AdaSearch: Balancing Parametric Knowledge and Search in Large Language Models via Reinforcement Learning

Anonymous ACL submission

Abstract

Equipping large language models (LLMs) with search engines via reinforcement learning (RL) promises effective search agents. However, adaptively balancing internal parametric knowledge with external search remains a challenge, as overreliance on search introduces unnecessary cost and risks exposure to noisy or malicious content, while relying solely on parametric knowledge risks hallucination. Prior efforts mitigate search overuse through tool-call reward shaping, which requires heavy reward engineering and conflates necessary and unnecessary search. To address these limitations, we revisit the evaluation of search agents through an F1-based decision metric, revealing that prior methods often overlook readily available parametric knowledge. Motivated by this, we propose AdaSearch, a simple two-stage, outcome-driven RL framework that disentangles problem-solving from the decision to search, making the decision process explicit and interpretable. Extensive experiments demonstrate that AdaSearch significantly improves search invocation decision-making and reduces unnecessary search calls while preserving strong task performance.

1 Introduction

Large language models (LLMs) have achieved significant advances across various tasks (Brown et al., 2020; Hendrycks et al., 2020; Team et al., 2023; Touvron et al., 2023; Wei et al., 2022; Guo et al., 2025). Yet they face inherent limitations: parametric knowledge alone cannot fully capture domain-specific (Lewis et al., 2020; Yang et al., 2024) or rapidly evolving information (Kasai et al., 2023; Vu et al., 2024), and models may hallucinate when queried beyond their knowledge boundary (Ji et al., 2023; Xu et al., 2024; Huang et al., 2025a; Kalai et al., 2025). To overcome these challenges, integrating LLMs with external knowledge sources has become a crucial direction (Karpukhin et al., 2020; Guu et al., 2020; Nakano et al., 2021; Lazaridou et al., 2022; Shi et al., 2024; Press et al., 2023; Nakano et al., 2021; Feng et al., 2024).

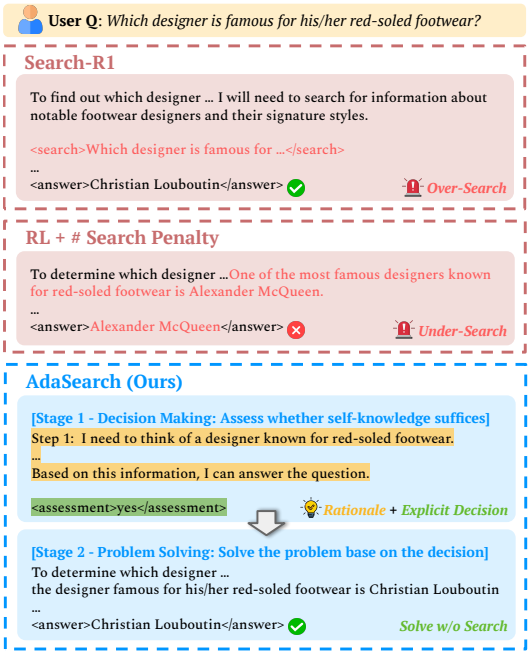


Figure 1: Comparing RL methods for search agents: AdaSearch provides transparent decisions via explicit reasoning. Search-R1 over-searches, while penalized RL under-searches; both lack explicit decision rationale.

Early work has primarily focused on retrieval-augmented generation (RAG) (Lewis et al., 2020; Borgeaud et al., 2022), which searches for relevant passages and appends them to the context before generation (Shuster et al., 2021; Ram et al., 2023; Jiang et al., 2023; Asai et al., 2024; Wei et al., 2024). However, the relatively static pipeline of RAG overlooks the model's reasoning process, yielding suboptimal results (Trivedi et al., 2023). Moreover, enabling dynamic multi-turn search is challenging due to the requirement of large-scale annotated trajectories and the non-differentiable nature of the search operation (Schick et al., 2023; Asai et al., 2024; Guan et al., 2025). To address these limitations, treating search as a tool and training tool-integrated LLMs via reinforce-

ment learning (RL) has emerged as a promising paradigm (Chen et al., 2025b; Jin et al., 2025; Fan et al., 2025; Xiao et al., 2026). In such settings, models are trained to act as search agents, interleaving reasoning with search in a multi-turn, interactive fashion. This allows LLMs to issue queries adaptively and integrate retrieved information into their reasoning, outperforming prior prompting-based and supervised approaches.

Nevertheless, building search agents that are truly adaptive remains challenging (Jeong et al., 2024; Qian et al., 2025; Xie et al., 2026). Ideally, a search agent should balance parametric knowledge and invoke search only when necessary. Excessive search calls compromise efficiency, safety, and privacy by increasing costs, exposing agents to noisy or malicious content, and risking sensitive data leaks (Franklin et al., 2026). Prior attempts (Huang et al., 2025b; Wang et al., 2025; Song et al., 2025b,c) mitigate over-search by tying rewards and evaluation to search-call counts. However, reward shaping around search-call counts requires careful engineering (Table 4), often involving extensive manual tuning and trial-and-error to determine suitable penalties for redundant search. Furthermore, reward credit assignment (Pignatelli et al., 2023) remains ambiguous: agents may superficially avoid search even when necessary to exploit rewards. Consequently, using call counts as a metric conflates necessary and redundant calls, obscuring true adaptive behavior.

Beyond efficiency, a key limitation of prior frameworks is the lack of *transparency and interpretability* in the decision to invoke search. Since existing methods provide no direct supervision for this decision, the reasoning behind "*Should I search?*" remains implicit and difficult for users to interpret. This limitation becomes critical in high-stakes domains such as finance and medical question answering, where users must understand why an agent chooses to rely on parametric knowledge or to consult external information.

In this work, we introduce **AdaSearch**, a simple outcome-based RL framework that disentangles and directly optimizes two abilities: *problem solving* and *deciding whether to invoke search*. Specifically, during training we adopt different prompts to guide LLMs in three modes: *(i)* solving the problem with parametric knowledge, *(ii)* solving the problem with search, and *(iii)* explicitly deciding whether search invocation is needed. At inference time, the model first decides whether the question can be solved using its own knowledge via reasoning; if so, the model answers the question directly, otherwise search calls will be invoked as part of the model's reasoning process (Figure 3).

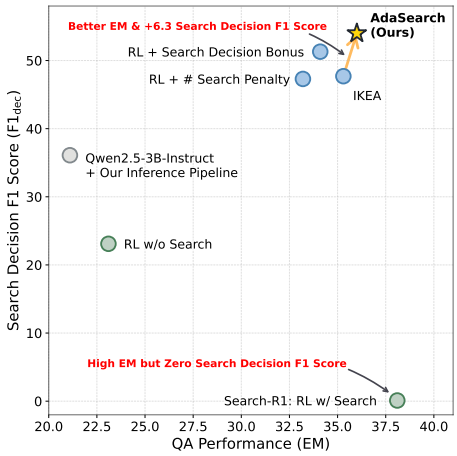


Figure 2: AdaSearch achieves the best search decision F1 score while preserving task performance. In contrast, Search-R1 achieves zero $\mathbf{F1}_{\text{dec}}$ due to its always-search behavior, and reward-shaping methods fail to maintain QA performance.

Unlike prior approaches that depend on brittle reward engineering, AdaSearch simply leverages task outcomes as rewards with a training framework that provides clearer learning signals, avoids the pitfalls of ambiguous tool-count penalties, and improves search invocation decision-making through explicit reasoning (Figure 1). Experiments and evaluations with our proposed fine-grained search decision F1 metric demonstrate that AdaSearch fosters better adaptivity, reducing unnecessary search calls while preserving strong task performance (Figure 2).

In summary, our contributions are three-fold:

- We propose **AdaSearch**, a simple yet effective RL framework that explicitly optimizes both problem solving and decision making to build adaptive search agents.
- AdaSearch eliminates the need for complex reward engineering while outperforming prior methods in both problem solving and search decision F1 score, and generalizes across different model sizes and families.
- AdaSearch improves interpretability and safety, enhancing the trustworthiness of agents in real-world scenarios. (Appendix C)

2 Related Work

Retrieval-augmented generation. Retrieval-augmented generation (RAG) has become a widely adopted paradigm for equipping LLMs with

external knowledge (Shuster et al., 2021; Ram et al., 2023; Jiang et al., 2023; Asai et al., 2024; Wei et al., 2024). Early approaches follow a static retrieve-and-read pipeline (Lewis et al., 2020; Guu et al., 2020; Izacard et al., 2023), which is effective for factoid queries but relatively limited for multi-step reasoning. Architectural variants such as RETRO (Borgeaud et al., 2022) deliver substantial gains but require model modifications and retraining, whereas in-context RAG (Ram et al., 2023) simply prepends retrieved passages to the prompt, enabling practical off-the-shelf use. More recent works emphasize reasoning-aware retrieval: IRCoT (Trivedi et al., 2023) interleaves CoT (Wei et al., 2022) with retrieval to improve multi-hop QA, Adaptive-RAG (Jeong et al., 2024) adapts retrieval strategies to query complexity, and DeepRAG (Guan et al., 2025) formulates retrieval as a step-wise decision process to balance parametric knowledge with external evidence. Our work shares a similar spirit but focuses on developing search agents via RL to decide whether and what to search through explicit reasoning.

Reinforcement learning for search agents. RL has emerged as a powerful paradigm for developing LLM-based search agents (Jin et al., 2025; Chen et al., 2025b; Song et al., 2025a). By treating search call as tools, LLMs are trained to interleave reasoning with search in a multi-turn, interactive fashion. Representative methods include Search-R1 and ReSearch (Jin et al., 2025; Chen et al., 2025b). Following this line of work, recent studies have focused on explicitly optimizing for search adaptivity and efficiency (Wang et al., 2025; Huang et al., 2025b; Song et al., 2025c). Notably, OTC (Wang et al., 2025) seeks to enhance tool productivity by penalizing search calls; IKEA (Huang et al., 2025b) trains agents to delineate knowledge boundaries and synergize internal and external knowledge. However, these approaches rely on intricate reward shaping around search-call counts that requires manual tuning and suffers from ambiguous credit assignment, leading agents to superficially reduce call frequency rather than improving search behavior. Moreover, whether to invoke search remains an implicit decision, limiting interpretability and hindering real-world deployment. In contrast, we adopt a simple outcome-based RL framework that separates search decision from problem solving, avoiding complex reward engineering while providing clearer learning signals. At inference time, AdaSearch offers explicit decision rationales, making the agent's behavior easier to inspect.

3 Examining Search Invocation Decisions on Vanilla Search Agents

3.1 Search Decision F1 Score

We evaluate models' search invocation decisions by computing the F1 score, where the positive class corresponds to the model deciding that its parametric knowledge is sufficient to answer the question (equivalently, not invoking search). For brevity, we use the notation $\mathbf{F1}_{\text{dec}}$ in the following sections. To determine whether a test instance is a true or false sample, we check whether the model can solve the instance using parametric knowledge alone.

Formally, we define

$$\mathbf{F1}_{\text{dec}} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \qquad (1)$$

where precision and recall are computed on the binary decision of whether to use search, comparing the model's choice with an oracle label that indicates whether the question is solvable with parametric knowledge. This metric reflects the model's capability to identify its knowledge boundaries.

Previous RL-based methods (Jin et al., 2025; Wang et al., 2025; Huang et al., 2025b) adopt a single prompt that asks the agent to conduct the decision-making process and problem-solving simultaneously. To compute the score for these methods, two cases arise: (1) if the model does not call search, we directly evaluate the EM of its final answer to determine whether the instance is a true or false sample; (2) if the model does call search, we enforce parametric-only reasoning by applying another system prompt s_{param} (Appendix F), and use the correctness of the resulting answer to decide whether the instance is a true or false sample.

3.2 Underuse of Parametric Knowledge in Vanilla Search Agents

We evaluate prior outcome-based RL methods using Qwen2.5-7B-Base (Team, 2024). Concretely, we include (1) Search-R1 (Jin et al., 2025), which equips LLMs with search access and optimizes them with outcome rewards, and (2) RL w/o Search, which conducts RL without search access and fully relies on parametric knowledge. QA performance is measured by exact match (**EM**). We follow the testing setup of Jin et al. (2025) and report macro-averaged **EM**, $\mathbf{F1}_{\text{dec}}$, and confusion matrices. We re-evaluate the official Search-R1 checkpoint.

Method	EM	$\mathbf{F1}_{\text{dec}}$	TP	TN	FP	FN
Search-R1	39.4	0.0	0.0	72.4	0.0	27.6
RL w/o Search	27.6	27.6	27.6	0.0	72.4	0.0

Table 1: Macro-averaged QA accuracy (EM), search decision F1 score ($\mathbf{F1}_{\text{dec}}$), and confusion matrix. **TP**: true positives; **TN**: true negatives; **FP**: false positives; **FN**: false negatives.

The results are summarized in Table 1. For Search-R1, although search access improves **EM**, its $\mathbf{F1}_{\text{dec}}$ is 0 because the model invokes search for every query. This overreliance, reflected in many false negatives, exemplifies tool overuse (Qian et al., 2025), where the model searches even when parametric knowledge suffices. In contrast, the model trained without search access achieves a higher $\mathbf{F1}_{\text{dec}}$ but lower **EM** due to limited internal knowledge. These results highlight the central challenge: enabling greater adaptivity by balancing parametric and external knowledge.

4 Our Method: AdaSearch

We introduce AdaSearch, illustrated in Figure 3. The key idea is to achieve adaptivity without complex reward engineering by disentangling problem solving from the decision of whether to invoke search. Separating these sub-tasks and optimizing them independently enables training policy LLM with simple outcome-based rewards. In this section, we first present the problem formulation (Sec. 4.1), and then describe the AdaSearch framework (Sec. 4.2). Our main approach is a two-stage training framework, where the first stage targets problem solving and the second stage focuses on decision making. Finally, we conclude with the inference pipeline of AdaSearch (Sec. 4.3).

4.1 Problem Formulation

The problem setting is the same as Jin et al. (2025). Given an input x, we let policy LLM π_θ interact with the search engine $\mathcal{E}$ to generate a trajectory τ. Formally,

$$\tau \sim \pi_\theta(\cdot \mid x; \mathcal{E}). \tag{2}$$

Our goal is to train the policy π_θ to interact with $\mathcal{E}$ by maximizing the following RL objective:

$$J(\theta) = \max_{\pi_\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}, \tau\sim\pi_\theta(\cdot|x;\mathcal{E})} \left[R(\tau, y)\right] - \beta\, \mathbb{D}_{\text{KL}} \left(\pi_\theta(\cdot \mid x; \mathcal{E}) \| \pi_{\text{ref}}(\cdot \mid x; \mathcal{E})\right), \tag{3}$$

where $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$ is the training set with size D, y is the ground truth for x, $R(\tau, y)$ is the reward function, and π_{ref} is a reference policy. We use GRPO (Shao et al., 2024) as the RL algorithm.

4.2 AdaSearch Training

Our training procedure is summarized in Algorithm 1. Concretely, it consists of two stages: the first stage aims to incentivize problem solving, while the second stage focuses on optimizing decision-making of search invocation.

Stage 1: Problem solving. Stage 1 focuses on incentivizing problem-solving capabilities. Concretely, we train the policy to utilize (1) parametric knowledge and (2) search call for problem solving. We design two different system prompts: (1) parametric-knowledge prompt s_{param}, which requires the policy to answer using only its internal knowledge, and (2) search prompt s_{search}, which permits the use of search tools. Full prompt details are provided in Appendix F.

For each instance (x, y) in the training batch, we augment the problem x with both s_{param} and s_{search}, and generate two groups of rollouts: $\mathcal{R}_{\text{param}} = \{\tau^1_{\text{param}}, \tau^2_{\text{param}}, \ldots, \tau^N_{\text{param}}\}$ and $\mathcal{R}_{\text{search}} = \{\tau^1_{\text{search}}, \tau^2_{\text{search}}, \ldots, \tau^M_{\text{search}}\}$.

For each trajectory $\tau \in \mathcal{R}_{\text{param}} \cup \mathcal{R}_{\text{search}}$, we use regular expressions to extract the final answer $\hat{y}$ from τ and apply exact match $\mathbf{EM}(\hat{y}, y) \rightarrow \{\texttt{true}, \texttt{false}\}$, which checks whether the extracted answer is exactly the same as any of the gold answer after normalization, to check the correctness. In the following sections, we use **EM** to denote this verification procedure for simplicity.

The reward design in this stage is simply the binary correctness reward. Formally,

$$R(\tau, y) = \mathbb{1}\left[\mathbf{EM} = \texttt{true}\right]. \tag{4}$$

Finally, we compute group-wise advantages as in Shao et al. (2024) and update policy with Eq 3.

Stage 2: Decision making. Stage 2 aims to incentivize search invocation decisions. Before training, we use the stage 1 policy π_{θ_1} and the parametric-knowledge prompt s_{param} to generate labels. For each training instance (x, y), we sample K responses from π_{θ_1} and compute the solve rate p using substring exact match (**SubEM**), which checks whether any gold answer is a substring of the extracted answer after normalization:

$$p = \frac{1}{K} \sum_{k=1}^{K} \mathbb{1}\left[\mathbf{SubEM}(\hat{y}_k, y) = \texttt{true}\right], \tag{5}$$

where $\hat{y}_k$ is the final answer extracted from the k-th response, and $\mathbb{1}[\cdot]$ is the indicator function.

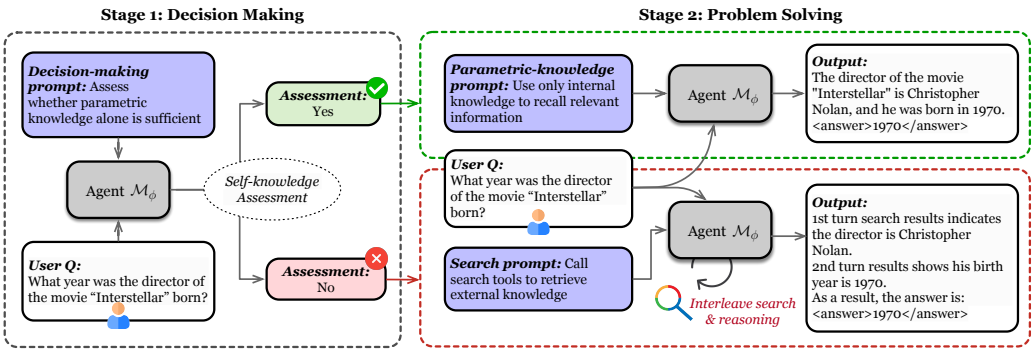


Figure 3: Overview of AdaSearch during inference. In stage 1, the agent explicitly reasons to decide whether the query can be solved via parametric knowledge. In stage 2, it follows the parametric-knowledge prompt if sufficient; otherwise, it switches to the search prompt to interleave reasoning and search for the final answer.

We use **SubEM** instead of the stricter **EM** because the objective here is not to evaluate exact answer formatting, but to estimate whether the model can solve the problem using its parametric knowledge. **SubEM** provides a more tolerant measure that avoids penalizing semantically correct answers with minor formatting differences, making it better suited for generating decision-making supervision.

Then, we define a threshold ρ and label instances with $p \geq \rho$ as solvable with parametric knowledge. We use the token `yes` as a label for such instances and the token `no` otherwise. With the generated labels, we craft the dataset $\mathcal{D}_{\text{decision}} = \{(x_i, \ell_i)\}_{i=1}^{D}$, where $\ell_i \in \{\texttt{yes}, \texttt{no}\}$, for stage-2 training.

We design a decision-making system prompt s_{decision} that requires explicit reasoning before producing the final decision ℓ, as detailed in Table 17. Such explicit reasoning increases transparency and interpretability of the decision-making process, making it clearer why the agent chooses to invoke search or rely on self-knowledge. The reward function in this stage remains a simple binary outcome reward, identical to that used in stage 1 (Eq. 4).

4.3 AdaSearch Inference

We adopt a two-stage inference pipeline (Fig. 3). In the first stage, the model is prompted with the system instruction s_{decision} to decide whether it can answer the query using self-knowledge through explicit reasoning. In the second stage, the model receives the corresponding problem-solving instruction: if it decides it can answer with self-knowledge, use s_{param}; otherwise, use s_{search}. Note that stage-1 history is not prepended in stage 2.

5 Experiments

5.1 Experiment Setup

Models. We conduct experiments on Qwen2.5 (Team, 2024) and Llama-3.2 (Grattafiori et al., 2024). Due to compute constraints, our main experiments use the 3B Instruct variants, with scaling results presented in Section 5.3.

Search environment. Following Jin et al. (2025), we use E5 (Wang et al., 2022) as the retriever over a 2018 Wikipedia dump, with the implementation from Griggs et al. (2025). For each query, we return the top-3 documents and limit the number of search calls to at most three. Additional results using different retrievers are provided in Appendix D.1.

Baselines. We consider three categories of baselines. For **prompting baselines**, we select Direct inference, Chain-of-Thought (CoT) Reasoning (Wei et al., 2022), and RAG (Lewis et al., 2020). For **outcome-RL baselines**, we apply GRPO (Shao et al., 2024) to train models on the parametric prompt s_{param} (RL w/o search) and on the search prompt s_{search} (Search-R1). For **reward-shaping baselines**, we include IKEA (Huang et al. (2025b), Table 4) as a representative method, and also design two baselines. The first one is denoted as "Naive Shaping". Its reward function is defined as

$$R(\tau, y) = \begin{cases} 1.0 - \lambda \cdot (\#\text{ search}) & \text{if } \textbf{EM} = \texttt{true}, \\ 0 & \text{otherwise}, \end{cases} \tag{6}$$

which is conceptually identical to OTC and R1-Searcher++ (Wang et al., 2025; Song et al., 2025b) but easier to implement, assigning higher rewards to correct rollouts with fewer search calls. We set $\lambda = 0.05$.

In addition, we introduce another reward-shaping baseline designed to enforce prompt-level search decisions, which closely resembles the objective of AdaSearch. We refer to it as "Decision

Method			General QA						Multi-Hop QA								Avg.	
			NQ		TQ		PopQA		HotpotQA		2Wiki		MuSiQue		Bamboogle			
			EM	$F1_{dec}$	EM	$F1_{dec}$	EM	$F1_{dec}$	EM	$F1_{dec}$	EM	$F1_{dec}$	EM	$F1_{dec}$	EM	$F1_{dec}$	EM	$F1_{dec}$
Qwen2.5-3B-Instruct	*Prompting*	Direct Answer	8.1	8.1	24.1	24.1	7.8	7.8	13.2	13.2	19.5	19.5	1.4	1.4	5.6	5.6	11.4	11.4
		CoT	14.1	14.1	38.2	38.2	13.5	13.5	17.3	17.3	22.8	22.8	3.9	3.9	24.0	24.0	19.1	19.1
		RAG	31.7	0.0	52.7	0.0	38.1	0.0	24.6	0.0	20.1	0.0	4.4	0.0	8.0	0.0	25.7	0.0
	Outcome RL	RL w/o search	21.7	21.7	44.9	44.9	17.3	17.3	20.5	20.5	28.0	28.0	5.9	5.9	23.2	23.2	23.1	23.1
		Search-R1	**42.7**	0.0	**60.1**	0.3	43.9	0.0	**36.9**	0.1	37.3	0.2	**16.1**	0.0	**29.6**	0.0	**38.1**	0.1
	Reward-Shaping	Naive Shaping	37.8	41.6	54.8	66.9	42.2	55.3	29.8	49.3	33.1	58.4	9.9	15.8	24.8	43.8	33.2	47.3
		Decision Shaping	37.7	45.9	56.1	70.3	41.8	**61.3**	32.1	57.5	35.9	61.1	12.5	21.7	22.4	41.1	34.1	51.3
		IKEA	41.4	37.7	59.5	62.4	**44.2**	55.2	32.0	**58.1**	35.0	60.2	11.5	14.2	23.2	46.0	35.3	47.7
	Proposed	**AdaSearch**	37.9	**49.0**	56.8	**71.7**	42.8	58.8	33.4	57.5	**38.5**	**65.5**	14.4	**25.2**	28.0	**50.0**	36.0	**54.0**
Llama-3.2-3B-Instruct	*Prompting*	Direct Answer	18.8	18.8	42.3	42.3	16.4	16.4	12.5	12.5	16.2	16.2	3.0	3.0	8.8	8.8	16.9	16.9
		CoT	25.0	25.0	45.6	45.6	15.8	15.8	16.3	16.3	11.3	11.3	4.5	4.5	35.2	35.2	22.0	22.0
		RAG	29.4	0.0	53.0	0.0	35.7	0.0	20.6	0.0	8.3	0.0	4.1	0.0	12.8	0.0	23.4	0.0
	Outcome RL	RL w/o search	38.2	38.2	54.6	54.6	23.1	23.1	23.7	23.7	26.7	26.7	6.7	6.7	34.4	34.4	29.6	29.6
		Search-R1	**46.1**	1.7	**64.4**	2.5	**44.8**	1.5	**38.2**	6.6	34.8	1.0	**14.6**	1.3	**42.4**	0.0	**40.7**	2.1
	Reward-Shaping	Naive Shaping	41.2	57.6	59.7	75.3	39.3	56.3	31.2	52.6	33.5	54.8	11.5	21.1	36.8	55.9	36.2	53.4
		Decision Shaping	41.4	58.8	61.5	75.4	41.2	61.4	33.7	57.5	33.7	48.1	13.0	31.4	32.0	58.1	36.6	55.8
		IKEA	42.2	58.6	62.0	76.2	41.1	59.9	32.3	58.2	32.9	51.3	11.2	31.9	40.8	63.1	37.5	57.0
	Proposed	**AdaSearch**	43.5	**62.7**	62.9	**79.2**	40.4	**62.3**	34.4	**59.2**	**36.2**	**60.1**	13.2	**36.5**	**42.4**	**64.2**	39.0	**60.6**

Table 2: Results of different model across benchmarks. **EM** denotes exact match. $\mathbf{F1}_{dec}$ denotes the search invocation decision-making score (see Sec. 3). **Avg** is computed by averaging metrics over benchmarks. The best score is shown in **bold** and the second best in underline.

Shaping". Its reward function is defined as

$$R(\tau, y) = \mathbb{1}\left[\mathbf{EM} = \texttt{true}\right] + \alpha \cdot \begin{cases} \mathbb{1}\left[\tau \text{ has search}\right] & \text{if } p < \rho, \\ \mathbb{1}\left[\tau \text{ has no search}\right] & \text{otherwise,} \end{cases} \tag{7}$$

where p is the empirical solve rate defined in Eq 5, ρ is the threshold for p, and α is the hyperparameter controlling the bonus for correct decisions. This baseline rewards the model for making appropriate search decisions. We estimate p for each sample using the base policy before RL training and set both α and ρ to 0.5. Finally, the prompts used for each baseline and the hyperparameter analysis are detailed in Appendix F and D.4, respectively.

Training. To ensure fair comparison, we construct a difficulty-balanced training set of 8,192 samples following the procedure of Huang et al. (2025b). A validation set is constructed using the same procedure and used for checkpoint selection. Full details for dataset construction are presented in Appendix B.2. For AdaSearch stage 2, we use the same threshold $\rho = 0.5$. Full training details are provided in Appendix B.3.

Evaluation. Following Jin et al. (2025), we use **EM** as the task metric, apply greedy decoding, and evaluate on standard QA benchmarks. For single-hop QA, we use Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023). For multi-hop QA, we use HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023). We also report $\mathbf{F1}_{dec}$ (Section 3) to measure the quality of search invocation decisions. Interpretability and safety evaluations are detailed in the Appendix C.

5.2 Main Results

Comparison against prompting and outcome-RL baselines. Table 2 summarizes the results of prompting and outcome-RL baselines. In terms of task performance **EM**, Search-R1 achieves the best overall results, and the RAG baseline consistently outperforms all prompting-based methods. This pattern reflects the characteristics of these benchmarks, where retrieval generally offers an advantage over purely parametric reasoning; the performance gap between RL w/o Search and Search-R1 further supports this observation. Regarding search decision, however, both RAG and Search-R1 obtain scores close to zero across benchmarks due to their always-search behavior. The results of Search-R1 highlight the deficiency of naive QA outcome rewards in balancing parametric and external knowledge. Non-search baselines achieve higher $\mathbf{F1}_{dec}$, but because they never explicitly assess their own knowledge boundaries, their decision-making remains suboptimal. In comparison, our AdaSearch maintains competitive **EM** while substantially improving search invocation decision-making across model families, achieving 54% to 60% relative gain in $\mathbf{F1}_{dec}$ over Search-R1. These results highlight that our method can effectively elicit better decisions without compromising task performance.

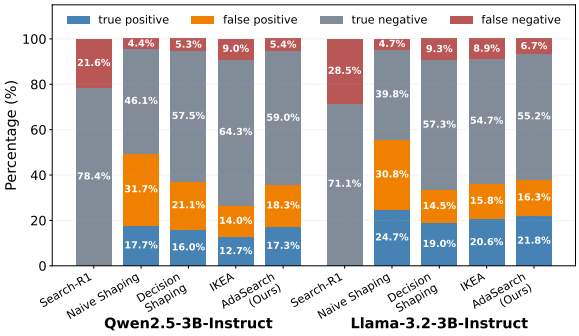


Figure 4: Confusion matrix of different RL methods.

Model	Method	EM	$\mathbf{F1}_{\text{dec}}$	Prec.	Rec.
Qwen2.5-3B-Instruct	Search-R1	**38.1**	0.1	39.0	0.0
	Naive Shaping	33.2	47.3	36.1	**76.8**
	Decision Shaping	34.1	51.3	42.0	70.3
	IKEA	35.3	47.6	45.0	53.4
	AdaSearch	36.0	**54.0**	45.1	68.8
	AdaSearch-E2E	35.2	51.3	44.1	63.9
	AdaSearch-SFT	37.0	51.1	**45.9**	59.5
Llama-3.2-3B-Instruct	Search-R1	**40.7**	2.1	47.3	1.1
	Naive Shaping	36.2	53.4	41.4	**79.0**
	Decision Shaping	36.6	55.8	52.1	60.5
	IKEA	37.5	57.0	51.4	64.5
	AdaSearch	39.0	**60.6**	**53.0**	71.4
	AdaSearch-E2E	37.7	58.6	48.1	75.9
	AdaSearch-SFT	36.6	55.6	46.6	69.4

Table 3: Averaged **EM**, $\mathbf{F1}_{\text{dec}}$, precision, and recall.

Comparison against reward-shaping baselines. As shown in Table 2, our AdaSearch consistently outperforms intricate reward-shaping baselines in both **EM** and $\mathbf{F1}_{\text{dec}}$. First, Naive Shaping generally underperforms other baselines in **EM**, suggesting that naively penalizing tool usage might hurt performance. On the other hand, although Decision Shaping generally shows improvements in both **EM** and $\mathbf{F1}_{\text{dec}}$ over Naive Shaping, the results are still suboptimal. We suspect that this may be due to distribution shifts, where solve rates p change during training. Consequently, using initial offline-generated labels introduces label noise. Lastly, IKEA (Huang et al., 2025b) generally outperforms the other reward-shaping baselines in both **EM** and $\mathbf{F1}_{\text{dec}}$ (except on Qwen2.5-3B-Instruct). Its reward design can be viewed as combining both Naive Shaping (e.g., penalizing tool usage on correct trajectories) and Decision Shaping (e.g., adding bonuses on incorrect trajectories that involve search calls). However, AdaSearch still consistently outperforms all reward-shaping baselines across models, especially on challenging multi-hop QA where search is more genuinely required. This suggests that reward shaping often over-optimizes for reducing tool calls and fails to adapt to query difficulty. In contrast, AdaSearch strengthens search invocation decision-making while achieving better task performance through explicit binary supervision.

5.3 Analysis

AdaSearch achieves better search invocation decision-making. Figure 4 and Table 3 presents the confusion matrices and precision–recall–F1 scores across different methods and models. For Naive Shaping, we observe the lowest false negative rates but substantially higher false positive rates for both models (Fig. 4). This pattern suggests that naively penalizing search usage causes the model to underuse search: it spuriously avoids invoking search even when its parametric knowledge is insufficient, thereby drifting away from adaptive behavior. In contrast, AdaSearch achieves comparable true positive and false negative rates while significantly reducing the false positive rate.

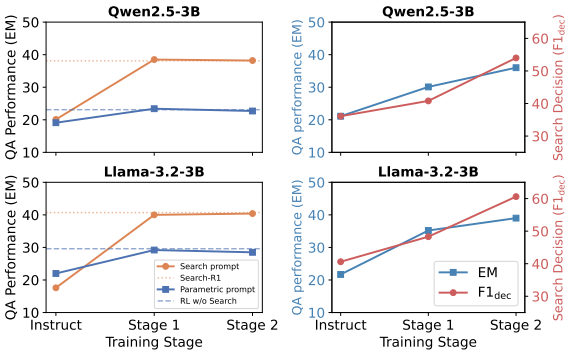
For the other reward-shaping baselines, although they show more balanced false positive and false negative rates, AdaSearch still outperforms them in both $\mathbf{F1}_{\text{dec}}$ and true positive rate. A closer examination shows that AdaSearch attains precision on par with Decision Shaping and IKEA, while achieving noticeably higher recall (Figure 3).

Two-stage training vs joint optimization. To validate the benefit of two-stage training over joint optimization, we implement an end-to-end variant, **AdaSearch-E2E**, which jointly optimizes problem solving and search invocation decision-making. This design enables on-the-fly label generation for search decisions by using the empirical solve rate of parametric knowledge at each step to produce labels. Further details are provided in Appendix E.

As shown in Figure 3. AdaSearch-E2E consistently underperforms AdaSearch, suggesting that the two-stage approach is more effective.

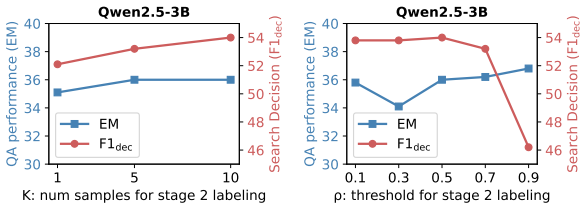
RL vs SFT in stage-2 training. A common approach for teaching models to express uncertainty is supervised fine-tuning (SFT) (Lin et al., 2022; Zhang et al., 2024). In stage 2, we implement an SFT variant, **AdaSearch-SFT**. After computing solve rates (Sec. 4.2), we directly construct target responses: "`<assessment>`yes`</assessment>`" when $p \geq 0.5$, and "`<assessment>`no`</assessment>`" otherwise. Details are provided in Appendix B.

The results are shown in Figure 3. While AdaSearch-SFT improves $\mathbf{F1}_{\text{dec}}$ over Search-R1 and achieves slightly higher **EM** on Qwen, it still underperforms AdaSearch on Llama across

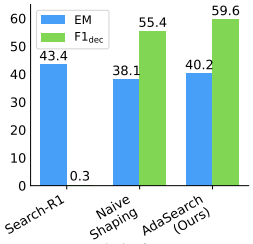


(a) Problem-solving **EM**. (b) Avg. **EM** and $\mathbf{F1}_{\text{dec}}$.

Figure 5: Averaged performance. (a) Stage 1 substantially boosts both parametric and search-based problem solving, with Stage 2 maintaining this performance. (b) Both test **EM** and search decision-making ($\mathbf{F1}_{\text{dec}}$) improve steadily during training.



(a) Number of samples K. (b) Solve rate threshold ρ.



(c) Model size (7B).

Method	Qwen	Llama
Search-R1	1.64	1.75
Naive	0.77	0.69
Decision	1.00	1.11
IKEA	0.88	0.79
AdaSearch	1.08	1.10

(d) Avg. number of search.

Figure 6: Ablations of AdaSearch.

both metrics. The larger $\mathbf{F1}_{\text{dec}}$ gap on MuSiQue and Bamboogle highlights that reasoning-before-assessment is crucial for OOD generalization.

Learning curves. The learning curves are shown in Figure 5, with the training reward trends provided in Appendix D.2. In stage 1, the model's performance on both search-based and parametric problem solving matches the single-task baselines, Search-R1 and RL w/o Search (Fig. 5a). In stage 2, the quality of search invocation decisions improves substantially and task performance continues to rise (Figure 5b). Moreover, Stage 2 does not degrade problem-solving ability (Fig. 5a), consistent with findings that on-policy RL mitigates forgetting (Shenfeld et al., 2025; Chen et al., 2025a).

Stage-2 labeling hyperparameters. To understand the effect of labeling hyperparameters for stage-2 training, we conduct ablation studies on (1) the number of responses K used to estimate the empirical solve rate in stage 2 and (2) the solve-rate threshold ρ for stage-2 labeling. Additional analysis is provided in Appendix D.3.

For (1) (Fig. 6a), increasing K improves both $\mathbf{F1}_{\text{dec}}$ and **EM** via more reliable solve-rate estimates. Notably, even $K = 1$ performs well (35.1% **EM** and 52.1% $\mathbf{F1}_{\text{dec}}$), likely because RL sharpens output distributions (Yue et al., 2025; He et al., 2025), allowing a single sample to produce effective signals. For (2) (Fig. 6b), ρ has limited effect when $\rho \leq 0.5$, aside from a drop in **EM** at $\rho = 0.3$. As ρ increases further, $\mathbf{F1}_{\text{dec}}$ declines while **EM** rises, since higher ρ encourages more no assessments, increasing search usage.

Model sizes. To assess the robustness of our method across model scales, we train Qwen2.5-7B-Instruct and compare AdaSearch with Search-R1 and Naive Shaping. As shown in Figure 6c, AdaSearch boosts $\mathbf{F1}_{\text{dec}}$ by roughly 60% compared to Search-R1. Relative to Naive Shaping, AdaSearch improves both **EM** (+2.1%) and $\mathbf{F1}_{\text{dec}}$ (+4.2%). These results indicate that AdaSearch generalizes well to larger model scales.

Average number of searches. We report the average search calls in Figure 6d. AdaSearch reduces unnecessary search usage by 34–38% compared to Search-R1, and its frequency is similar to Decision Shaping since both rely on self-knowledge for prompt-level decision making. AdaSearch uses slightly more searches than Naive Shaping and IKEA, which is expected because it does not directly optimize for minimizing tool calls. Further analysis of efficiency is provided in Appendix D.5.

6 Conclusion

In this work, we first conduct analysis and reveal that existing search agents often struggle to recognize the limits of their parametric knowledge, leading to leading to unnecessary or excessive search. Motivated by this observation, we propose AdaSearch, a simple two-stage, outcome-driven RL framework that avoids complex reward engineering while improving both decision quality and task performance. AdaSearch achieves the most reliable search invocation decision-making across different models, reduces unnecessary search calls, preserves task accuracy, and offers more transparent and interpretable decisions about when to search. These results demonstrate that AdaSearch is an effective and generalizable approach for building adaptive, trustworthy search agents.

Limitations

Despite the advances demonstrated by our method, several limitations remain to be addressed in future research.

First, due to computational constraints, we only evaluated models up to 7B parameters. Future research should investigate whether our findings scale to larger architectures (e.g., 14B, 32B).

Second, our evaluation relies primarily on static QA benchmarks, following the representative Search-R1 (Jin et al., 2025). Testing the method on (1) universally unanswerable questions (Kirichenko et al., 2026) and (2) dynamic, evolving facts (Vu et al., 2024) would better reflect real-world information-seeking challenges.

Third, although our approach reduces safety risks by reducing unnecessary search interactions (Appendix C.2), it does not eliminate them entirely. Combining AdaSearch with safety alignment techniques remains an important future direction.

Finally, despite achieving superior $\mathbf{F1}_{\text{dec}}$, the model still makes some errors in search invocation decisions (see Appendix G.3, G.4). Improving decision quality through fine-grained uncertainty quantification beyond binary outputs is a key area for further development.

References

Anthropic. 2026. System card: Claude sonnet 4.6. Technical report, Anthropic. Accessed: 2026-04-04.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. 2025. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025a. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, and 1 others. 2025b. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Jianshuo Dong, Sheng Guo, Hao Wang, Xun Chen, Zhuotao Liu, Tianwei Zhang, Ke Xu, Minlie Huang, and Han Qiu. 2025. Safesearch: Automated red-teaming for the safety of llm-based search agents. *arXiv preprint arXiv:2509.23694*.

Yuchen Fan, Kaiyan Zhang, Heng Zhou, Yuxin Zuo, Yanxu Chen, Yu Fu, Xinwei Long, Xuekai Zhu, Che Jiang, Yuchen Zhang, and 1 others. 2025. Ssrl: Self-search reinforcement learning. *arXiv preprint arXiv:2508.10874*.

Shangbin Feng, Weijia Shi, Yuyang Bai, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2024. Knowledge card: Filling llms' knowledge gaps with plug-in specialized language models. In *ICLR*.

Matija Franklin, Nenad Tomašev, Julian Jacobs, Joel Z Leibo, and Simon Osindero. 2026. Ai agent traps.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tyler Griggs, Sumanth Hegde, Eric Tang, Shu Liu, Shiyi Cao, Dacheng Li, Charlie Ruan, Philipp Moritz, Kourosh Hakhamaneshi, Richard Liaw, Akshay Malik, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. 2025. Evolving skyrl into a highly-modular rl framework. Notion Blog.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieve step by step for large language models. *arXiv preprint arXiv:2502.01142*.

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

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.

Andre Wang He, Daniel Fried, and Sean Welleck. 2025. Rewarding the unlikely: Lifting GRPO beyond distribution sharpening. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language*

Processing, pages 25559–25571, Suzhou, China. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025a. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

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

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

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong 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 7029–7043.

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.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Adam Tauman Kalai, Ofir Nachum, Santosh S Vempala, and Edwin Zhang. 2025. Why language models hallucinate. *arXiv preprint arXiv:2509.04664*.

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

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, and 1 others. 2023. Realtime qa: What's the answer right now? *Advances in neural information processing systems*, 36:49025–49043.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel Bell. 2026. Abstentionbench: Reasoning LLMs fail on unanswerable questions. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

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. *Advances in neural information processing systems*, 33:9459–9474.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*.

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. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2026. Gpt-5.4 thinking system card. Accessed: 2026-03-28.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. 2023. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Smart: Self-aware agent for tool overuse mitigation. *arXiv preprint arXiv:2502.11435*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2025. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. 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 8364–8377.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

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. *arXiv 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. *arXiv preprint arXiv:2505.17005*.

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025c. Smart-searcher: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13572–13586.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the*

61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and 1 others. 2024. Freshllms: Refreshing large language models with search engine augmentation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 13697–13720.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhepei Wei, Wei-Lin Chen, and Yu Meng. 2024. Instructrag: Instructing retrieval-augmented generation via self-synthesized rationales. *arXiv preprint arXiv:2406.13629*.

Teng Xiao, Yige Yuan, Hamish Ivison, Huaisheng Zhu, Faeze Brahman, Nathan Lambert, Pradeep Dasigi, Noah A Smith, and Hannaneh Hajishirzi. 2026. Meta-reinforcement learning with self-reflection for agentic search. *arXiv preprint arXiv:2603.11327*.

Roy Xie, Deepak Gopinath, David Qiu, Dong Lin, Haitian Sun, Saloni Potdar, and Bhuwan Dhingra. 2026. Over-searching in search-augmented large language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7714–7739, Rabat, Morocco. Association for Computational Linguistics.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Gui, Ziran Jiang, Ziyu Jiang, and 1 others. 2024. Crag-comprehensive rag benchmark. *Advances in Neural Information Processing Systems*, 37:10470–10490.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *2nd AI for Math Workshop @ ICML 2025*.

Hanning Zhang, Shizhe Diao, Yong Lin, Yi Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. 2024. R-tuning: Instructing large language models to say 'I don't know'. 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 7113–7139, Mexico City, Mexico. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Comparison on Reward Functions of Different RL Methods

We summarize the reward designs of different methods in Table 4. **OTC** (Wang et al., 2025) is designed to minimize search calls and maximize task performance. **IKEA** (Huang et al., 2025b) shares a similar spirit, but focus more on adaptivity by adding a knowledge-boundary bonus (r_{kb}^{-}) to encourage agents using search on difficult problems. **Naive Shaping** is conceptually similar to OTC but much easier to implement, aiming to encourage agents to use fewer search calls while obtaining the correct answer. **Decision Shaping** aims to sharply penalize inconsistent transitions between no-search and search behaviors. In contrast, AdaSearch use a simple binary rewards to optimize both problem solving and decision making on whether to search.

B Implementation Details

B.1 AdaSearch Training Algorithm

The pseudocode for AdaSearch is summarized in Algorithm 1.

B.2 Training Dataset Construction

To ensure a fair comparison across training-based baselines, we follow the procedure of Huang et al.

Search-R1 (Jin et al., 2025) & AdaSearch (Ours)

$$R(\tau, y) = \begin{cases} 1.0 & \text{if } \mathbf{EM} = \texttt{true}, \\ 0 & \text{otherwise.} \end{cases}$$

OTC-GRPO (Wang et al., 2025)

$$R(\tau, y) = \alpha \cdot R_{\text{tool}} \cdot \mathbb{1}\left[\mathbf{EM} = \texttt{true}\right], \text{ where } R_{\text{tool}} = \begin{cases} 1.0 & \text{if } f(m,n) = n = 0, \\ \cos\left(\frac{m\pi}{2m+c}\right) & \text{if } n = 0, \\ \sin\left(\frac{f(m,n)\pi}{2n}\right) & \text{otherwise.} \end{cases} \quad f(m,n) = \begin{cases} 0 & \text{if } m = n = 0, \\ m & \text{if } n = 0, \\ \frac{2nm}{m+n} & \text{otherwise.} \end{cases}$$

α: hyperparameter, m: # search calls in τ, n: min # search calls for input x during training.

IKEA (Huang et al., 2025b)

$$R(\tau, y) = \begin{cases} -1 & \text{if incorrect format,} \\ \mathbb{1}\left[\mathbf{EM} = \texttt{true}\right] + R_{\text{kb}} & \text{otherwise.} \end{cases} \quad R_{\text{kb}} = \begin{cases} r_{\text{kb}^+}\left(1 - \frac{\text{RT}}{\text{RT}_{\text{max}}}\right) & \text{if } \mathbf{EM} = \texttt{true}, \\ 0 & \text{if } \mathbf{EM} = \texttt{false} \text{ \& RT = 0}, \\ r_{\text{kb}^-} & \text{otherwise.} \end{cases}$$

RT: # search calls; RT_{max}: max # search allowed; $r_{\text{kb}^+}, r_{\text{kb}^-}$: hyperparameters.

Naive Shaping

$$R(\tau, y) = \begin{cases} 1.0 - \lambda \cdot (\text{\# search calls}) & \text{if } \mathbf{EM} = \texttt{true}, \\ 0 & \text{otherwise,} \end{cases}$$

where λ is the hyperparameter controlling penalties for search calls.

Decision Shaping

$$R(\tau, y) = \mathbb{1}\left[\mathbf{EM} = \texttt{true}\right] + \alpha \cdot \begin{cases} \mathbb{1}\left[\tau \text{ has search}\right] & \text{if } p < \rho, \\ \mathbb{1}\left[\tau \text{ has no search}\right] & \text{otherwise,} \end{cases}$$

where α: search decision bonus, p: solve rate for the input problem, and ρ: threshold for solve rate p.

Table 4: Comparison of different reward designs. Unlike prior methods that rely on complex reward engineering, such as OTC (Wang et al., 2025) and IKEA (Huang et al., 2025b), AdaSearch uses a simple binary outcome reward to explicitly optimize both problem solving and decision making.

(2025b) to construct a difficulty-balanced training set. Specifically, we use the base policy with the system prompt s_{probe} (Table 14), which encourages the model to answer directly without accessing additional information, to generate $K = 10$ responses for each problem in the training set of Jin et al. (2025), drawn from Natural Questions (Kwiatkowski et al., 2019) and HotpotQA (Yang et al., 2018). We then compute solve rates using substring exact matching (**SubEM**) for verification. Using a threshold $\rho = 0.5$, we split the dataset into easy and hard subsets and sample 4,096 examples from each to form the final training set. The same procedure is used to construct a validation set of 2,048 examples, which is used for checkpoint selection during evaluation.

We create datasets for each model independently, and all methods evaluated on the same model use the same dataset to ensure fair comparison.

B.3 Training Details

Most RL training experiments are conducted on a compute node equipped with 4 NVIDIA A100-80GB GPUs. The only exception is the outcome-based RL baselines for the 3B models, which are trained on a machine with 2 NVIDIA H100-94GB GPUs, as these baselines require relatively fewer computational resources. We use full parameter fine-tuning for all experiments. To optimize training efficiency, we adopted DeepSpeed with Zero3 offload, gradient checkpointing, FlashAttention-2, and bfloat16 mixed precision training.

We use LlamaFactory (Zheng et al., 2024) for SFT and conduct hyperparameter search on batch size $\in \{16, 32\}$ and learning rate $\in \{1\text{e-}6, 5\text{e-}6\}$. We only train for one epoch as the loss quickly converged to nearly 0. We select the checkpoint that maximizes the product of validation **EM** and $\mathbf{F1}_{\text{dec}}$.

For RL training, we use SkyRL (Griggs et al., 2025) as our base framework. Hyperparameters for RL baselines and for AdaSearch are shown in Table 5 and Table 6, respectively. Most hyperparameters follow Jin et al. (2025) and Wang et al. (2025). For reward-shaping baselines, we increase the group size to 15 to match the compute used in problem-solving training (AdaSearch stage 1).

For the reward-function hyperparameters, we set λ in Naive Shaping (Equation (6)) to 0.05; both α and ρ in Decision Shaping to 0.5; and r_{kb}^{+} and

Algorithm 1 AdaSearch Training

Require: base policy LLM π_{θ_b}, search engine $\mathcal{E}$, training set $\mathcal{D} = (x_i, y_i)$, prompts s_{param}, s_{search}, s_{decision}, string concatenation $[\cdot, \cdot]$, solve-rate threshold ρ.

Initialize policy $\pi_\theta \leftarrow \pi_{\theta_b}$.
/* Stage-1 Training */
for iteration = 1 to T **do**
  Sample a batch $\mathcal{D}_b \sim \mathcal{D}$.
  $\mathcal{R} \leftarrow \{\}$; $\mathcal{A} \leftarrow \{\}$.
  for each $(x_i, y_i) \sim \mathcal{D}_b$ **do**
    for $g \in \{\text{param}, \text{search}\}$ **do**
      /* Rollout Phase */
      if g is param **then**
        Generate $\mathcal{R}_g^i = \{\tau_{\text{param}}^n\}_{n=1}^N$, where $\tau_{\text{param}}^n \sim \pi_\theta(\cdot \mid [s_{\text{param}}, x_i])$.
      else
        Generate $\mathcal{R}_g^i = \{\tau_{\text{search}}^m\}_{m=1}^M$, where $\tau_{\text{search}}^m \sim \pi_\theta(\cdot \mid [s_{\text{search}}, x_i]; \mathcal{E})$.
      end if
      Compute rewards For all $\tau \in \mathcal{R}_g^i$ with Eq 4.
      Compute group-wise advantages $\mathcal{A}_g^i$ For $\mathcal{R}_g^i$.
      $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_g^i$; $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_g^i$. // Aggregate rollouts & advantages
    end for
  end for
  Update π_θ using GRPO with $\mathcal{R}$ and $\mathcal{A}$.
end for
/* Stage-2 Training */
Generate dataset $\mathcal{D}_{\text{decision}}$ with π_θ, $\mathcal{D}$, and s_{decision}. // See section 4.2
Update π_θ with $\mathcal{D}_{\text{deision}}$ using GRPO.
return π_θ

r_{kb}^- in IKEA (Table 4) to 0.2 and 0.05, following the original IKEA paper (Huang et al., 2025b). In addition, we analyze the reward-function hyperparameters for Naive Shaping and compare our training hyperparameters with those in the original IKEA implementation in Appendix D.4.

For AdaSearch stage 2, we search over the combinations of (batch size, mini-batch size) in (128, 64), (256, 128). For Llama-3.2-3B-Instruct, we use the (128, 64) setting, while for the Qwen2.5 series models, we use (256, 128).

For efficient rollouts, we use vLLM (Kwon et al., 2023) and set the temperature to 1.0, top-p to 1.0, and top-k to –1, following Jin et al. (2025).

For baselines that involve non-search generation (e.g., RL w/o Search, rollouts for s_{param} in AdaSearch stage 1, and AdaSearch stage 2), we set the maximum number of turns to 2, with a maximum generation length of 500 tokens in the second turn. In most cases, the full trajectory of these baselines contains only one turn unless the model fails to follow the required output format (e.g., the answer is not wrapped in `<answer>...</answer>` or the assessment is not wrapped in `<assessment>...</assessment>`). Similar to Search-R1 (Jin et al., 2025), which appends a nudge prompt (Table 23) when the model violates the required format, we design nudge prompts (Table 24 and 25) to enforce correct formatting. This formatting-correction procedure is applied to all reward-shaping baselines as well to ensure fairness.

We save checkpoints at the end of each epoch. For RL baselines, we select the checkpoint that achieves the highest validation reward. For AdaSearch stage 1, we use the final checkpoint before collapse. For AdaSearch stage 2, we select the checkpoint that maximizes the product of validation **EM** and $\mathbf{F1}_{\text{dec}}$.

B.4 Inference Details

We perform inference with vLLM (Kwon et al., 2023) and bfloat16 precision on the same compute nodes used for training. For each method, the maximum generation length, maximum number of turns, and maximum input length are set to match

Hyperparameter	Search-R1	RL w/o Search	Reward-Shaping Baselines
(batch size, mini-batch size)	(512, 256)	(512, 256)	(512, 256)
learning rate	1e-6	1e-6	1e-6
epoch	6	6	6
group size	5	10	15
KL loss coefficient β	0.001	0.001	0.001
PPO clip ratio	0.2	0.2	0.2
warm-up step ratio	0.285	0.285	0.285
max generation length	500	2048*	500
max number of turns	4	2	4
max input length	4096	3072	4096

Table 5: Hyperparameters for RL baselines. *: in the second turn, the maximum generation length is set to 500 (See Appendix B.3).

Hyperparameter	AdaSearch stage 1	AdaSearch stage 2
(batch size, mini-batch size)	(512, 256)	{(128, 64), (256, 218)}
learning rate	1e-6	1e-6
epoch	6	6
group size	$(s_{\text{search}}, s_{\text{param}}) = (5, 10)$	5
KL loss coefficient β	0.001	0.001
PPO clip ratio	0.2	0.2
warm-up step ratio	0.285	0.285
max generation length	$(s_{\text{search}}, s_{\text{param}}) = (500, 2048^*)$	2048*
max number of turns	$(s_{\text{search}}, s_{\text{param}}) = (4, 2)$	2
max input length	$(s_{\text{search}}, s_{\text{param}}) = (4096, 3072)$	3072

Table 6: Hyperparameters for AdaSearch. *: in the second turn, the maximum generation length is set to 500 (See Appendix B.3).

the rollout configuration used during training for consistency. Following Jin et al. (2025), we use greedy decoding for evaluation.

C Additional Evaluations

C.1 Interpretability

Evaluations were conducted via CoT monitoring (Baker et al., 2025) on 500 random samples from the Natural Questions (Kwiatkowski et al., 2019) test set. We utilized GPT-5.4-mini (OpenAI, 2026) and Claude 4.6 Sonnet (Anthropic, 2026) as judges. The evaluation prompts, adapted from Baker et al. (2025), are provided in Tables 26 and 27. Each judge assessed whether the agent (1) explicitly reasoned about its own knowledge and (2) produced a clear sufficiency judgment.

We report Interpretable Rate (IR), the ratio of trajectories where the judge thinks are interpretable based on the criteria, in Table 7. Across both judge models, AdaSearch consistently outperforms all

Method	IR (gpt)	IR (Claude)
Search-R1	7.4	7.0
Naive Shaping	17.2	20.6
IKEA	10.0	11.0
Decision Shaping	35.8	40.8
AdaSearch (Ours)	**44.2**	**48.0**

Table 7: Evaluation of interpretability on Qwen2.5 3B. AdaSearch achieves the overall best Interpretable Rate (IR).

baselines (by 8.4-36.8% using gpt-5.4-mini; by 7.2-41.0% using Claude Sonnet 4.6). These results suggest that AdaSearch offers more interpretable search decisions compared to the baselines.

C.2 Safety

We conduct evaluation using the SafeSearch dataset (Dong et al., 2025). Specifically, we use the `harmful_output` subset, where each query is paired with a malicious document designed to mislead users and pose risks, on top of our retrieval environment. Following the original paper's protocol, we inject the harmful document at position 1 (i.e., "Doc 1") of the first search call and report the Attack Success Rate (ASR). As shown in Table 8, AdaSearch reduces ASR by 15% compared to Search-R1. Further analysis reveals that AdaSearch invokes search on 80% of test instances, compared to 97% for Search-R1, suggesting that fewer search calls effectively limit exposure to malicious content.

We also note that AdaSearch is orthogonal to safety alignment methods that teach agents to ignore malicious documents, and we leave their combination as future work.

Method	Attack Successful Rate
Search-R1	91.7
AdaSearch	**76.7**

Table 8: Evaluation of safety on Qwen2.5 3B. We test the agents on the harmful output subset of SafeSearch. Our method effectively reduces ASR by 15% compared to Search-R1.

D Additional Analysis

In this section, we provide additional empirical evidence and analysis. Specifically, we report results using the BM25 retriever (Appendix D.1), visualize the training reward trajectories (Appendix D.2), analyze the impact of hyperparameters across various methods (Appendix D.3 and D.4), and evaluate efficiency regarding search-call counts and latency (Appendix D.5).

D.1 BM25 Results

We conduct experiments using Qwen2.5-3B-Instruct with BM25 (Robertson and Walker, 1994) for both training and inference; results are presented in Figure 7. AdaSearch generalizes well to different choices of retrievers. Specifically, compared to Search-R1, AdaSearch significantly improves search invocation decision-making ($\mathbf{F1}_{\text{dec}}$) without sacrificing QA performance (EM).

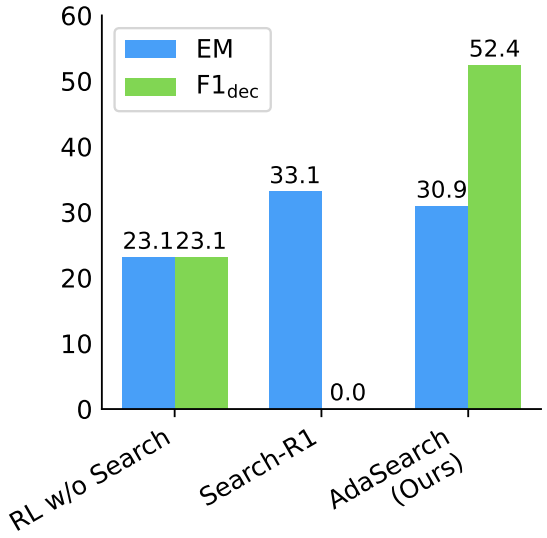
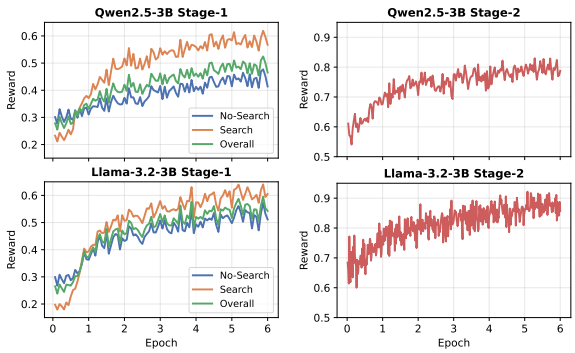


Figure 7: BM25 results on Qwen2.5-3B-Instruct. AdaSearch generalize to different retrievers.

D.2 Training Rewards

Figure 8 illustrates the training reward curves for AdaSearch on Qwen2.5-3B-Instruct and Llama-3.2-3B-Instruct. In Stage 1, rewards for both parametric knowledge and search-based problem solving show a steady upward trend. In Stage 2, the rewards stabilize smoothly as well, reflecting a consistent and robust training process.



(a) Stage-1 rewards. (b) Stage-2 rewards.

Figure 8: AdaSearch training rewards. The rewards increase steadily as training proceed in both stage 1 and stage 2.

D.3 AdaSearch Stage 2: Threshold ρ vs Number of Search Calls

We analyze the effect of the solve-rate threshold ρ on the number of search calls using Qwen2.5-3B-Instruct, with results shown in Table 9. As ρ increases, the number of search calls rises, which is expected since a higher threshold encourages the model to invoke search more often. However, when ρ becomes too large (e.g., 0.9), $\mathbf{F1}_{\text{dec}}$ drops substantially. This suggests overuse of search, where the model chooses to search even when its parametric knowledge is sufficient, as reflected in the decrease in recall and the increase in precision.

ρ	**EM**	$\mathbf{F1}_{\text{dec}}$	Precision	Recall	Avg. Search
0.1	35.8	53.8	44.5	69.3	0.99
0.3	34.1	53.8	43.4	72.5	1.00
0.5	36.0	54.0	45.1	68.8	1.08
0.7	36.2	53.2	48.9	59.6	1.12
0.9	36.8	46.2	55.9	41.0	1.21

Table 9: Analysis on the effect of ρ for AdaSearch stage 2 (Section 4.2).

D.4 Hyperparameters of reward-shaping baselines

We analyze the reward-function hyperparameter of Naive Shaping and Decision Shaping, and the training hyperparameters of IKEA (Huang et al., 2025b) on Qwen2.5-3B-Instruct.

Naive shaping. We vary the value of λ in Equation 6 over 0.05, 0.1, 0.25, and report **EM**, $\mathbf{F1}_{\text{dec}}$, precision, recall, and the micro-averaged number of search calls. The results are shown in Table 10. As expected, the average number of search calls decreases as λ increases, due to the stronger penalty on invoking search. $\mathbf{F1}_{\text{dec}}$ increases, which can be attributed to the increase in recall, while precision remains relatively stable. Lastly, **EM** decreases with larger λ, possibly because the model underuses search, as reflected by the lower recall and reduced number of search calls.

λ	**EM**	$\mathbf{F1}_{\text{dec}}$	Precision	Recall	Avg. Search
0.05	33.2	47.3	36.1	76.8	0.77
0.1	33.0	47.8	35.7	81.3	0.74
0.25	32.1	48.7	35.9	85.6	0.67

Table 10: Analysis on the effect of λ for Naive Shaping baseline (Eq 6).

Decision shaping. We vary the value of α in Equation 7 over 0.1, 0.5, 1.0 and report **EM**, $\mathbf{F1}_{\text{dec}}$, precision, recall, and the micro-averaged number of search calls. The results are shown in Table 11. Increasing α improves $\mathbf{F1}_{\text{dec}}$ and recall, as expected from assigning a larger bonus to correct decisions. In contrast, precision, average search-call counts, and task **EM** decrease. These trends suggest that the model may underuse search, reflected in higher recall but lower precision and fewer search calls.

α	**EM**	$\mathbf{F1}_{\text{dec}}$	Precision	Recall	Avg. Search
0.1	36.2	49.5	44.7	57.5	1.12
0.5	34.1	51.3	42.0	70.3	1.00
1.0	33.8	52.5	41.6	76.4	0.98

Table 11: Analysis on the effect of α for Decision Shaping baseline (Eq 7).

IKEA. We compare our training hyperparameters for IKEA (Huang et al., 2025b) (Table 5) with those used in the original IKEA paper. In the original work, the authors train with both batch size and mini-batch size set to 256, a smaller learning rate of 5e-7, a much larger warm-up ratio of 0.75, a slightly larger group size of 16, and a total of 120 training steps (approximately 4 epochs). The performance comparison is shown in Table 12. We find that using the original hyperparameters results in IKEA being undertrained. Therefore, we adopt the hyperparameters in Table 5 for our experiments.

Setting	**EM**	$\mathbf{F1}_{\text{dec}}$	Precision	Recall	Avg. Search
Original	32.3	12.4	42.0	8.0	1.06
Ours (Table 5)	35.3	47.6	45.0	59.5	0.88

Table 12: Comparison of hyperparameters settings for IKEA (Huang et al., 2025b)

D.5 Comparison on efficiency

In this section, we compare the efficiency of each method in terms of the average number of search calls and the average latency, as shown in Table 13. All averages are computed over the full set of testing samples. In addition, we estimate the total search cost on the evaluation benchmarks using the pricing of the Google Search API (5 USD per 1,000 queries)[1].

Regarding the average number of search calls, AdaSearch substantially reduces unnecessary

[1] https://developers.google.com/custom-search/v1/overview#pricing

searches and costs compared to Search-R1, achieving a 34% reduction. AdaSearch also incurs a similar number of search calls as Decision Shaping, as both methods decide whether to invoke search on prompt-level based on self-knowledge. Compared to methods that explicitly minimize search calls (Naive Shaping and IKEA), AdaSearch uses slightly more searches. This behavior is expected, since AdaSearch does not directly optimize for minimizing tool calls. Instead, it focuses on avoiding unnecessary searches when parametric knowledge is sufficient, while still achieving a significant reduction relative to Search-R1.

In terms of average latency, although AdaSearch introduces an additional inference stage that explicitly decides whether to invoke search, its latency remains lower than that of Search-R1 by 20%. These results demonstrate that AdaSearch not only reduces the average number of searches (and thus API cost) but also decreases inference latency, while providing transparent and interpretable decision rationales that are valuable in real-world applications. Compared to reward-shaping baselines, AdaSearch incurs a slightly higher latency, but it achieves higher task performance (**EM**) and demonstrates better discrimination between necessary and unnecessary search calls, as reflected by $\mathbf{F1}_{\text{dec}}$ (Table 2).

Finally, our framework is orthogonal to reward-shaping methods. Additional reward terms can be incorporated to penalize undesirable search behaviors, such as issuing duplicate queries. We leave a detailed exploration of this direction to future work.

Method	Avg. Search	Est. Costs	Avg. Latency
Qwen2.5-3B-Instruct			
Search-R1	1.64	424.3	0.111
Naive Shaping	0.77	198.4	0.059
Decision Shaping	1.00	259.0	0.074
IKEA	0.88	228.6	0.066
AdaSearch	1.08	279.2	0.089

Table 13: Average number of searches, their estimated costs, and the average latency.

E AdaSearch-E2E

We unify the two stages in AdaSearch to obtain an end-to-end variant, AdaSearch-E2E. The overall training pipeline is shown in Algorithm 2. For each training instance (x, y), we construct three variants of the input by applying the parametric knowledge prompt s_{param}, the search prompt s_{search}, and the decision-making prompt s_{decision}.

The pseudo labels for decision making are generated on the fly: for each training instance (x, y), we estimate the empirical solve rate p from $\mathcal{R}_{\text{param}}$ using $\mathbf{SubEM}$, and assign pseudo labels according to whether p exceeds a predefined threshold ρ. The reward function is simply the binary outcome reward, and the remaining optimization follows GRPO (Shao et al., 2024).

The hyperparameters largely follow those of our two-stage method (Table 6), except for the batch and mini-batch sizes. We search over {(256, 128), (512, 256)} and use (256, 128) for Qwen and (512, 256) for Llama.

Algorithm 2 AdaSearch-E2E Training

Require: base policy LLM π_{θ_b}; search engine $\mathcal{E}$; training set $\mathcal{D} = (x_i, y_i)$; prompts s_{param}, s_{search}, s_{decision}; string concatenation $[\cdot, \cdot]$; solve-rate threshold ρ.

Initialize policy $\pi_\theta \leftarrow \pi_{\theta_b}$
/* End-to-End Training */
for iteration = 1 to T **do**
  Sample a batch $\mathcal{D}_b \sim \mathcal{D}$.
  $\mathcal{R} \leftarrow \{\}$; $\mathcal{A} \leftarrow \{\}$.
  for each $(x_i, y_i) \sim \mathcal{D}_b$ **do**
    for $g \in \{\text{param}, \text{search}, \text{decision}\}$ **do**
      /* Rollout Phase */
      if g is param **then**
        Generate $\mathcal{R}_g^i = \{\tau_{\text{param}}^n\}_{n=1}^N$, where $\tau_{\text{param}}^n \sim \pi_\theta(\cdot \mid [s_{\text{param}}, x_i])$.
      else if g is decision **then**
        Generate $\mathcal{R}_g^i = \{\tau_{\text{decision}}^k\}_{k=1}^K$, where $\tau_{\text{decision}}^k \sim \pi_\theta(\cdot \mid [s_{\text{decision}}, x_i])$.
      else
        Generate $\mathcal{R}_g^i = \{\tau_{\text{search}}^m\}_{m=1}^M$, where $\tau_{\text{search}}^m \sim \pi_\theta(\cdot \mid [s_{\text{search}}, x_i]; \mathcal{E})$.
      end if
      Compute rewards For all $\tau \in \mathcal{R}_g^i$ with Eq 4.
      Compute group-wise advantages $\mathcal{A}_g^i$ For $\mathcal{R}_g^i$.
      $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_g^i$; $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}_g^i$. // Aggregate rollouts & advantages
    end for
  end for
  Update π_θ using GRPO with $\mathcal{R}$ and $\mathcal{A}$.
end for
return π_θ

F Prompt Templates

Knowledge-Probing System Prompt s_{probing}

You are a helpful assistant. Your goal is to answer questions using step-by-step reasoning. Use only your internal knowledge to recall and connect relevant information. Do NOT assume access to external tools or documents. Provide the final answer only after completing your reasoning, following the output instructions below.

Output Instructions
1. Wrap your final answer within <answer> and </answer> tags. For example: <answer>Wilhelm Conrad Röntgen</answer>.
2. Only include the direct final answer within these tags; place all explanations and reasoning outside.

Table 14: Knowledge-probing system prompt for dataset construction.

Search System Prompt s_{search}

Answer the given question. You must conduct reasoning 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 <output> and </output>. You can search as many times as you 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>xxx</answer>.

Table 15: System prompt for search, adapt from Jin et al. (2025).

Parametric-knowledge System Prompt s_{param}

Answer the given question. Always conduct and show your step-by-step reasoning first, then give the final answer. When providing the answer, wrap it inside <answer> and </answer> without detailed illustrations. For example, <answer>xxx</answer>.

Table 16: Parametric-knowledge system prompt. This prompt is used for CoT and RL w/o Search baselines as well.

Decision-Making System Prompt s_{decision}

Your goal is to assess whether your knowledge is sufficient to answer the given question. Think step-by-step and then output the final assessment by following the instructions below:

Output Instructions
1. Always output your step-by-step reasoning first.
2. After your reasoning, output a single assessment token wrapped within <assessment> ... </assessment> tags.
   - If you are confident you can answer the question directly, output: <assessment>yes</assessment>
   - Otherwise, output: <assessment>no</assessment>
3. Only include the final assessment token within these tags; place all reasoning outside the tags.

Table 17: Decision-making system prompt.

System Prompt for Reward-shaping Baselines
Answer the given question. You must conduct reasoning 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 <output> and </output>. You need to make every search call count and gain helpful results. If you find no further external knowledge needed, you can directly provide the answer inside <answer> and </answer> without detailed illustrations. For example, <answer>xxx</answer>.

Table 18: System prompt for reward-shaping baselines (Naive Shaping, Awareness Shaping, and IKEA), adapt from Wang et al. (2025)

System Prompt for Direct Answer Baseline
Answer the given question. When providing the answer, wrap it inside <answer> and </answer> without detailed illustrations. For example, <answer>xxx</answer>.

Table 19: System prompt for direct answer baseline.

System Prompt for RAG Baseline
Answer the given question based on the provided documents. When providing the answer, wrap it inside <answer> and </answer> without detailed illustrations. For example, <answer>xxx</answer>.

Table 20: System prompt for RAG baseline.

Default User Prompt Template
Question: {question}

Table 21: Default user prompt template, where `question` will be replaced with the actual input question.

RAG User Prompt Template
Doc 1: {document_1} Doc 2: {document_2} Doc 3: {document_3} Question: {question}

Table 22: RAG User prompt template, where `document_i` and `question` will be replaced with the retrieved document and the actual input question, respectively.

Retry Prompt (Search)
My previous action is invalid. If I want to search, I should put the query between <search> and </search>. If I want to give the final answer, I should put the answer between <answer> and </answer>. Let me try again.

Table 23: Retry prompt used when the model produces an invalid action under the search prompt s_{search}.

Retry Prompt (Parametric-knowledge)
I should put the answer between <answer> and </answer>.

Table 24: Retry prompt used when the model fails to wrap the answer in <answer>...</answer> under the parametric-knowledge prompt s_{param}.

Retry Prompt (Parametric-knowledge; When search is invoked)
My previous action is invalid. I should rely on my own step-by-step reasoning to solve the problem and then put the answer between <answer> and </answer>. Let me try again.

Table 25: Retry prompt used when the model produces a search call under the parametric-knowledge prompt s_{param}.

CoT Monitoring Prompt Template (gpt-5.4-mini)

You are given the following fact-seeking Question and the corresponding Trajectory from a search agent. The search agent can access a search engine by wrapping its search query with the tags <search>...</search>. The documents will be returned within the tags <output>...</output>. At the end of the trajectory, the agent will provide its answer within the tags <answer>...</answer>. Tokens outside of these tags are the reasoning tokens of the agent.

Question
`{question}`

Agent Trajectory
`{trajectory}`

Instructions
Your task is to determine if the agent trajectory contains explicit reflection on whether it should invoke the search engines, or whether its own knowledge is sufficient to answer the question, based on the following criteria:

Criteria:
- Did the agent explicitly reason about its own knowledge or relevant facts before deciding whether to invoke search?
- Did the agent produce an explicit judgment on whether its own knowledge is sufficient (e.g., "I can answer this directly based on my knowledge" or "I do not have sufficient knowledge to answer this")?

Notice that simply invoking search without first assessing whether its own knowledge is sufficient is not an explanation, as it does not reflect on the agent's own knowledge or its limitations.
Notice that simply recalling facts and providing the final answer is not an explanation, as it does not include an explicit judgment on whether the agent's own knowledge is sufficient. However, if the agent recalls relevant facts and then explicitly concludes that its knowledge is or is not sufficient to answer the question, this counts as a valid reflection.

Grading Report
Provide an explanation of how the Trajectory is rated against the criteria above. Please only consider the criteria mentioned above and do not consider anything else.

Final Answer Selected from one of the options listed below.
A. The agent explicitly reflected on whether its own knowledge is sufficient to answer the question.
B. The agent did not explicitly reflect on whether its own knowledge is sufficient to answer the question.

Your response should be like: your grading report + final answer.

Table 26: CoT monitoring prompt using gpt-5.4-mini as the judge for interpretability evaluation. The placeholders `question` and `trajectory` are replaced with the actual input question and agent output, respectively.

CoT Monitoring Prompt Template (Claude 4.6 Sonnet)

You are given the following fact-seeking Question and the corresponding Trajectory from a search agent. The search agent can access a search engine by wrapping its search query with the tags <search>...</search>. The documents will be returned within the tags <output>...</output>. At the end of the trajectory, the agent will provide its answer within the tags <answer>...</answer>. Tokens outside of these tags are the reasoning tokens of the agent.

Question
`{question}`

Agent Trajectory
`{trajectory}`

Instructions
Your task is to determine if the agent trajectory contains explicit reflection on whether it should invoke the search engines, or whether its own knowledge is sufficient to answer the question, based on the following criteria:

Criteria:
- Did the agent explicitly reason about its own knowledge or relevant facts before deciding whether to invoke search?
- Did the agent produce an explicit judgment on whether its own knowledge is sufficient (e.g., "I can answer this directly based on my knowledge" or "I do not have sufficient knowledge to answer this")?

Notice that simply invoking search without first assessing whether its own knowledge is sufficient is not an explanation, as it does not reflect on the agent's own knowledge or its limitations.
Notice that simply recalling facts and providing the final answer is not an explanation, as it does not include an explicit judgment on whether the agent's own knowledge is sufficient. However, if the agent recalls relevant facts and then explicitly concludes that its knowledge is or is not sufficient to answer the question, this counts as a valid reflection.

Grading Report
Provide an explanation of how the Trajectory is rated against the criteria above. Please only consider the criteria mentioned above and do not consider anything else.

Final Answer
Selected from one of the options listed below.
A. The agent explicitly reflected on whether its own knowledge is sufficient to answer the question.
B. The agent did not explicitly reflect on whether its own knowledge is sufficient to answer the question.

Your response should be like: your grading report, followed by your final answer on the last line in this exact format: "Final Answer: [A]" or "Final Answer: [B]".

Table 27: CoT monitoring prompt using Claude 4.6 Sonnet as the judge for interpretability evaluation. The placeholders `question` and `trajectory` are replaced with the actual input question and agent output, respectively. Differences from the prompt for gpt-5.4-mini (Table 26), which help parse model responses more easily, are highlighted in red.

G Case Studies

G.1 Case 1: Parametric Knowledge is Sufficient

Input Question (Answer: Christian Louboutin)
Which designer is famous for his/her red-soled footwear?

Table 28: Input question. The answer is “Christian Louboutin".

AdaSearch Stage-1 Output (Answer: Christian Louboutin)
Step 1: I need to think of a designer known for red-soled footwear. Step 2: One famous designer known for red-soled footwear is Christian Louboutin. Step 3: Christian Louboutin is a French fashion designer who is renowned for his distinctive red-soled high-heeled shoes. Step 4: Based on this information, I can answer the question. <assessment>yes</assessment>

Table 29: AdaSearch stage-1 output. The model explicitly evaluates its own knowledge and concludes that it is sufficient.

AdaSearch Stage-2 Output (Answer: Christian Louboutin)
To determine which designer is famous for his/her red-soled footwear, we need to consider some of the most well-known designers and their iconic designs. 1. Christian Louboutin is a renowned French fashion designer who is famous for creating red-soled high-heeled shoes. 2. Other designers like Jimmy Choo and Alexander McQueen have also designed red-soled shoes, but Christian Louboutin is particularly known for this specific style. Given this information, the designer famous for his/her red-soled footwear is Christian Louboutin. <answer>Christian Louboutin</answer>

Table 30: AdaSearch stage-2 output. The model correctly answer the question using parametric knowledge.

Search-R1 Output (Answer: Christian Louboutin)

To find out which designer is famous for their red-soled footwear, I will need to search for information about notable footwear designers and their signature styles.

<search>Which designer is famous for red-soled footwear</search>
<output>
Doc 1: "Christian Louboutin"
Christian Louboutin Christian Louboutin (; born 7 January 1964) is a French fashion designer whose high-end stiletto footwear incorporates shiny, red-lacquered soles that have become his signature. Initially a freelance designer for fashion houses, he started his own shoe salon in Paris, with his shoes finding favor with celebrity clientele. He has partnered with other organizations for creative projects including limited edition pieces, gallery exhibits, and even a custom bar. His company has since branched out into men's footwear, handbags, fragrances and makeup. Louboutin was born and raised in Paris's 12th arrondissement. He was the only son of Roger, a
Doc 2: "Christian Louboutin"
tempered to 28,000 Mexican pesos (1471.89 dollars). Christian Louboutin Christian Louboutin (; born 7 January 1964) is a French fashion designer whose high-end stiletto footwear incorporates shiny, red-lacquered soles that have become his signature. Initially a freelance designer for fashion houses, he started his own shoe salon in Paris, with his shoes finding favor with celebrity clientele. He has partnered with other organizations for creative projects including limited edition pieces, gallery exhibits, and even a custom bar. His company has since branched out into men's footwear, handbags, fragrances and makeup. Louboutin was born and raised in Paris's 12th arrondissement. He
Doc 3: "Christian Louboutin"
on his high heel shoes, commonly referred to as ""sammy red soled shoes"". His signature red soles were first created in 1993. ""My sketches were not reproduced exactly as I had designed them and I couldn't figure out why. The two dimensional sketch was so powerful on paper, but when turned into a three-dimensional object, it was somehow lacking energy. Frustrated after having tried different things to liven up the design, I spontaneously grabbed my assistant's red nail polish and painted the sole. I instantly knew that this would be a success!"". Christian Louboutin's red-bottom colour code is Pantone 18-1663
</output>

From the search results, it is clear that Christian Louboutin is famous for his red-soled footwear.

<answer>Christian Louboutin</answer>

Table 31: Search-R1 output. The agent overuses search on problems where its parametric knowledge is sufficient.

Naive Shaping Output (Answer: Christian Louboutin)
To determine which designer is famous for their red-soled footwear, let's consider some prominent designers and their notable styles. One of the most famous designers known for red-soled footwear is Alexander McQueen. He was known for his bold and innovative designs, including red-soled shoes that became iconic in his collections. Given this information, the designer famous for his/her red-soled footwear is Alexander McQueen. <answer>Alexander McQueen</answer>

Table 32: Naive Shaping output. The model produces an incorrect answer without explicit self-knowledge assessment.

G.2 Case 2: Parametric knowledge is Insufficient

Input Question (Answer: Bardney)
Southrey is a village in the civil parish of which village in the West Lindsey district of Lincolnshire, England?

Table 33: Input question. The answer is “Bardney".

AdaSearch Stage-1 Output (Answer: Bardney)
Step 1: Identify the key information in the question. The question asks about a civil parish that contains the village of Southrey. Step 2: Recall that a civil parish is a territorial unit of local government in England, and it is often part of a larger administrative division such as a district. Step 3: Think about the West Lindsey district in Lincolnshire, England. I need to find which village in this district contains Southrey. Step 4: Without specific knowledge of the exact location, I would need to search for the civil parish that includes Southrey. However, I don’t have immediate access to this information. Step 5: Given the lack of specific knowledge about the exact civil parish containing Southrey, I cannot answer the question directly. <assessment>no</assessment>

Table 34: AdaSearch stage-1 output. After self-assessment, the model is not confident in answering using parametric knowledge.

AdaSearch Stage-2 Output (Answer: Bardney)

To determine the village in the West Lindsey district of Lincolnshire, England, that is the civil parish of Southrey, I will need to search for this information.

<search>Village in the West Lindsey district of Lincolnshire that is the civil parish of Southrey</search>
<output>
Doc 1: Bardney
consists of crafts, trade and community stands, funfair rides, bar and BBQ, sports, children's races, car boot sales, classic vehicles, and an Exemption Dog Show. Originally part of the Wraggoe Wapentake, and of the Lincoln poor law union in the Parts of Lindsey, the parish is now part of West Lindsey District, and Lincolnshire County Council. The civil parish has become part of a group that consists of Bardney, Apley and Stainfield parishes. The villages of Southrey and Kingthorpe are also included. The 2011 electoral arrangements are: Bardney lies between 7 and 17 metres above sea level, on the edge
Doc 2: Snelland
Snelland Snelland is a village and civil parish in the West Lindsey district of Lincolnshire, England It is situated approximately north-east from the city and county town of Lincoln and about from the town of Market Rasen. The civil parish includes the hamlet of Swinthorpe. The population at the 2011 census was included in the civil parish of Friesthorpe. Snelland is listed in ""Domesday Book"" of 1086 as ""Sneleslunt"", with seven households. The nearby deserted medieval village of Swinthorpe is listed as ""Sonetorp"", with three households. There is evidence that a Cistercian grange existed at Swinthorpe, attached to Kirkstead Abbey.
Doc 3: "South Thoresby"
South Thoresby South Thoresby is a village and civil parish in the East Lindsey district of Lincolnshire, England. It is situated north-east from Horncastle and south-east from Louth. South Thoresby is about east from the A16 in the Lincolnshire Wolds. The civil parish extends much further to the south, over the A16 and to include the hamlets of Calceby (a former medieval village) and Driby, extending southwards to border with Langton by Spilsby. To the south-west is Haugh. It lies close to the northern boundary with Aby with Greenfield. The parish council administers Swaby, South Thoresby and Haugh, although these
</output>

From the search results, it is clear that Southrey is a village in the civil parish of Bardney. Therefore, the village in the West Lindsey district of Lincolnshire, England, that is the civil parish of Southrey is Bardney.

<answer>Bardney</answer>

Table 35: AdaSearch stage-2 output. The model invokes search and produces a correct answer.

Naive Shaping Output (Answer: Bardney)
To determine the village in the West Lindsey district of Lincolnshire, England, that is the civil parish of Southrey, we need to find information about the civil parishes and their boundaries. Southrey is a village in the civil parish of Little Sleaford. Little Sleaford is a village located in the West Lindsey district of Lincolnshire, England. Therefore, the civil parish of Southrey is Little Sleaford. <answer>Little Sleaford</answer>

Table 36: Naive Shaping output. The model produces an incorrect answer without explicit self-knowledge assessment.

G.3 Case 3: AdaSearch Underuse Search

Input Question (Answer: Mary Elizabeth Braddon)
Who is the author of Lady Audley's Secret?

Table 37: Input question. The answer is "Mary Elizabeth Braddon".

AdaSearch Stage-1 Output (Answer: Mary Elizabeth Braddon)
Step 1: I need to recall the author of the novel "Lady Audley's Secret." Step 2: I know that "Lady Audley's Secret" is a novel. Step 3: I remember that the author of "Lady Audley's Secret" is Mrs. Wilkie Collins. Step 4: I can confirm that Mrs. Wilkie Collins is indeed the author of this novel. Step 5: Based on my knowledge, I can answer the question directly. <assessment>yes</assessment>

Table 38: AdaSearch stage-1 output. The model holds a false belief, which leads to an incorrect self-assessment.

AdaSearch Stage-2 Output (Answer: Mary Elizabeth Braddon)
To determine the author of "Lady Audley's Secret," we need to consider the historical context and the well-known works of that era. "Lady Audley's Secret" is a novel published in 1862. The author of this novel is Mrs. Wilkie Collins, a prominent English novelist of the Victorian era. Therefore, the author of "Lady Audley's Secret" is <answer>Mrs. Wilkie Collins</answer>

Table 39: AdaSearch stage-2 output. The model maintains its false belief, resulting in insufficient search.

G.4 Case 4: AdaSearch Overuse Search

Input Question (Answer: Politician)
What is Javier Alva Orlandini's occupation?

Table 40: Input question. The answer is "Politician".

AdaSearch Stage-1 Output (Answer: Politician)
Step 1: I need to determine Javier Alva Orlandini's occupation. Step 2: I do not have any information about Javier Alva Orlandini in my knowledge base. Step 3: Without specific information about this person, I cannot answer the question directly. <assessment>no</assessment>

Table 41: AdaSearch stage-1 output. The model is not confident in its self-knowledge and therefore abstains.

AdaSearch Stage-2 Output (Search) (Answer: Politician)

To determine Javier Alva Orlandini's occupation, I will need to search for information about him. Let me start with a search.
<search>Javier Alva Orlandini occupation</search>
<output>
Doc 1: "Javier Alva Orlandini"
Javier Alva Orlandini Javier Alva Orlandini (born December 11, 1927) is a lawyer and a Peruvian politician. A prominent member of the political party Popular Action, he serves as the President of the party. He served as 2nd Vice President of Peru during the government of Fernando Belaúnde from 1980 to 1985.. Throughout his life he became a Deputy (1963-1968), Minister of Government and Police (1965-1966), Senator (1980-1985 and 1990-1992), President of the Senate (1981-1982) and Congressman (1995-2000). In 2002, he became a Magistrate of the Constitutional Court, at the same time being elected by Congress as the President of
Doc 2: "Luis Orlandini"
Luis Orlandini Luis Orlandini (born 1964) is a Chilean guitarist and professor at the University of Chile and the Pontifical Catholic University of Chile. Orlandini studied guitar with Ernesto Quezada at the University of Chile and then with Eliot Fisk at the School of Music of Cologne. In 1989, he obtained First Place at the ARD International Music Competition in Munich, thus initiating an international career that has led him to perform in various countries and record with different European labels. He performs mainly as a soloist, but has also played with groups of different natures of chamber music, one
Doc 3: "Javier Alva Orlandini"
the Court for a 3-year term. He was a candidate for the Presidency of the Republic in 1985, but got 4% of the popular vote, being defeated by Alan García. He later served as President of Peru's constitutional tribunal. He has written several books, including ""Responding to the Dictatorship"", ""Yesterday, Today and Tomorrow"", the ""Vicious Circle"", ""I Minister"", ""Word of Honor"" and ""Yes I swear"", and numerous legal articles. The audits, representative legislative, legal, academic and professional are of extensive quality and are recognized by the Peruvian people and their various institutions that have given many honorary degrees. Javier Alva
</output>

Based on the search results, Javier Alva Orlandini is a lawyer and a Peruvian politician. He served as the President of the party Popular Action and was the 2nd Vice President of Peru during the government of Fernando Belaúnde from 1980 to 1985.

<answer>Politician</answer>

Table 42: AdaSearch stage-2 output. The model produces a correct answer after search.

AdaSearch Stage-2 Output (Parametric Knowledge) (Answer: Politician)

To determine Javier Alva Orlandini's occupation, we need to look up information about him. Given that the question does not provide specific details, we can infer based on common knowledge or a search.

Javier Alva Orlandini is a Mexican politician who has held various positions in the Mexican government. Given his background, it is most likely that he is a politician.

However, without a specific source, we can't be certain. For the purpose of this question, we will assume his occupation is based on his known roles in Mexican politics.

<answer>Politician</answer>

Table 43: Model output under the parametric-knowledge system prompt s_{param}. The model is uncertain about its answer, even though the answer is correct.

G.5 Case 5: Multi-hop Failure of AdaSearch

Input Question (Answer: 1978)
The director of Panic 5 Bravo was born in what year?

Table 44: Input question. The answer is “1978".

AdaSearch Stage-1 Output (Answer: 1978)
Step 1: To answer this question, I need to know the director of the movie "Panic 5 Bravo". However, there is no widely known movie or film titled "Panic 5 Bravo" that I am aware of. This suggests it might be a fictional or obscure title. Step 2: Without knowing the director of this specific movie, I cannot determine the year of birth of the director. Step 3: Since I do not have the necessary information to answer the question, I cannot provide a year of birth for the director. <assessment>no</assessment>

Table 45: AdaSearch stage-1 output. The model abstains because it lacks sufficient information.

AdaSearch Stage-2 Output (Answer: 1978)

To find out the year the director of "Panic 5 Bravo" was born, I will need to gather information about this film and its director. Let's start by searching for more details about the film "Panic 5 Bravo".
<search>The director of Panic 5 Bravo and the year of birth</search>
<output>
Doc 1: "Panic (2000 film)"
Panic (2000 film) Panic is a 2000 American crime film directed by Henry Bromell and starring William H. Macy, Neve Campbell, Tracey Ullman, John Ritter and Donald Sutherland. Despite not being a box office success, ""Panic"" received universal critical acclaim from critics, some of which called it one of the best movies of 2000. The performances of William H. Macy, John Ritter, and Donald Sutherland were hailed by critics. Alex (Macy), a sad-eyed mournful man, goes into psychotherapy disclosing that he is a hit man and also tells the doctor (Ritter) after a few sessions that he is attracted to
Doc 2: "Panic (2000 film)"
originality from the tone: quirky yet convincing, irreverent yet moral."" Panic (2000 film) Panic is a 2000 American crime film directed by Henry Bromell and starring William H. Macy, Neve Campbell, Tracey Ullman, John Ritter and Donald Sutherland. Despite not being a box office success, ""Panic"" received universal critical acclaim from critics, some of which called it one of the best movies of 2000. The performances of William H. Macy, John Ritter, and Donald Sutherland were hailed by critics. Alex (Macy), a sad-eyed mournful man, goes into psychotherapy disclosing that he is a hit man and also tells the doctor
Doc 3: "A Town Called Panic (film)"
A Town Called Panic (film) A Town Called Panic () is a 2009 internationally co-produced stop-motion animated adventure fantasy comedy family film directed by Stéphane Aubier and Vincent Patar and co-written by Aubier, Guillaume Malandrin, Patar, and Vincent Tavier. The film is based on the TV series of the same name and stars Aubier, Jeanne Balibar, Nicolas Buysse, Véronique Dumont, Bruce Ellison, Frédéric Jannin, Bouli Lanners, and Patar, among others. It premiered at the 2009 Cannes Film Festival and was the first stop-motion film to be screened at the festival. The film was released theatrically in Belgium on 17 June
</output>

From the search results, it seems "Panic 5 Bravo" is not a widely known film. However, the first result mentions a film called "Panic (2000 film)" directed by Henry Bromell. Given this information, I will now search for Henry Bromell's birth year.

Table 46: AdaSearch stage-2 output (part 1). The model invokes search but makes a faulty inference (highlighted in red).

AdaSearch Stage-2 Output (Continued) (Answer: 1978)

<search>Henry Bromell birth year</search>
<output>
Doc 1: "Henry Bromell"
Henry Bromell Alfred Henry Bromell (September 19, 1947 – March 18, 2013) was an American novelist, screenwriter, and director. Bromell joined the crew of NBC police drama """" in 1994. He served as a writer and co-executive producer for the show's third season. He contributed to writing seven episodes for the season. He was promoted to executive producer for the fourth season and wrote a further 17 episodes. He scaled back his involvement with the fifth season and became a consulting producer. He wrote a further two episodes before leaving the crew at the end of the season in 1997.
Doc 2: "Henry Bromell"
screenwriter and director Caroline Thompson. He then married writer Trish Soodik, who died of cancer in January 2009; they had a son, William. His third wife was Sarah. Bromell died March 18, 2013 of a heart attack, at UCLA Santa Monica hospital at age 65. Henry Bromell Alfred Henry Bromell (September 19, 1947 – March 18, 2013) was an American novelist, screenwriter, and director. Bromell joined the crew of NBC police drama """" in 1994. He served as a writer and co-executive producer for the show's third season. He contributed to writing seven episodes for the season. He was promoted
Doc 3: "Henry Bromell"
He contributed to a total of 26 episodes as a writer over three seasons with the series. He returned as a co-writer and co-executive producer for the feature-length follow-up ""Homicide: The Movie"" in 2000. He wrote and produced for many television series, including ""Chicago Hope"", ""Northern Exposure"", ""Homicide: Life on the Street"", ""Brotherhood"", ""Carnivàle"", and ""Rubicon"". He was a consulting producer, and later Executive Producer on the Showtime series ""Homeland"" at the time of his death and wrote four episodes: ""The Good Soldier"", ""Representative Brody"", ""Q&A"", and ""Broken Hearts"". He was awarded a for ""The Good Soldier"", and he was
</output>

From the search results, Henry Bromell was born in 1947. Therefore, the director of "Panic (2000 film)" was born in 1947.

<answer>1947</answer>

Table 47: AdaSearch stage-2 output. Continuing from part 1, the model uses the wrong director as the search query, which ultimately leads to an incorrect answer.