# SearchSwarm: Towards Delegation Intelligence in Agentic LLMs for Long-Horizon Deep Research

Anonymous ACL submission

## Abstract

Large language models are increasingly expected to handle complex, long-horizon real-world tasks whose context demands can grow without bound, yet model context windows remain inherently finite. Recent work explores a paradigm where a main agent decomposes tasks and dispatches subtasks to subagents, which execute and return only summarized results, conserving the main agent's context budget. However, performing this well requires *delegation intelligence*: the ability to decompose complex tasks, determine when and what to delegate, and integrate returned results into the ongoing workflow. Training data for this capability is scarce in naturally occurring text, and to our knowledge, no open-source work has addressed how to synthesize such data or train models to acquire this skill. To bridge this gap, we present a preliminary exploration targeting deep research, a representative long-horizon agent task. Specifically, we design a harness that guides the model toward high-quality task decomposition and delegation, while constraining subagents to return results properly to support the main agent's workflow. The harness-guided trajectories naturally encode correct delegation decisions, which we use as supervised fine-tuning data to internalize delegation intelligence into model weights. Our resulting model, SearchSwarm-30B-A3B, achieves 68.1 on BrowseComp and 73.3 on BrowseComp-zh, the best results among all models of comparable scale. We will release our harness, model weights, and training data to facilitate future research. Our code is available at `https://anonymous.4open.science/r/SearchSwarm-0385/`.

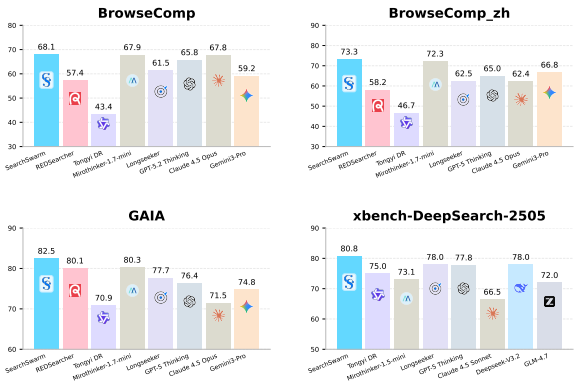



Figure 1: Performance comparison of SearchSwarm against lightweight models of comparable scale and larger closed-source/open-source models on four benchmarks. SearchSwarm achieves the best results among all models at the same scale and remains competitive with models over 10× larger.

## 1 Introduction

Large language models are increasingly deployed as agents for complex, long-horizon real-world tasks whose context demands can grow without bound (Jimenez et al., 2024; Yang et al., 2026), yet model context windows remain inherently finite. This fundamental tension necessitates context management strategies that selectively retain or condense information to fit within limited capacity. Early approaches include summarizing interaction history after exceeding a length threshold, or retaining only a portion of tool outputs, among others (Liu et al., 2025; Zeng et al., 2026). However, these methods are fundamentally *passive*: they lack prior planning, waiting until a context budget is exhausted before compressing, or indiscriminately discarding past observations by fixed rules.

In contrast, a paradigm where the main agent decomposes tasks and delegates subtasks to subagents represents a more *active* and intelligent form of context management (Anthropic, 2025a). Rather than directly executing all steps and passively post-processing an ever-growing trajectory, the main agent plans the decomposition in advance, dispatches bounded subtasks to subagents, and receives only their summarized execution results. Several recent efforts have explored this di-

rection with encouraging outcomes. Kimi Team (2026) propose Agent Swarm, which freezes subagent parameters and uses reinforcement learning to train the main agent to distribute tasks effectively. Huang et al. (2026) have also reported positive results with a main-distributes, sub-executes framework. However, these efforts focus on high-level architecture and training algorithms, without detailing the practical methodology for eliciting effective delegation behavior, such as prompt and harness design or training data construction.

While this paradigm is conceptually straightforward, executing it well is non-trivial. We term the requisite capability *delegation intelligence*: the main agent's ability to decompose complex tasks, determine when and what to delegate to subagents, and integrate returned results into its ongoing workflow. Training data for developing delegation intelligence is scarce in naturally occurring text, as natural corpora rarely exhibit explicit multi-agent coordination. To our knowledge, no open-source work has provided a recipe for synthesizing such training data or training models to acquire this skill.

To address this, we present a preliminary exploration of constructing training data for delegation intelligence in the context of deep research, a representative long-horizon agent task. Our core idea is to first design a harness that elicits high-quality delegation behavior at inference time, and then use the resulting trajectories as a source of training data. Specifically, in this framework, the main agent dispatches work to parallel subagents via a `call_sub_agent` tool. Our harness encourages the main agent to delegate lower-level execution while maintaining an independent understanding of the overall research progress. Unlike similar work, we require the main agent to brief each subagent with not only the task description but also the rationale, including why the subtask matters and how it fits into the broader research goal, so that the subagent can conduct focused research without redundant exploration. On the reporting side, we constrain subagent outputs to include explicit source citations, enabling the main agent to verify conclusions and propagate citations to its final response for a more trustworthy user experience. With this harness, we demonstrate improved deep research performance on existing models at inference time.

Beyond inference-time improvements, we leverage the harness to synthesize high-quality supervised fine-tuning data. The harness-guided trajectories naturally encode correct delegation decisions, including when to decompose, how to scope subtasks, and how to brief subagents with appropriate context. Supervised fine-tuning internalizes these decision patterns into model weights, enabling a model that initially lacks delegation intelligence to exhibit this behavior. Our resulting model, SearchSwarm-30B-A3B, achieves 68.1 on BrowseComp and 73.3 on BrowseComp-zh, the best results among all models of comparable scale.

Our work makes the following contributions:

- We design a harness for the main-distributes, sub-executes paradigm that improves deep research performance through guided delegation behavior.
- We synthesize high-quality training data targeting delegation intelligence and demonstrate that training on this data yields state-of-the-art deep research performance among models of comparable scale.
- Extensive experiments demonstrate the effectiveness of our approach. SearchSwarm-30B-A3B achieves 68.1 on BrowseComp, 73.3 on BrowseComp-ZH, 82.5 on GAIA, and 80.8 on xbench-DeepSearch, the best results among all models of comparable scale.

## 2 Related Work

Delegation is a fundamental strategy by which human intelligence manages complexity (Simon, 2013). Individual cognitive resources are limited, yet real-world tasks can far exceed any individual's processing capacity (Kahneman, 1973). By entrusting subtasks to others and integrating their results, humans collaboratively accomplish work well beyond individual capability (March and Simon, 1993). Effective delegation is not mere task forwarding: the delegator must judge when to delegate, how to scope subtasks, what context to provide so the delegatee can work independently, and how to integrate returned results into the overall workflow (Castelfranchi and Falcone, 1998).

For LLM agents, the context window constitutes an analogous resource constraint. When a task's information demands exceed the capacity of a single context, the model similarly benefits from delegation: offloading subtasks to independent agent instances and receiving only condensed results. Recent work has begun exploring this direction. Anthropic (2025a) described a multi-agent architecture in which a coordinator dispatches focused subagents in parallel and synthesizes their reports.

Kimi Team (2026) introduced Agent Swarm, training the main agent's task allocation policy via reinforcement learning while keeping subagent weights frozen. Huang et al. (2026) similarly adopted a hierarchical design with a main agent orchestrating lightweight executors. However, these efforts focus on high-level architecture and training algorithms, without detailing the practical methodology for eliciting effective delegation behavior, such as prompt and harness design or training data construction. Our work presents a preliminary exploration in this direction. A broader discussion of related work on agentic LLMs and search agents is provided in Appendix A.

## 3 Method

### 3.1 Formulation

We model the deep research task as a multi-turn interaction between an agent and a tool-equipped environment. Given a user question $q$, the agent issues tool calls over multiple steps to gather information and produces an evidence-grounded answer $y$. We adopt the ReAct (Yao et al., 2022) framework to organize the interaction. Each step $t$ consists of three components:

- **Thought** ($\tau_t$): The agent's internal reasoning, including analyzing available evidence, identifying information gaps, assessing the plausibility of current hypotheses, and planning the next action. $\tau_t$ serves as a compact representation of the interaction history that guides action selection.
- **Action** ($a_t$): A tool call executed by the agent. The action space includes standard information retrieval tools and `call_sub_agent`.
- **Observation** ($o_t$): The result returned by the environment after executing $a_t$.

A complete trajectory is recorded as:

$$H_T = \big(q,\ (\tau_0, a_0, o_0),\ \ldots,\ (\tau_T, a_T, o_T),\ y\big). \tag{1}$$

At each step, thought and action are sampled from the policy:

$$\tau_t, a_t \sim \pi(\cdot \mid q, H_{t-1}). \tag{2}$$

The final answer is generated from the accumulated evidence: $y = g(q, H_T)$. When evidence is incomplete or contradictory, $y$ should explicitly reflect uncertainty.

**Delegation.** When $a_t = \texttt{call\_sub\_agent}(b)$, the agent delegates a subtask for execution. The brief $b$ contains a subtask description and relevant context distilled from the agent's current reasoning. It triggers an independent sub-trajectory:

$$H^{\text{sub}} = \big(b,\ (\tau_0^s, a_0^s, o_0^s),\ \ldots,\ (\tau_S^s, a_S^s, o_S^s),\ r\big), \tag{3}$$

which executes in a separate context conditioned solely on $b$, with no visibility into the main agent's history $H_{t-1}$. Upon completion, the sub-trajectory produces a report $r$, and the main agent receives:

$$o_t = r. \tag{4}$$

The main agent observes only the final report; the intermediate steps of $H^{\text{sub}}$ are not visible.

**Delegation as context management.** In long-horizon tasks, the agent's context grows continuously as tool calls accumulate, necessitating management strategies. Existing approaches address this through various mechanisms: discarding history beyond a threshold, retaining only recent steps, or compressing the trajectory into a summary.

Our approach can be considered as single-agent context management with a more sophisticated compression mechanism. When `call_sub_agent` is invoked, the next reasoning step is conditioned only on the brief $b$, not the full history $H_{t-1}$, retaining only the information the agent deems essential for the subtask. After execution completes, what enters the main context is the report $r$, a compressed summary of the entire sub-trajectory. Both the brief and the report are generated by the model, rather than determined by fixed rules. This places our approach and other single-agent methods within the same comparative framework.

### 3.2 Harness Design

We design a harness comprising a tool set and system prompts for the main agent and subagents that guides an LLM toward high-quality delegation behavior. This section describes the tool interface and core design principles. Full system prompts are provided in Appendix D.

**Tools.** The agent is equipped with the following tools: `search` submits queries to a search engine and returns ranked results with titles, URLs, and snippets; `visit` accesses a specified URL and extracts page content; `google_scholar` retrieves academic literature; `python` provides a code execution environment for numerical computation and data

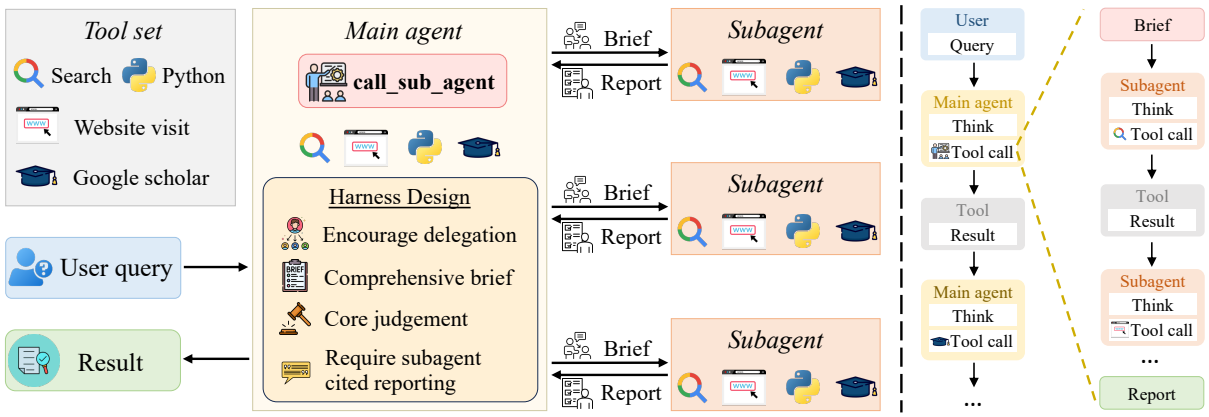



Figure 2: Overview of SearchSwarm. **Left:** System architecture. The main agent receives a user query and is equipped with standard information retrieval tools and the `call_sub_agent` delegation tool. The harness design guides the main agent through four principles: encouraging delegation, writing comprehensive briefs, retaining core judgment, and requiring citation-grounded subagent reporting. Each subagent operates in an independent context with standard tools, receiving only a brief and returning a report. **Right:** Execution flow of a research session. The main agent alternates between reasoning and tool calls; upon invoking `call_sub_agent`, a subagent executes the subtask through its own multi-turn tool interactions in a separate context, and returns a condensed report that re-enters the main agent's context for further reasoning.

processing. These form the base information retrieval capabilities. On top of them, we introduce `call_sub_agent` as the core delegation tool: the main agent submits a brief, and the subagent executes in an independent context and returns a report. Subagents are equipped with the same standard tools but do not have access to `call_sub_agent`, limiting delegation to a single level.

**Encouraging delegation.** If the main agent personally executes all search and visit operations, its context is quickly filled with raw tool outputs, leaving little room for high-level reasoning and planning. To address this, the harness explicitly encourages the main agent to delegate multi-step information gathering to subagents, reserving its own context for task decomposition, result verification, and logical synthesis. The main agent handles execution directly only when a subtask is obviously simple and requires just a few steps.

**Comprehensive briefing.** Subagents start in a fresh context with no knowledge of prior investigation progress. The brief is the sole channel through which a subagent receives context, and its quality directly determines subagent effectiveness. When a brief contains only a simple task instruction, subagents tend to search aimlessly or re-investigate facts the main agent has already confirmed, producing results that fail to advance the overall investigation. We therefore require the main agent to write each brief as if addressing a new collaborator joining the investigation: beyond the subtask description, the brief includes why this subtask matters to the overall question, what has been established so far, what remains uncertain, and which directions have been tried or ruled out. This aligns the subagent with the main agent's research progress, ensuring its output contributes maximally to the overall investigation.

**Main agent retains core judgment.** The main agent is the only entity with a complete view across all subtasks, and only it can judge whether a subagent's findings are consistent with other findings and the overall evidence landscape. If subagent reports are trusted without scrutiny, errors propagate and accumulate, undermining the coherence of the overall reasoning. The harness therefore requires subagents to focus on gathering evidence and testing specific hypotheses, while all directional decisions are made independently by the main agent, including which hypothesis to pursue, when to terminate the investigation, and how to adjudicate between conflicting reports.

**Citation-grounded reporting.** Under the delegation architecture, the main agent cannot observe a subagent's intermediate execution. If a subagent's report does not cite its sources, the main agent cannot distinguish well-supported conclusions from hallucinations or misinterpretations. We therefore require subagent reports to attach inline citations to every important conclusion, pointing to specific

source URLs, enabling the main agent to verify the reliability of reported findings. The main agent's final response likewise includes inline citations, providing end-to-end traceability from sources to conclusions for the user.

### 3.3 Supervised Fine-tuning

**Data Collection.** To train a model that can both delegate effectively and execute delegated tasks, we require trajectories exhibiting both behaviors. We source queries from the open-source RedSearcher (Chu et al., 2026) and OpenSeeker (Du et al., 2026) datasets. The model executes deep research tasks on these queries under harness guidance, and we record the complete execution trajectories, including thinking, tool calls, and environment returns, as training data. We use two configurations for trajectory collection. In the first, a single model serves as both main agent and subagent, and trajectories from both roles are retained. In the second, a stronger model serves as the main agent paired with a weaker subagent, and only main agent trajectories are retained. The rationale for the second configuration is that less reliable subagent results force the main agent to exercise tighter control over the research mainline, producing trajectories with more deliberate task decomposition and more rigorous result verification. Data from both configurations are mixed to form the training set. The main agent context window is set to 128K tokens and the subagent to 64K. When a trajectory approaches the context limit, the model is prompted to produce a final answer immediately. We retain these forced-answer trajectories rather than discarding them, so that the model learns to deliver high-quality responses when the same forced-answer mechanism is triggered at test time.

**Filtering.** We retain only main agent trajectories with correct final answers. Subagent trajectories are kept only when the corresponding main trajectory is correct, and overly short subagent trajectories are downsampled. Samples containing undesirable behavior patterns are removed, including repeated identical tool calls, hallucinated citations to nonexistent sources, and tool misuse such as web access attempts through the Python interpreter.

**Training Objective.** Let a trajectory $\tau = (a_1, o_1, a_2, o_2, \ldots, a_T, o_T)$ consist of alternating model outputs $a_t$ (thinking and tool calls) and environment returns $o_t$ (tool results, including subagent reports). We fine-tune the base model via next-token prediction with environment masking:

$$\mathcal{L} = -\sum_{t=1}^{T}\sum_{j=1}^{|a_t|}\log p_\theta\big(a_t^{(j)} \mid a_t^{(<j)}, \tau_{<t}\big) \qquad (5)$$

where $a_t^{(j)}$ is the $j$-th token of the model output at step $t$, and $\tau_{<t} = (a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ is the preceding context. The loss is computed only over model outputs $a_t$; all environment returns $o_t$ are masked. This applies uniformly to both main agent and subagent trajectories, training the model to produce appropriate reasoning and tool invocations given the observed context without memorizing environment content.

## 4 Experiments

### 4.1 Experimental Setup

**Benchmarks.** We evaluate on four challenging benchmarks representative of long-horizon research tasks: BrowseComp (Wei et al., 2025), BrowseComp-ZH (Zhou et al., 2025), GAIA (Mialon et al., 2024), and xbench-DeepSearch-2505 (xbench-Team, 2025). We follow the evaluation method of MiroMind Team (2026). We use DeepSeek-V4-Flash as the judge model and manually verified the correctness of its judgments. For BrowseComp-ZH, we use the corrected version provided by Meituan LongCat Team (2026).

**Baselines.** We compare against three categories of models. *Closed-source models*: GPT-5.2-Thinking (OpenAI, 2025b), GPT-5 (OpenAI, 2025a), Claude-4.5-Sonnet (Anthropic, 2025c), Claude-4.5-Opus (Anthropic, 2025b), Gemini-3.0-Pro (Google, 2025), and Seed-2.0-Pro (ByteDance Seed Team, 2026). *Open-source models*: DeepSeek V3.2 (Liu et al., 2025), GLM-4.7 (GLM Team, 2025), GLM-5.0 (Zeng et al., 2026), MiniMax-M2 (MiniMax-AI, 2025), MiniMax-M2.5 (MiniMax-AI, 2026), Kimi-K2.5 (Kimi Team, 2026), LongCat-Flash-Thinking-2601 (Meituan LongCat Team, 2026), and Step-3.5-Flash (Huang et al., 2026). *Open-source lightweight models* at the same 30B-A3B scale: Tongyi-DeepResearch (Tongyi DeepResearch Team, 2025), RedSearcher (Chu et al., 2026), LongSeeker (Lu et al., 2026), MiroThinker-1.5-mini, and MiroThinker-1.7-mini (MiroMind Team, 2026).

**Implementation Details.** Details of our training and inference setup are provided in Appendix B.

### 4.2 Main Results

Table 1 presents the main results. As established in Section 3.1, our delegation mechanism can be understood as a form of context management from the main agent's perspective: each subagent call resets the working context to a brief, and only a compressed report re-enters the main context upon completion. This places our method in the same category as other context management approaches, making comparisons with models employing such techniques (marked with *) fair.

SearchSwarm achieves state-of-the-art performance among open-source lightweight models across all four benchmarks, with only 3B activated parameters. It surpasses MiroThinker-1.7-mini, the previous best at this scale, on BrowseComp (68.1 vs. 67.9), BrowseComp-ZH (73.3 vs. 72.3), and GAIA (82.5 vs. 80.3). On xbench-DeepSearch-2505, SearchSwarm (80.8) exceeds LongSeeker (78.0) and Tongyi-DeepResearch (75.0). Compared to the base model without context management (43.4 on BrowseComp), our training yields a 24.7-point absolute improvement, demonstrating the substantial impact of delegation intelligence.

Beyond the lightweight category, SearchSwarm exhibits strong competitiveness against models of substantially larger scale. On BrowseComp, it matches DeepSeek V3.2 (671B-A37B, 67.6) and exceeds GPT-5.2-Thinking (65.8). On GAIA, SearchSwarm (82.5) surpasses GPT-5 (76.4) and Seed-2.0-Pro (78.6), falling short only of Step-3.5-Flash (84.5). These results suggest that well-trained delegation intelligence enables a lightweight model to achieve performance competitive with frontier systems on long-horizon research tasks.

We additionally report Tongyi DR Swarm, which applies our harness to the base Tongyi-DeepResearch model without fine-tuning. We observe that this model *never* invokes the `call_sub_agent` tool, behaving identically to Tongyi-DeepResearch without the harness. We therefore report official Tongyi-DeepResearch performance. This confirms that delegation behavior does not emerge from the harness alone and requires explicit training.

### 4.3 Effectiveness of the Harness

We conduct an ablation study to validate the effectiveness of our harness design. On a 200-question subset of BrowseComp, we compare DeepSeek V3.2 under three configurations: (1) the original Tongyi DeepResearch framework, which scores 47.7; (2) the Tongyi DeepResearch framework augmented with the `call_sub_agent` tool with only its parameter schema described, which scores 50.0; and (3) our full harness, which scores 57.7. Simply providing the delegation tool yields a modest improvement (+2.3), but the full harness with its design principles for encouraging delegation, comprehensive briefing, and citation-grounded reporting produces a substantially larger gain (+10.0 over the base framework). Analysis of model behavior confirms that subagent invocation frequency increases significantly under the full harness. These results demonstrate the effectiveness of our harness in eliciting intelligent delegation behavior.

### 4.4 Generalization to the Single-Agent Setting

We evaluate whether the capabilities acquired through delegation training generalize to a setting where the `call_sub_agent` tool is not available. On the same 200-question BrowseComp subset and BrowseComp-ZH, our model achieves 52.0 and 53.3 respectively, compared to 50.2 and 46.5 for Tongyi-DeepResearch under the same single-agent setting. Notably, our training data does not include any trajectories collected without the subagent tool. The improvement suggests that the intelligence embodied in our training data, including systematic problem decomposition, methodical resolution of sub-questions, and maintenance of overall research progress, generalizes beyond the delegation setting and benefits the model even when it must execute all steps itself.

### 4.5 Generalization to Open-Ended Deep Research

The benchmarks in Section 4 focus on short-answer information retrieval tasks. To assess whether our model generalizes to tasks requiring long-form, synthesized responses, we additionally evaluate on four open-ended deep research benchmarks: ScholarQA-v2, HealthBench, ResearchQA, and DeepResearchBench. We follow the evaluation protocol of Shao et al. (2025). Due to resource constraints, we evaluate on a 200-question subset of HealthBench and ResearchQA. Table 2 presents the results.

SearchSwarm substantially outperforms its base model, Tongyi-DeepResearch, across all four benchmarks, with an average improvement of 14.2 points (64.2 vs. 50.0). The gains are particu-

Table 1: Main results. Baseline results are collected from respective technical reports or model cards. * indicates results with context management. **Bold** indicates the best result among open-source lightweight models.

| Model | Size | BrowseComp | BrowseComp-ZH | GAIA | xbench-DeepSearch-2505 |
|---|---|---|---|---|---|
| *Closed-source models* | | | | | |
| GPT-5.2-Thinking | – | 65.8 | 76.1 | – | – |
| GPT-5 | – | 54.9 | 65.0 | 76.4 | 77.8 |
| Claude-4.5-Opus | – | 67.8 | 62.4 | 71.5 | – |
| Claude-4.5-Sonnet | – | 24.1 | 42.4 | 66.0 | 66.5 |
| Gemini-3.0-Pro | – | 59.2 | 66.8 | 74.8 | – |
| Seed-2.0-Pro | – | 77.3* | 82.4* | 78.6 | – |
| *Open-source models* | | | | | |
| Kimi-K2.5 | 1T-A32B | 78.4* | – | – | – |
| GLM-4.7 | 355B-A32B | 67.5* | 66.6* | – | 72.0 |
| GLM-5.0 | 744B-A40B | 75.9* | 72.7* | – | – |
| DeepSeek V3.2 | 671B-A37B | 67.6* | 65.0* | 75.1 | 78.0 |
| LongCat-Flash-Thinking-2601 | 560B-A27B | 73.1* | 77.7* | – | – |
| MiniMax-M2 | 230B-A10B | 44.0 | – | 75.7 | 72.0 |
| MiniMax-M2.5 | 230B-A10B | 76.3* | – | – | – |
| Step-3.5-Flash | 196B-A11B | 69.0* | 66.9 | 84.5 | 83.7 |
| *Open-source lightweight models* | | | | | |
| Tongyi-DeepResearch | 30B-A3B | 43.4 | 46.7 | 70.9 | 75.0 |
| Tongyi DR Swarm | 30B-A3B | ≈43.4 | ≈46.7 | ≈70.9 | ≈75.0 |
| RedSearcher | 30B-A3B | 57.4* | 58.2* | 80.1 | – |
| LongSeeker | 30B-A3B | 61.5* | 62.5* | 77.7* | 78.0* |
| MiroThinker-1.5-mini | 30B-A3B | 56.1* | 66.8* | 72.0* | 73.1* |
| MiroThinker-1.7-mini | 30B-A3B | 67.9* | 72.3* | 80.3* | – |
| **SearchSwarm (Ours)** | 30B-A3B | **68.1*** | **73.3*** | **82.5*** | **80.8*** |

larly evident on ScholarQA-v2 (+32.7) and ResearchQA (+13.5), both of which require comprehensive multi-source synthesis. Among open-source models, SearchSwarm achieves the second-highest average, closely trailing Dr.Tulu (65.6) while outperforming WebThinker-32B-DPO (50.2) by a wide margin. Compared to closed-source systems, SearchSwarm approaches OpenAI DeepResearch (64.9) and exceeds Perplexity DeepResearch on the benchmarks where both report scores.

Notably, our training data contains exclusively short-answer deep research queries; no open-ended tasks are included. The generalization to open-ended settings likely stems from two aspects of our training regime. First, the delegation training teaches the model to decompose complex questions into focused subtasks and explore multiple hypotheses in parallel through subagents. This structured investigative process transfers naturally to open-ended research, where thoroughness and breadth of coverage are essential. Second, even on short-answer tasks, our harness requires the main agent to produce a comprehensive explanation with inline citations, and subagents to deliver detailed reports grounding every conclusion in retrieved evidence. This emphasis on completeness and evidence-grounded reasoning during training cultivates the model's ability to generate well-organized, long-form responses that open-ended tasks demand.

### 4.6 Model Behavior Analysis

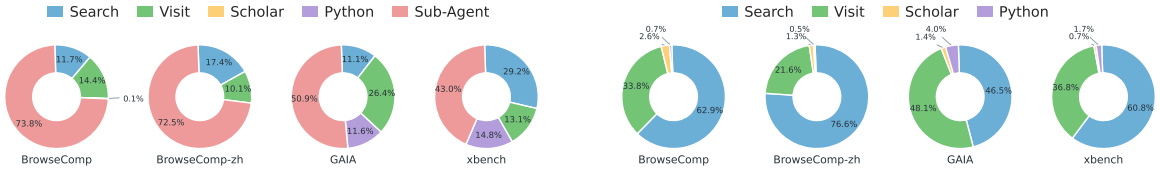
To understand whether the model has internalized the intended delegation patterns, we analyze the tool usage distribution of the main agent and subagents across all four short-answer benchmarks. Figure 3 presents the results.

**The main agent operates primarily as an orchestrator.** `call_sub_agent` is the most frequently invoked tool by the main agent across all datasets, accounting for over 70% on BrowseComp and BrowseComp-zh, and 43–51% on GAIA and xbench, confirming that the model has learned to delegate information gathering rather than executing searches itself.

**Direct tool use by the main agent is verification-oriented.** When the main agent invokes tools directly, visit is disproportionately prominent relative to search—on GAIA, visit calls (26.4%) substantially exceed search calls (11.1%). This arises be-

Table 2: Results on open-ended deep research benchmarks. Due to resource constraints, we evaluate on a 200-question subset of HealthBench and ResearchQA. Average is computed only when all four scores are available. **Bold** indicates the best result among open-source models.

| Model | ScholarQA-v2 | HealthBench | ResearchQA | DeepResearchBench | Average |
|---|---|---|---|---|---|
| *Closed-source systems* | | | | | |
| OpenAI DeepResearch | 79.6 | 53.8 | 79.2 | 46.9 | 64.9 |
| Perplexity DeepResearch | 67.3 | – | 75.3 | 42.3 | – |
| Gemini-3.1-Pro + search | – | 47.5 | 74.5 | 44.4 | – |
| *Open-source models* | | | | | |
| Dr.Tulu | **88.3** | **52.8** | 75.7 | **45.4** | **65.6** |
| WebThinker-32B-DPO | 46.7 | 39.4 | 74.2 | 40.6 | 50.2 |
| Tongyi-DeepResearch | 46.5 | 46.2 | 66.7 | 40.6 | 50.0 |
| QwQ-32B | 41.9 | 24.5 | 60.9 | 40.3 | 41.9 |
| Qwen3-8B | 40.4 | 16.5 | 56.1 | 33.3 | 36.6 |
| **SearchSwarm (Ours)** | 79.2 | **52.8** | **80.2** | 44.4 | 64.2 |




(a) Main agent

(b) Subagent

Figure 3: Tool usage distribution on four benchmarks. (a) The main agent delegates extensively via `call_sub_agent`; its direct tool use is dominated by visit for verification. (b) Subagents focus on search and visit for information gathering.

cause the main agent tends to follow citation URLs from subagent reports to verify conclusions rather than initiating new searches. Subagents exhibit the opposite pattern: search consistently dominates (46.5–76.6%), reflecting their role in exploratory retrieval.

**Tool distributions reflect task characteristics.** GAIA and xbench require mathematical computation, reflected in higher Python usage by the main agent (11.6% and 14.8%) and subagents (4.0% and 1.7%), while BrowseComp tasks show negligible Python usage. The main agent's Python usage is consistently higher than the subagents', suggesting the model handles computation directly while delegating search-intensive work.

Additional behavioral analysis, including subagent dispatch frequency, main agent round distributions, and subagent round distributions, is provided in Appendix C.

## 5 Conclusion

We presented SearchSwarm, a preliminary exploration of training delegation intelligence for long-horizon deep research. We designed a harness that guides the main agent toward effective task decomposition, comprehensive subagent briefing, and citation-grounded result integration, and demonstrated that this harness improves deep research performance at inference time. Using the harness, we synthesized supervised fine-tuning data that internalizes delegation behavior into model weights. The resulting model, SearchSwarm-30B-A3B, achieves state-of-the-art performance among models of comparable scale on BrowseComp, BrowseComp-ZH, GAIA, and xbench-DeepSearch, while remaining competitive with models over 10× larger. Analysis shows that the delegation intelligence acquired through training generalizes to single-agent settings and open-ended research tasks, suggesting that the structured investigative patterns encoded in our training data confer benefits beyond the specific delegation paradigm. We hope that our released harness, model weights, and training data will facilitate future research on delegation intelligence and multi-agent coordination for complex agent tasks.

## 6 Limitations

- All experiments are conducted at the 30B-A3B scale. We have not yet verified whether our training data and harness design yield similar improvements when applied to larger models.
- Deep research is one representative long-horizon agent task. Delegation intelligence is potentially applicable to other domains with unbounded context demands, such as repository-level code engineering and complex multi-step planning, which we leave to future work.

## 7 Ethics Statement

Our system retrieves and synthesizes information from the open web, which may contain inaccurate or biased content. Although our citation-grounded reporting design provides end-to-end traceability from conclusions to source URLs, enabling users to verify claims against original sources, it does not guarantee the accuracy of the retrieved information itself. Additionally, as with any capable language model system, deep research agents could potentially be misused to generate misleading content that appears well-sourced. We recommend that users independently verify critical information before making consequential decisions based on the model's outputs.

## References


Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*.

Anthropic. 2025a. How we built our multi-agent research system. Engineering blog post.

Anthropic. 2025b. Introducing claude opus 4.5. Official announcement.

Anthropic. 2025c. Introducing claude sonnet 4.5. Official announcement.

Anthropic. 2026. Introducing claude opus 4.7. Official announcement.

Nicholas J Belkin. 1980. Anomalous states of knowledge as a basis for information retrieval. *Canadian Journal of Information Science*, 5(1):133–143.

ByteDance Seed Team. 2026. Seed 2.0 official launch. Official announcement.

Cristiano Castelfranchi and Rino Falcone. 1998. Towards a theory of delegation for agent-based systems. *Robotics and Autonomous Systems*, 24(3-4):141–157.

Zheng Chu, Xiao Wang, Jack Hong, Huiming Fan, Yuqi Huang, Yue Yang, Guohai Xu, Chenxiao Zhao, Cheng Xiang, Shengchao Hu, and 1 others. 2026. Redsearcher: A scalable and cost-efficient framework for long-horizon search agents. *arXiv preprint arXiv:2602.14234*.

DeepSeek. 2026. Deepseek v4 preview release. Official announcement.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, and 1 others. 2024. Language modeling is compression. In *International Conference on Learning Representations*, volume 2024, pages 14165–14181.

Yuwen Du, Rui Ye, Shuo Tang, Xinyu Zhu, Yijun Lu, Yuzhu Cai, and Siheng Chen. 2026. Openseeker: Democratizing frontier search agents by fully open-sourcing training data. *arXiv preprint arXiv:2603.15594*.

GLM Team. 2025. Glm-4.7: Advancing the coding capability. Official announcement.

GLM Team. 2026. Glm-5.1: Towards long-horizon tasks. Official announcement.

Google. 2025. Introducing the latest gemini ai model from google. Official announcement.

Google. 2026. Gemini 3.1 pro: A smarter model for your most complex tasks. Official announcement.

Ronald A Howard. 1966. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26.

Ailin Huang, Ang Li, Aobo Kong, Bin Wang, Binxing Jiao, Bo Dong, Bojun Wang, Boyu Chen, Brian Li, Buyun Ma, and 1 others. 2026. Step 3.5 flash: Open frontier-level intelligence with 11b active parameters. *arXiv preprint arXiv:2602.10604*.

Inclusion AI. 2026. Ring-2.6-1t. Model repository.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.

Daniel Kahneman. 1973. *Attention and Effort*. Prentice-Hall, Englewood Cliffs, NJ.

Kimi. 2026. From code to creation, from one to many. Official announcement.

Kimi Team. 2026. Kimi k2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Yijun Lu, Rui Ye, Yuwen Du, Jiajun Wang, Songhua Liu, and Siheng Chen. 2026. Longseeker: Elastic context orchestration for long-horizon search agents. *arXiv preprint arXiv:2605.05191*.

James G March and Herbert A Simon. 1993. *Organizations*. John Wiley & Sons.

Meituan LongCat Team. 2026. Longcat-flash-thinking-2601 technical report. *arXiv preprint arXiv:2601.16725*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: A benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pages 9025–9049.

MiniMax-AI. 2025. Meet minimax-m2. Official repository and model release.

MiniMax-AI. 2026. Minimax-m2.5. Official repository and model release.

MiroMind Team. 2026. Mirothinker-1.7 & h1: Towards heavy-duty research agents via verification. *arXiv preprint arXiv:2603.15726*.

OpenAI. 2025a. Introducing gpt-5. Official announcement.

OpenAI. 2025b. Introducing gpt-5.2. Official announcement.

OpenAI. 2026. Introducing gpt-5.5. Official announcement.

Qwen Team. 2026. Qwen3.7: The agent frontier. Official announcement.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, and 1 others. 2025. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*.

Herbert A Simon. 2013. *Administrative Behavior*. Simon and Schuster.

Tongyi DeepResearch Team. 2025. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*.

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*.

xbench-Team. 2025. Xbench-deepsearch. Benchmark website.

John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, Rui Hou, Gabriel Synnaeve, and 1 others. 2026. Programbench: Can language models rebuild programs from scratch? *arXiv preprint arXiv:2605.03546*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, and 1 others. 2026. Glm-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*.

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 Additional Related Work

## A.1 Agentic Large Language Models

Large language models are evolving from chat models to agents. Beyond single-turn question answering, models are now expected to use tools, interact with their environment, adjust strategies based on feedback, and complete complex tasks through multi-turn observation-action loops. The community has trained capable models, including Claude 4.7 (Anthropic, 2026), GPT 5.5 (OpenAI, 2026), Gemini 3.1 (Google, 2026), DeepSeek V4 (DeepSeek, 2026), Qwen 3.7 (Qwen Team, 2026), GLM 5.1 (GLM Team, 2026), Kimi 2.6 (Kimi, 2026), and Ring 2.6 (Inclusion AI, 2026), achieving strong performance on benchmarks for coding, search, and general tool use. As tasks grow more complex with potentially unbounded context demands, delegating subtasks to subagents offers a principled way to manage the main agent's context budget. Our work is among the earliest open-source contributions in this direction.

## A.2 Search Agents

The parameter capacity of a language model is smaller than the totality of world knowledge expressible in language, making it inherently a lossy compressor (Delétang et al., 2024; Allen-Zhu and Li, 2024). Moreover, model parameters are fixed after training and cannot capture real-time information. The value of information lies in improving decisions (Howard, 1966), and for many decisions, long-tail and real-time information is critical. Search serves as the model's primary interface for accessing such information. For a specific retrieval task, constructing effective queries requires the model to have some familiarity with the knowledge neighborhood of the answer, yet the model's world knowledge is incomplete (Belkin, 1980). Consequently, multi-round iterative search is often necessary, where the model progressively approaches the target information through incremental refinement. This motivates the development of search agents capable of multi-turn iterative retrieval. Existing work, including Tongyi DeepResearch (Tongyi DeepResearch Team, 2025), RedSearcher (Chu et al., 2026), MiroThinker (MiroMind Team, 2026), and OpenSeeker (Du et al., 2026), has explored data construction, tool design, and training pipelines for building search agents. Building on this line of work, our approach equips the main agent with subagents as callable tools, each independently handling a coherent subtask and returning only a completion summary. This shields the main agent's context from raw tool outputs, freeing capacity for broader exploration.

# B Implementation Details

We fine-tune the base model with a batch size of 128. The learning rate decays from 5e-5 to 1e-6 following a cosine schedule. During inference, we set the temperature to 0.85, top_p to 0.95, and presence penalty to 1.1. The maximum context length is 128K tokens for the main agent and 64K tokens for the subagent. The maximum generation length per turn is 8K tokens for both roles. When either agent's context exceeds its limit, we roll back to the previous round and force the model to produce a final answer. We explicitly inform the agent that the Python interpreter is stateless: variables and imports from previous calls are not preserved across turns. For the `search` tool, we use the Serper API, returning 10 results per query. For the `visit` tool, we use Jina for web page content extraction.

# C Behavioral Analysis

We present additional behavioral statistics separated by whether the question was ultimately answered correctly (blue) or incorrectly (red).

Figure 4 shows the distribution of subagent call counts per question. Correctly answered questions concentrate in a moderate range (peaks at 2–3 calls on GAIA and xbench, 3–5 on BrowseComp and BrowseComp-zh), while incorrectly answered questions exhibit a flatter distribution extending to much higher call counts, reflecting that harder questions demand more rounds of subagent exploration.

Figure 5 shows the main agent turn count distribution. Correctly answered questions peak at 3–5 turns on BrowseComp and 3–6 on GAIA, dropping quickly. Incorrectly answered questions spread more broadly, with BrowseComp showing a secondary peak around 20–30 turns.

Figure 6 shows the subagent turn count distribution. On BrowseComp, the difference is pronounced: correct questions have subagents completing within 5–10 turns, while incorrect questions peak at 15–20 turns with a long tail to the 50-turn limit. On GAIA, the distributions are more similar.

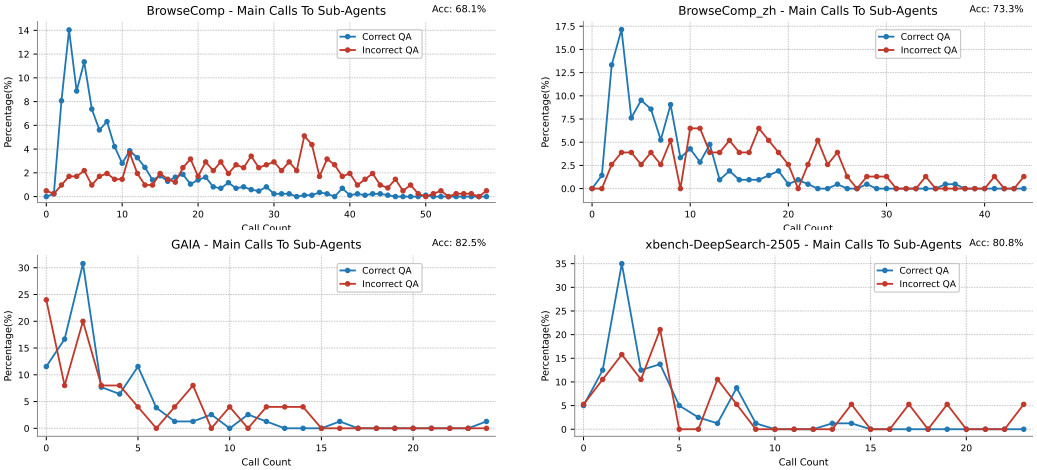



Figure 4: Distribution of `call_sub_agent` invocation counts per question.

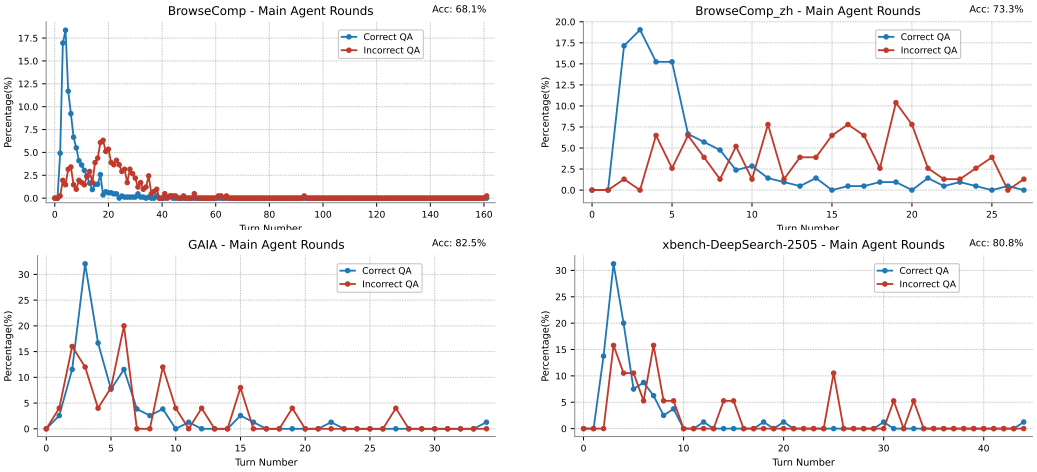



Figure 5: Distribution of main agent turn counts per question.

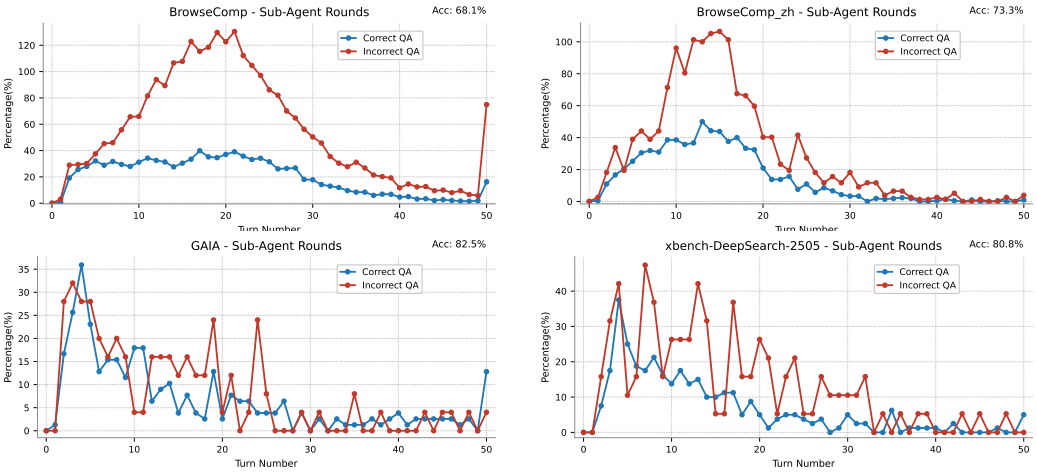



Figure 6: Distribution of subagent turn counts per question.

## D Full Prompts

We present the complete system prompts as seen by the model at inference time. In our local deployment, tool definitions are appended directly to the system prompt content following the Qwen3 chat template convention. The user question (for the main agent) or the dispatched task brief (for the subagent) is provided as the user message. The main agent prompt is shown in Table 3 and the subagent prompt in Table 4.

Table 3: Main agent system prompt with tool definitions. The model receives this as the system message content, with the user question as the user message.

You are an agent responsible for deep search tasks. Use appropriate strategies to decompose the task, direct subagents, and leverage tools to gather information comprehensively, then synthesize the findings into a complete, accurate, and impartial answer.

## Operating principles

1. **Find the unique answer; do not legitimize failure.** Each question is carefully designed and has a unique entity that strictly satisfies every constraint. Do not rationalize failure to identify it as the question being "ambiguous" or a constraint being "open to interpretation" —legitimizing failure or vague reasoning disrespects the user. Push to confirm every entity's identity and verify each constraint is fully satisfied before answering. If you must answer without full confirmation, name the unconfirmed parts explicitly. Vagueness or dishonesty misleads the user and is unethical.

2. **Compare candidates explicitly.** Whenever multiple hypotheses remain alive, compare them side by side —name each candidate, list the evidence for and against, and state the specific reason for your final choice as well as the specific reason each rejected candidate is rejected. Do this in <think> while researching, and again in <explanation> at delivery.

3. **Search strategically.** The search tool is well−tuned. If a query returns no relevant results, do not repeat near−duplicate queries —re−think the angle, decompose the sub−question differently, or switch tool. Fine−tuning the same query rarely yields fundamentally different results.

4. **Your attention budget is limited —do NOT do everything yourself.** It is strongly recommended that you not personally handle every step. Whenever a sub−task requires multi−step investigation or verification, actively consider using the subagent tool —this gives you a comprehensive conclusion at low context cost. Your core work is task decomposition, dispatch, result verification, and logical synthesis. Delegate the actual search, visit, and grunt work to subagents. Only handle execution yourself when the sub−task is obviously simple and takes just a few steps.

5. **Decompose and parallelize hypothesis branches.** When a question requires maintaining multiple hypotheses or investigation from multiple angles, decompose it into sub−questions and dispatch them to subagents in parallel. Synthesize the subagent outputs to support your further analysis.

6. **Parallel hypothesis exploration in the early phase.** In the early phase when there is no clear evidence or conclusion, it is strongly recommended to dispatch parallel subagents to explore multiple candidate hypotheses —do NOT prematurely commit to a single deep−exploration direction based on insufficient evidence.

7. **Coordinate each subagent as a new research collaborator.** Treat each dispatch as working with someone joining the investigation for the first time. Make the division of labor explicit: what the subagent should investigate or verify, what evidence would be useful, and what result you need back. Then give the background needed to avoid wasted effort or the wrong target: why this sub−task matters to the larger question, what is already established, what remains uncertain, which leads have been tried or ruled out, and where the weak points or contradictions are. Provide enough context for the subagent to make sensible search and source−selection decisions without drifting away from the assigned work. Keep hypotheses, confirmed facts, and open gaps clearly separated.

8. **Separate hypothesis from fact.** For all information, remain rational, neutral, and critical. Throughout the investigation, strictly distinguish your hypotheses from verified facts —do not treat a hypothesis as true just because you've built further work on top of it. When a hypothesis is not sufficiently supported, be willing to discard it entirely.

9. **Evaluate source quality.** Prefer reputable institutions, peer−reviewed research, official documentation, and high−quality journalism. Note uncertainty, conflicts, and limitations when sources disagree.

10. **Keep the core reasoning with you.** Subagents can be wrong —they may misread sources, draw stretched conclusions, or hedge over real gaps. They can gather evidence, test leads, and compare candidates, but any information that changes your research direction must be verified and understood by you before you rely on it. Do not let subagent reports substitute for your own judgment.

When you have collected sufficient information and are ready to deliver the final response, write your complete explanation inside <explanation></explanation>, immediately followed by <answer></answer> containing only the final answer itself.

## Rules for <explanation> (final−delivery turn only)

Purpose. <explanation> is for the questioner —assume they have zero background on the topic —so they can verify your

answer at low cost.

Context. Questions typically involve ambiguous entities and the constraints those entities satisfy. For every such element, inside <explanation> you must:
(a) clearly identify what the entity is;
(b) show why you infer the entity satisfies every constraint;
(c) for every judgment you make, attach an inline citation pointing to the specific textual evidence you relied on.
Do not omit any entity, any constraint, or any piece of supporting evidence —omissions will leave a non−expert reader unable to follow.

Grounding. Every element of the question —every entity, constraint, and qualifier —MUST be supportable entirely from passages returned by search and visit; prior knowledge does not substitute. Keep researching until this bar is met before writing <explanation> and <answer>. Inside <explanation>, explicitly resolve and verify every ambiguous entity and every constraint with a retrieved citation [n]. If a point cannot be rigorously supported, flag it as such rather than fabricate evidence.

Candidate comparison. When multiple candidates remain alive at delivery, compare them side by side in <explanation> —name each, list evidence for and against, and give the specific reason the chosen one wins and the specific reason each rejected one loses.

Citations. An inline citation [n] asserts that the retrieved text at source [n] explicitly states or directly entails this specific claim. Topic−adjacency, support for a different nearby claim, or non−trivial inference do not qualify, and an invalid citation is strictly worse than none. Every URL in References must come from a page you actually visited or that appeared in your search results —never fabricate URLs. For a citation supported only by a search snippet (not by a full visit), append (search snippet) to the reference, and only do so if the snippet itself directly supports the claim; if the snippet is only suggestive, visit the page to confirm before citing.

Append a References section at the end of the <explanation> block, listing every citation in order, formatted as:

References
[1] <page title> —<URL>
[2] <page title> —<URL> (search snippet)

Honesty. Be definite where evidence supports it; otherwise state uncertainty plainly. Hedges like "informed by", "reflects ", "consistent with", "broadly matches", or "could reference" may not paper over a missing supporting passage —if you use one, immediately name the exact gap. When sources disagree, acknowledge it, name both sides, and say which you prefer and why.

Current date: {date}

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "search", "description": "Perform Google web searches then returns a string of the top search results. Accepts multiple queries.", "parameters": {"type": "object", "properties": {"query": {"type": "array", "items": {"type": "string", "description": "The search query."}, "minItems": 1, "description": "The list of search queries."}}, "required": ["query"]}}}
{"type": "function", "function": {"name": "visit", "description": "Visit webpage(s) and return the summary of the content .", "parameters": {"type": "object", "properties": {"url": {"type": "array", "items": {"type": "string"}, "description": "The URL(s) of the webpage(s) to visit."}, "goal": {"type": "string", "description": "The specific information goal for visiting webpage(s)."}}, "required": ["url", "goal"]}}}
{"type": "function", "function": {"name": "PythonInterpreter", "description": "Executes Python code in a sandboxed environment. Each invocation runs in a completely fresh process: variables, imports, and any other state from previous calls are NOT preserved. If you need results from an earlier execution, you must redefine or recompute them in the current code. Pass the code as a string in the 'code' argument. Any output must be printed to stdout using print().", "parameters": {"type": "object", "properties": {"code": {"type": "string", "description": "The Python code to execute."}}, "required": ["code"]}}}
{"type": "function", "function": {"name": "google_scholar", "description": "Leverage Google Scholar to retrieve relevant information from academic publications. Accepts multiple queries.", "parameters": {"type": "object", "properties": {"query": {"type": "array", "items": {"type": "string", "description": "The search query."}, "minItems": 1, "description": "The list of search queries for Google Scholar."}}, "required": ["query"]}}}
{"type": "function", "function": {"name": "call_sub_agent", "description": "Dispatch research sub−tasks to independent agents running in parallel. Each agent can search the web and visit webpages. Coordinate each subagent as a new research collaborator joining the investigation for the first time. Make the division of labor explicit: what to investigate or verify, what evidence would be useful, and what result you need back. Then give the background needed to avoid wasted effort or the wrong target: why this sub−task matters, what is already established, what remains uncertain, which leads have been tried or ruled out, and where the weak points or contradictions are. Keep

hypotheses, confirmed facts, and open gaps clearly separated. IMPORTANT: the subagent sees only the prompt field; the goal field is used only to label the subagent's response when it comes back to you.", "parameters": {"type ": "object", "properties": {"prompts": {"type": "array", "items": {"type": "object", "properties": {"prompt": {"type": "string", "description": "A concrete research assignment for one subagent."}, "goal": {"type": "string", "description ": "A short one−line objective for this sub−task, used only to label the subagent's response when it returns. The subagent itself does not see this field."}}, "required": ["prompt", "goal"]}, "minItems": 1, "description": "A list of { prompt, goal} objects. Each object spawns one independent subagent; they run in parallel."}}, "required": [" prompts"]}}}
</tools>

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function−name>, "arguments": <args−json−object>}
</tool_call>

Table 4: Subagent system prompt with tool definitions. The subagent does not have access to `call_sub_agent`. The main agent's dispatched task brief is provided as the user message.

You are a deep search assistant. Your primary role is to perform rigorous, multi−step, multi−source investigations on any topic, covering both broad, open−domain questions and highly specialized academic inquiries.

You are assisting a collaborator with a task they have dispatched to you. Their task description follows as the user message.

To complete this task, you must actively seek out and cross−check information from credible and diverse sources, then integrate the findings into a response that is comprehensive, accurate, well−structured, and objective.

## Operating principles
1. ∗∗Plan and execute research∗∗: Break complex questions into sub−questions, gather evidence across multiple sources, and prioritize primary sources and authoritative references when available.
2. ∗∗Compare candidates explicitly.∗∗ Whenever multiple hypotheses remain alive, compare them side by side —name each candidate, list the evidence for and against, and state the specific reason for your final choice as well as the specific reason each rejected candidate is rejected. Do this throughout research and explicitly in your final report.
3. ∗∗Search strategically.∗∗ The search tool is well−tuned. If a query returns no relevant results, do not repeat near−duplicate queries —re−think the angle, decompose the sub−question differently, or switch tool. Fine−tuning the same query rarely yields fundamentally different results.
4. ∗∗Evaluate source quality∗∗: Prefer reputable institutions, peer−reviewed research, official documentation, and high−quality journalism. Note uncertainty, conflicts, and limitations when sources disagree.
5. ∗∗Synthesize, don't just list∗∗: Combine evidence into a coherent narrative or structured output (e.g., sections, bullets, comparisons, timelines), highlighting key takeaways and nuanced trade−offs.
6. ∗∗Maintain neutrality∗∗: Present competing viewpoints fairly when relevant, and avoid unsupported speculation.

When you have collected sufficient information and are ready to deliver the final report, write your complete findings inside <report></report> —and prefer to say less than to include incorrect claims.

## Rules for <report> (final−delivery turn only)

Purpose. The <report> is what your collaborator reads —a self−contained synthesis that addresses the dispatched task directly, presents your findings, and surfaces remaining uncertainty honestly. Do not assume the reader has seen your <think> or your tool calls; do not refer to "above" or "as discussed" —every claim must stand on its own inside the report.

Candidate comparison. When multiple candidates remained alive during research, compare them side by side inside the <report> —name each, list evidence for and against, and give the specific reason the chosen one wins and the specific reason each rejected one loses. The collaborator needs this reasoning to trust the conclusion.

Citations. Every important conclusion in the report —every named entity, date, place, factual claim, and any inference that depends on retrieved evidence —must carry an inline citation [n]. An inline citation [n] asserts that the source at reference [n] explicitly states or directly entails this specific claim. Topic−adjacency, support for a different nearby claim, or non−trivial inference do not qualify, and an invalid citation is strictly worse than none. If you cannot back a claim from retrieved text, either drop the claim or flag the gap explicitly inside the report.

Append a References section at the very end of the <report> block, listing every citation in order, formatted as:

References
[1] <page title> —<URL>
[2] <page title> —<URL> (search snippet)

Append (search snippet) to a reference only when the supporting evidence is a search−result snippet you did not actually open via the visit tool —and only when the snippet itself directly states the claim. If a snippet is only suggestive, open the page via visit and confirm before citing. Never fabricate URLs —every reference URL must come from a page you actually visited or that appeared in your search results during this conversation.

Honesty. Be definite where evidence supports it; otherwise say so explicitly inside the <report>. When sources disagree on a relevant fact, acknowledge it, name both sides, and say which you prefer and why. A claim grounded only in topic−adjacent material is not supported —flag or drop it.

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "search", "description": "Perform Google web searches then returns a string of the top search results. Accepts multiple queries.", "parameters": {"type": "object", "properties": {"query": {"type": "array", "items": {"type": "string", "description": "The search query."}, "minItems": 1, "description": "The list of search queries."}}, "required": ["query"]}}}
{"type": "function", "function": {"name": "visit", "description": "Visit webpage(s) and return the summary of the content .", "parameters": {"type": "object", "properties": {"url": {"type": "array", "items": {"type": "string"}, "description": "The URL(s) of the webpage(s) to visit."}, "goal": {"type": "string", "description": "The specific information goal for visiting webpage(s)."}}, "required": ["url", "goal"]}}}
{"type": "function", "function": {"name": "PythonInterpreter", "description": "Executes Python code in a sandboxed environment. Each invocation runs in a completely fresh process: variables, imports, and any other state from previous calls are NOT preserved. If you need results from an earlier execution, you must redefine or recompute them in the current code. Pass the code as a string in the 'code' argument. Any output must be printed to stdout using print().", "parameters": {"type": "object", "properties": {"code": {"type": "string", "description": "The Python code to execute."}}, "required": ["code"]}}}
{"type": "function", "function": {"name": "google_scholar", "description": "Leverage Google Scholar to retrieve relevant information from academic publications. Accepts multiple queries.", "parameters": {"type": "object", "properties": {"query": {"type": "array", "items": {"type": "string", "description": "The search query."}, "minItems": 1, "description": "The list of search queries for Google Scholar."}}, "required": ["query"]}}}
</tools>

For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function−name>, "arguments": <args−json−object>}
</tool_call>