SE-Search: Self-Evolving Search Agent via Memory and Dense Reward

Anonymous ACL submission

Abstract

Retrieval augmented generation (RAG) reduces hallucinations and factual errors in large language models (LLMs) by conditioning generation on retrieved external knowledge. Recent search agents further cast RAG as an autonomous, multi-turn information-seeking process. However, existing methods often accumulate irrelevant or noisy documents and rely on sparse reinforcement learning signals. We propose **S**elf-**E**volving **Search**, a Self-Evolving Search agent that improves online search behavior through three components, memory purification, atomic query training, and dense rewards. SE-Search follows a *Think-Search-Memorize* strategy that retains salient evidence while filtering irrelevant content. Atomic query training promotes shorter and more diverse queries, improving evidence acquisition. Dense rewards provide fine-grained feedback that speeds training. Experiments on single-hop and multi-hop question answering benchmarks show that `SE-Search-3B` outperforms strong baselines, yielding a 10.8 point absolute improvement and a 33.8% relative gain over Search-R1.[1]

1 Introduction

Searching for information (Case and Given, 2016) is a common activity in daily life. Search engines such as Google and Bing return ranked web pages for a user query, enabling fast access to relevant content. As queries become more complex and users expect higher precision, traditional information retrieval techniques (Kobayashi and Takeda, 2000) may fail to capture subtle intent and return results that do not match the user's context.

Large language models (LLMs) (Zhao et al., 2023) have shown strong capabilities in language understanding, reasoning, and information integration. However, they do not reliably access up-to-date external knowledge and may produce factual errors. Retrieval-augmented generation (RAG) (Lewis et al., 2020) reduces these problems by conditioning generation on retrieved external passages. Yet fixed RAG pipelines limit an LLM's ability to decide when to search and what to search for, which reduces flexibility when interacting with real-world applications.

Recent work (Li et al., 2025a) presents LLM-based agents (Guo et al., 2024) that interpret user intent, plan search strategies, conduct multi-turn search actions, and accumulate retrieved evidence. These systems combine the strengths of LLMs and search engines and are often referred to as search agents. Representative methods such as Search-R1 (Jin et al., 2025) use reinforcement learning (RL) (Kaelbling et al., 1996; Shao et al., 2024) with rule-based rewards to train agents to interact with a search tool over a large corpus.

Despite promising results, existing search agent methods face three key challenges. **Noisy search results:** Agents typically retrieve top-K documents after issuing a query, and many are irrelevant or noisy, which provides limited support for answering user questions. MAIN-RAG (Chang et al., 2025) reduces noise through multi-LLM filtering and scoring, but it operates in a training-free RAG setting. **Limited search diversity and underexplored search frequency:** Prior methods (Shi et al., 2025) often generate similar queries across search steps, which limits exploration of diverse and informative evidence. Although O^2-Searcher (Mei et al., 2025) introduces a query diversity reward based on embedding similarity, the additional encoding step reduces efficiency. **Sparse evolutionary feedback signals:** Search-R1 provides supervision at the final answer level and does not explicitly reward query formulation, formatting, or appropriate search frequency. As a result, agents may produce overly long queries regardless of question complexity, which limits effectiveness on multi-hop questions. TooRL (Qian et al., 2025)

[1] We will make the code and model weights publicly available upon acceptance.

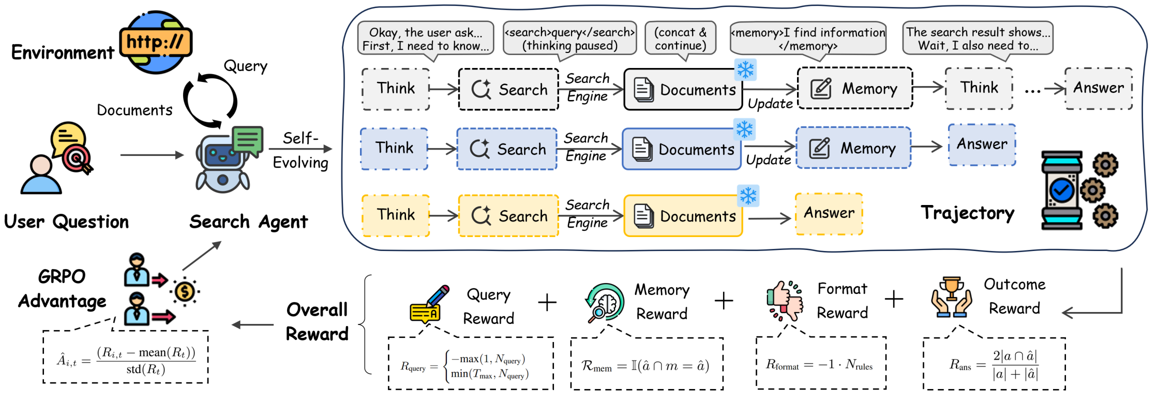

Figure 1: Training scheme of SE-Search. For each question, the search agent generates diverse trajectories comprising the steps *think*, *search*, *memorize*, and *answer*. These trajectories are optimized using the GRPO algorithm (Shao et al., 2024) and four carefully designed rewards: Query, Format, Memory, and Outcome.

proposes richer rewards, but it targets general tool use rather than search-specific behavior.

Inspired by Darwin's theory of evolution (Ruse, 1975; ang Gao et al.), which emphasizes adaptation to changing environments, we focus on how reward and feedback signals guide agents on complex real-world queries. These signals do not provide facts directly, but they shape how an agent discovers, processes, and uses external information. We therefore propose `SE-Search`, a self-evolving search agent that improves an LLM's autonomous search behavior. SE-Search adopts a *memorize-after-search* paradigm and extracts useful evidence with a **Memory Purification** template to reduce noisy retrievals. Unlike DeepAgent (Li et al., 2025d), which relies on auxiliary LLMs for memory management, SE-Search uses the agent's self-memory. To promote diverse search behaviors and appropriate search frequency, we introduce an **Atomic Query** strategy motivated by Atom-Searcher (Deng et al., 2025), which guides the agent to generate multiple distinct atomic queries and perform multi-step searches. Finally, we design **Dense Rewards** composed of four components, Query, Memory, Outcome, and Format, to provide fine-grained RL feedback, improve behavioral discipline, and stabilize training.

We summarize the contributions as follows.

- We propose SE-Search, a self-evolving search agent that improves adaptability to complex, real-world questions.
- We steer the agent's evolution by introducing three mechanisms: Memory Purification, Atomic Query, and Dense Rewards.
- We demonstrate the effectiveness and generalizability of SE-Search on seven diverse and challenging QA benchmarks.

2 Problem Formulation

We model question answering task as

$$a = \pi_\theta(Q) \tag{1}$$

where Q is the user question, a is the generated answer, and π_θ denotes the LLM.

In retrieval augmented generation, the system retrieves knowledge k from an external corpus $\mathcal{D}$ using a retriever $\mathcal{R}(\cdot)$ and then generates the answer conditioned on Q and k.

$$a = \pi_\theta(Q \mid k), \quad k = \mathcal{R}(Q \mid \mathcal{D}) \tag{2}$$

However, this fixed workflow is often insufficient for complex knowledge discovery tasks that require iterative reasoning and adaptive retrieval. Recent advances in search agents support a more dynamic paradigm in which retrieval is interleaved with reasoning, and the model output y may include intermediate reasoning steps r and the final answer a.

$$a = \pi_\theta\left(Q \mid \{r_t \otimes k_t\}_{t=1}^{T}\right), k_t = \mathcal{R}(q_t \mid \mathcal{D}), q_t \in r_t \tag{3}$$

where q_t denotes subqueries generated during reasoning to retrieve intermediate knowledge k_t, and the operator $\otimes$ indicates that retrieved knowledge is integrated into the reasoning trajectory. Ignoring the intermediate trajectory, we employ supervised fine-tuning (SFT) to optimize only the final answer using the next-token prediction loss, which is defined as follows.

$$\max_\theta \mathbb{E}\big[\log \pi_\theta(\hat{a})\big] \tag{4}$$

where $\mathbb{E}$ denotes expectation. This objective maximizes the expected log likelihood that π_θ assigns to the correct answer $\hat{a}$.

3 Approach

In this section, we present SE-Search, a self-evolving search agent that improves autonomous search behavior. We first formalize the search agent setting and its optimization objective. We then describe three components: memory purification, atomic query generation, and dense rewards.

3.1 Self-Evolving Search Agent

Figure 1 illustrates the optimization scheme of SE-Search. Given a user question, the agent interacts with a web environment through a search tool to generate multiple trajectories τ. Each trajectory is represented as $\tau = (\tau_1, \tau_2, \ldots, \tau_T)$, where step t is denoted as $\tau_t = (a_t, c_t)$. The action a_t is selected from {<think>, <search>, <documents>, <memory>, <answer>}, and the content c_t corresponds to a reasoning thought, a search query, retrieved passages, a distilled summary, or a final answer. The agent aims to gather relevant evidence across steps and produce an accurate final answer.

Formally, we optimize the following objective:

$$\max_{\theta} \mathbb{E}\big[\log \pi_\theta(\hat{a}) + \alpha \cdot \mathcal{C}(\cup_{t=1}^{T} k_t, \hat{K})\big] \tag{5}$$

where $\mathcal{C}(\cdot, \cdot)$ measures the coverage of retrieved documents relative to the expected knowledge $\hat{K}$, T denotes the number of retrieval steps, and $\alpha \in (0, 1)$ is a weighting factor. Search-R1 (Jin et al., 2025) shows that loss masking can exclude retrieved tokens from gradient optimization. Following this insight, we approximate retrieval using the search queries and memory contents and rewrite the objective accordingly.

$$\max_{\theta} \mathbb{E}\big[\log \pi_\theta(\hat{a}) + \alpha \cdot \sum_t \log \pi_\theta(\hat{m}_t) + \gamma \cdot \sum_t \log \pi_\theta(\hat{q}_t)\big] \tag{6}$$

where γ is an additional weighting factor. This objective accounts for three aspects: search queries, memory, and final answers. Due to the lack of ground truth trajectory, supervised fine-tuning is not feasible. We adopt a post-training reinforcement learning framework and use a rule-based reward function to optimize trajectory generation.

3.2 Memory Purification.

Prior methods forward-retrieve documents to the LLM without filtering, which forces the model to reason over irrelevant and noisy content. Inspired by Memory-R1 (Yan et al., 2025), we introduce

You are a capable reasoning assistant, able to perform multiple search calls and memory purification to answer questions. You must reason through the available information using <think> and </think>. If you lack knowledge, you can call a search engine using <search> query </search> and it will return the top three results between <documents> and </documents>. After each search, extract useful information from these documents and supplement or revise your memory between <memory> and </memory>. You may send multiple search requests if needed. Do not repeat search queries. Once you have sufficient information, provide a concise final answer using <answer> and </answer>. For example, <answer> Donald Trump </answer>. Question {**question**}

Figure 2: Prompt template for SE-Search.

memory purification to filter and consolidate relevant facts from retrieval documents and store key information using the special token <memory>. Figure 2 shows the employed prompt require LLM to supplement or revise previous memory knowledge by incorporating retrieved knowledge k into latest memory m_t. The update process is modeled as:

$$m_t = \pi_\theta(m_{t-1} \mid k_t) \tag{7}$$

This strategy filters and integrates only useful information, thereby maintaining coherent and evolving knowledge. To explicitly encourage selective extraction from noisy retrieved documents, we define a memory reward. Following AutoRefine (Shi et al., 2025), the memory reward is measured using Cover Exact Match (CEM) between the memory contents m and the correct answers $\hat{a}$.

$$\mathcal{R}_{\text{mem}} = \mathbb{I}(\hat{a} \cap m = \hat{a}) \tag{8}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

3.3 Atomic Query

In Algorithm 1, we propose an atomic query counting method that constrains query length and enforces diversity across queries. Based on the number of valid atomic queries N_{query} in a trajectory, we define a query reward that guides when and how the agent invokes the search tool.

$$R_{\text{query}} = \begin{cases} -\max(1, N_{\text{query}}) & \text{if the answer is correct} \\ \min(T_{\max}, N_{\text{query}}) & \text{if the answer is incorrect} \end{cases} \tag{9}$$

where $T_{\max}$ is the maximum allowed number of search action turns, and N_{query} denotes the number of valid search queries in the trajectory. This reward serves two purposes. It encourages the agent to produce diverse queries that improve evidence coverage. It also discourages unnecessary search when the final answer is correct and encourages more search when the final answer is incorrect.

3.4 Dense Rewards

Apart from the query and memory rewards, the dense reward includes two additional components: an outcome-based reward that directly measures the correctness of the model's final answer and a format-based reward that encourages adherence to the prescribed reasoning structure. Rather than using a binary exact match signal, we treat the predicted answer and the ground-truth answer as sets and compute the F1 score between the model's predicted answer a and the ground-truth answer $\hat{a}$.

$$R_{\text{ans}} = \frac{2|a \cap \hat{a}|}{|a| + |\hat{a}|} \tag{10}$$

We define the format reward through three rules that penalize unreasonable trajectories to prevent mode collapse and degenerate behavior. These rules include cases where the trajectory exceeds the predefined maximum number of search action turns T_{max}, contains invalid actions, or includes unmatched or incorrectly ordered special tokens. Let N_{rules} denote the total number of format violations in the trajectory, and we define

$$R_{\text{format}} = -1 \cdot N_{\text{rules}} \tag{11}$$

Combining all dense reward components, we obtain the overall dense reward

$$R_{\text{Dense}} = R_{\text{ans}} + \alpha \cdot R_{\text{mem}} + \gamma \cdot \mu \cdot R_{\text{query}} + \gamma \cdot R_{\text{format}} \tag{12}$$

where R_{mem} and R_{query} denote the memory and query rewards defined earlier, and α and γ are weighting coefficients that balance the reward terms. To improve RL stability, we gradually reduce the influence of the query reward during training by multiplying it with a time-dependent decay factor. The cosine decay schedule is defined as

$$\mu = \begin{cases} \frac{1}{2}(cos(\frac{t_{\text{iter}}}{t_{\text{decay}}}\pi) + 1), & t_{\text{iter}} \leq t_{\text{decay}} \\ 0, & t_{\text{iter}} > t_{\text{decay}} \end{cases} \tag{13}$$

where t_{iter} is the current training iteration and t_{decay} is the number of decay steps.

3.5 Agentic Reinforcement Learning

To avoid a separate value estimator, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as our agentic reinforcement learning framework. For a question Q from the dataset D, the policy model π_θ samples an interleaved sequence $y_0, k_0, y_1, k_1, \ldots, y_T, k_T$. Following Search-R1 (Jin et al., 2025), external retrieval tokens k are masked during loss computation, so the objective depends only on the model outputs y.

Algorithm 1: Atomic Query Counting

Input: Search query set $\mathcal{Q}$ from a trajectory, initialize valid query set $\mathcal{Q}_v \leftarrow \varnothing$, length bounds $L_{\min}, L_{\max}$, similarity threshold T.

for $q_i \in \mathcal{Q}$ **do**
  if $L_{\min} \leq |q_i| \leq L_{\max}$ **then**
    for $q_j \in \mathcal{Q}_v$ **do**
      Compute matching segment lengths ℓ_k between q_i and q_j via SequenceMatch;

$$\text{ratio}(q_i, q_j) = \frac{2\sum_k \ell_k}{|q_i| + |q_j|} \tag{14}$$

      if *ratio*$(q_i, q_j) \leq T$ **then**
        $\mathcal{Q}_v = \mathcal{Q}_v \cup \{q_i\}$;
      end if
    end for
  end if
end for
Return $N_{query} \leftarrow |\mathcal{Q}_v|$;

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{Q\sim\mathcal{D}, y_i\sim\pi_\theta}\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min(r_{i,t}\hat{A}_{i,t}, \text{clip}\left(r_{i,t}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}) - \beta\mathcal{KL}\left[\pi_\theta \,\|\, \pi_{\text{ref}}\right]\Big] \tag{15}$$

where π_θ denotes the current actor and π_{ref} denotes a fixed reference policy. The probability ratio is computed using the previous policy $\pi_{\theta_{\text{old}}}$.

$$r_{i,t} = \frac{\pi_\theta(y_{i,t} \mid Q, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid Q, y_{i,<t})} \tag{16}$$

For each question, we use a group of G sampled outputs and normalize their rewards to estimate advantages.

$$\hat{A}_{i,t} = \frac{(R_{i,t} - \text{mean}(R_t))}{\text{std}(R_t)} \tag{17}$$

where R is the overall dense reward in Eq. 12.

4 Experiments

4.1 Experiment Settings

Benchmarks and Datasets. We evaluate SE-Search on seven question answering benchmarks that require single-hop or multi-hop retrieval. The

Methods	Single-Hop QA (EM) NQ	TriviaQA	PopQA	Avg.	Multi-Hop QA (EM) HotpotQA	2Wiki	Musique	Bamboogle	Avg.	QA (EM) Avg.
w/o Retrieval										
Direct Generation	0.106	0.288	0.108	0.167	0.149	0.244	0.020	0.024	0.109	0.134
SFT	0.249	0.292	0.104	0.215	0.186	0.248	0.044	0.112	0.148	0.176
R1-Instruct (Guo et al., 2025)	0.210	0.449	0.171	0.277	0.208	0.275	0.060	0.192	0.184	0.224
R1-Base (Guo et al., 2025)	0.226	0.455	0.173	0.285	0.201	0.268	0.055	0.224	0.187	0.229
Workflow w/ Retrieval										
Naive RAG (Lewis et al., 2020)	0.348	0.544	0.387	0.426	0.255	0.226	0.047	0.080	0.152	0.270
IRCoT (Trivedi et al., 2022a)	0.111	0.312	0.200	0.208	0.164	0.171	0.067	0.240	0.161	0.181
Agent w/ Retrieval										
Search-o1 (Li et al., 2025c)	0.238	0.472	0.262	0.324	0.221	0.218	0.054	0.320	0.203	0.255
Search-R1-Instruct (Jin et al., 2025)	0.397	0.565	0.391	0.451	0.331	0.310	0.124	0.232	0.249	0.336
Search-R1-Base (Jin et al., 2025)	0.421	0.583	0.413	0.472	0.297	0.274	0.066	0.128	0.191	0.312
ReSearch-Instruct (Chen et al., 2025)	0.365	0.571	0.395	0.444	0.351	0.272	0.095	0.266	0.246	0.331
ReSearch-Base (Chen et al., 2025)	0.427	0.597	0.430	0.485	0.305	0.272	0.074	0.128	0.195	0.319
ZeroSearch-Base (Sun et al., 2025)	0.430	0.616	0.414	0.487	0.338	0.346	0.130	0.139	0.238	0.345
StepSearch-Base (Wang et al., 2025)	-	-	-	-	0.329	0.339	0.181	0.328	0.294	-
O^2-Searcher (Mei et al., 2025)	0.444	0.597	0.429	0.490	0.388	0.374	0.160	0.344	0.317	0.391
AutoRefine-Instruct (Shi et al., 2025)	0.436	0.597	0.447	0.493	0.404	0.380	0.169	0.336	0.322	0.396
AutoRefine-Base (Shi et al., 2025)	0.467	0.620	0.450	**0.512**	0.405	0.393	0.157	0.344	0.325	0.405
InForage (Qian and Liu, 2025)	0.421	0.597	**0.452**	0.490	0.409	**0.428**	0.172	0.360	0.342	0.405
CriticSearch (Zhang et al., 2025)	-	-	-	-	0.414	0.409	0.180	0.368	0.343	-
SE-Search-3B (Ours)	**0.475**	**0.624**	0.423	0.507	**0.450**	0.361	**0.183**	**0.424**	**0.355**	**0.420**

Table 1: Accuracy comparison of SE-Search-3B against baseline methods using Qwen2.5-3B (Qwen et al., 2025) across multiple QA benchmarks. **Bold** indicates best results, higher values denote better performance.

single-hop datasets include Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2022). The multi-hop datasets include HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020), Musique (Trivedi et al., 2022b), and Bamboogle (Press et al., 2022). We use exact match (EM) for all datasets. Following Search-R1 (Jin et al., 2025), we train SE-Search on a combined training set of NQ and HotpotQA.

Baselines. We compare SE-Search with three classes of methods. (1) Methods without retrieval include direct LLM generation, supervised fine-tuning (SFT), and R1-style training (Guo et al., 2025); (2) Workflows with retrieval include naive RAG that retrieves only from the input question and IRCoT (Trivedi et al., 2022a), which interleaves retrieval with chain-of-thought; (3) Agents with retrieval include retrieval-augmented methods such as Search-o1 (Li et al., 2025c), Search-R1 (Jin et al., 2025), ReSearch (Chen et al., 2025), ZeroSearch (Sun et al., 2025), StepSearch (Wang et al., 2025), O^2-Searcher (Mei et al., 2025), InForage (Qian and Liu, 2025), CriticSearch (Zhang et al., 2025), and AutoRefine (Shi et al., 2025).

Implementation Details. To emulate a standard search setting, we use the external corpus (Karpukhin et al., 2020) used by Search-R1 and adopt E5-base-v2 (Wang et al., 2022) as the retriever. By default, the retriever returns the top three documents per query, and the backbone LLM is Qwen2.5-3B (Qwen et al., 2025). The query length bounds $L_{\min}$ and $L_{\max}$ are set to 20 and 120 characters. The similarity threshold is $T = 0.3$. The maximum number of search turns is $T_{\max} = 5$, and the decay steps t_{decay} are set to 150. The reward weights are $\alpha = 0.1$ and $\gamma = 0.01$.

4.2 Main Performance

Table 1 presents the main experimental results comparing SE-Search with baseline methods. The Avg. column reports the average accuracy. From these results, we make the following key observations.

SE-Search outperforms other methods. Under the same retriever, corpus, training data, and Qwen2.5-3B setting, SE-Search consistently outperforms Search-R1 and recent methods, including InForage and AutoRefine, across seven benchmarks covering both single-hop and multi-hop QA. SE-Search achieves an average EM accuracy of 0.420, demonstrating that the proposed memory purification mechanism and dense reward design effectively enhance the self-evolution capability of search agents during information seeking.

SE-Search shows particularly strong gains on multi-hop QA benchmarks. The performance improvements are more evident on complex multi-hop QA tasks. For instance, compared with AutoRefine, SE-Search improves HotpotQA by 4.5 percentage points, corresponding to an 11.1% relative gain, and improves Bamboogle by 8 percentage points,

Method	Single-Hop QA			Multi-Hop QA				
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	Avg.
Search-R1	0.424	0.630	0.471	0.346	0.296	0.070	0.169	0.344
Search-R1	0.426	0.610	0.436	0.377	0.315	0.120	0.260	0.363
w/ MP	+0.002	−0.020	−0.035	+0.031	+0.019	+0.050	+0.091	+0.019
	↑ 0.47%	↓ 3.17%	↓ 7.43%	↑ 8.96%	↑ 6.42%	↑ 71.43%	↑ 53.85%	↑ 5.52%
Search-R1	**0.442**	0.609	0.427	0.381	0.365	0.163	**0.371**	0.394
w/ MP, AQ	+0.016	−0.001	−0.009	+0.004	+0.050	+0.043	+0.111	+0.031
	↑ 3.76%	↓ 0.16%	↓ 2.06%	↑ 1.06%	↑ 15.87%	↑ 35.83%	↑ 42.69%	↑ 8.54%
Search-R1	0.429	**0.661**	**0.467**	**0.436**	**0.374**	**0.177**	0.328	**0.410**
w/ MP, AQ, DR	−0.013	+0.052	+0.040	+0.055	+0.009	+0.014	−0.043	+0.016
(SE-Search)	↓ 2.94%	↑ 8.55%	↑ 9.37%	↑ 14.44%	↑ 2.47%	↑ 8.59%	↓ 11.59%	↑ 4.06%

Table 2: Ablation results showing the effect of individual SE-Search components. The first row reports the baseline method. Subsequent rows show performance after adding Memory Purification, Atomic Query, and Dense Rewards.

corresponding to a 23.2% relative gain. These improvements can be attributed to two key design choices. Atomic queries decompose the original question into multiple subqueries, enabling more effective retrieval across steps, while memory purification reduces noise accumulation during multi-step search and reasoning.

4.3 Ablation Studies

We conduct an ablation study to evaluate the contributions of SE-Search's three key components. For a controlled comparison, all models use the same retriever and corpus, are trained for 200 steps, and are evaluated on 500 randomly selected samples from each benchmark. Table 2 reports accuracy on seven benchmarks and also reflects how each component changes search behavior through the final performance. We compare four configurations. (1) Search-R1 (baseline); (2) Search-R1 with MP, augmented with Memory Purification and the memory reward $\mathcal{R}_{\text{mem}}$; (3) Search-R1 with MP and AP, adding Atomic Query on top of Memory Purification; (4) Search-R1 with MP, AP, and DR (SE-Search), the full configuration that further includes Dense Reward $\mathcal{R}_{\text{dense}}$;

Each component provides consistent performance gains. Memory Purification encourages multi-turn search and improves Musique by 5 points and Bamboogle by 9.1 points, corresponding to relative increases of 71.43% and 53.85%. Atomic Query further strengthens multi-turn search and yields relative improvements of 15.87% on 2Wiki, 35.83% on Musique, and 42.69% on Bamboogle. Overall, the full SE-Search configuration improves performance across seven benchmarks and attains the highest average accuracy.

4.4 Generalization and Scalability

Search agent follows consistent scaling trend. We conduct a generalization study across backbone models of different sizes to examine the scaling law (Kaplan et al., 2020). We keep the retriever, corpus, and evaluation protocol unchanged and vary only the backbone scale. Table 3 reports EM and F1 scores using different backbone LLMs, including Qwen2.5-3B, 7B, and 14B, where EM measures exactness, and F1 captures partial overlap. The results are consistent with the scaling law, showing that SE-Search achieves higher average EM and F1 and improves performance on most datasets as the scale of the backbone model increases.

4.5 Additional Analysis

Search frequency decreases while accuracy improves Figure 3 illustrates the evolution of mean accuracy and the mean number of search calls on the training data and evaluation benchmarks. The accuracy of SE-Search increases from 0.36 to approximately 0.41, corresponding to an improvement of about 14%, while the average number of search calls decreases from 1.53 to 1.32, corresponding to a reduction of about 14%. We observe that Memory Purification tends to recall a larger set of documents to extract useful information during the early stages of RL. In contrast, Atomic Query generates diverse and high-quality search queries that retrieve fewer but more relevant documents from search engines in a more efficient manner.

SE-Search calls the search engine more often on complex questions. Figure 3(b) compares search frequency across training steps and question types, where multi-hop questions represent higher compositional complexity than single-hop ones. Compared with Search-R1, which relies on a fixed sin-

Methods	Single-Hop QA			Multi-Hop QA				
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	Avg.
SE-Search-3B (EM)	0.475	0.624	0.423	0.450	0.361	0.183	0.424	0.420
SE-Search-7B (EM)	0.467	0.629	0.461	0.438	0.355	0.195	0.488	0.433
SE-Search-14B (EM)	0.508	0.681	0.473	0.478	0.404	0.213	0.544	0.472
SE-Search-3B (F1)	0.567	0.705	0.468	0.573	0.431	0.271	0.536	0.507
SE-Search-7B (F1)	0.556	0.705	0.504	0.559	0.422	0.290	0.601	0.519
SE-Search-14B (F1)	0.598	0.751	0.514	0.602	0.478	0.294	0.631	0.553

Table 3: Generalization results showing the scaling trend across backbone LLMs of sizes 3B, 7B, and 14B.

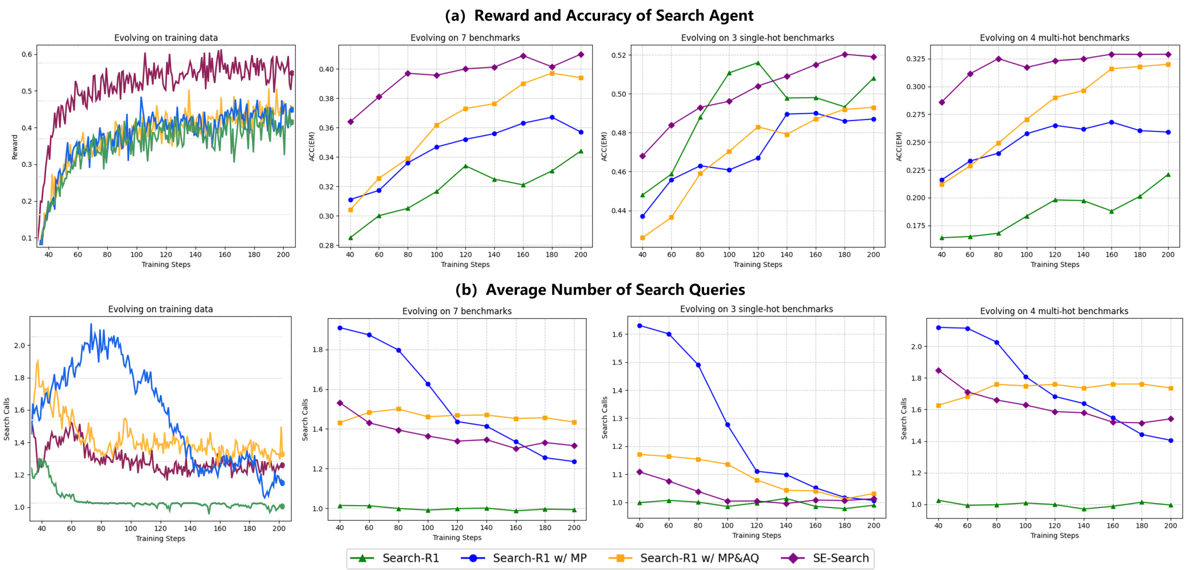

Figure 3: SE-Search's evolution of (a) EM accuracy and (b) search calls on training set and benchmarks.

gle search call, SE-Search issues an average of approximately 1.54 search calls on multi-hop questions while remaining close to 1.0 on single-hop benchmarks. This pattern suggests that SE-Search adjusts tool use to question complexity, allocating more retrieval to gather evidence for multi-hop reasoning while avoiding unnecessary calls on simpler questions.

SE-Search produces more atomic and diverse queries. Prior work emphasizes higher search frequency while largely overlooking query quality. Figure 4(a) reports the average character length and variance of search queries for three methods, including SE-Search at approximately 50, AutoRefine at approximately 65, and Search-R1 at approximately 90. These results suggest that shorter and more focused queries are more effective at retrieving relevant and high-quality documents. Figure 4(b) further compares query diversity using the similarity ratio defined in Eq. 14. SE-Search achieves a similarity ratio of about 0.5, whereas AutoRefine reaches a higher value of about 0.6. Higher query similarity indicates greater redundancy in retrieved results, which may introduce repetitive or biased information and degrade the contextual quality provided to the LLM.

Memory Purification stores key information for answers. We evaluate the information stored in memory using CEM, which measures whether the memory contains evidence related to the ground-truth answer. Figure 4(c) shows that the mean CEM of the memory reaches approximately 0.65 at the final stage of training. This result demonstrates that the agent successfully extracts useful information from retrieved documents and stores key facts in memory, rather than retaining noisy or irrelevant details. Consistent with the training objective, the blue line in Figure 3(b) indicates that the agent issues more search actions during early training to purify memory with important information and filter noise, which leads to higher memory rewards and more reliable memory contents.

Dense Rewards help convergence. We further examine the effect of dense rewards, which introduce a format reward and replace the EM-based outcome reward with an F1-based reward. Figure 4(c)

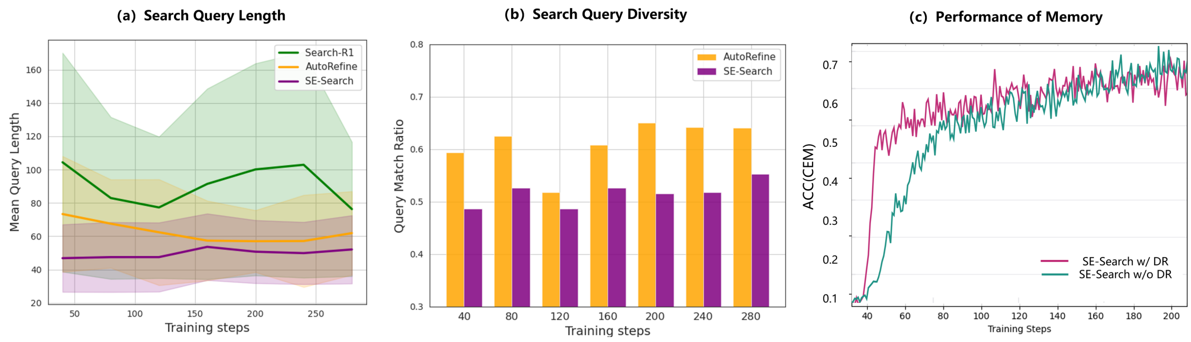

Figure 4: Statistics for (a) search query lengths, (b) search query diversity, (c) accuracy of memory contents.

shows that dense rewards improve the accuracy of memory content during the early stages of training, indicating earlier learning of how to record answer-relevant evidence. In particular, the format reward discourages invalid outputs and reduces unproductive exploration trajectories, thereby accelerating convergence with fewer RL exploration steps. Moreover, the F1-based outcome reward provides graded feedback even when the answer is not an exact match, which yields denser learning signals than EM and stabilizes training. Consistent with this, Figure 3(a) shows that SE-Search with dense rewards attains higher reward values than SE-Search without dense rewards, and this improvement is mainly attributed to the F1-based outcome reward.

5 Related Works

5.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) improves generative models by grounding outputs in external knowledge (Lewis et al., 2020). A typical RAG system consists of a retriever and a generator, and surveys summarize four common paradigms (Gao et al., 2023), including Sequential RAG, Branching RAG, Conditional RAG, and Loop RAG. Conventional RAG often uses a fixed retrieve then generate workflow and does not let the model decide when to search during reasoning. SE-Search instead targets autonomous search behavior with explicit memory, exploration, and behavioral constraints.

5.2 Search Agent

Search agents extend RAG by invoking a search tool within a multi-step reasoning process, as in Search-o1 (Li et al., 2025c). Related systems such as WebThinker (Li et al., 2025e) and WebDancer (Wu et al., 2025) also support autonomous retrieval during reasoning. Other methods focus on training strategies that improve search behavior, including Search-R1 (Jin et al., 2025), ReSearch (Chen et al., 2025), AutoRefine (Shi et al., 2025), and WebSailor (Li et al., 2025b). A common limitation is that retrieved context often contains noise that accumulates across steps and weakens later reasoning. SE-Search addresses this issue with Think-Search-Memorize and Atomic Query to improve how evidence is selected and retained.

5.3 Reinforcement Learning

Reinforcement learning (RL) provides a general framework for improving LLM behavior. Common approaches include proximal policy optimization (PPO) (Schulman et al., 2017), direct preference optimization (DPO) (Rafailov et al., 2023), and group relative policy optimization (GRPO) (Shao et al., 2024). Extensions such as dynamic sampling policy optimization (DAPO) (Yu et al., 2025) and group sequence policy optimization (GSPO) (Zheng et al., 2025) further improve training stability and efficiency. GRPO uses group-based normalization and avoids a separate critic. In this work, we design a fine-grained reward function that adapts GRPO to search agent optimization.

6 Conclusions

In this paper, we propose SE-Search, a self-evolving search agent that enhances the self-memory, self-exploration, and self-discipline capabilities of LLMs through three evolution directions, including memory purification, atomic query, and dense rewards. SE-Search addresses limitations in prior work that overlook the impact of irrelevant and noisy documents and that lack fine-grained feedback during RL. Extensive experiments on both single-hop and multi-hop question answering benchmarks demonstrate that SE-Search consistently outperforms existing methods.

7 Limitations

Although SE-Search outperforms existing methods on single-hop and multi-hop QA benchmarks, it has several limitations. First, SE-Search does not use live web search; the retrieval corpus is static and therefore does not contain real-time information from the internet. Second, highly complex tasks, such as BrowseComp (Wei et al., 2025), remain underexplored. Third, the method requires manual selection and tuning of several hyperparameters for dense rewards. Finally, SE-Search supports only a single tool and lacks additional capabilities (e.g., page browsing and code execution).

References

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. A survey of self-evolving agents: On path to artificial super intelligence, 2025. *URL https://arxiv. org/abs/2507.21046*.

Donald O Case and Lisa M Given. 2016. Looking for information: A survey of research on information seeking, needs, and behavior.

Chia-Yuan Chang, Zhimeng Jiang, Vineeth Rakesh, Menghai Pan, Chin-Chia Michael Yeh, Guanchu Wang, Mingzhi Hu, Zhichao Xu, Yan Zheng, Mahashweta Das, and 1 others. 2025. Main-rag: Multi-agent filtering retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2607–2622.

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

Yong Deng, Guoqing Wang, Zhenzhe Ying, Xiaofeng Wu, Jinzhen Lin, Wenwen Xiong, Yuqin Dai, Shuo Yang, Zhanwei Zhang, Qiwen Wang, and 1 others. 2025. Atom-searcher: Enhancing agentic deep research via fine-grained atomic thought reward. *arXiv preprint arXiv:2508.12800*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

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

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

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

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

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

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

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

Mei Kobayashi and Koichi Takeda. 2000. Information retrieval on the web. *ACM computing surveys (CSUR)*, 32(2):144–173.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

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

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, and 1 others. 2025b. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*.

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

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and 1 others. 2025d. Deepagent: A general reasoning agent with scalable toolsets. *arXiv preprint arXiv:2510.21618*.

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

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

Jianbiao Mei, Tao Hu, Daocheng Fu, Licheng Wen, Xuemeng Yang, Rong Wu, Pinlong Cai, Xinyu Cai, Xing Gao, Yu Yang, and 1 others. 2025. o^2-searcher: A searching-based agent model for open-domain open-ended question answering. *arXiv preprint arXiv:2505.16582*.

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

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

Hongjin Qian and Zheng Liu. 2025. Scent of knowledge: Optimizing search-enhanced reasoning with information foraging. *arXiv preprint arXiv:2505.09316*.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Michael Ruse. 1975. Charles darwin's theory of evolution: an analysis. *Journal of the History of Biology*, pages 219–241.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Yaorui Shi, Shihan Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. 2025. Search and refine during think: Autonomous retrieval-augmented reasoning of llms. *arXiv e-prints*, pages arXiv–2505.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

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

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

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

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. 2025. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, and 1 others. 2025. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Yuanheng Zhu, Qichao Zhang, Zijie Zhao, and Dongbin Zhao. 2025. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. *arXiv preprint arXiv:2511.12159*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

A Appendix

A.1 Training and Evaluation Datasets

Following Search-R1 and other RL-based methods, we merged NQ and HotpotQA to form the training dataset, comprising 169,615 samples. The evaluation set comprises 51,713 samples drawn from the test set of four benchmarks(NQ, TriviaQA, PopQA, Bamboogle) and the development sets of three benchmarks (HotpotQA, 2Wiki, and Musique).

A.2 GRPO Hyperparameters

We list the key hyperparameters used with the VeRL framework (Sheng et al., 2025). In Table 4, we adopt the same data, actor, and rollout configuration as Search-R1 and AutoRefine. For the dense reward design, the atomic query reward is computed for query lengths in the range from 10 to 120 and uses a similarity threshold of 0.3. The decay step is set to 150, which is half of the total training steps.

Table 4: Hyperparameters in RL.

Module	Hyper-parameter	Value
Data	Max Documents Length	512
	Total Training Steps	320
	Max Response Length	2048
	Retriever Topk	3
Actor	Training Batch Size	256
	Micro Training Batch Size	64
	Learning Rate	1×10^{-6}
	KL Coefficient β	0.001
	Clip Ratio ϵ	0.2
Rollout	Max Search Actions	5
	Group Size G	10
	Temperature	1.0
	Top P	0.95
Reward	L_{min}	10
	L_{max}	120
	Similarity Threshold T	0.3
	t_{decay}	150
	Reward weights α	0.1
	Reward weights γ	0.01

A.3 More Experiments Results

To provide a more comprehensive characterization of SE-Search, Figure 5 shows its training dynamics, plotted as average accuracy, across seven benchmarks: Natural Questions (NQ), TriviaQA, PopQA, Bamboogle, HotpotQA, 2Wiki, and Musique.

A.4 Case Studies

Table 5 lists the atomic queries derived from a complex user question, illustrating the self-planning and self-exploration of SE-Search. Table 6 and Table 7 show the memory contents and full trajectories generated by SE-Search for two randomly selected examples from Musique and HotpotQA. These results demonstrate SE-Search's self-memory and self-discipline.

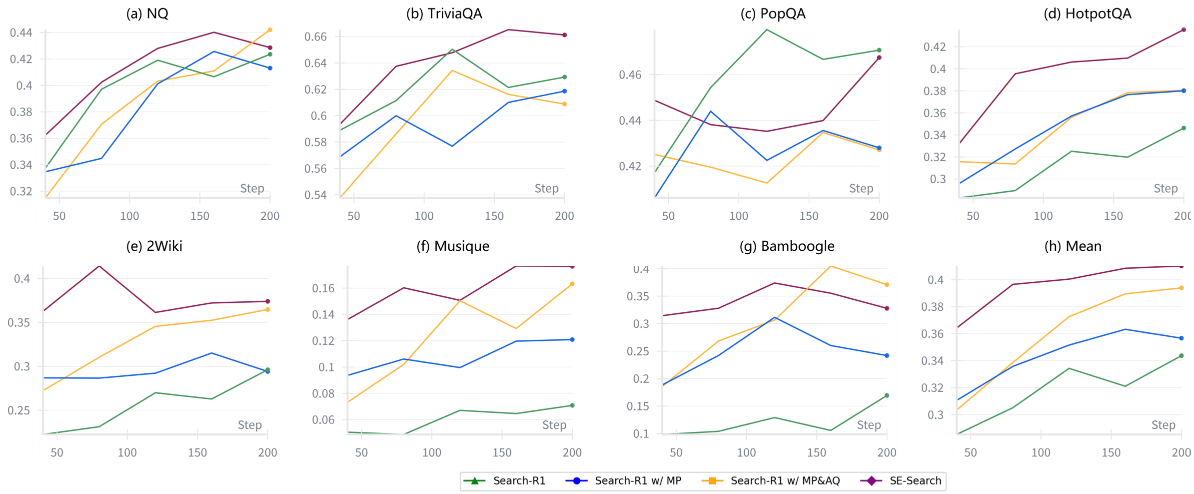

Figure 5: SE-Search's evolution of EM accuracy across seven benchmarks.

Table 5: The list of atomic search queries illustrates self-planning and self-exploration.

Question Which film has the director died earlier, Kati Patang or A Thief In The Dark?
Ground Truth A Thief In The Dark

SE-Search
1. <search> Director of Kati Patang </search>
<memory> ... the director of Kati Patang is Shakti Samanta. </memory>
2. <search> Director of A Thief In The Dark </search> <memory> ... the director of A Thief In The Dark is Donald W. Thompson. </memory>
<think> I need to determine which director died earlier, Shakti Samanta or Donald W. Thompson. </think>
3. <search> Death date of Shakti Samanta </search> <memory> ... Shakti Samanta died on 9 April 2009. </memory>
4. <search> Death date of Donald W. Thompson </search> <memory> ... Thompson died on February 19, 2011. </memory>
<answer> A Thief In The Dark </answer>.

Table 6: Case Studies illustrating self-memory and self-discipline. Key evidence is highlighted **Blue**.

Case 1 Who is the spouse of the actor who played Scarlett in Gone with the Wind?
Ground Truth Laurence Olivier

SE-Search
To answer the question "Who is the spouse of the actor who played Scarlett in Gone with the Wind?", I'll follow these steps.
1. <think>I need to identify the actor who played Scarlett in "Gone with the Wind". </think>
2. <search> Actor who played Scarlett in Gone with the Wind </search>
<documents>Doc 1(Title "Gone with the Wind (film)") **The leading roles are played by Vivien Leigh (Scarlett), Clark Gable (Rhett), Leslie Howard (Ashley), and Olivia de Havilland (Melanie).** ... Doc 3(Title "Vivien Leigh") Vivien Leigh **Vivien Leigh** (; born Vivian Mary Hartley, and styled as Lady Olivier after 1947; 5 November 1913–8 July 1967) was an English stage and film actress. **She won two Academy Awards for Best Actress, for her iconic performances as Scarlett O'Hara in ""Gone with the Wind"" (1939)** </documents>
3. <memory> The actor who played Scarlett in "Gone with the Wind" is Vivien Leigh.</memory>
4. <think>Now I need to find who Vivien Leigh is married to. </think>
5. <search> Vivien Leigh husband </search>
<documents> Doc 1(Title "Vivien Leigh") ..., **the public strongly identified Leigh with her second husband, Laurence Olivier**, who was her spouse from 1940 to 1960. Leigh and Olivier starred together in many stage productions, with Olivier often directing, and in three films. Her first husband Leigh Holman also spent considerable ... </documents>
6. <memory> Vivien Leigh is married to Laurence Olivier. </memory>
7. <answer> Laurence Olivier </answer>.

Table 7: Case Studies illustrating self-memory and self-discipline. Key evidence is highlighted **Blue**.

Case 2 How many ethnic minorities were looked at differently in the city where the headquarters of the only group larger than Vilaiyaadu Mankatha's record label is located?
Ground Truth two

SE-Search
To answer the question "How many ethnic minorities were looked at differently in the city where the headquarters of the only group larger than Vilaiyaadu Mankatha's record label is located?", let's follow these steps.
1. <think>First, I need to find out which group larger than Vilaiyaadu Mankatha's record label is located in the headquarters city. </think>
2. <search> Which group larger than Vilaiyaadu Mankatha's record label is located in the headquarters city? </search>
<documents> Doc 1(Title "Tharangini Records") Primarily established as the first recording and distributing music label in Kerala for Malayalam films... **Tharangini Records (currently Tharangni) is the music company** founded by noted Indian Playback singer K. J. Yesudas in 1980 at Thiruvananthapuram, Kerala. It became the first recording centre ...</documents>
3. <memory> ..., the group larger than Vilaiyaadu Mankatha's record label is Tharangini Records. </memory>
4. <search> What city is the headquarters of Tharangini Records? </search>
<documents> Doc 1(Title "Tharangini Records") Tharangini Records Tharangini Records (currently Tharangni) is the music company founded by noted Indian Playback singer K. J. Yesudas in 1980 ... **The office and studio were moved to Abhiramapuram, Chennai, Tamil Nadu licensed under Dr. K. J. Yesudas Audio and Visual Corporation Ltd.**) ...</documents>
5. <memory> ..., the headquarters of Tharangini Records is located in Chennai, Tamil Nadu. </memory>
6. <search> How many ethnic minorities were looked at differently in the city of Chennai, Tamil Nadu? </search>
<documents> Doc 1(Title "Demographics of Chennai") ... however, the population growth rate was slow. Hindus form the majority of Chennai's population **but the city also has substantial Muslim and Christian minorities.** </documents>
7. <memory> ..., the city of Chennai, Tamil Nadu has a substantial Muslim and Christian minority. </memory>
8. <answer> Two </answer>.