:2505.07903.

Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V. Vasilakos. 2026. Agentic retrieval-augmented generation: A survey on agentic rag. *Preprint*, arXiv:2501.09136.

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

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025b. R1-searcher++: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning. *Preprint*, arXiv:2505.17005.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12991–13013.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2026a. Zerosearch: Incentivize the search capability of llms without searching. *Preprint*, arXiv:2505.04588.

Jingbo Sun, Wenyue Chong, Songjun Tu, Qichao Zhang, Yaocheng Zhang, Jiajun Chai, Xiaohan Wang, Wei Lin, Guojun Yin, and Dongbin Zhao. 2026b. Autosearch: Adaptive search depth for efficient agentic rag via reinforcement learning. *Preprint*, arXiv:2604.17337.

Zhongxiang Sun, Qipeng Wang, Weijie Yu, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Yang Song, and Han Li. 2025. Rearter: Retrieval-augmented reasoning with trustworthy process rewarding. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1251–1261.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Preprint*, arXiv:2108.00573.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *Preprint*, arXiv:2212.10509.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. 2025a. Chain-of-retrieval augmented generation. *Preprint*, arXiv:2501.14342.

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

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

Peilin Wu, Mian Zhang, Kun Wan, Wentian Zhao, Kaiyu He, Xinya Du, and Zhiyu Chen. 2026. Hiprag: Hierarchical process rewards for efficient agentic retrieval augmented generation. *Preprint*, arXiv:2510.07794.

Peilin Wu, Mian Zhang, Xinlu Zhang, Xinya Du, and Zhiyu Zoey Chen. 2025a. Search wisely: Mitigating sub-optimal agentic searches by reducing uncertainty. *Preprint*, arXiv:2505.17281.

Weiqi Wu, Xin Guan, Shen Huang, Yong Jiang, Pengjun Xie, Fei Huang, Jiuxin Cao, Hai Zhao, and Jingren Zhou. 2025b. Masksearch: A universal pre-training framework to enhance agentic search capability. *Preprint*, arXiv:2505.20285.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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

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

Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Liu Weichuan, Lei Hou, and Juanzi Li. 2025. Seakr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27022–27043.

Chenlu Ye, Zhou Yu, Ziji Zhang, Hao Chen, Narayanan Sadagopan, Jing Huang, Tong Zhang, and Anurag Beniwal. 2025. Beyond correctness: Harmonizing process and outcome rewards through rl training. *Preprint*, arXiv:2509.03403.

Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Guojun Yin, and Wei Lin. 2025. Promoting efficient reasoning with verifiable stepwise reward. *Preprint*, arXiv:2508.10293.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita-Velez, Yue Liao, Hongru Wang, and 6 others. 2026a. The landscape of agentic reinforcement learning for llms: A survey. *Preprint*, arXiv:2509.02547.

Ningning Zhang, Chi Zhang, Zhizhong Tan, Xingxing Yang, Weiping Deng, and Wenyong Wang. 2026b. Credible plan-driven rag method for multi-hop question answering. *Preprint*, arXiv:2504.16787.

Zhen Zhang, Xinyu Wang, Yong Jiang, Zile Qiao, Zhuo Chen, Guangyu Li, Feiteng Mu, Mengting Hu, Pengjun Xie, and Fei Huang. 2025. Kbm: Delineating knowledge boundary for adaptive retrieval in large language models. *Preprint*, arXiv:2411.06207.

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

Hanna Zubkova, Ji-Hoon Park, and Seong-Whan Lee. 2025. Sugar: Leveraging contextual confidence for smarter retrieval. *Preprint*, arXiv:2501.04899.

A Frequently Asked Questions (FAQs)

A.1 Code and Data Availability

To facilitate future research and allow independent verification of our results, all code and data associated with SAAS are released via an anonymous repository: https://anonymous.4open.science/r/SAAS-50B0. This repository provides a comprehensive experimental framework. Specifically, it details our exact methodology, including the rollout generation process, reward calculations, and the precise stage-wise reinforcement learning setups. Coupled with the release of all processed datasets, standardized prompts, search interface templates, and full evaluation scripts, this repository grants researchers unrestricted access to every component required to reproduce our work.

A.2 What are the advantages of SAAS?

SAAS presents several advantages over existing agentic search methods by explicitly modeling the agent's evolving search boundary. This search boundary awareness improves search efficiency while preserving the ability to acquire external evidence when search is genuinely needed:

Reduction of unnecessary search. Many questions can be answered using the model's parametric knowledge alone, yet RL-based search agents trained with outcome-only rewards may still trigger search because unnecessary tool use is not explicitly penalized. SAAS addresses this issue by contrasting search-disabled and search-enabled rollouts to identify low-reliance questions where the model's parametric knowledge is sufficient. For these questions, search actions are penalized, encouraging the model to avoid unnecessary search at the question level.

Suppression of redundant search. For questions that require external evidence, the key challenge is not only whether to search, but also when to stop searching. Outcome-only rewards can encourage the model to continue issuing search calls even after sufficient external evidence has been collected. SAAS uses the minimum successful search trajectory as a reference and penalizes searches beyond this sufficient-evidence boundary. As a result, it suppresses redundant search without discouraging evidence acquisition when search is necessary.

Stable accuracy-efficiency trade-off. SAAS does not simply minimize search count. Instead, it adopts stage-wise optimization to first develop basic reasoning and tool-use capability, and then applies boundary-aware reward assignment to refine search behavior. This design reduces search count while maintaining competitive answer accuracy across single-hop and multi-hop benchmarks as well as different model backbones.

A.3 Why not simply penalize every search action?

A uniform penalty on all search actions is too coarse because the usefulness of search actions depends on the question and the current policy. For questions that can be answered using the model's parametric knowledge alone, any search action is unnecessary and should be discouraged. In contrast, for questions that require external evidence under the current policy, search is useful until sufficient evidence has been collected, after which additional search calls become redundant. Treating these cases identically can therefore suppress necessary evidence acquisition and harm answer accuracy, while still providing no guidance on when the model should initiate or terminate search. SAAS addresses this limitation through boundary-aware reward assignment based on the estimated search necessity of the current policy. When a question does not require search, SAAS penalizes search actions to reduce question-level over-search. When search is required, SAAS allows evidence acquisition but penalizes searches beyond the minimum successful search trajectory, thereby suppressing step-level redundant search. This design reduces both unnecessary and redundant searches while preserving tool-use ability when external evidence is genuinely needed.

A.4 Why is the search boundary modeled on-policy?

The search boundary is policy-dependent and evolves throughout RL training. As the model improves, questions that initially require external evidence may become solvable using parametric knowledge alone. For questions that still require search, the boundary for terminating search can also shift, since the policy may learn to acquire sufficient evidence with fewer search actions. A boundary derived only from the base model or from fixed heuristic rules can therefore become misaligned with the current policy. SAAS avoids this mismatch by modeling the search boundary on-policy: it contrasts search-disabled and search-enabled rollout groups from the current model to determine whether search should be initiated and

Method	TriviaQA		PopQA		NQ		HotpotQA		2wiki.		Musique		Bamboogle		AVG	
	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC	ACC	SC
Qwen2.5-3B-Instruct																
GRPO	69.3	1.87	46.3	1.91	46.4	1.86	53.7	2.30	45.4	2.63	21.8	2.88	42.4	2.34	46.5	2.26
Ours	69.2	0.66	45.1	1.01	43.6	0.72	52.9	1.16	43.9	1.44	20.9	1.62	44.8	1.23	45.8	1.13
Qwen2.5-7B-Instruct																
GRPO	73.9	2.26	48.0	2.40	48.3	2.32	57.6	3.02	49.2	3.86	25.3	3.63	48.8	3.10	50.2	2.94
Ours	74.0	0.56	47.4	1.01	47.8	0.74	53.6	0.96	45.9	1.18	22.6	1.30	49.6	1.02	48.7	0.97
Qwen3-4B-Instruct																
GRPO	74.6	1.70	49.4	1.65	49.5	1.70	61.7	2.37	50.2	2.89	29.0	2.83	60.8	2.35	53.6	2.21
Ours	73.6	0.69	49.0	1.19	49.6	0.91	58.2	1.25	53.3	1.79	27.3	1.72	58.4	1.20	52.8	1.25

Table 4: Efficiency Analysis. Accuracy and search count comparison between outcome-based GRPO and SAAS across three backbones. ACC denotes answer accuracy (%), and SC denotes average search count per question.

how much search is sufficient. This keeps the search-regulation signal updated during training and enables boundary-aware reward assignment to suppress both question-level unnecessary search and step-level redundant search. The ablation without on-policy boundary modeling performs worse than the full method, further confirming the importance of tracking the evolving search boundary.

A.5 How does SAAS differ from prior efficient agentic search methods?

Prior efficient agentic search methods typically regulate search behavior through heuristic triggers, confidence signals, search-depth constraints, or process rewards. Although these mechanisms can reduce search frequency, they do not explicitly align search decisions with the model's evolving search boundary: whether the current policy can answer from parametric knowledge alone, and whether the acquired external evidence is already sufficient. Unlike these approaches, SAAS dynamically tracks the agent's actual competence during training. By anchoring the reward mechanism directly to the policy's evolving state, it discriminatively evaluates the necessity of search for each specific question. This ensures that the agent is penalized for over-search only when search initiation is strictly unnecessary or when the gathered evidence is already sufficient, fundamentally differing from the static constraints used in previous methods.

B Additional Experiments

B.1 SAAS Setup

During training, we use grouped rollouts to estimate the search label $\mathcal{S}(q)$ and compute the search reward. For each question, the policy samples $N = 8$ trajectories, which are evenly split into two groups: four search-disabled rollouts and four search-enabled rollouts. The search-disabled group estimates whether the question can be solved from the model's parametric knowledge, while the search-enabled group estimates whether search can help solve difficult questions. Following the search-label definition in the method section, we set the grouping threshold to $\delta = 2$: a question is labeled as NoSearch if at least δ search-disabled rollouts answer it correctly, and as NeedSearch if no search-disabled rollout succeeds but at least one search-enabled rollout is correct.

For SAAS-specific reward settings, we set the search penalty coefficient in R_i^{search} to $\alpha = 0.05$. For NoSearch questions, the reward penalizes every search call according to N_i; for NeedSearch questions, it penalizes only extra calls beyond $N_{\min}$, the minimum number of searches among correct search-enabled trajectories. During both training and evaluation, each query retrieves the top $k = 3$ documents. The maximum number of search calls in a trajectory is set to 5.

Meanwhile, we implement the GRPO algorithm based on the slime[1] framework. The detailed hyperparameter settings are listed in Table 6.

B.2 Case Study (Q5)

To qualitatively illustrate the over-search patterns discussed above, we provide two representative case studies in Figure 7 and Figure 8. These examples compare SAAS with the GRPO baseline trained only with correctness rewards. The first case focuses on unnecessary search, where the question can be directly answered from parametric

[1] https://github.com/THUDM/slime

Method	TriviaQA		PopQA		NQ		HotpotQA		2wiki.		Musique		Bamboogle		AVG	
	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR	SOR	QOR
Qwen2.5-3B-Instruct																
GRPO	26.4	100.0	10.4	100.0	16.5	100.0	11.6	100.0	5.0	100.0	11.1	100.0	28.5	100.0	15.6	100.0
Ours	21.2	41.6	6.5	61.8	10.5	46.2	8.6	52.2	4.4	73.7	6.6	82.9	15.7	65.6	10.5	60.6
Qwen2.5-7B-Instruct																
GRPO	30.1	100.0	12.5	100.0	18.4	100.0	12.1	100.0	3.7	100.0	8.1	100.0	22.6	100.0	15.4	100.0
Ours	11.5	29.4	2.1	45.8	8.7	52.6	4.1	40.7	1.7	36.9	2.7	78.0	13.3	38.1	6.3	45.9
Qwen3-4B-Instruct																
GRPO	25.4	100.0	11.4	100.0	16.5	100.0	10.2	100.0	2.7	100.0	6.7	100.0	20.4	100.0	13.3	100.0
Ours	10.5	27.4	3.6	47.1	8.3	44.2	4.8	39.5	2.2	65.4	4.5	56.9	8.8	52.1	6.1	47.5

Table 5: Over-search comparison between outcome-based GRPO and SAAS. SOR and QOR denote step-level and question-level over-search ratios, respectively. Lower values indicate fewer redundant and unnecessary searches.

Hyperparameter	Value
Learning Rate / Scheduler	1×10^{-6} / Constant
Warmup Ratio / Epochs	0.285 / 1
Batch Size / Global Batch Size	512 / 4096
Rollouts / Rollout Temp.	8 / 1.0
KL Coeff. / Clip Ratio (ϵ)	0.001 / 0.2
Max Prompt / Response Len.	512 / 512

Table 6: Training hyperparameter settings of SAAS.

knowledge but outcome-based optimization still encourages searches. The second case focuses on redundant search after search has already been triggered: both methods obtain the correct answer, but outcome-based GRPO continues issuing extra queries, whereas SAAS stops earlier once sufficient evidence has been collected.

B.3 Efficiency Analysis (Q6)

Table 4 illustrates the accuracy-efficiency trade-off of SAAS compared to a standard GRPO baseline across seven QA benchmarks and three model backbones. Effectiveness is quantified by answer accuracy, while efficiency is tracked via the average search count. To establish a rigorous comparison, the GRPO baseline is trained solely with an outcome-based F1 reward.

SAAS significantly reducing search overhead without compromising answer quality. On Qwen2.5-3B-Instruct, SAAS reduces the average RC from 2.26 to 1.13, corresponding to a 50.0% reduction in search count, while the average ACC decreases only slightly from 46.5% to 45.8%. The same trend becomes even more pronounced on Qwen2.5-7B-Instruct: SAAS lowers the average RC from 2.94 to 0.97, reducing by about 67.0%, while retaining a competitive average ACC of 48.7% compared with 50.2% for GRPO. On Qwen3-4B-Instruct, SAAS also reduces the average RC from 2.21 to 1.25, with only a 0.8% drop in average accuracy. These results indicate that the trade-off between model's performance and efficiency is not tied to a specific backbone, but generalizes across models.

Outcome-based GRPO introduces a severe search burden. As shown in Table 4, GRPO consistently incurs high average search count across all three backbones, reaching 2.26, 2.94, and 2.21 on Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen3-4B-Instruct, respectively. This observation is consistent with the motivation: when the reward only reflects final answer's correctness, search is treated as a uniformly beneficial action because additional information may increase the chance of matching the gold answer. As a result, GRPO lacks an explicit signal for distinguishing necessary information acquisition from redundant tool use, and thus tends to learn a conservative policy that repeatedly searches even after sufficient information has been obtained. In contrast, SAAS introduces search-necessity aware supervision to couple search decisions with the model's actual knowledge boundary. Search is encouraged when external evidence is needed, but unnecessary or redundant search actions are penalized. This mechanism explains why SAAS can substantially reduce search count while retaining comparable accuracy: it does not simply suppress search, but reallocates search actions to samples and reasoning stages where external evidence is most valuable.

B.4 Parameter Sensitivity Analysis

Finally, we analyze the sensitivity of the grouping hyperparameter δ used for GRPO-grouped search-

Question: Panda is a national animal of which country?
Gold Answer: China **Both correct; Search counts:** SAAS 0 vs. GRPO 2

SAAS: Direct Answering without Search

Think

<think> To determine which country the panda is a national animal of, I need to recall the common knowledge about pandas. Pandas are native to China. They are often associated with China and are considered a national symbol or animal emblem of China. </think>

Answer

<answer> China </answer>

Outcome-based GRPO: Unnecessary Search before Answering

Think

<think> To determine which country the panda is a national animal of, I need to first recall if I already have this information or if I need to search for it. </think>

Search 1

<search> panda is a national animal of which country. </search>
<information> Doc 1(Title: ``Giant panda") While the dragon has often served as China's national symbol ... Doc 2(Title: ``Giant pandas around the world") Giant pandas around the world As of 2018, there ... Doc 3(Title: ``Giant pandas around the world") These zoos ... </information>

Think

<think> The search results indicate that giant pandas are associated with China, but I need to confirm if they are the national animal of China. </think>

Search 2

<search> is the giant panda the national animal of China. </search>
<information> Doc 1(Title: ``Giant panda") While the dragon has often served as China's national symbol ... Doc 2(Title: ``Giant panda") Giant panda The giant panda (``Ailuropoda melanoleuca", literally … Doc 3(Title: ``Giant panda") triplets was announced in China; it was the ... </information>

Think

<think> The search results confirm that the giant panda is associated with China and is often used in international contexts, including as a national symbol. </think>

Answer

<answer> China </answer>

Table 7: Case study on unnecessary search.The question can be answered using parametric knowledge alone, yet outcome-based GRPO still triggers searches, whereas SAAS answers directly without search.

necessity estimation. Figure 6 reports the average ACC and RC under different values of δ on Qwen2.5-3B-Instruct, aiming to verify whether the default setting provides reliable necessity estimates for training.

The default setting $\delta = 2$ yields the most effective necessity estimation for downstream policy learning. With $\delta = 2$, SAAS achieves the highest average ACC of 45.8% while maintaining a low average RC of 1.13. When $\delta = 1$, the average RC remains comparable at 1.05, but the average ACC drops to 43.1%. This indicates that overly coarse grouping can still suppress searches, but it provides less accurate search-necessity estimates and therefore weakens the reward signal used to distinguish samples that require external information from those that do not.

Using more groups does not improve training effectiveness because finer grouping may introduce noisier necessity estimates. When $\delta = 3$ and $\delta = 4$, the average ACC decreases to 43.7% and 43.0%, respectively, while the average RC remains close to the default setting. These results suggest that increasing δ does not produce a more useful supervision signal; instead, excessively fine grouping may make the estimated evidence demand less stable, leading to less consistent search decisions during training. Overall, the sensitivity analysis

Question: Industrial city in Germany on the Rhine-Herne canal?
Gold Answer: Duisburg **Both correct; Search counts:** SAAS 1 vs. GRPO 4

SAAS: Direct Answering without Search

Think

<think> To find an industrial city in Germany on the Rhine-Herne Canal, I need to recall that the Rhine-Herne Canal connects the Rhine River to the North Sea. Major industrial cities along this canal would likely be in the region of North Rhine-Westphalia, which is known for its industrial heritage. </think>

Search 1

<search> industrial city in Germany on the Rhine-Herne canal. </search>
<information> Doc 1(Title: ``Rhine–Herne Canal") Rhine–Herne Canal The Rhine–Herne Canal () is a transportation … connects the harbour in Duisburg on the ... Doc 2(Title: ``Rhine–Herne Canal") Fish that inhabit ... Doc 3(Title: ``Rhine–Herne Canal") rail and river) centre working in ... </information>

Think

<think> Based on the search results, Duisburg is a major industrial city in Germany located on the Rhine-Herne Canal. </think>

Answer

<answer> Duisburg </answer>

Outcome-based GRPO: Unnecessary Search before Answering

Think

<think> To find an industrial city in Germany on the Rhine-Herne Canal, I need to first identify cities that are located on this canal. I'll start by searching for information about the Rhine-Herne Canal and the cities that are situated along it. </think>

Search 1

<search> industrial city in Germany on the Rhine-Herne Canal. </search>
<information> Doc 1(Title: ``Rhine–Herne Canal") Rhine–Herne Canal The Rhine–Herne Canal () is ... and connects the harbour in Duisburg on the ... Doc 2(Title: ``Rhine–Herne Canal") Fish that inhabit ... Doc 3(Title: ``Rhine–Herne Canal") rail and river) centre working in association with ... </information>

Think

<think> … Now, I need to identify an industrial city along this canal ... </think>

Search 2

<search> industrial cities in the Ruhr area of North Rhine-Westphalia, Germany. </search>
<information> Doc 1(Title: Ruhr) Ruhr The Ruhr (,) is a polycentric ... Doc 2(Title: Ruhr) connecting to Stadtbahn line U47 ... Doc 3(Title: Rhine-Ruhr) and urban areas within it often ... </information>

Think

<think>... I need to identify which of these cities is specifically located on the Rhine-Herne... </think>

Search 3

<search> cities on the Rhine-Herne Canal. </search>
<information> Doc 1(Title: ``Rhine–Herne Canal") Rhine–Herne Canal The Rhine–Herne ... Doc 2(Title: ``Rhine–Herne Canal") Fish that inhabit the ... Doc 3(Title: ``Rhine–Herne Canal") and ... </information>

Think

<think> … Let me search for information about Duisburg's industrial status... </think>

Search 4

<search> industrial status of Duisburg, Germany. </search>
<information> Doc 1(Title: ``Duisburg") has campuses in ... Doc 2(Title: ``Duisburg") Duisburg () is a city in ... Doc 3(Title: ``Landschaftspark Duisburg-Nord") Landschaftspark ... </information>

Think

<think> … Duisburg is an industrial city located on the Rhine-Herne Canal. Therefore … </think>

Answer

<answer> Duisburg </answer>

Table 8: Case study on redundant search behavior. Both methods answer correctly, but outcome-based GRPO repeatedly issues additional queries before producing the final answer, whereas SAAS reaches the answer with substantially fewer search calls.

supports the choice of $\delta = 2$, which provides the best balance between accurate necessity estimation and stable policy optimization.

These efficiency gains are particularly pronounced on tasks prone to excessive searches. Multi-hop benchmarks such as 2WikiMultihopQA and MuSiQue require the model to gather information across multiple reasoning steps, which creates more opportunities for unnecessary tool calls once the relevant information has already been obtained. Under the GRPO baseline, the outcome-based F1 reward does not distinguish necessary information acquisition from redundant searches, so the model tends to keep querying in these complex settings, reaching 3.86 and 3.63 search count on 2WikiMultihopQA and MuSiQue on Qwen2.5-7B-Instruct. By explicitly regularizing search necessity, SAAS learns to stop searching when additional information provides limited marginal value, reducing the corresponding search count to 1.18 and 1.30.

B.5 Over-search Behavior Compared with Outcome-Based GRPO

Table 5 further analyzes the over-search behavior of SAAS and the outcome-based GRPO baseline. We report SOR and QOR to evaluate two complementary aspects of search overhead: SOR measures redundant search actions after retrieval has been initiated, while QOR measures unnecessary search on questions that can be answered from the model's parametric knowledge.

SAAS substantially reduces unnecessary searches across model backbones. On Qwen2.5-3B-Instruct, GRPO yields an average QOR of 100.0%, indicating that it triggers search for all questions answerable from the model's parametric knowledge. In contrast, SAAS reduces the average QOR to 60.6%, corresponding to a 39.4% reduction in question-level over-search. The improvement becomes more pronounced on Qwen2.5-7B-Instruct, where the average QOR decreases from 100.0% to 45.9%. On Qwen3-4B-Instruct, GRPO still exhibits a QOR of 100.0 across all datasets, whereas SAAS reduces the average QOR to 47.5%. These results show that SAAS consistently suppresses unnecessary search actions, rather than simply reducing search on a specific backbone.

SAAS also mitigates redundant search after search has been triggered. Across all three backbones, SAAS consistently lowers the average SOR compared with GRPO. On Qwen2.5-3B-Instruct, the average SOR decreases from 15.6% to 10.5%. On Qwen2.5-7B-Instruct, the reduction is more substantial, from 15.4% to 6.3%, with particularly large improvements on TriviaQA, PopQA, HotpotQA, and MuSiQue. The same trend also holds for Qwen3-4B-Instruct, where the average SOR drops from 13.3% to 6.1%. These results indicate that the proposed method not only learns whether searches should be invoked, but also improves the model's ability to terminate search once sufficient information has been collected.

The reduction in both QOR and SOR demonstrates that SAAS addresses over-search at different decision levels. Question-level improvements show that the model avoids initiating search when external information is unnecessary, while step-level improvements indicate fewer redundant searches after search begins. This behavior is consistent with the design of SAAS, which encourages search only when it is expected to provide meaningful additional information. Consequently, SAAS achieves more targeted tool use and alleviates the over-search tendency induced by outcome-only optimization.

C Implementation Details

C.1 Benchmark Dataset

To comprehensively evaluate SAAS across questions with different degrees of reliance on search, we conduct experiments on seven open-domain QA benchmarks. The single-hop setting includes TriviaQA, PopQA and Natural Questions (NQ), which generally have lower search boundary because many questions focus on one factual relation and can often be answered using the model's parametric knowledge. In contrast, the multi-hop setting includes HotpotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle, where questions require the model to gather evidence across reasoning steps, making them more search-dependent and often requiring multiple search actions to answer correctly.

TriviaQA (Joshi et al., 2017): TriviaQA is built from trivia questions collected from online sources. For each question, it normalizes the reference answers and gathers related evidence documents from Wikipedia and the web, so that the answer can be checked against external textual evidence.

PopQA (Mallen et al., 2023): PopQA is built from Wikidata triples by converting relations such as occupations, birthplaces, or authorship into natural-language questions. The object in each Wikidata

LLM-as-a-Judge Evaluation

You are an evaluation assistant. Please determine if the model output is equivalent to the labeled answer.
Question: {question}
Labeled Answer: {labeled answer}
Model Output: {pred answer}
Did the model give an answer equivalent to the labeled answer? Please respond with "Correct" if they are equivalent, or "Incorrect" if they are not equivalent. Do not include any other text.

SOR Semantic Consistency Evaluation

You are an expert in Semantic Analysis. Evaluate whether Response 1 and Response 2 provide the same core answer to the given Search Query.
Output `True` if their answers are semantically consistent, otherwise output `False`.
<Inputs>
Search Query: {query}
Response 1: {response1}
Response 2: {response2}
</Inputs>
Provide your output as a single boolean value strictly enclosed in tags.
Example: `<answer>True</answer>`

Search-Enabled QA Prompt

Answer the given question. You must conduct reasoning inside `<think>` and `</think>` first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by `<search>query </search>` and it will return the top searched results between `<information>` and `</information>`. You can search as many times as your want. If you find no further external knowledge needed, you can directly provide the answer inside `<answer>` and `</answer>`, without detailed illustrations. For example, `<answer>Beijing </answer>`. Question: {question}

Search-Disabled QA Prompt

Answer the given question. You must conduct reasoning inside `<think>` and `</think>` first. After reasoning, you can directly provide the answer inside `<answer>` and `</answer>`, without detailed illustrations. For example, `<answer>Beijing </answer>`. Question: {question}

Table 9: Prompt templates used for answer evaluation, SOR semantic-consistency evaluation, search-enabled and search-disabled question answering throughout our experiments.

triple is used as the short answer, producing factual questions that test whether a model knows specific entity-relation facts.

Natural Questions (NQ) (Kwiatkowski et al., 2019): NQ comprises real-world anonymized queries issued to the Google search engine. The dataset pairs each query with a corresponding Wikipedia page, and provides human annotations for both long-answer passages and short answer spans.

HotpotQA (Yang et al., 2018): HotpotQA is built from Wikipedia and asks questions that require evidence from multiple pages. Examples are annotated with supporting sentences, making it suitable for evaluating if a model can retrieve and combine evidence for bridge or comparison questions.

2WikiMultiHopQA (Ho et al., 2020): 2WikiMultiHopQA is constructed from Wikidata relations and linked Wikipedia passages. It uses predefined reasoning patterns to generate multi-hop questions and provides annotations for intermediate entities, evidence documents, and the reasoning path needed to reach the answer.

MuSiQue (Trivedi et al., 2022): MuSiQue is built by composing connected single-hop questions into harder multi-hop questions. This construction reduces shortcut solutions, since the final answer depends on following a sequence of reasoning steps supported by paragraph-level evidence.

Bamboogle (Press et al., 2023): Bamboogle is a manually written set of compositional questions designed to require multiple independent facts. Its questions are created to be difficult to answer from a single clue, so models usually need to search for and combine evidence across steps.

C.2 Baseline Details

In this section, we provide detailed descriptions of each baseline used in our comparison.

Direct Inference directly prompts the base model to answer each question without the search engine. This baseline evaluates the model's ability to answer from parametric knowledge alone and serves as the reference for measuring the benefit and cost of external search.

Rejection Sampling Fine Tuning (Ahn et al., 2024) fine-tunes the model on trajectories selected through rejection sampling. For each training question, the model generates one search-enabled trajectory and one search-disabled trajectory; only trajectories that lead to correct answers are retained for fine-tuning. This baseline tests whether learning from successful sampled trajectories is sufficient to acquire useful search behavior without reinforcement learning.

Search-R1 (Jin et al., 2025) trains LLMs to reason and interact with a search engine through reinforcement learning. During generation, the model can issue search queries, read retrieved evidence, and continue reasoning before producing the final answer. Its training objective mainly rewards final answer correctness, making it a representative outcome-based RL baseline for agentic search.

StepSearch (Wang et al., 2025b) improves search efficiency from a planning perspective. It decomposes complex question answering into stepwise reasoning and search decisions, and uses proximal policy optimization to train the model to plan when and what to retrieve at each step. By coordinating searches with the evolving reasoning process, StepSearch serves as a planning-based baseline for evaluating whether better search scheduling can reduce inefficient searches.

HiPRAG (Wu et al., 2026) introduces hierarchical process rewards for efficient agentic search-augmented generation. Instead of relying only on final answer correctness, it provides reward signals over search actions and reasoning steps, helping the model improve answer quality and search efficiency. We include HiPRAG as a strong process-reward baseline for comparison with SAAS.

C.3 Evaluation Metrics

To comprehensively evaluate each method, we employ four metrics from two aspects: answer quality and search behavior. Answer quality is evaluated using Accuracy (ACC), while search behavior is evaluated using Search Count (SC), Question-level Over-search Ratio (QOR), and Step-level Over-search Ratio (SOR). All prompts used for evaluation are provided in Appendix E.

Accuracy (ACC) measures whether the final answer is correct. Following prior work (Chen et al., 2025; Li et al., 2025b), we use an LLM judge to determine whether the model output is equivalent to the ground truth. Given a test set $D = \{q_i\}_{i=1}^{|D|}$, ACC is calculated as:

$$\mathrm{ACC} = \frac{1}{|D|} \sum_{i=1}^{|D|} \mathbb{I}\big(\mathrm{Judge}(\hat{y}_i, y_i) = 1\big), \quad (10)$$

where $\hat{y}_i$ denotes the model prediction for question q_i, y_i denotes the ground truth, and $\mathbb{I}(\cdot)$ is the indicator function.

Search Count (SC) measures the average number of search calls among all trajectories. Let τ_i denote the generated trajectory for question q_i, and let N_i denote the number of search calls in τ_i. SC is defined as:

$$\mathrm{SC} = \frac{1}{|D|} \sum_{i=1}^{|D|} N_i. \quad (11)$$

A lower SC indicates that the model uses fewer search calls on average.

Question-level Over-search Ratio (QOR) measures unnecessary search at the question level. We define $D_{\text{para}} = \{q_i\}_{i=1}^{|D_{\text{para}}|}$ as the subset of questions that can be correctly answered using the model's parametric knowledge without search. For

each question $q_i \in D_{\text{para}}$, $o_i = 1$ if the generated trajectory τ_i still contains search actions. QOR is calculated as:

$$\text{QOR} = \frac{\sum_{i=1}^{|D_{\text{para}}|} o_i}{|D_{\text{para}}|}. \quad (12)$$

A lower QOR indicates that the model better leverages its parametric knowledge to avoid unnecessary search.

Step-level Over-search Ratio (SOR) measures redundant search at the step level. Following (Wu et al., 2026), for each question $q_i \in D$, let $\mathcal{S}_{\tau_i}$ denote the set of search queries in the generated trajectory τ_i. For each search query $s \in \mathcal{S}_{\tau_i}$, $o_s = 1$ if the model's answer to s is semantically consistent with the retrieved documents, indicating that this search step is redundant. SOR is defined as:

$$\text{SOR} = \frac{\sum_{i=1}^{|D|} \sum_{s \in \mathcal{S}_{\tau_i}} o_s}{\sum_{i=1}^{|D|} |\mathcal{S}_{\tau_i}|}. \quad (13)$$

A lower SOR indicates that the model performs fewer redundant search steps.

D Related Work

D.1 Agentic Search

Moving beyond static retrieve-then-generate RAG, agentic search studies how models can dynamically interleave reasoning with external search (Singh et al., 2026; Li et al., 2025a; Zhang et al., 2026a). ReAct (Yao et al., 2023) introduced the reasoning and acting paradigm, enabling models to decide when to invoke tools. IRCoT (Trivedi et al., 2023) adapted this paradigm to more complex question answering by interleaving retrieval with chain-of-thought reasoning, so that intermediate reasoning steps can be grounded in newly retrieved evidence. Along this line, Self-RAG (Asai et al., 2023) and FLARE (Jiang et al., 2023) introduced reflection tokens and confidence-driven retrieval triggers to make search decisions more adaptive during decoding. Chain-of-Retrieval (Wang et al., 2025a) and Search-o1 (Li et al., 2025b) further refine this process by supporting dynamic query formulation, while DeepResearcher (Zheng et al., 2025) and WebThinker (Li et al., 2025c) extend agentic search to live web environments that require browsing and evidence synthesis. More recently, reinforcement learning has been adopted to train tool-use policies (Qian et al., 2025; Feng et al., 2025; Li et al., 2025d). Representative systems optimize search-augmented reasoning through task-success rewards, encouraging models to invoke search and exploit retrieved evidence when it improves final answers (Jin et al., 2025; Chen et al., 2025; Song et al., 2025a; Dao and Le, 2025; Jiang et al., 2025a). To reduce the cost and instability of relying on live search engines during training, ZeroSearch (Sun et al., 2026a) and SSRL (Fan et al., 2025) simulate search environments, while MaskSearch (Wu et al., 2025b) uses search-augmented mask prediction pre-training to elicit generalizable search abilities. However, these outcome-driven approaches often underemphasize search efficiency, leaving models prone to over-search.

D.2 Efficient Agentic Search

While Agentic search improves complex reasoning, it often introduces unnecessary or redundant tool calls. Existing efforts can be broadly grouped into three directions. One direction improves planning ability to orchestrate when and what to query. DRAGIN (Su et al., 2024) dynamically schedules searches based on real-time information needs, while PAR-RAG (Zhang et al., 2026b) employs complexity-aware top-down planning to prevent irrelevant exploration and redundant retrieval. The second direction investigates retrieval necessity. Early studies (Dhole, 2025; Jeong et al., 2024) primarily depended on auxiliary classifiers or routers to evaluate query difficulty and guide search. In contrast, more recent methods (Zubkova et al., 2025; Yao et al., 2025; Huanshuo et al., 2025; Zhang et al., 2025) decode self-awareness signals from the LLM's internal hidden states or explicitly delineate the parametric knowledge boundary to decide when search is truly required. Beyond these directions, reinforcement learning directly optimizes tool-use efficiency. Several approaches incorporate efficiency metrics into rewards to penalize suboptimal tool usage. For example, SEM (Sha et al., 2025) trains intrinsically search-efficient models, Search Wisely (Wu et al., 2025a) mitigates unnecessary searches by reducing model uncertainty, and IKEA (Huang et al., 2025) introduces a knowledge-boundary aware reward to prioritize parametric knowledge and resort to search only when internal knowledge is insufficient. Other works focus on adaptive exploration. R1-Searcher++ (Song et al., 2025b) and AutoSearch (Sun et al., 2026b) use RL to incentivize dynamic knowledge acquisition and flexible search depths, while S3 (Jiang et al., 2025b) decouples the searcher from a frozen gener-

ator and trains it with a Gain-Beyond-RAG reward to improve downstream utility with limited data. Recent frameworks integrate process rewards for finer-grained supervision. ReARTeR (Sun et al., 2025) evaluates and refines individual steps using process rewards. HiPRAG (Wu et al., 2026) and parallel works on process rewards (Ye et al., 2025; Yue et al., 2025) introduce hierarchical supervision to constrain reasoning paths and curb over-searching. Despite these advances, these methods often fail to precisely align search actions with parametric knowledge, lacking a unified mechanism to trigger and halt search based on knowledge boundaries.

E Prompt Set

To facilitate reproducibility and provide transparency into the evaluation and agentic retrieval behaviors, we present the specific prompt sets used throughout our experiments in Table 9. These prompts instantiate the answer evaluator, the SOR semantic-consistency evaluator, the search-enabled and search-disabled question-answering settings. Together, they standardize the evaluation protocol and orchestrate the reasoning–search process described in the main text.

F The Use of Large Language Models

In preparing this paper, we made limited use of Large Language Models (LLMs). Specifically, LLMs were used to polish the writing, correct grammatical errors, and improve sentence clarity and readability without altering the scientific content. LLMs were also used to assist with minor formatting tasks, such as improving the presentation of tables and organizing related-work descriptions. All model-assisted suggestions were carefully reviewed and revised by the authors, who retain full responsibility for the scientific accuracy, technical claims, and integrity of the final manuscript.