SimpleDeepSearcher: Deep Information Seeking via Web-Powered Reasoning Trajectory Synthesis

Shuang Sun[1]*, Huatong Song[1]*, Yuhao Wang[1], Ruiyang Ren[1], Jinhao Jiang[1], Junjie Zhang[1], Fei Bai[1], Jia Deng[1], Wayne Xin Zhao[1]†, Zheng Liu[2], Lei Fang[3]†, Zhongyuan Wang[2], Ji-Rong Wen[1]

[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Beijing Academy of Artificial Intelligence [3]DataCanvas Alaya NeW
{sunshuanguns, batmanfly}@gmail.com

Abstract

Retrieval-augmented generation (RAG) systems have advanced large language models (LLMs) in complex deep search scenarios requiring multi-step reasoning and iterative information retrieval. However, existing approaches face critical limitations that lack high-quality training trajectories or suffer from the distributional mismatches in simulated environments and prohibitive computational costs for real-world deployment. This paper introduces SimpleDeepSearcher, a lightweight yet effective framework that bridges this gap through strategic data engineering rather than complex training paradigms. Our approach synthesizes high-quality training data by simulating realistic user interactions in live web search environments, coupled with a multi-criteria curation strategy that optimizes the diversity and quality of input and output side. Experiments on five benchmarks across diverse domains demonstrate that SFT on only 871 curated samples yields significant improvements over RL-based baselines. Our work establishes SFT as a viable pathway by systematically addressing the data-scarce bottleneck, offering practical insights for efficient deep search systems. Our code and data are available at https://github.com/RUCAIBox/SimpleDeepSearcher.

1 Introduction

In recent years, retrieval-augmented generation (RAG) methods have significantly enhanced LLMs by incorporating external knowledge retrieval mechanisms (Lewis et al., 2020; Zhao et al., 2024; Gao et al., 2024). Recent advancements have extended these capabilities to complex *deep search* scenarios that demand multi-step reasoning with iterative information retrieval and synthesis (Alzubi et al., 2025). Traditional RAG systems typically treat retrieval as an external auxiliary module, following a fixed pipeline of "retrieval–reranking–reading" (Qi et al., 2020). In contrast, deep search scenarios require the model to internalize the abilities of "when to retrieve, how to retrieve, and how to reason based on retrieved content," in order to address more flexible and complex tasks.

To address the complex reasoning demands in deep search scenarios, early research explored prompt-based strategies that guide models to decompose questions, generate queries, and retrieve information iteratively (Jiang et al., 2024; Teng et al., 2025; Li et al., 2025a). Other studies have attempted to improve model performance through supervised fine-tuning (SFT) (Wang et al., 2025); however, there is currently a lack of high-quality trajectory data of reasoning and search interactions for training (Jin et al., 2025). To further enhance the model's autonomous search capabilities, Reinforcement Learning (RL) (Sutton et al., 1999) is considered as a promising solution to train models through real-time interaction with the environment (Nakano et al., 2021; Song et al., 2025; Jin et al., 2025; Zheng et al., 2025). However, most RL-based approaches operate within artificial environments using static document corpora, creating a distributional mismatch with real-world web dynamics. Moreover, the inherent computational intensity of RL training escalates exponentially when interfacing with live search APIs (Sun et al., 2025).

Given the overhead and complexity of RL-based training, we hypothesize that SFT remains a viable pathway for building efficient deep search systems. While SFT offers a streamlined training process, it faces the critical challenge of lacking high-quality training data in deep search scenarios. On the one hand, existing QA datasets often lack the diversity and complexity of questions and search-oriented purposes on the Web, which are essential for deep search training. On the other hand, traditional answer annotations omit the crit-

* Equal contributions.
† Corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 13705–13720
November 4-9, 2025

ical reasoning traces (search operations, evidence synthesis, and efficient decision paths) required for teaching search-integrated reasoning strategies.

In this paper, we propose SimpleDeepSearcher, an efficient search-with-think framework that utilizes strategic data engineering rather than complex training paradigms. Our core methodology centers on a three-fold process for constructing high-quality training data. First, we develop a data synthesis framework grounded in real web search environments, simulating realistic user search behaviors to generate multi-turn reasoning trajectories. Second, we propose a diversity-aware query sampling strategy to optimize domain coverage, semantic complexity, and knowledge unit density. Moreover, we adopt a four-dimensional response curation that enforces format standardization, reasoning efficiency, question difficulty, and search effectiveness. By systematically addressing both query and response-side quality through automated pipelines, SimpleDeepSearcher can obtain high-quality supervised signals based on real web search for complex reasoning to facilitate the SFT process.

Experimental results show that our SFT method significantly boosts model performance on five representative benchmarks with only 871 high-quality training samples. Compared to prompt-based methods, SimpleDeepSearcher achieves a 48.3% improvement, and compared to RL-based RAG methods achieves a 24.9% improvement. This demonstrates that our framework effectively balances performance and efficiency, providing a simple yet powerful approach to enhancing deep search capabilities. Furthermore, our framework is highly extensible that can be combined with other types of training data, the framework is also applicable to RL-based training.

Our main contributions are as follows:

• We propose a real web-based data synthesis framework that simulates realistic user search behaviors, generating multi-turn reasoning and search trajectories.

• We design a multi-criteria data curation strategy that jointly optimizes both input question selection and output response filtering through orthogonal filtering dimensions.

• Experimental results demonstrate that SFT on only 871 samples enables SimpleDeepSearcher to outperform strong baselines (especially RL-based baselines) on both in-domain and out-of-domain benchmarks.

2 Method

In this section, we propose SimpleDeepSearcher for complex deep search tasks by leveraging multi-stage data construction strategies.

2.1 Overview

To address the resource-intensive limitations of deep search systems, we propose SimpleDeepSearcher, a framework that achieves intelligence search through efficient supervised fine-tuning (SFT) with minimal training data. For constructing high-quality SFT data, we establish a systematically designed data synthesis and curation pipeline, as illustrated in Figure 1.

First, we replace static document retrieval (Karpukhin et al., 2020) with real-time network interactions, simulating human search behavior through an iterative cycle of "reasoning-searching-summarizing-generating." By directly processing raw HTML content via commercial search APIs, we capture diverse web information features—ranging from structured data snippets to unstructured narrative discourse. Based on this, we first filter input queries using domain heterogeneity, keyword diversity, and knowledge unit complexity to construct a maximally informative training foundation while ensuring selected queries align with real-world web search scenarios. Additionally, we apply a filtering mechanism to LLM-synthesized responses, implementing a four-dimensional quality filter that simultaneously optimizes format standardization, reasoning path control, question difficulty, and search effectiveness to guarantee response quality.

The framework's modular design offers three distinctive advantages: First, it exposes the model to authentic search artifacts and noise patterns through real web interactions. Second, our multidimensional filtering strategy enables state-of-the-art performance with remarkably small SFT datasets, eliminating dependency on resource-heavy RL training. Third, the decoupled architecture between data synthesis and model constraints provides exceptional flexibility that our curated datasets can enhance any LLMs while maintaining compatibility with emerging reasoning architectures and alternative training paradigms including RL. Since the searched content is not generated by the LLM, we mask out these tokens during the SFT process.

Our methodology achieves unprecedented efficiency in search-oriented model training, reducing

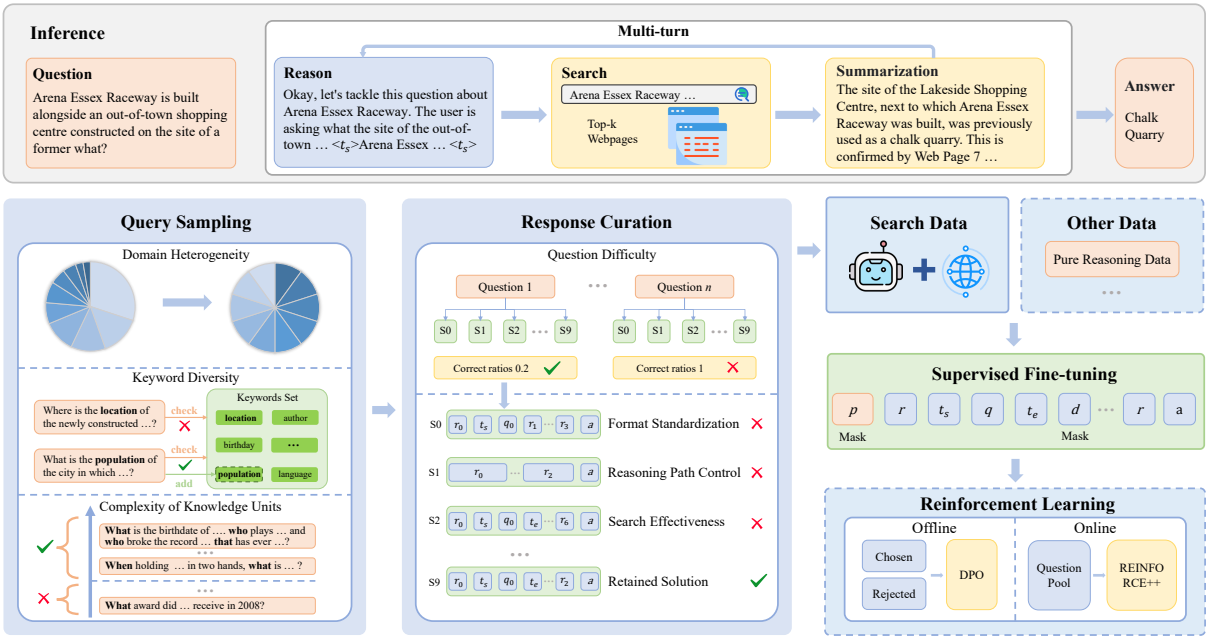

Figure 1: Overall framework of our proposed SimpleDeepSearcher approach. r denotes the reasoning content, q represents the search query, and d refers to the retrieved document after summarization. t_s and t_e are special tokens indicating the beginning and end of the search query, and a denotes the final answer.

computational demands while maintaining competitive performance through strategic data quality optimization rather than brute-force data quantity.

2.2 Data Synthesis in Real Web Environment

Typically, traditional retrieval-augmented generation (RAG) systems rely on closed and static knowledge corpora (*e.g.,* a Wikipedia snapshot). Such knowledge corpora exhibit two primary limitations: firstly, the content they contain often consists of refined and condensed segments (Chen et al., 2024); secondly, the information within these knowledge corpora lacks timeliness. Consequently, RAG systems are limited in their ability to simulate authentic user search behaviors, as users typically search within open, dynamic, and complex web environments where the information is not only diverse in format and varied in quality but is also frequently accompanied by redundancy and noise. In light of this, our data synthesis approach does not rely on curated document collections but is instead grounded in the real, open web environment. This authentic web environment also places greater demands on the model's capabilities for information extraction, synthesis, and reasoning.

Building upon the widely adopted iterative deep search process (Li et al., 2025a) of reason-search-summarize-generate, we develop an automated pipeline for large-scale training data synthesis. For each query, our framework at each iteration (1) initiates web searches via commercial APIs, (2) extracts and processes raw HTML content, (3) applies an LLM to reason over multi-source evidence, and (4) continues for the next iteration or stop iteration. By sampling multiple reasoning paths per query, we capture nuanced decision-making processes inherent to real-world information synthesis.

Our data synthesis strategy is firmly rooted in real web scenarios, which substantially enriches the diversity and authenticity of training samples. Building on this, this strategy enables scaling of high-quality trajectory data of reasoning and search interactions.

2.3 Diversity-aware Query Sampling

To engineer a deep search architecture with advanced query comprehension and reasoning capabilities, we implement a strategic repurposing of open-domain question answering (QA) resources. These curated datasets offer natural language questions that inherently require multi-hop information retrieval operations, thereby exhibiting strong task alignment with the cognitive demands of deep search systems (Zheng et al., 2025). Our selection protocol combines single-hop and multi-hop QA benchmarks through principled composition, ensuring coverage of both atomic and composite reasoning paradigms.

However, empirical evidence suggests that naive dataset scaling yields diminishing returns in SFT (Zhou et al., 2023). The efficacy of such approaches fundamentally depends on the intrinsic diversity and informational entropy of training instances. While existing open-domain QA corpora provide substantial volume, systematic analysis reveals three critical limitations: (1) domain-specific overrepresentation creating skewed knowledge distributions, (2) repetitive syntactic patterns reducing linguistic variability (Parmar et al., 2022), and (3) semantic simplicity thresholds below real-world query complexity. These factors collectively induce model brittleness and constrain cross-domain generalization potential (see Appendix B for details). To address these critical limitations, we introduce a diversity-aware query sampling strategy to implement systematic data filtering through tripartite orthogonal criteria:

Domain Heterogeneity encompasses the systematic classification of query semantics across distinct knowledge domains (*e.g.,* history, science, politics). This dimension ensures a balanced distribution of questions across different domains, thereby reducing domain-specific biases and enhancing generalization capabilities.

Keyword Diversity focuses on the distributional diversity of core semantic constituents (definition provided in Appendix B). we ensure non-redundant exposure to low-frequency conceptual entities, multi-order relational dependencies, and contextually ambiguous referential expressions. Such systematic variation compels the model to transcend superficial lexical pattern matching, instead developing reasoning architectures essential for interpreting complex entity interactions (Linzen, 2020).

Complexity of knowledge units captures the frequency of interrogative terms used in questions (*e.g.,* what, when), which serve as indicators of syntactic and semantic complexity. Questions with greater inquiry potential are given priority, ensuring comprehensive modeling of implicit reasoning chains triggered by diverse question formulations.

We developed a systematic query selection framework incorporating three complementary dimensions: domain heterogeneity, keyword diversity, and complexity of knowledge units. First, we partition the dataset into domain-specific clusters using the LLM-generated semantic classifications. Within each domain cluster, queries are ranked by knowledge unit complexity scores derived from conceptual density analysis. Subsequently, we perform iterative selection using a greedy algorithm that maximizes keyword diversity while maintaining inter-domain balance. The detailed procedure for query sampling is presented in Algorithm 1.

Algorithm 1 Diversity-aware Query Sampling

Input: Annotated dataset D with domains $d_1, d_2, \ldots, d_m$, target number of queries N

$N_d \leftarrow N/m$
$S \leftarrow \emptyset$ ▷ Initialize the target set
for i = 1 to n **do**
  $D_{d_i} \leftarrow \{x \in D \mid \text{domain}(x) = d_i\}$
  Sort D_{d_i} by descending interrogative words
  while $|S_{d_i}| < N_d$ and $D_{d_i} \neq \emptyset$ **do**
    $K \leftarrow \emptyset$ ▷ Initialize the keyword set
    for each sample x in D_{d_i} **do**
      if $|S_{d_i}| \geq N_d$ **then**
        break
      end if
      $kw \leftarrow \text{keywords}(x)$
      if $x \notin S$ and $kw \cap K = \emptyset$ **then**
        $S \leftarrow S \cup \{x\}$
        $K \leftarrow K \cup \text{keywords}(x)$
        $D_{d_i} \leftarrow D_{d_i} \setminus \{x\}$
      end if
    end for
  end while
end for
return S

2.4 Multi-Dimention Response Curation

Building upon the aforementioned data synthesis and query sampling strategies, we have successfully generated high-quality training data derived from real-world web environments. However, due to the inherent unpredictability of LLM reasoning, the quality of synthesized data exhibits considerable variability despite meticulous control over input and generation processes. Three primary issues are observed: (i) Formatting irregularities, such as inconsistent reasoning languages, non-standard formats for search and reasoning steps, and heterogeneous answer formats; (ii) reasoning redundancy, including hypothesis overgeneration, fabricated retrieval content, and excessive validation loops; (iii) inefficient search strategies, including redundant search exploration, contextual myopia and failure to retrieve relevant information.

The presence of low-quality reasoning outputs in language models not only compromises performance and transparency but also introduces noise into training signals, leading to inefficient computational resource utilization. To address these challenges, we developed a systematic filtering proto-

col that selects optimal solutions through rigorous evaluation of multiple responses per query.

To mitigate these issues, we impose strict constraints on both the format and content of sampled responses, retaining only those that satisfy all predefined criteria. Our filtering strategy, structured around four pillars, ensures retention of high-quality reasoning data while promoting efficient search integration.

Format Standardization. Filter out responses with mixed reasoning languages or incorrect reasoning and search formats, and correct answers with formatting errors to ensure consistency and standardization across all responses. Responses exhibiting mixed languages, irregular reasoning structures, or formatting inconsistencies were excluded. Automated correction aligned remaining answers with standardized templates.

Reasoning Path Control. Strictly limit the use of reflection expressions (*e.g.,* alternatively, wait, etc.) and control the length of reasoning to avoid unnecessary and redundant reasoning steps. Reasoning models tend to hypothesize, infer, and reflect based on internal knowledge, often resulting in delayed use of search tools and inefficient reasoning. By regulating the reasoning path, the model can learn to seamlessly integrate search into its inference process and adopt more efficient reasoning strategies.

Question Difficulty. Filter out questions with consistently high accuracy across multiple reasoning attempts and prioritize those with lower accuracy. Accuracy obtained from multiple samples can serve as a proxy for question difficulty. Selecting more challenging questions helps enhance the model's ability to handle complex queries.

Search Effectiveness. Among multiple candidate responses, prioritize those with fewer search steps and more diverse search content. This encourages the model to not only invoke search capabilities but also to learn how to formulate effective subqueries based on the original question for efficient information retrieval.

Based on the above dimensions, we first collect metadata for each response, such as the number of search steps, reasoning length, and accuracy. Subsequently, responses are filtered sequentially based on *format standardization* and *reasoning path control*. Then, based on *question difficulty*, questions with high accuracy are removed. For each remaining question, we retain multiple high-quality responses that meet all constraints and sort them by search steps. According to *search effectiveness*, the response with the fewest search steps is selected as the final answer. Through this process, we ultimately obtained 871 high-quality question-answer pairs. This multi-criteria approach not only enhances model training efficiency but also provides insights into optimal human-AI reasoning patterns.

3 Experiments

3.1 Experimental Setup

Datasets. We sample training data from single-hop and multi-hop knowledge-intensive QA datasets to cover a wide range of domains and question difficulty. For single-hop questions, we use Natural Questions (Kwiatkowski et al., 2019) and SimpleQA (Wei et al., 2024). For multi-hop questions, we use HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Tang and Yang, 2024), and MultiHopRAG (Tang and Yang, 2024). To test the model's performance on out-of-domain data, we select Bamboogle (Press et al., 2022), FRAMES (Krishna et al., 2024), and GAIA (Mialon et al., 2023) benchmarks. In addition, we further conduct evaluations on the more challenging benchmarks including xbench-DeepSearch, BrowseComp-ZH (Zhou et al., 2025), and BrowseComp-EN (Wei et al., 2025). These datasets are not used during training and help evaluate how well the model works on new domains. We evaluate on 500 randomly sampled instances from the validation sets of HotpotQA, 2WikiMultiHopQA, and MuSiQue. For GAIA, we use 103 examples from the text-only validation subset (Li et al., 2025b), while for BrowseComp-EN we randomly sample 300 instances. For the remaining benchmarks, we use their full test sets.

Metrics. We report results using two metrics: F1 score and LLM-as-Judge (LasJ). The F1 score captures the word-level similarity between the predicted and golden answers, while LasJ leverages GPT-4o-mini to evaluate the correctness of the predicted response.

Baselines. We consider following type of baselines: *Naive Generation*: Direct generation of answers without retrieval. *Standard RAG*(Zhao et al., 2024): Directly retrieve relevant documents by querying the original question. *Search-o1* (Li et al., 2025a): Encourages the model

Models	Methods	2Wiki†		MuSiQue†		Bamboogle‡		Frames‡		GAIA‡	
		F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ	F1	LasJ
Qwen-7B	Directly Gen	27.7	26.8	9.6	6.2	18.2	17.6	12.6	10.1	13.6	6.8
	Standard RAG	34.8	34.8	17.2	14.6	31.5	31.2	13.9	13.5	-	-
	Search-o1	48.0	51.2	21.5	20.6	57.9	59.2	30.9	35.0	24.3	21.4
	R1-Searcher	63.4	66.4	**29.0**	26.8	68.2	68.8	34.4	40.3	24.1	20.4
	DeepResearcher	59.7*	66.6*	27.1*	29.3*	71.0*	72.8*	-	-	-	-
	SimpleDeepSearcher	**70.6**	**79.8**	28.2	**29.4**	**74.5**	**76.8**	**44.9**	**55.3**	**39.3**	**36.9**
Qwen-32B	Directly Gen	31.7	31.2	13.3	12.4	25.7	25.6	15.6	14.2	18.6	13.9
	Standard RAG	43.7	45.0	19.5	16.8	40.8	40.8	19.4	19.4	-	-
	Search-o1	64.9	74.8	29.1	30.6	74.4	78.4	47.2	56.8	36.5	34.0
	SimpleDeepSearcher	**71.9**	**81.2**	**30.6**	**33.0**	**78.1**	**80.0**	**50.1**	**60.8**	**42.1**	**40.8**
DDQ-32B	Directly Gen	36.9	36.2	19.6	16.0	32.6	32.8	27.8	29.2	14.8	9.7
	Standard RAG	48.1	50.0	24.0	21.6	42.6	46.4	26.5	28.9	-	-
	Search-o1	49.6	55.2	25.4	23.8	65.7	68.0	32.2	38.7	23.2	24.3
	SimpleDeepSearcher	**69.0**	**77.4**	**32.9**	**33.6**	**80.5**	**83.2**	**52.2**	**63.8**	**42.0**	**41.7**
QwQ-32B	Directly Gen	39.6	39.8	18.9	17.4	29.6	29.6	28.1	31.3	16.8	11.7
	Standard RAG	48.4	50.6	21.8	19.4	42.5	46.4	27.4	31.6	-	-
	Search-o1	69.4	78.0	34.3	36.4	78.7	78.4	51.6	64.4	38.3	37.9
	SimpleDeepSearcher	**75.6**	**84.4**	**34.8**	**37.4**	**83.4**	**88.0**	**56.8**	**68.8**	**48.9**	**50.5**

Table 1: Performance comparisons between SimpleDeepSearcher and the baselines on QA benchmarks. The best results are in **bold** and the second-best are underlined. †/‡ represents in-domain/out-domain datasets. Results marked with * are cited from their official paper or report. *Qwen-7B*, *Qwen-32B*, *DDQ-32B* are the abbreviations of Qwen-2.5-7B-Instruct, Qwen-2.5-32B-Instruct, and Deepseek-Distilled-Qwen-2.5-32B, respectively.

Model	Xbench-DeepSearch	BrowseComp-ZH	BrowseComp-EN
Webthink-RL	24.0*	7.3*	2.8*
WebDancer-32B	**38.7***	14.1*	2.5*
SimpleDeepSearcher	30.0	**14.5**	**4.3**

Table 2: Results on the more challenging Xbench-DeepSearch, BrowserComp-ZH, and BrowseComp-EN benchmarks. The results are evaluated with LLM-as-Judge. Results marked with * are cited from other papers or reports. The best results are in **bold**.

Category	Method	Bamboogle		GAIA	
		F1	LasJ	F1	LasJ
	Ours	**74.5**	**76.8**	**39.3**	**36.9**
Query Sampling	w/o DD	69.7	70.4	35.6	35.8
	w/o KD	73.2	76.0	32.9	31.1
	w/o CIW	71.7	74.4	32.1	29.1
Environment	w/o Online	74.0	74.4	30.4	28.2
Response Curation	w/o FS	72.8	75.2	38.0	**36.9**
	w/o RPC	71.7	74.4	31.6	30.1
	w/o QD	67.1	70.4	32.9	32.0
	w/o SE	72.6	73.6	37.7	35.0

Table 3: Results of variants of SimpleDeepSearcher on Bamboogle and GAIA.

to perform self-initiated retrieval using prompts. *RAG-RL*: R1-Searcher (Song et al., 2025), DeepResearcher (Zheng et al., 2025), WebThinker-RL (Li et al., 2025b), and WebDancer (Wu et al., 2025), the open-source models trained with reinforcement learning to enable self-initiated retrieval. We conduct experiments using the following model backbones with an online search engine, including Qwen-2.5-7B-Instruct, Qwen-2.5-32B-Instruct, Deepseek-Distilled-Qwen-2.5-32B, and QwQ-32B.

Implementation Details. Our experimental setup consists of three main components: SFT, generation, and query sampling. In the SFT phase, we use a total batch size of 64 and train for 6 epochs with a learning rate of 1e-5, warmup ratio of 0.03, and a sequence length of 30,000 tokens. During fine-tuning, external retrieval documents are masked to avoid learning from noisy or spurious information. For generation, all models are configured with a maximum sequence length of 20,480 tokens, temperature of 0.6, top-p of 0.95, and top-k of 40. During query sampling, we used QwQ-32B to annotate each query with its corresponding domain and keywords. For data synthesis, we employed QwQ-32B as the reasoning model and Google Search API as the search engine, with a maximum of 10 search calls and 15 reasoning turns per query. For each query, we sampled 10 candidate responses. All prompts used in the experiments are provided in Appendix F.

3.2 Main Results

Table 1 presents the main results of the proposed SimpleDeepSearcher and baselines across five representative datasets.

Firstly, SimpleDeepSearcher consistently outperforms all existing baseline methods across five benchmark datasets. Specifically, it achieves the best performance not only on in-domain datasets (*i.e.,* 2Wiki, MuSiQue) but also shows substantial improvements on out-of-domain datasets (*i.e.,* Bamboogle, FRAMES, GAIA), demonstrating its strong generalization ability.

Besides, SimpleDeepSearcher consistently outperforms reinforcement learning-based methods such as R1-Searcher and DeepResearcher across most evaluation metrics. These approaches are trained on large-scale datasets using complex reinforcement learning algorithms. In contrast, our method relies on supervised fine-tuning with only 871 training examples. This demonstrates that our framework achieves strong performance while maintaining high data efficiency, offering a simple yet effective alternative for improving deep search capabilities.

Thirdly, SimpleDeepSearcher achieves stable and substantial performance improvements across models with diverse backbones and parameter scales, ranging from 7B to 32B. For instance, compared to Search-o1, it achieves relative improvements of 48.3%, 42.6%, and 11.5% on Qwen2.5-7B-Instruct, DeepSeek-R1-Distill-Qwen-2.5-32B, and QwQ-32B, respectively. This demonstrates the strong generalization ability of our distillation and self-distillation strategies, with the selected data consistently leading to performance gains across heterogeneous model architectures.

In addition, table 2 presents the experimental results on more complex QA datasets. These datasets are specifically designed for AI agents, requiring models to possess end-to-end planning, search, reasoning, and summarization capabilities. Notably, our model still demonstrates strong performance compared to models trained with reinforcement learning. This result further underscores the robust generalization ability of our model.

4 Further Analysis

4.1 Ablation Study

To validate the effectiveness of the proposed SimpleDeepSearcher, we conduct a comprehensive ablation analysis using Qwen2.5-7B-Instruct on the Bamboogle and GAIA datasets. We conduct detailed ablation studies on three main aspects: (1) Query Sampling: *w/o DD* removes domain diversity filter, *w/o KD* removes keyword diversity filter, *w/o CIW* removes coverage of interrogative words filter; (2) Environment: *w/o Online* uses local dense dense retrieval to synthesize training data; (3) Response Curation: *w/o FR* removes format regularization filter, *w/o RPC* removes reasoning path control filter, *w/o QD* removes question difficulty filter, *w/o SC* search count filter. As observed, all ablated variants exhibit a decline in performance compared to our full method, underscoring the integral contribution of each component. Among them, *w/o QD* leads to the most significant performance drop, suggesting that question difficulty plays a crucial role in training. More challenging questions are more likely to stimulate the model's autonomous retrieval capabilities during reasoning.

Method	Bamboogle		GAIA	
	F1	LasJ	F1	LasJ
Distilled (Ours)	74.5	76.8	**39.3**	36.9
w. DPO	**75.0**	**79.2**	39.0	**37.9**
w. Reinforce++	73.8	75.8	29.4	24.3

Table 4: Evaluation Results of RL-based Methods.

Model	#Alternatively	#Search	Output Length
QwQ-32B	7.933	2.390	867.148
QwQ-32B-SFT	4.051	2.329	581.731

Table 5: Statistical analysis of model outputs.

4.2 Effect of Post-SFT RL

Recent studies have investigated the integration of RL and RAG (Song et al., 2025; Jin et al., 2025; Zheng et al., 2025). We further examine the advantages and limitations of applying RL after SFT.

We apply DPO and REINFORCE++ to conduct offline and online reinforcement learning, respectively. As shown in Table 4, the model trained with DPO achieves further improvements over the SFT baseline, demonstrating the effectiveness of offline preference optimization (see Appendix C for details). In contrast, the model trained with REINFORCE++ produces significantly shorter responses (see Appendix D for details) and shows notable performance degradation on both the Bamboogle and GAIA benchmarks. This suggests that online RL mainly triggers retrieval behavior, but brings little benefit to models that are already good at retrieval. We hypothesize that the success of offline DPO stems from its ability to leverage high-quality trajectories generated by a strong LLM. These trajectories provide informative preference

Model	Plan.	Search	Summ.
Qwen-7B	0.416	0.455	0.363
Qwen-7B-SFT	**0.590**	**0.677**	**0.584**
QwQ-32B	0.623	0.680	0.594
QwQ-32B-SFT	**0.629**	**0.713**	**0.624**

Table 6: Proportion of instances containing the correct answer at each stage of the inference process (Planning, Search, and Summarization), before and after SFT.

signals and stable supervision, allowing the model to refine its reasoning and search strategies.

4.3 Effect of SFT on Redundancy

In this part, we analyze how SFT impacts redundant reasoning and search behavior. Specifically, we focus on three indicators: (1) the frequency of the reflective word "alternatively", which signals hesitation or divergent reasoning; (2) the average length of reasoning chains, measured by output length; and (3) the number of search calls made during inference. Our analysis is based on the QwQ-32B model, evaluated on the 2Wiki, MuSiQue, and Bamboogle datasets. As shown in Table 5, the average use of "alternatively" and the overall output length are both significantly reduced after SFT. Moreover, the model issues fewer search queries. These results indicate that our self-distillation approach improves both the reasoning clarity and search efficiency of the model. This improvement can be attributed to the high-quality training data selected through our proposed method.

4.4 Effect of SFT on Stage-wise Performance

In this part, we analyze how training improves the performance of each sub-task in our approach, including iterative search, planning, and summarization. We evaluate the proportion of cases in which the final answer appears during each subprocess to quantify the efficiency of that stage. To eliminate interference from the summarization stage, all summarization models are kept identical during inference, with detailed settings provided in Appendix E. The results are shown in Table 6. We can observe substantial improvements across all components, with the search component showing the most significant gain. This suggests that training effectively enhances the model's ability to generate more coherent reasoning and search trajectories, leading to more accurate information retrieval and improved overall model performance.

Models	Summarization Model	Bamboogle		GAIA	
		F1	LasJ	F1	LasJ
Qwen-7B-SFT	before training	70.8	71.2	28.0	26.2
	after training	67.5	68.8	23.9	21.4
	QwQ-32B	**74.5**	**76.8**	**39.3**	**36.9**
	GPT-4o-mini	70.9	**76.8**	33.7	32.0
QwQ-32B-SFT	before training	**83.5**	**88.0**	**48.9**	**50.5**
	after training	83.9	86.4	43.2	47.6
	GPT-4o-mini	80.0	80.8	40.5	44.7

Table 7: Performance comparison across all benchmarks using different summarization models.

Training Data	Bamboogle		GAIA		AIME	
	F1	LasJ	F1	LasJ	F1	LasJ
- Reasoning	74.5	76.8	**39.3**	36.9	13.3	13.3
+ Reasoning	**76.9**	**80.8**	37.2	**37.9**	**20.0**	**20.0**

Table 8: Results of the SimpleDeepSearcher trained w/ and w/o reasoning data across three benchmarks.

4.5 Effect of Summarization Model

This part investigates the impact of the summarization model on overall performance. We fix the reasoning model and conduct a comparative analysis of overall performance using different summarization models. As shown in Table 7, QwQ-32B demonstrates the strongest summarization capability and is therefore selected as the summarization model for all reasoning models. Furthermore, using fine-tuned models for summarization leads to performance degradation on downstream tasks compared to their pre-trained counterparts. This might be attributed to the reduced long-text summarization ability of the fine-tuned models, due to the distributional shifts on a limited task and domain of the training data. This decline is more pronounced for models with fewer parameters.

4.6 Effect of Additional Reasoning Data

We further investigate the impact of incorporating complex mathematical reasoning data on Qwen2.5-7B-Instruct. As shown in Table 8, this leads to consistent performance gains across all benchmarks. Furthermore, Figure 2 and Table 9 reveals significant alterations in the model's behavioral patterns on two kinds of tasks: for tasks emphasizing complex reasoning (*e.g.,* AIME, GAIA), the model generates longer and more in-depth reasoning outputs; for search tasks (*e.g.,* Bamboogle), the model performs more searches and explores more thoroughly. These findings suggest that incorporating complex reasoning data helps the model learn to adapt its reasoning and search strategies to

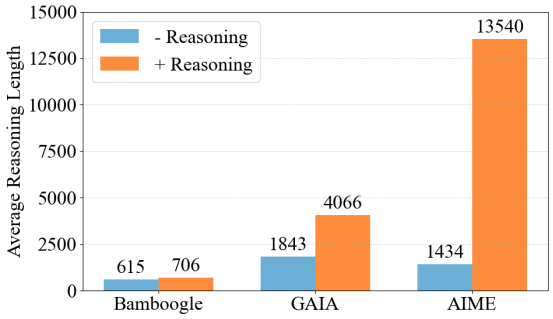

Figure 2: Average reasoning length across different benchmarks w/ and w/o reasoning data for training.

Training Data	Search Count		
	Bamboogle	GAIA	AIME
- Reasoning	1.552	1.757	0
+ Reasoning	1.672	1.845	0

Table 9: Average search count across different benchmarks of the model trained w/ and w/o reasoning data.

the specific demands of a task. This adaptability is critical for addressing complex and diverse queries.

5 Conclusion

In this work, we present SimpleDeepSearcher, a lightweight yet effective framework for deepsearch tasks, addressing the limitations of existing RAG methods that rely heavily on complex training paradigms or suffer from distributional mismatches. By leveraging realistic web search simulations and a multi-criteria data curation strategy, we construct high-quality training trajectories that enable efficient supervised fine-tuning. Despite using only 871 curated samples, our method achieves substantial gains over RL-based baselines across diverse in-domain and out-of-domain benchmarks. Our results highlight the potential of strategic data engineering to empower deep search reasoning.

Limitation

Despite our substantial efforts, this work is subject to two limitations stemming. Due to limitations in training resources and hardware, we conducted distillation training on 7B and 32B models. In future work, we plan to train and evaluate our framework on larger-scale models (*i.e.,* 72B) to further verify its generalization capability and robustness. Additionally, because of the inherent difficulty in synthesizing multi-hop data, the original data used for distillation primarily consisted of relatively simple multi-hop questions. If more realistic and challenging multi-hop queries can be synthesized in the future, applying our framework for filtering and training may yield even better performance.

Ethics Statement

We strictly adhere to ethical standards. We follow the relevant licenses and guidelines for dataset usage, ensuring that no personal or offensive information is included. AI assistance was only utilized during the paper refinement process. Our trained models do not display any potential biases or discriminatory behavior, and we rigorously comply with research ethics throughout the entire development and evaluation process.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China under Grant No. 92470205 and 62222215, Beijing Natural Science Foundation under Grant No. L233008 and Beijing Municipal Science and Technology Project under Grant No. Z231100010323009.

References

Salaheddin Alzubi, Creston Brooks, Purva Chiniya, Edoardo Contente, Chiara von Gerlach, Lucas Irwin, Yihan Jiang, Arda Kaz, Windsor Nguyen, Sewoong Oh, and 1 others. 2025. Open deep search: Democratizing search with open-source reasoning agents. *arXiv preprint arXiv:2503.20201*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Jiangning Liu, Wenwei Zhang, Kai Chen, and Feng Zhao. 2024. Mindsearch: Mimicking human minds elicits deep ai searcher. *arXiv preprint arXiv:2407.20183*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

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*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Jinhao Jiang, Jiayi Chen, Junyi Li, Ruiyang Ren, Shijie Wang, Wayne Xin Zhao, Yang Song, and Tao Zhang. 2024. Rag-star: Enhancing deliberative reasoning with retrieval augmented verification and refinement. *CoRR*, abs/2412.12881.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

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.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain QA of LLMs. In *The Twelfth International Conference on Learning Representations*.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2024. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. *arXiv preprint arXiv:2409.12941*.

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.

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

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

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? *arXiv preprint arXiv:2005.00955*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

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

Mihir Parmar, Swaroop Mishra, Mor Geva, and Chitta Baral. 2022. Don't blame the annotator: Bias already starts in the annotation instructions. *arXiv preprint arXiv:2205.00415*.

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*.

Peng Qi, Haejun Lee, Oghenetegiri Sido, Christopher D Manning, and 1 others. 2020. Retrieve, rerank, read, then iterate: Answering open-domain questions of arbitrary complexity from text. *arXiv preprint arXiv:2010.12527*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294*.

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

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

Richard S Sutton, Andrew G Barto, and 1 others. 1999. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. 2025. Atom of thoughts for markov llm test-time scaling. *arXiv preprint arXiv:2502.12018*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. 2025. Chain-of-retrieval augmented generation. *CoRR*, abs/2501.14342.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*.

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*.

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

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*.

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*.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *CoRR*, abs/2402.19473.

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

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, and 1 others. 2025. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*.

A Related Work

Retrieval-Augmented LLMs. To improve the factual precision of LLM-generated texts (Zhao et al., 2024), researchers enhance LLMs with retrieval-augmented generation (RAG) (Guu et al., 2020). Various approaches have been proposed, such as branching-based methods (Kim et al., 2024), summarization-based methods (Li et al., 2023), and adaptive retrieval techniques (Jeong et al., 2024). With the increase in model parameters, LLMs have demonstrated chain-of-thought reasoning capabilities (Wei et al., 2022), and many researchers to integrated such reasoning with RAG via prompt engineering (Shao et al., 2023; Trivedi et al., 2023). Other studies have attempted to distill retrieval abilities into smaller models through supervised fine-tuning (Asai et al., 2024). However, these approaches limit the model's capacity with a fixed reasoning path.

Enhancing LLMs with Search. Recently, several deep search frameworks are proposed (Alzubi et al., 2025). They integrate large language models with search engines in a more flexible and dynamic manner. Search-o1 (Li et al., 2025a) simulates deep search in LLMs through prompt engineering, allowing them to retrieve information independently during multi-step reasoning. R1-Searcher (Song et al., 2025) and Search-R1 (Jin et al., 2025) equip large language models with retrieval tools and train them end-to-end using reinforcement learning. This approach effectively enhances the model's ability to interleave reasoning with retrieval during inference. However, due to the inherent complexity of RL and its high computational demands, conducting large-scale experiments on full-sized LLMs remains challenging. SimpleDeepSearcher synthesizes high-quality training data via broad query sampling and precise filtering, enabling strong deep search performance with minimal training cost.

B Details of Diversity-Aware Query Sampling

In analyzing open-source data, we identified three critical limitations:

(1) Domain-specific overrepresentation creating skewed knowledge distributions. As shown in the Figure 3, we present the domain distribution of the pre-filtered data. It can be observed that certain domains (such as film and geography) account for a considerable proportion. This imbalance risks inducing uneven knowledge distributions in the training data.

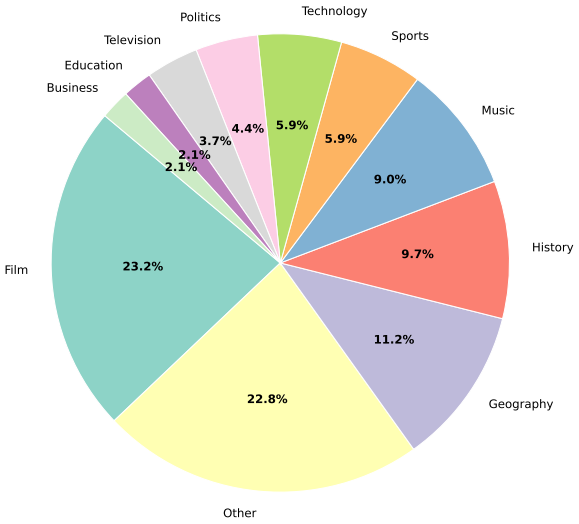

Figure 3: Domain distribution of the data before filtering.

(2) Repetitive syntactic patterns reducing linguistic variability. Due to the construction methods of open-source datasets, we observe substantial redundancy in syntactic structures. A typical example is the prevalence of "A and B" style comparative queries (*e.g.,* "Do both directors of films Paper Bullets and Karakolda Ayna Var share the same nationality?" vs. "Do both directors of films Jatt Juliet and Sciopèn share the same nationality?"). Similarly, numerous queries repetitively compare identical attributes such as age.

(3) Semantic simplicity thresholds below real-world query complexity. Many queries in open-source datasets are overly simplistic, such as "What nationality is John Harbaugh's father?". Such questions impose only minimal demands on deep search or reasoning, as they can often be answered through a single lookup. Consequently, their utility in fostering more advanced model capabilities is limited.

We define "core semantic constituents" as follows:

- Key entities (*e.g.,* films, people, locations)
- Critical attributes (*e.g.,* age, duration, population)
- Core relationships (*e.g.,* comparison, causality)
- Measurement dimensions (*e.g.,* time, quantity)

For example, in the query "Which film whose director is younger, Charge It To Me or Danger: Diabolik?", the extracted keywords based on the above schema are "film" and "age".

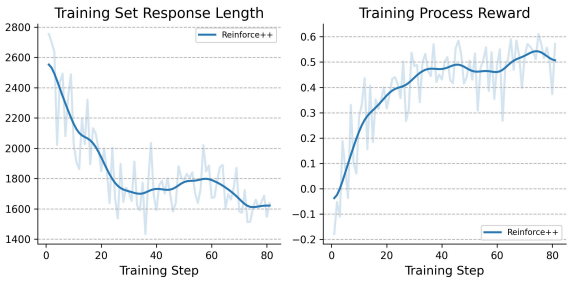

Figure 4: Changes in Sequence Length and Reward During REINFORCE++ Training.

C DPO Detailed Settings

Our objective was to identify answer trajectories that were both correct and demonstrated efficient reasoning and search paths. To this end, we construct preference pairs (R_w, R_l), where R_w denotes the preferred trajectory and R_l the rejected one. We repurposed our previously established pipeline for query sampling and data synthesis. During the data synthesis stage, we generate responses using the strongest SFT-trained model, SDS-QwQ-32B-SFT, and the target model to be optimized, SDS-Qwen-7B-SFT. Responses generated by SDS-QwQ-32B-SFT that pass both the formatting and reasoning path control checks are treated as positive examples, while those generated by SDS-Qwen-7B-SFT that fail these checks are treated as negative examples. Ultimately, we construct a dataset consisting of approximately 875 training pairs.

For Direct Preference Optimization (DPO) training, we utilize a learning rate of 5×10^{-7}, a β of 0.1, training for 5 epochs with a batch size of 256, a warm-up ratio of 0.1, and a maximum sequence length of 10000.

D REINFORCE++ Detailed Settings

To construct the reinforcement learning (RL) dataset, we utilized the model that had been trained though SimpleDeepSearcher to perform rollout sampling on the training sets of 2Wiki and HotpotQA. For each question, eight candidate responses were generated. From this pool, we selected 2480 samples corresponding to questions with one to six correct answers, ensuring diversity in the RL training data.

The reward function employed in REINFORCE++ consists of two components: an answer reward and a format penalty. The answer reward is calculated as the F1 score between the predicted answer and the reference answer, providing a direct measure of response accuracy. In addition, a discrete format penalty of -2 is applied if any of the following undesirable behaviors are detected:

• *Self-Retrieved Content:* The model fabricates content that is not retrieved from external sources.

• *Contains Gibberish:* The generated output contains nonsensical, irrelevant, or corrupted text segments.

• *Excessive Analytical Markers:* The response contains more than 5 occurrences of phrases such as *Alternatively*, *Wait*, or *Hmm*, which are treated as signals of incoherent reasoning.

• *Lack of Boxed Answers or Excessive Reasoning Length:* The model either executes more than 8 retrieval steps or the token length of the analytical content between any two retrievals exceeds 8,096 tokens.

If none of these conditions are met, no penalty is applied. To maintain on-policy training throughout the RL process, we adjusted the batch size to ensure that learning was based on the most recent policy rollouts. Figure 4 shows the variations in response length and reward values observed during the training process.

E Model Performance Enhancement Analysis Settings

We conduct a comparative analysis of Qwe2.5-7B-Instruct and QwQ-32B before and after training across the 2Wiki, Bamboogle, and MuSiQue benchmarks. During inference, we fix the summarization model to QwQ-32B across all comparisons to eliminate potential interference from the summarization component (the impact of the summarization model will be further discussed in Section 4.5).

F Instruction Templates

Instruction for Annotation

You are an advanced semantic analyzer. For the given question, perform the following tasks step by step:

1. **Domain Identification**:

- Determine the broad subject category (domain) this question belongs to.

- Examples: film, history, biology, geography, politics, technology, etc (or any other suitable domain)

2. **Key Point Extraction**:

- Identify 2-4 core semantic components that are crucial for answering

- Include:

• Key entities (e.g., films, people, locations)

• Critical attributes (e.g., age, duration, population)

• Core relationships (e.g., comparison, causality)

• Measurement dimensions (e.g., time, quantity)

- Exclude filler words and non-essential descriptors\n

Output Requirements:

- Use JSON format: {{"domain": "...", "key_points": [...]}}

- Keep key_points concise (1-2 words each)

- Use lowercase for all outputs

- Separate multiple key_points with commas\n

Examples:

Question: "Which film whose director is younger, Charge It To Me or Danger: Diabolik?"

Output: {{"domain": "film", "key_points": ["director", "age"]}}\n

Now process this question:

{{Question}}

Instruction for LLM as Judge

Given a Question and its Golden Answer, verify whether the Predicted Answer is correct. The prediction is correct if it fully aligns with the meaning and key information of the Golden Answer. Respond with True if the prediction is correct and False otherwise.

Golden Answer may have multiple options, and matching any one of them is considered correct.\n

Question: {question}

Golden Answer: {reference}

Predicted Answer: {prediction}

Instruction for Reasoning Model

You are a reasoning assistant with the ability to perform web searches to help you answer the user's question accurately. You have special tools:\n\n

- To perform a search: write <|begin_search_query|> your query here <|end_search_query|>.\n

Then, the system will search and analyze relevant web pages, then provide you with helpful information in the format <|begin_search_result|> ...search results... <|end_search_result|>.\n\n

Whenever you encounter a topic, fact, or piece of information you are uncertain about or need further details on, please perform a search to gather more accurate, up-to-date, or specific information. You can repeat the search process multiple times if necessary. The maximum number of search attempts is limited to {MAX_SEARCH_LIMIT}.\n\n

Once you have all the information you need, continue your reasoning.\n\n

Remember:\n

- Use <|begin_search_query|> to request a web search and end with <|end_search_query|>.\n
- When done searching, continue your reasoning.\n
- Do not generate <|begin_search_result|> and <|end_search_result|> tags yourself.\n\n

Please answer the following question. You should think step by step to solve it.\n\n

Provide your final answer in the format \\boxed{YOUR_ANSWER}.\n\n

Question:\n{question}\n\n

Instruction for Summarization Model

Task Instruction:\n\n

You are tasked with reading and analyzing web pages based on the following inputs: **Previous Reasoning Steps**, **Current Search Query**, and **Searched Web Pages**. Your objective is to extract relevant and helpful information for **Current Search Query** from the **Searched Web Pages** and seamlessly integrate this information into the **Previous Reasoning Steps** to continue reasoning for the original question.\n

Guidelines:\n

1. **Analyze the Searched Web Pages:**

- Carefully review the content of each searched web page.
- Identify factual information that is relevant to the **Current Search Query** and can aid in the reasoning process for the original question.\n

2. **Extract Relevant Information:**

- Select the information from the Searched Web Pages that directly contributes to advancing the **Previous Reasoning Steps**
- Ensure that the extracted information is accurate and relevant.\n

3. **Output Format:**

- Present the helpful information for current search query: beginning with `**Final Information**` as shown below.

Final Information\n

[Helpful information]\n

Inputs:

- **Previous Reasoning Steps:**

{prev_reasoning}\n

- **Current Search Query:**

{search_query}\n

- **Searched Web Pages:**

{document}\n

Now you should analyze each web page and find helpful information based on the current search query "{search_query}" and previous reasoning steps.