SEARCH SELF-PLAY: PUSHING THE FRONTIER OF AGENT CAPABILITY WITHOUT SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has become the mainstream technique for training LLM agents. However, RLVR highly depends on well-crafted task queries and corresponding ground-truth answers to provide accurate rewards, which requires massive human effort and hinders the RL scaling processes, especially under agentic scenarios. Although a few recent works explore task synthesis methods, the difficulty of the synthetic agentic task can hardly be controlled to provide effective RL training advantages. Towards more effective agentic training, we explore self-play training for deep search agents, in which the learning LLM utilizes multi-turn search engine calling and acts simultaneously as a task proposer and a problem solver. The task proposer aims to generate deep search queries with well-defined ground-truth answers and increasing task difficulty. The problem solver tries to handle the generated search queries and predict the correct ground-truth answer. To ensure each proposed search query has accurate ground truth, we collect all the searching results from the proposer's trajectory as the external knowledge, then conduct retrieval-augmentation generation (RAG) to test whether the proposed query can be correctly answered with all necessary search documents provided. Within our search self-play (SSP) game, the proposer and the solver co-evolve their agentic capabilities via both competition and cooperation. With substantial experimental results, we find that SSP can significantly improve search agents' performance uniformly on various benchmarks without any supervision under both from-scratch and continuous RL training setups.

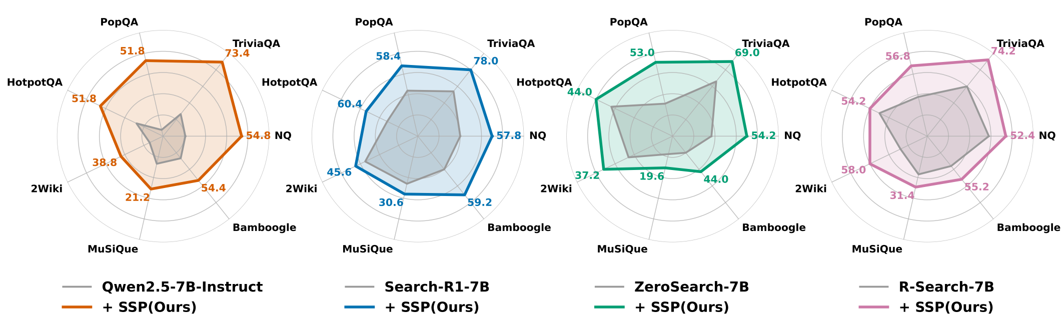


Figure 1: Performance gains of deep search agents continuously trained via our Search Self-play (SSP) across various agentic benchmarks. Our SSP agents uniformly surpass several strong open-source baselines without any agentic data annotation and additional supervision.

1 INTRODUCTION

The rapid evolution of large language models (LLMs) has culminated in systems with unprecedented reasoning capabilities. Recent breakthroughs, demonstrated by models like O1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025a), have dramatically advanced the ability of AI to tackle complex, multi-step problems. To translate these powerful capabilities into practical, real-world applications, the field has seen a surge in the development of sophisticated AI agents. A new wave of systems, such as DeepResearch (OpenAI, 2025), Grok-3 (x.ai, 2025), and Kimi-Researcher (Moonshot AI, 2025), now function as autonomous agents capable of performing complex, end-to-end research tasks.

Central to the functionality of these advanced agents is their ability to perform deep, iterative information seeking. This is where Deep Search emerges as a critical reasoning paradigm for Large Language Models (LLMs) to solve complicated questions. In this paradigm, models autonomously explore vast knowledge bases, iteratively retrieving and reasoning over evidence to find accurate, verifiable answers to complex, open-ended questions (Huang et al., 2025b). In contrast to traditional Retrieval-Augmented Generation (RAG), Deep Search employs multi-hop reasoning, dynamic query reformulation, and self-guided exploration to emulate a human-like investigative process (Lewis et al., 2020; Xi et al., 2025). This set of capabilities is crucial for applications that demand high precision and traceability, such as scientific literature review (OpenAI, 2025), legal analysis (Li et al., 2024), and fact-checking (Wei et al., 2024).

Recent advances leverage reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025a) to train LLM agents for Deep Search (Jin et al., 2025b;a; Chen et al., 2025b; Team, 2025a). In this paradigm, the agent performs a *rollout*—a sequence of actions like querying a search engine and navigating web pages—to arrive at an answer for a given question. This final answer is then programmatically compared against a known *ground-truth* answer to assign a clear, verifiable reward (e.g., 1 for correct, 0 for incorrect), which is used to optimize the agent's policy. The primary advantage of RLVR is that it provides a direct and unambiguous training signal, which is critical for learning complex reasoning chains. By tying the reward to factual correctness, it mitigates the risk of reward hacking common in systems that rely on proxy or estimated rewards, thus fostering more reliable and trustworthy agents (Lightman et al., 2023). However, this approach relies heavily on large-scale, expert-curated question-answer (QA) pairs (Team, 2025b; Wei et al., 2025). The labor-intensive and time-consuming nature of acquiring and validating this data leads to scarcity, which in turn poses a significant bottleneck to the scalability of RLVR systems (Liang et al., 2025b).

To mitigate the data scarcity issue, existing approaches typically synthesize QA pairs as a preliminary step before the RL post-training phase (Li et al., 2025; Gao et al., 2025a). However, this strategy suffers from two critical limitations: Scalability is inherently constrained (Villalobos et al., 2024), as each synthesized QA pair must be rigorously validated for correctness and logical coherence to ensure training fidelity. Offline synthesis lacks adaptability (Guo et al., 2025b). It cannot dynamically adjust question difficulty or complexity in response to the evolving capabilities of the agent during training. This rigidity ultimately limits both learning efficiency and performance gains. Consequently, existing approaches lack a scalable, self-sustaining algorithm that can autonomously generate high-quality QA pairs without human intervention.

Self-play methods, pioneered by AlphaGo Zero (Silver et al., 2017), offer a promising solution to these challenges. Through a self-sustaining learning loop, an agent learns by playing against itself and continuously improves through reinforcement learning. This paradigm effectively addresses both data scarcity and adaptability issues by autonomously generating training examples that naturally scale in difficulty as the agent improves.

Inspired by this self-play approach, we introduce Search Self-Play (SSP), a framework for training continuously self-improving search agents through RLVR without human supervision. This framework centers on a single LLM that adopts two alternating roles: a *question proposer* and a *problem solver*. The proposer generates progressively challenging and verifiable deep search queries, which the solver then attempts to answer using multi-turn search interactions. A core component of SSP is its validation mechanism: each query is verified via RAG using the full set of documents that the proposer itself retrieved. This process guarantees that a well-defined, ground-truth answer exists and is recoverable from the provided evidence. This closed-loop design allows the agent to autonomously generate high-quality training signals, eliminating the need for human-annotated data while maintaining reward fidelity. Through this iterative game of competition and collaboration, the proposer and solver co-evolve, systematically improving the agent's core skills in search, reasoning, and self-verification. Our empirical results show that SSP yields substantial and consistent improvements across multiple benchmarks in both from-scratch and continual learning settings, establishing a scalable pathway toward self-supervised agentic learning.

2 Related Work

2.1 Deep Search

Deep Search agents have recently attracted substantial attention and rapid progress (Xi et al., 2025). Proprietary systems such as DeepResearch (OpenAI, 2025), Grok-3 (x.ai, 2025), and Kimi-Researcher (Moonshot AI, 2025) demonstrate strong performance on web search and information synthesis tasks, but their model designs, training data, and training procedures remain opaque. In contrast, open-source efforts—including Search-R1 (Jin et al., 2025c), R1-Searcher (Song et al., 2025), DeepResearcher (Zheng et al., 2025), and ZeroSearch (Sun et al., 2025b)—leverage agentic reinforcement learning (RL) to enhance question-answering capabilities, yet still rely on existing training sets. To push the capability frontier of web-search agents, works such as WebDancer (Wu et al., 2025), WebSailor (Li et al., 2025), and ASearcher (Gao et al., 2025b) propose data synthesis pipelines; however, these synthesis processes remain offline. Our approach departs from this paradigm: via self-play, it simultaneously improves the agent's problem-proposing and problem-solving abilities without human-annotated data, freeing training from the constraints of offline synthetic corpora and from the limitations of the model's internal knowledge.

2.2 Self-Play in Large Language Models

The self-play paradigm, wherein a model adopts dual roles to create a self-improving loop, has recently been explored to advance the capabilities of language models without reliance on human data. For instance, (Cheng et al., 2024) applied self-play to the *Adversarial Taboo* game to enhance model reasoning, but their method employs offline updates and remains confined to a simple word game. Other works, such as (Fang et al., 2025; Liang et al., 2025a), generate problems from seed data but only train the solver model, leaving the problem-generation capability static and preventing a co-evolutionary dynamic. A more advanced line of research trains both the proposer and solver models concurrently. (Chen et al., 2025a; Huang et al., 2025a; Kuba et al., 2025) have demonstrated the effectiveness of this approach on diverse tasks like mathematics, code generation, and instruction following. Our work makes two fundamental distinctions from these prior self-play approaches. First, our Proposer generates problems with inherent ground truth by crafting corresponding questions from externally retrieved and verifiable information. Second, by equipping the Proposer with search tools, our framework actively grounds the problem-generation process in external knowledge, thereby breaking free from the limitations of the model's internal, static knowledge base.

3 Methodology

Denote $\mathcal{N} := \{\boldsymbol{x} = (x^1, x^2, \dots, x^N) : N \in \mathbb{N}_+, x^i \in \mathcal{V}, i = 1, 2, \dots, N\}$ as the natural language sequence space, where $\mathcal{V}$ is the vocabulary set of tokens. An auto-regressive next-token prediction policy of LLM π_θ iteratively outputs next-token $x^{i+1} \sim \pi_\theta(\cdot|\boldsymbol{x}^{1:i})$ with θ as the model parameters, where $\boldsymbol{x}^{1:i} = (x^1, x^2, \dots, x^i)$ is the length-i prefix of the natual language sequence $\boldsymbol{x}$.

The a search agent trajectory can be written as: $\boldsymbol{\tau} = (\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{o}_1, \boldsymbol{y}_2, \boldsymbol{o}_2, \dots, \boldsymbol{y}_{T-1}, \boldsymbol{o}_{T-1}, \boldsymbol{y}_T)$, where $\boldsymbol{x} \in \mathcal{N}$ is the input prompt, each $\boldsymbol{y}_t \in \mathcal{N}$ is the LLM output at the t-th step, and $\boldsymbol{o}_t \in \mathcal{N}$ is the corresponding observation returned by the search tools at the i-th step. We model the search agent exploration as a token-level Markov decision process (Littman, 1994) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$. $\mathcal{S}$ is naturally the language sequence space $\mathcal{N}$. $\mathcal{A}$ is the vocabulary set $\mathcal{V}$ for token-level action generation. The transition $\mathcal{T}$ directly appends the newly-generated token y_t^{i+1} to the end of $\boldsymbol{y}_t^{1:i}$ if $\boldsymbol{y}_t^{1:i}$ has not formed a complete search tool call, or additionally appends the t-step observation $\boldsymbol{o}_t$ if $\boldsymbol{y}_t$ is finished. $r(\boldsymbol{\tau})$ assigns the outcome reward to $\boldsymbol{\tau}$ as the judgment of the agent's performance, which we will discuss design details in 3.2. Given a search agent system prompt $\boldsymbol{x}_{\text{sys}}$ and a user query $\boldsymbol{q}$, we can use a LLM policy π_θ to induce the search agent policy $u(\cdot|\boldsymbol{x}) = \pi_\theta(\cdot|\boldsymbol{x}_{\text{sys}}, \boldsymbol{q})$. For notation simplification, we denote $\boldsymbol{\tau} \sim u(\cdot|\boldsymbol{x})$ as collecting the trajectory $\boldsymbol{\tau}$ from the search agent policy $u(\cdot|\boldsymbol{x})$.

3.1 Search Self-play Design

We focus on exploring the benefits of self-play training for deep search agents, enabling LLMs to self-improve their agentic capabilities without additional supervision. To achieve this objective,

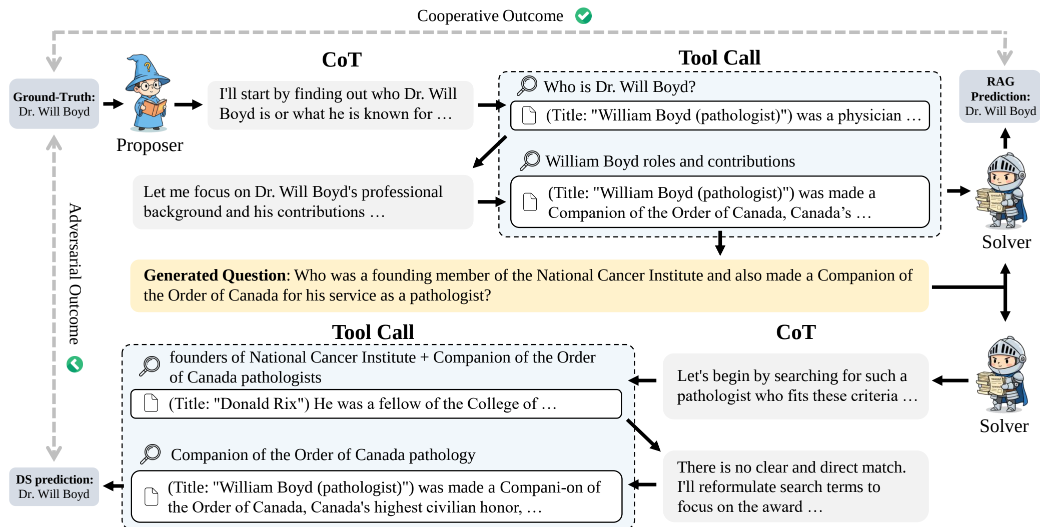


Figure 2: Examples of Search Self-Play with a given ground-truth *"Dr. Will Boyd"*. Provided with the ground-truth, the *proposer* iteratively uses search tools to excavate implicit factual evidence, then generates a challenging search question. Then the solver uses all search results from the proposer's trajectory as the RAG materials to predict the answer without searching, for the validation of the question's correctness. Once verified, the solver follows the ordinary deep search pipeline to explore the solution via multi-turn agentic rollout.

we consider using the search agent to act as a *question proposer* to generate challenging questions via multi-turn deep search tool usage. Meanwhile, given the generated questions, we let the same LLM agent act as a *problem solver* to seek the answer, as ordinary deep search agents do. The proposer aims to generate increasingly challenging questions to puzzle the solver, whereas the solver is dedicated to improving its answer correctness, no matter how difficult the generated questions are. Based on the above rules, the search self-play can be regarded as a zero-sum adversarial game. We suppose both the proposer and the solver can evolve through this intense competition.

However, the above SSP rules can be easily hacked: the proposer can constantly generate incorrect questions so that the solver can never solve. Hence, to verify the correctness of the generated question from the proposer, we collect all the search results in the proposer's trajectory as the RAG documents, and let the solver answer without using search tools. If the proposer's question is correct and the corresponding search actions are meaningful, with the RAG documents, the solver should have sufficient information to correctly predict the ground-truth answer. By this additional verification progress, we successfully avoid the search self-play game from degeneration. The verification constraint requires the proposer and the solver to cooperate, which enhances the SSP game with both cooperation and competition. An example of the search self-play game is shown in Figure 2.

3.2 Search Self-play Modeling

We use different system prompts $\boldsymbol{x}_{\text{propose}}$ and $\boldsymbol{x}_{\text{solve}}$ to let the LLM learning policy π_θ act as the proposer and the solver, respectively. Given a ground-truth answer $\boldsymbol{a}$, the policy for the question proposer is $u(\cdot|\boldsymbol{a}) = \pi_\theta(\cdot|\boldsymbol{x}_{\text{propose}}, \boldsymbol{a})$. After the proposer generates a question $\boldsymbol{q}$, the solver policy tries to settle the question with the policy $v(\cdot|\boldsymbol{q}) = \pi_\theta(\cdot|\boldsymbol{x}_{\text{solve}}, \boldsymbol{q})$. Denote $\boldsymbol{\tau}$ and $\boldsymbol{\rho}$ as the corresponding trajectories of the proposer and the solver, respectively. Then the adversarial self-play training objective is:

$$\min_u \max_v \mathbb{E}_{\boldsymbol{a}^*\sim\mathcal{D}, \boldsymbol{\tau}\sim u(\cdot|\boldsymbol{a}=\boldsymbol{a}^*), \boldsymbol{\rho}\sim v(\cdot|\boldsymbol{q}=\mathcal{Q}(\boldsymbol{\tau}))}[r(\mathcal{A}(\boldsymbol{\rho}), \boldsymbol{a}^*)], \tag{1}$$

where $\boldsymbol{a}^*$ is a ground-truth answer drawn from a pre-defined answer set $\mathcal{D}$. $\mathcal{Q}(\cdot)$ and $\mathcal{A}(\cdot)$ extract the generated question and predicted answer from the proposer trajectory $\boldsymbol{\tau}$ and the solver trajectory $\boldsymbol{\rho}$, respectively. $r(\mathcal{A}(\boldsymbol{\tau}), \boldsymbol{a}^*)$ is a binary outcome judgment function to check whether the solver's prediction $\mathcal{A}(\boldsymbol{\tau})$ and the ground-truth answer $\boldsymbol{a}^*$ are semantically equivalent (which means

Algorithm 1 Search Self-play training process.

Require: LLM policy π_θ; ground-truth answer set $\mathcal{D}$; proposer and solver prompts $(\boldsymbol{x}_{\text{propose}}, \boldsymbol{x}_{\text{solve}})$.

for each parameter-updating step **do**
  Sample a batch of ground-truth answers $\{\boldsymbol{a}_i^*\}_{i=1}^B \sim \mathcal{D}$ with batch size B.
  Proposer generates candidate questions $\mathbb{Q} = \{\mathcal{Q}(\boldsymbol{\tau}_i)\}_{i=1}^B$ with each $\boldsymbol{\tau}_i \sim \pi_\theta(\cdot|\boldsymbol{x}_{\text{propose}}, \boldsymbol{a}_i^*)$.
  Filter out valid questions as $\mathbb{Q}^*$ with format rules and the RAG constraint:

$$r(\mathcal{A}(\boldsymbol{\sigma}_i), \boldsymbol{a}_i^*) = 1, \text{ for } \boldsymbol{\sigma}_i \sim \pi_\theta(\boldsymbol{x}_{\text{solve}}, \mathcal{Q}(\boldsymbol{\tau}_i)).$$

  for each question $\boldsymbol{q}_i \in \mathbb{Q}^*$ **do**
    Solver explores n trajectories for solution: $\boldsymbol{\rho}_i^j \sim \pi_\theta(\cdot|\boldsymbol{x}_{\text{solve}}, \boldsymbol{q}_i), j = 1, 2, \ldots, n$
    Compute solver's reward of each trajectory: $r_{\text{solve},i}^j = r(\mathcal{A}(\boldsymbol{\rho}_i^j), \boldsymbol{a}_i^*)$
    Compute proposer's reward in expectation: $\bar{r}_{\text{propose},i} = 1 - \frac{1}{n}\sum_{j=1}^n r_{\text{solve},i}^j$
  end for
  Update π_θ with solver's trajectories and outcomes $\{(\boldsymbol{\rho}_i^j, r_{\text{solve},i}^j)\}$ via GRPO.
  Update π_θ with proposer's trajectories and outcomes $\{(\boldsymbol{\tau}_i, \bar{r}_{\text{propose},i})\}$ via REINFORCE.
end for

$r(\mathcal{A}(\boldsymbol{\tau}), \boldsymbol{a}^*) = 1$). To ensure accurate judgment, we implement $r(\mathcal{A}(\boldsymbol{\tau}), \boldsymbol{a}^*)$ with an LLM-as-a-judge function, where the corresponding prompt and judge critics are described in Appendix D.

To make sure the generated question $\boldsymbol{q} = \mathcal{Q}(\boldsymbol{\tau})$ is solvable and correct with respect to the ground-truth $\boldsymbol{a}^*$, we need additional constraints. Therefore, we use the solver agent to $v(\cdot|\boldsymbol{q}, \boldsymbol{O}_T) = \pi_\theta(\cdot|\boldsymbol{x}_{\text{solve}}, \boldsymbol{q}, \boldsymbol{O}_T)$ to verify the correctness of the generated question $\mathcal{Q}(\boldsymbol{\tau})$, where $\boldsymbol{O}_T = (\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T) = \mathcal{O}(\boldsymbol{\tau})$ is the collection of all the search results from the proposer trajectory. Then the proposer and the solver need to cooperate to maximize the solver's answer accuracy under RAG setups:

$$\max_u \mathbb{E}_{\boldsymbol{a}^* \sim \mathcal{D}, \boldsymbol{\tau} \sim u(\cdot|\boldsymbol{a}=\boldsymbol{a}^*), \boldsymbol{\sigma} \sim v(\cdot|\boldsymbol{q}=\mathcal{Q}(\boldsymbol{\tau}), \boldsymbol{O}_T=\mathcal{O}(\boldsymbol{\tau}))}[r(\mathcal{A}(\boldsymbol{\sigma}), \boldsymbol{a}^*)]. \tag{2}$$

In practice, we find that jointly optimizing both the cooperation and competition objectives suffers from training inefficiency. Because the cooperative objection in equation 2 requires the proposed question to be completely correct, otherwise the optimization of equation 1 could lose its effectiveness due to the reward hacking. Therefore, we use rejection sampling (Liu et al., 2025) strategy for the cooperative objective instead. More specifically, we dynamically filter the generated questions with $r(\mathcal{A}(\boldsymbol{\sigma}), \boldsymbol{a}^*) = 1$ to collect a full batch of valid questions to optimize the adversarial objective in equation 1. Therefore, the overall training objective of search self-play is:

$$\begin{aligned} &\min_u \max_v \mathbb{E}_{\boldsymbol{a}^* \sim \mathcal{D}, \boldsymbol{\tau} \sim u(\cdot|\boldsymbol{a}=\boldsymbol{a}^*), \boldsymbol{\rho} \sim v(\cdot|\boldsymbol{q}=\mathcal{Q}(\boldsymbol{\tau}))}[r(\mathcal{A}(\boldsymbol{\rho}), \boldsymbol{a}^*)], \\ &\text{s.t. } \mathbb{E}_{\boldsymbol{\sigma} \sim v(\cdot|\boldsymbol{q}=\mathcal{Q}(\boldsymbol{\tau}), \boldsymbol{O}_T=\mathcal{O}(\boldsymbol{\tau}))}[r(\mathcal{A}(\boldsymbol{\sigma}), \boldsymbol{a}^*)] = 1. \end{aligned} \tag{3}$$

3.3 SSP Implementation

We describe the updating details of the SSP optimization as equation 3 in Algorithm 1. As discussed in Section 3.2, invalid questions generated by the proposer will hinder the training effectiveness of SSP. Therefore, we applied two filtering strategies to improve the quality of generated questions: *rule-based filtering* and *RAG verification*.

Rule-based filtering ensures LLM has legal output for the question generation task. More specifically, each proposer's output should have a correct format for extracting the question (within `<question></question>` tags). Furthermore, we conduct several additional rule-based checks to pre-filter the low-quality questions and reduce the computational consumption before the RAG verification process. The bad cases to filter include: (1) empty question string; (2) no search tool invoked; (3) excessively short question; (4) containing the original answer in the question.

After the rule-based filtering, we applied the RAG verification process as described in Section 3.2. We collect the search results from the proposer's trajectory as the RAG documents, then let the solver answer the generated question with the provided RAG materials. To further increase the robustness of the verification judgment, we mix some unrelated documents from other trajectories within the same batch to simulate more real RAG scenarios. Details and ablation studies about adding irrelevant RAG noises are discussed in Section 4.3.2.

When the outcome reward calculation finished, each generated question $\boldsymbol{q}_i$ had been tried by the solver for n times, yielding a group of trajectories $\{\boldsymbol{\rho}_i^j\}_{j=1}^n$ and corresponding binary rewards $\{r_{\text{solve},i}^j\}_{j=1}^n$, where $r_{\text{solve},i}^j = r(\mathcal{A}(\boldsymbol{\rho}_i^j), \boldsymbol{a}_i^*)$. A natural updating method for the solver's policy v is Group Relative Policy Optimization (GRPO) (Shao et al., 2024), which uses the average reward of the group as a baseline to reduce variance. The solver aims to maximize its reward, so its loss function for a given question $\boldsymbol{q}_i$ is:

$$\nabla_\theta \mathcal{L}_{\text{GRPO}}(\theta) = \frac{1}{B}\sum_{i=1}^{B}\left[\frac{1}{n}\sum_{j=1}^{n}\frac{1}{|\boldsymbol{\rho}_i^j|}\sum_{t=1}^{|\boldsymbol{\rho}_i^j|}\nabla_\theta \log \pi_\theta(\rho_i^{j,t}|\boldsymbol{q}_i, \boldsymbol{\rho}_i^{j,1:t-1}) \cdot \hat{A}_i^j - \beta \nabla_\theta \mathrm{KL}[\pi_\theta || \pi_{\text{ref}}]\right] \quad (4)$$

where the advantage $\hat{A}_i^j = r_{\text{solve},i}^j - \frac{1}{n}\sum_{k=1}^n r_{\text{solve},i}^k$ is calculated for the j-th trajectory as the difference between its reward and the group's average reward for question $\boldsymbol{q}_i$.

Conversely, the proposer is updated to generate questions that are more challenging for the solver, which aligns with the min-max objective in equation 3. As defined in Algorithm 1, the proposer receives a high reward if the solver fails. We use the REINFORCE (Williams, 1992) algorithm to update the proposer's policy u. The loss function aims to increase the log-probability of generating trajectories that result in high proposer reward (i.e., low solver success rate):

$$\nabla_\theta \mathcal{L}_{\text{REINFORCE}}(\theta) = \frac{1}{B}\sum_{u=1}^{B}\left[R(\boldsymbol{\tau}_i)\sum_{t=1}^{|\boldsymbol{\tau}_i|}\nabla_\theta \log \pi_\theta(\tau_i^t|\boldsymbol{a}_i^*, \boldsymbol{\tau}_i^{1:t-1})\right], \quad (5)$$

where $R(\boldsymbol{\tau}_i) = 1 - \frac{1}{n}\sum_{j=1}^n r_{\text{solve},i}^j$. This update encourages the proposer to generate increasingly difficult questions to continuously challenge the task solver. Unlike prior question proposing methods, which only use LLMs' internal knowledge, our SSP utilizes interactions with external environments to acquire information for question generation. Moreover, our SSP verifies the correctness of the generated question with a verifiable RAG pipeline, which is more credible than previous synthetic methods, such as majority vote (Huang et al., 2025a).

4 Experiments

4.1 Experimental Setups

Benchmarks. Seven standard question-answering benchmarks are adopted in our evaluation, including NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2022), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2022). Following the practice in prior works (Sun et al., 2025c;a; Zhao et al., 2025; Gao et al., 2025a; Deng et al., 2025; Tan et al., 2025), we randomly sample 500 QA pairs for evaluation on each benchmark to reduce evaluation overhead while maintaining statistical reliability, while all 125 test samples in Bamboogle are used for evaluation.

Baselines. To verify the effectiveness of SSP, we select open-source pretrained LLMs of different sources and model sizes used for deep search, including Qwen2.5-Instruct (Yang et al., 2024), LLaMA3.1-8B (Dubey et al., 2024), Qwen3-8B (Yang et al., 2025), Search-R1 (Jin et al., 2025b;a), ZeroSearch (Sun et al., 2025a), and R-Search (Zhao et al., 2025).

Search Tools. A local E5 retriever with a Wikipedia 2018 corpus (Karpukhin et al., 2020) is incorporated in our training and evaluation, which retrieves the top-3 related documents for each query.

Evaluation Metrics. Following recent work (Gao et al., 2025a), we adopt LLM-as-a-judge as standard metric for evaluation. Qwen2.5-32B-Instruct (Yang et al., 2024) is deployed as the judge model. All results are reported in terms of pass@1 accuracy.

Training Details. We implemented our method using the SGLang asynchronous multi-turn tool-integrated rollout in veRL (Sheng et al., 2024). The proposer is optimized using REIN-

Table 1: Main experimental results. SSP delivers strong gains across from-scratch training, generalization across architectures, continual training on search-specialized agents, and scaling to larger models. All scores are on a 100-point scale. Bold (black) indicates the better score within each baseline v.s. +SSP pair.

	GeneralQA			Multi-HopQA				
Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	**Avg**
From-Scratch Training on Base and Instruct Models								
Qwen2.5-7B-Base	32.0	33.2	25.0	18.0	10.8	11.0	26.4	22.3
+ SSP	**54.2**$^{+22.2}$	**73.6**$^{+40.4}$	**56.0**$^{+31.0}$	**52.8**$^{+34.8}$	**33.2**$^{+22.4}$	**24.0**$^{+13.0}$	**47.2**$^{+20.8}$	**48.7**$^{+26.4}$
Qwen2.5-7B-Instruct	44.2	64.0	36.4	45.0	32.8	16.8	51.2	41.5
+ SSP	**54.8**$^{+10.6}$	**73.4**$^{+9.4}$	**51.8**$^{+15.4}$	**51.8**$^{+6.8}$	**38.8**$^{+6.0}$	**21.2**$^{+4.4}$	**54.4**$^{+3.2}$	**49.5**$^{+8.0}$
Generalization Across Model Families								
LLaMA-3.1-8B	50.2	65.2	45.8	34.6	19.4	11.4	30.4	36.7
+ SSP	**58.0**$^{+7.8}$	**75.8**$^{+10.6}$	**55.4**$^{+9.6}$	**44.2**$^{+9.6}$	**34.4**$^{+15.0}$	**16.2**$^{+4.8}$	**40.0**$^{+9.6}$	**46.3**$^{+9.6}$
Qwen3-8B	53.6	76.0	50.8	54.2	48.0	26.6	58.4	52.5
+ SSP	**56.0**$^{+2.4}$	**78.2**$^{+2.2}$	**55.0**$^{+4.2}$	**58.0**$^{+3.8}$	**51.5**$^{+3.5}$	**28.0**$^{+1.4}$	**67.2**$^{+8.8}$	**56.3**$^{+3.8}$
Continual Training on Search-Specialized Agents								
ZeroSearch-7B	52.2	66.6	50.2	43.2	34.6	17.6	40.8	43.6
+ SSP	**54.2**$^{+2.0}$	**69.0**$^{+2.4}$	**53.0**$^{+2.8}$	**44.0**$^{+0.8}$	**37.2**$^{+2.6}$	**19.6**$^{+2.0}$	**44.0**$^{+3.2}$	**45.9**$^{+2.3}$
Search-R1-7B	56.6	75.4	57.2	58.2	45.2	29.6	55.2	53.9
+ SSP	**57.8**$^{+1.2}$	**78.0**$^{+2.6}$	**58.4**$^{+1.2}$	**60.4**$^{+2.2}$	**45.6**$^{+0.4}$	**30.6**$^{+1.0}$	**59.2**$^{+4.0}$	**55.7**$^{+1.8}$
R-Search-7B	50.8	71.0	53.8	54.0	56.4	29.8	53.6	52.8
+ SSP	**52.4**$^{+1.6}$	**74.2**$^{+3.2}$	**56.8**$^{+3.0}$	**54.2**$^{+0.2}$	**58.0**$^{+1.6}$	**31.4**$^{+1.6}$	**55.2**$^{+1.6}$	**54.6**$^{+1.8}$
Scaling to Larger Models								
Qwen2.5-14B-Instruct	56.0	77.0	53.8	57.0	48.4	26.6	64.8	54.8
+ SSP	**57.4**$^{+1.4}$	**77.8**$^{+0.8}$	**54.6**$^{+0.8}$	**61.2**$^{+4.2}$	**49.4**$^{+1.0}$	**28.0**$^{+1.4}$	**69.6**$^{+4.8}$	**56.9**$^{+2.1}$
Qwen2.5-32B-Instruct	58.0	78.4	53.4	57.0	48.4	27.4	63.2	55.1
+ SSP	**62.6**$^{+4.6}$	**82.8**$^{+4.4}$	**55.0**$^{+1.6}$	**62.8**$^{+5.8}$	**49.2**$^{+0.8}$	**32.0**$^{+4.6}$	**69.6**$^{+6.4}$	**58.5**$^{+3.4}$

FORCE (Williams, 1992), while the solver is updated with GRPO (Shao et al., 2024). Key hyperparameters are: learning rate 1e-6 with 5 warmup steps; global batch size 256 (mini-batch 128); maximum sequence lengths of 4,096 tokens for prompts and up to 8,192 tokens for responses; and a training horizon of approximately 150–200 steps. More implementation details and prompts used in each agent role are provided in Appendix A & D; full configurations are summarized in Table 4.

4.2 Main Results

The main experimental results, summarized in Table 1, demonstrate that Search Self-Play is a highly effective and versatile method for enhancing LLM agent capabilities.

Our primary finding is that SSP yields substantial improvements when training models from scratch without any external supervision. The gains are particularly pronounced for base models that have not undergone instruction tuning; for instance, applying SSP to Qwen2.5-7B-Base results in an impressive average improvement of 26.4 points, including a remarkable **+40.4** gain on TriviaQA. SSP also benefits instruction-tuned models, improving Qwen2.5-7B-Instruct by 8.0 points on average. Moreover, SSP proves to be model-agnostic, consistently enhancing models from different architectural families, including LLaMA-3.1 and Qwen3.

Additionally, SSP serves as an effective continual training strategy. Even when applied to strong open-source models that have already been extensively trained on search-oriented tasks (e.g., Search-R1, R-Search), our method consistently yields further performance improvements. Furthermore, this performance gain holds as we scale to larger models, where applying SSP to Qwen2.5-32B-Instruct helps achieve state-of-the-art results, securing the best scores on five of the seven benchmarks. These comprehensive results validate that SSP is a robust framework for enhancing agent capability across diverse model sizes, architectures, and initial search skill levels.

Table 2: Ablation on training schemes. Full search self-play (**+SSP**) significantly outperforms fixed-opponent variants, underscoring the necessity of co-evolution for robust performance gains.

	GeneralQA			Multi-HopQA				
Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	**Avg.**
Qwen2.5-7B-Instruct	0.442	0.640	0.364	0.450	0.328	0.168	0.512	0.415
+SSP (Solver-Only)	0.452	0.682	0.466	0.472	0.326	0.210	0.488	0.442
+SSP (Proposer-Only)	0.524	0.690	0.504	0.448	0.288	0.142	0.320	0.417
+SSP	**0.548**	**0.734**	**0.518**	**0.518**	**0.388**	**0.212**	**0.544**	**0.495**

4.3 Ablation Studies

4.3.1 Efficacy of Self-Play versus Fixed-Opponent Training

The co-evolution of the proposer and solver is critical for pushing the frontier of agent capability. We investigate this core hypothesis through an ablation study comparing our full Search Self-Play framework against two fixed-opponent schemes: training only the solver, denoted as Solver-Only, and training only the proposer, denoted as Proposer-Only. As detailed in Table 2, the results clearly demonstrate the superiority of SSP. Our SSP achieves the highest average score, substantially outperforming both fixed-opponent variants. The training dynamics, analyzed in Figure 3, reveal the reasons behind this performance gap.

Solver-Only. Figure 3 (a) reveals the underlying issue with Solver-Only. The solver's in-game reward rapidly saturates near 0.9, indicating that it quickly masters the static distribution of tasks from the fixed proposer. Lacking a progressively challenging curriculum, the solver begins to overfit. This is confirmed by its performance on held-out evaluation sets (Figure 3(b) and (c)), where scores on NQ and 2Wiki initially improve but then degrade over time.

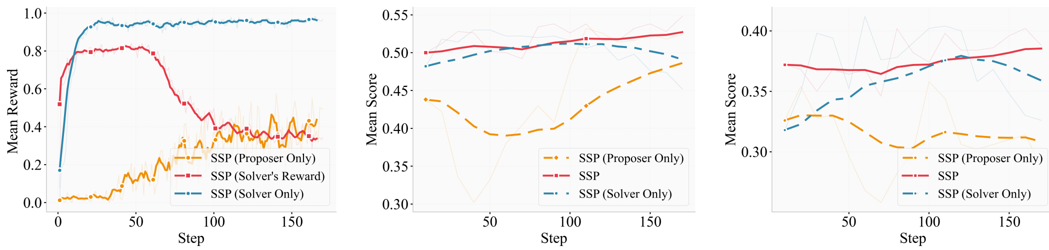
Proposer-Only. Conversely, the Proposer-Only setting shows different limitations. While its in-game reward also rises, its evaluation performance on NQ and 2Wiki initially declines before a slight recovery. We attribute this partial recovery to the proposer learning general tool-use skills, which incidentally aids the fixed solver. This effect is more pronounced on simpler GeneralQA benchmarks like NQ, where this setting eventually surpasses Solver-Only. However, this general skill enhancement is insufficient for complex multi-hop reasoning, resulting in weaker performance on Multi-HopQA datasets compared to Solver-Only.

In stark contrast to the flawed dynamics of fixed-opponent training, our full SSP framework facilitates a healthy co-evolution. The training dynamics for SSP offer a compelling narrative. As shown in Figure 3(a), the solver's in-game reward initially rises, but unlike the saturating curve of the Solver-Only setting, it later experiences a slight decline. This dip is not a sign of performance degradation, but rather crucial evidence of the proposer's co-evolution: it has learned to generate more difficult tasks that challenge the improving solver, thus lowering its success rate. This dynamic creates a robust and adaptive curriculum where task difficulty perpetually adjusts to the solver's current skill level, preventing overfitting and forcing continuous learning. Consequently, this internal adversarial pressure translates into the stable and sustained performance gains observed on the external evaluation benchmarks in Figure 3 (b) and (c). This confirms that mutual co-evolution between the proposer and solver is decisively superior to fixed-opponent training.

4.3.2 Ablation on the RAG verification

A critical component of our SSP framework is the RAG verification, which uses RAG process to ensure that each proposed question is valid and answerable given the evidence collected by the proposer. We conduct an ablation study to quantify the impact of this strategy and also optimize its configuration. All experiments in this section are conducted on Qwen2.5-7B-Instruct, with results presented in Table 3.

First, we compare our SSP against a variant with no RAG verification. As shown in results, removing RAG verification leads to a significant performance degradation, particularly on GeneralQA benchmarks. This result confirms our hypothesis that the RAG verification is crucial for quality



(a) In-Game Training Reward (b) Evaluation Score on NQ (c) Evaluation Score on 2Wiki

Figure 3: Training dynamics of different SSP variants. (a) shows the in-game reward. (b) and (c) display the evaluation accuracy on the held-out NQ and 2Wiki datasets over training steps.

control; it effectively prunes invalid questions where the proposed ground-truth answer is not actually supported by the retrieved documents, thus preventing the solver from being trained on noisy or incorrect data.

Next, we examine the impact of introducing noisy documents into the context used for RAG verification. These documents are randomly sampled from the corpus and were *not* retrieved by the proposer during its search trajectory. The goal of this strategy is to prevent the proposer from "hacking" the self-play dynamic. Without such regularization, the proposer could exploit the RAG-based solver's access to a complete set of retrieved documents by crafting questions that are answerable in the verifier's omniscient context, but unfairly difficult for a solver operating under realistic search constraints. A notable failure mode involves generating questions with non-unique answers: the proposer may locate and reinforce evidence for one particular answer, which the RAG verifier accepts, while a search-based solver might reasonably retrieve evidence supporting a different, equally plausible answer and be wrongly penalized.

Table 3: Ablation on the RAG verification. Performance (Avg. Score) improves with a moderate number of noisy context documents.

Config.	**GeneralQA**	**Multi-HopQA**
No RAG Verifi.	49.5	36.7
RAG verification w/ Noisy Docs:		
0 Noisy Docs	58.5	38.2
1 Noisy Docs	58.5	36.9
4 Noisy Docs	**60.0**	**41.6**
7 Noisy Docs	57.8	35.9

By introducing noisy documents, we challenge the solver in RAG verification. Now, it must verify the ground truth answer from a context that contains relevant and irrelevant information, making it more difficult to validate ambiguous questions. This forces the proposer to generate more robust questions whose answers are strongly and uniquely entailed by the evidence, rather than being trivially verifiable only in a noise-free context. The results show that adding a moderate amount of noise is beneficial. The configuration with 4 noisy documents achieves the best performance on both benchmarks. However, adding too much noise (e.g., 7 noisy documents) becomes counterproductive, as the excessive irrelevant information likely confuses the RAG solver, reducing its ability to accurately verify question answerability and leading to a drop in performance. Therefore, we use the setting with 4 noisy documents for our main experiments.

5 Conclusion

We introduce Search Self-Play, a reinforcement learning approach for training deep search agents by making the LLM policy act as both the question proposer and the problem solver, which alleviates the reliance on human-crafted supervision. Through a self-play game grounded in multi-turn search engine interactions, SSP dynamically generates verifiable search tasks with controllable difficulty and ensures reward accuracy via retrieval-augmented verification of ground-truth answers. Extensive experiments demonstrate that SSP consistently enhances search agent performance across diverse benchmarks under both from-scratch and continuous training setups, without requiring any external human supervision. These results highlight the potential of self-play as a scalable and autonomous paradigm for agentic LLM training, paving the way for more efficient and self-sustaining reinforcement learning systems in complex information-seeking scenarios.

REFERENCES

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-Questioning Language Models, August 2025a. URL http://arxiv.org/abs/2508.03682. arXiv:2508.03682 [cs].

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning, March 2025b. URL http://arxiv.org/abs/2503.19470. arXiv:2503.19470 [cs].

Pengyu Cheng, Tianhao Hu, Han Xu, Zhisong Zhang, Yong Dai, Lei Han, nan du, and Xiaolong Li. Self-playing adversarial language game enhances llm reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 126515–126543, 2024.

Yong Deng, Guoqing Wang, Zhenzhe Ying, Xiaofeng Wu, Jinzhen Lin, Wenwen Xiong, Yuqin Dai, Shuo Yang, Zhanwei Zhang, Qiwen Wang, Yang Qin, and Changhua Meng. Atom-Searcher: Enhancing Agentic Deep Research via Fine-Grained Atomic Thought Reward, August 2025. URL http://arxiv.org/abs/2508.12800. arXiv:2508.12800 [cs].

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. SeRL: Self-Play Reinforcement Learning for Large Language Models with Limited Data, May 2025. URL http://arxiv.org/abs/2505.20347. arXiv:2505.20347 [cs].

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025a.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond Ten Turns: Unlocking Long-Horizon Agentic Search with Large-Scale Asynchronous RL, August 2025b. URL http://arxiv.org/abs/2508.07976. arXiv:2508.07976 [cs].

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*, 2025a.

Yiduo Guo, Zhen Guo, Chuanwei Huang, Zi-Ang Wang, Zekai Zhang, Haofei Yu, Huishuai Zhang, and Yikang Shen. Synthetic data rl: Task definition is all you need. *arXiv preprint arXiv:2505.17063*, 2025b.

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

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-Evolving Reasoning LLM from Zero Data, August 2025a. URL http://arxiv.org/abs/2508.05004. arXiv:2508.05004 [cs].

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, Jianye Hao, Kun Shao, and Jun Wang. Deep Research Agents: A Systematic Examination And Roadmap, June 2025b. URL http://arxiv.org/abs/2506.18096. arXiv:2506.18096 [cs].

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O Arik, and Jiawei Han. An empirical study on reinforcement learning for reasoning-search interleaved llm agents. *arXiv preprint arXiv:2505.15117*, 2025a.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, 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*, 2025b.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning, April 2025c. URL `http://arxiv.org/abs/2503.09516`. arXiv:2503.09516 [cs].

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

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)*, pp. 6769–6781, 2020.

Jakub Grudzien Kuba, Mengting Gu, Qi Ma, Yuandong Tian, and Vijai Mohan. Language Self-Play For Data-Free Training, September 2025. URL `http://arxiv.org/abs/2509.07414`. arXiv:2509.07414 [cs].

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Haitao Li, Junjie Chen, Jingli Yang, Qingyao Ai, Wei Jia, Youfeng Liu, Kai Lin, Yueyue Wu, Guozhi Yuan, Yiran Hu, et al. Legalagentbench: Evaluating llm agents in legal domain. *arXiv preprint arXiv:2412.17259*, 2024.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. WebSailor: Navigating Super-human Reasoning for Web Agent, July 2025. URL `http://arxiv.org/abs/2507.02592`. arXiv:2507.02592 [cs].

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. Beyond pass@ 1: Self-play with variational problem synthesis sustains rlvr. *arXiv preprint arXiv:2508.14029*, 2025a.

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. Beyond Pass@1: Self-Play with Variational Problem Synthesis Sustains RLVR, August 2025b. URL `http://arxiv.org/abs/2508.14029`. arXiv:2508.14029 [cs].

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Haoxiong Liu, Yifan Zhang, Yifan Luo, and Andrew C Yao. Augmenting math word problems via iterative question composing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24605–24613, 2025.

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

Moonshot AI. Kimi researcher. Project page, Moonshot AI, 2025. URL `https://moonshotai.github.io/Kimi-Researcher/`. Accessed: 2025-09-19.

OpenAI. Deep research system card. System card, OpenAI, 2025. URL `https://cdn.openai.com/deep-research-system-card.pdf`. Version 2025a.

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.

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

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

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, March 2025. URL `http://arxiv.org/abs/2503.05592`. arXiv:2503.05592 [cs].

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

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. ZeroSearch: Incentivize the Search Capability of LLMs without Searching, May 2025b. URL `http://arxiv.org/abs/2505.04588`. arXiv:2505.04588 [cs].

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, et al. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*, 2025c.

Jiejun Tan, Zhicheng Dou, Yan Yu, Jiehan Cheng, Qiang Ju, Jian Xie, and Ji-Rong Wen. HierSearch: A Hierarchical Enterprise Deep Search Framework Integrating Local and Web Searches, August 2025. URL `http://arxiv.org/abs/2508.08088`. arXiv:2508.08088 [cs].

MiroMind AI Team. Miroflow: An open-source agentic framework for deep research. `https://github.com/MiroMindAI/MiroFlow`, 2025a.

Tongyi DeepResearch Team. Tongyi-deepresearch. `https://github.com/Alibaba-NLP/DeepResearch`, 2025b.

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

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: will we run out of data? limits of llm scaling based on human-generated data. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A Simple Yet Challenging Benchmark for Browsing Agents, April 2025. URL `http://arxiv.org/abs/2504.12516`. arXiv:2504.12516 [cs].

Jerry Wei, Chengrun Yang, Xinying Song, Yifeng Lu, Nathan Hu, Jie Huang, Dustin Tran, Daiyi Peng, Ruibo Liu, Da Huang, et al. Long-form factuality in large language models. *Advances in Neural Information Processing Systems*, 37:80756–80827, 2024.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebDancer: Towards Autonomous Information Seeking Agency, May 2025. URL `http://arxiv.org/abs/2505.22648`. arXiv:2505.22648 [cs].

x.ai. Grok 3 beta — the age of reasoning agents, 2025. URL `https://x.ai/news/grok-3`. Accessed: 2025-09-19.

Yunjia Xi, Jianghao Lin, Yongzhao Xiao, Zheli Zhou, Rong Shan, Te Gao, Jiachen Zhu, Weiwen Liu, Yong Yu, and Weinan Zhang. A survey of llm-based deep search agents: Paradigm, optimization, evaluation, and challenges. *arXiv preprint arXiv:2508.05668*, 2025.

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.

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

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An Open-Source LLM Reinforcement Learning System at Scale, May 2025. URL `http://arxiv.org/abs/2503.14476`. arXiv:2503.14476 [cs].

Qingfei Zhao, Ruobing Wang, Dingling Xu, Daren Zha, and Limin Liu. R-search: Empowering llm reasoning with search via multi-reward reinforcement learning. *arXiv preprint arXiv:2506.04185*, 2025.

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, April 2025. URL `http://arxiv.org/abs/2504.03160`. arXiv:2504.03160 [cs].

A Implementation Details

A.1 Training Hyperparameter

The primary hyperparameters for our main experiments are detailed in Table 4. Any parameters not explicitly mentioned in the table adhere to the default settings provided by the Verl framework (Sheng et al., 2024).

Table 4: Experimental hyperparameter configuration. The table is divided into two sections: Base Settings for the main training process, and specific settings for the Search Self-Play module.

Base Settings	
Parameter	**Value**
Global Batch Size	256
Mini-batch Size	128
Learning Rate	$1e^{-6}$
Learning Rate Warmup Steps	5
Max Prompt Length	4096
Max Response Length	8192
KL Loss Coefficient	0.01
Validation Temperature	0.0
Rollout Temperature	1.0
Search Self-Play Settings	
Parameter	**Value**
— Proposer —	
Proposer Warm-up Steps	-1 (disabled)
Proposer Advantage Estimator	`REINFORCE`
Proposer Samples (n)	1
— Solver —	
Solver Advantage Estimator	`grpo`
Solver Samples (n)	5
— Other —	
Extraction Failure Strategy	`reuse`
Use RAG Verification	True
Noisy RAG Documents	4

A.2 Rewards Design

Solver. A simple binary outcome reward is adopted for Solver:

$$r_{\text{solve}} = \mathbb{I}(y = y^*) \tag{6}$$

where y^* is the ground-truth answer and $\mathbb{I}(\cdot)$ is the indicator function that checks for equality between the predicted and true answers.

Proposer. The Solver attempts to answer the same question n times, where the average success rate for that question is $\bar{r}_{solve} = \frac{1}{n}\sum_{i=1}^{n} r_{solve}^{(i)}$. Thus the reward for Proposer can be formulated as:

$$r_{\text{propose}} = 1 - \bar{r}_{\text{solve}} \tag{7}$$

Tool-integrated rollout is an interactive sequence of reasoning and tool invocation, with the search tool providing external information. Following Search-R1 (Jin et al., 2025b;a), we mask the content within the <information></information> tags to exclude their loss computation during training to maintain stability. Proposer and Solver responses are required to follow our specified format, using tags such as <think>, <search>, <answer>, and <question>. Responses that do not adhere to this format will receive zero reward.

A.3 OTHERS

For the baseline models used for continuous RL training (Search-R1 (Jin et al., 2025b;a), ZeroSearch (Sun et al., 2025a), and R-Search (Zhao et al., 2025)), we start the SSP training from the best-performing checkpoint as reported in their respective papers. For example, Search-R1-7B corresponds to the *SearchR1-nq_hotpotqa_train-qwen2p5-7b-em-ppo-v0p2* checkpoint, ZeroSearch-7B to the *ZeroSearch_wiki_V2_Qwen2.5_7B* checkpoint, and R-Search-7B to the *R-Search-7b-grpo* checkpoint.

During the question proposing rollout, a batch of questions is significantly reduced in size after passing through the online filter. To mitigate this, we replenish the batch using dynamic sampling (Yu et al., 2025) or a QA replay buffer. This ensures that the rewards within a training batch are less sparse, thus contributing to a more stable training process.

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 TRAINING DYNAMICS

We provide a granular view into the training process by analyzing the training dynamics of our SSP framework on the Qwen2.5-7B-Base model, configured with the Replay Buffer (Periodic Reset) strategy. Figure 4 illustrates several key metrics throughout the training, showcasing how the agent's behavior and performance co-evolve.

As shown in Figure 4a, the average number of search tool calls per trajectory steadily increases over time. This trend is a clear indicator that through search self-play, the agent learns to conduct more extensive and complex multi-step searches to solve problems, significantly enhancing its tool-use capabilities. Simultaneously, Figure 4b shows that the solver's response length also grows, suggesting it learns to generate more detailed and comprehensive answers. In contrast, the prompt length remains relatively stable, indicating consistent task generation from the proposer.

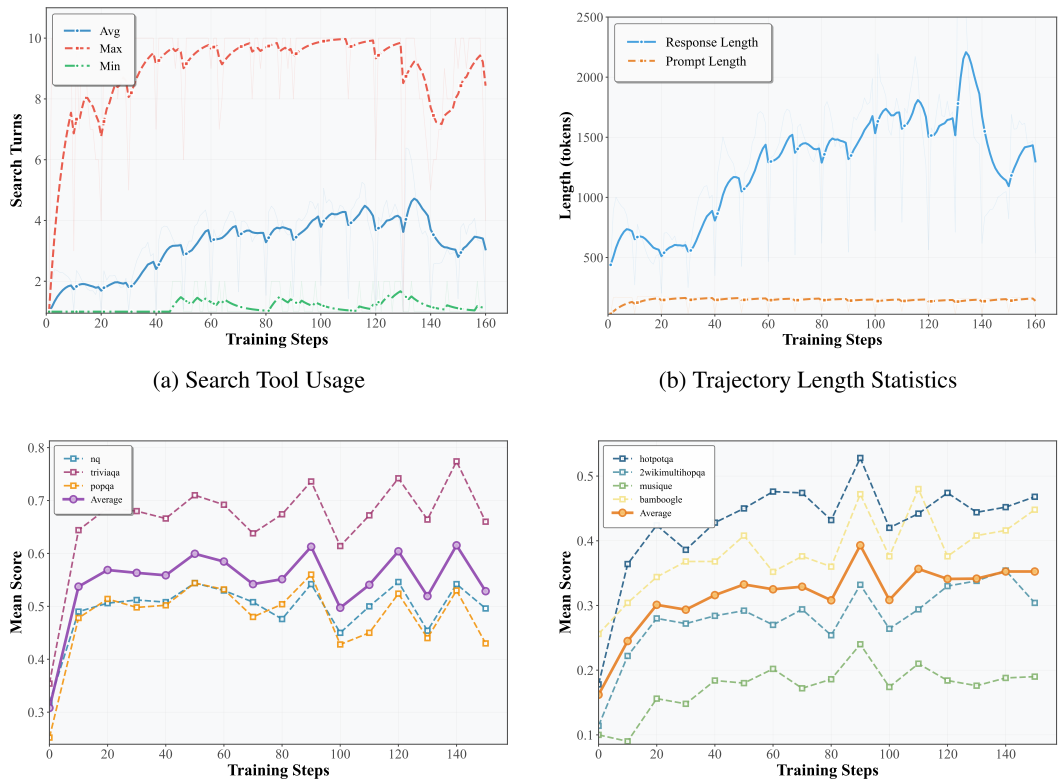
Figures 4c and 4d demonstrate a consistent and significant improvement in accuracy across both GeneralQA and Multi-HopQA datasets as training progresses. Notably, the slope of performance improvement gradually decreases in later training stages. We hypothesize that this plateau is partially attributable to a resource-imposed constraint: the maximum number of search steps was capped at 10 to conserve computational resources. This limit may have become a bottleneck, preventing the agent from exploring even deeper reasoning paths. We believe that scaling this search step limit could unlock further performance improvements.

B.2 ABLATION ON BATCH COMPLETION STRATEGIES

In our SSP framework, the proposer's generation process is stochastic, and not all generated questions pass the online filter. This can result in training batches smaller than the target batch size, leading to sparse reward signals and potential training instability. To address this, we investigate four distinct batch completion strategies to ensure a full batch is always available for the RL update step. All experiments are conducted on the Qwen2.5-7B-Base model, with results summarized in Table 5.

The strategies are as follows:

- **Dummy Padding:** Invalid slots in a batch are filled with a generic, non-informative "dummy" problem. This is the simplest approach but provides no learning signal for the padded slots.
- **Dynamic Resampling:** The proposer continues to generate new questions until a full batch of valid problems is collected. This ensures every sample in the batch is novel but can be computationally expensive if the valid question rate is low.
- **Replay Buffer (Full Reuse):** We maintain a replay buffer of all previously generated valid questions. Invalid slots are filled by sampling from this buffer. This guarantees a dense training signal but risks solver overfitting and proposer policy stagnation.



(a) Search Tool Usage

(b) Trajectory Length Statistics

(c) Performance on GeneralQA Benchmarks

(d) Performance on Multi-HopQA Benchmarks

Figure 4: Training dynamics for Qwen2.5-7B-Base with SSP. (a) The agent learns to use the search tool more frequently. (b) Solver response length increases while prompt length remains stable. (c, d) Evaluation scores on both GeneralQA and Multi-HopQA datasets show consistent improvement throughout training.

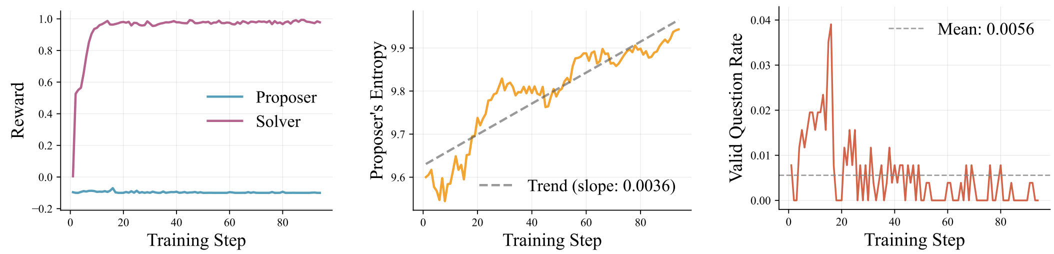
- **Replay Buffer (Periodic Reset):** This is the strategy used in our main experiments. It is identical to Full Reuse, but the replay buffer is cleared every 10 training steps. This is designed to balance the efficiency of reuse with the need for data novelty.

Table 5: Ablation study on batch completion strategies. Performance is evaluated on the Qwen2.5-7B-Base model. The Periodic Reset strategy achieves the best results, demonstrating the importance of balancing data reuse and novelty.

Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	**Avg.**
Qwen2.5-7B-Base (Baseline)	32.0	33.2	25.0	18.0	10.8	11.0	26.4	22.3
Dummy Padding	48.4	68.8	49.6	40.8	22.6	19.0	40.8	41.4
Dynamic Resampling	48.4	66.4	45.8	44.6	31.4	17.6	42.4	42.4
Replay Buffer (Full Reuse)	51.2	67.8	49.0	46.0	31.0	17.8	**48.0**	44.4
Replay Buffer (Periodic Reset)	**54.2**	**73.6**	**56.0**	**52.8**	**33.2**	**24.0**	47.2	**48.7**

As shown in Table 5, the choice of strategy has a profound impact on training outcomes. The **Dummy Padding** approach yields the smallest improvement over the baseline. Its low performance can be attributed to severe reward sparsity; with many invalid proposals, both the proposer and solver receive fewer learning signals per iteration, hindering effective optimization. **Dynamic Resampling** performs slightly better, as it guarantees a full batch of novel, valid questions. However, this comes at a high, often prohibitive, computational cost, as it requires repeated generation cycles.

The introduction of a **Replay Buffer (Full Reuse)** provides a significant performance boost, improving the average score from 42.4 to 44.4. This strategy allows the solver to learn more thoroughly



(a) In-Game Reward Dynamics (b) Proposer Policy Entropy (c) Valid Question Rate

Figure 5: Training dynamics of SSP when the proposer receives a negative reward for format errors. (a) The proposer's reward (blue) trends to -0.1 as it stops producing valid questions, while the solver's reward (purple) spuriously increases due to overfitting on a question pool. (b) The proposer's policy entropy steadily rises as the agent tries to escape the negative rewards, leading to more random and less valid outputs. (c) Consequently, the rate of generating valid questions collapses, halting productive self-play training and demonstrating the instability caused by punitive rewards.

from each valid question generated by the proposer by reusing it for multiple training updates. This densifies the reward signal and enhances training efficiency. However, its gains are ultimately limited. We hypothesize this is because unbounded reuse allows the solver to train on the same questions too many times, leading to overfitting on the static pool of questions within the ever-growing buffer. Concurrently, the proposer's learning signal diminishes as the solver masters these old questions, potentially causing policy degradation.

The **Replay Buffer (Periodic Reset)** strategy emerges as the clear winner, achieving the highest scores across nearly all benchmarks and boosting the average score to 48.7. This method represents an effective trade-off between sufficient data exposure and novelty. By reusing questions for a limited period, it allows the solver to learn sufficiently from each generated task, ensuring the reward signal remains dense. However, by periodically clearing the buffer, it prevents the solver from learning the same questions too many times, thus mitigating the overfitting observed with full reuse. Concurrently, this forces the proposer to continuously generate novel questions to populate the fresh buffer, maintaining a strong co-evolutionary pressure. This result validates our choice of this strategy for the main experiments, as it fosters the most stable and effective self-play dynamic.

B.3 The Impact of Proposer's Reward Design

We analyze the sensitivity of our Search Self-Play framework to the reward function by conducting an experiment on the effect of a punitive reward structure for the proposer. In our main configuration, the proposer receives a zero reward for generating an invalid or malformed question. In this experimental variant, we introduce a small penalty, setting reward as -0.1 for any question that fails the online filter. For this setup, the proposer is optimized using the REINFORCE algorithm with a single sample per prompt ($n = 1$).

The results, shown in Figure 5, demonstrate that this seemingly minor change leads to a catastrophic failure of the training process. The proposer's average reward becomes sparse and progressively declines (Figure 5a), directly corresponding to a collapse in the valid question generation rate, which plummets towards -0.1 (Figure 5c). This phenomenon can be explained as a negative feedback loop: the penalty for format errors encourages the agent to explore away from its current policy. However, this exploration, manifested as an increase in policy entropy (Figure 5b), makes the generation more random and thus more likely to produce invalid outputs. This "death spiral" effectively halts the creation of new training instances. Meanwhile, the solver's reward appears to increase, but this is a misleading artifact of overfitting. As the supply of new, valid questions from the proposer dwindles, the solver is repeatedly trained on a small, static buffer of past questions, failing to generalize its capabilities. This experiment critically underscores that the proposer's reward design is paramount for stable co-evolution in SSP; a punitive approach can destabilize the entire self-play dynamic, highlighting the need for a carefully calibrated reward scheme.

C Self-play Examples

Table 6, 7, 8, 9, 10, and 11 provide examples of trajectories in our SSP game.

Table 6: Example 1: Question Proposing

Proposer creates a question that leads to the answer *Roger (alien)*

<think>Given that I have a single search requirement, it’s essential to design a straightforward but intricate-enough route to come to the term "Roger (alien)" that would normally require a direct search. This will differentiate "Roger (alien)" from other similar or general terms to ensure the question is precise and unambiguous.

Starting with "Roger (alien)," this seems like a reference to a specific character or entity. "Roger" is a common name. The term "alien" specifically refers to an alien from other galaxies or universes.

I’ll brainstorm some ’associated concepts’ related to "Roger (alien)." If it is a specific cultural reference or a name from specific media content, I can build my questions around these concepts.

Let’s begin by searching for ’associated’ or ’preceding’ facts which connect to "Roger (alien)" without specifying the answer directly.</think>

<search> Roger alien </search>

<information> (Title: "Roger (American Dad!)") Roger is a fictional character in the adult animated sitcom ""American Dad!"", voiced by Seth MacFarlane. The character was created and designed by Seth MacFarlane. Roger is a centuries-year-old grey space alien living with the Smith family, around whom the show revolves. Having . . .

(Title: "Big O (mecha)") locks itself and lets Roger out because R. Dorothy is in danger. In ""Enemy Is . . .

(Title: "Roger (Hellboy)") Roger the Homunculus is a fictional character featured in the . . . </information>

<think>It appears that the most prominent "Roger (alien)" refers to a character named Roger from the animated sitcom, "American Dad!," who is known to be an extraterrestrial being from another planet. Given this, the associated concept is now clear, and we can build the question accordingly. We need to find a particular instance or unique detail linked to Roger from American Dad! which would require a solver to search to reach the final answer.</think>

<search> Roger alien american dad! </search>

<information> (Title: "Roger (American Dad!)") Roger is a fictional character in the adult animated . . .

(Title: "Roger (American Dad!)") be shown on the series. Although in the episode ""Lost in Space"", a brief clip . . .

(Title: "American Dad!") alien race. To date, Roger’s birth planet has yet to be revealed . . . </information>

<think>Based on the information gathered, I see that there’s a notable character detail that can be used—the fact that Roger has a backstory linked to Roswell crash. Given that it’s quite specific and links directly to the search results, we can use this idea to form our single-search question.</think>

<question> Who is the grey alien character from American Dad! that was involved in the Roswell incident in 1947? </question>

Table 7: Example 1: Question Answering

Solver answers the question raised by Proposer: *Who is the grey alien character from American Dad! that was involved in the Roswell incident in 1947?*

<think>To answer this question, I need to identify a grey alien character from American Dad! that has a connection to the Roswell incident in 1947. I’m not sure off-hand which character is known for this, as the show features many alien characters. Searching for the details will likely retrieve the relevant information.</think>

<search> Grey alien character from American Dad! Roswell incident 1947 </search>

<information> (Title: "Grey alien") as ""visitors"", appear in two episodes of ""South Park"", and Roger . . .

(Title: "Grey alien") small-bodied beings with smooth grey-colored skin, enlarged hairless heads and large . . .

(Title: "Roger (American Dad!)") Roger is a fictional character in the adult animated sitcom ""American Dad!"", voiced by Seth MacFarlane. The character was created and designed by Seth MacFarlane. Roger is a centuries-year-old grey space alien living with the Smith family, around whom the show revolves. Having lived on Earth since crashing in Roswell, New Mexico in 1947, Roger came to live with the Smiths after rescuing main character Stan Smith at Area 51 four years prior to the beginning of the series. . . . </information>

<think>Based on the provided information, the grey alien character from American Dad! that was involved in the Roswell incident in 1947 is Roger. The text mentioned that Roger is a grey space alien living with the Smith family, and he has been on Earth since crashing in Roswell, New Mexico in 1947.</think>

<answer> Roger </answer>

Table 8: Example 2: Question Proposing

Proposer creates a question that leads to the answer *Castle Drogo*
<think>The target answer is Castle Drogo. To create a question that requires three sequential searches, I need to find an associated concept for the final answer and reverse-engineer the question from there. First, let's identify an associated concept for Castle Drogo. Castle Drogo is a 20th-century castle located in the South West of England, situated in Devon. The associated concept could be the architect behind it, Sir Edwin Lutyens. Next, I'll search for information connecting Sir Edwin Lutyens to Castle Drogo to find a 'preceding fact' for the third search.</think>
<search> Sir Edwin Lutyens and Castle Drogo </search>
<information> (Title: "Castle Drogo") Castle Drogo Castle Drogo is a country house and castle near Drewsteignton, Devon, England. Constructed between 1911 and 1930, it was the last castle to be built in England. The client was Julius Drewe, the hugely successful founder of the Home and Colonial Stores. Drewe chose the site in the belief that it formed part of the lands of his supposed medieval ancestor, Drogo de Teigne. The architect he chose to realise his dream was Edwin Lutyens, then at the height of his career. Lutyens lamented Drewe's determination to . . . (Title: "Castle Drogo") purely decorative. Additionally, the castle had electricity and lifts from the outset, with power being supplied by two turbines on the river below. The castle is a Grade I listed building. . . . (Title: "Castle Drogo") the village of Drewsteignton in the belief . . . </information>
<think>From the search results, we know that Edwin Lutyens designed Castle Drogo. The preceding fact for the third search could be the architect who designed Castle Drogo. Now, I need to find another piece of information that leads logically to Edwin Lutyens as the answer, which will be the second search. I know that Sir Edwin Lutyens was a notable 20th-century English architect, known for his numerous significant works, including collaborations with Gertrude Jekyll. Let's explore his collaboration with Jekyll further. Lutyens frequently collaborated with the famous garden designer Gertrude Jekyll. I'll find a specific project or fact related to this collaboration.</think>
<search> Edwin Lutyens and Gertrude Jekyll </search>
<information> (Title: "Gertrude Jekyll") wife Julia Hammersley. Her younger brother, Walter Jekyll (an Anglican priest; sometime Minor Canon of Worcester Cathedral and Chaplain of Malta) . . . (Title: "Gertrude Jekyll") was awarded the Victoria Medal of Honour of the Royal Horticultural Society in 1897 and the Veitch Memorial Medal of the society in 1929. Also in 1929, she was given the George Robert White Medal . . . (Title: "Hestercombe House") the house by Henry Hall in the 1870s. The Edwardian garden was laid out by Gertrude Jekyll and Edwin Lutyens between 1904 and 1906 for . . . </information>
<think>From the search results, I can see that Edwin Lutyens and Gertrude Jekyll collaborated on several projects, particularly Hestercombe House. The preceding fact for the second search could be a well-known collaboration project of Edwin Lutyens and Gertrude Jekyll. For the final step, I need to identify an associated concept related to Hestercombe House that leads to Edwin Lutyens as the answer. I can create a sequential search question that starts from a well-known architectural or garden feature associated with Hestercombe House. The Hestercombe Garden is a well-known aspect. The next fact for the first search could be related to a designer or architect's masterpiece associated with Hestercombe Garden.</think>
<search> Hestercombe Garden architect </search>
<information> (Title: "Hestercombe House") petrol and air. When the house and gardens were inherited . . . (Title: "Hestercombe House") the house by Henry Hall in the 1870s. The Edwardian garden was laid out by Gertrude Jekyll and Edwin Lutyens between 1904 and 1906 for the Hon E.W.B. Portman, resulting in a garden ""remarkable for the bold, concise pattern of its layout, and for the minute attention to detail everywhere to be seen in the variety . . . (Title: "Hestercombe House") the hardy herbaceous planting espoused by Miss Jekyll. Lutyens also designed the orangery about 50 m east of the main house between 1904–09, which is now Grade I listed . . . </information>
<think>The search results indicate that Hestercombe House and Garden were designed by Edwin Lutyens and Gertrude Jekyll. This association can lead us to the architect Edwin Lutyens in a search starting from a prominent aspect of the Hestercombe Garden. The question can start by referencing a key feature of Hestercombe Garden and logically lead to Edwin Lutyens and later Castle Drogo. To formulate the question, let's start referring to the specific feature, such as the orangery built by Edwin Lutyens.</think>
<question> What is the name of the 20th-century castle in Devon, England, designed by the architect who, alongside Gertrude Jekyll, created the famous Edwardian garden at Hestercombe House, known for its listed orangery? </question>

Table 9: Example 2: Question Answering

Solver answers the question raised by Proposer: *What is the name of the 20th-century castle in Devon, England, designed by the architect who, alongside Gertrude Jekyll, created the famous Edwardian garden at Hestercombe House, known for its listed orangery?*
<think>The question is asking for the name of a 20th-century castle in Devon, England. This castle is designed by an archite ct who also worked with Gertrude Jekyll on the Hestercombe House gardens. The Hestercombe House gardens are f amous for their Edwardian style and especially known for their listed orangery. Given this information, I need to find: 1. The architect who collaborated with Gertrude Jekyll on the Hestercombe House gardens. 2. A 20th-century castle in Devon, England, designed by that architect.</think> <search> architect who collaborated with Gertrude Jekyll on Hestercombe House gardens </search> <information> (Title: "Hestercombe House") the house by Henry Hall in the 1870s. The Edwardian garden was laid out by Gertrude Jekyll and Edwin Lutyens between 1904 and 1906 for the Hon E.W.B. Portman, resulting in a garden ""remarkable for the bold, concise pattern of its layout, and for the minute attention to detail everywhere to be seen in the variety and imaginative handling of contrasting materials, whether cobble, tile, flint, or thinly coursed local stone"". Jekyll and Lutyens were leading participants of the Arts and Crafts movement. Jekyll is remembered for her outstanding designs and subtle, painterly approach to the arrangement of the gardens . . . (Title: "Hascombe Court") Hascombe Court Hascombe Court is a estate in Hascombe, Surrey, best known for its vast garden designed by Gertr ude Jekyll. Hascombe Court is listed Grade II on the National Heritage List for England, and its gardens are also Grad e II listed on the Register of Historic Parks and Gardens. . . . (Title: "Gertrude Jekyll") over 300 photographs taken by Jekyll. From 1881, when she laid out the gardens for Munstead House, built for her mother by John James Stevenson, Jekyll provided designs or planned planting for some four hundred gardens. More than half were directly commissioned, but many were . . . </information> <think>Doc 1 mentions that Edwin Lutyens and Gertrude Jekyll collaborated on the garden at Hestercombe House. With this in mind, I can now search for a 20th-century castle in Devon, England designed by Edwin Lutyens.</think> <search> 20th-century castle in Devon, England designed by Edwin Lutyens </search> <information> (Title: "Castle Drogo") Castle Drogo Castle Drogo is a country house and castle near Drewsteignton, Devon, England. Constructed between 1911 and 1930, it was the last castle to be built in England. The client was Julius Drewe, the hugely successful founder of the Home and Colonial Stores. Drewe chose the site in the belief that it formed part of the lands of his supposed medieval ancestor, Drogo de Teigne. The architect he chose to realise his dream was Edwin Lutyens, then at the height of his career. Lutyens lamented Drewe's determination to . . . (Title: "Devon County War Memorial") The Devon County War Memorial is a First World War memorial, designed by Sir Edwin Lutyens and situated on Cathedral Green in Exeter, the county town of Devon, in the south west of England. It is one of fifteen War Crosses designed by Lutyens with similar characteristics . . . (Title: "Devon County War Memorial") memorials were built across Britain. Amongst the most prominent designers of memorials was the architect Sir Edwin Lutyens, described by Historic England as ""the leading English architect of his generation"". Prior to the First World War, Lutyens established . . . </information> <answer> Castle Drogo </answer>

Table 10: Example 3: Question Proposing

Proposer creates a question that leads to the answer *George Smith*
<think>To create a question that leads to the answer "George Smith," which requires three searches, I need to work backwards and establish a logical sequence of information that will lead to this name. Let's start by identifying an associated concept or related information. 1. **Associated Concept**: George Smith might be a person involved in a specific historical event, a notable figure in a certain field, or someone connected to a particular organization or place. For this, I'll explore who George Smith might be and find a notable fact or place relevant to this person. 2. **Search Process**: - **Search #3**: Start with the target "George Smith." I need to identify an event or place associated with him. - **Search #2**: Once the preceding fact is discovered from Search #3, I'll use that information to identify another intermediary fact. - **Search #1**: Finally, will determine the starting point using the information from Search #2. Let's start with Search #3 to establish the initial associated concept.</think>
<search> Who is George Smith and what is he known for? </search>
<information> (Title: "George P. Smith (politician)") George Peter Smith (August 12, 1873 – November 29, . . . (Title: "George Smith (historian)") George Smith (31 August 1800 – 30 August 1868) was an English businessman . . . (Title: "George Smith (artist)") George Smith (1713/14 – 7 September 1776) was . . . </information>
<think>From the search results, George Smith (politician) from Alberta provides a specific context. George P. Smith played a role in the development of Duhamel, Alberta, and the establishment of influential relationships with indigenous peoples. I will use this in the backward search chain. **Search #3**: Identify a fact related to George P. Smith.</think>
<search> What significant role did George P. Smith play in the community of Duhamel, Alberta? </search>
<information> (Title: "George P. Smith (politician)") George Peter Smith (August 12, 1873 – November 29, . . . (Title: "George P. Smith (politician)") with fraud as they believed that the money which was . . . (Title: "George Wilbert Smith") George Wilbert Smith (April 24, 1855 – August 1, 1931) . . . </information>
<think>From the information, George P. Smith played a significant role in founding the Trading Store in Duhamel, Alberta, and developing it into a hub for fur trading with indigenous peoples. **Search #3 Key Fact**: George P. Smith bought and established a Trading Store in Duhamel, Alberta. Next, we will now build on this fact in Search #2.</think>
<search> What role did trading stores play in the development of settlements like Duhamel, Alberta, in the early 20th century? </search>
<information> (Title: "Duhamel, Alberta") Duhamel is a hamlet in the . . . (Title: "Duhamel, Alberta") soil, and vegetation within the circles had apparently been . . . (Title: "Exchange District") was a new route for shipping goods from Eastern Canada . . . </information>
<think>From the search, trading stores played a crucial role in connecting settlers with resources and indigenous peoples. Tr ading stores in early settlements were central to economic and social development. **Search #2 Key Fact**: Trading stores were central hubs for economic activities, distributing goods and establishing trade relations with indigenous peoples. Finally, we build the starting point for Search #1.</think>
<search> Which early 20th-century Alberta town was a key trading center due to the presence of a significant fur-trading activity? </search>
<information> (Title: "History of Alberta") providing an important channel for the flow . . . (Title: "Lamson & Hubbard Trading Company") Lamson & Hubbard Trading Company (also referred to as . . . (Title: "Northern Traders Company") The Northern Traders Company was an enterprise . . . </information>
<think>From the search, Alberta towns like Red Deer could have been key trading centers due to fur trading activity. Red Deer's role fits well because it was an economic hub with connections to the indigenous communities. **Search #1 Key Fact**: Red Deer, Alberta, was a key trading center due to its fur-trading activity. Now I can construct the final question ensuring it meets all the established requirements and necessitates precisely three searches to reach the final answer.</think>
<question> What was the name of the politician who established a crucial trading store in the early 20th century, which transformed a small community into an important trading hub for fur and developed strong relationships with the indigenous peoples in Alberta? </question>

Table 11: Example 3: Question Answering

Solver answers the question raised by Proposer: *What was the name of the politician who established a crucial trading store in the early 20th century, which trans formed a small community into an important trading hub for fur and developed strong relationships with the indigenous peoples in Alberta?*

<think>The question asks for a politician who established a trading store in Alberta, Canada in the early 20th century. This politician seems to have had substantial influence in the region, not just in politics but also in trade, and built strong relationships with indigenous peoples. The region mentioned, Alberta, was an important area for fur trading during this period.

Given that the politician founded a crucial trading store, it's likely that they were known for their contributions to Canadian history, particularly in the province of Alberta. Notable figures in Alberta during the early 20th century would be relevant here, particularly those linked with economic development through trade or industry, and who engaged in dialogue or relationships with indigenous groups.

To identify this specific politician, more precise historical information about fur trading in Alberta in the early 20th ce ntury is required, especially someone known for such a significant act of setting up a trading hub.</think>

<search> politician who established a trading store in Alberta early 20th century </search>

<information> (Title: "James Ramsey (politician)") James ""Merchant Prince"" Ramsey (April 4, 1864 – December 23, 1939) was a provincial politician and business man from Alberta, Canada . . .

(Title: "Alexander Livingstone (Alberta politician)") Alexander Livingstone (August 2, 1884 – October 26, 1944) was a Canadian politician and municipal councillor in Edmonton, Alberta. Livingstone was born in Bathgate, Scotland . . .

(Title: "Cornelius Hiebert") Cornelius ""Don"" Hiebert (August 2, 1862 – March 21, 1919) was a politician from Alberta, Canada. Hiebert was born in Sea of Azov, Russia and lived in Russia . . . </information>

<think>The initial search results did not directly identify a politician who established a fur trading store or created a significa nt trading hub in early 20th-century Alberta. James Ramsey and Alexander Livingstone were notable business and political figures, but they are primarily mentioned for their work in department stores in Edmonton, not for a specific trading store related to fur or indigenous relations. Cornelius Hiebert was involved in setting up a general store in a founding phase of a village, but not detailed circumstances specifically in trading of fur.

A second search is needed with a more concentrated focus on fur traders and key individuals who had influence in relation to indigenous communities.</think>

<search> politician from Alberta early 20th century established fur trading store </search>

<information> (Title: "James Ramsey (politician)") James ""Merchant Prince"" Ramsey (April 4, 1864 . . .

(Title: "Cornelius Hiebert") Cornelius ""Don"" Hiebert (August 2, 1862 – March 21, 1919) was . . .

(Title: "Alexander Livingstone (Alberta politician)") Alexander Livingstone (August 2 . . . </information>

<think>Despite the multiple searches, there appears a lack of direct information providing an exact match regarding a prominent fur trader politician in Alberta from the early 20th century. The information given points primarily to notable business and political figures linked with department stores or general stores in Alberta but not specifically to fur trading or close relations to indigenous communities as mentioned in the question. As the information is still scattered and does not directly answer the specific individual queried, a reconsideration and possibly with a broader search is required to uncover the correct specific person's name and connection with fur trading in Alberta.</think>

<search> politician who established a fur trading store in Alberta early 1900s </search>

<information> (Title: "George P. Smith (politician)") George Peter Smith (August 12, 1873 – November 29, 1942) was a politician and former Minister of the Crown from Alberta, Canada. He was born in Lieury, Ontario and came west in 1901 while working for an insurance company. After visiting Duhamel (a mission settlement . . .

(Title: "Cornelius Hiebert") Cornelius ""Don"" Hiebert (August 2, 1862 – March 21, 1919) was . . .

(Title: "Alexander Livingstone (Alberta politician)") Alexander Livingstone (August 2 . . . </information>

<think>The information provided initially presents George P. Smith as someone who bought a trading store in 1902 after visiting Duhamel in Alberta and developed a relationship with influential settlers and with Métis and Saulteaux peoples. He formed a lucrative fur trading business, leading to the establishment of a trading hub important for fur trade. This matches well with the criteria outlined in the question. The second document did not specifically mention fur trade, while the third one was about a prominent figure, but not specifically in the context of fur trading. So it strongly indicates that George P. Smith is the person sought.

Given the direct relevance of George P. Smith's background and activities in developing a fur trading hub and formin g relationships with indigenous communities in Alberta, the answer to the question appears to be George P. Smith.</think>

<answer> George P. Smith </answer>

D PROMPTS

Prompts used in our search self-play experiments are listed as follows.

Proposer Prompt

You are an expert question creator. Your primary task is to reverse-engineer a challenging question from a given answer. The question you create must require a solver to perform 'n' sequential searches to solve it. I will provide you with the target answer and the required number of searches, 'n'.

Your Creation Process & Tools:
1. Analyze Scope and Target: Begin by analyzing the provided 'Answer' (your target) and the required number of searches, 'n' (the path's length). This establishes your final destination and the complexity of the logical chain you need to construct.

2. Build the Question by Working Backwards:
This is the core of the process. You will start from the destination and work your way back to the starting point, step by step.
2.1. The Crucial First Step: Connection and Discovery
Start with the final 'Answer', but do not search for the answer itself directly.
Instead, first analyze the 'Answer'. Brainstorm and identify a closely related yet distinct 'Associated Concept' (e.g., a related historical event, a key figure, a geographic location, a unique attribute, its parent category, etc.).
Perform an exploratory search with the goal of finding the 'bridging information' that connects this 'Associated Concept' to your final 'Answer'.
From your search results, extract a unique, verifiable 'preceding fact'. This is a piece of information that, when searched, would logically lead a user to your final 'Answer'. This 'preceding fact' becomes the answer to Search #n-1.
2.2. Iterate Backwards
Now, treat this newly found 'preceding fact' as your new target.
From this point on, you can search for this new target directly to find the preceding piece of information that leads to it. This becomes the answer to the next search in the backward chain (Search #n-2).
2.3. Construct the Full Chain
Continuously repeat the iterative process from Step 2.2, using each new fact as the target for the next backward search, until you have constructed a complete logical chain of 'n' links.
The very first piece of information you uncover in this process (the one at the start of the chain) will become the initial clue for your question.

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

Here are three example questions:
Question 1: {example1}
Question 2: {example2}
Question 3: {example3}

Critical Rules:
1. Strictly Fact-Based: You must not create questions based on assumptions. The entire logical path to the solution must be grounded in the information you find through searching.

2. No Spoilers: The question must not contain any direct clues that reveal the answer or the intermediate steps.
3. Search is Mandatory: The question must be impossible to answer from general knowledge alone. It must necessitate the search process you have designed.
4. Adhere to Search Count: The number of searches required to solve the question must precisely match the specified 'search count'.
5. Unique Answer: The designed question must be deterministic, leading to a single, unambiguous final answer. The clues at each step must be precise enough to prevent a solver from reasonably arriving at a different, valid conclusion.

The answer I provided is: {answer}.
You need to create a question that requires {n} searches.
When you have enough information to construct a question, please first check whether the constructed question meets all requirements, especially whether the question is too simple. After checking that all conditions are met, you need to provide the final constructed question in your final response, placing the final question between `<question>` and `</question>` tags, for example: `<question>` ... `</question>`.

RAG Solver Prompt

Answer the given question based on the provided materials. You should first conduct very concise reasoning within 50 words, and then directly provide your answer without detailed illustrations after saying 'Answer:'.

Materials: {materials}

Question: {question}

Solver Prompt (Search-R1 / ZeroSearch / Qwen series / LLaMA 3.1)

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

Solver Prompt (R-Search)

You are a helpful assistant that can solve the given question step by step. For each step, start by explaining your thought process. If additional information is needed, provide a specific query enclosed in `<search>` and `</search>`. The system will return the top search results within `<observation>` and `</observation>`. You can perform multiple searches as needed. When you know the final answer, use `<original_evidence>` and `</original_evidence>` to provide all potentially relevant original information from the observations. Ensure the information is complete and preserves the original wording without modification. If no searches were conducted or observations were made, omit the evidence section. Finally, provide the final answer within `<answer>` and `</answer>` tags. {question}

LLM-as-a-judge Prompt

Please determine whether the model's answer is consistent with the reference answer:

Question: {question}
Model Answer: {prediction}
Reference Answer: {ground_truth}

Evaluation Criteria:
1. The model answer must accurately respond to the question and be consistent with the reference answer in meaning.
2. For numerical questions, the values must be equal or very close.
3. For textual questions, the core meaning must be correct.
4. Differences in wording or language are allowed as long as the core answer is the same.
5. If the model answer includes the correct answer and does not contain conflicting information, it is also considered correct.

Please respond only with "Correct" or "Incorrect". Do not provide any additional explanation.

E USE OF LARGE LANGUAGE MODELS

Large Language Models were used to aid in the writing and polishing of this manuscript. Specifically, LLMs assisted with language refinement and improving clarity. All core content and intellectual contributions are solely the work of the authors.