WIDESEEK-R1: EXPLORING WIDTH SCALING FOR BROAD INFORMATION SEEKING VIA MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in large language models (LLMs) have largely focused on depth scaling, where a single agent solves long-horizon problems through multi-turn reasoning and tool use. However, as tasks grow broader, the key bottleneck shifts from individual competence to organizational capability. In this work, we explore width scaling through multi-agent systems as a complementary approach to broad information seeking. Existing multi-agent systems often rely on handcrafted workflows and turn-taking interactions that fail to parallelize work effectively. To bridge this gap, we propose WIDESEEK-R1, a lead-agent–subagent framework trained via multi-agent reinforcement learning (MARL) to jointly learn scalable orchestration and parallel execution. Using a shared LLM with isolated contexts and specialized tools, WIDESEEK-R1 jointly optimizes the lead agent and parallel subagents on a curated dataset of 20,000 broad information-seeking tasks. Extensive experiments show that WIDESEEK-R1-4B achieves an item F1 score of 40.0% on the WideSearch benchmark, comparable to that of single-agent DeepSeek-R1-671B. Furthermore, WIDESEEK-R1-4B exhibits consistent performance gains as the number of parallel subagents increases, highlighting the effectiveness of width scaling. The anonymous project page is at `https://sites.google.com/view/wideseek-r1`.

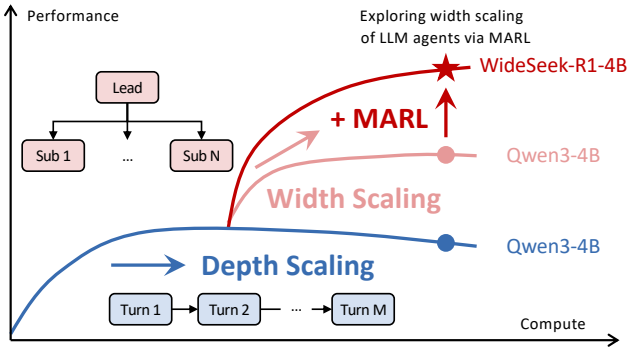


Figure 1: Comparison of depth and width scaling. While depth scaling enhances performance through sequential multi-turn interactions, width scaling orchestrates multi-agent systems for parallel execution. WIDESEEK-R1 advances width scaling via MARL for joint orchestration and execution.

1 INTRODUCTION

Recent advances in large language models (LLMs) (Guo et al., 2025; Kimi Team et al., 2025; Google, 2025) have significantly improved single-agent capabilities in multi-turn reasoning and tool use. Existing work mainly explores *depth scaling*, using longer chains of thought and sequential actions for long-horizon tasks. However, as tasks expand in breadth, the primary bottleneck shifts from individual competence to organizational capability (Kimi Team et al., 2026). This motivates a complementary dimension of *width scaling* with multi-agent systems, where a lead agent decomposes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

broad objectives into independent subtasks and orchestrates parallel subagents to tackle problems beyond the capacity of a single agent.

Broad information seeking (Wong et al., 2025) serves as an ideal testbed to explore the width scaling dimension. Unlike deep research (Mialon et al., 2023; Wei et al., 2025a), which intensively investigates a single complex query, broad information seeking gathers and synthesizes attributes of multiple entities into a structured tabular format across diverse subtasks. Single-agent methods suffer from two limitations in such scenarios. First, context pollution (Anthropic, 2026) degrades the agent's performance as its context accumulates irrelevant information from previous subtasks. Second, sequential execution restricts efficiency by forcing the agent to process independent subtasks serially. These limitations underscore the necessity of multi-agent systems, which naturally enable context isolation and parallel execution for effective width scaling.

However, existing multi-agent systems have yet to fully realize the potential of width scaling, primarily because few systems are trained end-to-end to learn scalable orchestration and parallel execution. At the orchestration level, most prior work (Li et al., 2023; Wu et al., 2024) relies on handcrafted workflows rather than learned agents, hindering flexible and scalable coordination of multiple agents. At the execution level, current systems (Hu et al., 2025; Li et al., 2025) typically process subtasks one at a time and adopt turn-taking interactions that serialize progress and fail to parallelize subtasks. As a result, the performance of multi-agent systems is bottlenecked by limited scalability and insufficient parallelization.

To bridge this gap, we introduce WIDESEEK-R1, a lead-agent–subagent system trained via multi-agent reinforcement learning (MARL) to synergize scalable orchestration and parallel execution for broad information seeking. We instantiate the lead agent and the subagents using a shared LLM with different tools and isolated contexts. The lead agent handles task decomposition and multi-turn orchestration by delegating subtasks, while subagents execute them in parallel and return their findings. To enable multi-agent learning beyond multi-hop QA, we curate 20,000 broad information-seeking tasks and train WIDESEEK-R1-4B via MARL on trajectories from both roles, jointly optimizing scalable orchestration and parallel information seeking.

We conduct extensive experiments to demonstrate that WIDESEEK-R1 pushes the boundaries of width scaling. On the WideSearch benchmark, WIDESEEK-R1-4B achieves an item F1 score of 40.0%, comparable to that of single-agent DeepSeek-R1-671B, and significantly outperforms multi-agent 8B baselines. Furthermore, we investigate the scaling properties of both depth and width dimensions. While depth scaling quickly reaches a plateau, WIDESEEK-R1-4B exhibits continuous performance gains as the number of parallel subagents increases. We also evaluate our method on standard QA benchmarks and perform ablation studies on role assignment and training data to validate that MARL jointly improves orchestration and parallel information seeking.

In summary, our contributions are threefold:

- We introduce WIDESEEK-R1, a multi-agent system trained via MARL to synergize scalable orchestration and parallel execution for broad information seeking.
- We open-source a large-scale dataset of 20,000 broad information-seeking tasks, offering a complementary training resource to existing multi-hop datasets.
- We demonstrate the effectiveness of width scaling with WIDESEEK-R1-4B, which achieves performance comparable to DeepSeek-R1-671B and exhibits consistent gains as the number of parallel agents increases.

2 RELATED WORK

Scaling Dimensions in LLMs. The evolution of LLMs has been primarily driven by two scaling paradigms: training-time scaling and test-time scaling. Training-time scaling enhances foundational capabilities of LLMs by increasing model parameters, dataset size, and total training compute (Kaplan et al., 2020; Hoffmann et al., 2022), whereas test-time scaling boosts performance by allocating more compute at inference time. A prominent line of work scales depth at test time, where reasoning models (Jaech et al., 2024; Guo et al., 2025) and agentic models (Kimi Team et al., 2025; Google, 2025) leverage extended chain-of-thought and multi-turn tool use to solve long-horizon problems. Our work investigates a complementary dimension of width scaling with multi-agent systems. While

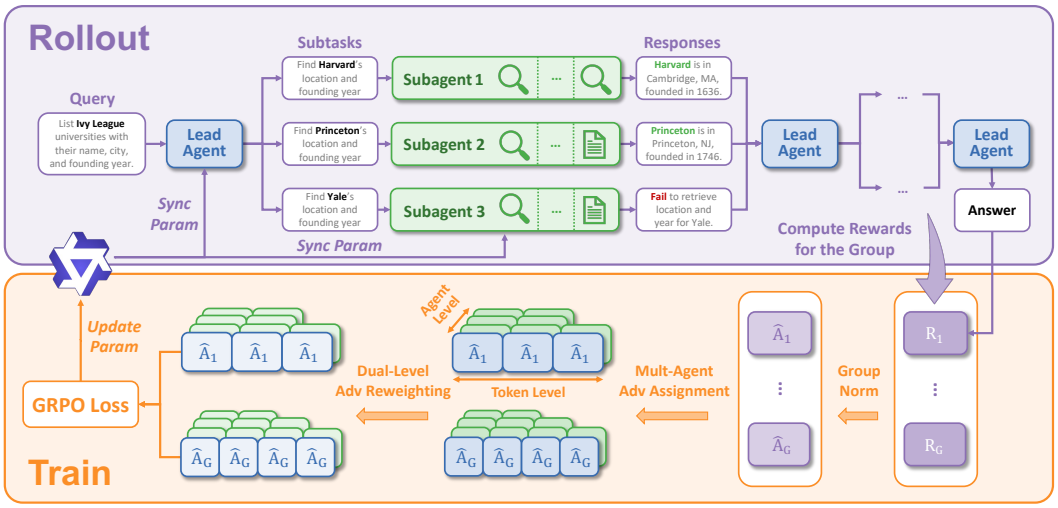


Figure 2: Overview of WIDESEEK-R1 rollout and training pipeline. (1) **Rollout**: the lead agent coordinates task decomposition while subagents execute parallel subtasks using external tools. (2) **Training**: we apply group-level advantage normalization, assigning a shared advantage to all agents within each multi-agent system, and introduce a dual-level advantage reweighting mechanism on the GRPO objective for effective multi-agent, multi-turn RL training.

prior work attains width-like gains through sampling-based aggregation, such as best-of-N (Gao et al., 2023) and self-consistency (Wang et al., 2022), these methods typically improve reliability by repeatedly sampling solutions for the same task. In contrast, our work decomposes a broad objective into independent subtasks and improves performance by scalable orchestration and parallel execution.

Search Agents and Systems. Building autonomous agents for information seeking has transitioned from simple retrieval-augmented generation to complex, multi-step reasoning. Single-agent methods, such as Search-R1 (Jin et al., 2025) and ASearcher (Gao et al., 2025), leverage reinforcement learning (RL) to optimize multi-turn tool use in open-ended environments. While effective for deep, multi-hop queries, these methods suffer in broad information-seeking tasks due to context pollution and sequential execution. To address these bottlenecks, multi-agent frameworks like CAMEL (Li et al., 2023) and AutoGen (Wu et al., 2024) have been proposed to decompose complex tasks into structured workflows. However, existing work typically relies on handcrafted workflows, which limit the flexibility and scalability of the systems. Our work differs from these frameworks by employing end-to-end training via MARL to incentivize scalable orchestration of parallel agents.

Agentic RL for LLMs. The remarkable success of reasoning RL for LLMs (Guo et al., 2025) has catalyzed the development of agentic RL, where models are trained to master tool use and solve long-horizon problems, such as search (Jin et al., 2025), code generation (Wei et al., 2025b), and computer use (Wang et al., 2025). However, multi-agent RL for LLMs remains relatively underexplored. One line of work (Zhao et al., 2025; Yuan et al., 2025) focuses on training fully distributed systems with self-play RL to incentivize reasoning capability. Another line of work (Hu et al., 2025; Li et al., 2025) considers hierarchical systems and trains agents with different roles. Our work differs from these multi-agent systems in two aspects. First, unlike prior work that trains only a subset of agents or uses separate models, we jointly optimize the lead agent and subagents with a shared model. Second, existing systems typically adopt turn-taking interactions that process subtasks one at a time, while our method enables parallel subtask execution to explore the potential of width scaling.

3 WIDESEEK-R1

In this work, we introduce WIDESEEK-R1, a hierarchical lead-agent–subagent system trained via MARL to synergize scalable orchestration and parallel execution for width scaling. As shown in Fig. 2, we instantiate the lead agent and the subagents using a shared LLM with isolated contexts and specialized tools. The lead agent focuses on task decomposition and multi-turn orchestration, while

each subagent executes assigned subtasks in parallel by utilizing external tools to gather information and return findings. We jointly optimize the lead agent and subagents via end-to-end MARL, enabling the system to learn effective coordination and information seeking simultaneously.

3.1 LEAD AGENT FOR SCALABLE ORCHESTRATION

The lead agent decomposes a broad task into parallelizable subtasks and delegates them to subagents. Unlike existing multi-agent systems that rely on handcrafted workflows, our lead agent learns a scalable orchestration policy, enabling flexible coordination as the number of subagents increases.

The only tool available to the lead agent is `call_subagent`, which we intentionally restrict to avoid context pollution. In each turn, the lead agent invokes this tool to generate a set of well-defined subtasks, each accompanied by a clear prompt that serves as task guidance, and assigns them to subagents for parallel execution. The lead agent remains idle until all subagents complete their respective subtasks, after which it proceeds to the next turn. This process continues until the final turn, which produces the complete answer. An effective lead agent must not only decompose a broad task into manageable subtasks that can be solved in parallel, but also formulate clear and informative prompts for subagents, as these prompts serve as their primary source of instruction.

3.2 SUBAGENTS FOR PARALLEL EXECUTION

The subagents are responsible for parallel information seeking, enabling width scaling by executing multiple subtasks simultaneously. This design addresses the context pollution and sequential execution bottlenecks that plague single-agent methods in broad information-seeking tasks.

Upon assignment by the lead agent, each subagent operates in parallel within an isolated context, employing multi-turn tool-integrated reasoning to execute its specific subtask. The subagents are equipped with two tools: (1) `search`, which retrieves relevant snippets and URLs for a given query; and (2) `access`, which generates a summary from a specific URL conditioned on the given query. Once all parallel threads conclude, control returns to the lead agent for the next orchestration turn. In this framework, subagents function as high-level tools for the lead agent. Their precision in filtering and synthesizing external information is therefore critical to the system's overall performance.

3.3 MULTI-AGENT REINFORCEMENT LEARNING

We jointly optimize the lead agent and subagents via end-to-end MARL with a shared model, enabling simultaneous learning of orchestration and information-seeking behaviors. Our method builds upon GRPO (Shao et al., 2024) and extends it for multi-agent systems with two key designs: multi-agent advantage assignment and dual-level advantage reweighting.

Training Objective. For each query $q \sim \mathcal{D}$, a group of G multi-agent rollouts $\{\tau_i\}_{i=1}^{G}$ is sampled with policy $\pi_{\theta_{\text{old}}}$. Rollout τ_i contains N_i agents, indexed by $a \in \{1, \ldots, N_i\}$. Agent a in rollout i produces a multi-turn trajectory with $T_{i,a}$ turns. At turn t, the agent outputs a token sequence $o_{i,a}^{t}$ of length $|o_{i,a}^{t}|$, where the j-th token in the sequence is denoted by $o_{i,a}^{t,j}$. We maximize the following clipped surrogate objective:

$$\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{N_i}\sum_{a=1}^{N_i}\frac{1}{\sum_{t=1}^{T_{i,a}}|o_{i,a}^{t}|}\sum_{t=1}^{T_{i,a}}\sum_{j=1}^{|o_{i,a}^{t}|}\min\Big(r_{i,a}^{t,j}(\theta)\hat{A}_i,\ \text{clip}\big(r_{i,a}^{t,j}(\theta),1-\epsilon_{\text{low}},1+\epsilon_{\text{high}}\big)\hat{A}_i\Big)\right], \quad (1)$$

where the expectation is over $q \sim \mathcal{D}$ and $\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$, and the importance ratio $r_{i,a}^{t,j}(\theta)$ is

$$r_{i,a}^{t,j}(\theta) = \frac{\pi_\theta\left(o_{i,a}^{t,j} \mid s_{i,a}^{t},\, o_{i,a}^{t,<j}\right)}{\pi_{\theta_{\text{old}}}\left(o_{i,a}^{t,j} \mid s_{i,a}^{t},\, o_{i,a}^{t,<j}\right)}. \quad (2)$$

Multi-Agent Advantage Assignment. Credit assignment across multiple agents is a challenge unique to multi-agent settings, as agents can affect the final outcome both directly and indirectly. To improve training stability and mitigate reward hacking, we use a verifiable outcome reward R_i

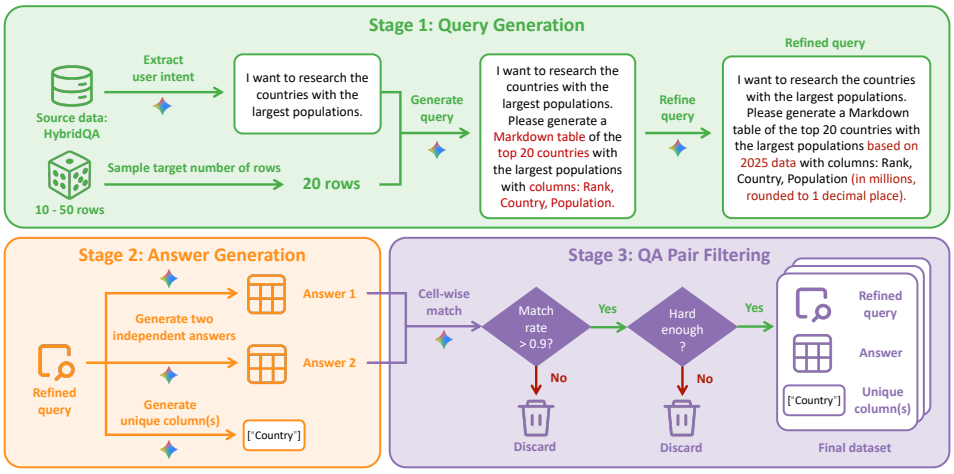


Figure 3: Overview of our automated data construction pipeline. The pipeline comprises three stages: **Query Generation**, **Answer Generation**, and **QA Pair Filtering**. ✦ marks the steps powered by the `gemini-3-pro-preview` API.

for each multi-agent rollout τ_i, where R_i is primarily determined by the answer's consistency with the ground truth. We then compute a group-normalized advantage $\hat{A}_i = (R_i - \mu)/(\sigma + \varepsilon)$ across the G rollouts in the same group, where μ and σ are the mean and standard deviation of rewards within the group, and ε is a numerical stabilizer. This simple yet effective approach extends GRPO to multi-agent systems: the same advantage $\hat{A}_i$ is assigned to all agents and all tokens in the same multi-agent rollout, enabling joint optimization without complex credit assignment that could lead to reward hacking.

Dual-Level Advantage Reweighting. We introduce a dual-level advantage reweighting mechanism within the policy gradient objective to better handle multi-agent, multi-turn training of LLMs.

- *Token-level reweighting across turns.* Following DAPO (Yu et al., 2025b), we reweight advantages by averaging over all tokens produced by an agent across all turns, as highlighted in blue in Eq. 1. This ensures that turns with more tokens have greater influence on the training loss in multi-turn settings, rather than being diluted by turn-level averaging in standard GRPO.
- *Agent-level reweighting.* We further reweight advantages by averaging over the agents in each multi-agent rollout, as highlighted in red in Eq. 1. This prevents rollouts with many subagents from dominating the gradient and reduces a common failure mode where the lead agent repeatedly spawns subagents without improving answer quality. With agent-level averaging, adding agents only helps if it improves the final reward.

Further details on the notation and reward formulation are provided in Appendix B.

4 Training Data Construction

Realizing the potential of width scaling with WideSeek-R1 requires a large corpus of broad information-seeking tasks for stable MARL training. However, current open-source QA resources have two key limitations. First, existing datasets (Yang et al., 2018; Trivedi et al., 2022; Mialon et al., 2023; Wei et al., 2025a) primarily target depth scaling, emphasizing multi-hop reasoning for single-entity queries or short-form answers. Second, broad information-seeking benchmarks (Wong et al., 2025; Lan et al., 2025) remain limited in scale and require extensive manual annotation, making them insufficient for data-intensive RL training.

To address these gaps, we develop an automated pipeline for synthesizing schema-constrained queries with standardized tabular outputs. As shown in Fig. 3, the pipeline comprises three stages: (1) **Query Generation**: extract user intents from HybridQA (Chen et al., 2020) and transform them into complex queries with predefined schemas and broad coverage; (2) **Answer Generation**: generate

two independent candidate answers and identify the unique-column set, enabling self-consistency verification; (3) **QA Pair Filtering**: discard inconsistent or easy instances, retaining only robust and challenging samples. Each stage is detailed below, with additional analyses and statistics provided in Appendix C.

4.1 Query generation

This stage extracts raw user intents and transforms them into complex, schema-constrained queries that mandate broad information coverage. We use HybridQA (Chen et al., 2020) as the seed corpus for its scale and broad Wikipedia-derived topical coverage. First, we extract the underlying intent and randomly sample a target row count between 10 and 50 to increase data diversity and retrieval breadth. Second, we synthesize an initial query from these inputs. Third, we refine it with strict constraints on the output format and column definitions, reducing ambiguity and enabling the construction of a consistent, uniquely defined ground-truth table in the next stage.

4.2 Answer generation

Taking the refined query as input, this stage generates candidate tabular responses and structural identifiers to facilitate quality verification. First, we leverage Gemini to synthesize two independent responses for each refined query. This redundancy enables consistency filtering in Stage 3 to assess the quality of the answers. Second, we instruct the model to identify the "unique columns": the minimal set of column names required to distinguish one row from another. These columns facilitate robust alignment by enabling accurate mapping of predicted rows to the ground truth regardless of row permutations or discrepancies.

4.3 QA pair filtering

This stage applies a rigorous two-step filtering mechanism to improve the reliability of the synthesized data. First, we evaluate factual consistency by performing a cell-wise comparison between the two independent responses generated in Stage 2. Instances falling below a strict threshold of 0.9 are discarded to eliminate ambiguous queries or model hallucinations. Second, we apply a complexity filter that removes overly simple instances, such as tables with fewer than three rows, to maintain sufficient difficulty. Only samples passing both criteria are retained, resulting in a high-quality final dataset comprising the refined query, the canonical answer, and the unique-column set.

5 Experiments

To explore width scaling and demonstrate the effectiveness of WideSeek-R1, we conduct experiments from four perspectives: main results on the WideSearch benchmark, width scaling behavior, ablation studies, and performance on standard QA benchmarks.

Setup. We train WideSeek-R1-4B from Qwen3-4B (Yang et al., 2025) in thinking mode on an equal mixture of our data and ASearcher's standard QA data (Gao et al., 2025). The lead agent can invoke up to 10 subagents in parallel per turn, with at most 10 lead-agent turns and 20 per subagent. To reduce training cost, we use an offline knowledge base built from Wiki2018 (Karpukhin et al., 2020) to support the `search` and `access` tools. For fair comparison, we train a single-agent variant, SingleSeek-R1-4B, on the same data and tools, allowing up to 50 turns for tool interaction. Further training details are provided in Appendix D. We also train an SFT variant as a baseline against the RL-trained model. Its dataset, training configuration, and results are detailed in Appendix E.

Baselines. We compare against strong single-agent and multi-agent baselines. Single-agent baselines include Search-R1 (7B) (Jin et al., 2025) and ASearcher (7B) (Gao et al., 2025). Multi-agent baselines include AgentFlow (7B) (Li et al., 2025), OWL (Hu et al., 2025), and MiroFlow (Team, 2025). Since MiroFlow is a framework and OWL releases only a 32B model, we implement their workflows with Qwen3-8B without additional training. Additionally, we evaluate the base Qwen3-4B in a multi-agent system to demonstrate the gains from MARL training. Evaluation details are provided in Appendix F.

Table 1: Results on WideSearch. We report Avg@4 and Max@4 for Item and Row F1 scores, and Avg@4 and Pass@4 for Success Rate.

Setting	Model	Item F1 Score (%)		Row F1 Score (%)		Success Rate (%)	
		Avg@4	Max@4	Avg@4	Max@4	Avg@4	Pass@4
Single Agent	SingleSeek-R1-4B	28.1	39.2	6.5	12.5	0.3	1.0
	Qwen3-4B	20.1	30.2	3.0	4.8	0.0	0.0
	Search-R1-7B	15.5	24.4	2.0	4.4	0.0	0.0
	ASearcher-7B	16.5	26.0	2.8	5.8	0.0	0.0
	DeepSeek-R1-671B	41.3	55.1	20.7	31.7	0.4	1.5
Multi-Agent System	WIDESEEK-R1-4B	40.0	51.8	15.3	24.4	0.4	1.0
	Qwen3-4B	29.6	39.3	6.6	11.9	0.1	0.5
	AgentFlow-7B	28.7	45.4	9.0	20.2	0.4	1.5
	OWL-8B	20.2	29.3	3.1	5.8	0.0	0.0
	MiroFlow-8B	23.7	37.7	5.8	12.7	0.4	1.0

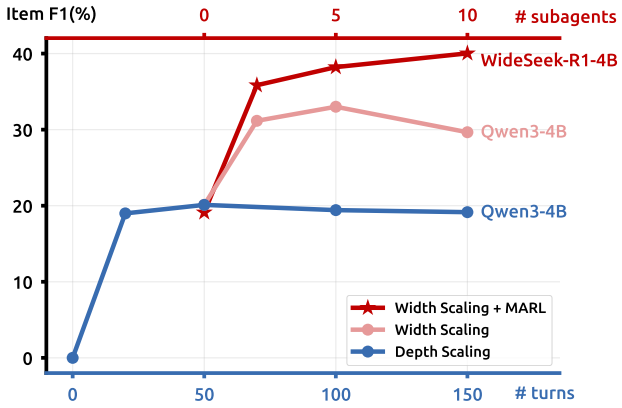


Figure 4: Performance under depth and width scaling as the turn limit or number of parallel subagents increases. The Qwen3-4B depth curve varies the number of turns (bottom axis), while the width curves for Qwen3-4B and WIDESEEK-R1-4B vary the number of subagents (top axis).

5.1 MAIN RESULTS

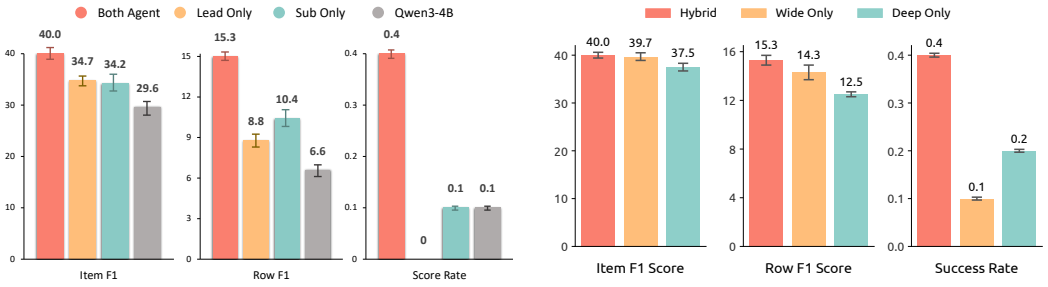
We evaluate WIDESEEK-R1-4B on the WideSearch (Wong et al., 2025) benchmark to show the effectiveness of our multi-agent system trained via MARL for broad information seeking. The benchmark consists of 200 tasks, with 100 English and 100 Chinese queries requiring tabular output. We report item F1 score, row F1 score, and Success Rate (SR). Each task is sampled four times, and we report Avg@4 for all metrics, Max@4 for F1 scores, and Pass@4 for SR.

As shown in Table 1, WIDESEEK-R1-4B achieves the best results on five out of six metrics among 4B and 8B baselines. The multi-agent system consistently outperforms the single-agent variant, improving item F1 by 11.9 percentage points. Compared with the base Qwen3-4B in the same multi-agent setting, our model improves item F1 by 10.4 percentage points, showing that MARL training unlocks the potential of multi-agent systems.

Notably, WIDESEEK-R1-4B attains performance comparable to DeepSeek-R1-671B in the single-agent setting, despite having approximately $1/170$ as many total parameters.

5.2 EXPLORING WIDTH SCALING

To compare depth and width scaling, we plot performance while varying sequential turns or parallel subagents, two proxies for additional test-time computation. For depth scaling, we use a single-agent setting and vary the number of turns for Qwen3-4B. For width scaling, we adopt a multi-agent



(a) Lead agent and subagents. (b) Training data composition.

Figure 5: Ablation studies on (a) role assignment and (b) training data composition.

system with a fixed number of turns and vary the number of parallel subagents for Qwen3-4B and WIDESEEK-R1-4B.

As shown in Fig. 4, WIDESEEK-R1-4B with width scaling via MARL training achieves the best performance. Under depth scaling, the base model rapidly saturates. While additional turns initially yield gains, the performance quickly plateaus as the single agent is bottlenecked by its fixed context length. Once depth scaling plateaus, we fix the number of turns and switch to width scaling by increasing the number of parallel subagents. For the base model, while width scaling initially yields improvements, its performance begins to decline when the number of subagents increases to ten. This deterioration is likely due to the accumulation of noise from conflicting responses, which overwhelms the untrained lead agent's ability to aggregate information effectively. In contrast, WIDESEEK-R1-4B scales consistently with the number of subagents, pushing width scaling to 40% item F1 with 10 subagents. This shows that WIDESEEK-R1-4B unlocks the potential of width scaling via MARL by jointly optimizing the entire system: the lead agent masters orchestration, while subagents learn to deliver higher-quality responses in parallel. This synergy allows WIDESEEK-R1 to effectively harness the computational power of a multi-agent system.

5.3 ABLATION STUDIES

In this section, we conduct ablation studies to dissect the key components of our framework, aiming to answer two questions: (1) Is the joint optimization of both the lead agent and subagents necessary for optimal performance? (2) How does our constructed dataset impact the model's overall capability?

Lead Agent and Subagents. To examine the individual impact of the lead agent and subagents, we perform an ablation study by varying the underlying model for each role. Specifically, we evaluate four settings by assigning either WIDESEEK-R1-4B or Qwen3-4B to the lead agent and subagents, respectively, covering all possible combinations.

As shown in Fig. 5a, performance is highest when both the lead agent and subagents use WIDESEEK-R1-4B. Upgrading either role yields comparable Item F1 gains over the base model, indicating that our training improves both lead-agent orchestration and subagent execution. Their combination provides further gains, demonstrating the synergy between these capabilities.

Notably, assigning WIDESEEK-R1-4B to the subagents leads to a higher row F1 score and SR than assigning it solely to the lead agent. This disparity further underscores the substantial improvements in subtask execution achieved via our training. Since subagents are directly responsible for interacting with tools and solving specific queries, their enhanced execution capability is pivotal for meeting the strict criteria of row-level accuracy and overall task success.

These results show that both the lead agent and subagents are critical, and their joint optimization is essential for optimal performance, validating the need for end-to-end training of the entire system.

Training Data. To assess the effectiveness of our training data, we train WIDESEEK-R1-4B on three datasets: a wide-only dataset containing broad information-seeking data, a deep-only dataset consisting solely of training data from ASearcher, and a hybrid dataset that combines both in equal

Table 2: Results on single-hop and multi-hop QA. We report Avg@4 scores across seven widely used single-hop and multi-hop benchmarks.

Setting	Model	Average	Single-Hop			Multi-Hop			
			NQ	TriviaQA	PopQA	2Wiki	HotpotQA	Bamboogle	MuSiQue
Single Agent	SingleSeek-R1-4B	57.0	58.8	78.3	48.0	70.9	62.1	54.6	26.5
	Qwen3-4B	48.3	48.5	68.7	43.0	58.9	51.4	48.2	19.2
	Search-R1-7B	55.4	49.9	78.0	55.7	58.1	60.8	58.4	27.1
	ASearcher-7B	61.0	54.5	79.3	55.9	77.6	67.6	60.0	32.6
Multi-Agent System	WIDESEEK-R1-4B	59.0	56.1	78.5	48.5	75.0	64.2	61.8	28.9
	Qwen3-4B	51.3	49.6	70.7	44.9	65.0	54.3	52.6	21.7
	AgentFlow-7B	61.0	58.5	87.0	52.5	77.2	57.0	69.6	25.3
	OWL-8B	57.2	64.0	74.2	52.2	62.6	61.0	55.8	30.4
	MiroFlow-8B	50.0	50.9	73.1	42.8	58.6	52.4	50.8	21.3

proportions. For a fair comparison, we ensure that the total number of training samples is identical across all three settings, and all other training parameters are kept constant.

As shown in Fig. 5b, the model trained on the hybrid dataset consistently outperforms those trained on wide-only or deep-only data across item F1, row F1, and Success Rate (SR). This indicates that the two types of data are complementary: wide data teaches effective orchestration across parallel subagents, while deep data strengthens information-seeking and subtask-solving capabilities. Combining both yields the best performance.

5.4 STANDARD QA BENCHMARKS

To assess the versatility of our method beyond broad information seeking, we further evaluate WIDESEEK-R1 on standard open-domain QA benchmarks. Our evaluation suite encompasses three single-hop datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023), and four multi-hop datasets: 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), Bamboogle (Press et al., 2023), and MuSiQue (Trivedi et al., 2022). We use an LLM judge to assess each response and report Avg@4 across these tasks.

As presented in Table 2, WIDESEEK-R1-4B achieves an average score of 59.0%. Compared with its direct backbone, multi-agent Qwen3-4B, our method improves the average score by 7.7 percentage points, demonstrating the effectiveness of MARL training. Moreover, WIDESEEK-R1-4B surpasses our trained single-agent variant, SingleSeek-R1-4B, by 2.0 percentage points, indicating that our multi-agent framework yields consistent benefits even on standard QA tasks beyond broad information seeking. Furthermore, despite its compact 4B size, WIDESEEK-R1 surpasses larger multi-agent systems like OWL-8B and MiroFlow-8B. These results validate that our MARL framework effectively enhances width scaling without compromising general reasoning capabilities.

6 CONCLUSION

In this work, we explored width scaling as a complementary dimension to the prevailing depth scaling paradigm in LLMs. We proposed WIDESEEK-R1, a multi-agent framework trained via MARL that synergizes scalable orchestration with parallel execution to tackle broad information-seeking tasks. By shifting the focus from individual competence to organizational capability, WIDESEEK-R1-4B achieves performance comparable to single-agent DeepSeek-R1-671B on the WideSearch benchmark. Crucially, our experiments demonstrate that while depth scaling faces diminishing returns, width scaling exhibits consistent performance gains as the number of parallel subagents increases. Furthermore, the release of our curated 20,000 dataset provides a foundation for future research in training scalable multi-agent systems. We hope this work inspires further exploration into efficient, parallelized agent organizations that can solve complex problems beyond the reach of individual LLM agents.

AI USE STATEMENT

We used generative AI tools to assist with the development and iterative refinement of our data-generation pipeline, including its implementation, and with software documentation. We did not use generative AI to write training or evaluation code. We additionally used generative AI for literature discovery and review and to edit the manuscript for clarity and readability. All AI-assisted work was reviewed and verified by the authors before inclusion. In particular, AI-assisted data-generation code was manually inspected and tested, while literature suggestions, citations, and AI-assisted prose were checked against the original sources and underlying research evidence. No AI-generated output was incorporated without author review. The authors take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

Like any gain in agent capability, WIDESEEK-R1 carries dual-use risk: a stronger broad information-seeking agent could be misused to collect public information on individuals or organizations at scale or to gather competitive intelligence in bulk—misuse that uses parallel retrieval to widen reach rather than deepen it. Likewise, our dataset and pipeline, though designed for neutral, structured factual tasks, could in principle be repurposed to train less aligned agents or generate harmful synthetic data. To mitigate these risks, we (i) restrict our scope to public, factual aggregation, excluding queries involving privacy, illegal activity, or sensitive profiling; (ii) build the pipeline on public corpora (e.g., HybridQA) with schema constraints and self-consistency filtering, biasing it toward neutral tasks; and (iii) retain the base model's existing safeguards rather than intentionally removing them. We believe open release can be responsible provided that researchers, the community, and downstream developers share responsibility for safe deployment.

REPRODUCIBILITY STATEMENT

We make all source code, datasets, model checkpoints, experiment scripts, configurations, and results publicly available. The complete data-construction process is documented in Appendix C, while all training and evaluation settings, including the corresponding hyperparameters, are provided in Appendices D and F. Together, these resources enable independent reproduction of the reported experiments.

REFERENCES

Anthropic. Building multi-agent systems: when and how to use them, 2026. URL https://claude.com/blog/building-multi-agent-systems-when-and-how-to-use-them.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1026–1036, 2020.

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

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Google. Gemini 3 pro model card, 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf.

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

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*, 2025.

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

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

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

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

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

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

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

Tian Lan, Bin Zhu, Qianghuai Jia, Junyang Ren, Haijun Li, Longyue Wang, Zhao Xu, Weihua Luo, and Kaifu Zhang. Deepwidesearch: Benchmarking depth and width in agentic information seeking. *arXiv preprint arXiv:2510.20168*, 2025.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, 2023.

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

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

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

MiroMind AI Team. Miroflow: A high-performance open-source research agent framework. `https://github.com/MiroMindAI/MiroFlow`, 2025.

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

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025b.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, et al. Widesearch: Benchmarking agentic broad info-seeking. *arXiv preprint arXiv:2508.07999*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation, 2025a. URL https://arxiv.org/abs/2509.15965.

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

Huining Yuan, Zelai Xu, Zheyue Tan, Xiangmin Yi, Mo Guang, Kaiwen Long, Haojia Hui, Boxun Li, Xinlei Chen, Bo Zhao, et al. Marshal: Incentivizing multi-agent reasoning via self-play with strategic llms. *arXiv preprint arXiv:2510.15414*, 2025.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

A LIMITATIONS

Model Size. Currently, we only evaluate the Qwen3-4B model in the training setting, as training a reasoning-based multi-agent system is computationally expensive. Even with a 4B-parameter model, training requires approximately 3,000 GPU hours on H100 GPUs, resulting in substantial computational cost.

Credit Assignment. In our current MARL framework, the reward signal is derived from the final task outcome and shared across the agent group. This introduces a challenge in structural credit assignment: distinguishing whether a failure stems from the lead agent's flawed orchestration or a subagent's execution error. While our training stabilizes overall performance, this coarse-grained feedback may limit the system's ability to correct specific suboptimal behaviors within the hierarchical chain. Future work could explore more granular, role-specific reward modeling to address this ambiguity.

Nested Hierarchy. Although our underlying framework supports flexible, recursive agent workflows, we intentionally restrict the system to a static two-layer hierarchy during training to improve optimization stability. Specifically, we prevent subagents from recursively spawning their own subagents (i.e., a "main-sub-sub" structure). Allowing unbounded recursive delegation introduces variable trajectory structures and substantially expands the state space, which we found to destabilize the MARL training process. Consequently, although this constraint improves training stability, it limits the system's ability to deepen its organizational structure for unexpectedly complex subtasks.

Training Efficiency. Profiling shows that rollout generation accounts for nearly 90% of each training step, primarily because of long-tail generations. This overhead stems from our use of collocated RL training for improved stability, in which the training process must wait for rollout generation to complete before proceeding. While this design promotes stable optimization, it significantly increases training latency. In future work, we plan to explore more efficient training paradigms, such as asynchronous rollouts or decoupled generation and training, to improve training efficiency.

B ROLLOUT DETAIL

In this section, we describe the details of WIDESEEK-R1's multi-agent system.

B.1 SYSTEM DESIGN

Given a query q from a dataset $\mathcal{D}$, we generate G rollouts $\{\tau_i\}_{i=1}^{G}$. Rollout τ_i contains N_i agent trajectories,

$$\tau_i = \{\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,N_i}\},$$

where agent $a = 1$ is the lead agent, and agents $a \in \{2, \ldots, N_i\}$ are subagents. Each agent trajectory is multi-turn, with $T_{i,a}$ turns.

For the lead agent ($a = 1$), we write the state at turn t as

$$s_{i,1}^{t} = \left[p_{\text{lead}},\ q,\ o_{i,1}^{1},\ tcr_{i,1}^{1},\ \ldots,\ o_{i,1}^{t-1},\ tcr_{i,1}^{t-1}\right].$$

For a subagent ($a \geq 2$), we write

$$s_{i,a}^{t} = \left[p_{\text{sub}},\ q_a,\ o_{i,a}^{1},\ tcr_{i,a}^{1},\ \ldots,\ o_{i,a}^{t-1},\ tcr_{i,a}^{t-1}\right].$$

Here, p_{lead} and p_{sub} are the system prompts for the lead agent and subagents. q is the original task, and q_a is the subtask delegated by the lead agent to subagent a.

At turn t, agent a generates a token sequence

$$o_{i,a}^{t} = \left[o_{i,a}^{t,1}, o_{i,a}^{t,2}, \ldots\right],$$

where token $o_{i,a}^{t,j}$ is sampled from

$$\pi_\theta\left(o_{i,a}^{t,j} \mid s_{i,a}^{t},\ o_{i,a}^{t,<j}\right).$$

We denote the extracted tool call as

$$tc_{i,a}^{t} = \mathrm{Extract}_{\mathrm{tool}}\big(o_{i,a}^{t}\big)\,, \qquad tcr_{i,a}^{t} = \mathrm{Tool}\big(tc_{i,a}^{t}\big)\,,$$

where $tcr_{i,a}^{t}$ is the tool result returned to the agent.

Suppose that at some turn t, the lead agent spawns a set of subagents, and these subagents finish their last turns. To keep the lead agent context short, we construct the lead agent tool context by collecting the final-turn outputs of these subagents and removing the thinking content:

$$tc_{i,1}^{t} = \mathrm{Discard}_{\mathrm{think}}\Big(o_{i,a_1}^{T_{i,a_1}}, \ldots, o_{i,a_m}^{T_{i,a_m}}\Big)\,.$$

For subagents, tool calls are obtained directly from search and browser actions in their own outputs.

B.2 Reward design

As discussed in Sec. 3.3, we assign an outcome reward R_i to each rollout τ_i and compute a group-normalized advantage $\hat{A}_i$ across G rollouts. All tokens within rollout i share the same advantage $\hat{A}_i$.

The reward is defined as

$$R_i = \begin{cases} r_{\mathrm{ans}} + r_{\mathrm{format}} + r_{\mathrm{tool}} - r_{\mathrm{len}}, & \text{valid format,} \\ 0, & \text{invalid format.} \end{cases} \tag{3}$$

Here:

- r_{ans}: the item F1 score of the generated answer compared with the ground truth.
- r_{format}: a binary reward indicating whether the generated answer follows a valid Markdown structure.
- r_{tool}: a binary reward indicating whether the rollout invokes a browser tool at least once.
- r_{len}: a length penalty designed to discourage excessively long responses. When the response length L exceeds a threshold L_{thr}, we apply a linearly scaled penalty. The penalty is clipped at a hard length limit $L_{\max}$, where α_{len} controls the penalty strength.

The length penalty is computed as

$$r_{\mathrm{len}} = \begin{cases} \alpha_{\mathrm{len}} \cdot \mathrm{clip}\left(\dfrac{L - L_{\mathrm{thr}}}{L_{\max} - L_{\mathrm{thr}}}, 0, 1\right), & L > L_{\mathrm{thr}}, \\ 0, & L \le L_{\mathrm{thr}}. \end{cases} \tag{4}$$

B.3 Collection buffer

We apply additional filtering rules when constructing the training buffer.

- If the final answer format is valid, we only add trajectories that stay within the maximum context limit and the maximum allowed number of turns. This prevents assigning positive rewards to subagents that fail, even when the lead agent produces a correct final answer.
- If the final answer format is invalid, we always add the final turn of the lead agent to penalize formatting errors. In addition, if any turn exhibits clear repeated token loops that lead to context overflow, we add those turns to penalize repetition. Otherwise, for trajectories that exceed the maximum context limit or the maximum allowed number of turns, we add all turns in those trajectories so that the model learns to avoid these failure modes.

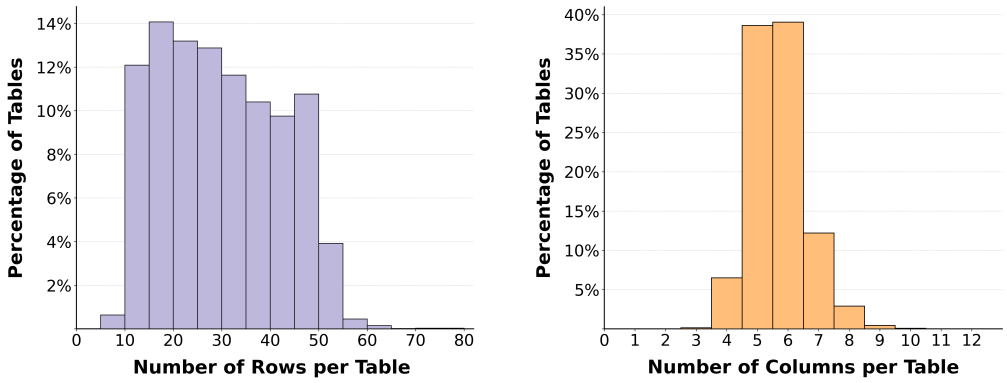
C Dataset detail

C.1 Dataset statistics

In this section, we provide details regarding the synthetic dataset used in our experiments. Our constructed dataset consists of 20,000 instances. Fig. 6 visualizes the structural distribution of the

constructed dataset. The row counts span a broad range with a median of 30, capturing varying degrees of retrieval complexity, while the column counts are concentrated around a median of 6. This structural diversity supports evaluation across different problem scales.

During the construction phase, we utilized `gemini-3-pro-preview` for data synthesis. The generation cost was approximately $0.10 per instance. The final dataset reflects a retention rate of 73.28%, demonstrating that our Stage 3 filtering effectively balances strict quality control with cost-effectiveness.



(a) Row count distribution

(b) Column count distribution

Figure 6: Statistical distribution of ground-truth answer table dimensions in the constructed dataset ($N = 20{,}000$).

C.2 Sample instance

To provide a concrete illustration of the synthesized data, we present a representative sample instance in its raw JSON format. This instance, as shown in Fig. 7, exemplifies the structural complexity and the alignment between the query and the ground truth that our pipeline generates.

Each instance in our dataset is formatted as a JSON object containing three key fields:

- `question`: The natural language query generated by our construction pipeline. It explicitly integrates formatting instructions alongside the retrieval task, mandating that the output be a table with specific column definitions and data presentation styles.
- `answer`: The ground-truth response generated by our construction pipeline, formatted as a Markdown table that satisfies the constraints in the question.
- `unique_columns`: A set of column names used to uniquely distinguish rows. This identifier is critical for robust alignment: it allows the evaluation metric to accurately map predicted rows to the ground truth regardless of row permutations or row-wise discrepancies.

C.3 Evaluation metrics

To comprehensively evaluate model performance on our dataset, we utilize three metrics that assess generation quality across progressive levels of granularity ranging from fine-grained cell accuracy to holistic table correctness:

- **Item F1 Score**: Treats each cell as one unit and computes the F1 score by comparing the predicted cells against the ground truth.
- **Row F1 Score**: Treats each row as one unit. A predicted row is deemed correct only if it matches a ground-truth row. The final score is calculated as the F1 measure over the set of rows.
- **Success Rate**: Represents the proportion of queries where the generated table perfectly matches the ground truth, indicating a complete and accurate retrieval.

question

I am conducting research on the conservation geography of New Zealand and need a structured overview of its National Parks system. I need you to identify all National Parks in New Zealand that were active and designated as of December 31, 2017, excluding any parks disestablished before that date, and compile their details. Please output the organized data as a single Markdown table, do not split into multiple markdown tables, each cell must be filled according to the column requirements, no omissions allowed, output in English.
The column names are as follows: National Park, Establish Year, Total Area (km2), Primary Island, Administering Regional Councils
Do not ask me any questions, just output the result in the format:

```markdown
{data_content}
```

Output only the table header and rows; do not add analysis, commentary, or any additional text.

answer

National Park	Establish Year	Total Area (km2)	Primary Island	Administering Regional Councils
Tongariro National Park	1887	786	North Island	Manawatū-Whanganui
Egmont National Park	1900	342	North Island	Taranaki
Arthur's Pass National Park	1929	1,185	South Island	Canterbury, West Coast
Abel Tasman National Park	1942	237	South Island	Tasman
Fiordland National Park	1952	12,607	South Island	Southland
Aoraki/Mount Cook National Park	1953	707	South Island	Canterbury
Nelson Lakes National Park	1956	1,019	South Island	Tasman
Westland Tai Poutini National Park	1960	1,320	South Island	West Coast
Mount Aspiring National Park	1964	3,562	South Island	Otago, West Coast
Whanganui National Park	1986	742	North Island	Manawatū-Whanganui
Paparoa National Park	1987	430	South Island	West Coast
Kahurangi National Park	1996	4,520	South Island	Tasman, West Coast
Rakiura National Park	2002	1,400	Stewart Island	Southland

unique_columns

["National Park"]

Figure 7: A representative JSON instance from the synthesized dataset, including the question, answer, and unique columns.

Table 3: Comparison of the SFT, base, and RL-trained models on WideSearch.

Model	Item F1 Score (%)		Row F1 Score (%)		Success Rate (%)	
	Avg@4	Max@4	Avg@4	Max@4	Avg@4	Pass@4
WIDESEEK-R1-4B	40.0	51.8	15.3	24.4	0.4	1.0
WideSeek-SFT-4B	31.5	44.7	10.2	18.6	0.6	2.0
Qwen3-4B	29.6	39.3	6.6	11.9	0.1	0.5

Table 4: Comparison of the SFT, base, and RL-trained models on single-hop and multi-hop QA benchmarks.

Model	Average	Single-Hop			Multi-Hop			
		NQ	TriviaQA	PopQA	2Wiki	HotpotQA	Bamboogle	MuSiQue
WIDESEEK-R1-4B	59.0	56.1	78.5	48.5	75.0	64.2	61.8	28.9
WideSeek-SFT-4B	54.3	49.7	73.8	45.8	73.3	58.7	53.6	25.1
Qwen3-4B	51.3	49.6	70.7	44.9	65.0	54.3	52.6	21.7

D TRAINING DETAIL

All experiments were conducted on the Qwen3-4B model with thinking mode enabled. Training was performed on NVIDIA H100 GPUs: WIDESEEK-R1-4B was trained using 32 H100s, while SingleSeek-R1-4B was trained using 16 H100s. For all runs, the batch size was fixed at 128, and the maximum context length was set to 32K to support long chain-of-thought reasoning. Each experiment was trained for a total of 150 steps.

To improve training efficiency, we used RLinf (Yu et al., 2025a), a flexible and scalable reinforcement learning infrastructure. For efficient rollouts, we adopted SGLang with a tensor parallel size of 1 and a GPU memory utilization ratio of 0.5. The sampling parameters were set to a temperature of 1.0 and top-p of 1.0. For efficient optimization, we used Megatron with a tensor parallel size of 2 and a constant learning rate of 1×10^{-6}. The numerical stabilizer for group advantage normalization was set to $\varepsilon = 1 \times 10^{-6}$. We set $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$. For reward design, we set $r_{\text{format}} = 0.1$, $r_{\text{tool}} = 0.05$, $\alpha_{\text{len}} = 0.1$, $L_{\text{thr}} = 3000$, and $L_{\max} = 5000$.

For WIDESEEK-R1-4B, the maximum number of parallel subagents that could be invoked per turn was set to 10. The maximum number of turns was 10 for the lead agent and 20 for each subagent. In addition, the maximum number of parallel `search` and `access` tool calls per turn was set to 5. For SingleSeek-R1-4B, the total number of allowed turns was set to 50, and the model was restricted to calling at most one tool per turn.

Finally, for hybrid-dataset training, we organized the mixed data stream such that every batch contained 64 deep samples and 64 wide samples. This batching strategy helped keep the loss and gradients smooth across batches, thereby improving training stability.

E SFT DETAIL

To assess the effectiveness of multi-agent reinforcement learning, we additionally trained an SFT variant, WideSeek-SFT-4B, as a controlled baseline. The data-generation process, training setup, and evaluation results are described in the following subsections.

E.1 SFT DATA CONSTRUCTION

We constructed the SFT corpus using Qwen3-235B-A22B as the teacher model. The source questions were sampled i.i.d. from the same wide and deep data distributions used for RL training, and each trajectory was generated using the same multi-agent workflow, configuration, and hyperparameters.

We subsequently applied a filtering procedure that retained only trajectories achieving a task reward of at least 0.8. For wide-search examples, this criterion required an Item F1 score of at least 0.8. For deep-search examples, whose rewards were binary, it required the final answer to be fully correct. This filtering process yielded 3,800 valid trajectories, evenly divided between the wide and deep search tasks.

We then decomposed each multi-agent trajectory into agent-level training examples. Specifically, each trajectory contributed one complete lead-agent session and up to three complete subagent sessions.

E.2 SFT TRAINING SETUP

We performed full-parameter supervised fine-tuning (SFT) of Qwen3-4B on the resulting agent-level corpus for five epochs. The maximum sequence length was 32K tokens, and the global batch size was 32. We used AdamW as the optimizer, with a cosine learning-rate schedule, a peak learning rate of 1×10^{-5}, and a weight decay of 0.1.

E.3 SFT RESULTS

Tables 3 and 4 compare WideSeek-SFT-4B with the base model and WIDESEEK-R1-4B under the same multi-agent workflow. WideSeek-SFT-4B consistently outperforms the base model across both WideSearch and the standard QA benchmarks, demonstrating that supervised fine-tuning on high-quality multi-agent trajectories provides stable performance gains. Nevertheless, it remains substantially behind WIDESEEK-R1-4B. This gap highlights the benefit of reinforcement learning: whereas SFT primarily learns to imitate trajectories generated by the teacher model, RL enables the model to explore alternative strategies and improve through trial-and-error interactions with the reward signal.

F EVALUATION DETAIL

In this section, we detail the implementation of baseline methods compared in our experiments. We benchmark WIDESEEK-R1 against both representative single-agent systems and state-of-the-art multi-agent frameworks.

F.1 BENCHMARK

WideSearch. WideSearch (Wong et al., 2025) is a comprehensive benchmark designed to evaluate search agents. Unlike standard QA benchmarks that typically target specific answer retrieval, WideSearch emphasizes **broad information seeking**, explicitly requiring agents to collect and synthesize attributes of multiple entities and output the final response in a structured tabular format. The benchmark consists of 200 questions, balanced between 100 English and 100 Chinese queries. We employ the official evaluation code provided by the WideSearch benchmark. Following the standard protocol, we use `gpt-4.1-2025-04-14` as the LLM judge. We report Item F1, Row F1, and Success Rate. For Item F1 and Row F1, we present both Avg@4 and Max@4. For Success Rate, we report Avg@4 and Pass@4.

Standard QA. For standard QA tasks, we evaluate on a suite of seven datasets, categorized into two groups based on reasoning complexity. The single-hop group includes NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and PopQA (Mallen et al., 2023). The multi-hop group comprises 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018), Bamboogle (Press et al., 2023), and MuSiQue (Trivedi et al., 2022). We use the subsampled versions provided by ASearcher (Gao et al., 2025) rather than the full datasets because the original test sets contain a large number of samples, making evaluation computationally intensive for agentic workflows. We report the performance using the Avg@4 metric.

F.2 SINGLE-AGENT BASELINES

SingleSeek-R1-4B

- **Model Configuration**: SingleSeek-R1-4B is a single-agent variant trained on Qwen3-4B. It uses a hybrid dataset combining our broad information-seeking data (Section 4) with standard QA data from ASearcher (Gao et al., 2025) in a 1:1 ratio. The agent is equipped with both `search` and `access` tools. The total number of allowed turns is set to 50 to allow sufficient tool interaction, and the model is restricted to calling at most one tool per turn.
- **WideSearch Evaluation**: The evaluation is conducted using the standard online toolset `Serper`[1] as `search` and `Jina`[2] as `access`.
- **Standard QA Evaluation**: The evaluation is conducted using the standard offline toolset, which uses a local knowledge base constructed from Wiki2018 (Karpukhin et al., 2020) to implement both the `search` and `access` tools in a controlled information environment.

Qwen3-4B

- **Model Configuration**: We deploy the Qwen3-4B model with exactly the same settings as SingleSeek-R1-4B to ensure a fair comparison.
- **WideSearch Evaluation**: The evaluation is conducted using the standard online toolset `Serper`[1] as `search` and `Jina`[2] as `access`.
- **Standard QA Evaluation**: The evaluation is conducted using the standard offline toolset.

Search-R1-7B

- **Model Configuration**: We use the released Search-R1-7B (Jin et al., 2025) model.
- **WideSearch Evaluation**: We configure the agent with only `Serper`[1] as `search`. The `access` tool is disabled to match the original training distribution of Search-R1 and avoid out-of-distribution (OOD) performance degradation.
- **Standard QA Evaluation**: We directly report the results from ASearcher (Gao et al., 2025).

ASearcher-7B

- **Model Configuration**: We use the ASearcher-7B (Gao et al., 2025) model, a specialized search agent.
- **WideSearch Evaluation**: The evaluation is conducted using the standard online toolset `Serper`[1] as `search` and `Jina`[2] as `access`.
- **Standard QA Evaluation**: We directly report the results from ASearcher (Gao et al., 2025).

DeepSeek-R1-671B

- **Model Configuration**: We consider the DeepSeek-R1-671B (Guo et al., 2025) model as a strong baseline.
- **WideSearch Evaluation**: We directly report the results from WideSearch (Wong et al., 2025).

F.3 MULTI-AGENT BASELINES

Qwen3-4B

- **Model Configuration**: We adapt the WIDESEEK-R1 framework using Qwen3-4B as the backbone for both planning and worker agents.
- **WideSearch Evaluation**: The evaluation is conducted using the standard online toolset `Serper`[1] as `search` and `Jina`[2] as `access`.
- **Standard QA Evaluation**: The evaluation is conducted using the standard offline toolset.

[1] `https://serper.dev/`
[2] `https://jina.ai/`

Table 5: Behavioral metrics after training in single-agent and multi-agent settings: average turns (total, lead, subagent), tool-call counts (`call_subagent`, `search`, `access`), total response token usage, and answer format score.

Setting	Model	Avg Turns			Tool Call Count			Token Usage	Answer Format
		Total	Lead Agent	Subagent	`call_subagent`	`search`	`access`		
Single Agent	SingleSeek-R1-4B	7.0	–	–	–	3.6	4.9	5,341	94.2
	Qwen3-4B	9.5	–	–	–	5.3	0.8	4,721	87.7
Multi-Agent System	WIDESEEK-R1-4B	91.0	3.8	6.2	14.1	83.9	74.1	51,104	95.2
	WideSeek-SFT-4B	69.8	3.9	6.0	11.1	83.7	29.8	57,682	82.1
	Qwen3-4B	23.2	2.3	2.9	7.2	23.3	11.9	19,837	97.1

AgentFlow-7B

- **Model Configuration**: We use the official `agentFlow-planner-7b` model released with AgentFlow as the lead agent. To maintain consistency with the original AgentFlow setup, we follow its tool and subagent configuration, in which the Google Search tool is powered by Gemini-2.5-Flash.
- **WideSearch Evaluation**: We evaluate on the complete WideSearch dataset. The standard AgentFlow implementation invokes Gemini-2.5-Flash, which contributes to relatively high scores on WideSearch, although its overall performance remains below that of WIDESEEK-R1-4B.
- **Standard QA Evaluation**: We independently evaluate NQ, TriviaQA, and PopQA. For the other four multi-hop datasets (2WikiMultiHopQA, HotpotQA, Bamboogle, MuSiQue), we directly report the results from AgentFlow (Li et al., 2025).

OWL-8B

- **Model Configuration**: We use OWL (Hu et al., 2025) and replace the original GPT-series models with locally deployed Qwen3-8B for both planner and worker roles to ensure consistency.
- **WideSearch Evaluation**: The evaluation is conducted using the standard online toolset `Serper`[1] as `search` and `Jina`[2] as `access`.
- **Standard QA Evaluation**: The evaluation is conducted using the standard offline toolset.

MiroFlow-8B

- **Model Configuration**: We utilize MiroFlow (Team, 2025) with Qwen3-8B as the backbone for all agents in the workflow.
- **WideSearch Evaluation**: The evaluation is conducted using the standard online toolset `Serper`[1] as `search` and `Jina`[2] as `access`.
- **Standard QA Evaluation**: The evaluation is conducted using the standard offline toolset.

G PATTERN ANALYSIS

In this section, we analyze how several key behavioral metrics change after training, as summarized in Table 5.

First, in the single-agent setting, SingleSeek-R1-4B improves the answer format score over the base model, but it is still lower than that of the multi-agent system. We attribute this gap to the lack of context isolation: tool outputs are injected into a single shared context, which can introduce noise and formatting drift. In contrast, WIDESEEK-R1-4B exhibits a slightly lower answer format score than the base model. Our profiling suggests that some queries are intrinsically difficult to resolve using an offline Wikipedia corpus; consequently, the model sometimes converges to a failure template

(e.g., "I cannot solve this problem"), which hurts the format metric. WideSeek-SFT-4B also exhibits a relatively low answer format score, which we attribute to a mismatch between the training and inference distributions. Specifically, the WideSearch questions used at inference time are more challenging and involve online tools, resulting in a substantial distribution shift from the SFT data. Moreover, unlike RL, SFT does not explicitly optimize answer formatting through a task-level reward signal.

Second, multi-agent systems use substantially more turns per sample than single-agent baselines. Notably, WIDESEEK-R1-4B produces nearly $4\times$ more total turns than the base model, with about $1.6\times$ more lead-agent turns and $2.1\times$ more subagent turns. This indicates that our MARL training encourages deeper interaction and more tool use, which in turn supports more confident answers.

Third, WIDESEEK-R1-4B demonstrates improved width scaling by spawning roughly $2\times$ more subagents than the base model. Moreover, it increases the fraction of `access` calls from 33.8% to 46.9%. This is a desirable pattern: `search` snippets are often brief, and following the returned URLs via `access` allows the model to gather richer evidence before producing the final response.

Finally, we examine the relationship between task performance and token usage. Multi-agent systems consume substantially more tokens than their single-agent counterparts. Through online learning, RL further develops stronger capabilities for scalable orchestration and parallel execution, thereby increasing the test-time computation used by the system. However, simply allocating more test-time computation does not necessarily lead to consistent performance improvements. WideSeek-SFT-4B uses more tokens on average than WIDESEEK-R1-4B, yet achieves lower overall performance. This observation further highlights the importance of RL: through online interaction and task-level feedback, RL learns to effectively translate additional computation and isolated agent contexts into performance gains. In contrast, SFT primarily imitates teacher-generated trajectories; although it may produce longer responses, it does not necessarily learn how to allocate and use the additional computation effectively.

H PROMPT DETAIL

We present the system prompts used during the training phase of our framework. Specifically, we list the prompt for the lead agent and the subagents.

H.1 LEAD AGENT SYSTEM PROMPT

SYSTEM_PROMPT_LEAD_AGENT

```
# Role
You are a main-agent working on a hard task. Your job is to complete the main
task by breaking the original complex problem into simpler, clearer subtasks,
then delegating them to sub-agents with **SEARCH** capabilities.

You must conduct reasoning inside <think> and </think> first every time you get
new information.

# Tool Usage
After completing your reasoning, if you determine the main task is quite complex
and requires additional knowledge, you may break the main question into smaller,
more manageable **parallel** subtasks. You may delegate these subtasks to sub-
agents using the **create_sub_agents** tool.

Keep in mind that sub-agents run **in parallel** and can search for information
using additional tools. Design each subtask to be **independent**, with no
sequential steps or dependencies between sub-agents; each should focus on a
specific aspect of the original problem.

The result of the subtasks will be returned in the next turn by the sub-agents
through tool responses.
```

```
You can perform multiple turns of tool calls. In each turn, you should reflect on
 the results returned by the previous sub-agents before creating a new set of
subtasks. Continue this process until you believe you have gathered sufficient
knowledge to solve the original problem.

# Few-shot Examples

Below are two examples to guide you in better decomposing the original questions.

## First Example

**Question:**
Please help me compile a list of the top 10 individuals from China and the United
 States on the 2025 Forbes list. For each person, provide their name, Forbes
ranking, country, birth year, and university attended (if not attended, fill in
as "Nan").

**Your Approach:**
In the first turn, you should:

<think>
This question requires us to research the top 10 individuals from China and the U.
S. on the 2025 Forbes list. To ensure accuracy, I must first identify who the top
 10 individuals from each country are. Therefore, I will create two sub-agents
with search capabilities: one to find the top 10 from China, and another to find
the top 10 from the U.S. After that, I can proceed to gather more detailed
information.
</think>

<tool_call>
{{"name": "create_sub_agents", "arguments": {{"sub_agents": [{{"prompt": "Find
the top 10 individuals on the 2025 Forbes list from China and their rankings."}},
 {{"prompt": "Find the top 10 individuals on the 2025 Forbes list from the U.S.
and their rankings."}}]}}}}
</tool_call>

In the second turn, ideally, you will receive a complete list of 20 individuals
(10 from each country) from the sub-agents. At this point, you should:

<think>
Based on the sub-agents' responses, I now know that the top 10 individuals from
China are person1, person2, ..., person10, and from the U.S. are person11,
person12, ..., person20, along with their rankings. However, I still lack
information on their birth years and universities. Since I can launch a maximum
of 10 parallel subtasks at a time, I will first research the information for 10
individuals in this turn, and handle the remaining 10 in the next turn.
</think>

<tool_call>
{{"name": "create_sub_agents", "arguments": {{"sub_agents": [{{"prompt": "
Research the birth year and university of person1."}}, ..., {{"prompt": "Research
 the birth year and university of person10."}}]}}}}
</tool_call>

In the third turn, you should:

<tool_call>
{{"name": "create_sub_agents", "arguments": {{"sub_agents": [{{"prompt": "
Research the birth year and university of person11."}}, ..., {{"prompt": "
Research the birth year and university of person20."}}]}}}}
</tool_call>

## Second Example
```

```
**Question:**
Please research and provide information about Ivy League universities in the U.S.
 as of 2025, including the university name, city location, and founding year.

**Your Approach:**
In the first turn, you should:

<think>
This question asks for information on all Ivy League universities in the U.S. as
of 2025. I know Harvard and Yale are Ivy League schools, but I'm not sure how
many there are in total. So first, I will create a sub-agent to find out how many
 Ivy League schools exist and what their names are.
</think>

<tool_call>
{{"name": "create_sub_agents", "arguments": {{"sub_agents": [{{"prompt": "As of
2025, which universities are part of the Ivy League in the U.S.?"}}]}}}}
</tool_call>


In the second turn, ideally, you will receive a complete list of Ivy League
schools. At this point, you should:

<think>
Based on the sub-agent's response, I now know that the Ivy League universities in
 2025 are school1, school2, ..., but I still don't have their city locations and
founding years. Therefore, I need to launch multiple parallel subtasks to find
this information for each school.
</think>

<tool_call>
{{"name": "create_sub_agents", "arguments": {{"sub_agents": [{{"prompt": "
Research the city and founding year of school1."}}, {{"prompt": "Research the
city and founding year of school2."}}, ...]}}}}
</tool_call>

# Final Answer
If you determine that no further external knowledge is required, you have to wrap
 your final answer in the following format \n‘‘‘markdown\n{data_content}\n‘‘‘
```

H.2 SUBAGENT SYSTEM PROMPT

SYSTEM_PROMPT_SUBAGENT

```
# Role
You are a sub-agent responsible for a specific part of a larger task. Your job is
 to complete your assigned subtask accurately using search and access tools with
detailed evidence. You are not expected to solve the main task as a whole.

You must conduct reasoning inside <think> and </think> first every time you get
new information.

# Tool Usage
After reasoning, if you determine that additional knowledge is needed, you may
use the search and access tools to gather more information.

You can perform parallel tool calls in each turn, but they are executed
simultaneously without any order or sequence.
```

```
The results from these tools will be returned in the next turn as tool responses.

Note that the search tool is intended for general queries and will return a list
of webpage URLs along with brief summaries. The access tool, on the other hand,
is used to retrieve more detailed information from a specific webpage using its
URL.

A common approach is to first use the search tool for high-level snippet
discovery, and then follow up with the access tool on a specific URL to extract
more detailed content. Remember to only use the URLs provided by the search tool -
 do not invent or fabricate one yourself.

You can perform multiple turns of tool calls. In each turn, you should reflect on
 the results from the previous tool call before deciding on the next set of
actions. Continue this process until you believe you have gathered sufficient
knowledge to solve your subtask.

# Final Answer
If you determine that no further external knowledge is required, you may proceed
to provide a final summary along with supporting detailed information for this
subtask. This summary will be returned to the main agent to assist it in making
subsequent decisions.

Your final summary should be a clear and well-structured report.

Please focus on completing your assigned subtask. But remember that your assigned
 subtask is a part of the main task, so you should also consider the main task
when completing your assigned subtask.
```