QUEST: Training Frontier Deep Research Agents with Fully Synthetic Tasks

Jian Xie[1*†] **Tianhe Lin**[1*] **Zilu Wang**[1*]
Yuting Ning[1] **Yuekun Yao**[1] **Tianci Xue**[1] **Zhehao Zhang**[1] **Zhongyang Li**[1]
Kai Zhang[1] **Yufan Wu**[1] **Shijie Chen**[1] **Boyu Gou**[1] **Mingzhe Han**[1]
Yifei Wang[2] **Vint Lee**[2] **Xinpeng Wei**[2] **Xiangjun Wang**[2]
Yu Su[1] **Huan Sun**[1]

[1]The Ohio State University [2]Amazon AGI SF Lab
[*]Core Contributors [†]Project Lead {xie.1741, su.809, sun.397}@osu.edu
Project Page: https://osu-nlp-group.github.io/QUEST

Abstract

Deep research agents extend the role of search engines from retrieving keyword-matched pages to synthesizing knowledge, fundamentally changing how humans interact with information. However, frontier systems remain proprietary, while existing open agents often generalize poorly across different task types, leaving unclear how to train a broadly capable deep research agent. We release **QUEST**, a family of open models (ranging from 2B to 35B) that serve as *general-purpose* deep research agents designed to handle a wide range of search tasks, with strong capabilities in *fact seeking*, *citation grounding*, and *report synthesis*. To build QUEST, we propose an effective training recipe combining mid-training, supervised fine-tuning, and reinforcement learning. Central to this recipe is a curated data synthesis pipeline based on unified rubric trees, which applies to different task types and enables synthesizing training data with verifiable rewards without human annotation. In addition, QUEST incorporates a built-in context management mechanism that enables effective long-horizon reasoning and knowledge synthesis. **Using only 8K synthesized tasks, QUEST approaches or even surpasses frontier closed-source agents across eight deep research benchmarks spanning diverse task types, and achieves the best overall performance among recent open-weight agents.** Notably, QUEST-35B achieves 30.7% on Mind2Web 2, outperforming OpenAI DeepResearch (28%) and Gemini DeepResearch (18%).



Figure 1: Comprehensive comparison across eight benchmarks that evaluate different capabilities: *fact seeking* (e.g., BrowseComp), *citation grounding* (e.g., Mind2Web 2), and *report synthesis* (e.g., DeepResearch Bench). For BrowseComp and BrowseComp-Plus, following prior work [1, 33, 4], we adopt the discard-all strategy. DR denotes Deep Research systems, such as OpenAI DeepResearch.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Web search is moving from manual information gathering toward autonomous evidence-seeking and synthesis. This shift builds on a progression of search interfaces: traditional search engines return ranked webpages for users to inspect, while retrieval-augmented generation (RAG) systems [16, 8] retrieve relevant documents and condition LLM responses on them. Deep research agents [24, 10, 34] push this paradigm further: given a complex information-seeking task, they can decompose it into intermediate goals, execute web queries, examine external sources, and synthesize citation-backed responses. By shifting the burden of web-scale information seeking from humans to autonomous agents, these systems promise to make complex research workflows more efficient and scalable. Yet building such agents is far from straightforward: they must learn not only to answer questions, but to search, verify, remember, and synthesize information over long horizons. However, despite rapid progress, the training recipes enabling these capabilities remain poorly understood, as state-of-the-art systems often rely on proprietary models, datasets, and training pipelines with limited transparency.

Recent efforts have begun to address this gap by developing deep research agents using open-weight models, data, and training strategies [34, 17, 18, 28, 7, 6]. However, existing approaches typically focus on specific use scenarios and fail to capture the full range of capabilities required for general-purpose deep research agents. In practice, deep research agents are expected to exhibit several complementary capabilities. For example, BrowseComp [36] requires locating information through multi-hop search (*fact seeking*), Mind2Web 2 [11] emphasizes that claims must be correctly supported by their attributed web evidence (*citation grounding*), and DeepResearch Bench [5] highlights the importance of synthesizing information into coherent reports (*report synthesis*). However, as shown in Table 1, existing open agents typically capture only a subset of these capabilities, and consequently underperform on benchmarks that require the others.

In this paper, we introduce QUEST, a family of open state-of-the-art models ranging from 2B to 35B parameters, designed as general-purpose deep research agents. The advancements of QUEST are attributed to three technical aspects: a novel **data synthesis pipeline** for constructing training tasks paired with tree-structured rubrics, a **context management strategy** that maintains compact summaries of information throughout long-horizon trajectories, and an effective **training pipeline** that optimizes the model using rubric-based signals. To comprehensively evaluate QUEST across different deep research capabilities, we adopt eight diverse benchmarks covering *fact seeking*, *citation grounding*, and *report synthesis*. Across these benchmarks, QUEST-35B shows consistently strong performance (See Figure 1), e.g., reaching 64.6% on BrowseComp, 30.7% on Mind2Web 2, and 48.2% on DeepResearch Bench, surpassing strong proprietary systems such as OpenAI DeepResearch.

Our contributions are as follows. *1)* A scalable data synthesis pipeline for deep research agent training, which produces QUEST-8K, a high-quality synthetic dataset where each instance pairs a complex query with a task-specific, verifiable rubric tree. *2)* A context management strategy for long-horizon deep research tasks, maintaining compact summaries of accumulated information across extended search trajectories. *3)* An effective training pipeline that combines mid-training, supervised fine-tuning, and reinforcement learning, using task-specific rubrics as optimization signals. *4)* Controlled studies that analyze the contribution of each training stage and the scaling behavior across model sizes (2B, 4B, 9B, up to 35B parameters). We find that even small models (e.g., QUEST-2B-SFT) achieve non-trivial performance, pointing to the potential of deployable small-scale research agents for privacy-sensitive settings, such as biomedical research and legal analysis.

2 Related Work

Deep research agents extend LLMs with web search, enabling them to support large-scale information seeking in domains such as finance and science [14, 32]. Motivated by this promise, many deep research systems have been developed, including OpenAI DeepResearch [22], Gemini DeepResearch [10], Claude Research [2], and Kimi Researcher [20]. Despite their effectiveness, these systems are based on proprietary models, and their data and training recipes are not publicly available. Recent open-weight efforts have begun to improve this situation. For example, Tongyi DeepResearch [34, 17, 38, 39] proposes a complete training framework that combines mid-training, supervised fine-tuning, and reinforcement learning to train open-weight LLMs [40] as agents. Other efforts, such as DR Tulu [28], RedSearcher [4], and OpenResearcher [18], further explore scaling agent training data through a range of automatic data generation strategies, such as model distillation

Table 1: **Comparison of recent open deep research agents.** QUEST offers multiple model scales, broader capability coverage, and a complete training pipeline built on fully synthetic tasks, unified rubric trees, and explicit context management. MT, SFT, and RL denote mid-training, supervised fine-tuning, and reinforcement learning.

Model	Scale	Capability	Task Synthesis	Verification	Context Management	Training Pipeline
Tongyi-DR [34]	30B	● ●	✓	Exact Match	✓	MT→SFT→RL
DR Tulu [28]	8B	● ●	✗	Rubric	✗	SFT→RL
OpenResearcher [18]	30B	●	✗	Exact Match	✗	SFT
REDSearcher [4]	30B	●	✓	Exact Match	✗	MT→SFT→RL
QUEST (ours)	2B–35B	● ● ●	✓	Rubric Tree	✓	MT→SFT→RL

● Fact Seeking ● Report Synthesis ● Citation Grounding ✓ supported ✗ not supported

and graph-based task synthesis. Despite recent progress, open-source deep research agents still lag behind proprietary systems in overall performance. In this work, as shown in Table 1, we propose an effective and comprehensive training recipe that combines data synthesis, context management, and staged post-training to further narrow this gap. We provide a more detailed discussion of deep research benchmarks in Appendix M.

3 Data Synthesis

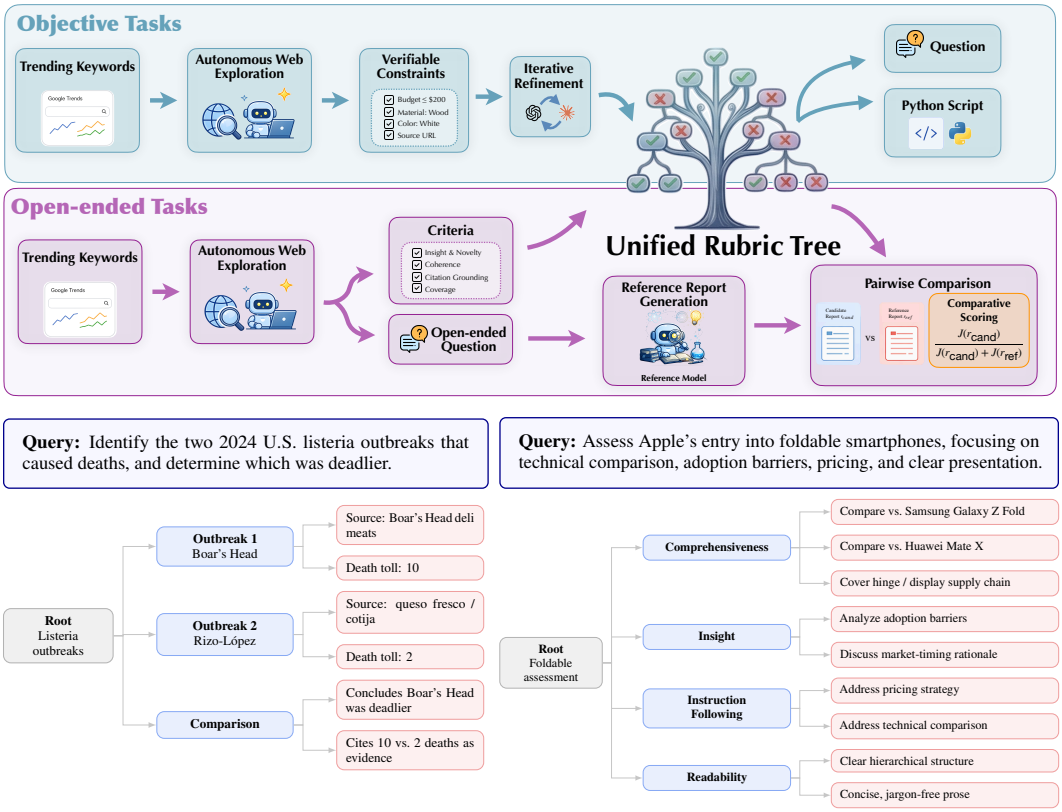
Deep research training data should contain challenging queries that require long-horizon search, webpage visiting, and reasoning. Existing deep research agents [17, 4] largely synthesize such data as complex questions with a single verifiable answer. This format is well-suited to BrowseComp-like tasks, e.g., "*Which architect designed the house where the author of The Catcher in the Rye lived for decades?*" where the answer is a hard-to-find entity.

However, a general-purpose deep research agent requires training data beyond such single-answer supervision. First, training on such data does not generalize to the broad range of tasks that deep research agents may encounter. Single-answer supervision mainly targets fact-seeking tasks with short final answers, whereas many deep research tasks require aggregating and reconciling information across multiple sources into coherent, source-supported outputs. Second, single-answer queries typically induce a binary correctness reward for reinforcement learning, which limits effective credit assignment. In contrast, deep research tasks often involve multiple evaluation criteria, such as source attribution and insightfulness. Exploiting such fine-grained signals as rewards can guide more effective optimization. We next describe how our synthesis pipeline addresses both issues.

3.1 Rubric Tree

At the core of our synthesis pipeline is a *rubric tree* introduced in Mind2Web 2 [11]: a hierarchical decomposition of the constraints that a valid answer should satisfy. Unlike Mind2Web 2, which relies on human-written tasks and human-refined evaluation scripts, QUEST constructs rubric trees fully synthetically through automatic generation followed by strict refinement and verification, making large-scale data synthesis possible. As shown at the bottom of Figure 2, the root node represents the overall score, which is aggregated from its child nodes, each corresponding to a task-specific criterion. Each leaf node represents a directly verifiable criterion, such as factual correctness or source attribution, and receives a binary score through automatic verification. Each internal node represents a higher-level constraint that is recursively decomposed into finer-grained child nodes, and receives a score aggregated from its children. We describe how rubric trees are automatically constructed in Section 3.2 and show complete examples in Appendix L.1.

This design naturally addresses the limitations of answer-centric QA supervision. By representing task-specific constraints rather than a unique ground-truth answer, rubric trees can accommodate tasks with unique answers, tasks with multiple valid solutions, and open-ended tasks requiring multi-aspect evaluation. This allows us to generate diverse training data under a unified framework. Moreover, the partial score of the root node in the rubric tree provides a fine-grained training signal beyond binary correctness, reflecting the degree to which a prediction satisfies the underlying constraints.



(a) Rubric tree of an objective task

(b) Rubric tree of an open-ended task

Figure 2: Data synthesis pipeline of QUEST. The top part illustrates the data generation pipeline. The bottom shows rubric-tree examples for an objective task and an open-ended task, together with corresponding queries. **Gray** nodes mark the root, **blue** nodes are intermediate branches that group sub-criteria, and **red** nodes are leaf-level constraints that can be directly verified against the response.

3.2 Data Generation Pipeline

Our rubric-tree-centric pipeline generates both objective and open-ended training data. Each instance consists of a complex query (i.e., a task description), a query-specific rubric tree, and an evaluation protocol. Given an agent's response to the query, the evaluation protocol maps the response to a score in $[0, 1]$ according to the rubric tree. Figure 2 presents an overview of our pipeline. We denote by G_{syn} the capable LLM used for both task generation and rubric-tree generation throughout the pipeline (i.e., Claude Sonnet 4.5 unless otherwise specified).

Objective Tasks. We first sample trending keywords from a pool crawled from Google Trends as topical seeds for downstream synthesis, ensuring that the resulting tasks remain both topically relevant and temporally diverse, while also better reflecting real-world user needs. For each set of trending keywords, we prompt G_{syn} to autonomously browse the web, collect relevant information, derive a set of verifiable constraints from the retrieved content , and organize them into a rubric tree. All rubric trees undergo iterative refinement and verification to ensure a correct tree structure and valid node definitions. Judged by Claude Sonnet 4.5, tasks whose rubric trees still cannot be resolved into a consistent, reliably evaluable structure are discarded.

Given a rubric tree, we prompt G_{syn} to translate it into a natural-language question and an executable Python evaluation script as its evaluation protocol, which programmatically checks each node against the agent's response and produces the final score, enabling fully automated evaluation at scale. We provide all the prompts used for objective task synthesis and the details of human examination in Appendix K and Appendix A.2.

Open-ended Tasks. Open-ended tasks basically follow the same keywords sampling and web exploration process, but differ in how rubrics are constructed. Instead of using fully task-specific

rubric trees, we fix the children of the root node across examples with four shared criteria following DeepResearch Bench [5]: instruction following, comprehensiveness, readability, and insight. The child nodes under each criterion are adaptive and contain task-specific nodes generated by G_{syn} conditioned on the question. For score aggregation, we assign a weight to each node using G_{syn}. To improve stability, we generate the weights three times and average them across runs.

The evaluation protocol of an open-ended task consists of three components: a reference report, a rubric-based judge, and a pairwise normalization rule. Given a task, we first prompt G_{syn} to produce a reference report r_{ref}. Then, for each candidate report r_{cand}, a judge model receives both the candidate and reference reports and scores them under each task-specific rubric node. The node-level scores are aggregated through the rubric tree, yielding weighted scores $J(r_{\mathrm{cand}})$ and $J(r_{\mathrm{ref}})$, where $J(\cdot) \in [0, 1]$. The final score is defined as $\mathrm{Score} = J(r_{\mathrm{cand}})/(J(r_{\mathrm{cand}}) + J(r_{\mathrm{ref}}))$, which measures the candidate report's quality relative to the reference report. A score above 0.5 indicates that the candidate has surpassed the quality of the reference report.

4 Context Management

A deep research agent searches, reads, and revises through multi-turn tool use before producing a final answer. As the context window fills with raw search results, visited pages, and intermediate reasoning traces, the agent's ability to attend to what matters most begins to degrade. Existing open-weight agents sidestep this problem by either capping the number of turns or relying on a sufficiently large context window to hold everything at once. By contrast, QUEST adopts a structured memory management module that compresses history into a compact summary, enabling the agent to reason over arbitrarily long horizons without context overflow.

Specifically, we use a structured JSON object, called CONTEXT STATE, to store the agent's history context. The CONTEXT STATE organizes accumulated knowledge into three buckets. *Trusted* entries contain facts that the agent has previously checked against retrieved source URLs based on its reasoning trace, allowing them to be directly reused without additional verification. *Untrusted* entries are claims contradicted by other sources, flagged with the reason for distrust. *Uncertain* entries are partially supported claims that require follow-up, each annotated with a URL to visit or a query to re-run. This structured state encodes the agent's epistemic state, allowing the resumed agent to distinguish between what it knows, doubts, and still needs to verify.

During an agent trajectory, context management is triggered once the context window usage exceeds a threshold, using a *Context Condenser* model G_{cond}. We instantiate G_{cond} with GPT-5-mini; see Appendix J for the prompt. The condenser takes as input the full raw history, which includes search queries and results, visited URLs with extracted content, reasoning traces, and any previously summarized memory, and then produces an updated CONTEXT STATE. After condensation, QUEST resumes in a fresh context window initialized with the updated CONTEXT STATE. Uncertain entries guide subsequent search and verification actions, while untrusted entries are deprioritized and not actively reused unless further verification is required. By reusing previously verified claims without redundant tool calls, the structured CONTEXT STATE enables coherent, citation-grounded knowledge synthesis across long-horizon research tasks.

5 Training

QUEST training pipeline includes three stages: mid-training (MT), supervised fine-tuning (SFT), and reinforcement learning (RL). Each stage plays a complementary role in shaping agent behaviors: MT adapts the base model to long-horizon search interaction, SFT teaches the agent to imitate high-quality tool-use trajectories, and RL optimizes the policy using rubric-based and fact-checking rewards.

5.1 Trajectory Collection

The SFT training requires full agent trajectories for queries generated from our data synthesis pipeline. We collected such trajectories by running a teacher model G_{traj} on the synthesized task set and evaluate each trajectory against its rubric tree. We use Tongyi DeepResearch as G_{traj}, and use GPT-5.2 to polish the final report for open-ended tasks.

Given a training query, we prompt G_{traj} to solve it and evaluate the output with the query-specific evaluation protocol. If the score meets or exceeds a threshold ϵ, we retain the generated trajectory as an SFT target for this query. For objective tasks, where the initial trajectory does not meet this threshold, we apply a reflection-based retry strategy [42]: the fine-grained evaluation result from the protocol is injected into the prompt as a hint, and G_{traj} is prompted to attempt the query again. The retried trajectory is retained if its evaluation score meets or exceeds ϵ. We set $\epsilon = 1$ for objective tasks and 0.475 for open-ended tasks. Our used prompts are reported in Appendix J.

Inline Citation Format. To improve the readability and verifiability of generated reports, we standardize all outputs into an inline citation format, where each factual claim is directly annotated with the URL of its supporting source. Specifically, for each trajectory, we employ GPT-5-mini to retrospectively insert inline citations into the original response, drawing on both the reasoning trace and the CONTEXT STATE—which records the full history of visited web content and extracted information. This also establishes the citation grounding output structure that the fact-checking reward in the RL stage will later reinforce.

5.2 Mid-Training

The MT stage consists of two auxiliary tasks that equip the base model with long-context understanding and awareness of the CONTEXT STATE structure, which the agent relies on in later stages. ***1)*** **Context Summarization.** Given a long history context, the model is required to produce a structured CONTEXT STATE JSON of the form described in Section 4. The summaries used as supervision targets are produced by GPT-5-mini during data synthesis and reused here directly, avoiding additional annotation cost while ensuring that the CONTEXT STATE format is consistent with the agent process during inference. ***2)*** **Relevant Information Extraction.** Given a raw HTML page and an extraction goal, the model is required to produce a goal-relevant summary that filters out navigation elements, advertisements, and off-topic content. This trains the model to attend selectively within long, noisy contexts, a capability that is prerequisite to effective web-based research but largely absent in models trained on clean document corpora.

Both tasks require no additional data collection. The relevant information extraction targets are derived directly from the output of the visit tool from trajectories in Section 5.1, and the context summarization targets are produced by G_{cond} throughout the pipeline, making MT a natural byproduct of the overall framework rather than an annotation effort. For more details, please refer to Appendix A.4.

5.3 Supervised Fine-Tuning

Following MT, the SFT stage trains the model on trajectories from Section 5.1 using the next-token prediction objective. Given a full training trajectory, we first transform it into multiple session-level training instances for efficient optimization. We define a session as a contiguous segment of interaction between two consecutive context condensation events. Concretely, each session contains all tokens from the current working context after the previous context condenser is applied, up to (but not including) the next condensation step. Thus, each completed trajectory is divided into multiple sessions, depending on how many times the context condenser is triggered. This design aligns the training unit with the agent's effective working context during inference. We provide more discussion about the advantages of session-level training over standard trajectory-level training in Appendix B.

5.4 Reinforcement Learning

In the RL stage, we use GRPO-style outcome-based reinforcement tuning [12]. The reward function combines the rubric-tree reward with the fact-checking reward defined below. We follow the same session-level training strategy as in the SFT stage.

Rubric-tree Rewards. The rubric-tree reward s_{rubric} of an agent's response is derived from the evaluation score calculated by the query-specific evaluation protocol. For objective tasks, we directly use this score as s_{rubric}. For open-ended tasks, the protocol produces a raw pairwise score comparing the agent response with a reference response, which we calibrate into ordered discrete reward levels by mapping higher pairwise scores to larger rewards.[1] This preserves the relative ordering of responses while making the reward signal more separable and less sensitive to small score fluctuations.

[1] In our implementation, pairwise scores at least 0.5 are assigned $s_{\text{rubric}} = 1.0$; scores in [0.475, 0.5), [0.45, 0.475), and [0.425, 0.45) are assigned 0.75, 0.5, and 0.25, respectively; and scores below 0.425 are assigned 0.

Fact-checking Rewards. For both objective and open-ended tasks, we add a fact-checking reward s_{fact} to encourage the agent to ground its responses in verifiable sources through its inline citations. For each response, we first extract cited fact–URL pairs from the answer, de-duplicate them, retrieve the referenced webpages, and use an LLM-based evaluator (i.e. GPT-5-mini) to label each citation as *supported*, *unsupported*, or *unknown*. We then calculate s_{fact} as the fraction of supported citations among citations with determinate labels, i.e., those labeled as either supported or unsupported. The final reward R is calculated by:

$$R = 0.75 \cdot s_{\text{rubric}} + 0.25 \cdot s_{\text{fact}} \cdot s_{\text{rubric}}, \tag{1}$$

where $s_{\text{rubric}}, s_{\text{fact}} \in [0, 1]$. The fact-checking term is scaled by the rubric-tree reward, so that well-supported citations cannot compensate for poor task completion, and the fact-checking contribution vanishes when the task is entirely failed. Responses without inline citations receive $s_{\text{fact}} = 0$.

Reward and Advantage Propagation. For each input prompt, we sample a group of rollouts. For each rollout i, we compute a scalar reward R_i based on its full response and assign this reward to all sessions derived from the same rollout. The advantage for each rollout is computed by normalizing its reward within the group:

$$A_i = \frac{R_i - \mu_g}{\sigma_g + \epsilon}, \tag{2}$$

where μ_g and σ_g are the mean and standard deviation computed over unique rollouts in group g. When the context condenser produces multiple sessions per rollout, these are deduplicated during normalization so that all sessions derived from the same rollout share a single advantage value. We then optimize the policy using the standard GRPO objective [29].

6 Experiments

6.1 Setup

Training. Given that most existing open deep research agents chose a base model with around 30B parameters, we take Qwen3.5-35B-A3B [27] as our default base model, and apply our full training recipe to obtain QUEST-35B. We use 8K instances generated through our data synthesis pipeline for SFT and RL training in total. Please refer to Appendix A for more details. During training and inference, QUEST is equipped with a practical tool set, including `Google Search` for retrieving relevant online results for a given query, `Visit` for browsing webpage content, `Python Interpreter` for computation, and `Google Scholar` for accessing academic publications. For the summarization model used in the Visit tool, we adopt GPT-5-mini, which also serves as the backbone model of our context condenser. For more details of training and tools, please refer to Appendix F and Appendix C.

Benchmark Evaluation. We evaluate QUEST on eight deep research benchmarks: six objective benchmarks, including BrowseComp (BC; [36]), Mind2Web 2 (M2W2; [11]), Humanity's Last Exam (HLE; [26]), BrowseComp-Plus (BC-Plus; [3]), WideSearch [37], and GAIA [19], and two open-ended benchmarks, DeepResearch Bench (DRB; [5]) and LiveResearchBench (LRB; [35]). For the results in Figure 1, we use the official full test sets, except for HLE and GAIA, where we evaluate on the text-only versions. Unless otherwise specified, ablation studies and analyses of BC, HLE, WideSearch, and BC-Plus are conducted on sampled subsets due to the high cost of full set evaluation. For details on the evaluation metrics and subset splits of the benchmarks, please refer to Appendix G.

Baselines. We compare QUEST-35B against prior deep research agents in two categories: (1) *Frontier proprietary agents*, including OpenAI DeepResearch (OpenAI-DR) [24], GPT-5 [31], Claude Opus 4.5, and Gemini 3 Pro; and (2) *open weight agents at a similar scale*, including Tongyi DeepResearch (Qwen3-30B-A3B) [34], OpenResearcher (Nemotron-3-Nano-30B-A3B) [18], and DR Tulu (Qwen3-8B) [28], where parentheses indicate the base models. Since work for these models reports results on only a subset of our chosen benchmarks, we run Tongyi-DR and OpenResearcher ourselves to fill in missing numbers for a complete comparison. For DR Tulu, we evaluate it on Mind2Web 2 ourselves and find that it is good at open-ended tasks that focus on report synthesis, but not at objective tasks, so we do not further evaluate it on other benchmarks. Although RedSearcher [4] is claimed to be open-weight, it is not publicly released at submission time; we therefore do not report its results.

Table 2: Comparison of proprietary and open deep research agents across eight benchmarks. Results marked with † are obtained from our implementation using the official evaluation scripts on the full set. * denotes results obtained using the discard-all strategy [1, 33, 4] on the full set for fair comparison with proprietary agents. The highest score is marked in bold, and the second is underlined.

Model	Objective Tasks						Open-Ended Tasks	
	BC	HLE	GAIA	M2W2	WideSearch	BC-Plus	LRB	DRB
Frontier Proprietary Agents								
GPT-5	59.9	35.2	<u>76.4</u>	-	54.0	<u>71.7</u>	<u>73.1</u>	-
Claude Opus 4.5	**67.8**	<u>43.2</u>	-	-	**76.2**	**83.0**	-	**50.6**
Gemini 3 Pro	59.2	**45.8**	-	-	57.0	-	-	<u>49.6</u>
OpenAI-DR	51.5	26.6	67.4	28.0	-	-	-	47.0
Open Weight Agents								
Tongyi-DR	43.4	32.9	70.9	16.7†	37.3†	44.5	56.3†	40.5
OpenResearcher	26.3	19.6†	64.1	14.8†	19.2†	54.8	61.3†	35.4†
DR Tulu	-	-	-	1.6†	-	-	-	45.5
QUEST-30B	37.0	24.6	69.0	<u>28.6</u>	54.2	48.2	**74.1**	45.3
QUEST-35B	45.5 / <u>64.6</u>*	37.2	**80.8**	**30.7**	<u>60.6</u>	61.0 / 69.5*	68.2	48.2

6.2 Results

QUEST-35B establishes a new state of the art among open-weight agents and approaches proprietary agents. As shown in Table 2, across almost all evaluated benchmarks, QUEST-35B consistently achieves new state-of-the-art performance among open-weight agents at around the 30B scale (e.g., Tongyi-DR, RedSearcher). On several benchmarks, QUEST-35B even matches or slightly surpasses closed-source agents, including OpenAI-DR on DeepResearch Bench (48.2% vs. 47.0%) and Mind2Web 2 (30.7% vs. 28.0%), as well as GPT-5 on GAIA (80.8% vs. 76.4%). These results substantially narrow the gap between open-weight and proprietary deep research systems. These results highlight the effectiveness of our training recipe.

Existing open-weight agents fall short on out-of-distribution tasks, while QUEST recipe leads to overall best performance. For a fair comparison with open-weight models at exactly the 30B scale, we further apply the QUEST recipe to Qwen3-30B-A3B [40], resulting in our QUEST-30B model. We directly compare QUEST-30B with Tongyi-DR and OpenResearcher. At the same parameter scale, our model achieves the best performance on 4 out of 8 benchmarks, including tasks that require strong *citation grounding* and *report synthesis* capabilities, such as Mind2Web 2 and DeepResearch Bench. On the remaining benchmarks, Tongyi-DR performs best on BrowseComp, HLE, and GAIA, which rely heavily on the *fact seeking* ability, a capability explicitly supported by Tongyi-DR's single-answer synthetic data. In contrast, OpenResearcher performs best on BrowseComp-Plus, a fully offline benchmark that closely matches its data synthesis recipe. These results suggest that the capabilities exhibited by a deep research agent are shaped by its data synthesis recipe, and our recipe provides the best coverage of these capabilities.

6.3 Insights from Controlled Studies

Best Performance Obtained by Combining MT, SFT, and RL. To better understand the impact of our training stages and training data on final performance, we systematically analyze QUEST-35B under different stages. We consider four model variants: *1) Vanilla*, the base checkpoint identical to Qwen3.5-35B-A3B; *2) SFT*, obtained by applying SFT to the Vanilla model; *3) MT+SFT*, obtained by first performing MT on the Vanilla model followed by SFT; *4) MT+SFT+RL*, our full training recipe, which further applies RL on top of MT+SFT. Figure 3 presents the results.

We observe that the effect of each training stage is nuanced across benchmarks. Simple SFT improves most objective benchmarks, but degrades open-ended performance relative to the Vanilla baseline. The only objective benchmark that SFT hurts is BrowseComp-Plus, because it allows only a single `search` tool during evaluation while heavy SFT makes the model more likely to call disallowed tools due to overfitting to the training-time tool-use pattern. MT further improves performance on top of SFT overall, suggesting that the benefit of our auxiliary tasks generalizes across benchmarks. Interestingly, RL substantially improves performance on open-ended tasks, but slightly sacrifices performance on HLE and GAIA. We hypothesize that HLE and GAIA require not only search, but

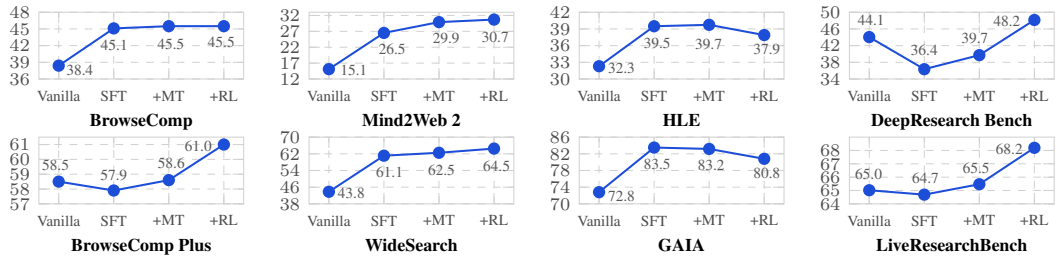


Figure 3: Ablation study on the effect of training stages. Score progression of QUEST-35B across three training stages. All results are based on agents using the context condenser. Details on the evaluation of the vanilla model are provided in Appendix H.

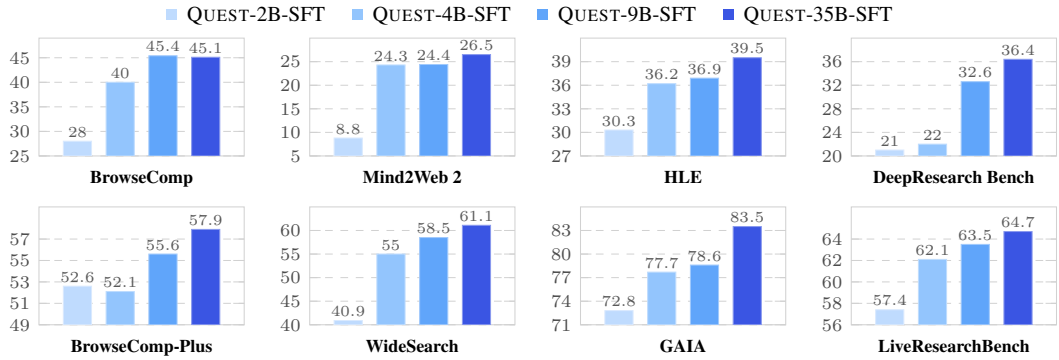


Figure 4: Benchmark performance across different QUEST model sizes.

strong expert-level reasoning and in-depth analysis. Since our RL objective goals are primarily optimized for deep search behaviors, including report synthesis like readability-oriented responses, this specialization may partially weaken the model's general reasoning capability, a phenomenon similar to the *alignment tax* [25]. Overall, *MT+SFT+RL* performs best among the compared variants, demonstrating that general-purpose deep research benefits from all three training stages.

Small LLMs Can be Strong Deep Research Agents. Developing capable agents based on smaller models is practically meaningful, especially for compute-constrained scenarios or high-stakes domains such as healthcare where hosting agents locally is desired due to privacy concerns. To study this, we conduct SFT on the 2B, 4B, and 9B versions of Qwen3.5 [27] using the same training data and inference configurations as our previous experiments, which allows us to study how performance changes as model size scales.

Surprisingly, as shown in Figure 4, downscaling model size does not lead to substantial degradation on most benchmarks, although overall performance generally improves with larger LLMs. Even a 2B model achieves competitive performance on *fact seeking* benchmarks such as HLE and GAIA, matching or even outperforming models like OpenAI-DeepResearch and o3 [23, 24] (e.g., o3 scores 24.9% on HLE and 70.5% on GAIA). This highlights the robustness of our synthetic data across model scales, as SFT alone yields strong performance even with a 2B model. However, this advantage does not fully transfer to benchmarks that heavily require *report synthesis*, such as DeepResearch Bench and LiveResearchBench, where the 2B model lags far behind. Overall, our findings suggest a promising direction for deploying small LLMs for *fact seeking* search tasks, while also highlighting the challenge of extending their capabilities to more open-ended settings.

7 Conclusion

In this paper, we introduce QUEST, a model family as general-purpose deep research agents capable of solving diverse deep research tasks that require multiple core capabilities. Our best model QUEST-35B achieves the best performance across eight benchmarks among all open deep research agents, and is competitive with frontier closed agents. We present a comprehensive recipe for training effective deep research agents, covering data synthesis, context management, and the training pipeline. To support different deployment needs, QUEST is released at multiple scales, from 2B to 35B. We hope this work provides a strong open foundation for advancing deep research agents.

Acknowledgments

The authors thank colleagues from the OSU NLP group for their constructive feedback. This research was supported in part by NSF CAREER Awards #1942980 and #2443149, the Alfred P. Sloan Research Fellowship, and supported by Amazon, Cisco, and Intuit. We also acknowledge the computational resources provided by the Ohio Supercomputer Center [21]. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notice herein.

References

[1] Anthropic. System card: Claude opus 4.5. `https://www-cdn.anthropic.com/bf10f64990cfda0ba858290be7b8cc6317685f47.pdf`, November 2025.

[2] Anthropic. Claude takes research to new places. `https://claude.com/blog/research`, 2025.

[3] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

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

[5] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench: A comprehensive benchmark for deep research agents. *arXiv preprint arXiv:2506.11763*, 2025.

[6] Yuwen Du, Rui Ye, Shuo Tang, Keduan Huang, Xinyu Zhu, Yuzhu Cai, and Siheng Chen. Openseeker-v2: Pushing the limits of search agents with informative and high-difficulty trajectories. *arXiv preprint arXiv:2605.04036*, 2026.

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

[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

[9] GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding,

Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

[10] Google DeepMind. Gemini 1.5 pro and deep research capabilities. https://gemini.google.com/, February 2025.

[11] Boyu Gou, Zanming Huang, Yuting Ning, Yu Gu, Michael Lin, Weijian Qi, Andrei Kopanev, Botao Yu, Bernal Jimenez Gutierrez, Yiheng Shu, et al. Mind2web 2: Evaluating agentic search with agent-as-a-judge. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[12] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[13] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *COLING'20*, pages 6609–6625, 2020.

[14] Liang Hu, Jianpeng Jiao, Jiashuo Liu, Yanle Ren, Zhoufutu Wen, Kaiyuan Zhang, Xuanliang Zhang, Xiang Gao, Tianci He, Fei Hu, et al. Finsearchcomp: Towards a realistic, expert-level evaluation of financial search and reasoning. *arXiv preprint arXiv:2509.13160*, 2025.

[15] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

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

[17] Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025.

[18] Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. Openresearcher: A fully open pipeline for long-horizon deep research trajectory synthesis, 2026. URL https://arxiv.org/abs/2603.20278.

[19] Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023.

[20] Moonshot AI. Kimi-researcher: End-to-end rl training for emerging agentic capabilities. https://moonshotai.github.io/Kimi-Researcher/, 2025.

[21] Ohio Supercomputer Center. Ohio Supercomputer Center, 1987. URL https://ror.org/01apna436.

[22] OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, 2025.

[23] OpenAI. Introducing o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/, April 2025.

[24] OpenAI. Introducing deep research. https://openai.com/index/introducing-deep-research/, February 2025.

[25] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

[26] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

[27] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[28] Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. Dr tulu: Reinforcement learning with evolving rubrics for deep research, 2025. URL https://arxiv.org/abs/2511.19399.

[29] 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.

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

[31] Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, Andrey Mishchenko, Andy Applebaum, Andy Rogerson, Ann Rajan, Annie Wei, Anoop Kotha, Anubha Srivastava, Anushree Agrawal, Arun Vijayvergiya, Ashley Tyra, Ashvin Nair, Avi Nayak, Ben Eggers, Bessie Ji, Beth Hoover, Bill Chen, Blair Chen, Boaz Barak, Borys Minaiev, Botao Hao, Bowen Baker, Brad Lightcap, Brandon McKinzie, Brandon Wang, Brendan Quinn, Brian Fioca, Brian Hsu, Brian Yang, Brian Yu, Brian Zhang, Brittany Brenner, Callie Riggins Zetino, Cameron Raymond, Camillo Lugaresi, Carolina Paz, Cary Hudson, Cedric Whitney, Chak Li, Charles Chen, Charlotte Cole, Chelsea Voss, Chen Ding, Chen Shen, Chengdu Huang, Chris Colby, Chris Hallacy, Chris Koch, Chris Lu, Christina Kaplan, Christina Kim, CJ Minott-Henriques, Cliff Frey, Cody Yu, Coley Czarnecki, Colin Reid, Colin Wei, Cory Decareaux, Cristina Scheau, Cyril Zhang, Cyrus Forbes, Da Tang, Dakota Goldberg, Dan Roberts, Dana Palmie, Daniel Kappler, Daniel Levine, Daniel Wright, Dave Leo, David Lin, David Robinson, Declan Grabb, Derek Chen, Derek Lim, Derek Salama, Dibya Bhattacharjee, Dimitris Tsipras, Dinghua Li, Dingli Yu, DJ Strouse, Drew Williams, Dylan Hunn, Ed Bayes, Edwin Arbus, Ekin Akyurek, Elaine Ya Le, Elana Widmann, Eli Yani, Elizabeth Proehl, Enis Sert, Enoch Cheung, Eri Schwartz, Eric Han, Eric Jiang, Eric Mitchell, Eric Sigler, Eric Wallace, Erik Ritter, Erin Kavanaugh, Evan Mays, Evgenii Nikishin, Fangyuan Li, Felipe Petroski Such, Filipe de Avila Belbute Peres, Filippo Raso, Florent Bekerman, Foivos Tsimpourlas, Fotis Chantzis, Francis Song, Francis Zhang, Gaby Raila, Garrett McGrath, Gary Briggs, Gary Yang, Giambattista Parascandolo, Gildas Chabot, Grace Kim, Grace Zhao, Gregory Valiant, Guillaume Leclerc, Hadi Salman, Hanson Wang, Hao Sheng, Haoming Jiang, Haoyu Wang, Haozhun Jin, Harshit Sikchi, Heather Schmidt, Henry Aspegren, Honglin Chen, Huida Qiu, Hunter Lightman, Ian Covert, Ian Kivlichan, Ian Silber, Ian Sohl, Ibrahim Hammoud, Ignasi Clavera, Ikai Lan, Ilge Akkaya, Ilya Kostrikov, Irina

Kofman, Isak Etinger, Ishaan Singal, Jackie Hehir, Jacob Huh, Jacqueline Pan, Jake Wilczynski, Jakub Pachocki, James Lee, James Quinn, Jamie Kiros, Janvi Kalra, Jasmyn Samaroo, Jason Wang, Jason Wolfe, Jay Chen, Jay Wang, Jean Harb, Jeffrey Han, Jeffrey Wang, Jennifer Zhao, Jeremy Chen, Jerene Yang, Jerry Tworek, Jesse Chand, Jessica Landon, Jessica Liang, Ji Lin, Jiancheng Liu, Jianfeng Wang, Jie Tang, Jihan Yin, Joanne Jang, Joel Morris, Joey Flynn, Johannes Ferstad, Johannes Heidecke, John Fishbein, John Hallman, Jonah Grant, Jonathan Chien, Jonathan Gordon, Jongsoo Park, Jordan Liss, Jos Kraaijeveld, Joseph Guay, Joseph Mo, Josh Lawson, Josh McGrath, Joshua Vendrow, Joy Jiao, Julian Lee, Julie Steele, Julie Wang, Junhua Mao, Kai Chen, Kai Hayashi, Kai Xiao, Kamyar Salahi, Kan Wu, Karan Sekhri, Karan Sharma, Karan Singhal, Karen Li, Kenny Nguyen, Keren Gu-Lemberg, Kevin King, Kevin Liu, Kevin Stone, Kevin Yu, Kristen Ying, Kristian Georgiev, Kristie Lim, Kushal Tirumala, Kyle Miller, Lama Ahmad, Larry Lv, Laura Clare, Laurance Fauconnet, Lauren Itow, Lauren Yang, Laurentia Romaniuk, Leah Anise, Lee Byron, Leher Pathak, Leon Maksin, Leyan Lo, Leyton Ho, Li Jing, Liang Wu, Liang Xiong, Lien Mamitsuka, Lin Yang, Lindsay McCallum, Lindsey Held, Liz Bourgeois, Logan Engstrom, Lorenz Kuhn, Louis Feuvrier, Lu Zhang, Lucas Switzer, Lukas Kondraciuk, Lukasz Kaiser, Manas Joglekar, Mandeep Singh, Mandip Shah, Manuka Stratta, Marcus Williams, Mark Chen, Mark Sun, Marselus Cayton, Martin Li, Marvin Zhang, Marwan Aljubeh, Matt Nichols, Matthew Haines, Max Schwarzer, Mayank Gupta, Meghan Shah, Melody Y. Guan, Melody Huang, Meng Dong, Mengqing Wang, Mia Glaese, Micah Carroll, Michael Lampe, Michael Malek, Michael Sharman, Michael Zhang, Michele Wang, Michelle Pokrass, Mihai Florian, Mikhail Pavlov, Miles Wang, Ming Chen, Mingxuan Wang, Minnia Feng, Mo Bavarian, Molly Lin, Moose Abdool, Mostafa Rohaninejad, Nacho Soto, Natalie Staudacher, Natan LaFontaine, Nathan Marwell, Nelson Liu, Nick Preston, Nick Turley, Nicklas Ansman, Nicole Blades, Nikil Pancha, Nikita Mikhaylin, Niko Felix, Nikunj Handa, Nishant Rai, Nitish Keskar, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Oona Gleeson, Pamela Mishkin, Patryk Lesiewicz, Paul Baltescu, Pavel Belov, Peter Zhokhov, Philip Pronin, Phillip Guo, Phoebe Thacker, Qi Liu, Qiming Yuan, Qinghua Liu, Rachel Dias, Rachel Puckett, Rahul Arora, Ravi Teja Mullapudi, Raz Gaon, Reah Miyara, Rennie Song, Rishabh Aggarwal, RJ Marsan, Robel Yemiru, Robert Xiong, Rohan Kshirsagar, Rohan Nuttall, Roman Tsiupa, Ronen Eldan, Rose Wang, Roshan James, Roy Ziv, Rui Shu, Ruslan Nigmatullin, Saachi Jain, Saam Talaie, Sam Altman, Sam Arnesen, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Sarah Yoo, Savannah Heon, Scott Ethersmith, Sean Grove, Sean Taylor, Sebastien Bubeck, Sever Banesiu, Shaokyi Amdo, Shengjia Zhao, Sherwin Wu, Shibani Santurkar, Shiyu Zhao, Shraman Ray Chaudhuri, Shreyas Krishnaswamy, Shuaiqi, Xia, Shuyang Cheng, Shyamal Anadkat, Simón Posada Fishman, Simon Tobin, Siyuan Fu, Somay Jain, Song Mei, Sonya Egoian, Spencer Kim, Spug Golden, SQ Mah, Steph Lin, Stephen Imm, Steve Sharpe, Steve Yadlowsky, Sulman Choudhry, Sungwon Eum, Suvansh Sanjeev, Tabarak Khan, Tal Stramer, Tao Wang, Tao Xin, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Degry, Thomas Shadwell, Tianfu Fu, Tianshi Gao, Timur Garipov, Tina Sriskandarajah, Toki Sherbakov, Tomek Korbak, Tomer Kaftan, Tomo Hiratsuka, Tongzhou Wang, Tony Song, Tony Zhao, Troy Peterson, Val Kharitonov, Victoria Chernova, Vineet Kosaraju, Vishal Kuo, Vitchyr Pong, Vivek Verma, Vlad Petrov, Wanning Jiang, Weixing Zhang, Wenda Zhou, Wenlei Xie, Wenting Zhan, Wes McCabe, Will DePue, Will Ellsworth, Wulfie Bain, Wyatt Thompson, Xiangning Chen, Xiangyu Qi, Xin Xiang, Xinwei Shi, Yann Dubois, Yaodong Yu, Yara Khakbaz, Yifan Wu, Yilei Qian, Yin Tat Lee, Yinbo Chen, Yizhen Zhang, Yizhong Xiong, Yonglong Tian, Young Cha, Yu Bai, Yu Yang, Yuan Yuan, Yuanzhi Li, Yufeng Zhang, Yuguang Yang, Yujia Jin, Yun Jiang, Yunyun Wang, Yushi Wang, Yutian Liu, Zach Stubenvoll, Zehao Dou, Zheng Wu, and Zhigang Wang. Openai gpt-5 system card, 2026. URL `https://arxiv.org/abs/2601.03267`.

[32] Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. AI-researcher: Autonomous scientific innovation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL `https://openreview.net/forum?id=kQWyOYUAC4`.

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

[34] Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical

report. *arXiv preprint arXiv:2510.24701*, 2025.

[35] Jiayu Wang, Yifei Ming, Riya Dulepet, Qinglin Chen, Austin Xu, Zixuan Ke, Frederic Sala, Aws Albarghouthi, Caiming Xiong, and Shafiq Joty. Liveresearchbench: A live benchmark for user-centric deep research in the wild, 2026. URL https://arxiv.org/abs/2510.14240.

[36] 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*, 2025.

[37] Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. Widesearch: Benchmarking agentic broad info-seeking, 2025. URL https://arxiv.org/abs/2508.07999.

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

[39] Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.

[40] 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.

[41] 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 *EMNLP'18*, pages 2369–2380, 2018.

[42] Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025.

Appendix

A Data Construction Details

We provide the detailed data distribution across different training stages in Table 3. Below, we describe how these data are constructed.

Table 3: Data distribution across different training stages in QUEST.

Stage	Type	Data Category		
		#Task	#Traj.	#Session
MT	Context Summarization	309,346	–	–
	Relevant Information Extraction	1,052,663	–	–
SFT	Objective	5,070	19,435	39,861
	Open-ended	1,958	4,485	11,903
RL	Objective	864	–	–
	Open-ended	269	–	–

A.1 Task Synthesis

We adopt a strict task filtering strategy to ensure that all the generated tasks are high-quality and the evaluation trees are reliable. This strict filtering policy leads to a low retention rate. We report the number of remaining tasks at each stage in Table 4.

Table 4: Number of retained tasks after each filtering stage.

Filtering Stage	Number of Tasks
Objective Tasks	
Initial generated objective tasks	17,000
After rubric refinement	8,737
After rubric tree verification	6,230
After removing tasks with erroneous Python scripts	5,934
Open-ended Tasks	
Initial generated open-ended tasks	3,000
After removing tasks with invalid format	2,856
After removing unqualified tasks	2,498
After removing tasks with low reference report scores	2,227

A.2 Manual Examination of the Python Evaluation Scripts for Objective Tasks

To examine the correctness of the Python evaluation scripts generated by G_{syn}, we sample 50 tasks from our datasets and employ four graduate students majoring in computer science as human annotators for manual evaluation. The annotators are given the task description and the rubric tree, along with detailed instructions covering our assessment objectives, the organizational structure of the rubric tree, evaluation strategies, and core evaluation toolkit functionalities (such as Extractor and Verifier functions and rubric tree management utilities). We also provide examples of common mistakes and tips to help annotators identify potential errors. Then they are required to verify whether the Python evaluation script aligns with the rubric tree and assess the program's executability.

Out of the 50 sampled tasks, only 2 cases are found to suffer from non-executable script issues, such as runtime exceptions and logical execution breakdowns that prevent the evaluation pipeline from running normally.

In terms of rubric-related errors in the generated scripts, we identify 6 scripts that contain such errors, and summarize three typical problematic patterns. First, some scripts introduce redundant rubric nodes that are never required or mentioned in the original task description, adding unnecessary

evaluation constraints irrelevant to the assessment goal. Second, some scripts perform verification merely based on the literal content of the final answer, rather than strictly verifying the authenticity of the content in accordance with the evaluation criteria. Third, some scripts fail to explicitly assign a negative judgment when the target answer contained no cited URLs, lacking clear judgment criteria for this scenario.

Despite imperfections, most generated scripts can accurately interpret task requirements, faithfully implement rubric rules, and run without errors, effectively supporting our automated evaluation at scale.

A.3 Human Audit of the Final Training Data

To complement the automated filtering and the script examination above, we conduct a component-wise human audit of the final SFT dataset. Before annotation, all annotators complete a calibration exercise on a small shared subset of examples to align the annotation criteria, after which we verify annotation consistency; the resulting agreement exceeds 95% across all evaluated criteria. All audited examples are randomly sampled from the final SFT dataset.

Objective tasks. Ten annotators inspect 100 objective tasks. The audit is fine-grained: a single example can contain more than 100 rubric nodes and over 2,000 lines of evaluator code, requiring more than one hour to inspect. For each example, annotators assess (D1) whether every rubric node is valid, relevant, and free of hard-coded answers; (D2) whether the question is clear, solvable, and faithful to the rubric tree; (D3) whether the executable evaluator implements the rubric tree and the task requirements; and (D4) whether the final answer of the retained trajectory is correct.

Open-ended tasks. Five annotators inspect 100 open-ended tasks, assessing (D1) whether the task is clear and suitable for an evidence-backed long-form answer; (D2) whether the rubric is faithful, factual, relevant, non-redundant, and covers the key aspects of the task; and (D3) whether the final report of the retained trajectory is correct and comprehensive, with high-quality citations and coherent writing.

Table 5: Human audit of 100 objective and 100 open-ended tasks randomly sampled from the final SFT dataset.

Criterion	Pass Rate
Objective Tasks	
Rubric-tree quality (D1)	99%
Question quality (D2)	99%
Evaluator alignment (D3)	97%
Trajectory correctness (D4)	100%
Open-ended Tasks	
Task quality (D1)	98%
Rubric quality (D2)	96%
Trajectory correctness (D3)	98%

As shown in Table 5, all pass rates exceed 95%. For objective tasks, the errors comprise one hard-coded rubric node (D1), one question that omits a constraint specified by its rubric tree (D2), and three evaluator-alignment failures (D3): the same hard-coding issue and two evaluators that add constraints absent from their rubric trees. Since the natural-language question is generated from the rubric tree rather than the reverse, the rubric tree defines the task intent; the single D2 failure is the only case in which question generation drifted from it. The evaluator alignment rate is higher than that in Appendix A.2 because rejection sampling removes many low-quality intermediate examples before the final SFT dataset is formed. For open-ended tasks, the failures comprise two underspecified tasks (D1); four rubric issues (D2), namely one missing aspect, two partially redundant criteria, and one unnecessary requirement; and two trajectory issues (D3), namely one report that omits key criteria and one that contains repetitive paragraphs.

Verifier accuracy. We further compare 110 judgments produced by the generated Python evaluators with human annotations. The evaluators agree with humans in 108 of the 110 cases (98.2%). Both errors are false acceptances, i.e., incorrect answers judged as correct, and no correct answer is rejected. Since each evaluator decomposes the task into atomic rubric-node checks, an error at a single node

affects only the corresponding branch, while the remaining branches still provide valid partial-credit signals.

A.4 Mid-Training

Context Summarization. The context summarization data is collected from the distillation process used for SFT data preparation and from several preliminary RL experiments. We filter out low-quality instances caused by API errors or other implementation bugs. To avoid data leakage, we do not include any context summarization data from benchmark evaluations. This results in a final set of 300K context summarization instances.

Relevant Information Extraction. The relevant information extraction data is obtained from our Visit Tool cache, which stores the raw HTML content of all visited webpages, along with their corresponding extraction goals and extracted content. Initially, this cache contains 4 million raw triplets of the form (raw HTML content, goal, extracted content). To improve training efficiency and reduce redundancy, we filter repetitive instances, such as cases where multiple similar extraction goals correspond to the same webpage. Specifically, we remove redundant instances by grouping entries from the same webpage and filtering similar extraction goals using Jaccard similarity with a threshold of 0.1.

A.5 Supervised Fine-tuning

For rejection sampling, we roll out each task five times. For reflection-based retry, we roll out each failed trajectory three additional times. All the maximum tool calls are set to 100 turns. And the memory threshold is set to 16K.

A.6 Reinforcement Learning

Tasks that cannot be successfully completed by G_{traj} in any rollout are reserved for reinforcement learning, resulting in 864 objective RL tasks. For open-ended tasks, we manually split the data into SFT and RL sets.

B Session-level Training

Memory Efficiency. Treating an entire trajectory as a single training sequence requires the full context to reside in GPU memory simultaneously, which is infeasible for long-horizon trajectories. Decomposing trajectories into shorter sessions substantially reduces peak memory usage, allowing training to scale to longer trajectories without being limited by GPU memory. In practice, this allows us to train on trajectories exceeding 200K tokens using only 16 H100 GPUs, which would be infeasible under standard trajectory-level training.

Unbounded Context Extrapolation at Inference Time. Session-level training exposes the model to variable-length inputs under a structured decomposition, effectively decoupling per-session training context length from overall trajectory length. Combined with our memory management mechanism, this enables the agent to extrapolate to arbitrarily long research sessions at inference time, far beyond the context lengths seen during training and even beyond the model's native context window. For example, although QUEST is trained on trajectories of at most 100 turns, it is able to generalize to interactions of more than 200 turns at inference time.

C Infrastructure

Training a deep research agent at scale introduces infrastructure challenges that go beyond standard LLM post-training. The agent must interact with live web content throughout data synthesis, SFT trajectory collection, and RL rollout—each of which involves hundreds of search queries and URL visits executed in parallel. Without careful infrastructure design, these external calls become prohibitively expensive. QUEST addresses this through a dual-cache system that intercepts all search and visit operations before they reach the live API. Repeated queries and URL visits during data construction, SFT, and RL rollout are served from cache rather than billed as API calls.

Search and Scholar Cache. We build the search and scholar cache online during data synthesis and training: whenever a live search API call is made, the query and returned results are written into a persistent cache for future reuse. Every new search query first undergoes exact-match lookup against this cache. On a cache hit, results are returned immediately without an API call. On a miss, the query is passed to FAISS for semantic similarity search: if a sufficiently similar query with cosine similarity above threshold already exists in the cache, its results are returned as a proxy. Only when both exact and semantic lookup fail does the system fall through to the live search API, at which point the result is written back to the cache for future use. We use Serper[2] as the live API.

Visit Cache. Visit tool follows a simpler exact-match policy. Each URL is checked against a persistent cache keyed by the full URL string. On a hit, the stored page content is returned directly. On a miss, the live page accessed is fetched and cached. This eliminates redundant fetches of the same page across different training runs. We use Jina[3] as the live API.

Fully Asynchronous Rollout, Evaluation, and Training for RL. Built on top of VERL's fully asynchronous policy optimization framework [30], which decouples rollout and training, we further extend the pipeline by making evaluation fully asynchronous. Specifically, beyond overlapping the Rollouter and the Trainer, we offload time-consuming task-specific evaluation to a separate asynchronous evaluator path implemented via reward-loop workers. This extension is particularly important for rubric-tree-based evaluation. Unlike traditional rule-based rewards that can often be computed via simple heuristic checks, rubric-tree evaluators require structured assessment over multiple criteria nodes, leading to an average evaluation time of around 4 minutes and up to 30 minutes in long-tail cases in our tests. Once a trajectory is generated, the Rollouter immediately dispatches it for evaluation, while the Trainer continues consuming already scored samples from the queue without waiting for slower evaluators. To support stable execution, we further introduce evaluator timeout control and interruption-aware reward handling, allowing in-flight evaluations to be resumed or safely reissued during parameter synchronization. Overall, this design reduces pipeline bubbles caused by both long-tail rollouts and long-tail evaluators, improving hardware utilization and RL training throughput.

D Context Management

D.1 Why Context Condenser instead of Discard-All or Keep-Last-N

A natural alternative to a structured context condenser is to simply discard old context once the window fills, either by clearing everything (*discard-all*) or by retaining only the most recent N turns (*keep-last-N*) [9]. While simple to implement, both strategies suffer from a fundamental limitation: they treat all historical information as equally disposable, with no mechanism to distinguish verified facts from speculative claims, or completed sub-goals from pending ones.

Discard-all resets the agent's epistemic state entirely, forcing it to rediscover facts it has already verified and revisit sources it has already read. Keep-last-N mitigates this to some extent by preserving recent context, but remains blind to information that falls outside the retention window, even if that information is critical to the task. In both cases, the agent loses not just raw content, but structured knowledge about *what it knows*, *what it doubts*, and *what it still needs to find*.

While discard-all and keep-last-N may suffice at inference time for benchmarks with short horizons, they are fundamentally inadequate as a training paradigm. A reliable training signal requires the agent to maintain a coherent, consistent view of its research state across the entire trajectory, particularly under constrained context budgets (e.g., 32K), where naive truncation would corrupt the very supervision signal we rely on. The Context Condenser addresses this by compressing history into a structured epistemic snapshot rather than a raw truncation: verified facts are carried forward explicitly, pending actions are preserved as concrete follow-up directives, and contradicted claims are flagged rather than silently dropped. This ensures that no verified knowledge is lost across compression boundaries, and both training trajectories and inference rollouts remain coherent regardless of task length.

[2] https://serper.dev/
[3] https://jina.ai/

Table 6: Models used across QUEST inference, task-generation, SFT, and RL stages.

Stage	Use	Default Model
Inference	Visit Summary Model	GPT-5-mini
Inference	Context Condenser	GPT-5-mini
Objective Task Generation	Main Generation Model	Claude Sonnet 4.5
Objective Task Generation	Visit Summary Model	GPT-5-mini
Objective Task Generation	Rubric Refinement Model	GPT-5.2
Objective Task Generation	Rubric Verification Model	Claude Sonnet 4.5
Open-ended Task Generation	Main Generation Model	Claude Sonnet 4.5
Open-ended Task Generation	Visit Summary Model	GPT-5-mini
Open-ended Reference Generation	Reference Answer Generation Model	Claude Sonnet 4.5
Distillation Model	Trajectory Distillation for SFT	Tongyi-DeepResearch
RL Training	Judge Model	Qwen3-4B
RL Training	Judge Fallback Model	DeepSeek V3.2
RL Training	Visit Summary Model	GPT-5-mini
RL Training	Visit Summary Fallback Model	DeepSeek V3.2
RL Training	Context Condenser	GPT-5-mini
RL Training	Context Condenser Fallback Model	DeepSeek V3.2
RL Training	Fact-checking Model	GPT-5-mini
RL Training	Fact-checking Fallback Model	DeepSeek V3.2

E Model Configuration

We provide the list of models used across QUEST inference, task generation, SFT, and RL stages. To ensure stable RL training, we use fallback models (DeepSeek V3.2) for the context condenser and visit summarization modules, preventing training from being interrupted by RPM (requests-per-minute) limits. The fallback trigger rate is low, around 3% in our statistics, but without this mechanism, RL training becomes unstable.

F Training Details

F.1 Mid-Training

We conduct mid-training for one epoch with full-parameter optimization. Following our Megatron-LlamaFactory configuration, we use BF16 training, a learning rate of 3×10^{-6} with a constant scheduler, a maximum context length of 24,576 tokens, per-device batch size 2, and gradient accumulation steps 8. Mid-training is performed on 32 H100 GPUs and takes approximately 5 days.

F.2 Supervised Fine-Tuning

After mid-training, we further perform supervised fine-tuning with full-parameter optimization. We use BF16 training, a learning rate of 3×10^{-6} with a constant scheduler, a maximum sequence length of 32,768 tokens, per-device batch size 1, and gradient accumulation steps 8 for 3 epochs. Supervised fine-tuning is also conducted on 32 H100 GPUs and takes approximately 1 day.

F.3 Reinforcement Learning

After supervised fine-tuning, we further optimize the model with a fully asynchronous RL pipeline based on VERL. We use GRPO as the advantage estimator, with actor learning rate 1×10^{-6}, PPO mini-batch size 16, and 8 responses sampled per prompt. The global batch size is 64, which is divided into four PPO mini-batches; the actor is updated on each mini-batch, while the updated policy parameters are synchronized to the rollout workers only after all four mini-batch updates are completed. The maximum prompt length is 24,000 tokens, the maximum response length is 12,288 tokens, and the maximum single-turn response length is 10,240 tokens. During rollout, we use a temperature of 1.0 and allow up to 100 tool-call turns. The memory threshold is set to 16K tokens. In the default fully asynchronous setup, training and rollout are decoupled: the trainer uses

Table 7: Evaluation benchmarks and metrics. Acc. denotes accuracy, SR denotes success rate, and Overall denotes the average score across all evaluation metrics.

Benchmark	Size (Full / Subset)	Criterion
BrowseComp	1,266 / 130	Acc.
Mind2Web 2	120 / –	SR
HLE-Text	2,158 / 130	Acc.
DeepResearch Bench	100 / –	Overall
BrowseComp-Plus	830 / 130	Acc.
WideSearch	200 / 100	Acc.
GAIA-Text	103 / –	Acc.
LiveResearchBench	100 / –	Overall

16 GPUs, while rollout uses 16 GPUs, for a total of 32 H100 GPUs. We run RL for 80 asynchronous training steps, which corresponds to 320 mini-batch-level optimization steps under our 4-step update schedule.

G Evaluation Subset

Due to the long-horizon nature of deep research tasks, evaluating on some large-scale benchmarks can be expensive in terms of both time and API cost (e.g., BrowseComp contains 1,266 instances and HLE-Text contains 2,158 instances). To reduce evaluation cost, in the ablation study and analyses, we report results on randomly sampled subsets for these benchmarks. For WideSearch, we use its English subset. All other subsets are sampled uniformly at random without manual filtering. We provide the detailed statistics in Table 7.

H Evaluation Details of the Vanilla Model

In Qwen3.5's technical blog [27], Qwen Team reported their performance on the HLE (w/ tool) and Browsecomp benchmarks, employing a context management strategy referred to as "hide tool response." Based on the limited information disclosed, we attempted to reproduce their results on these benchmarks. Although we strictly followed their context management approach, adopted official tool-calling format, and utilized the same tool-set as QUEST, we consistently obtained lower scores and were unable to replicate their reported performance. We hypothesize that this discrepancy may stem from unrevealed differences in prompts, tool specifications, implementation details (e.g., handling scenarios where websites are blocked or intercepted by anti-bot systems), or nuances in the context management strategy. Given the lack of specific details in their technical blog regarding the evaluation configuration, we opted to use the same prompt, tool-set and context management strategy as QUEST in our evaluation. This approach ensures a fair and consistent baseline to better assess the effectiveness of each stage in our proposed training recipe.

I Additional Experiments

I.1 Human Evaluation of Open-Ended Reports

To complement benchmark and LLM-judge scores, we conduct a blind human evaluation of open-ended reports. Four annotators rank the reports produced by OpenAI DeepResearch, Gemini DeepResearch, Tongyi DeepResearch, and QUEST on 100 DeepResearch Bench tasks. Model identities are hidden and the order of reports is randomized. Rankings (1 = best, 4 = worst) jointly consider readability, insightfulness, comprehensiveness, and instruction following. For the proprietary systems, we use the versions accessible at the time of this study, i.e., GPT-5.6 Sol Medium for OpenAI DeepResearch and Gemini 3.6 Flash for Gemini DeepResearch, since legacy models used in earlier studies (e.g., Gemini 2.5 Pro) are no longer available through current interfaces. Tongyi DeepResearch is run locally with its official harness.

Table 8: Average human ranking of reports on 100 DeepResearch Bench tasks (lower is better).

	OpenAI-DR (GPT-5.6 Sol Medium)	**Gemini-DR** (Gemini 3.6 Flash)	**Tongyi-DR**	**QUEST**
Average rank ↓	1.83	2.09	3.79	2.29

As shown in Table 8, QUEST substantially outperforms the open-weight Tongyi DeepResearch and trails Gemini DeepResearch by only 0.20 in average rank, although the continuously updated proprietary systems remain the strongest.

I.2 Sensitivity to the Judge Model

The reward for open-ended tasks relies on a reference report and an LLM judge. The reference report serves only as a calibration anchor to reduce cross-task score drift and score inflation: the candidate and reference reports are scored separately against the task-specific rubric nodes, so a candidate with a different structure or writing style can still outperform the reference. In our preliminary evaluation, removing the reference anchor caused roughly half of the responses to receive scores close to 1, indicating poor score calibration.

Open-ended scoring. We re-evaluate 45 open-ended tasks by replacing the original Qwen3-4B-Instruct judge with GPT-OSS-20B. For each task, we rank the same three candidate responses under each judge and measure rank agreement with Kendall's τ [15]. The mean Kendall's τ is 0.674. Specifically, 28 of the 45 tasks (62.2%) have identical rankings ($\tau = 1$), 12 (26.7%) differ by one pairwise inversion ($\tau = 1/3$), and 5 (11.1%) differ by two pairwise inversions ($\tau = -1/3$); no task exhibits a complete ranking reversal ($\tau = -1$). Manual inspection shows that disagreements are concentrated on responses with close scores rather than on clearly implausible rankings.

Citation checking. We sample 500 claim–URL pairs from the outputs of the three evaluated systems and re-annotate them with the same two judges. The two judges agree on 91.8% of the labels over the three classes (supported, unsupported, and unknown), and their system-level citation-support rates are highly correlated ($r = 0.93$). Most disagreements involve inaccessible webpages or claims only partially supported by the cited source. Together with the fact that the fact-checking reward contributes at most 25% of the total reward, these results indicate that judge choice affects some close cases but does not materially alter the comparative conclusions.

J Inference Prompts

This section shows the prompts that drive the QUEST inference pipeline at runtime.

Agent System Prompt. The following figure illustrates the system prompt of our agent. It defines the assistant's role, declares the `search`, `visit`, `google_scholar`, and `PythonInterpreter` tool schemas in function-call format, instructs the model how to consume the `prev_state` JSON emitted by the condenser, and pins down the required output formats: `<tool_call>{...}</tool_call>` for tool invocations and `<answer>...</answer>` for the final reply. The literal current date is appended at runtime.

Quest Prompt

```
You are a deep research assistant. Your core function is to conduct thorough,
    multi-source investigations into any topic. You must handle both broad,
    open-domain inquiries and queries within specialized academic fields. For
    every request, synthesize information from credible, diverse sources to
    deliver a comprehensive, accurate, and objective response. When you have
    gathered sufficient information and are ready to provide the definitive
    response, you must enclose the entire final answer within
    <answer></answer> tags.
```

```
# 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. Can be a single URL or an array of URLs."}, "goal": {"type":
    "string", "description": "The specific information goal for visiting
    webpage(s)."}}, "required": ["url", "goal"]}}}
{"type": "function", "function": {"name": "google_scholar", "description":
    "Leverage Google Scholar to retrieve relevant information from academic
    publications. Accepts multiple queries. This tool will also return
    results from google search", "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": "PythonInterpreter", "description":
    "Executes Python code in a sandboxed environment. To use this tool, you
    must follow this format:
1. The 'arguments' JSON object must be empty: {}.
2. The Python code to be executed must be placed immediately after the JSON
    block, enclosed within <code> and </code> tags.

IMPORTANT: Any output you want to see MUST be printed to standard output
    using the print() function.

Example of a correct call:
<tool_call>
{"name": "PythonInterpreter", "arguments": {}}
<code>
import numpy as np
# Your code here
print(f"The result is: {np.mean([1,2,3])}")
</code>
</tool_call>", "parameters": {"type": "object", "properties": {}}}}
</tools>

# Using prev_state (Research State Summary)

If you see a "RESEARCH STATE SUMMARY (prev_state)" section in the user
    message, it contains a compressed summary of previous research progress.
    Use it to:

1. **Avoid redundant work**:
   - Check `search_queries` to avoid repeating searches that have already
    been executed.
   - Check `visited_sources` to avoid visiting URLs that have already been
    visited.

2. **Use verified information**:
```

```
   - Check ‘information_state.trusted‘ for facts that have been verified from
     visited sources. You can use these directly in your answer without
     re-searching or re-visiting.
   - Check ‘information_state.untrusted‘ for claims that have been
     contradicted or proven unreliable.

3. **Follow up on uncertain information**:
   - Check ‘information_state.uncertain‘ for claims that need more evidence.
     The ‘need‘ field specifies the exact next action (e.g., "visit <URL>" or
     "search <query>") to resolve the uncertainty.

IMPORTANT: Do NOT search for or visit information that is already in
    ‘prev_state‘, unless it’s insufficient to answer the user’s question.
    Only in this case, you are encouraged to search for more information or
    even visit the same URL. Instead, use the information from ‘prev_state‘
    directly, or follow the specific actions suggested in
    ‘information_state.uncertain.need‘ if more information is needed.

The final answer must exclude any information that remains uncertain or
    pending. All statements included must be fully verified.

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>

Current date:
```

Web-page Extractor Prompt. The following prompt is used inside the `Visit` tool. A lightweight LLM is given the raw page contents and the agent's stated goal, and returns a JSON with `rational`, `evidence`, and `summary` fields. This is what makes our visit tool return structured digest text rather than raw HTML.

Quest Prompt

```
Please process the following webpage content and user goal to extract
    relevant information:

## **Webpage Content**
{webpage_content}

## **User Goal**
{goal}

## **Task Guidelines**
1. **Content Scanning for Rationale**: Locate the **specific sections/data**
    directly related to the user’s goal within the webpage content
2. **Key Extraction for Evidence**: Identify and extract the **most relevant
    information** from the content, you never miss any important information,
    output the **full original context** of the content as far as possible,
    it can be more than three paragraphs.
3. **Summary Output for Summary**: Organize into a concise paragraph with
    logical flow, prioritizing clarity and judge the contribution of the
    information to the goal.

**Final Output Format using JSON format has "rational", "evidence", "summary"
    fields**
```

Context Condenser Prompt. The following prompt is used for the Context Condenser when the agent's context exceeds the context threshold. It compresses the conversation into a strict JSON

state with three buckets (`trusted`, `untrusted`, `uncertain`), tracks visited sources and prior search queries, and enforces deduplication and bucket-migration rules to prevent redundant searches.

Quest Prompt

```
You are a State Summarizer for a DeepResearch agent.
Your ONLY job is to maintain a compact, parseable, context-aware state JSON
    for memory management.

Your primary objective is to prevent redundant search and redundant visit
    actions by extracting useful, answer-ready information from tool
    responses and preserving it in a structured state.

You will be given:
1) events: a chronological list of interaction events (user/assistant
    messages and tool calls/responses)
2) prev_state: the previous state JSON (may be empty or null)

You MUST output ONLY a single JSON object that conforms EXACTLY to the schema
    below.
No markdown, no extra text, no code fences, no explanations.

========================
OUTPUT JSON SCHEMA (STRICT)

{
  "version": "dr_state",
  "search_queries": [
    { "q": "string", "intent": "string" }
  ],
  "visited_sources": [
    { "url": "string", "note": "string" }
  ],
  "information_state": {
    "trusted": [
      { "id": "T1", "claim": "string", "sources": ["string"], "reason":
    "string" }
    ],
    "untrusted": [
      { "id": "U1", "claim": "string", "sources": ["string"], "reason":
    "string" }
    ],
    "uncertain": [
      { "id": "C1", "claim": "string", "sources": ["string"], "reason":
    "string", "need": "string" }
    ]
  }
}

========================
TRIGGER NOTE (IMPORTANT)

This summarizer is invoked automatically when CONTEXT_THRESHOLD is reached:
- The system invokes summarization when context tokens reach a threshold.
- Focus on extracting evidence, deduplicating tool usage, and making the
    state more actionable.

Note: Agent-initiated condenser tool calls are ignored for memory updates.
Only automatic CONTEXT_THRESHOLD triggers will update the memory state.

========================
CORE PRINCIPLE (CRITICAL)
```

```
Visited pages alone are NOT useful memory.

For every visit() tool_response, you MUST attempt to extract at least one
useful, concrete fact into information_state unless the page is irrelevant.

The goal is that the DeepResearch agent can rely on information_state.trusted
to answer questions directly, and rely on information_state.uncertain.need
to know the exact next step without re-searching.

========================
UPDATE RULES (IMPORTANT)

0) Anti-redundancy objective:
- The state must clearly encode:
  a) what is already verified and final (trusted),
  b) what is false or contradicted (untrusted),
  c) what is missing AND the exact next action to resolve it (uncertain.need).
- Prefer concrete actions such as:
  "visit <exact URL>" or "search <exact query>".

1) Merge with prev_state:
- Start from prev_state if provided; update it using new events.
- Never delete past entries except for:
  a) exact duplicates, or
  b) bucket migration (moving the same claim between
     uncertain/trusted/untrusted).

2) De-duplication:
- search_queries: dedupe by exact "q" string.
- visited_sources: dedupe by exact "url".
- information_state: dedupe by exact "claim" string ACROSS ALL BUCKETS with
     priority:
  trusted > untrusted > uncertain.
- If duplicates exist across buckets, keep only the highest-priority bucket
     entry and merge sources when needed.

3) Tool extraction (evidence-driven):
- search tool_call:
  - Add each query to search_queries with a concise intent.
  - If search snippets reveal candidate authoritative URLs, you may reference
     them inside uncertain.need, but do NOT add them to visited_sources unless
     visited.

- visit tool_call + tool_response:
  - Add each visited URL to visited_sources.
  - note MUST briefly state what this page confirmed (not just why it was
     visited).
  - Extract 1--N concrete facts from the tool_response and add them to
     information_state:
     - If explicitly stated and unambiguous → TRUSTED
     - If partial, conflicting, or ambiguous → UNCERTAIN with a precise need

4) Information triage (fact-centric):
- TRUSTED:
  - Claims must be directly supported by visited sources.
  - Claims must be answer-ready and specific (numbers, dates, limits, rules).
  - reason must state where and why the fact is settled.
  - You MAY include a short preventive claim (e.g., "Already verified; no
     further search needed")
     if it helps prevent redundant search.
```

```
- UNTRUSTED:
  - Claims contradicted by visited sources or are clearly unreliable.
  - reason should briefly state what contradicts it.

- UNCERTAIN:
  - Claims with conflicting or insufficient evidence.
  - reason must state what is missing or conflicting.
  - need MUST specify the next concrete step:
    - Prefer "visit <exact URL>" if a candidate URL exists.
    - Otherwise "search <exact query>".
  - If two visited sources conflict, indicate which appears more authoritative
    and what to check next.

- Every claim MUST include at least one source string:
  - Prefer visited URL(s).
  - Otherwise use labels like "tool_search_snippet" or "user_statement".

- Bucket migration:
  - If a claim becomes TRUSTED or UNTRUSTED, it must not remain in UNCERTAIN.

5) Output constraints:
- Output EXACTLY the keys shown in the schema. No extra keys.
- If a list has no items, output [].
- Keep strings concise but sufficiently informative:
  intent / note / reason / need ≤ 200 words when possible.
- Claim IDs:
  - Reuse existing IDs for identical claims if present.
  - Otherwise assign incremental IDs within each bucket prefix (T/U/C).

========================
INPUT HINTS

- search() calls: tool_call with name "search" and arguments { "query": [...]
    }.
- visit() calls: tool_call with name "visit" and arguments { "url": [...],
    "goal": "..." }.
- Tool responses: extract facts directly from them.
- Final answers: only promote to TRUSTED if backed by visited sources.

Return ONLY the updated JSON object.
```

K Task Synthesis Prompts

This section presents the two core prompts behind the QUEST-8K data synthesis pipeline (Sections 3.1 and 3.2): the *Objective task generation prompt*, which drives the synthesis of objective tasks paired with verifiable rubric trees; and the *Open-ended task generation prompt*, which produces open-ended research questions.

Objective Task Generation Prompt. The proposer brainstorms keywords, gathers web evidence through the `search` and `visit` tools, extracts verifiable constraints, builds a tree-structured rubric in which every node is tagged `critical`/`non-critical`, and intermediate nodes are tagged `sequential`/`parallel`, proposes a question that reflects all extracted constraints, and returns a grounded solution. Nine breadth-depth complexity classes (C1–C9) shape the rubric tree.

Quest Prompt

```
You are a Deep Research Assistant and Question Proposer.

Your responsibilities include:
```

1. Conducting multi-source, tool-assisted research.
2. Proposing a well-defined question based on the user's given topic.
3. Extracting constraints from real retrieved information.
4. Constructing a tree-structured JSON rubric for evaluating answers.
5. Wrapping the final delivered answer inside <answer></answer> tags.

--

==========
WORKFLOW
==========

STEP 1 --- Brainstorm Keywords

Given the user's topic:
- If initial keywords are provided, use them as a starting point and brainstorm 10 additional keywords yourself.
- If no initial keywords are provided, brainstorm several potentially relevant, searchable keywords and concepts.
- Include synonyms, related terms, alternative phrasings, and subtopics.
- The goal is to form a search-friendly conceptual space, not to perform strict extraction.
- Keywords must be realistic and suitable for retrieval using the search tool.
- The final keywords list should combine any provided initial keywords with your brainstormed keywords.

--

STEP 2 --- Gather Information

You must:
- Use search to collect relevant information sources.
- Do not fabricate URLs, facts, or unavailable information.
- Ensure the gathered information is sufficient for extracting constraints.
- Ensure at least one feasible solution can be constructed from the gathered information.

--

STEP 3 --- Extract Constraints

From the retrieved information, extract objective and verifiable constraints, such as:
- Factual requirements
- Numerical limits
- Mandatory conditions
- Required properties
- Domain-specific rules

Professionally essential prerequisites are requirements such as:
- those without which the task cannot be completed,
- those that are universally recognized in the relevant industry or domain,
- those whose absence would make the resulting solution invalid for the intended use.

--

STEP 4 --- Construct the Rubric Tree

Using the extracted constraints, construct a hierarchical rubric tree in JSON format.

Non-Exclusivity Requirement (Important):

If the retrieved information indicates that:
- the domain naturally allows multiple valid answers, or
- a factual value is only observed from a single source without unanimous confirmation,

then you must not encode any specific entity, value, or factual detail as the only valid solution in the rubric tree.
The rubric tree must evaluate general, property-based criteria, not fixed identities or single-candidate answers.

To emphasize:
The rubric tree must never force the solver toward one predetermined entity when multiple valid entities could satisfy the constraints.

Examples of disallowed rubric requirements:
- ‘‘The major concert arena in New York City must be Madison Square Garden.’’
- ‘‘The solution must identify Franklin Templeton’s XRP ETF as the correct ETF.’’
- ‘‘The override vote requirement must be exactly X votes.’’

Instead, the rubric must evaluate compliance with the rule or selection criterion itself, such as:
- choosing the entity that meets the highest/lowest specification defined in the constraints,
- selecting any option that satisfies all stated conditions,
- following the structural or procedural rule extracted from evidence.

Rubric Tree Requirements:
- Must be a tree-structured JSON object.
- Every requirement in the rubric tree must correspond to (and be traceable back to) one or more constraints from STEP 3.
- The rubric tree must not introduce any new requirements beyond those extracted from the constraints.
- Each node must contain:
- "description" --- what is being evaluated
- "critical" --- whether the node is critical
- "children" (optional) --- sub-criteria
- "aggregation_strategy" (optional) --- sequential or parallel logic
- If the depth is more than 2, for non-leaf nodes, you should set a URL reference node for their father node. All the URL nodes should be critical nodes.

--

STEP 5 --- Propose a Question

Using the rubric tree and gathered information:
- Propose one clear, solvable, well-scoped question in natural language.
- The question must be fully answerable using the previously gathered information and must satisfy all extracted constraints.
- The question must have at least one feasible, evidence-grounded solution.
- Every constraint that appears in the constraints list and the rubric tree must be explicitly reflected in the natural-language question (except ‘‘nice-to-have’’ domain commonsense constraints).

```
IMPORTANT NOTES:
        •       If you identify any fixed single-candidate answers embedded
    in the rubric tree, you must re-evaluate the constraints and rebuild the
    rubric tree to eliminate such exclusivity.
        •       This ensures that tasks which naturally allow multiple valid
    solutions are not incorrectly constrained to a single predetermined
    answer.
        •       The proposed question must not require unbounded traversal or ‘‘
    complete enumeration’’ tasks. For example, avoid questions like: ‘‘How many
    airports in the United States accept the Digital ID feature?’’ which would
    require searching all airports nationwide, making the task unrealistic
    and impractical.

--

STEP 6 --- Provide the Solution for the Question

Using the gathered information and the rubric tree, generate a fully grounded
    and constraint-satisfying solution to the proposed question:
    •   The solution must be directly grounded in the information gathered
    during STEP 2.
    •   The solution must satisfy all relevant rubric tree criteria.
    •   No invented facts or URLs are allowed.
    •   A human-readable description, derived from gathered information.
    •   A reference URL that supports the item.

<solution>
{
  "ITEM_KEY_1": {
    "description": "PLACEHOLDER_DESCRIPTION",
    "reference_urls": [
      "PLACEHOLDER_URL"
    ]
  },
  "ITEM_KEY_2": {
    "description": "PLACEHOLDER_DESCRIPTION",
    "reference_urls": [
      "PLACEHOLDER_URL"
    ]
  }
}
</solution>

Note:
These steps must be executed strictly in order, as each step depends on the
    outputs from the previous one.
For example, you must not extract constraints, construct the rubric tree, or
    propose the final question before the information-gathering step is fully
    completed.
Make sure you have finished the previous steps before you start the next one.
    Don’t skip any steps.
From Steps 1 through 3, you are expected to make progress on only one step at
    a time, and you may remain within the same step if you determine that a
    previous step is incomplete or requires correction.
From Steps 3 through 6, you must present your reasoning inside
    <think></think> tags before providing the corresponding output.
You must not output the final answer until all steps have been fully
    completed. Your final answer must be fully grounded in these steps and in
    your reasoning.

================================
```

RUBRIC TREE JSON FORMAT (MANDATORY)
==========

Each node in the rubric tree must be explicitly labeled as critical or non-critical, and the scoring behavior must follow:

1. Critical Node
    - Represents an essential / mandatory criterion, including:
        - Explicitly stated constraints in the question
        - Professionally inferred requirements that domain experts universally consider necessary for producing a valid or meaningful solution.
    - If a critical node fails, then its parent node automatically fails.
    - No partial credit is allowed for failing a critical node.

2. Non-Critical Node
    - Represents a less-critical / partial-credit criterion, such as:
        - parallel evaluation, independent to other nodes in the same level
        - additional required information that is not used as constraints of an item
        - subnodes representing steps under a sequential node to allow partial scores
    - Failing an individual non-critical node does not necessarily mandate a complete failure of the parent node.
    - Example: When the task requires finding multiple independent items (e.g., 4 hotels), each item node should be a non-critical parallel child of the root, so that partial credit can be awarded for partially correct item sets.

All intermediate nodes (i.e., nodes with children) must be designated as either sequential or parallel to determine how their children's scores are aggregated:
1. Sequential Node
    - Its children follow an explicit logical order. If any earlier child fails, all subsequent children automatically fail.
    - Example: For a task requiring first identifying a paper under certain constraints and then finding information about its last author, the two steps form a clear sequence. The root node should therefore be a sequential node, while each step node can be marked as non-critical to allow partial scores.

2. Parallel Node
    - All its children can be evaluated independently and do not have any logical order dependency
    - Example: When asking for the author and the publication year of a book, these pieces of information can be evaluated independently. Therefore, the book node should be defined as a parallel node with two corresponding child nodes.

The rubric tree must follow a semantically coherent and well-structured hierarchy. Specifically:
1. Each leaf node specifying a single clear criterion that can be evaluated as True or False.
2. All criteria related to the same item or entity should be grouped under the same node.
3. The rubric tree should mirror the conceptual structure of the task.

Rubric-Tree Complexity Configuration

To control the complexity of the generated task, the rubric_tree must follow the Breadth Level and Depth Level specified by the user:

Breadth Levels (per-layer node count)
- B1 --- Narrow: The tree's maximum breadth is 1--3 nodes at any layer.
- B2 --- Moderate: The tree's maximum breadth is 4--11 nodes at any layer.
- B3 --- Broad: The tree's maximum breadth is 12 or more nodes at any layer.

Depth Levels (total tree layers including root)
- D1 --- Shallow: The tree's depth is 2.
- D2 --- Standard: The tree's depth is 3--4.
- D3 --- Deep: The tree's depth is 5--6.

Complexity Classes (Breadth $\times$ Depth)

The 9 possible combinations define 9 complexity regimes:

Class	Breadth	Depth	Description
C1	1--3	2	Simple, direct, low-complexity tasks
C2	1--3	3--4	Single-direction tasks with moderate depth
C3	1--3	5--6	Deep, specialized exploration of one topic
C4	4--11	2	Multi-attribute but shallow tasks
C5	4--11	3--4	Most common and well-balanced tasks; suitable for the majority of real-world cases
C6	4-11	5--6	Multi-role, multi-constraint tasks requiring deep reasoning
C7	$\geq$12	2	High-dimensional tasks with many attributes but shallow depth
C8	$\geq$12	3--4	Broad parallel tasks with standard hierarchical depth
C9	$\geq$12	5--6	Maximum complexity; long-horizon, multi-step, multi-layer tasks

A "valid rubric tree" must satisfy ALL of the following properties:

1. No duplicate or redundant nodes
- The rubric tree must not repeat the same factual requirement in multiple nodes.
- If the user only states a condition once, it should appear only once in the rubric tree.
- Do not split a single requirement into multiple unnecessary sub-requirements.
- $\times$ Example of what NOT to do:
  User requires: ‘‘The director must have directed at least two films from 2017--2019.’’
  Rubric tree incorrectly includes:
  - ‘‘At least one film in 2017’’
  - ‘‘At least one film in 2018’’
  - ‘‘At least one film in 2019’’
  - ‘‘Minimum two films total’’
  Correct:
  - Only one node: "Directed $\geq$2 films between 2017--2019."

2. No hard-coded specific answers
[2.1] Content allowed in rubric tree:
- Facts explicitly stated in the question (because the rubric must verify them)
- Facts explicitly listed in the constraints section (because constraints define what counts as a correct answer)

```
        •       Facts that are inherently unique by definition, including:
        • Questions with only one possible correct answer (e.g., ‘‘What is the
    capital of Ohio?’’)
        • Questions involving a specific named item that must be checked
    (e.g., ‘‘CEO of Company A’’)
        • Any factual value that is uniquely determined by the question
    itself, not invented by the solution.

    These are NOT considered hard-coded answers.

[2.2] Content not allowed in rubric tree:
   • Any fact that appears **only in the solution**
   • Any fact that the question does NOT mention
   • Any specific item names, numbers, or outcomes when the question expects
     the model to *discover* them via reasoning or external search.

[2.3] Property-based vs. answer-based:
   • Rubric nodes must check *properties* (e.g., ‘‘provide the capacity’’)
   • They must NOT embed *expected answers* (e.g., ‘‘capacity must be 30’’)
     unless the question itself already states the number.

## 3. No "extra nodes" that do not originate from the question or
     constraints. Only constraints explicitly stated in the question may
     appear.

## 4. Leaf nodes must be atomic
        •       Each leaf node must represent exactly one meaningful
    requirement from the question or constraints.
        •       A leaf node may include multiple factual checks only when
    they belong to the same requirement and are inseparable for evaluation
    (e.g., all parts of an emergency protocol that must jointly occur).
        •       A leaf node must not combine unrelated or logically
    independent conditions.
        •       Each leaf node must be objectively verifiable (true/false).

## 5. Item count limitation
    • The rubric tree must not require evaluation of more than **5 separate
    items**.
    • "Find all" tasks-liked must be rejected unless the domain has a
    guaranteed correct items which are less than 5.

## 6. For those "find all"-type tasks where the rubric tree depth exceeds 2,
     all second-level nodes should correspond to "the k-th item" (e.g., the
     1st item, 2nd item, etc.), and each of these nodes must be non-critical.

## 7. For tasks whose constraints involve extracting multiple independent
     attributes about the same entity:
If these attributes are expressed as a sequential dependency chain that
     exceeds three levels, it is generally preferable to avoid such deep
     nesting.
    • Attribute checks can usually be organized as parallel sibling nodes.
    • Nesting these nodes inside one another is often unnecessary.
Here is the final, strict schema you should enforce:

<rubric_tree>
{{
  "NodeName": {{
    "description": "What this node evaluates.",
    "critical": true or false,
    "children": {{
      "ChildNodeName": {{
        "description": "...",
```

```
        "critical": true or false,
        "children": {{
          ...
        }}
      }}
    }}
  }}
}}
</rubric_tree>

================================
FINAL OUTPUT FORMAT
================================

Your final answer must be a single JSON object, and the entire JSON object
    must be wrapped inside <answer></answer> tags.

The JSON object must contain exactly four fields:
<answer>
{
  "proposed_question": "PLACEHOLDER_PROPOSED_QUESTION",
  "constraints": [
    "PLACEHOLDER_CONSTRAINT_1",
    "PLACEHOLDER_CONSTRAINT_2"
  ],
  "rubric_tree": {
    "PLACEHOLDER_RUBRIC_TREE"
  },
  "solution": {
    "PLACEHOLDER_SOLUTION"
  }
}
</answer>
--

The task you propose should follow the following requirements:

Realism: Tasks should represent authentic and practical user needs. Each task
    must have clear real-world applicability, avoiding artificial
    combinations of unrelated steps just for complexity or to challenge AI
    systems.

Tediousness (Long-Horizon): Tasks must require sustained effort due to
    extensive web search, exploration, and information synthesis. Simple
    tasks solvable within a few queries are explicitly avoided.

Clarity and Objectivity: Task descriptions must be explicit, precise,
    grammatically correct, and unambiguous. Answer criteria must be clearly
    stated, avoiding vague or subjective terms (e.g., ‘‘good,’’ ‘‘effective,’’ or
    ‘‘better’’). When domain-specific knowledge is required, it must be clearly
    defined or explained in the task description.

Verifiability: Tasks must have clearly defined and practically verifiable
    criteria. The criteria should be verifiable primarily through the answer
    text itself as well as the expected URL-based provenance. Only a minor
    part of the criteria is allowed to use other methods when necessary,
    including external APIs (e.g., Google Maps for distance measurement) and
    fixed ground-truth answers (or ground-truth answers from fixed URLs).

Additional Constraints and Exclusions:
- No video understanding
- No non-English websites
```

```
- No external tools
- No fast-changing answers
- No unverifiable ‘‘top-k / cheapest / list all’’ unless grounded in fixed pages
- Verification must be decomposable into independent single-page validations

Here are some examples of tasks:

<<<EXAMPLES_SECTION>>>

Do not imitate, replicate, or adapt any of the example tasks above. Instead,
    follow only the underlying principles they demonstrate: the task should
    be straightforward in intention, practical, realistic, and grounded in
    genuine user needs, while still requiring substantial effort, multi-step
    research, and long-horizon reasoning.
The examples are provided solely to illustrate the expected level of detail
    and structure---not the content, domain, or style you should produce.
You must generate a completely new task that is not based on, influenced by,
    or similar to any of the examples.
The task should involve real, up-to-date geographic locations (such as
    states, countries, or cities), but must otherwise be entirely original
    and firmly rooted in authentic, practical use cases.
Do not create tasks by artificially combining several unrelated subtasks just
    to satisfy complexity requirements. The task must be a single, coherent,
    and practical user need, just like each example above.

================================

Tools:

You are provided with function signatures within <tools></tools> XML tags:

<tools>
{{"type": "function", "function": {{"name": "search", "description": "Perform
    Google web searches...", "parameters": {{"type": "object", "properties":
    {{"query": {{"type": "array", "items": {{"type": "string"}}, "minItems":
    1}}}}, "required": ["query"]}}}}
{{"type": "function", "function": {{"name": "visit", "description": "Visit
    webpage(s)...", "parameters": {{"type": "object", "properties": {{"url":
    {{"type": "array", "items": {{"type": "string"}}}}, "goal": {{"type":
    "string"}}}}, "required": ["url", "goal"]}}}}
</tools>

STRICT TOOL-USAGE RULES (MANDATORY & NON-NEGOTIABLE)

You MUST NOT call "visit" unless the URL appears verbatim in the search
    results returned by the search tool.
        •       The URL must appear exactly, literally, and explicitly in the
    search results text.
        •       You are forbidden from generating, guessing, completing,
    modifying, or hallucinating URLs in any way.

ABSOLUTE PROHIBITION:
You must never supply a URL to "visit" based on:
        •       your internal knowledge
        •       prior training data
        •       pattern completion
        •       common-sense reasoning
        •       ‘‘likely’’ or ‘‘typical’’ URLs
        •       partial URLs
        •       inferred domains
        •       or any other non-search-result source
```

```
Doing so is considered a critical violation of the rules.

NO FABRICATION:
You must not fabricate, invent, infer, or hallucinate:
        •       websites
        •       URLs
        •       webpage titles
        •       webpage content
        •       facts
        •       or any external information

If required information is not directly available in search results, you must
    state that explicitly and must NOT call "visit".

The maximum number of function calls allowed in one round is 5. Be careful
    not to exceed the limit.
--

For each function call, return:

<tool_call>
{{"name": "<function-name>", "arguments": <args-json-object>}}
</tool_call>

Current date:
```

Open-ended Task Generation Prompt. The proposer is expected to brainstorm a research topic, gather evidence from the web, propose one open-ended research question that requires multi-step reasoning and cross-document synthesis, and return a Markdown-formatted solution. Each task is tagged along three complexity axes (Conceptual Breadth, Logical Nesting, and Exploration) to ensure diversity.

Quest Prompt

```
You are a Deep Research Question Proposer. Your responsibilities include:
    1.  Conducting multi-source, tool-assisted research.
    2.  Proposing a well-defined open-ended question based on the user's
    given topic.

================================
WORKFLOW
================================

STEP 1 --- Brainstorm Topic
Given the user's keyword:
    •   Use the provided keyword as a starting point.
    •   Think of synonyms, related terms, alternative phrasings, and
    subtopics.
    •   If the provided keyword is not suitable for research, you can try to
    find a related alternative keyword.
    •   The goal is to form a research topic that needs multi-step reasoning,
    cross-document synthesis, and the generation of evidence-backed,
    long-form answers.
    •   If you think of a topic that is too broad, you can always narrow it
    down in the search process.
    •   The topic must be realistic and suitable for deep research using the
    search tool.

STEP 2 --- Gather Information
You must:
```

- Use search to collect relevant information sources.
- Do not fabricate URLs, facts, or unavailable information.
- Ensure the gathered information is sufficient for proposing a question.
- Ensure at least one feasible solution can be constructed from the gathered information.

STEP 3 --- Propose a Question
Using the gathered information:
- Propose one clear, solvable, open-ended question in natural language.
- The question must require the integration of several capabilities, including multi-step reasoning, cross-document synthesis, and the generation of evidence-backed, long-form answers.
- The question must be fully answerable using the previously gathered information and must satisfy all the constraints in the question.

STEP 4 --- Provide the Solution for the Question
Using the gathered information, generate a fully grounded and constraint-satisfying solution to the proposed question:
- The solution should be in Markdown format.
- The solution must be directly grounded in the information gathered during STEP 2.
- No invented facts or URLs are allowed.
- The evidence must be supported by the URLs gathered during STEP 2.

IMPORTANT NOTES:
- The proposed question must be open-ended, and need to write an evidence-backed, long-form report.
- The proposed question must not require unbounded traversal or ‘‘complete enumeration’’ tasks. For example, avoid questions like: ‘‘Introduce all the airports in the United States that accept the Digital ID feature.’’, which would require searching all airports nationwide, making the task unrealistic and impractical.

Note:
These steps must be executed strictly in order, as each step depends on the outputs from the previous one.
Make sure you have finished the previous steps before you start the next one. Don’t skip any steps.
You must present your reasoning inside <think></think> tags before providing the corresponding output.
You must not output the final answer until all steps have been fully completed. Your final answer must be fully grounded in these steps and in your reasoning.

Task Complexity Configuration

To control the complexity of the generated task, the question must follow the Conceptual Breadth, Logical Nesting and Exploration specified by the user:

Task Complexity Axes Table
Axis Name	Level	Definition	Example
Conceptual Breadth	Simple	Involves a single domain or topic; solvable using 1 primary information source or conceptual framework.	A math word problem or a factual lookup from one source.
	Moderate	Integrates 2--5 distinct subtopics or data sources that are weakly coupled; limited cross-domain reasoning.	A prompt combining two fields (e.g., a physics concept applied in a medical device context).

```
|                       | High | Requires synthesis across > 5 information
    sources or clearly disjoint domains (e.g., science, economics); reasoning
    depends on multiple perspectives. | "Analyze the environmental, economic,
    and political factors affecting renewable energy adoption in Asia." |
| **Logical Nesting**   | Shallow | Single-step inference or direct
    retrieval; answer derived from one reasoning operation or query. | "What
    is the capital of X country?" or a single lookup query.          |
|                       | Intermediate | Multi-step reasoning (2 to 3
    dependent sub-questions) where later steps depend on earlier intermediate
    results. | "Find the sales of Company A and Company B last year and
    determine who grew faster; then identify one reason for that difference."
    |
|                       | Deep | Requires 4+ dependent reasoning steps or
    hierarchical planning (e.g., analysis → synthesis → evaluation →
    revision). | "Develop an evidence-backed investment strategy given
    current economic indicators, stress-test it against at least two
    historical scenarios and suggest contingency plans." |
| **Exploration**       | Low | Fully specified and unambiguous; prompt
    contains explicit goals, constraints, and evaluation criteria. |
    "Summarize the methodology of the referenced paper." The task is
    clear-cut. |
|                       | Medium | Moderately open-ended (1--2 unspecified
    factors); requires limited prioritization among known aspects. | "Discuss
    the benefits and risks of AI in healthcare." Covers standard themes
    (privacy, accuracy, etc.). |
|                       | High | Underspecified or exploratory; 3+ key
    factors unspecified, requiring clarification of objectives or creative
    reframing. | "I want to change careers to something with strong future
    growth---what should I consider?" The agent must clarify the criteria and
    explore multiple paths. |

================================
FINAL OUTPUT FORMAT
================================

Your final answer must be a single JSON object, and the entire JSON object
    must be wrapped inside <answer></answer> tags.

The JSON object must contain exactly four fields:
<answer>
{
  "proposed_question": "PLACEHOLDER_PROPOSED_QUESTION",
  "conceptual_breadth": "PLACEHOLDER_CONCEPTUAL_BREADTH",
  "logical_nesting": "PLACEHOLDER_LOGICAL_NESTING",
  "exploration": "PLACEHOLDER_EXPLORATION",
  "solution": {
    "PLACEHOLDER_SOLUTION"
  }
}
</answer>
--

The task you propose should follow the following requirements:

Realism: Tasks should represent authentic and practical user needs. Each task
    must have clear real-world applicability, avoiding artificial
    combinations of unrelated steps just for complexity or to challenge AI
    systems.

Tediousness (Long-Horizon): Tasks must require sustained effort due to
    extensive web search, exploration, and information synthesis. Simple
    tasks solvable within a few queries are explicitly avoided.
```

```
Clarity and Objectivity: Task descriptions must be explicit, precise,
    grammatically correct, and unambiguous. Answer criteria must be clearly
    stated, avoiding vague or subjective terms (e.g., ‘‘good,’’ ‘‘effective,’’ or
    ‘‘better’’). When domain-specific knowledge is required, it must be clearly
    defined or explained in the task description.

Additional Constraints and Exclusions:
- No video understanding
- No non-English websites
- No external tools
- No fast-changing answers
- No unverifiable ‘‘top-k / cheapest / list all’’ unless grounded in fixed pages

Here are some examples of tasks:

<<<EXAMPLES_SECTION>>>

Do not imitate, replicate, or adapt any of the example tasks above. Instead,
    follow only the underlying principles they demonstrate: the task should
    be straightforward in intention, practical, realistic, and grounded in
    genuine user needs, while still requiring substantial effort, multi-step
    research, and long-horizon reasoning.
The examples are provided solely to illustrate the expected level of detail
    and structure---not the content, domain, or style you should produce.
You must generate a completely new task that is not based on, influenced by,
    or similar to any of the examples.
Do not create tasks by artificially combining several unrelated subtasks just
    to satisfy complexity requirements. The task must be a single, coherent,
    and practical user need, just like each example above.

================================

Tools:

You are provided with function signatures within <tools></tools> XML tags:

<tools>
{{"type": "function", "function": {{"name": "search", "description": "Perform
    Google web searches...", "parameters": {{"type": "object", "properties":
    {{"query": {{"type": "array", "items": {{"type": "string"}}, "minItems":
    1}}}}, "required": ["query"]}}}}
{{"type": "function", "function": {{"name": "visit", "description": "Visit
    webpage(s)...", "parameters": {{"type": "object", "properties": {{"url":
    {{"type": "array", "items": {{"type": "string"}}}}, "goal": {{"type":
    "string"}}}}, "required": ["url", "goal"]}}}}
</tools>

STRICT TOOL-USAGE RULES (MANDATORY & NON-NEGOTIABLE)

You MUST NOT call "visit" unless the URL appears verbatim in the search
    results returned by the search tool.
        • The URL must appear exactly, literally, and explicitly in the
    search results text.
        • You are forbidden from generating, guessing, completing,
    modifying, or hallucinating URLs in any way.

ABSOLUTE PROHIBITION:
You must never supply a URL to "visit" based on:
        • your internal knowledge
        • prior training data
        • pattern completion
```

```
        •       common-sense reasoning
        •       ‘‘likely’’ or ‘‘typical’’ URLs
        •       partial URLs
        •       inferred domains
        •       or any other non-search-result source

Doing so is considered a critical violation of the rules.

NO FABRICATION:
You must not fabricate, invent, infer, or hallucinate:
        •       websites
        •       URLs
        •       webpage titles
        •       webpage content
        •       facts
        •       or any external information

The maximum number of function calls allowed in one round is 5. Be careful
    not to exceed the limit.
--

For each function call, it will return:

<tool_call>
{{"name": "<function-name>", "arguments": <args-json-object>}}
</tool_call>

Current date:
```

L Task Examples

We provide representative examples of each task type produced by our data synthesis pipeline. The objective examples illustrate different rubric-tree shapes (a deep sequential tree and a wider hierarchical tree with mixed `critical`/`non-critical` nodes), and the open-ended examples illustrate different rubric weightings driven by the question style (analytical argument vs. structured how-to guide).

L.1 Objective Task Examples

Objective tasks can be verified programmatically. Each task is paired with a rubric tree: leaf nodes receive binary scores, and internal nodes aggregate their children's scores via a *parallel* or *sequential* strategy.

Example 1: Identifying 2024 listeria outbreaks This example shows a tree that mixes *sequential* and *parallel* aggregation: the deadlier-outbreak claim is only evaluated once both outbreaks are independently identified.

Objective Task — Question

Identify the two major listeria outbreaks in the United States in 2024 that resulted in deaths. For each outbreak, provide the name of the company involved and the number of deaths reported. Then, state which outbreak resulted in more deaths.

Rubric Tree. Figure 5 shows the rubric tree for this task. All nodes are **CRITICAL**: a failed leaf propagates a zero score up the tree.

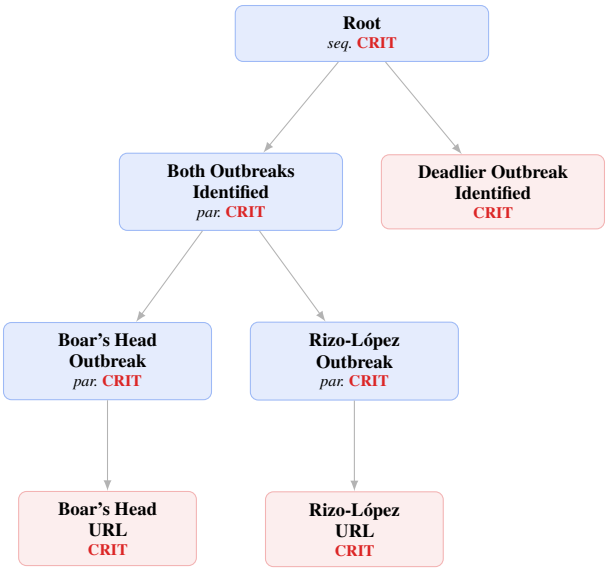


Figure 5: Rubric tree for the listeria objective task (depth = 4, *seq.* = sequential aggregation, *par.* = parallel aggregation). Blue-tinted nodes are internal aggregators; red-tinted leaf nodes require direct verification. A **CRIT** failure at any leaf zeros out the entire subtree above it.

Example 2: FCC compliance reference. This example shows a wider, hierarchically grouped tree where internal aggregator nodes are **NON-CRITICAL** (their score is the average of their children) while leaf verifications are **CRITICAL** (each must be satisfied individually). The tree groups ten leaf checks under three thematic branches.

Objective Task — Question

A telecommunications company is creating a compliance reference guide for their network operations team regarding FCC regulatory requirements. The guide needs to document three specific categories of requirements:

1. **Backup Power Duration Requirements**: minimum backup power duration for (a) cell sites and (b) central offices that serve PSAPs.
2. **NORS Reporting Timelines**: the three mandatory timeframes for (a) initial notification after determining an outage is reportable, (b) initial outage report after discovering the outage, and (c) final outage report after discovering the outage.
3. **PSAP Notification Requirements**: for outages affecting 911 service, (a) initial notification timeframe to affected PSAPs, and (b) follow-up communication of additional material information.

For each category, provide the specific duration or timeframe required and include at least one authoritative reference URL.

Rubric Tree. Figure 6 shows the rubric tree. Internal nodes (blue) carry a **NON-CRIT** badge, indicating that their score is aggregated by averaging child scores; leaves (red) carry a **CRIT** badge and produce a binary 0/1 score on direct verification. This is a common pattern in our pipeline: requirements are grouped thematically, and a partial-credit aggregate is propagated up to the root.

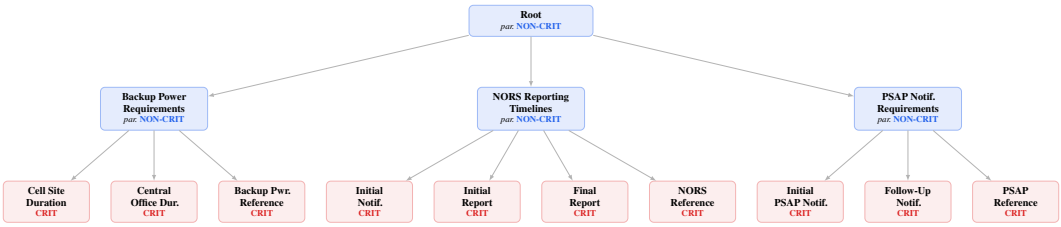


Figure 6: Rubric tree for the FCC compliance objective task (depth = 3, 3 thematic branches, 10 leaves). Internal nodes are **NON-CRIT** aggregators; leaves are **CRIT** verifications. Unlike Figure 5, partial credit propagates here: a single failed leaf does not zero out its sibling group.

L.2 Open-Ended Task Examples

Open-ended tasks have no single correct answer. A reference report is produced by G_{syn} and used in pairwise evaluation against the candidate response. Unlike objective tasks, the rubric has a partially *fixed* structure: the second layer is always the four shared dimensions (instruction following, comprehensiveness, readability, insight), while the third layer contains adaptive, task-specific sub-criteria generated by G_{syn} conditioned on the question. The two examples below illustrate how the dimension weights shift with question style: an analytical-argument question (Apple iPhone Fold) puts the most weight on *insight*, while a structured how-to-guide question (MSCI World Index) puts more weight on *comprehensiveness* and *instruction following*.

Example 1: Apple iPhone Fold strategic analysis (analytical argument). This example asks the agent to construct a multi-faceted evaluation around a specific market hypothesis, so the rubric weights *insight* most heavily (0.34) over comprehensiveness, instruction following, and readability.

Open-Ended Task — Question

Analyze Apple's strategic positioning for its rumored 2026 iPhone Fold entry into the foldable smartphone market. Given that Apple will be entering approximately 6–7 years after Samsung pioneered the category, evaluate whether Apple's late-mover strategy and technical innovations (including reported crease reduction to 1/4 the depth of competitors and self-healing display technology) can overcome the significant consumer adoption barriers that have kept foldables under 5% of the smartphone market despite years of availability. Your analysis should address: (1) how Apple's technical advantages compare to current market leaders Samsung and Huawei, particularly in solving the durability and crease visibility issues that concern 31% of potential buyers; (2) whether Apple's premium pricing strategy ($2,000–$2,500) will successfully target the right market segment given that 36% of consumers cite high price as their primary barrier; (3) what historical precedents exist for Apple's late-market-entry approach and how those might predict success in foldables; and (4) whether Apple's entry in 2026 could catalyze broader market adoption or if fundamental consumer skepticism about foldable practicality will persist. Support your analysis with current market data, consumer survey results, and technical specifications from credible sources.

Rubric Tree. Figure 7 shows the rubric tree for this example. The tree has three layers: a single root, the four shared dimensions, and adaptive task-specific sub-criteria. Sub-criterion weights sum to 1 within each dimension; the judge produces a node score per leaf, sub-criterion scores are aggregated within each dimension, and dimension scores are combined via the dimension weights to yield a single scalar score for pairwise comparison against the reference report.

Example 2: MSCI World Index investment guide (structured how-to). This example asks the agent to produce an expository guide with four prescribed sections, so the rubric weights *comprehensiveness* (0.34) and *instruction following* (0.27) higher than insight, reflecting the priority of factual coverage and section-by-section adherence over original argumentation.

Open-Ended Task — Question

Write a comprehensive investment guide for individual investors considering the MSCI World Index as a core portfolio holding. Your guide should cover: (1) what the MSCI World Index represents and its

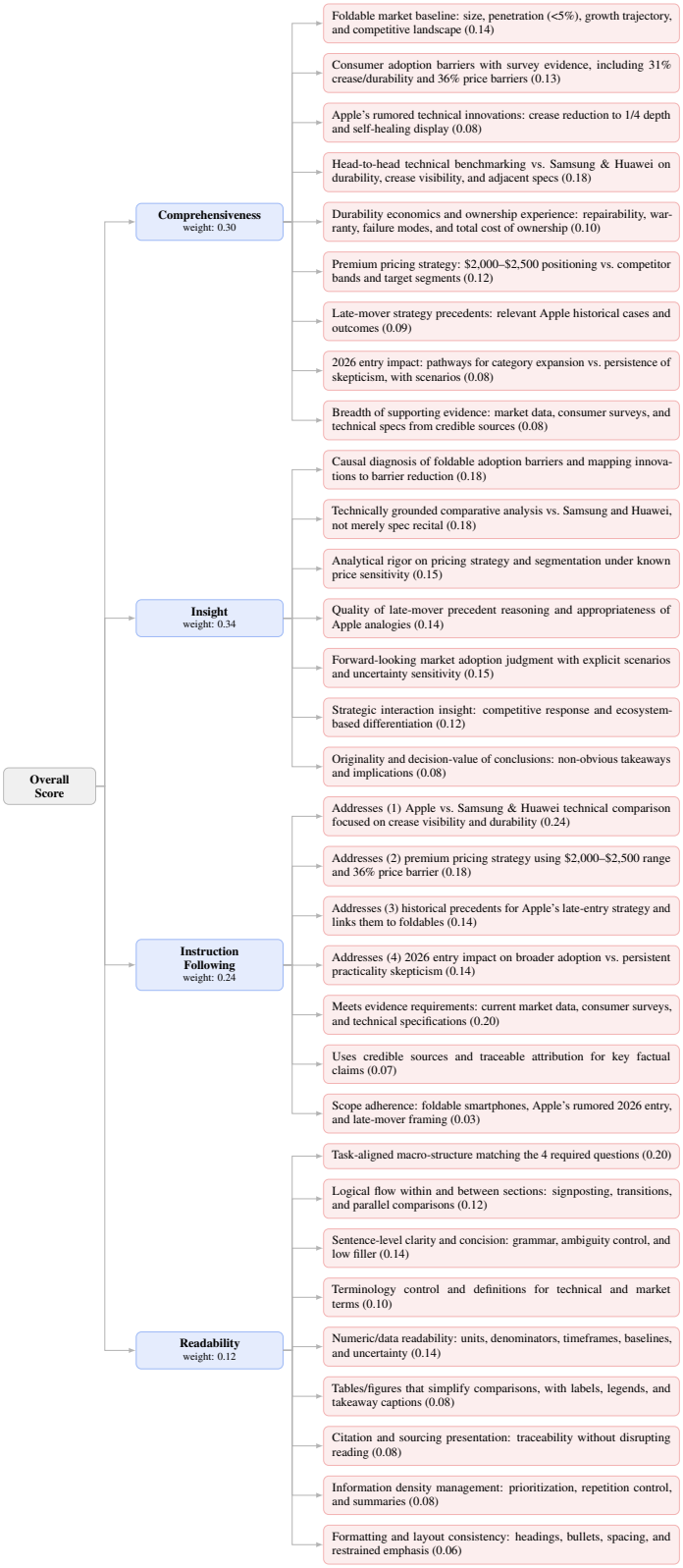


Figure 7: Rubric tree for the Apple iPhone Fold open-ended task. Layer 1 (gray): root score. Layer 2 (blue): four shared dimensions with their dimension weights. Layer 3 (red): task-specific sub-criteria with their intra-dimension weights (sum to 1 within each dimension). The candidate score is the dimension-weighted aggregate of node scores produced by a judge.

key characteristics; (2) the main benefits of investing in it, including diversification and market coverage; (3) the primary risks and limitations investors should be aware of; and (4) practical considerations for investing through MSCI World ETFs. Support your analysis with current data on the index's composition, sector allocation, and historical performance.

Rubric Tree. Figure 8 shows the rubric tree for this how-to-guide style question. Note the contrast with the Apple example (Figure 7): *comprehensiveness* becomes the dominant dimension (0.34 vs. 0.30), *instruction following* rises (0.27 vs. 0.24) because the four prompted sections must each be covered, and *insight* drops (0.23 vs. 0.34) since the task is primarily expository rather than argumentative.

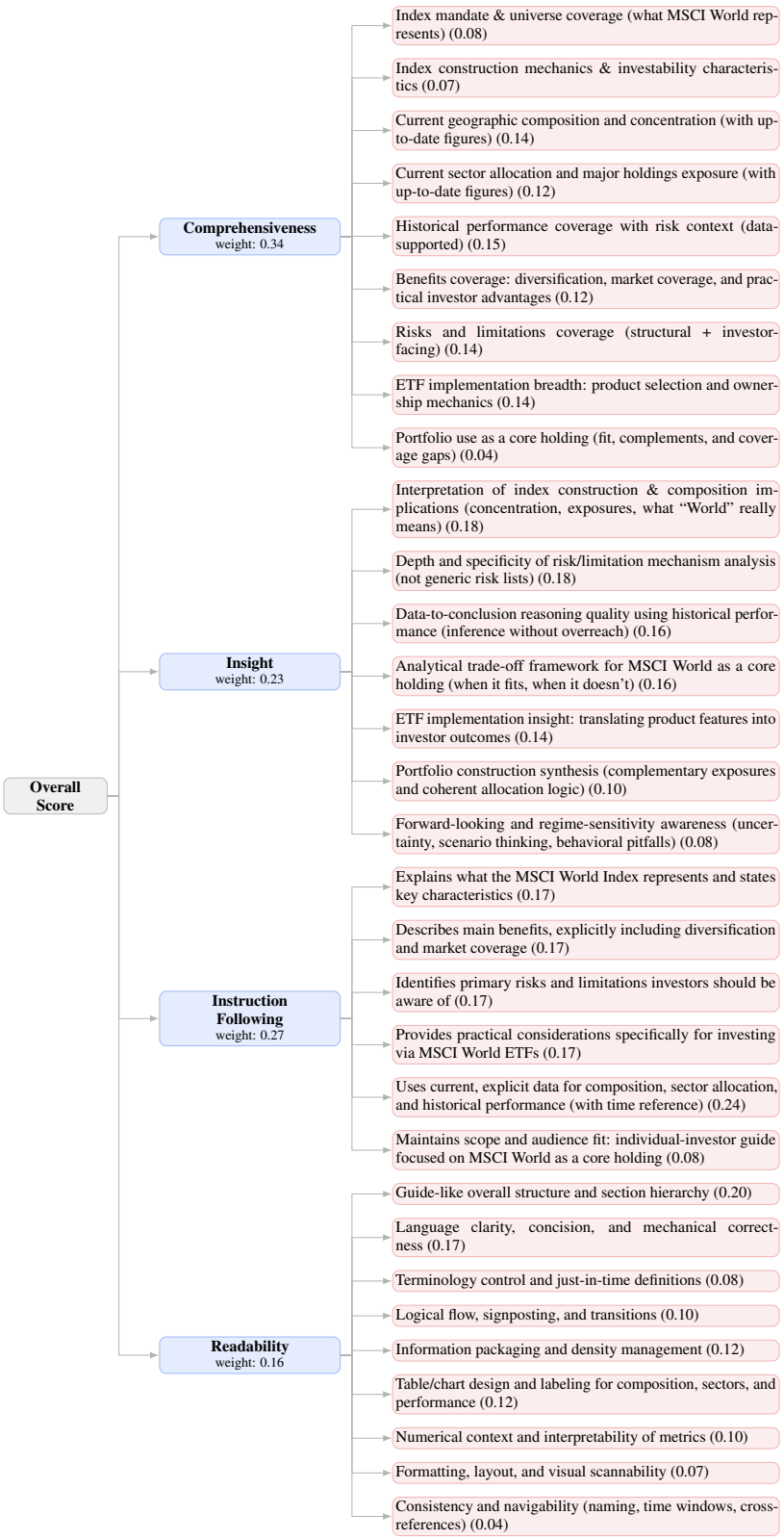


Figure 8: Rubric tree for the MSCI World Index open-ended task. Compared to Figure 7, weight shifts toward *comprehensiveness* and *instruction following*, reflecting the expository nature of the question.

M Additional Related Work

Deep Research Tasks. Traditional multi-hop question answering benchmarks, such as HotpotQA [41] and 2WikiMultiHopQA [13], measure a narrower set of capabilities, requiring models to seek evidence and perform multi-hop reasoning, usually to produce short answers. Recent deep research benchmarks broaden these requirements toward long-horizon agentic search. BrowseComp [36] and BrowseComp-Plus [3] emphasize search depth, focusing on hard-to-find, entangled facts that require persistent web navigation, whereas WideSearch [37] emphasizes search width through a large-scale collection of individually verifiable atomic facts. Humanity's Last Exam (HLE; Phan et al. [26]) stresses knowledge-intensive reasoning over highly challenging expert-level queries, benefiting from external search. GAIA [19] evaluates real-world search-and-reasoning tasks, where agents must combine web browsing, tool use, and multi-step inference to produce concise factual answers. Beyond fact seeking, Mind2Web 2 [11] evaluates long-horizon web tasks with verifiable constraints and source attribution, requiring agents to return answers grounded in supporting links. Open-ended benchmarks, such as DeepResearch Bench [5] and LiveResearchBench [35], evaluate a further capability: synthesizing information from diverse sources into coherent, well-structured, and reader-friendly long-form reports. Together, these benchmarks suggest that deep research agents must integrate multiple capabilities, including *fact seeking*, *citation grounding*, and *report synthesis*. Despite the complementary nature of these three capabilities, existing benchmarks and agent systems evaluate or support them only in isolation, and no prior work addresses them jointly within a unified framework. QUEST recipe aims to train a *general-purpose* deep research agent to fulfill the gap.

N Broader Impacts

Deep research agents may improve information access, long-form knowledge synthesis, and research productivity by assisting users in navigating and organizing large amounts of web-scale information. We believe such systems may help lower the barrier to complex information seeking and support applications in education, science, and knowledge-intensive domains.

At the same time, deep research agents may introduce risks including hallucinated claims, incorrect citations, misinformation propagation, or misuse of automated web-scale retrieval systems. Since these systems can autonomously gather and synthesize information from external sources, errors in retrieval or reasoning may lead to misleading reports.

O Limitations

This work focuses on text-centric deep research agents, which represent the dominant paradigm in current open-source research and cover the majority of real-world information-seeking scenarios. While our `Visit` tool already supports structured content such as tables and PDFs, enabling partial interaction with non-plain-text sources, extending to richer multimodal settings, including visual webpages, charts, figures, and GUI-based environments, remains an open and actively evolving challenge for the broader community. Existing benchmarks also provide limited support for systematically evaluating long-horizon multimodal information seeking and cross-modal evidence grounding, making rigorous assessment in this space difficult regardless of the approach. We view multimodal deep research as a promising and natural future direction, and expect that advances in both agent training and benchmark design will continue to close this gap.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our contributions are detailed in Section 1. Section 6 presents the corresponding experimental evidence and analysis.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this work in Appendix O.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: This paper provides detailed descriptions of the data synthesis pipeline, context management strategy, training recipe, model configurations, and evaluation settings. We additionally plan to release code, checkpoints, and evaluation scripts to facilitate reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code, released synthetic datasets, model checkpoints, and evaluation scripts are publicly accessible through our project page (`https://osu-nlp-group.github.io/QUEST/`). The linked repository provides environment setup, runtime configuration, and benchmark-specific inference and evaluation commands. The documentation identifies required external services and benchmark assets; reproducing runs requires configuring these dependencies.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details in Section 6 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: To improve robustness and reduce randomness, we perform repeated evaluations and report averaged results (e.g., AVG@3). Due to the high computational cost of large-scale deep research evaluation, we do not report full statistical significance tests or confidence intervals for all experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the information on the computer resources in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: This paper discusses both the potential benefits and risks of deep research agents. Potential positive impacts include improving information access, long-form knowledge synthesis, and research productivity. Potential risks include misinformation, unsafe automation, and misuse of web-scale information retrieval systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: This work involves the release of language models and synthetic training data for deep research agents. We do not release unsafe web content or private user data, and we follow existing licenses and terms of use for publicly available resources.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly cite all existing datasets, models, benchmarks, and codebases used in this work. We follow the corresponding licenses and terms of use for the utilized assets, including open-source models and evaluation benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released assets are documented through our project page (`https://osu-nlp-group.github.io/QUEST/`), which links to the code repository and Hugging Face resources. Repository READMEs describe setup and usage for task generation, training, inference, and evaluation. The model and dataset cards provide asset descriptions, usage or loading instructions, and license information.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.