

# An Empirical Study on Reinforcement Learning for Reasoning-Search Interleaved LLM Agents

Bowen Jin<sup>1\*</sup>, Jinsung Yoon<sup>2</sup>, Priyanka Kargupta<sup>1</sup>, Sercan Ö. Arık<sup>2</sup>, Jiawei Han<sup>1</sup>

<sup>1</sup> University of Illinois at Urbana-Champaign

<sup>2</sup> Google Cloud AI Research  
bowenj4@illinois.edu

## Abstract

Reinforcement learning (RL) has demonstrated strong potential in training large language models (LLMs) capable of complex reasoning for real-world problem solving. More recently, RL has been leveraged to create sophisticated LLM-based search agents that adeptly combine reasoning with search engine use. While the use of RL for training search agents is promising, the optimal design of such agents remains not fully understood. In particular, key factors—such as (1) reward formulation, (2) the choice and characteristics of the underlying LLM, and (3) the role of the search engine in the RL process—require further investigation. In this work, we conduct comprehensive empirical studies to systematically investigate these and offer actionable insights. We highlight several key findings: format rewards are effective in improving final performance, whereas intermediate retrieval rewards have limited impact; the scale and initialization of the LLM (general-purpose vs. reasoning-specialized) significantly influence RL outcomes; and the choice of search engine plays a critical role in shaping RL training dynamics and the robustness of the trained agent during inference. These establish important guidelines for successfully building and deploying LLM-based search agents in real-world applications. Code is available at <https://github.com/PeterGriffinJin/Search-RL>.

## 1 Introduction

Large language models (LLMs) [57] have demonstrated exceptional performance across a range of natural language processing tasks, including question answering [26], summarization [56], and open-ended text generation [8]. Recently, inspired by developments such as DeepSeek-R1 [9], reinforcement learning (RL) [14, 39] has been increasingly applied to LLMs to unlock more advanced reasoning capabilities [45]. LLMs trained via RL have shown strong performance in tasks requiring logical reasoning [49] and visual understanding [55], with promising applications emerging in specialized domains such as finance [25] and medicine [19]. However, these models often remain limited to self-contained reasoning and lack the ability to interact with external environments or leverage external tools. To address this, recent work has explored using RL to train LLMs as interactive agents, capable of engaging with external environments and invoking tools, as demonstrated in multi-turn game tasks [44] and user interface control tasks [24, 48].

A key agentic application for LLMs is *search*, where models decompose complex problems, perform multi-turn reasoning, and iteratively interact with search engines to retrieve relevant information. Prior work has explored prompt-based approaches [11, 41] and supervised fine-tuning (SFT) methods [3, 34] to equip LLMs with search capabilities. However, these approaches face key limitations: LLMs typically lack strong search proficiency from pretraining alone, and SFT requires costly manual

\*Part of the work was done while Bowen was a student researcher at Google.

annotation of intermediate reasoning trajectories, making it challenging to scale. In contrast, recent studies [4, 13, 37, 58] demonstrate that *RL with outcome-based rewards* offers an effective alternative for training LLMs to perform reasoning and search in an interleaved manner—forming what is referred to as an *LLM-based search agent*. This training paradigm enhances the model’s ability to interact with search engines while eliminating the need for explicit supervision of intermediate reasoning steps, thus enabling scalable and more flexible agent learning.

While recent RL-based methods have demonstrated the potential to train effective LLM-based search agents, several key questions remain underexplored: (1) *How does reward design affect search agent training?* Although prior work [13] shows that outcome-based rewards alone can activate reasoning and search capabilities, it is unclear whether auxiliary rewards such as format rewards (which signal adherence to the agentic action format [9]) or intermediate retrieval rewards (which iteratively incentivize outcome-relevant retrievals [22]) can further enhance performance. (2) *How does the backbone LLM influence RL dynamics?* As suggested by [6], the choice of the base model is critical. Factors such as model scale (e.g., 3B vs. 32B) and type (e.g., general-purpose vs. reasoning-specialized) can significantly impact the learning dynamics. (3) *How does the search engine choice affect the learned agent?* This includes understanding how the quality of different search engines influence RL training dynamics and whether the resulting agent remains robust when the retrieval system is changed at inference time.

In this paper, we conduct comprehensive empirical studies to address the aforementioned research questions. Our key findings are summarized as follows: (1) *Reward Design*. We observe that incorporating a format reward significantly improves performance, particularly when training from a base LLM rather than an instruction-tuned one. In contrast, intermediate retrieval rewards do not yield consistent performance improvements, suggesting limited utility. (2) *Underlying LLM Backbone*. General-purpose LLMs outperform reasoning-specialized LLMs in RL settings, likely due to the latter’s weaker instruction-following capabilities at the early stages of training. Furthermore, scaling up the backbone model generally enhances final performance, although with diminishing returns. (3) *Search Engine Choice*. The quality of the search engine used during training strongly influences RL dynamics. Training with a non-informative search engine (e.g., random noise) leads the agent to avoid retrieval altogether, while a weak engine (e.g., BM25 [33]) results in frequent but less efficient search calls. In contrast, strong engines (e.g., dense retrievers) yield more stable learning. At inference time, the search agent is generally robust to diverse retrieval systems, and stronger search engines consistently lead to better downstream performance.

## 2 Related Works

### 2.1 Large Language Models and Reinforcement Learning

RL [14, 39] offers a principled framework for sequential decision-making, where an agent optimizes its behavior by interacting with an environment and maximizing cumulative rewards. In the context of LLM tuning, RL was popularized by Reinforcement Learning from Human Feedback (RLHF) [12, 17, 28], which first trains a reward model from human preference data [20] and then fine-tunes the policy LLM via Proximal Policy Optimization (PPO). While PPO enables high-quality alignment, it incurs significant computational overhead due to iterative optimization steps. Recent efforts to strike a better balance include Group Relative Policy Optimization (GRPO) [35], which removes the dependency on a learned value function by leveraging group-based baseline estimation, and RLOO [2], a simplified variant of REINFORCE [47] tailored for LLM training. More recently, DAPO [53] extends GRPO by introducing four key innovations tailored for large-scale LLM reinforcement learning: clip-higher reward capping, dynamic sampling for adaptive data efficiency, a token-level objective for finer-grained supervision, and overlong reward shaping to handle extended sequences. In parallel, VAPO [54] builds upon PPO by proposing a value-model-augmented framework, incorporating value pretraining, a decoupled Generalized Advantage Estimator (GAE), and an auxiliary language modeling loss on positive examples to improve credit assignment and stability. While these advancements have significantly enhanced the scalability and efficiency of RL-based LLM tuning, their application to LLM-driven search and reasoning tasks remains underexplored, highlighting a critical direction for future exploration.

## 2.2 Large Language Models as Search Agents

LLMs [1, 40, 57] have demonstrated strong reasoning capabilities [9] but often struggle with hallucinations and insufficient domain-specific knowledge [21, 29]. To address these, recent efforts explore integrating LLMs with search engines to enable dynamic access to external knowledge. A prominent direction is to treat search engines as interactive tools that LLMs can call during inference [34]. This search-as-a-tool paradigm allows models to iteratively formulate queries, retrieve relevant content, and revise their responses based on external evidence [41]. Prompt-based methods such as IRCOT [41] and ReAct [52] enable interleaved reasoning and retrieval, while Toolformer [34] and self-RAG [3] uses supervised fine-tuning to learn when and how to call a search engine. However, these methods often depend on high-quality demonstration data, which is difficult to scale. As an alternative, RL offers a scalable and data efficient solution. Inspired by [9], recent works [4, 13, 37, 58] show that LLMs can acquire complex reasoning and search behaviors through RL only using outcome-based rewards. Despite this promise, there is still a lack of in-depth empirical study of different design choices in RL for LLM search agents.

## 3 Preliminary

**Reasoning-Search Interleaved LLM Agent (i.e., LLM-based Search Agent)** [13, 36]. We consider an agentic LLM that performs interleaved, *multi-turn* reasoning and search engine interactions. In each iteration, the LLM-based search agent first engages in *reasoning* to analyze the current context and identify what additional information is needed. It then formulates a search query to *retrieve* relevant external information, which is incorporated into the *context* for subsequent reasoning. This iterative process continues until the model determines that sufficient information has been gathered to produce a final *answer*. The overall interaction follows a multi-turn reasoning–search loop: reasoning  $\rightarrow$  search  $\rightarrow$  context  $\rightarrow$  reasoning  $\rightarrow$  search  $\rightarrow \dots \rightarrow$  reasoning  $\rightarrow$  answering. To facilitate this process [52], the reasoning steps are enclosed within `<think> </think>`, search queries are wrapped in `<search> </search>`, retrieved information is inserted into `<information> </information>`, and the final answer is placed within `<answer> </answer>`.

**RL for Training an LLM-based Search Agent.** In [13, 58], they propose an RL objective to explicitly incorporate a search engine  $\mathcal{R}$  during optimization for LLM search agent training. The objective is formalized as:

$$\max_{\pi_{\theta}} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta}(\cdot | x; \mathcal{R})} [r_{\phi}(x, y)] - \beta \mathbb{D}_{\text{KL}} [\pi_{\theta}(y | x; \mathcal{R}) \| \pi_{\text{ref}}(y | x; \mathcal{R})], \quad (1)$$

where  $\pi_{\theta}$  denotes the trainable policy,  $\pi_{\text{ref}}$  is a fixed reference model,  $r_{\phi}$  represents the reward function, and  $\mathbb{D}_{\text{KL}}$  denotes the KL divergence. Here,  $x$  are sampled from the dataset  $\mathcal{D}$ , and  $y$  denote the output sequence interleaving reasoning steps with search engine retrievals.

In contrast to prior approaches that generate rollouts exclusively from the model  $\pi_{\theta}(\cdot | x)$  [28, 31], [13, 58] augment the generation process by interleaving retrievals via  $\pi_{\theta}(\cdot | x; \mathcal{R})$ , which can be interpreted as  $\pi_{\theta}(\cdot | x) \otimes \mathcal{R}$ , where  $\otimes$  denotes a retrieval-reasoning composition.

The reward function  $r_{\phi}$  serves as the primary optimization signal. [13, 58] employ a rule-based reward system focusing exclusively on **final outcome rewards**, which evaluate the correctness of the final answer. In factual reasoning tasks, correctness is assessed using exact string match (EM) evaluation:  $r_{\phi}(x, y) = \text{EM}(a_{\text{pred}}, a_{\text{gold}})$ , where  $a_{\text{pred}}$  is the predicted final answer extracted from the model’s response  $y$ , and  $a_{\text{gold}}$  is the ground-truth answer. In other words,

$$r_{\phi}(x, y) = \begin{cases} 1 & \text{if } a_{\text{pred}} = a_{\text{gold}}, \\ 0 & \text{if } a_{\text{pred}} \neq a_{\text{gold}}, \end{cases} \quad (2)$$

Although prior methods have demonstrated strong performance, there remains a notable gap in empirical studies systematically evaluating key design choices—specifically, the effectiveness of different reward formulations, the influence of underlying LLM characteristics, and the impact of search engine selection—on the reinforcement learning process for training search agents.

## 4 RL Rewards for LLM-based Search Agents

In [13, 58], an outcome-driven reward (i.e., string exact match) is employed through the RL process to guide the LLM on learning the reasoning and interleaved search engine calling functionality.

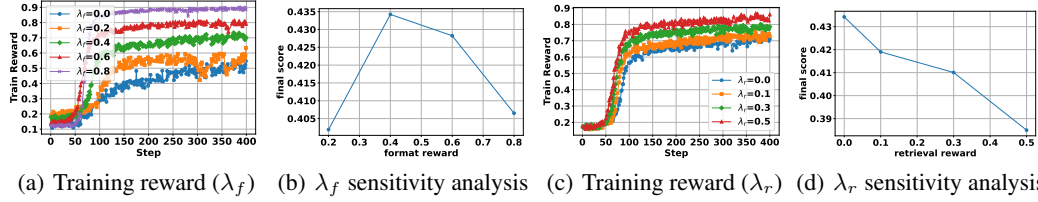


Figure 1: Empirical analyses on format reward and intermediate retrieval reward. (a) **Training reward curves** with varying **format reward** scaling factors ( $\lambda_f$ ); larger  $\lambda_f$  values lead to faster convergence. (b) **Impact of  $\lambda_f$**  on final model performance; a small  $\lambda_f$  is ineffective, while an excessively large  $\lambda_f$  may cause overfitting to format reward. (c) **Training reward curves** under different **intermediate retrieval reward** scaling factors ( $\lambda_r$ ); varying  $\lambda_r$  has limited effect on learning dynamics. (d) **Effect of  $\lambda_r$**  on final model performance; increasing  $\lambda_r$  degrades performance, suggesting that intermediate retrieval rewards are unnecessary, as the outcome reward sufficiently encourages effective query formulation. (LLM: Qwen2.5-7B-Base; RL Algorithm: PPO)

However, in search scenarios, the LLMs need to follow a specific format in order to call the search engine (*i.e.*, format reward) and the relevance of the intermediate search results can also guide the LLM on generating the proper queries (*i.e.*, intermediate retrieval reward).

#### 4.1 Format Reward

**Motivation.** When training an LLM-based search agent capable of reasoning and invoking external search engines, it is common to adopt the reasoning-action-observation workflow [52], where relevant content is wrapped within special tokens such as `<think>` `</think>`, `<search>` `</search>`, and `<information>` `</information>`. For instance, if the LLM fails to correctly format its search queries using `<search>` `</search>`, it cannot successfully trigger the search engine and retrieve the external information needed for problem solving. Thus, adhering to the prescribed format is critical for ensuring the effectiveness of the search agent. In this section, we explore how incorporating a format reward influences the RL training process of a search agent.

**Experimental Design.** In addition to the outcome reward defined in [13, 58], we introduce a format reward, resulting in the final reward function  $r_\phi(x, y)$ :

$$r_\phi(x, y) = \begin{cases} 1 & \text{if } a_{\text{pred}} = a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{True}, \\ 1 - \lambda_f & \text{if } a_{\text{pred}} = a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{False}, \\ \lambda_f & \text{if } a_{\text{pred}} \neq a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{True}, \\ 0 & \text{if } a_{\text{pred}} \neq a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{False}, \end{cases} \quad (3)$$

where  $f_{\text{format}}(\cdot)$  verifies whether the response  $y$  follows the correct reasoning-action-observation format, including the appropriate use of special tokens. We assign a reward of  $\lambda_f$  when the LLM generates an incorrect answer in the correct format, and a reward of  $1 - \lambda_f$  when the answer is correct but the format is incorrect. Details of the  $f_{\text{format}}(\cdot)$  implementation are provided in Appendix C. We follow [13] for the training and testing datasets and use exact match as the outcome reward. Detailed experimental settings can be found in Appendix G.1.

**Results.** Table 1 reports results across various datasets, LLM sizes, and RL algorithms. Detailed studies on  $\lambda_f$  using Qwen2.5-7B-Base and PPO are presented in Figures 1(a) and (b). We summarize the key findings as follows: (1) Adding a format reward consistently improves final model performance, particularly for base LLMs. This is because base LLMs lack strong instruction-following capabilities for search engine invocation, and the format reward helps mitigate this limitation. (2) Format reward accelerates RL convergence; larger  $\lambda_f$  values lead to faster convergence by explicitly guiding the model to issue correctly formatted search queries and interpret results effectively. (3) The choice of  $\lambda_f$  significantly impacts final performance. While a small  $\lambda_f$  is ineffective, an excessively large  $\lambda_f$  may cause overfitting, ultimately degrading final performance.

Table 1: Empirical study of the format reward. *Outcome only* refers to the RL variant with only the outcome reward. Base/Instruct refer to the version of the underlying LLM.  $\lambda_f = 0.2$  for 3B/14B and  $\lambda_f = 0.4$  for 7B. The best performance is set in bold.  $^\dagger$ / $^*$  represents in-domain/out-domain datasets.

Methods		General QA			Multi-Hop QA				Avg.
		NQ <sup>†</sup>	TriviaQA <sup>*</sup>	PopQA <sup>*</sup>	HotpotQA <sup>†</sup>	2wiki <sup>*</sup>	Musique <sup>*</sup>	Bamboogle <sup>*</sup>	
Qwen2.5-3B-Base/Instruct									
PPO	Outcome only (base)	0.406	0.587	0.435	0.284	0.273	0.049	0.088	0.303
	w. format reward	<b>0.428</b>	<b>0.607</b>	<b>0.459</b>	<b>0.371</b>	<b>0.387</b>	<b>0.150</b>	<b>0.323</b>	<b>0.389</b>
	Outcome only (instruct)	0.341	0.545	0.378	0.324	<b>0.319</b>	0.103	0.264	0.325
	w. format reward	<b>0.356</b>	<b>0.557</b>	<b>0.393</b>	<b>0.327</b>	0.314	<b>0.122</b>	<b>0.266</b>	<b>0.334</b>
GRPO	Outcome only (base)	0.421	0.583	0.413	0.297	0.274	0.066	0.128	0.312
	w. format reward	<b>0.429</b>	<b>0.602</b>	<b>0.435</b>	<b>0.372</b>	<b>0.383</b>	<b>0.148</b>	<b>0.307</b>	<b>0.382</b>
	Outcome only (instruct)	<b>0.397</b>	<b>0.565</b>	<b>0.391</b>	<b>0.331</b>	<b>0.310</b>	<b>0.124</b>	0.232	<b>0.336</b>
	w. format reward	0.346	0.552	0.371	0.297	0.300	0.098	<b>0.266</b>	0.319
Qwen2.5-7B-Base/Instruct									
PPO	Outcome only (base)	0.480	0.638	0.457	0.433	0.382	<b>0.196</b>	<b>0.432</b>	0.431
	w. format reward	<b>0.488</b>	<b>0.644</b>	<b>0.469</b>	<b>0.436</b>	<b>0.412</b>	0.187	0.403	<b>0.434</b>
	Outcome only (instruct)	<b>0.393</b>	<b>0.610</b>	0.397	0.370	<b>0.414</b>	0.146	0.368	<b>0.385</b>
	w. format reward	0.383	0.593	<b>0.399</b>	<b>0.376</b>	0.317	<b>0.151</b>	<b>0.371</b>	0.370
GRPO	Outcome only (base)	0.395	0.560	0.388	0.326	0.297	0.125	0.360	0.350
	w. format reward	<b>0.458</b>	<b>0.632</b>	<b>0.442</b>	<b>0.412</b>	<b>0.404</b>	<b>0.180</b>	<b>0.411</b>	<b>0.420</b>
	Outcome only (instruct)	<b>0.429</b>	<b>0.623</b>	<b>0.427</b>	<b>0.386</b>	<b>0.346</b>	<b>0.162</b>	<b>0.400</b>	<b>0.396</b>
	w. format reward	0.393	0.609	0.397	0.367	0.344	0.147	0.387	0.378
Qwen2.5-14B-Base/Instruct									
PPO	Outcome only (base)	0.486	0.676	<b>0.480</b>	<b>0.468</b>	<b>0.470</b>	<b>0.241</b>	<b>0.528</b>	<b>0.479</b>
	w. format reward	<b>0.499</b>	<b>0.680</b>	0.472	0.452	0.431	0.215	0.468	0.459
	Outcome only (instruct)	0.424	0.660	0.442	0.436	0.379	0.210	0.480	0.433
	w. format reward	<b>0.449</b>	<b>0.682</b>	<b>0.466</b>	<b>0.447</b>	<b>0.422</b>	<b>0.224</b>	<b>0.500</b>	<b>0.456</b>
GRPO	Outcome only (base)	0.415	0.680	0.488	0.451	0.461	0.230	0.508	0.462
	w. format reward	<b>0.500</b>	<b>0.693</b>	<b>0.500</b>	<b>0.481</b>	<b>0.488</b>	<b>0.261</b>	<b>0.516</b>	<b>0.491</b>
	Outcome only (instruct)	0.482	0.667	0.434	0.429	0.424	0.191	0.492	0.446
	w. format reward	<b>0.488</b>	<b>0.677</b>	<b>0.482</b>	<b>0.455</b>	<b>0.470</b>	<b>0.211</b>	<b>0.516</b>	<b>0.471</b>

## 4.2 Intermediate Retrieval Reward

**Motivation.** Beyond the outcome reward, which directly evaluates the correctness of the final answer after multiple search interactions, it is possible to incorporate intermediate retrieval rewards that assess the quality of the retrieved documents during each search step [22]. By assigning positive rewards to cases where relevant information is retrieved, the LLM can be encouraged to generate higher-quality queries that yield more relevant retrieval results [22]. We investigate whether introducing intermediate retrieval rewards benefits the RL training process of LLM-based search agents.

**Experimental Design.** Building upon the outcome reward from [13, 58] and the format reward introduced in Section 4.1, we incorporate a retrieval correctness component, resulting in the following final reward function  $r_\phi(x, y)$ :

$$r_\phi(x, y) = \begin{cases} 1 & \text{if } a_{\text{pred}} = a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{True}, \\ 1 - \lambda_f & \text{if } a_{\text{pred}} = a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{False}, \\ \lambda_f + \lambda_r & \text{if } a_{\text{pred}} \neq a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{True} \wedge f_{\text{ret}}(y) = \text{True}, \\ \lambda_f & \text{if } a_{\text{pred}} \neq a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{True} \wedge f_{\text{ret}}(y) = \text{False}, \\ 0 & \text{if } a_{\text{pred}} \neq a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{False}, \end{cases} \quad (4)$$

where  $f_{\text{ret}}(\cdot)$  determines whether the retrieved documents are relevant to the ground truth answer. The retrieved information can only be extracted when the rollout sequence follows the desired format, as described in Section 4.1 and the purpose of the intermediate retrieval reward is to provide a positive learning signal even when the final answer is incorrect. Thus, we introduce an additional reward term  $\lambda_r$  when  $a_{\text{pred}} \neq a_{\text{gold}} \wedge f_{\text{format}}(y) = \text{True}$ . In our experiments, we focus on short-form QA datasets, so we apply substring exact match as  $f_{\text{ret}}(\cdot)$ , following [11, 23], to evaluate whether ground truth appears in retrieved passages. Under this setting, even if LLM fails to generate correct final answer,

Table 2: Study of the intermediate retrieval reward.  $\lambda_r = 0.1$ . The best performance is set in bold.  $^\dagger$ / $^*$  represents in-domain/out-domain datasets.

Methods		General QA			Multi-Hop QA				Avg.
		NQ <sup>†</sup>	TriviaQA <sup>*</sup>	PopQA <sup>*</sup>	HotpotQA <sup>†</sup>	2wiki <sup>*</sup>	Musique <sup>*</sup>	Bamboogle <sup>*</sup>	
Qwen2.5-3B-Base									
PPO	w.o. retrieval reward	<b>0.428</b>	<b>0.607</b>	<b>0.459</b>	<b>0.371</b>	<b>0.387</b>	<b>0.150</b>	<b>0.323</b>	<b>0.389</b>
	w. retrieval reward	0.405	0.567	0.407	0.326	0.330	0.104	0.242	0.340
GRPO	w.o. retrieval reward	0.429	0.602	<b>0.435</b>	0.372	<b>0.383</b>	<b>0.148</b>	0.307	0.382
	w. retrieval reward	<b>0.434</b>	<b>0.605</b>	0.433	<b>0.379</b>	0.378	0.142	<b>0.323</b>	<b>0.385</b>
Qwen2.5-7B-Base									
PPO	w.o. retrieval reward	<b>0.488</b>	<b>0.644</b>	<b>0.469</b>	0.436	<b>0.412</b>	<b>0.187</b>	<b>0.403</b>	<b>0.434</b>
	w. retrieval reward	0.472	0.629	0.452	0.436	0.402	0.180	0.363	0.419
GRPO	w.o. retrieval reward	<b>0.458</b>	<b>0.632</b>	0.442	0.412	<b>0.404</b>	<b>0.180</b>	<b>0.411</b>	<b>0.420</b>
	w. retrieval reward	0.453	0.628	<b>0.450</b>	<b>0.416</b>	0.375	0.164	0.387	0.410

Table 3: Performance of general LLM and reasoning LLM trained with RL on search agent task. The best performance is set in bold.  $^\dagger$ / $^*$  represents in-domain/out-domain datasets.

Methods	General QA			Multi-Hop QA				Avg.
	NQ <sup>†</sup>	TriviaQA <sup>*</sup>	PopQA <sup>*</sup>	HotpotQA <sup>†</sup>	2wiki <sup>*</sup>	Musique <sup>*</sup>	Bamboogle <sup>*</sup>	
DeepSeek-R1-Distill-Qwen-7B								
PPO	0.389	0.542	0.402	0.334	0.326	0.122	0.290	0.344
GRPO	0.061	0.155	0.068	0.098	0.194	0.010	0.113	0.100
Qwen2.5-7B-Base								
PPO	<b>0.488</b>	<b>0.644</b>	<b>0.469</b>	<b>0.436</b>	<b>0.412</b>	<b>0.187</b>	0.403	<b>0.434</b>
GRPO	0.458	0.632	0.442	0.412	0.404	0.180	<b>0.411</b>	0.420

it can still receive a positive reward for issuing effective queries that retrieve relevant documents. Detailed experimental settings can be found in Appendix G.2.

**Results.** Performance comparisons with and without intermediate retrieval rewards are presented in Table 2. The effect of varying  $\lambda_r$  is illustrated in Figures 1(c) and (d). Key observations include: (1) Adding intermediate retrieval rewards does not significantly improve final performance for either PPO or GRPO. This may be attributed to the outcome reward already providing sufficient learning signal for generating effective queries, as a successful search engine call that retrieves relevant information directly contributes to producing the correct answer and receiving a positive reward. In contrast, the substring EM-based intermediate retrieval reward may overly constrains the retrieval trajectory and thus deviates the naturally learned trajectory from the outcome reward. (2) Varying  $\lambda_r$  has limited impact on learning dynamics. Increasing  $\lambda_r$  consistently leads to degraded performance, suggesting that intermediate retrieval rewards are unnecessary. The outcome reward alone is sufficient to encourage effective query formulation and downstream task success.

## 5 The Impact of Underlying Backbone LLM

In this section, we study how the choice of the LLM influences RL training for LLM-based search agents. Our investigation centers on two key characteristics of the base LLM: (1) type (*i.e.*, general-purpose vs. reasoning-optimized), and (2) scale (*i.e.*, 3B, 7B, 14B, and 32B).

### 5.1 Study of LLM types

**Motivation.** Effective training of LLM-based search agents via RL requires the LLM to possess two fundamental capabilities: *instruction following* and *reasoning*. Instruction following enables the model to learn how to properly issue search engine calls in the correct format, while reasoning equips the model to analyze retrieved information and solve complex problems. However, it remains underexplored whether general-purpose or reasoning-specialized LLMs provide a more suitable foundation for RL-based training.



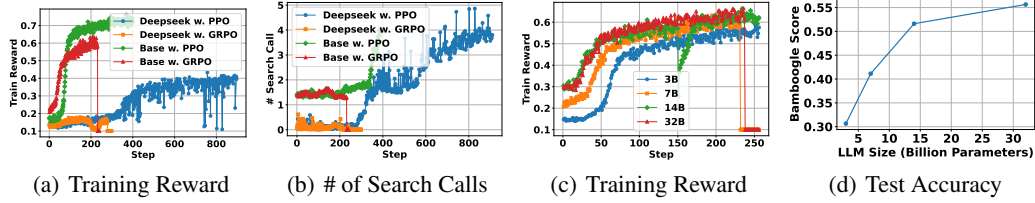


Figure 2: The study of underlying pretrained LLM for development of LLM-based search agents with RL. (a) **Training reward with different type of LLMs** - general-purpose LLM (Qwen2.5-7B-Base) and reasoning LLM (DeepSeek-R1-Distill-Qwen-7B). We observe that general-purpose LLM performs better than reasoning LLMs with both PPO and GRPO. (b) **# of Search engine calls with different type of LLMs**: General LLM learns to call the search engine faster than the reasoning LLM. This potentially stems from the fact the general LLMs are better for following instructions. (c) **Training reward with different size of LLMs**: Larger LLMs can lead to higher training reward. (d) **Test accuracy with different size of LLMs**: On the challenging Bamboogle dataset [30], the performance increases consistently as the LLM size increases.

**Experimental Design.** We follow the experimental setup in [13] and conduct RL training on two LLM variants: (1) Qwen2.5-7B-Base [50], a general-purpose 7B parameter pretrained LLM, and (2) DeepSeek-R1-Distill-Qwen-7B [9], a 7B reasoning-specialized model distilled from DeepSeek-R1. Both models are trained under identical conditions to ensure a fair comparison. Detailed experimental settings and results on 14B LLMs can be found in Appendix G.3 and F.

**Results.** The training reward and search engine call frequency curves are presented in Figures 2(a) and 2(b), respectively. Final performance results are summarized in Table 3. We observe the following key findings: (1) The RL training process is more stable and effective when initialized with the general-purpose LLMs compared to the reasoning-specialized ones. This suggests that the general-purpose ones already possess sufficient basic reasoning capabilities to support the search agent task without requiring specialized pretraining. (2) The reasoning LLM struggles to initiate search engine calls during the early stages of training, leading to insufficient exploration. In the absence of positive reward signals from successful rollouts involving search, the model fails to consistently learn to engage with the search engine. This behavior primarily stems from the reasoning LLMs’ limited instruction-following capabilities, which hinder their ability to learn the correct format for invoking the search API. (3) While the reasoning LLMs eventually learn to perform interleaved reasoning and retrieval when trained with PPO, this progress is slow and gradual. In contrast, training with GRPO leads to training collapse. We attribute this to PPO’s lower variance and more stable policy updates, which better support the complex exploration required for search-augmented reasoning tasks.

## 5.2 The Scale Up of LLM-based Search Agent

**Motivation.** Prior work has demonstrated that LLM capabilities improve predictably with increased model size, as described by scaling laws [10, 15]. However, it remains unclear whether similar scaling behavior holds when LLMs are further RL-tuned as search agents. Specifically, does increasing model size consistently improve the agent’s ability to reason and interact with search engines?

**Experimental Design.** We evaluate scaling laws using RL with both outcome-based rewards and additional format rewards, as introduced in Section 4.1. Following the experimental setup in [13], we train Qwen2.5 models of varying sizes (3B, 7B, 14B, 32B) on the NQ and HotpotQA training datasets using the GRPO algorithm. We use a fixed learning rate of  $5 \times 10^{-7}$ , and evaluate on the out-of-distribution Bamboogle dataset. Detailed experimental settings can be found in Appendix G.4.

**Results.** As shown in Figure 2(c), the training reward consistently improves with increasing LLM size, indicating that larger models are better able to learn effective reasoning and search engine usage. The corresponding inference performance is presented in Figure 2(d). While test performance also improves with model size, the rate of improvement diminishes. This suggests that the search agent task, unlike pure language modeling, relies less on parametric knowledge stored in large LLMs, and more on effective external information acquisition through retrieval.

Table 4: Final performance with different search engine for both training and inference. The best performance is set in bold. (LLM: Qwen2.5-7B-Base; RL: PPO)

Engine	NQ		TriviaQA		PopQA		HotpotQA		2wiki		Musique		Bamboogle		Avg.	
	Recall	EM	Recall	EM	Recall	EM	Recall	EM	Recall	EM	Recall	EM	Recall	EM	Recall	EM
Random	0.000	0.237	0.000	0.494	0.000	0.177	0.000	0.217	0.000	0.269	0.000	0.058	0.000	0.234	0.000	0.241
BM25	0.216	0.341	0.445	0.607	0.255	0.322	0.273	0.404	<b>0.216</b>	0.370	0.076	0.137	0.061	0.280	0.176	0.352
E5 (HNSW)	0.436	0.468	0.509	0.621	0.304	0.366	0.237	0.372	0.146	0.287	0.092	0.137	0.104	0.400	0.261	0.379
E5 (Exact)	<b>0.462</b>	<b>0.481</b>	<b>0.561</b>	<b>0.638</b>	<b>0.423</b>	<b>0.457</b>	<b>0.276</b>	<b>0.433</b>	0.198	<b>0.382</b>	<b>0.098</b>	<b>0.196</b>	<b>0.107</b>	<b>0.424</b>	<b>0.304</b>	<b>0.430</b>

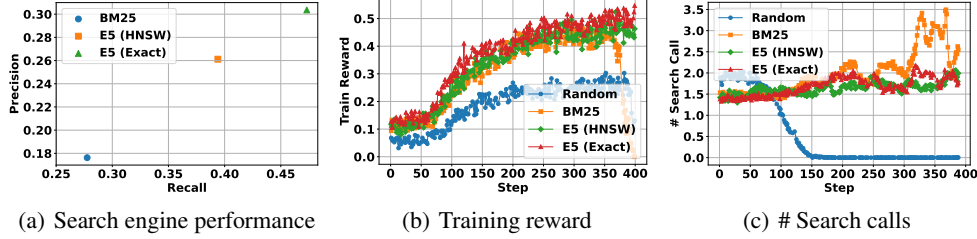


Figure 3: Effect of Search Engine Choice on RL Training Dynamics. (a) **Retrieval Quality Ranking:** E5 (Exact) > E5 (HNSW) > BM25 > Random. (b) **Training Stability and Performance:** Stronger search engines (e.g., E5 Exact, E5 + HNSW) lead to more stable training and higher final performance, while weaker engines (e.g., Random, BM25) achieve suboptimal outcomes. (c) **Search Engine Usage Behavior:** With Random Noise, the agent quickly learns to avoid using the search engine. With BM25, the agent gradually increases search calls to compensate for limited retrieval quality. With E5, the agent issues search calls more strategically, reflecting more efficient search behavior.

## 6 Improved LLM-based Search Agents with Stronger Search Engines

The choice of search engine plays a critical role in determining retrieval quality, which in turn influences both the RL training dynamics and the inference-time performance of the LLM-based search agent. During training, higher-quality search engines that provide more relevant information can encourage the agent to achieve its objectives with fewer search calls, as the retrieved content more effectively supports reasoning and decision-making. In contrast, lower-quality search engines that return less relevant information may lead the agent to either over-rely on its internal knowledge or issue multiple search queries to compensate for inadequate results. At inference time, the quality of the retrieved information directly impacts the agent’s ability to generate accurate and useful responses. In the following sections, we systematically investigate the effects of search engine choice on both the training and inference stages of search-augmented LLMs.

### 6.1 Training with Different Search Engines.

**Motivation.** During RL training, the LLM-based agent learns to interact with the search engine and receive positive reward feedback while it solves problems using the retrieved relevant information. A *strong* search engine provides more relevant results, leading to consistent positive outcome rewards. Consequently, the LLM learns to solve problems with fewer search calls. In contrast, a *weak* search engine discourages reliance on retrieval or forces the agent to issue multiple search queries to compensate for low-quality results. We empirically investigate how different search engines influence the RL training dynamics of an LLM-based search agent.

**Experimental Design.** We conduct experiments using the Qwen2.5-7B-Base model as the LLM and Proximal Policy Optimization (PPO) as the RL algorithm. Four search engine configurations are explored with the Wikipedia-18 corpus [16]: (1) **Random Noise:** Returns randomly selected passages for a given query. (2) **BM25** [33]: A sparse retrieval method based on exact token matching and term frequency. (3) **E5 (HNSW)** [27, 42]: A dense retrieval method that encodes queries and passages into semantic embeddings, using dot product similarity for matching. HNSW provides efficient approximate nearest neighbor (ANN) search at the cost of some accuracy. (4) **E5 (Exact Match)**



Table 5: Retriever generalization results across datasets and test retrievers. (Qwen2.5-7B-Base, PPO)

Train / Test Search Engine	BM25			E5 (HNSW)			E5 (Exact)			Google Search		
	Bamgb	GPQA	SimpleQA	Bamgb	GPQA	SimpleQA	Bamgb	GPQA	SimpleQA	Bamgb	GPQA	SimpleQA
BM25	0.280	0.273	0.243	0.432	0.293	0.159	0.424	0.323	0.259	0.496	0.313	0.540
E5 (HNSW)	0.240	0.298	0.270	0.400	0.288	0.169	0.440	0.273	0.254	0.528	0.333	0.603
E5 (Exact)	0.312	0.313	0.249	0.400	0.298	0.196	0.424	0.288	0.265	0.560	0.293	0.603
Average	0.277	0.295	0.254	0.411	0.293	0.175	0.429	0.295	0.259	0.528	0.313	0.582

[42]: A dense retrieval method using exact embedding matching without approximation, ensuring the highest retrieval accuracy. The retrieval performance of these methods follows the ranking: E5 (Exact) > E5 (HNSW) > BM25 > Random, as shown in Figure 3(a). Detailed experimental settings and case studies can be found in Appendix G.5 and H.

**Results.** The training reward curves and final test performance under different search engine settings are presented in Figure 3(c) and Table 4. We observe the following trends: (1) Training with stronger search engines (*e.g.*, **E5 (Exact)** and **E5 (HNSW)**) results in more stable RL training and better final performance. (2) Training with weaker search engines (*e.g.*, **Random** and **BM25**) leads to suboptimal final performance. The search engine call frequency during training is illustrated in Figure 3(d), revealing: (1) With **Random Noise**, the agent quickly learns to avoid using the search engine, as the retrieved information does not contribute to problem-solving. (2) With **BM25**, the agent gradually increases the number of search engine calls. Since BM25 offers limited retrieval quality, the agent needs to issue multiple, refined queries to obtain relevant information. (3) With **E5** (a stronger retriever), the agent learns to utilize the search engine judiciously, making a reasonable number of calls to acquire the necessary information efficiently.

## 6.2 Inference with Different Search Engines.

**Motivation.** In practical scenarios, certain search engines may be unsuitable for integration during RL training due to empirical constraints such as accessibility, cost, or API limitations. In such cases, it becomes necessary to train the agent using one search engine while employing a different one during inference. This raises important research questions: (1) To what extent does the difference in search engines between training and inference affect model performance? (2) Does utilizing a stronger search engine at inference lead to improved downstream performance?

**Experimental Design.** Following the training setup in [13], we investigate these questions by training the LLM-based search agent using three different search engines: (1) BM25 [33], (2) E5 [42] with approximate nearest neighbor (ANN) search implemented via HNSW [27], and (3) E5 with exact search. During inference, we additionally include the online Google Search API<sup>2</sup> as a stronger retrieval system. To study the importance of the search engine in challenging scenarios, we construct a combined benchmark consisting of 512 samples. This benchmark includes the full Bamboogle test set [30] (Bamgb), the complete GPQA-diamond dataset [32] (GPQA), and a randomly sampled subset from SimpleQA [46]. Detailed experimental settings can be found in Appendix G.6.

**Results.** As shown in Table 5, we observe the following: (1) LLM search agents trained with a specific search engine demonstrate strong generalization capabilities when evaluated with different search engines during inference. (2) Leveraging a more powerful search engine at inference time (*e.g.*, Google Search) consistently and significantly leads to improved performance, highlighting the importance of high-quality retrieval in downstream applications.

More studies on long-form generation tasks with outcome drive RL and data scaling study can be found in Appendix D and E, respectively.

<sup>2</sup><https://developers.google.com/custom-search/v1/overview>

## 7 Conclusion

In this work, we conduct comprehensive empirical studies on key design factors in training LLM-based search agents using reinforcement learning. Our investigation reveals that format rewards play an important role in certain scenarios, while intermediate retrieval rewards provide limited benefit and may not consistently improve the learning process. We demonstrate that the choice of the underlying LLM (whether a general-purpose model or one specialized for reasoning) and its scale significantly affect the final agent’s performance. Additionally, the selection of the search engine plays a non-trivial role in shaping both the RL training dynamics and the robustness of the agent during inference. These insights offer practical guidance for developing more capable and reliable LLM-based search agents, paving the way for their deployment in real-world applications. Interesting future directions include exploring more advanced reward modeling techniques, such as learned reward functions and preference-based feedback, as well as studying the agentic behaviors acquired through RL in broader scenarios, including tool use and software engineering.

## Acknowledgments and Disclosure of Funding

This research was supported in part by Apple PhD Fellowship, in part by US DARPA INCAS Program No. HR0011-21-C0165 and BRIES Program No. HR0011-24-3-0325, in part by the Office of Naval Research contract number N000142412612, in part by NSF grant numbers IIS-19-56151 and 2402873, in part by the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897 and the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) by NSF under Award No. 2118329, in part by Cisco, and in part by the Center for Intelligent Information Retrieval. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of the sponsors or the U.S. Government.

## References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3
- [2] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024. 2
- [3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023. 1, 3
- [4] Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Research: Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025. 2, 3
- [5] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*, 2019. 16
- [6] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025. 2
- [7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023. 16
- [8] Carlos Gómez-Rodríguez and Paul Williams. A confederacy of models: A comprehensive evaluation of llms on creative writing. *arXiv preprint arXiv:2310.08433*, 2023. 1

- [9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1, 2, 3, 7, 16
- [10] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. 7
- [11] Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan O Arik. Long-context llms meet rag: Overcoming challenges for long inputs in rag. In *The Thirteenth International Conference on Learning Representations*, 2024. 1, 5
- [12] Bowen Jin, Jinsung Yoon, Zhen Qin, Ziqi Wang, Wei Xiong, Yu Meng, Jiawei Han, and Sercan O Arik. Llm alignment as retriever optimization: An information retrieval perspective. *arXiv preprint arXiv:2502.03699*, 2025. 2
- [13] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025. 2, 3, 4, 5, 7, 9, 16, 18, 19
- [14] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. 1, 2
- [15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 7
- [16] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020. 8
- [17] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*, 10, 2023. 2
- [18] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. 16
- [19] Yuxiang Lai, Jike Zhong, Ming Li, Shitian Zhao, and Xiaofeng Yang. Med-r1: Reinforcement learning for generalizable medical reasoning in vision-language models. *arXiv preprint arXiv:2503.13939*, 2025. 1
- [20] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*, 2024. 2
- [21] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382, 2023. 3
- [22] Jiacheng Lin, Tian Wang, and Kun Qian. Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning. *arXiv preprint arXiv:2503.24289*, 2025. 2, 5
- [23] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. Ra-dit: Retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2023. 5
- [24] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025. 1

- [25] Zhaowei Liu, Xin Guo, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Zixuan Wang, Jiajie Xu, Weige Cai, Ziwei Yang, Xueqian Zhao, et al. Fin-r1: A large language model for financial reasoning through reinforcement learning. *arXiv preprint arXiv:2503.16252*, 2025. **1**
- [26] Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa: Surpassing gpt-4 on conversational qa and rag. *Advances in Neural Information Processing Systems*, 37:15416–15459, 2024. **1**
- [27] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018. **8, 9**
- [28] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. **2, 3**
- [29] Cheng Peng, Xi Yang, Aokun Chen, Kaleb E Smith, Nima PourNejatian, Anthony B Costa, Cheryl Martin, Mona G Flores, Ying Zhang, Tanja Magoc, et al. A study of generative large language model for medical research and healthcare. *NPJ digital medicine*, 6(1):210, 2023. **3**
- [30] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022. **7, 9, 20**
- [31] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023. **3**
- [32] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. **9, 20**
- [33] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009. **2, 8, 9**
- [34] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36: 68539–68551, 2023. **1, 3**
- [35] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. **2**
- [36] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025. **3**
- [37] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025. **2, 3**
- [38] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. Asqa: Factoid questions meet long-form answers. *arXiv preprint arXiv:2204.06092*, 2022. **16**
- [39] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999. **1, 2**
- [40] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. **3**
- [41] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022. **1, 3**

- [42] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022. 8, 9
- [43] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Lucas Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025. 16
- [44] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. 1
- [45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 1
- [46] Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024. 9, 20
- [47] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992. 2
- [48] Xiaobo Xia and Run Luo. Gui-rl: A generalist rl-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025. 1
- [49] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025. 1
- [50] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. 7
- [51] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018. 16
- [52] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. 3, 4
- [53] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. 2
- [54] Yufeng Yuan, Qiyang Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, Tiantian Fan, Zhengyin Du, Xiangpeng Wei, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025. 2
- [55] Yufei Zhan, Yousong Zhu, Shurong Zheng, Hongyin Zhao, Fan Yang, Ming Tang, and Jinqiao Wang. Vision-rl: Evolving human-free alignment in large vision-language models via vision-guided reinforcement learning. *arXiv preprint arXiv:2503.18013*, 2025. 1
- [56] Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57, 2024. 1
- [57] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023. 1, 3
- [58] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025. 2, 3, 4, 5, 16



## A Limitations

In this work, we conduct an empirical investigation into the use of reinforcement learning (RL) for training LLM-based search agents. Our study focuses on three key factors that influence the effectiveness of RL in this context: (1) reward formulation, (2) the choice and characteristics of the underlying language model, and (3) the role and quality of the search engine. However, our analysis is primarily confined to search-based agent scenarios. While this provides valuable insights into the challenges and design choices for RL in retrieval-augmented reasoning, it does not fully generalize to other classes of LLM-based agents. In particular, RL for more complex and open-ended agent behaviors—such as those exhibited by data science assistants, software engineering agents, or multi-tool task planners—remains underexplored. These domains may require more sophisticated reward structures, long-horizon credit assignment, and multi-step decision-making policies that go beyond the scope of search-oriented tasks. We leave a more comprehensive study of RL in broader agentic settings as an important future work direction.

## B Positive and Negative Societal Impacts

Our work aims to improve the efficiency and reliability of large language model (LLM)-based agents by leveraging reinforcement learning to optimize their interaction with external tools such as search engines. On the positive side, this line of research can enhance the capabilities of LLMs in high-stakes domains like scientific research, education, and healthcare, where accurate information retrieval and reasoning are critical. By making LLM-based agents more effective at querying and utilizing external knowledge sources, our approach has the potential to reduce hallucinations, improve transparency, and increase user trust in AI systems. However, as with all advances in powerful LLM-based agents, there are potential negative impacts. Improved autonomy in tool usage may lead to unintended misuse, such as generating convincing but misleading information or automating complex tasks without sufficient human oversight. Moreover, the deployment of search-augmented agents could exacerbate access disparities if such technologies are restricted to proprietary systems. Careful consideration of ethical deployment, transparency in agent decision-making, and equitable access to advanced AI capabilities is essential to mitigate these risks.

## C Format Reward Code

In this section, we provide the code to judge whether the rollout sequence is in a desired format for a search agent.

Listing 1: Format Reward Code.

```
def is_valid_sequence(text):
    # Find the position of "<|im_start|>assistant" with potential
    # whitespace
    assistant_pattern = r"<\\im_start\\|>assistant\\s*"
    assistant_match = re.search(assistant_pattern, text)

    if not assistant_match:
        return False, "Missing assistant marker"

    # Extract the content after the assistant marker
    start_pos = assistant_match.end()
    content = text[start_pos:]

    # Check for balanced tags
    tags_to_check = ["think", "search", "information", "answer"]
    for tag in tags_to_check:
        opening_count = len(re.findall(f"<{tag}>", content))
        closing_count = len(re.findall(f"</{tag}>", content))
        if opening_count != closing_count:
            return False, f"Mismatch in {tag} tags: {opening_count}
                           opening vs {closing_count} closing tags"
```

```

# Now check for proper sequence pattern and no extraneous content

# 1. First split the content by any tags we recognize
split_pattern = r"(</?(?:think|search|information|answer)>)"
parts = re.split(split_pattern, content)

# 2. Keep track of the current position in the expected sequence
state = "start" # start -> think -> search -> information ->
               think -> ... -> answer -> end

# 3. Check each part
for i, part in enumerate(parts):
    # Skip empty parts
    if not part.strip():
        continue

    # Check if this is a tag
    if re.match(r"</?(?:think|search|information|answer)>", part):
        # This is a tag, check if it's valid in the current state
        if part == "<think>" and state in ["start", "information"]]:
            state = "in_think"
        elif part == "</think>" and state == "in_think":
            state = "after_think"
        elif part == "<search>" and state == "after_think":
            state = "in_search"
        elif part == "</search>" and state == "in_search":
            state = "after_search"
        elif part == "<information>" and state == "after_search":
            state = "in_information"
        elif part == "</information>" and state == "in_information":
            state = "information"
        elif part == "<answer>" and state == "after_think":
            state = "in_answer"
        elif part == "</answer>" and state == "in_answer":
            state = "end"
        else:
            return False, f"Unexpected tag {part} in state {state}"

    else:
        # This is content, check if it's valid in the current state
        if state in ["in_think", "in_search", "in_information", "in_answer"]]:
            # Content is allowed inside tags
            pass
        elif state in ["start", "after_think", "after_search", "information"]]:
            # Only whitespace is allowed between tags
            if part.strip():
                return False, f"Unexpected content '{part.strip()}' between tags (state: {state})"
            else:
                return False, f"Unexpected content in state {state}"

# Check final state
if state != "end":
    return False, f"Incomplete sequence, ended in state {state}"

return True, "Valid sequence format"

```

## D Reward for Long-form Answers

**Motivation.** It is demonstrated that rule-based outcome rewards are effective for training LLM-based search agents [13, 58]. However, their evaluation primarily focuses on short-form QA tasks, where answer correctness can be reliably measured using exact string matching. In real-world applications, many queries require long-form, open-ended answers, where evaluating correctness is inherently more subjective. We investigate whether rule-based outcome rewards remain effective in training LLM search agents for long-form QA tasks.

**Experimental Design.** We conduct experiments on two long-form QA datasets: ASQA [38] and ELI5 [5]. Models are trained on the ASQA training set and evaluated on its development set for in-distribution performance. Out-of-distribution performance is evaluated on the ELI5 dataset. Following common practice, we use the F1 score as the rule-based evaluation metric for both training and evaluation. Experiments are conducted on both Qwen2.5-3B-Base and Qwen2.5-7B-Base models. We compare against several baselines, including Direct Inference (with instruct LLMs), RAG [7] (with instruct LLMs), and R1 [9]. Notably, R1 represents an LLM, trained using the RL approach from DeepSeek-R1 with the same training data (ASQA training set). We also explore a variant of [13, 58] that incorporates the format reward, denoted as “SEARCH-R1 w. Outcome + Format reward”.

Table 6: Study on long-form question answering tasks.

Methods	Qwen2.5-3b			Qwen2.5-7b			Qwen2.5-14b		
	ASQA	ELI5	Avg.	ASQA	ELI5	Avg.	ASQA	ELI5	Avg.
Direct	0.251	0.199	0.225	0.303	0.201	0.252	0.289	0.199	0.244
RAG	0.301	0.193	0.247	0.317	0.202	0.259	0.285	0.193	0.239
R1	0.424	<b>0.275</b>	0.350	0.437	<b>0.280</b>	0.358	0.444	<b>0.278</b>	0.361
SEARCH-R1 w. Outcome + Format reward (PPO)	0.480	0.261	0.370	0.471	0.256	0.363	0.442	0.260	0.351
SEARCH-R1 w. Outcome + Format reward (GRPO)	<b>0.492</b>	0.272	<b>0.382</b>	<b>0.504</b>	0.275	<b>0.390</b>	<b>0.501</b>	0.273	<b>0.387</b>

**Results.** Performance comparisons are shown in Table 6. We observe that “SEARCH-R1 w. Outcome + Format reward” achieves competitive results on long-form QA tasks, demonstrating the effectiveness of rule-based outcome rewards with format rewards even in complex, open-ended scenarios. This suggests that RL guided by the proposed rewards remains a viable strategy for training search-augmented LLMs, even when the evaluation objective shifts from short-form to long-form answer generation.

## E Study of Data Scaling

**Motivation.** Although RL has shown strong potential in training LLM-based search agents [13], the impact of training data size on the RL process remains underexplored. While recent work has shown that extremely small datasets can be sufficient for reasoning-oriented RL with LLMs [43], it is unclear whether similar data efficiency holds in *agentic RL* settings, where models must learn to reason and interact with external tools in an interleaved manner.

**Experimental Design.** Following [13], we use the training sets from NQ [18] and HotpotQA [51] as our full training set. To investigate the effect of training data size, we construct subsets of varying sizes by randomly sampling  $k$  examples from the full dataset, where  $k \in 1, 10, 100, 1000, 10000$ . We conduct experiments using the Qwen2.5-3B-Base model and evaluate both PPO and GRPO as the underlying reinforcement learning algorithms.

**Results.** We present the training reward dynamics and the number of search engine calls across varying training data sizes in Figure 4. The final performance of LLM-based search agents trained with different dataset sizes is reported in Table 7. The results reveal several key observations: (1) Increasing the size of the training dataset generally leads to improved performance, particularly on more complex multi-hop question answering tasks such as HotpotQA and 2Wiki. (2) Smaller datasets lead to faster convergence and higher training rewards, which is likely attributable to overfitting. (3) For PPO, training with a single example fails to induce meaningful search behavior, while using 10 examples results in unstable training. In contrast, training with 100 or 1,000 examples enables the

Table 7: Final performance with different size of training data.  $\mathcal{D}$  is the training data. The best performance is set in bold.  $^\dagger$ / $^*$  represents in-domain/out-domain datasets. (LLM: Qwen2.5-3B-Base)

Methods	General QA			Multi-Hop QA				Avg.
	NQ <sup>†</sup>	TriviaQA <sup>*</sup>	PopQA <sup>*</sup>	HotpotQA <sup>†</sup>	2wiki <sup>*</sup>	Musique <sup>*</sup>	Bamboogle <sup>*</sup>	
PPO								
$ \mathcal{D}  = 1$	0.121	0.339	0.119	0.141	0.211	0.017	0.056	0.143
$ \mathcal{D}  = 10$	0.339	0.509	0.379	0.236	0.237	0.048	0.081	0.261
$ \mathcal{D}  = 100$	0.372	0.549	0.382	0.262	0.279	0.063	0.161	0.295
$ \mathcal{D}  = 1000$	<b>0.431</b>	<b>0.599</b>	<b>0.446</b>	0.348	0.355	0.136	0.298	0.373
$ \mathcal{D}  = 10000$	0.430	0.594	0.445	<b>0.369</b>	<b>0.383</b>	<b>0.155</b>	<b>0.315</b>	<b>0.384</b>
GRPO								
$ \mathcal{D}  = 1$	0.107	0.287	0.115	0.124	0.202	0.019	0.089	0.134
$ \mathcal{D}  = 10$	0.338	0.515	0.361	0.236	0.223	0.043	0.089	0.258
$ \mathcal{D}  = 100$	0.367	0.529	0.414	0.265	0.302	0.079	0.194	0.307
$ \mathcal{D}  = 1000$	0.421	0.594	<b>0.437</b>	0.363	0.364	<b>0.149</b>	<b>0.315</b>	0.377
$ \mathcal{D}  = 10000$	<b>0.435</b>	<b>0.599</b>	0.435	<b>0.365</b>	<b>0.379</b>	0.137	0.306	<b>0.379</b>

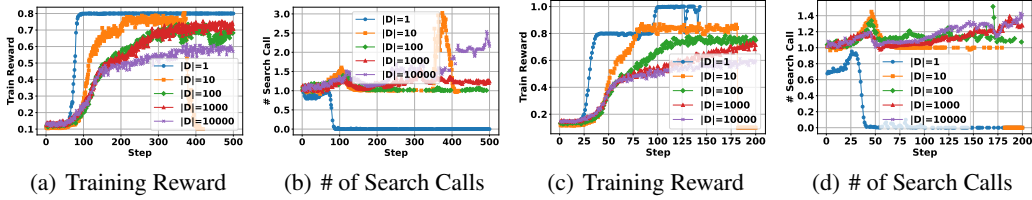


Figure 4: Data scaling effects in RL training for search agents. (a) **Training reward under PPO with varying dataset sizes**: Smaller training sets result in faster convergence and higher training rewards, likely due to overfitting. (b) **Number of search engine calls under PPO**: Training with a single example fails to induce search behavior, while 10 samples lead to unstable learning. In contrast, using 100 or 1,000 samples enables the model to learn stable search behavior, and training with 10,000 samples further improves performance. (c) **Training reward under GRPO with varying dataset sizes**: Similar to PPO, smaller datasets yield faster convergence and higher rewards, again suggesting potential overfitting. (d) **Number of search engine calls under GRPO**: A single training sample is insufficient for search behavior to emerge, whereas larger datasets facilitate stable learning of search interactions.

model to learn stable search behavior, and performance continues to improve with 10,000 examples. Similar trends are observed under GRPO training.

## F More Studies on LLM Types

In addition to the 7B model analysis in Section 5.1, we further investigate the impact of LLM initialization on RL performance using 14B-scale models. Specifically, we compare Qwen2.5-14B-Base as a general-purpose LLM and DeepSeek-R1-Distill-Qwen-14B as a reasoning-specialized LLM, evaluating both under PPO and GRPO training. The results, summarized in Table 8 and Figure 5, yield the following observations: (1) RL training is more stable and effective when initialized from the general-purpose LLM, suggesting that such models possess sufficient general reasoning capabilities to support the search agent task, even without reasoning-specific pretraining. (2) Although the reasoning-specialized LLM eventually learns to perform interleaved reasoning and retrieval with PPO and GRPO, the general-purpose LLM consistently achieves higher final performance—likely due to its stronger ability to generate effective search queries.

Table 8: LLM type study with 14B LLMs. The best performance is set in bold. <sup>†</sup>/<sup>\*</sup> represents in-domain/out-domain datasets.

Methods	General QA			Multi-Hop QA				Avg.
	NQ <sup>†</sup>	TriviaQA <sup>*</sup>	PopQA <sup>*</sup>	HotpotQA <sup>†</sup>	2wiki <sup>*</sup>	Musique <sup>*</sup>	Bamboogle <sup>*</sup>	
DeepSeek-R1-Distill-Qwen-14b								
PPO	0.475	0.634	0.465	0.401	0.363	0.211	0.476	0.432
GRPO	0.305	0.613	0.332	0.285	0.276	0.092	0.347	0.321
Qwen2.5-14b-Base								
PPO	0.499	0.680	0.472	0.452	0.431	0.215	0.468	0.459
GRPO	<b>0.500</b>	<b>0.693</b>	<b>0.500</b>	<b>0.481</b>	<b>0.488</b>	<b>0.261</b>	<b>0.516</b>	<b>0.491</b>

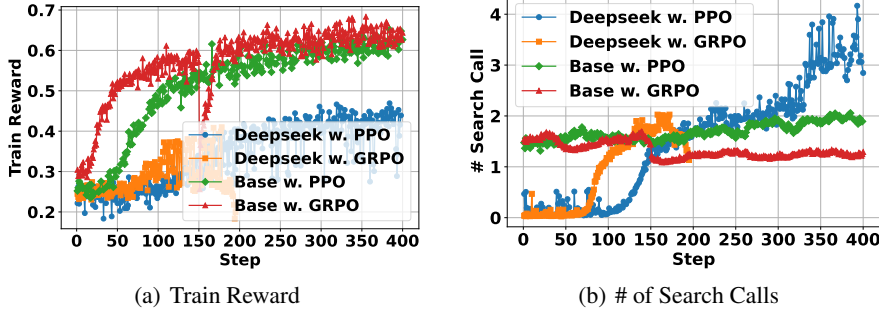


Figure 5: The study of underlying pretrained LLM for development of search agents with RL. (a) **Training reward with different type of LLMs** - general-purpose LLM (Qwen2.5-14B-Base) and reasoning LLM (DeepSeek-R1-Distill-Qwen-14B). We observe that general-purpose LLM performs better than reasoning LLMs with both PPO and GRPO. (b) **# of Search engine calls with different type of LLMs**: Both the general-purpose LLM and the reasoning-specialized LLM demonstrate the ability to learn when to call the search engine. However, the general-purpose LLM achieves better final performance, potentially due to its superior capability in formulating effective search queries.

## G Experimental Settings

In this section, we detail the experimental settings used in the studies presented in Sections 4, 5, 6.

### G.1 Section 4.1

We adopt the same training dataset as used in [13], consisting of the Natural Questions (NQ) and HotpotQA training sets. For PPO training, the policy LLM learning rates are set to  $1 \times 10^{-6}$  for Qwen2.5-3B and Qwen2.5-7B, and  $5 \times 10^{-7}$  for Qwen2.5-14B. The critic LLM learning rate is fixed at  $1 \times 10^{-5}$  across all model sizes. For GRPO training, we use a policy LLM learning rate of  $5 \times 10^{-7}$  for all models. The RL training batch size is set to 512, and the rollout temperature is fixed at 1. For GRPO, we set the group size to 5. Each model is trained for up to 600 steps, with early stopping triggered if training collapse is observed based on the reward curve. For the results reported in Table 1, we use either the final checkpoint at step 600 or the last checkpoint prior to collapse. All training jobs are conducted on a node equipped with 8 NVIDIA H100 GPUs. We conduct a grid search over  $\lambda_f \in 0.2, 0.4, 0.6, 0.8$  and select the best-performing value for each model: 0.2 for 3B, 0.4 for 7B, and 0.2 for 14B. We adopt E5 (exact) as the retriever and return the top-3 passages (each contains about 200 tokens).

### G.2 Section 4.2

We adopt the same training dataset as used in [13], consisting of the Natural Questions (NQ) and HotpotQA training sets. For PPO training, the policy LLM learning rates are set to  $1 \times 10^{-6}$  for both Qwen2.5-3B and Qwen2.5-7B. The critic LLM learning rate is fixed at  $1 \times 10^{-5}$  across all model sizes. For GRPO training, we use a policy LLM learning rate of  $5 \times 10^{-7}$  for both models. The RL



training batch size is set to 512, and the rollout temperature is fixed at 1. For GRPO, we set the group size to 5. Each model is trained for up to 600 steps, with early stopping triggered if training collapse is observed based on the reward curve. For the results reported in Table 1, we use either the final checkpoint at step 600 or the last checkpoint prior to collapse. All training jobs are conducted on a node equipped with 8 NVIDIA H100 GPUs. Based on the findings in Section 4.1, we fix  $\lambda_f$  to 0.2 for the 3B model and 0.4 for the 7B model. We then perform a grid search over  $\lambda_r \in 0.1, 0.3, 0.5$  and select the best-performing value for each model, which is 0.1 for both 3B and 7B. We adopt E5 (exact) as the retriever and return the top-3 passages (each contains about 200 tokens).

### G.3 Section 5.1

We adopt the same training dataset as used in [13], consisting of the Natural Questions (NQ) and HotpotQA training sets. For PPO training, the policy LLM learning rates are set to  $1 \times 10^{-6}$  for both Qwen2.5-7B and DeepSeek-R1-Distill-Qwen-7B. The critic LLM learning rate is fixed at  $1 \times 10^{-5}$  across all model sizes. For GRPO training, we use a policy LLM learning rate of  $5 \times 10^{-7}$  for both models. The RL training batch size is set to 512, and the rollout temperature is fixed at 1. For GRPO, we set the group size to 5. Each model is trained for up to 600 steps, with early stopping triggered if training collapse is observed based on the reward curve. For the results reported in Table 1, we use either the final checkpoint at step 600 or the last checkpoint prior to collapse. All training jobs are conducted on a node equipped with 8 NVIDIA H100 GPUs. We set  $\lambda_f$  as 0.2 and  $\lambda_r$  as 0 for all the experiment. We adopt E5 (exact) as the retriever and return the top-3 passages (each contains about 200 tokens).

### G.4 Section 5.2

We adopt the same training dataset as used in [13], consisting of the Natural Questions (NQ) and HotpotQA training sets. For GRPO training, we use a policy LLM learning rate of  $5 \times 10^{-7}$  for all models. The RL training batch size is set to 512, and the rollout temperature is fixed at 1. For GRPO, we set the group size to 5. Each model is trained for up to 600 steps, with early stopping triggered if training collapse is observed based on the reward curve. For the results reported in Table 1, we use either the final checkpoint at step 600 or the last checkpoint prior to collapse. All training jobs are conducted on a node equipped with 8 NVIDIA H100 GPUs. We set  $\lambda_f$  as 0.2 for all the experiment. We adopt E5 (exact) as the retriever and return the top-3 passages (each contains about 200 tokens).

### G.5 Section 6.1

We adopt the same training dataset as used in [13], consisting of the Natural Questions (NQ) and HotpotQA training sets. We adopt Qwen2.5-7B-Base as the LLM backbone and PPO as the RL method. For PPO training, the policy LLM learning rates are set to  $1 \times 10^{-6}$  and the critic LLM learning rate is fixed at  $1 \times 10^{-5}$ . Each model is trained for up to 600 steps, with early stopping triggered if training collapse is observed based on the reward curve. For the results reported in Table 1, we use either the final checkpoint at step 600 or the last checkpoint prior to collapse. All training jobs are conducted on a node equipped with 8 NVIDIA H100 GPUs. For BM25, we adopt the Pyserini implementation<sup>3</sup> and for E5, we adopt Faiss<sup>4</sup>. We adopt “HNSW64” for the ANN setting. We return the top-3 passages (each contains about 200 tokens) from each search engine.

### G.6 Section 6.2

We adopt the same training dataset as used in [13], consisting of the Natural Questions (NQ) and HotpotQA training sets. We adopt Qwen2.5-7B-Base as the LLM backbone and PPO as the RL method. For PPO training, the policy LLM learning rates are set to  $1 \times 10^{-6}$  and the critic LLM learning rate is fixed at  $1 \times 10^{-5}$ . Each model is trained for up to 600 steps, with early stopping triggered if training collapse is observed based on the reward curve. For the results reported in Table 1, we use either the final checkpoint at step 600 or the last checkpoint prior to collapse. All training jobs are conducted on a node equipped with 8 NVIDIA H100 GPUs. For BM25, we adopt the Pyserini implementation and for E5, we adopt Faiss. We adopt “HNSW64” for the ANN setting.

<sup>3</sup><https://github.com/castorini/pyserini>

<sup>4</sup><https://github.com/facebookresearch/faiss>

Given that Google Search API is expensive, we construct a testing set with 512 samples, including full test set from Bamboogle [30] (125 samples), GPQA-Diamond [32] (198 samples) and randomly selected 189 samples from SimpleQA [46]. We return the top-3 passages (each contains about 200 tokens) from each search engine.

## H Case Studies on LLMs Trained with Different Search Engines

Table 9: Case study A of LLM-based search agent trained with different search engines (*i.e.*, random noise, BM25, and E5).

<p><b>Question:</b> Which Hells Angel member stabbed and beat an attendant of the 1969 counterculture-era rock concert in the United States?</p>
<p><b>Ground Truth:</b> Alan Passaro</p>
<p><b>w. random noise:</b></p> <p>&lt;think&gt; To answer this question, we need to recall the incident involving the 1969 counterculture-era rock concert known as Woodstock. The incident in question is the killing of festival security guard Thomas Benson. The Hells Angel member responsible for the stabbing and subsequent death of Benson is CompareTag Stevenson. &lt;/think&gt;</p> <p>&lt;answer&gt; Manson &lt;/answer&gt;</p>
<p><b>w. BM25:</b></p> <p>&lt;think&gt; To answer this question, I'll need to identify the Hells Angel member who was involved in the stabbing and beating of an attendant at a 1969 counterculture-era rock concert in the United States. I'll use a search engine to gather information about this event and the responsible Hells Angel member. &lt;/think&gt;</p> <p>&lt;search&gt; Hells Angel member at 1969 counterculture-era rock concert in the United States &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "Altamont Free Concert") and for the negative portrayal of the Angels in the ""Gimme Shelter"" film. The conspirators reportedly used a boat to approach a residence where Jagger was staying on Long Island, New York, the plot failing when the boat was nearly sunk by a storm. Jagger's spokesperson has refused to comment on the matter ... &lt;/information&gt;</p> <p>&lt;think&gt; From the search results, I've identified that the 1969 counterculture-era rock concert in question is the Altamont Free Concert. Now, I need to find out which Hells Angel member was involved in stabbing and beating an attendant at this concert. &lt;/think&gt;</p> <p>&lt;search&gt; Hells Angel member who stabbed and beat an attendant at Altamont Free Concert &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "The Rolling Stones") and former members of the band. Two species, Petroperla mickjaggeri and Lapisperla keithrichardsi, were placed within a new family Petroperlidae. The new family was named in honour of the Rolling Stones, derived from the Greek ""petra"" that stands for ""stone"". The scientists referred to the fossils as ""Rolling Stoneflies"". The Rolling Stones' first concert was on 12 July 1962 at the Marquee Club in London ... &lt;/information&gt;</p> <p>&lt;think&gt; I have found that a Hells Angel member was responsible for stabbing and beating an attendant at the 1969 Altamont Free Concert. To identify which specific Hells Angel member is being referred to, I will search for the individual who committed this act. &lt;/think&gt;</p> <p>&lt;search&gt; individual Hells Angel member who stabbed and beat an attendant at Altamont Free Concert &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "The Rolling Stones") and former members of the band. Two species, Petroperla mickjaggeri and Lapisperla keithrichardsi, were placed within a new family Petroperlidae. The new family was named in honour of the Rolling Stones, derived from the Greek ""petra"" that stands for ""stone"". The scientists referred to the fossils as ""Rolling Stoneflies"". The Rolling Stones' first concert was on 12 July 1962 at the Marquee Club in London ... &lt;/information&gt;</p> <p>&lt;think&gt; After searching for the individual Hells Angel member who stabbed and beat an attendant at the Altamont Free Concert, I have identified that the member was "Ronnie Biggs" &lt;/think&gt;</p> <p>&lt;answer&gt; Charlie ISHAM &lt;/answer&gt;</p>
<p><b>w. E5:</b></p> <p>&lt;think&gt; I need to find out which Hells Angel member stabbed and beat an attendant of the 1969 counterculture-era rock concert in the United States. I'll search for it. &lt;/think&gt;</p> <p>&lt;search&gt; which Hells Angel member stabbed and beat an attendant of the 1969 counterculture-era rock concert in the United States &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "Death of Meredith Hunter") Death of Meredith Hunter Meredith Curly Hunter, Jr. (October 24, 1951 – December 6, 1969) was an 18-year-old African-American teen who was killed at the 1969 Altamont Free Concert. During the performance by The Rolling Stones, Hunter approached the stage, and was violently driven off by members of the Hells ... &lt;/information&gt;</p> <p>&lt;think&gt; I found out that during the 1969 Altamont Free Concert, a fan named Meredith Hunter was stabbed and beaten to death by Hells Angel Alan Passaro. Now I need to provide the answer. &lt;/think&gt;</p> <p>&lt;answer&gt; Alan Passaro &lt;/answer&gt;</p>

Table 10: Case B study of LLM-based search agent trained with different search engines (*i.e.*, random noise, BM25, and E5).

<b>Question:</b> Jacques Mathou starred in Delicatessen, the 1991 French movie in the style of which celebrated American animator?
<b>Ground Truth:</b> Terry Gilliam
<b>w. random noise:</b>
<p>&lt;think&gt; To answer this question, we need to consider the animated style of Jacques Mathou’s film "Delicatessen." The movie is known ... American animator. Therefore, the celebrated American animator that "Delicatessen" is in the style of is Tim Burton. &lt;/think&gt;</p> <p>&lt;answer&gt; Tim Burton &lt;/answer&gt;</p>
<b>w. BM25:</b>
<p>&lt;think&gt; To find out which celebrated American animator Jacques Mathou starred in the 1991 French movie "Delicatessen," I will search for information about the movie and its connections ... &lt;/think&gt;</p> <p>&lt;search&gt; Delicatessen 1991 French movie celebrated American animator &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "Jean-Pierre Jeunet") Jean-Pierre Jeunet Jean-Pierre Jeunet (; born 3 September 1953) is a French film director, producer, and screenwriter. His films are known to mix elements of fantasy, reality and science fiction either to create idealized realities or to give relevance to mundane situations ... &lt;/information&gt;</p> <p>&lt;think&gt; From the search results, I have gathered that "Delicatessen" is a 1991 French movie directed by Jean-Pierre Jeunet and Marc Caro. The movie is a post-apocalyptic black comedy. To find the ... &lt;/think&gt;</p> <p>&lt;search&gt; Delicatessen celebrated American animator connection &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: Delicatessen) items such as candy and mints are also usually available. Menus vary according to regional ethnic diversity. Although urban delis rely on ethnic meats (such as pastrami, corned beef and salami), supermarket delis rely on meats similar to their packaged meats (primarily ham, turkey and American bologna) ... &lt;/information&gt;</p> <p>&lt;think&gt; I have found that the movie "Delicatessen" is associated with Jean-Pierre Jeunet, who is a celebrated film ... for the animator mentioned in the question. &lt;/think&gt;</p> <p>&lt;search&gt; Jacques Mathou Delicatessen American animator &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "Betty Blue") going to the publisher’s house, slashes his face. Zorg induces him to drop charges by threatening him with violence, saying she is the only good thing in his life and she is all he has. Eddy’s mother dies and the friends go to the funeral in Marvejols. There, Eddy asks Zorg and Betty if they will live in the dead woman’s house and look after ... &lt;/information&gt;</p> <p>&lt;think&gt; Upon further search, I have found that Jacques Mathou starred in "Delicatessen," but he is not an animator. The question seems to be focused on an American animator related to "Delicatessen." Since I lack this information, .. &lt;/think&gt;</p> <p>&lt;answer&gt; Fail to answer &lt;/answer&gt;</p>
<b>w. E5:</b>
<p>&lt;think&gt; I need to find out which celebrated American animator Delicatessen, the 1991 French movie, is in the style of. I’ll search for it. &lt;/think&gt;</p> <p>&lt;search&gt; Delicatessen, the 1991 French movie in the style of which celebrated .. &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "Delicatessen (film)") Delicatessen (film) Delicatessen is a 1991 French post-apocalyptic black comedy film directed by Jean-Pierre Jeunet and Marc Caro, starring Dominique Pinon and Karin Viard. It was released in North America as ""presented by Terry Gilliam."" In a dilapidated apartment building in a post-apocalyptic country, food is in short supply ... &lt;/information&gt;</p> <p>&lt;think&gt; I found out that Delicatessen is a 1991 French post-apocalyptic black comedy film directed by Jean-Pierre Jeunet and Marc Caro. Now I need to find out which celebrated American animator it is in the style of. &lt;/think&gt;</p> <p>&lt;search&gt; which celebrated American animator is Delicatessen in the style of &lt;/search&gt;</p> <p>&lt;information&gt; Doc 1(Title: "Joseph Barbera") Joseph Barbera Joseph Roland Barbera (; ; March 24, 1911 – December 18, 2006) was an American animator, director, producer, storyboard artist, and cartoon artist, whose film and television cartoon characters entertained millions of fans worldwide for much of the 20th century. He was born to Italian immigrants in New York City, &lt;/information&gt;</p> <p>&lt;think&gt; I found out that Delicatessen is in the style of Terry Gilliam. &lt;/think&gt;</p> <p>&lt;answer&gt; Terry Gilliam &lt;/answer&gt;</p>

In Section 6.1, we have shown that LLMs trained with different search engines will learn different reasoning and search engine calling patterns. In particular, the LLM trained with random noise will learn to not call the search engine since the return from the search engine is not informative; the LLM trained with a weak search engine (*e.g.*, BM25) tend to call the search engine multiple times, while the LLM trained with a stronger search engine (*e.g.*, E5) can call the search engine in a more reasonable pattern.

In this section, we would like to show case studies of LLMs trained with different search engine as below in Table 9 and Table 10. The inference time search engine is the same to the training time search engine.

From these case studies, we find that during training, higher-quality search engines that provide more relevant information can encourage the agent to achieve its objectives with fewer search calls, as the retrieved content more effectively supports reasoning and decision-making. In contrast, lower-quality search engines that return less relevant information may lead the agent to either over-rely on its internal knowledge or issue multiple search queries to compensate for inadequate results.