TASKCRAFT: Automated Generation of Agentic Tasks

Anonymous ACL submission

Abstract

Agentic tasks, which require multi-step problem solving with autonomy, tool use, and adaptive reasoning, are becoming increasingly central to the advancement of NLP and AI. However, existing instruction data lacks tool interaction, and current agentic benchmarks rely on costly human annotation, limiting their scalability. We introduce TASKCRAFT, an automated workflow for generating difficulty-scalable, multi-tool, and verifiable agentic tasks with execution trajectories. TaskCraft expands atomic tasks using depth-based and width-based extensions to create structurally and hierarchically complex challenges. Inspired by bootstrap few-shot learning, a self-evolving prompt optimization is implemented to enhance sampling success and reduce latency. Experimental results from SFT on multiple LLMs demonstrate that TaskCraft data substantially enhances multi-hop reasoning and agentic capabilities. Further scaling with TaskCraft tasks and applying RL training yields substantial gains, achieving state-of-the-art performance on four agentic benchmarks. The resulting dataset includes 41k tool-intensive tasks across varied difficulty levels, including 12.6k tool executions and 5k sub-task decompositions.

1 Introduction

Agentic tasks—autonomous, multi-step problem-solving requiring tool use and adaptive reasoning—are increasingly pivotal in AI and NLP. Advances in language agents [17, 31, 6, 42, 43, 44] have shifted AI from passive assistance to proactive agency, enabling complex workflow execution. This is exemplified by systems combining reasoning frameworks like ReAct [37] with dynamic orchestration, where solution trajectories critically improve inference quality. However, the inherent complexity of such tasks challenges conventional annotation paradigms, necessitating novel approaches to model training and evaluation.

To assess advanced agent capabilities, benchmarks such as GAIA [11], BrowseComp [28], and Humanity's Last Exam (HLE) [12] have been introduced. GAIA evaluates reasoning, tool use, and web browsing through 466 real-world questions. BrowseComp comprises 1,266 tasks that test an agent's ability to retrieve and integrate complex online information. HLE includes 2,500 multi-modal questions across over 100 disciplines to measure advanced reasoning and domain knowledge. While these datasets have significantly contributed to agent evaluation, they suffer from scalability limitations due to the labor-intensive nature of data annotation. For example, creating HLE required 1,000 experts to label just 2,500 data points, hindering its ability to scale.

Prior work has explored the automatic generation of instruction-following data using large language models to alleviate the scalability issues of human-annotated datasets. A representative example is the Self-Instruct framework [27], which demonstrated that LLMs can generate high-quality, diverse instruction data for multi-turn dialogues. This approach has proven effective for supervised fine-tuning (SFT). However, these methods are primarily designed for static instruction-following scenarios and fall short in modeling agentic tasks, which require interaction with external tools and environments. Consequently, such data is insufficient for training or evaluating agents that operate in dynamic, real-world settings.

In this work, we introduce TASKCRAFT, an agentic workflow for the automated generation of agentic tasks. Our approach provides the following advantages:

- **Scalability.** The workflow supports adaptive difficulty, seamless multi-tool integration, and the generation of tasks beyond the capabilities of the task-generation agent, along with their corresponding trajectories.

- **Efficient Verification.** During each task extension, only incremental components undergo agentic validation, eliminating the need for full verification of the extended task.

Our approach begins by generating atomic tasks solvable with single-tool invocations, expanding them through depth-based and width-based extensions. Depth-based extension iteratively transforms key textual elements into new atomic tasks for progressive resolution. In contrast, width-based extension formulates tasks requiring resolution of multiple sub-tasks across distinct instances. To ensure high-quality tasks, we employ rejection sampling to verify scenarios where agents using external tools succeed while LLMs fail, validating genuine tool necessity. Linguistic analysis with LLMs facilitates rapid validation and task creation beyond current agent capabilities, enhancing efficiency and task-solving scope. To further improve the efficiency of workflow generation, we implement a self-evolving prompt optimization strategy inspired by bootstrap few-shot learning [5]. This iterative refinement improves rejection sampling pass rates while minimizing generation time.

The controlled generation process ensures inherent access to ground-truth execution trajectories, enabling precise interpretability, reproducibility, and verifiability. To assess the efficacy of generated tasks, SFT was applied to several base LLMs, equipping them with tool-use capabilities. Results show strong performance on multiple agentic benchmarks. Further scaling with TaskCraft tasks and applying reinforcement learning (RL) yields substantial gains, achieving state-of-the-art performance.

Based on this method, we created a task dataset consisting of about 41k tasks of varying difficulty. Each task necessitates different tools for resolution, such as search engines, web browsers, PDF readers, and image analysis. The dataset also includes approximately 12.6k trajectory data executed with tools, along with about 5k instances of multi-hop sub-task decomposition data.

Our key contributions are as follows:

- We introduce an automated agentic task generation workflow capable of producing scalable difficulty, efficient verification, and multi-tool supported tasks, along with their corresponding execution trajectories.

- We utilize prompt learning to facilitates the self-evolution of our generated workflow.

- We validate the effectiveness of our generated tasks via SFT training on multiple LLMs, leading to notable improvements in multi-hop reasoning and agentic benchmarks.

- we create a task dataset comprising about 41k agentic tasks of varying difficulty levels. The dataset also includes approximately 12.6k trajectory data, along with about 5k instances of multi-hop sub-task decomposition data.

2 Notations and Preliminary

Tool-Assisted Task Execution

As Figure 1 shown, given a task q, the agent extracts the input index i_T (e.g., document name, webpage title) for invoking a target tool T. We focus solely on steps that yield a valid tool context, omitting unrelated processes such as file location or search for simplicity. Executing tool T with i_T retrieves the associated context C. The LLM implicitly deduces the relationship R between C and the expected outcome, producing the final result a.

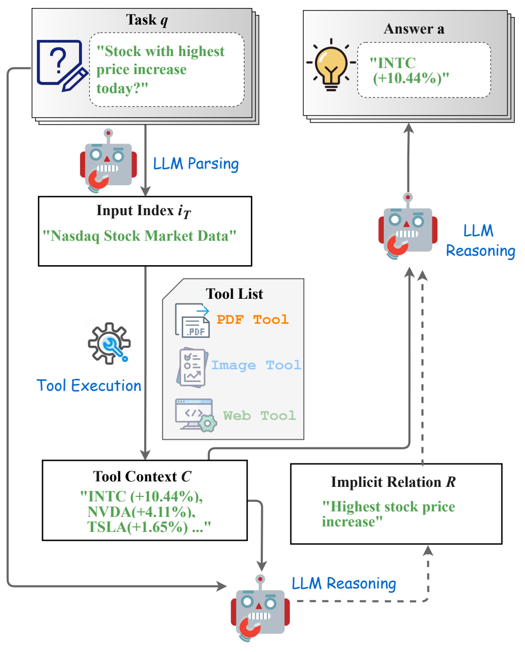


Figure 1: Execution flow of a single tool invocation.

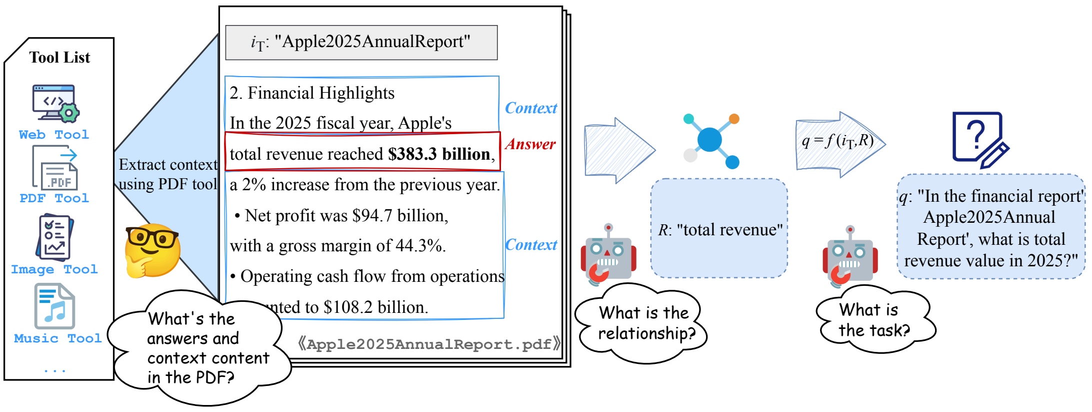

Tool List
Web Tool
PDF Tool
Image Tool
Music Tool
...
Extract context using PDF tool
What's the answers and context content in the PDF?
i_T: "Apple2025AnnualReport"
2. Financial Highlights
In the 2025 fiscal year, Apple's
total revenue reached $383.3 billion,
a 2% increase from the previous year.
• Net profit was $94.7 billion,
with a gross margin of 44.3%.
• Operating cash flow from operations
amounted to $108.2 billion.
Context
Answer
Context
《Apple2025AnnualReport.pdf》
What is the relationship?
R: "total revenue"
$q=f(i_T,R)$
What is the task?
q: "In the financial report' Apple2025Annual Report', what is total revenue value in 2025?"

Figure 2: Atomic task generation. From an unlabeled corpus, we extract i_T and derive textual content C via tool execution. LLM identifies candidate answers a from C, infers their relationship R, and constructs question q conditioned on i_T and R.

Atomic Task

An atomic task is resolved with a single target tool invocation. To simplify, we disregard search and file system operations, assuming a detailed input index i_T enables retrieval through finite navigation.

Given an answer a, the most direct approach to construct an atomic task involves prompting an LLM to generate the corresponding question. However, questions produced in this manner often suffer from low tool invocation rates, unpredictable difficulty levels, unregulated tool requirements, and inconsistent verification complexity (see Section 4.5 for more details).

To mitigate these issues, we assume an ideal search engine capable of retrieving precise data based on i_T (e.g., paper titles, image paths, music names, etc.). Under this assumption, we can construct a task question $q = f(i_T, R) \to a$, where f represents a sampling function that enables the LLM to generate the corresponding natural language representation of the question q based on the provided information.

3 Automated Task Generation Workflow

3.1 Atomic Task Generation

As Figure 2 shown, we begin by compiling a corpus of unlabeled data aligned with the tool's input requirements. From this corpus, we extract i_T and derive textual content C via tool execution. For example, browsing, PDF, and image comprehension tools yield webpage titles, PDF names, and image paths, from which we extract textual content C for answer sampling. We prompt an LLM to identify key candidate answers a from C and infer their relationship R with C, ultimately constructing question q conditioned on i_T and R.

3.2 Task Extension

In order to increase task difficulty in a scalable way, we adopted two extended task strategies: the *depth-based extension* and the *width-based extension*.

Depth-based extension. We aim to construct tasks requiring multiple sequential tool executions, where each step depends on the output of the previous one. To achieve this, a new sub-task must be derived from a known task q^n. The tool input index i_T at each stage exhibits strong extensibility due to (1) its frequent association with proper nouns, which are less likely to be memorized by LLMs, and (2) its natural suitability for recursive definition. Specifically, a single atomic task follows the formulation:

$$q^n = f(i_T^n, R^n) \to a. \quad (1)$$

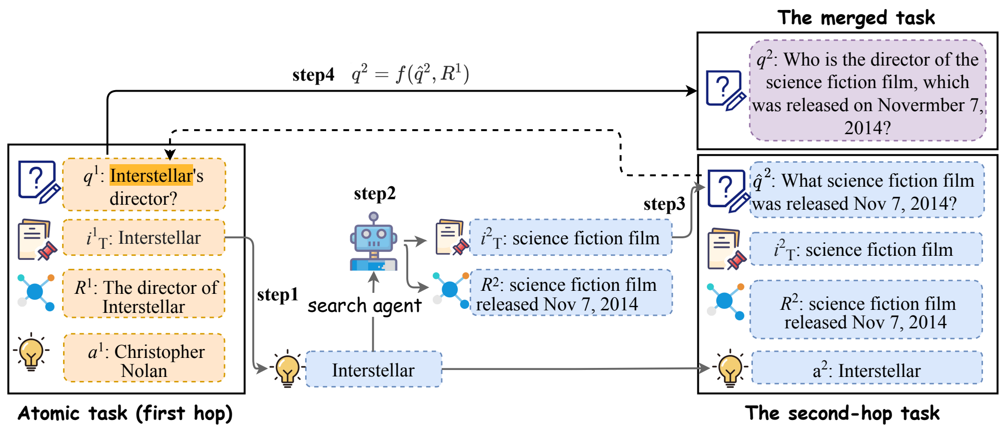
To extend a n-hot task q^n into a (n+1)-hop dependency task q^{n+1}, we can define the recursive formulation:

$$q^{n+1} = f(\hat{q}^{n+1}, R^n) \to a, \quad (2)$$

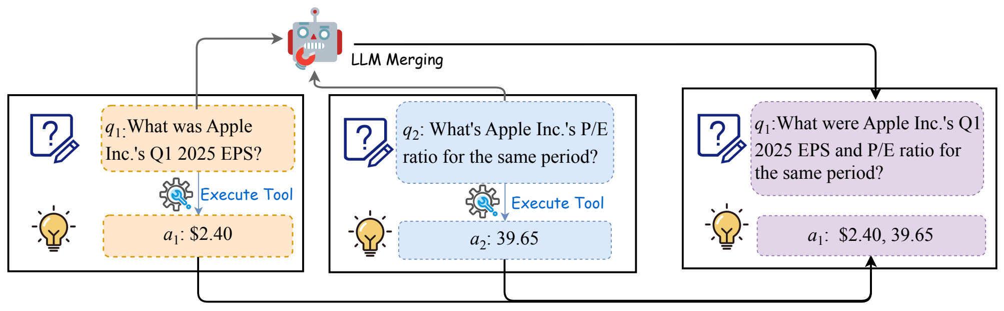
where we ensure that

$$\hat{q}^{n+1} = f(i_T^{n+1}, R^{n+1}) \to i_T^n. \quad (3)$$

Here, i_T^{n+1} denotes a new tool input index derived from i_T^n through reversible operations (e.g., retrieving lyrics from a song name or vice versa). To obtain i_T^{n+1} and its corresponding relationship R^{n+1},


The merged task
step4 $q^2 = f(\hat{q}^2, R^1)$
q^2: Who is the director of the science fiction film, which was released on November 7, 2014?
q^1: Interstellar's director?
i^1_T: Interstellar
R^1: The director of Interstellar
a^1: Christopher Nolan
step2
search agent
step1
Interstellar
i^2_T: science fiction film
R^2: science fiction film released Nov 7, 2014
step3
$\hat{q}^2$: What science fiction film was released Nov 7, 2014?
i^2_T: science fiction film
R^2: science fiction film released Nov 7, 2014
a^2: Interstellar
Atomic task (first hop)
The second-hop task

(a) Depth-based extension


LLM Merging
q_1:What was Apple Inc.'s Q1 2025 EPS?
Execute Tool
a_1: \$2.40
q_2: What's Apple Inc.'s P/E ratio for the same period?
Execute Tool
a_2: 39.65
q_1:What were Apple Inc.'s Q1 2025 EPS and P/E ratio for the same period?
a_1: \$2.40, 39.65

(b) Width-based extension

Figure 3: Strategy for task extension

we employ a search agent that retrieves supersets of i_T^n to mitigate cyclic generation risks. Specifically, the agent searchs textual content C^{n+1} as superset candidates. An LLM then analyzes C^{n+1} to derive the superset index i_T^{n+1} and its relationship R^{n+1} with i_T^n. This process ensures progressive context expansion and effective information association. The resulting i_T^{n+1} and R^{n+1} are synthesized into an intermediate sub-task candidate $\hat{q}^{n+1}$, which undergoes rigorous verification. Upon verification, the system generates the refined task q^{n+1} by integrating $\hat{q}^{n+1}$ with all historical relationships $\{R^1, R^2, ..., R^n\}$.

Width-based extension. The goal of the width-based extension is to generate a new task that needs to be decoupled into multiple sub-tasks to be completed. For simplicity, for two sub-tasks $q_1 \to a_1$ and $q_2 \to a_2$, the combined task q_{width} can be represented as

$$(q_{width} = q_1 + q_2) \to a_1 + a_2, \tag{4}$$

where the $+$ indicates using LLM to merge and rephrase two question strings.

Trajectory generation. Two strategies exist for generating execution trajectories in this task: (1) For simple tasks, such as atomic tasks, existing agents can directly infer and capture the trajectory, including tool selection, parameters, return results, and plans. (2) For complex tasks, such as depth-wise extension tasks, the sub-task trajectory is recorded while iteratively expanding and validating new atomic tasks.

3.3 Task Verification

Under this generation workflow, the verification of generated tasks can be easily performed in two distinct phases:

Atomic task verification: An atomic task is defined as a simple agent task solvable via a single tool call. During verification, we relax this definition slightly: for each candidate task, we evaluate the task agent's output within a limited number of tool-use steps (e.g., three) and compare it with an infer-LLM separately. A judge-LLM verifies whether only the agent's output contains the golden answer, retaining only validated tasks. (see Appendix E for more details)

Task extension verification: This process is conducted purely through linguistic analysis without agent involvement. During depth-wise extension, we first employ a judge-LLM to validate: (1) whether the obtained i_T^{n+1} and its relation R^{n+1} constitute a proper superset of i_T^n with logically sound relationships, and (2) whether the final input index i_T^n in q^n is appropriately replaced by $\hat{q}^{n+1}$ in the expanded task q^{n+1}. Furthermore, an infer-LLM derives the merged task, while the judge-LLM filters out tasks where the correct result is easily inferred, preventing information leakage that could render the task trivially solvable after merging.(see Appendix D for more details).

This framework ensures efficiency by applying agent reasoning only in atomic task verification at creation, while relying on LLM-based verification elsewhere for faster execution. It also enables complex task generation beyond agent capabilities, with reverse reasoning providing supervisory signals to enhance agent learning or reinforcement learning.

4 Experiments

4.1 Corpus Construction

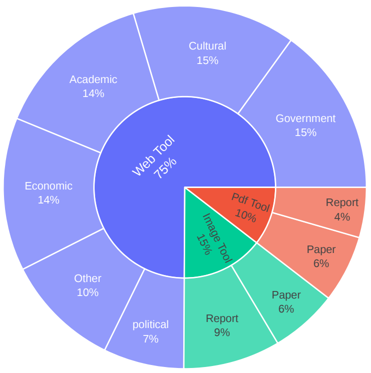


Figure 4: Corpus source distribution.

We collect seed documents across modalities to generate tool-specific atomic tasks, extracting key insights for relevance. For instance, our PDF processor constructs atomic tasks by combining titles with core findings, enhancing the need for agent-based PDF tool invocation. To support atomic task generation, we constructed a dataset comprising webpages, PDF files, and images. Webpage data constitutes the largest proportion (75%), sourced from up-to-date news across multiple domains. Image data accounts for 15%, primarily derived from financial reports and research papers, with filtering to retain images containing information beyond text. PDF data makes up 10%, originating from English financial documents and academic publications.

4.2 Synthetic Tasks Analysis

Human Evaluation. To verify the validity of the results, we randomly sampled 60 atomic tasks and 48 depth-based extension tasks using human evaluation and scored them.

Table 1: Human evaluation for the generated tasks.

Atomic	Linguistic fluency	91.7%
	Accuracy	95.0%
	Single answer	83.3%
	Information leakage	11.7%
Depth-based extension	Extended validity	82.3%
	Non-superset	8.5%

As shown in Table 1, these results highlight the overall effectiveness and controllability of task generation.

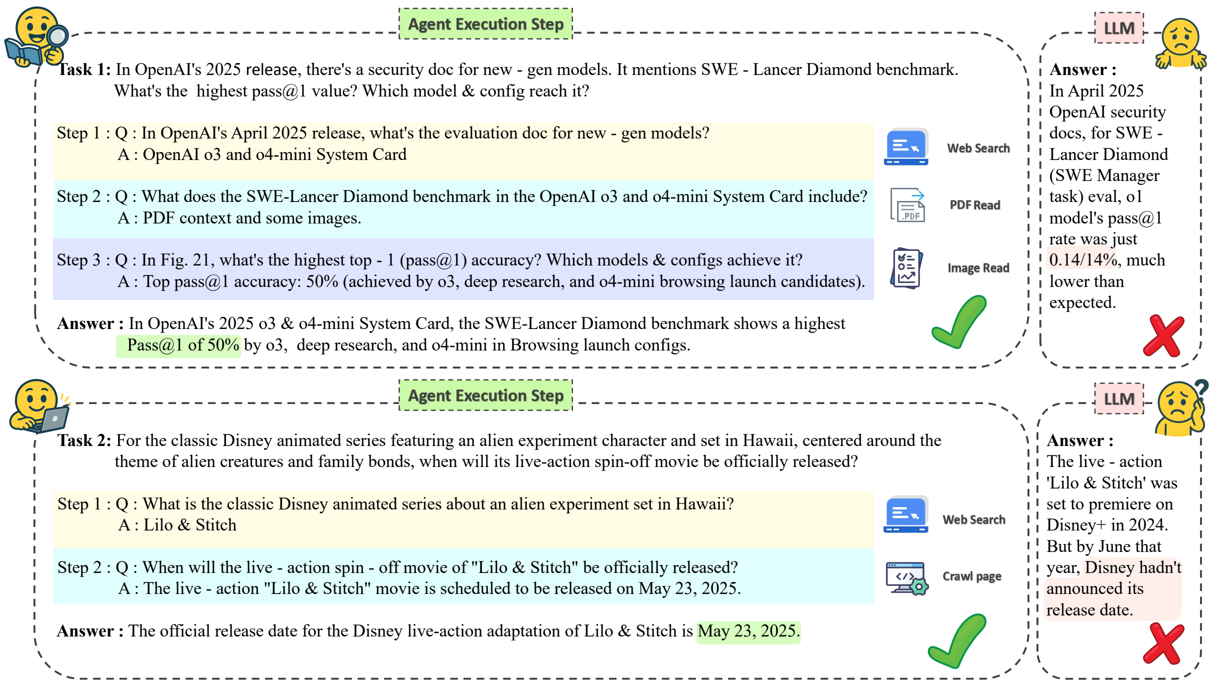
Agent reasoning analysis. To practically assess task difficulty, we sample 1,000 tasks and deploy both Smolagents [16] and Oagents [45], for execution and validation. While both agents performed identical tasks, Oagents incorporated advanced tool capabilities for refined analysis.

Responses were evaluated by comparing the agents' outputs to the golden answer, following a three-point scoring scheme: 2 for fully correct responses, 1 for answers that included the golden answer but contained additional information, and 0 for incorrect responses.

In Figure 6, task failure rates increase from web pages to PDFs and then to images within PDFs, indicating that multi-hop web search tasks are more manageable for agents, while complex comprehension challenges, such as PDF extraction and image interpretation, remain difficult. Additionally, these results demonstrate that our generated tasks span varying difficulty levels, including those that pose significant challenges for current agent capabilities.

Comparison with the GAIA dataset. Table 2 presents the accuracy comparison of Smolagent on the GAIA dataset and our generated dataset. The results indicate that tasks derived from different tool corpora align with GAIA's varying difficulty levels, with image understanding tasks posing the greatest challenge and achieving accuracy comparable to Level3 data.

Unlike GAIA, which requires extensive human annotation, our approach automates task generation, eliminating the need for labor-intensive data labeling while maintaining scalability and adapt-



Task 1: In OpenAI's 2025 release, there's a security doc for new - gen models. It mentions SWE - Lancer Diamond benchmark. What's the highest pass@1 value? Which model & config reach it?

Step 1 : Q : In OpenAI's April 2025 release, what's the evaluation doc for new - gen models?
A : OpenAI o3 and o4-mini System Card

Step 2 : Q : What does the SWE-Lancer Diamond benchmark in the OpenAI o3 and o4-mini System Card include?
A : PDF context and some images.

Step 3 : Q : In Fig. 21, what's the highest top - 1 (pass@1) accuracy? Which models & configs achieve it?
A : Top pass@1 accuracy: 50% (achieved by o3, deep research, and o4-mini browsing launch candidates).

Answer : In OpenAI's 2025 o3 & o4-mini System Card, the SWE-Lancer Diamond benchmark shows a highest Pass@1 of 50% by o3, deep research, and o4-mini in Browsing launch configs.

Answer : In April 2025 OpenAI security docs, for SWE - Lancer Diamond (SWE Manager task) eval, o1 model's pass@1 rate was just 0.14/14%, much lower than expected.

Task 2: For the classic Disney animated series featuring an alien experiment character and set in Hawaii, centered around the theme of alien creatures and family bonds, when will its live-action spin-off movie be officially released?

Step 1 : Q : What is the classic Disney animated series about an alien experiment set in Hawaii?
A : Lilo & Stitch

Step 2 : Q : When will the live - action spin - off movie of "Lilo & Stitch" be officially released?
A : The live - action "Lilo & Stitch" movie is scheduled to be released on May 23, 2025.

Answer : The official release date for the Disney live-action adaptation of Lilo & Stitch is May 23, 2025.

Answer : The live - action 'Lilo & Stitch' was set to premiere on Disney+ in 2024. But by June that year, Disney hadn't announced its release date.

Figure 5: Generated case examples requiring multiple tool calls for completion.

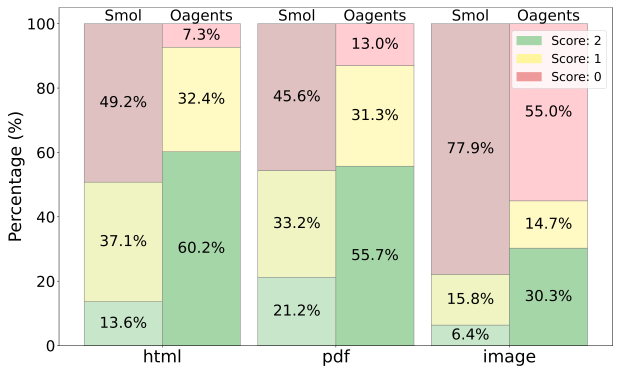


Figure 6: score distribution comparison

Table 2: Accuracy comparison of Smolagents on the GAIA dataset and our synthetic tasks.

GAIA	Level1	Level2	Level3	Avg.
	54.71	43.02	26.92	44.20
Synthetic Task	PDF	html	Image	Avg.
	54.4	50.7	22.1	42.4

ability for agent self-evolution and optimization.

4.3 Enhancing Task Generation Efficiency via Prompt Learning

We employ rejection sampling in both atomic task generation and task extension. To reduce the rejection rate and enhance sampling efficiency, several key challenges must be addressed:

- Efficiently extract candidate answers from the corpus to support atomic task formation and minimize rejections (Section 3.1).
- Guide the agent to find an input index i_T^{n+1}, ensuring coherent depth-wise extension.
- Prompt the LLM in depth-wise extension to articulate the relationship R^{n+1} between the previous input index i_T^n and observed content C^{n+1}, refining task construction and mitigating incoherence-related rejections.
- Integrate tasks to ensure precise substitution, i.e., $q^{n+1} = f(\hat{q}^{n+1}, R^n)$, and clarity while maintaining logical coherence.

We evaluate atomic task generation and task extension independently. For atomic task generation, three metrics are assessed: (1) pass rate, the ratio of validated atomic tasks to candidate tasks; (2) task density, the average number of validated tasks per document; and (3) sampling time, the processing time per document. For depth-based extension, two metrics are evaluated: (1) pass rate, the proportion of successful extensions over n_k attempts (set to 6); and (2) sampling time, the time required for each task extension.

To enhance the LLM's capability in identifying intermediate objectives, we employ bootstrap few-shot learning [5] to systematically optimize four prompts corresponding to key challenges. Each

Method	SFT	RL	GAIA (%)	WebWalker	BrowserComp	HLE
Qwen-2.5-7B-Instruct						
R1-Searcher [19]	✓	✓	20.4	-	-	-
WebSailor [8]	✓	✓	37.9	-	6.7	-
7.5k MHQA	✓		20.4	23.4	3.6	4.2
5k MHQA + 2.5k TaskCraft	✓		**34.0**	**52.6**	**6.4**	**13.2**
+ 6k TaskCraft (RL)	✓	✓	**40.8**	**55.6**	**8.0**	**15.6**
DeepSeek-R1-Distill-Llama-8B						
7.5k MHQA	✓		21.6	28.6	3.6	9.6
5k MHQA + 2.5k TaskCraft	✓		**33.0**	**59.4**	**7.6**	**12.8**
QwQ-32B						
Search-o1 [9]	✓	✓	39.8	34.1	-	-
SimpleDeepSearcher [20]	✓	✓	50.5	-	-	-
WebSailor [8]	✓	✓	50.5	-	-	-
WebThinker [10]	✓	✓	48.5	46.5	-	15.8
WebDancer [29]	✓	✓	51.5	43.2	2.8	-
Qwen-2.5-32B-Instruct						
Search-o1 [9]	✓	✓	28.2	-	-	-
SimpleDeepSearcher [20]	✓	✓	40.8	-	-	-
WebSailor [8]	✓	✓	53.2	-	10.5	-
7.5k MHQA	✓		38.8	36.8	5.6	10.8
5k MHQA + 2.5k TaskCraft	✓		**50.5**	**63.0**	**10.9**	**16.3**
+ 8k TaskCraft (RL)	✓	✓	**53.4**	-	-	-

Table 3: Performance on agentic task benchmarks.

prompt for atomic task generation is enhanced by appending 20 randomly sampled examples. Various prompt configurations are evaluated iteratively based on pass rates to select optimal examples. For depth-based extension, we optimize prompts using 10 randomly sampled examples, refining them to maximize task complexity.

Table 4: Effectiveness of generated task data in prompt learning and depth-wise extension across six extension attempts.

Method	Pass rate	Time
Atomic Task	54.9%	29.1s
+ Optimization	**68.1%**	**23.5s**
Depth-wise@6	41.0%	31.5s
+ Optimization	**51.2%**	**30.2s**

Table 4 examines atomic task generation and depth-wise task extension before and after prompt learning, highlighting the role of generated task data in enabling self-evolution within the generation workflow. These results validate the effectiveness of generated task data in enhancing sampling efficiency and supporting workflow adaptation. The optimized prompts are presented in Appendix E.2.

4.4 Agent Models Fine-Tuning

To validate the effectiveness of our synthetic tasks, we apply SFT to refine an LLM with tool-integrated reasoning in agentic scenarios. We conduct experiments using models from different families and scales, evaluating their performance on the GAIA [11] (a subset of GAIA comprising 103 search-tool-based tasks), WebWalker [30], BrowserComp [28], and HLE [12].

For SFT learning, to ensure the performance gains are not merely due to learning the output format, we use two types of training data: 7.5k tasks sampled from existing multi-hop QA datasets (denoted as MHQA, including HotpotQA and NQ), and 2.5k synthetic tasks via our pipeline. All tasks are converted into agent-compatible trajectories using Oagents. To further enhance model performance, we incorporate additional generated data and apply DAPO [39] for continued RL training.

As shown in Table 3, adding 2.5k TaskCraft tasks to 5k MHQA consistently boosts performance across all models and benchmarks, underscoring that data quality outweighs model size or architecture. Even without reinforcement learning, our data alone enables models to match state-of-the-art methods that use both SFT and RL. Scaling further with more TaskCraft tasks and RL leads to substantial gains, achieving new SOTA results. For instance, on WebWalker, our Qwen-2.5-7B-Instruct model significantly outperforms the prior best, including the larger QWQ-32B. The results demonstrate that TaskCraft data is highly scalable and effectively enhances agent model performance, enabling them to reach state-of-the-art levels.

4.5 Effectiveness of Tool Context in Constructing Agentic Tasks.

In atomic task generation, we incorporate the input index i_T and the tool-answer relation R to structure tasks. To evaluate its effectiveness, we conduct an ablation study where an LLM directly generates single-tool tasks q without using i_T or R. We assess performance via pass rate, resolution time, average tool usage, and usage variance.

Table 5: The effectiveness of tool context.

Method	Pass rate	Time	#Tool-use	σ^2
LLM only	18.5%	119.7s	2.8	1.2
Ours	**43.0%**	**86.7s**	**2.1**	**0.4**

Compared to direct GPT-4.1 prompting, our method significantly improves atomic task generation, achieving higher success rates and faster task construction. It produces more atomic and consistent tasks, with fewer and more stable tool invocations, highlighting the limitations of vanilla LLMs in agentic task design and the robustness of our structured workflow.

5 Related Work

5.1 Instruction Data Generation

Synthetic data has emerged as a promising solution for enhancing performance and enabling new capabilities. STaR [41] augments learning with chain-of-thought (CoT) rationales but often requires a substantial number of task queries beforehand. Methods such as Self-Instruct [27], Self-Chat [33], NuminaMath [7], and OpenMathInstruct-2 [23] generate data from minimal seed examples using LLMs, yet they struggle to extend task generation for multiple tool invocations. WizardLM [32] employs Evol-Instruct to incrementally enhance instruction complexity. However, it relies primarily on rule-based modifications, making its generated instructions unsuitable for agentic task scenarios. MetaMath [38] generates mathematical data by rewriting questions, but adapting agent tasks to environmental feedback presents challenges beyond simple rephrasing. WebInstruct [40] extracts question-answer pairs from a pre-training corpus across multiple domains; however, the generated questions often fail to incorporate tool utilization. AutoAct [14] uses a self-planning mechanism to generate planning trajectories for QA tasks.

5.2 Language Agent

Existing research on agentic task execution advances along two main axes: role specialization and functional partitioning. Role-based approaches, such as AutoGPT [17], AutoGen [31], and Camel [6], organize collaborative agents by dynamically assigning tools. In contrast, frameworks like Barcelona2, Omne, and AgentIM[1] adopt functional partitioning to optimize modular efficiency. SmolAgents [16] integrates ReAct [37] and CodeAct [26] into a hierarchical agent system for iterative code-based task execution. Magnetic-One [2] enhances multimodal performance by decoupling perception [34, 35], planning [18, 22], and execution [15, 26] modules. Dynamic orchestration mechanisms address real-time adaptation and robustness. Trase-Agent [24] adapts strategies based on feedback, while TapeAgents [1] uses asynchronous communication to improve coordination. Studies show that stable sub-agent interactions outperform complex centralized orchestration. To advance autonomy, AutoAgent [21] supports no-code agent customization via natural language coordination, modular workflows, and self-managing file systems. Hybrid systems like h2oGPTe-Agent [3] explore multi-agent optimization, achieving strong results in code generation, though cross-modal bottlenecks remain a challenge.

6 Conclusion

We present TASKCRAFT, an automated workflow for scalable, multi-tool, verifiable agentic task generation. Through width-based and depth-based extension, our framework constructs hierarchically complex challenges. Inspired by bootstrap few-shot learning, a self-evolving prompt optimization is introduced to improve sampling efficiency. Experiments with SFT across multiple LLMs confirm that TaskCraft data enhances multi-hop reasoning and agentic performance, which match the state-of-the-art RL models despite relying solely on SFT. Further scaling with TaskCraft tasks and applying RL training yields substantial gains, achieving state-of-the-art performance on four agentic benchmarks. The created dataset contains 41k tool-reliant tasks across diverse difficulty levels, with 12.6k executed trajectories and 5k multi-hop decompositions.

[1]These are closed-source frameworks.

7 Limitation

This work currently focuses on constructing atomic tasks for common tools, including browsing, PDF processing, and image analysis. Future iterations will enable users to generate atomic tasks tailored to their agents' specific tool requirements. Due to the dataset's scale, inter-task correlations and interactions remain underexplored and present opportunities for future investigation.

References

[1] Dzmitry Bahdanau, Nicolas Gontier, Gabriel Huang, Ehsan Kamalloo, Rafael Pardinas, Alex Piché, Torsten Scholak, Oleh Shliazhko, Jordan Prince Tremblay, Karam Ghanem, Soham Parikh, Mitul Tiwari, and Quaizar Vohra. 2024. Tapeagents: a holistic framework for agent development and optimization. *Preprint*, arXiv:2412.08445.

[2] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, and 1 others. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. *arXiv preprint arXiv:2411.04468*.

[3] H2O.ai. 2024. Autonomous agentic ai: execute multi-step workflows autonomously. [Online]. https://h2o.ai/platform/enterprise-h2ogpte/#AgenticAI.

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

[5] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. Dspy: Compiling declarative language model calls into self-improving pipelines.

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

[7] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9.

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

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

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

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

[12] Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, and 1 others. 2025. Humanity's last exam. *Preprint*, arXiv:2501.14249.

[13] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics.

[14] Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Jiang, Chengfei Lv, and Huajun Chen. 2024. AutoAct: Automatic agent learning from scratch for QA via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3003–3021, Bangkok, Thailand. Association for Computational Linguistics.

[15] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, and 1 others. 2024. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40.

[16] Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents.

[17] Significant-Gravitas. 2023. Autogpt. [Online]. https://github.com/Significant-Gravitas/AutoGPT.

[18] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2023.

Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3009.

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

[20] Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, and 1 others. 2025. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*.

[21] Jiabin Tang, Tianyu Fan, and Chao Huang. 2025. Autoagent: A fully-automated and zero-code framework for llm agents. *arXiv e-prints*, pages arXiv–2502.

[22] Jesus Tordesillas and Jonathan P How. 2021. Mader: Trajectory planner in multiagent and dynamic environments. *IEEE Transactions on Robotics*, 38(1):463–476.

[23] Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

[24] Trase. 2024. Meet trase systems. [Online]. https://www.trasesystems.com/.

[25] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics.

[26] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

[27] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

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

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

[30] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and 1 others. 2025. Webwalker: Benchmarking llms in web traversal. *arXiv preprint arXiv:2501.07572*.

[31] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

[32] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

[33] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.

[34] Dingkang Yang, Kun Yang, Yuzheng Wang, Jing Liu, Zhi Xu, Rongbin Yin, Peng Zhai, and Lihua Zhang. 2023. How2comm: Communication-efficient and collaboration-pragmatic multi-agent perception. *Advances in Neural Information Processing Systems*, 36:25151–25164.

[35] Kun Yang, Dingkang Yang, Jingyu Zhang, Hanqi Wang, Peng Sun, and Liang Song. 2023. What2comm: Towards communication-efficient collaborative perception via feature decoupling. In *Proceedings of the 31st ACM international conference on multimedia*, pages 7686–7695.

[36] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

[37] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

[38] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

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

[40] Xiang Yue, Tianyu Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 37:90629–90660.

[41] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. 2024. Star: Self-taught reasoner bootstrapping reasoning with reasoning. In *Proc. the 36th International Conference on Neural Information Processing Systems*, volume 1126.

[42] Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. 2023. Recurrentgpt: Interactive generation of (arbitrarily) long text. *Preprint*, arXiv:2305.13304.

[43] Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Xiangru Tang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023. Agents: An open-source framework for autonomous language agents.

[44] Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. 2024. Symbolic learning enables self-evolving agents.

[45] He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, Tianhao Peng, Xin Gui, Xiaowan Li, Yuhui Liu, Yuchen Eleanor Jiang, Jun Wang, Changwang Zhang, Xiangru Tang, Ge Zhang, and 5 others. 2025. Oagents: An empirical study of building effective agents. *Preprint*, arXiv:2506.15741.

A Data Statistics

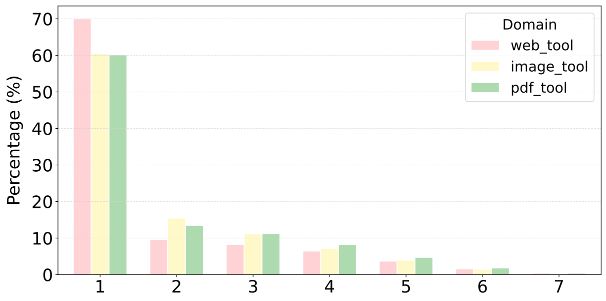


Figure 7: Analysis of all tasks.

As illustrated in Figure 7, task generation exhibits a hierarchical decay pattern across all domains as hop count increases, revealing distinct scalability trends:

- **pdf_tool domain**: Shows gradual performance attenuation with hop depth, 1-hop tasks accounting for 60.13% (8,115 tasks), decreasing to 13.49% (1,820 tasks) for 2-hop and 11.22% (1,514 tasks) for 3-hop. The sharp drop in 5-7 hop tasks (6.94% combined) indicates limited deep-extension capability, yet surpasses other domains in depth scalability.
- **image_tool domain**: Presents the most pronounced performance decay, with 1-3 hops comprising 87.10% (7,125/8,180 tasks) but only 5.71% (467 tasks) for 5-7 hops, highlighting fundamental constraints in deep hierarchical task generation.
- **web_tool domain**: In the web_tool domain, 1-hop tasks dominate, constituting 70.01% (13,467 tasks) of the total. However, this domain also has the highest absolute number of deep extensions, with 5-7 hop tasks accounting for 5.66% (1,089 tasks).

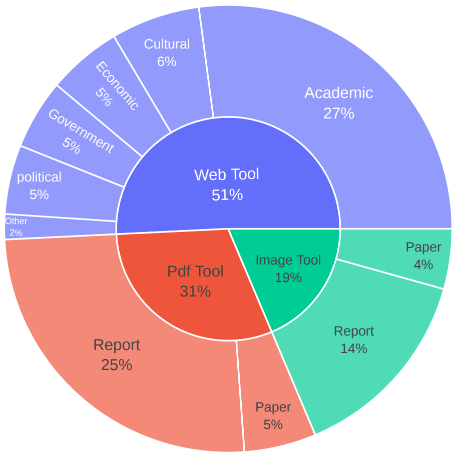


Figure 8: Distribution of atomic data.

Atomic task analysis. We collect data from webpages, PDF files, and images to support the generation of atomic tasks, which form the basis of the dataset, totaling 26,527 instances as shown in Figure 8.

Among them, atomic conclusions from web-based tools account for the largest proportion, reaching 50.77%, with sources spanning multiple domains: academic (27.11%), cultural (6.42%), economic (5.36%) and governmental (5.05%) resources. These derive from up-to-date news and curated online materials for relevance.

Image-based tools contribute 18.64% of the data, extracting structured insights (e.g., key trends, comparisons) from charts/tables in financial reports and research papers. Strict verification excludes conclusions directly replicating source text to avoid redundancy.

PDF-based extraction accounts for 30.59%, supplementing the dataset with findings from financial reports and academic publications. This multi-source approach enhances diversity while maintaining consistency in atomic fact representation.

By systematically integrating these extraction methods, we ensure high-quality task generation, providing a robust foundation for downstream model training and optimization.

B Experiments on Multi-hop QA Tasks

We first evaluate our models across three established multi-hop question answering benchmarks: HotpotQA [36], Musique [25], and Bamboogle [13]. These datasets present diverse challenges in reasoning and search, providing a robust evaluation platform.

We compare the baseline workflow (Search-R1 [4], which leverages reinforcement learning for LLM model optimization) with the agent workflow after applying SFT using the generated tasks.

Method	HotpotQA	Musique	Bamboogle	Avg.
Qwen2.5-3B-Base				
Search-R1	0.284	0.049	0.088	0.140
+ SFT	**0.344**	**0.111**	**0.280**	**0.245**
Qwen2.5-3B-Instruct				
Search-R1	0.324	0.103	0.264	0.230
+ SFT	**0.340**	**0.104**	**0.264**	**0.236**

Table 6: Performance across three datasets and two models. Avg. denotes average.

As shown in Table 6, our synthetic data proves valuable in SFT training, showing average performance improvements of +14.0% (Qwen2.5-3B-Base) and +6.0% (Qwen2.5-3B-Instruct) compared to their respective base workflows, validating our data generation approach. Compared to the Search-R1 baseline, the trained model demonstrates substantial improvements. This suggests that our synthetic data not only enhances immediate task execution but also optimizes RL initialization effectively.

C Scalability of TaskCraft Data

To further examine the scalability of TaskCraft-generated data, we trained a Qwen2.5-7B-instruct models on randomly sampled subsets of 1,000, 3,000, and 5,000 tasks and evaluated them on GAIA-103, using identical training and inference settings (pass@3, 3 epochs, learning rate = 1e-6, batch size = 16).

Table 7: GAIA-103 Performance by Data Size

Data Size	**Pass@3 on GAIA-103**
1,000	17.5%
3,000	31.1%
5,000	39.8%

As shown in Table 7, the results exhibit a clear upward trend, suggesting that larger TaskCraft training sets yield progressively better performance.

D Verification Requirements for Depth-Based Extension

Effective n-hop task extension requires rigorous verification to ensure valid multi-hop reasoning. The transformation must preserve superset validity:

$$\hat{q}^{n+1} = f(i_T^{n+1}, R^{n+1}) \rightarrow i_T^n \tag{5}$$

$$q^{n+1} = f(\hat{q}^{n+1}, R^n) \rightarrow a \tag{6}$$

Current depth-based extension methods often introduce two critical flaws when replacing tool inputs i_T without proper verification:

- **Pseudo-Superset Task**: Superficial substitutions that preserve semantic equivalence but lack genuine superset relationships
- **Information Leakage**: Premature disclosure of information that should only emerge through proper multi-step reasoning

These issues undermine the intended multi-hop reasoning process.

D.1 Pseudo-Superset Task

A fundamental limitation arises when replacing i_T with a semantically equivalent but non-superset index i_T^{n+1}. Consider the following task extension example:

Original task

Query (q^n)**:** How many travel trends for 2022 does 'Travel Trends 2025 | Our Annual Report' present?
Answer: 5

Substituting i_T ("Travel Trends 2025 | Our Annual Report") with the synonymous i_T^{n+1} ("2025 Annual Travel Trends Report") yields a intermediate task:

Intermediate task

Query ($\hat{q}^{n+1}$): What is the title of 2025 Annual Travel Trends Report?
Answer : Travel Trends 2025

Despite valid hop annotations, the intermediate question does not constitute an effective extension: it does not represent a necessary tool-use step. The core issue lies in the absence of a genuine superset relationship between i_T^n and i_T^{n+1}, leading to superficial expansion.

Extended task

Query (q^{n+1}): How many travel trends for 2022 does '2025 Annual Travel Trends Report' present?
Answer: 5

D.2 Information Leakage

A second failure mode occurs when expanded tasks inadvertently expose original answers, enabling large language models (LLMs) to bypass tool retrieval. For instance, consider the extended task:

Extended task

Query (q^{n+1}): In the AP Sports daily summary, Charter and Cox's proposed merger is valued at approximately \$34.5 billion. What is the exact amount?
Answer : 34.5B USD

While this query appropriately conceals the previous i_T^n ("Sports In Brief"), it directly reveals the answer "34.5B USD", allowing the LLM to bypass the intended retrieval process. This compromises the essential tool dependency required for multi-hop task answering.

D.3 Verification for Task Extension

To address these challenges, we propose a rigorous verification framework to ensure the validity of i_T^{n+1}, $\hat{q}^{n+1}$ and q^{n+1} in task extension.

D.3.1 Strict Superset Verification

i_T^{n+1} must be the index of a strict superset of i_T^n, and the relationship can be formalized as:

$$\hat{q}^{n+1} = f(i_T^{n+1}, R^{n+1}) \to i_T^n \qquad (7)$$

where R^{n+1} denotes hierarchical relations (e.g., *contains*, *part_of*). Valid extensions must introduce genuine depth, such as *"Sports In Brief"* $\to$ *"AP News's Sports Section"* (relation: *contains*), while rejecting synonymous substitutions. Additionally, invalid extensions that allow the LLM to derive i_T^n directly should be excluded.

D.3.2 Information Leakage Verification

$$q^{n+1} = f(\hat{q}^{n+1}, R^n) \to a \qquad (8)$$

The extended query q^{n+1} must adhere to the information-sealing principle to ensure proper tool-use reasoning. This requires that the query does not directly expose the original answer, and any query from which the LLM can directly obtain the answer should be filtered out.

D.4 Advantages of the Verification Framework

Our approach provides three key advantages:

- **Superset Integrity**: Guarantees valid hierarchical progression (e.g., *column* $\to$ *page* $\to$ *website*) without logical gaps.
- **Strict Tool Dependency**: Enforces authentic multi-hop reasoning by eliminating solution shortcuts, ensuring mandatory tool-use.
- **Transparent Reasoning**: Offers full explainability through explicit relation paths (R^n).

A properly expanded task under this framework would appear as follows:

Qualified Extended task

Query (q^{n+1}): According to the recurring AP News's sports section feature that regularly provides concise summaries of top sports events and highlights, what is the merger value currently being pursued by US cable giants Charter and Cox as they face increasing competition from streaming services?
Answer : 34.5B USD

E Core Prompts

This section presents key components of the verification prompts used in our framework.

E.1 Atomic task verification

The following prompt is used in atomic task verification (Section 3.3):

Atomic task verification

Task: Evaluate the *consistency* between the golden answer (GA) and another answer (AA, either agent or LLM-generated) as follows:

- **2 points (Fully Consistent)**: AA and GA are semantically equivalent, even if phrased differently.
  ....*(Example)*....
- **1 point (Partially Consistent)**: AA includes all GA information but adds valid extra details.
  ....*(Example)*....
- **0 points (Inconsistent)**: AA omits key GA information or contradicts it.
  ....*(Example)*....

The criteria prioritize semantic equivalence while accommodating informative expansions or reductions.
......

A task is retained as an atomic task if and only if: (1) the *AgentScore* strictly exceeds the *LLMScore*, and (2) the *AgentAnswer* is non-zero.

E.2 optimized prompts

The following prompts is optimized prompt mentioned in (Section 4.3):

Atomic Conclusion Extraction

Task: Extract standalone conclusions from document chunks meeting these criteria:

1. **Atomicity**: Extract only indivisible basic facts ....*(Example)*....
2. **Verifiability**: Include at least one definite identifier (numeric value, time, unique name) and reject vague expressions ....*(Example)*....
3. **Timeliness Handling**: Explicitly mark time ranges for time-sensitive information ....*(Example)*....
4. **Citation Integrity**: Embed complete content of cited references ....*(Example)*....

....*(Example)*....

Depth-wise Extension with i_T^{n+1} and R^{n+1}

Task: Identify a minimal unique superset for an input element based on its attributes, ensuring the superset+relationship uniquely points to the element.
....*(Example)*....
Relationship expression guidelines:

1. Clearly show hierarchical/ownership. Indicate position for series sub-items; clarify ownership for parts of a superset
2. Specify input content's positioning (e.g., time range, publication field, role in superset)
3. Use research/industry standard wording
4. Provide only necessary associations

....*(Example)*....

Logical Substitution: q^{n+1} as $f(\hat{q}^{n+1}, R^n)$

Task: Substitute elements in core queries using auxiliary queries while preserving:

1. **Complexity Balance**: The new query should be slightly more complex than the original core Query and require more steps to solve. But do not make too many changes to the core query.
2. **Answer Uniqueness**: The new query should point to the unique answer: golden answer, and should not point to other answers.
3. **Answer Concealment**: The new query must not reveal information about the golden answer.
4. **Natural Language Polish**: After merging, polish the question to make it conform to human expression habits without changing the original meaning. Do not modify the proper nouns appearing in it.

....(Example)....

E.3 Strict Superset Verification

The following prompt is used in Appendix D.3.1:

Strict Superset Verification

Task: Verify if index i_T^{n+1} uniquely determines subset i_T^n under relation R^n in given queries.
Criteria:

1. **SupersetSubset Relationship**:
   - i_T^{n+1} must be the index of a superset that properly contains i_T^n
   - $i_T^{n+1} \not\approx i_T^n$ (excluding synonym pairs like CAR/AUTOMOBILE)
2. **Relationship Validity**:
   - The relationship R^n must explicitly and uniquely link the superset to the subset (no many-to-one mappings)

......

F Further Training Detail

For SFT training, we synthesize 3,202 multi-hop tasks and their trajectories and apply content masking to search tool contexts in these trajectories.

For RL training, we follow the Search-R1 [4] and use the 2018 Wikipedia dump as a knowledge source and the E5 embedding model as a retriever. For fair evaluation, we fix the retrieval depth to 3 passages for all methods. We merge the training sets of NQ and HotpotQA to form a unified dataset. Evaluation is conducted on the test or validation sets of three datasets to assess both in-domain and out-of-domain performance. Exact Match is used as the evaluation metric. In the PPO settings, we set the learning rate of the policy LLM to 1e-6 and that of the value LLM to 1e-5. Training is conducted for 500 steps, with warm-up ratios of 0.285 and 0.015 for the policy and value models, respectively. We use Generalized Advantage Estimation with parameters $\lambda = 1$ and $\gamma = 1$. We employ vLLM for efficient LLM rollouts, configured with a tensor parallelism degree of 1 and a GPU memory allocation ratio of 0.6. Our sampling strategy utilizes a temperature parameter of 1.0 and top-p threshold of 1.0. For policy optimization, we apply KL divergence regularization with coefficient π=0.001 and implement a clip ratio ϵ=0.2. The action budget is constrained to 4, with a default retrieval depth of 3 passages per query.