



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 width-based and depth-based expansion to create structurally and hierarchically complex challenges. Empirical results show that these tasks improve prompt optimization in the generation workflow and enhance supervised fine-tuning of agentic foundation models. We present a large-scale synthetic dataset of approximately 32,000 tasks with varying difficulty to support future research on agent tuning and evaluation.

1 Introduction

Agentic tasks, involving autonomous multi-step problems that require tool use and adaptive reasoning, are becoming increasingly critical in AI and NLP advancement. As AI transitions from passive assistance to proactive agency, the demand for benchmarks that accurately capture real-world workflows has intensified. This shift is particularly evident in deep research, where agents engage with high-complexity problems through sustained reasoning and strategic tool application. While solution trajectories can significantly enhance agent capabilities, the complexity of these tasks makes large-scale human annotation impractical, necessitating alternative approaches to training and evaluation.

To assess advanced agent capabilities, benchmarks such as GAIA (Mialon et al., 2023), BrowseComp (Wei et al., 2025), and Humanity’s

Last Exam (HLE) (Phan et al., 2025) 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 (Wang et al., 2022), 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 agent’s capability, along with corresponding trajectories.
- **Efficient Verification.** During each task expansion, only incremental components un-

dergo agentic validation, eliminating the need for full verification of the extended task.

The core approach involves initially generating multiple atomic tasks, each solvable with a single tool, and then expanding them using depth-based and width-based expansion. For depth-based task expansion, we iteratively transform specific textual elements of the original task (such as key terms) into a new atomic task to support progressive resolution. In contrast, the width-based expansion formulates tasks that require resolving multiple sub-problems by integrating distinct problem instances.

To ensure high-quality agentic tasks, we employ a rejection sampling strategy during verification. For atomic tasks, we exclude cases where an agent using external tools can solve the task while an LLM cannot, ensuring that atomic tasks genuinely necessitate tool usage. For extension tasks, we leverage linguistic analysis with LLMs, enabling rapid validation and facilitating the creation of challenges beyond existing agent capabilities. This approach enhances efficiency and broadens problem-solving potential.

The controlled generation process ensures inherent access to ground-truth execution trajectories, enabling precise interpretability, reproducibility, and verifiability—critical for agent evaluation and reinforcement learning. To further validate task effectiveness, we implement a self-evolving prompt optimization strategy inspired by bootstrap few-shot learning (Khattab et al., 2024). This iterative refinement improves rejection sampling pass rates while minimizing generation time. Additionally, we leverage the generated task trajectories to train an agent foundation model (Jin et al., 2025). Experimental results show that an independent LLM, trained on these trajectories, effectively plans and invokes tools, yielding performance gains on HotpotQA (Yang et al., 2018a), Musique (Trivedi et al., 2022), and Bamboogle (Press et al., 2022).

Based on this method, we generated a task dataset comprising approximately 32,000 tasks of varying difficulty, each requiring different tools for resolution, including search, web browsing, PDF reading, and image understanding.

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 empirically evaluate task effectiveness through prompt learning, which facilitates the self-evolution of our workflow and holds potential for optimizing existing agent workflows. Additionally, supervised fine-tuning is applied to the agent foundation model to further enhance performance.
- We release a synthetic dataset comprising about 32k agentic tasks of varying difficulty levels, complete with their execution trajectories, to facilitate further research.

2 Notations and Preliminary

Tool-Assisted Task Execution

Given a task q , the agent searches for the input index i_T (e.g., document name, webpage title) as the invocation input for a target tool T . While this may involve a search tool or a file system tool, we omit these details 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 .

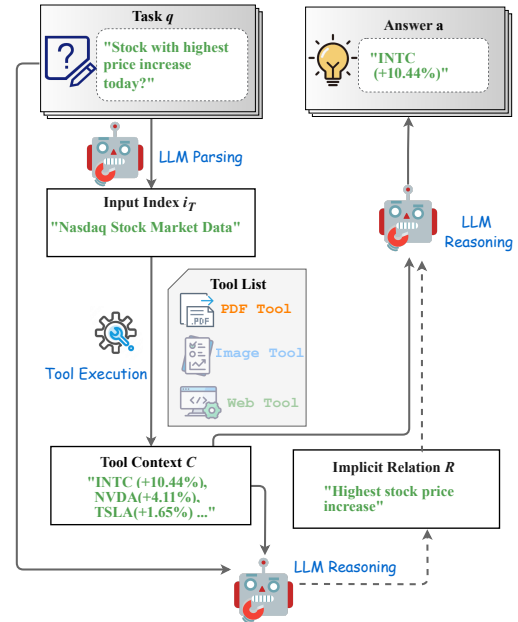


Figure 1: Execution flow of a single tool invocation.

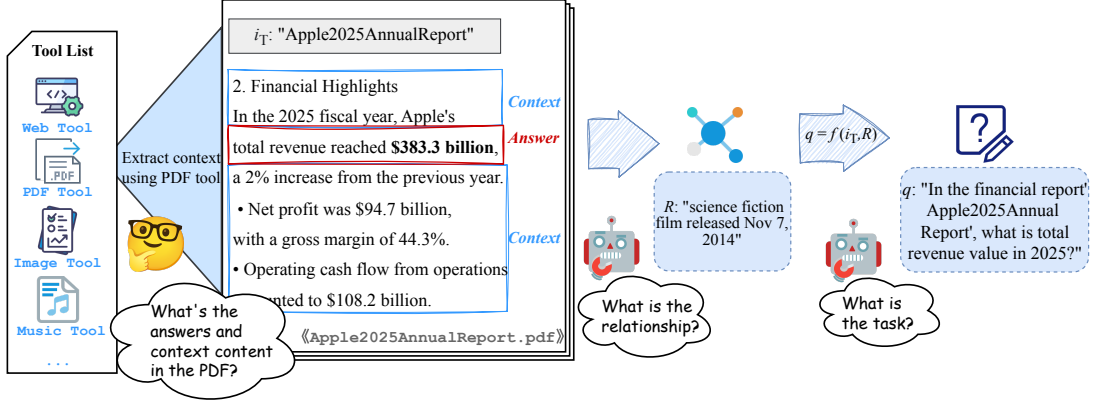


Figure 2: Atomic task generation workflow.

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 exp 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 $q = f(i_T, R) \rightarrow a$ (see Figure 2), where f represents a sampling function that enables the LLM to generate the corresponding natural language representation of the question based on the provided information.

3 Automated Task Generation Workflow

3.1 Atomic Task Generation

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 expansion* and the *width-based expansion*.

Depth-based expansion. 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 subproblem must be derived from a known problem. 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:

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

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

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

where we ensure that

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

Here, i_T^{n+1} can represent any form of tool input, such as a web title, document title, or music name. Many search processes exhibit reversibility, e.g., retrieving lyrics from a song name or vice versa. To obtain i_T^{n+1} and R^{n+1} , we employ a search agent to execute the retrieval process. To further mitigate cyclic generation risks, we encourage the agent to search for i_T^{n+1} whose extracted textual content C^{n+1} forms a superset of the current i_T , thereby

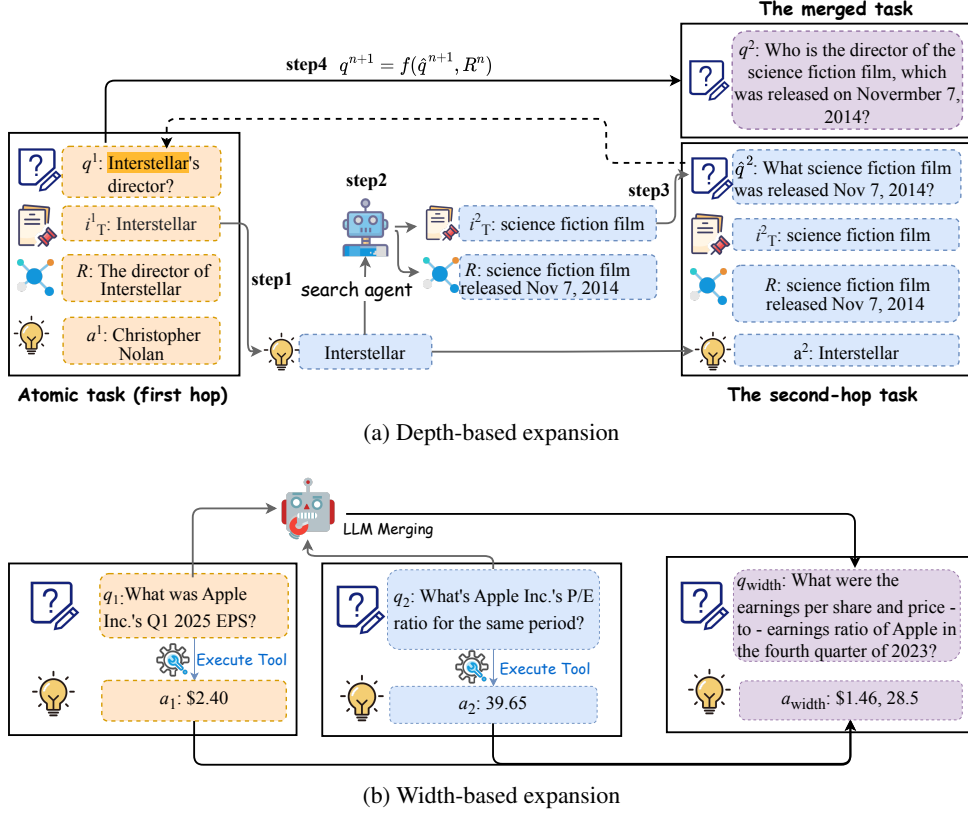


Figure 3: Strategy for task extension

expanding contextual coverage. Finally, we let the LLM analyze the relationship R' between the extracted C' and i_T .

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

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

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

3.3 Task validation

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

Atomic task verification: An atomic task is a simple agentic task, resolvable within a finite number of tool executions. For each candidate, we evaluate the task agent’s output within a limited number of tool-use steps (e.g., three) and compare it with the infer-LLM separately. The judge-LLM verifies whether only the agent’s output contains the golden answer, retaining only validated tasks.

Task expansion verification: This process solely involves linguistic analysis, without relying on any agent. During depth-wise extension, we analyze the pre- and post-merge tasks q^n and q^{n+1} using the judge-LLM to verify whether the last input index i_T^n in q^n has been replaced by $\hat{q}^{n+1}$ in q^{n+1} . Additionally, the 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 problem trivially solvable after merging.

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

4 Experiments

4.1 Corpus Construction

We collect seed documents to generate atomic tasks for various tools, extracting key conclusions to ensure relevance. For instance, the PDF tool constructs atomic tasks using a document’s title and

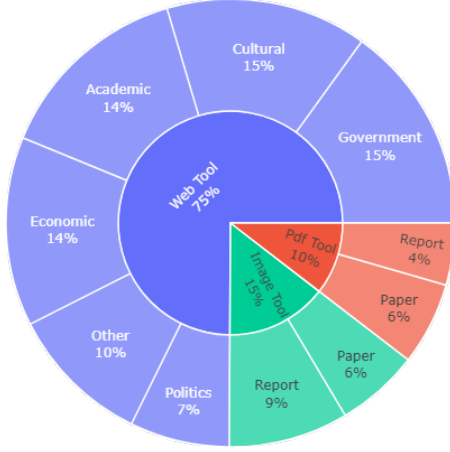


Figure 4: distribution of data source

critical insights to enhance usability. To support atomic task generation, we constructed a dataset comprising webpages, PDF files, and images. Web-page 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

To practically assess task difficulty, we sample 1,000 tasks and deploy both Smolagents (Roucher et al., 2025) and its enhanced variant, Smolagents+ (see appendix for details), for execution and validation. While both agents performed identical tasks, Smolagents+ incorporated advanced tool capabilities for refined analysis.

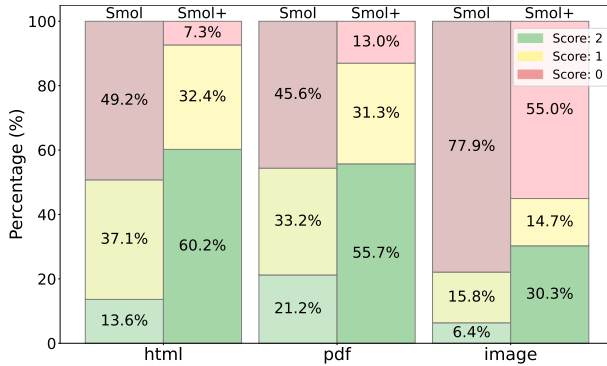


Figure 5: score distribution comparison

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.

We observed that task failure rates for the two tested agents increase progressively from web pages to PDFs and then to images within PDFs. This trend suggests that tasks requiring multi-hop web searches are comparatively easier for agents to complete, whereas more complex comprehension challenges—such as extracting information from PDFs and interpreting embedded images—remain difficult for the evaluated models.

Table 1: 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

Table 1 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 adaptability for agent self-evolution and optimization.

In Figure 5, 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.

Table 2: Effectiveness of generated task data in prompt learning and depth-wise extension across six expansion attempts.

Method	Pass rate	Time
Smolagent	54.9%	29.1s
+ Optimization	68.1%	23.5s
Smolagent+	41.0%	31.5s
+ Optimization	51.2%	30.2s

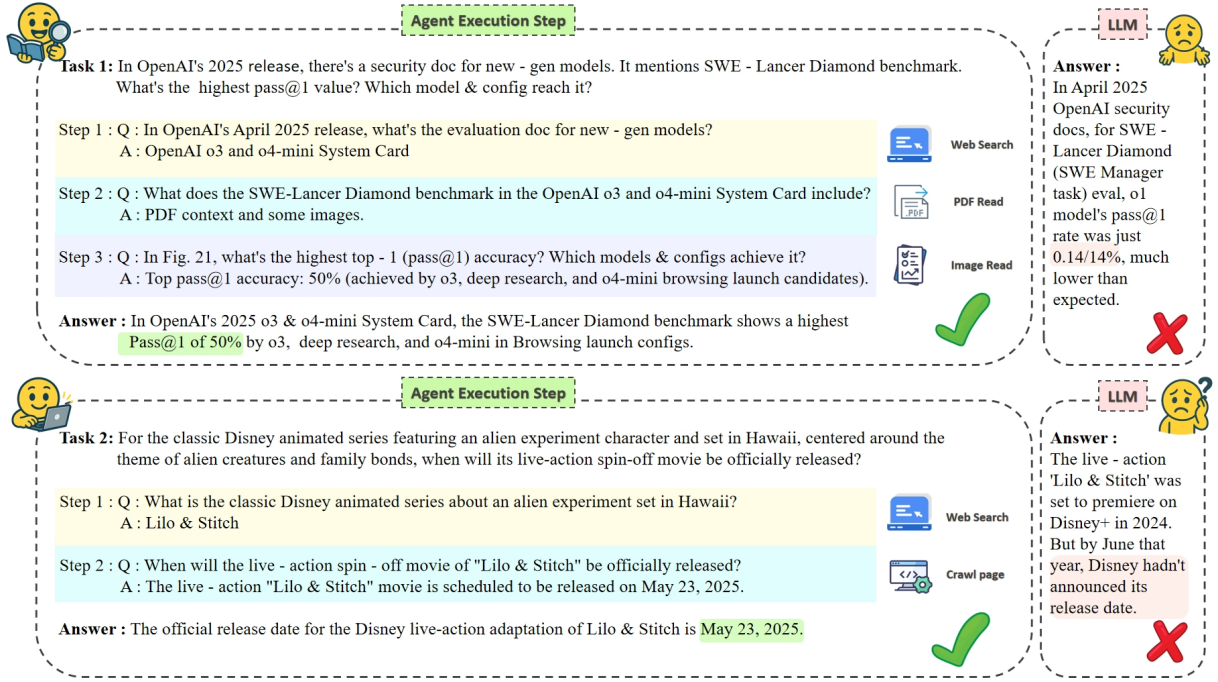


Figure 6: Generated case examples requiring multiple tool calls for completion.

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 expansion.
- Prompt the LLM in deep-wise extension to articulate the relationship R^{n+1} between the previous input index i_T^n and observed content C^{n+1} , refining problem construction and mitigating incoherence-related rejections.
- Integrate tasks to ensure precise substitution ($q^{n+1} = f(\hat{q}^{n+1}, R^n)$), and clarity while maintaining logical coherence.

Evaluation. We assess atomic task generation and task extension separately. For atomic task generation, we evaluate four key metrics: (1) pass rate, representing the proportion of successfully validated atomic tasks relative to candidate tasks. (2) task density, quantifying the average number of validated atomic tasks per document. (3) sampling

time, measuring the time required for processing each document. (4) token consumption, assessing the number of tokens utilized per document. For task extension, we evaluate three key metrics: (1) pass rate, the proportion of successful expansions across n_k attempts (set to 6 in our experiment). (2) sampling time, measuring the time required for extending each task. (3) token consumption, assessing the number of tokens utilized per extension.

Prompt Learning. Intuitively, providing the LLM with effective exemplars can further enhance its ability to identify intermediate objectives. To this end, we employ bootstrap few-shot learning (Khat-tab et al., 2024) to systematically optimize the four prompts corresponding to the aforementioned challenges, thereby facilitating the generated workflow.

For atomic task generation, each prompt is optimized by appending 20 randomly sampled examples. Multiple prompt configurations are then generated by varying these samples, followed by an iterative evaluation process where pass rates determine the optimal selection of inserted examples. For task extension, we focus on depth-wise expansion and adopt a similar strategy to optimize the prompts using 10 randomly sampled examples. These prompts are refined to maximize the number of hops.

Results. Table 3 examines atomic task generation and depth-wise task extension before and after prompt learning, highlighting the role of generated

Table 3: Effectiveness of generated task data in prompt learning and depth-wise extension across six expansion 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

task data in enabling self-evolution within both workflows. For atomic task generation, the data improves efficiency by reducing generation time by 19.2% (29.1 to 23.5 seconds) and increasing pass rate from 54.9% to 68.1%. Similarly, depth-wise extension benefits from the data, with pass rate rising by 10.2% (41.0% to 51.2%) across six expansion attempts, and generation time decreasing by 1.3 seconds (31.5 to 30.2 seconds). These results validate the effectiveness of generated task data in enhancing sampling efficiency and supporting workflow adaptation.

4.4 Fine-Tuning Agent Models Using Synthetic Trajectory

To validate the effectiveness of our synthetic multi-hop data method, we apply supervised fine-tuning (SFT) and reinforcement learning (RL) using the generated trajectory, refining an agent foundation model—an LLM with tool-integrated reasoning.

Evaluation. We evaluate our models on three multi-hop question answering benchmark datasets, as follows: HotpotQA (Yang et al., 2018b), Musique (Trivedi et al., 2023), and Bamboogle (Press et al., 2023). These datasets encompass a diverse range of search with reasoning challenges, enabling a comprehensive evaluation.

Baselines. We conduct a comprehensive evaluation by comparing various baseline models before and after SFT with generated tasks to assess performance improvements: (1) *Base workflow*: We implement agent workflows (Search-R1 without training) across different LLM models. (2) *Search-R1*: An agentic workflow leveraging reinforcement learning for LLM model optimization.

Implementation setup. We evaluate two model variants: Qwen2.5-3B-Base and Qwen2.5-3B-Instruct. To facilitate multi-hop reasoning, we synthesize 3,202 multi-hop tasks and their trajectories for SFT. Following the Chain-of-Action framework (Zhang et al., 2025), we apply content masking to

search tool contexts during training. Our search method, RL training data, and reinforcement learning strategy follow the Search-R1 (Jin et al., 2025). For further training details, refer to Appendix B.

Method	HotpotQA	Musique	Bamboogle	Avg.
Qwen2.5-3b-Base				
Base workflow	0.032	0.006	0.063	0.034
+ SFT	0.232	0.067	0.224	0.174
Search-R1	0.284	0.049	0.088	0.140
+ SFT	0.344	0.111	0.280	0.245
Qwen2.5-3b-Instruct				
Base workflow	0.190	0.037	0.112	0.113
+ SFT	0.221	0.049	0.248	0.173
Search-R1	0.324	0.103	0.264	0.230
+ SFT	0.340	0.104	0.264	0.236

Table 4: Performance across three datasets and two models. Avg. denotes average.

Results. As shown in Table 4, our method demonstrates significant performance improvements across three representative datasets and two model variants.

First, our synthetic data demonstrates significant value in standalone SFT training, achieving average performance improvements of +14.0% (Qwen2.5-3B-Base) and +6.0% (Qwen2.5-3B-Instruct) over the base workflow for their respective models. These gains validate the quality and effectiveness of our synthetic data generation methodology.

Second, compared to the Search-R1 baseline, the workflow with Qwen2.5-3b-Base achieves maximum gains of +19.2% on Bamboogle and +6.2% on Musique. The Qwen2.5-3B-Instruct maintains steady gains, with an average performance margin of +0.6%. The strong performance of our SFT-trained models underscores their suitability for subsequent reinforcement learning, suggesting that our synthetic data not only enhances immediate task execution but also provides a more effective initialization for RL optimization.

4.5 Effectiveness of Tool Context in Constructing Agentic Tasks.

In atomic task generation, we integrate the additional input index i_T along with the relational mapping R between the tool context and a given answer to systematically structure tasks.

To assess the efficiency of our atomic task generation approach, we perform an ablation study using an LLM to directly generate a task q that requires only one external tool to obtain the answer a , explicitly excluding the conditions i_T and R . Evaluation metrics include pass rate, task resolution

time, average tool usage, and the variance in tool usage frequency.

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 atomic tasks generated via direct prompting of GPT-4.1, our approach significantly enhances atomic task generation efficiency. Specifically, our workflow achieves a 24.5% higher pass rate (43.0% vs. 18.5%) while reducing task generation time by 28 seconds (86.7s vs. 119.7s), underscoring the limitations of vanilla LLMs in constructing agentic tasks. Furthermore, our atomic tasks exhibit greater atomicity, as evidenced by a lower average tool invocation count (2.1 vs. 2.8 per query). Task complexity also remains more stable and controllable, with a reduced variance in tool usage (0.4 vs. 1.2). These findings underscore the robustness of our workflow, validating its efficacy in structured task generation.

5 Related Work

5.1 Instruction data generation

Synthetic data has emerged as a promising solution for enhancing performance and enabling new capabilities. STaR (Zelikman et al., 2024) augments learning with chain-of-thought (CoT) rationales but often requires a substantial number of task queries beforehand. Methods such as Self-Instruct (Wang et al., 2022), Self-Chat (Xu et al., 2023b), NuminaMath (Li et al., 2024), and OpenMathInstruct-2 (Toshniwal et al., 2024) generate data from minimal seed examples using LLMs, yet they struggle to extend task generation for multiple tool invocations. WizardLM (Xu et al., 2023a) 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 (Yu et al., 2023) generates mathematical data by rewriting questions, but adapting agent tasks to environmental feedback presents challenges beyond simple rephrasing. WebInstruct (Yue et al., 2024) extracts question-answer pairs from a pre-training corpus across multiple domains; however, the generated questions often fail to incorporate tool utilization in their solutions.

5.2 Language Agent

Recent breakthroughs in large language models have catalyzed the development of autonomous agent systems. These systems demonstrate remarkable capabilities in complex tasks, through tool integration and reasoning strategy optimization. Tool-augmented architectures extend LLMs’ capabilities by integrating code execution (Qin et al., 2024; Wang et al., 2024), web browsing (Schick et al., 2023; Qin et al., 2024), and other tools (Qin et al., 2024), as exemplified by LangChain (LangChain, 2023). Studies indicate that the precision of tool definitions significantly impacts task completion efficiency. Reasoning enhancement strategies based on Chain-of-Thought (CoT) (Wei et al., 2022), ReACT (Yao et al., 2023), and other prompting techniques have substantially improved task planning capabilities. Recent work by Magnetic-One leverages o1-preview’s complex reasoning abilities for agent orchestration. Though challenges persist in maintaining stability during dynamic environment interactions. Reliability mechanisms like self-verification (Paul et al., 2023; Fu et al., 2025; Pan et al., 2024; Zhang et al., 2024), inference-time search (Koh et al., 2024; Chen et al., 2024; Song et al., 2024), and memory in multi-step tasks. FRIDAY (Wu et al., 2024) uses iterative self-optimization for continuous improvement.

6 Conclusion

We present TASKCRAFT, an automated workflow for scalable, multi-tool, verifiable agentic task generation. Through width-based and depth-based expansion, our framework constructs hierarchically complex challenges. Empirical results demonstrate its effectiveness in structured task generation, improving prompt optimization and supervised fine-tuning while reducing reliance on human annotation. Additionally, we release a large-scale synthetic dataset to support future advancements in agent tuning and evaluation.

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 time and cost constraints, the evaluation primarily assessed the problem-solving capabilities of smolagents and smolagents+.

References

- Ziru Chen, Michael White, Raymond Mooney, Ali Payani, Yu Su, and Huan Sun. 2024. When is tree search useful for llm planning? it depends on the discriminator. *arXiv preprint arXiv:2402.10890*.
- Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jinggang Wang, Xunliang Cai, and Weiran Xu. 2025. Agentrefine: Enhancing agent generalization through refinement tuning. *arXiv preprint arXiv:2501.01702*.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. [Search-rl: Training llms to reason and leverage search engines with reinforcement learning](#). *Preprint*, arXiv:2503.09516.
- 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.
- Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*.
- LangChain. 2023. Langchain: Build context-aware reasoning applications. [Online]. <https://github.com/langchain-ai/langchain>.
- 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.
- 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*.
- Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*.
- Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.
- 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, Mantas Mazeika, and 1090 others. 2025. [Humanity’s last exam](#). *Preprint*, arXiv:2501.14249.
- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.
- 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.
- 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.
- 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>.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*.
- Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.
- 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.
- 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*.

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

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

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

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. 2024. Os-copilot: Towards generalist computer agents with self-improvement. *arXiv preprint arXiv:2402.07456*.

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

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018a. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018b. [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.

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)*.

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*.

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

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.

Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. 2024. Towards efficient llm grounding for embodied multi-agent collaboration. *arXiv preprint arXiv:2405.14314*.

Yuxiang Zhang, Yuqi Yang, Jiangming Shu, Xinyan Wen, and Jitao Sang. 2025. [Agent models: Internalizing chain-of-action generation into reasoning models](#). *Preprint*, arXiv:2503.06580.

A Data Statistics

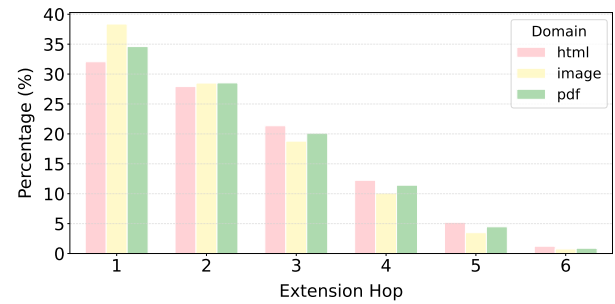


Figure 7: Analysis of all tasks.

As shown in Figure 7, task generation exhibits a hierarchical decline across all domains as the number of hop increase. The expansion of **depth-based** and **width-based** layers leads to progressively more complex and valuable tasks. Below are the detailed statistics for tasks synthesized through depth-based and width-based expansion.:

- **PDF domain:** 1-hop (34.6%) and 2-hop (28.52%) together constitute 63.12% of tasks, with 1-3 hop contributing 83.23% of the total. High-hop tasks (5-6 hop) account for only 5.34%, indicating balanced *initial-layer expansion* but *limited deep-extension efficiency*.
- **Image domain:** Exhibits the highest **1-hop proportion** (38.37%) and strong reliance on *shallow extensions* (1-3 hop: 85.66%). High-hop tasks contribute the lowest proportion (4.24%), likely due to the *shallow scalability* of its data structure.

- **HTML domain:** With the **largest total task count** (6,804), it shows a **1-hop proportion** of 32.05%, including the highest absolute 1-hop task count (2,181). While 1-3 hop tasks still dominate (81.32%), its **high-hop contribution** (5-6 hop: 6.44%) is the strongest across domains, suggesting *better deep-extension capability*.

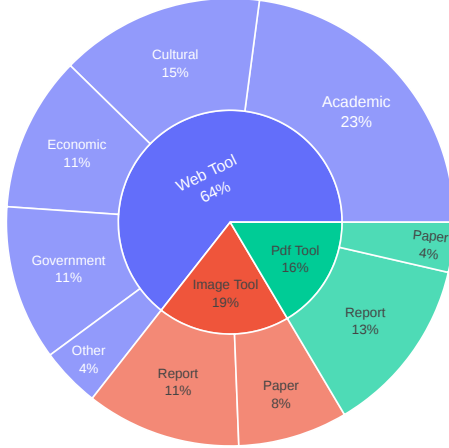


Figure 8: Distribution of atomic data.

Atomic task analysis. We collect data from web-pages, PDF files, and images to support the generation of atomic tasks, which form the basis of the dataset, totaling 14,291 instances as shown in Figure 8.

Among them, atomic conclusions extracted by web-based tools account for the largest proportion, reaching 64%. They are sourced from the latest news and relevant resources covering academic, cultural, economic, and governmental domains. Atomic conclusions extracted by image-based tools account for 19%, mainly derived from the charts and tables in English financial reports and research papers. Additional screening is carried out to ensure that the conclusions are not present in the original text. Data extracted by PDF-based tools accounts for 16%, also sourced from English financial reports and academic papers.

This data collection and task-construction process ensures the effectiveness of atomic task generation, laying a high-quality foundation for its subsequent expansion and optimization.

B 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 (Jin et al., 2025) 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 $\pi=0.001$ and implement a clip ratio $\epsilon=0.2$. The action budget is constrained to 4, with a default retrieval depth of 3 passages per query.

C Smolagents+

We developed Smolagents+, enhancing its web search capabilities, integrating multiple information sources, streamlining search results, and implementing a query rewriting strategy to optimize search performance.