Open, Reliable, and Collective: A Community-Driven Framework for Tool-Using AI Agents

Hy Dang[1], Quang Dao[2], Meng Jiang[1],
[1]University of Notre Dame, [2]Rose-Hulman Institute of Technology,
https://github.com/hydang99/opentools

Abstract

Tool-integrated LLMs retrieve information, perform computations, and take real-world actions, but their reliability depends on both *tool-use accuracy* and *intrinsic tool accuracy*, including tool correctness, stability, and safety. While prior work primarily emphasizes tool use, intrinsic tool accuracy remains under-examined. We introduce **OpenTools**[1], a community-driven and maintainable toolbox for discovering, using, evaluating, and contributing open-source tools. OpenTools standardizes tool interfaces, converts documented Python functions into reviewable bundles, supports maintainer-triggered evaluation, and combines non-executing risk inspection with optional advisory LLM review. A **public web demo**[2] allows users to run tools and agents, inspect evidence, contribute tests, and submit tools for maintainer review, while MCP enables controlled access from external applications. Experiments show that community-contributed, task-specific tools yield relative gains of 6% to 22% over an existing toolbox across multiple agent architectures, highlighting the importance of intrinsic tool accuracy. OpenTools is released under the Apache 2.0 license, with a demonstration video at https://www.youtube.com/watch?v=XhRwATwIBxU.

1 Introduction

Large language models (LLMs) have evolved from generative systems into general-purpose agents that can plan, reason, and interact with users over contexts (Grattafiori et al., 2024). Agentic frameworks support structured reasoning (Yao et al., 2023a; Li et al., 2024b; Zhang and Ding, 2024) and self-reflection (Ji et al., 2023; Liu et al., 2025), extending LLM capabilities beyond question answering and helping mitigate hallucinations (Yao et al., 2023a; Zhang et al., 2024; Zhu et al., 2025). However, token-only generation remains brittle when tasks require current knowledge or interaction with external systems (Paranjape et al., 2023; Qin et al., 2023). Tool-augmented language models (TALMs) address this limitation through retrieval systems (Gao et al., 2023; Jin et al., 2025), calculators (Schick et al., 2023), code interpreters (Wang et al., 2024a), and domain APIs (Wu et al., 2025; Arlt et al., 2025; Jang et al., 2025). While this delegation improves factuality and supports tasks, it also makes agent reliability dependent on external tools.

We argue that TALM reliability has two failure modes: **(i) tool-use accuracy**, which concerns selecting and invoking tools correctly, and **(ii) intrinsic tool accuracy**, which concerns tool correctness, stability, and robustness to drift. Most prior work emphasizes tool learning (Qin et al., 2024; Patil et al., 2024) while assuming that tools and their documentation are reliable. In practice, tools can fail because of incomplete coverage, API and dependency changes, nondeterminism, and silent errors. Tool access can also introduce security and credential risks (Winston and Just, 2025; Milev et al., 2025). Maintaining reliable tools therefore requires a shared interface, representative tests, pre-execution risk inspection, explicit evaluation states, and interoperable access across agent frameworks.

To address these challenges, we introduce OpenTools, an open-source, community-driven toolbox for tool-integrated LLMs. First, OpenTools standardizes tool schemas and wrappers, while a deterministic converter transforms supported Python functions into reviewable bundles with structured metadata. Second, its evaluation and maintenance workflow combines community-contributed tests, non-executing source inspection, secret-risk detection, policy-gated functional evaluation, and optional advisory LLM review of sanitized evidence. Evaluation records distinguish completed, failed, blocked, and not-run states, while continuous inte-

[1]Source Code and Implementation: https://github.com/hydang99/opentools
[2]Web Demonstration: https://huggingface.co/spaces/opentools/opentools

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 380–390
October 24-29, 2026 ©2026 Association for Computational Linguistics

gration supports reviewable updates to the shared tool inventory. Third, OPENTOOLS provides a public web interface and a Model Context Protocol (MCP) server. The web interface supports test cases, tool execution, and open-source tool submissions to further improve OPENTOOLS.

The community remains central to expanding tool coverage, contributing tools and failure cases, and reviewing evaluation evidence, while automation reduces the effort required to standardize and assess these contributions. Across diverse domains, replacing an existing toolbox with OPENTOOLS-T under otherwise fixed evaluation settings yields **6% to 22%** relative gains across all evaluated tool-using configurations. These consistent improvements demonstrate the practical value of a broader, task-relevant, standardized, and maintainable tool ecosystem. We believe OPENTOOLS provides a shared foundation for reproducible tool-reliability research and encourage the community to extend it with new tools, tests, and failure cases.

2 OPENTOOLS Framework

OPENTOOLS is an open-source, community-driven framework for building, evaluating, sharing, and deploying tools for tool-integrated language models. It supports two complementary workflows, shown in Figure 1: (1) a *Tool Accuracy and Maintenance Loop* for standardizing contributions, evaluating intrinsic tool reliability, and maintaining the shared toolbox, and (2) an *Agentic Workflow* for integrating tool collections into LLM agents. The framework uses common tool schemas, separates tool maintenance from agent policies, and records structured evaluation and execution evidence for reproducibility and debugging.

2.1 Tool Accuracy and Maintenance Loop

An agent may select the correct tool and arguments but still receive an incorrect or unavailable result. Tool behavior can change because of API updates, dependency drift, nondeterminism, rate limits, or backend failures. Workflow 1 therefore treats *intrinsic tool accuracy* as a maintained property rather than a fixed assumption.

Problem Definition. Given a tool τ, its source S_τ, a test suite $\mathcal{T}_t$, and an execution policy P, OPENTOOLS produces a reliability record

$$R_t(\tau) = \texttt{Evaluate}(\tau, S_\tau, \mathcal{T}_t, P).$$

The record includes the tool version, evaluation, static risk evidence, test outcomes, and execution status. The workflow consists of three stages: contribution and standardization, risk-aware evaluation, and community maintenance.

2.1.1 Tool Contribution and Standardization

The key design contribution of OPENTOOLS is a *review-oriented lifecycle* connecting tool standardization with automated evidence and maintainer review. Each tool has a natural-language description, typed JSON schema, output contract, and tool card recording dependencies, provenance, limitations, risk evidence, and evaluation history. For supported Python functions, a **deterministic converter** derives the schema from type annotations, generates a standardized wrapper, preserves the source, and packages the artifacts for review. Ambiguous entry points and unsupported signatures produce explicit errors rather than inferred conversions.

2.1.2 Risk-Aware Tool Evaluation

Layered Non-Executing Inspection. Before any permitted functional test, OPENTOOLS inspects submitted source without importing or running it. Its AST analyzer identifies syntax errors, credential access, filesystem writes, network and process capabilities, dynamic execution, and possible embedded secrets. When available, GITLEAKS and DETECT-SECRETS provide additional secret detection, while BANDIT and SEMGREP identify Python-specific and rule-based security issues. Their outputs are normalized into reports containing scanner status, rule, severity, source location, and sanitized evidence. Unavailable or failed scanners remain explicit, and potential secret values are redacted. These findings support risk assessment and help maintainers quickly identify potential risks in contributed tools.

Contributor-Facing Advisory Review. Contributors receive the generated tool card, conversion status, scanner coverage, detailed findings, and recommended actions. They may also request an optional LLM judge that summarizes concerns, required actions, and a recommendation from sanitized metadata and evidence. The LLM does not receive raw source code or credential values. Instead, contributors can use this evidence to refine their own tool implementations.

Gated Evaluation and Maintainer Review. The hosted interface inspects but does not execute

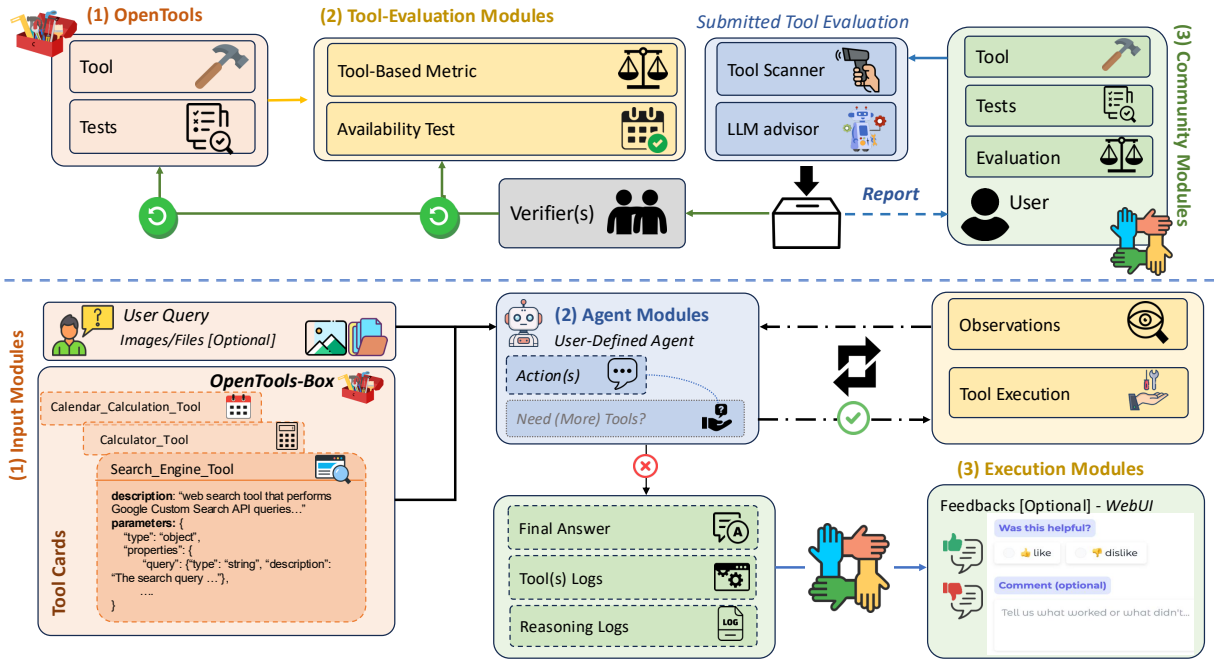

Figure 1: OPENTOOLS supports two complementary workflows. **Top, tool maintenance:** community members submit tools, tests, prior evaluations, and feedback. Non-executing scanners and an optional LLM advisor produce a review report, while human verifiers make the final acceptance decision and manually update the shared tools, tests, and reliability evidence. Accepted tools are assessed using tool-specific metrics and availability tests. **Bottom, agentic use:** users provide a query, optional files, and tool cards from the OPENTOOLS-BOX. Agent modules select actions and request tool execution, execution modules return observations, and the system produces a final answer with tool and reasoning logs. Users may provide optional feedback through the WebUI.

or automatically publish newly uploaded tools. Maintainers run functional tests separately for explicitly selected tools that satisfy the configured risk policy. Scanner findings, test results, and optional LLM recommendations provide supporting evidence, while a human maintainer makes the final acceptance decision. OPENTOOLS separates contribution-time risk assessment from deployment-time execution controls: its non-executing inspection provides structured evidence for maintainer review, while runtime isolation and access control can be configured according to the requirements of each deployment environment.

2.1.3 Community-Driven Maintenance

Community members contribute tools, tests, and failure reports through the repository or web interface. Maintainers review contribution bundles, and accepted cases become regression tests for later evaluations. To refresh reliability evidence, maintainers can manually reevaluate selected tools, regenerate tool cards and the shared inventory, and review the resulting changes before committing them. This process connects community feedback with repeatable evaluation while preserving maintainer control over the shared toolbox.

2.2 Agentic Workflow

Whereas the Maintenance Loop focuses on intrinsic tool reliability, the agentic workflow uses OPENTOOLS as a toolbox for tool-integrated LLM agents. It packages tool collections for general use across agent frameworks and provides end-to-end execution with transparent, debuggable traces.

Problem Definition. Given a user query q, a toolbox $\mathcal{S}$, and an agent policy π (e.g., prompting-based planning/routing or custom control flow), the agentic workflow produces an answer y by executing tool calls and intermediate reasoning steps. It also outputs an execution trace $\mathcal{L}$ that records tool usage and outcomes for reproducibility and failure diagnosis: $(q, \mathcal{S}, \pi) \rightarrow (y, \mathcal{L})$.

Workflow Overview. The agentic workflow has two phases: *(i) Agent Setup and Tool Access* and *(ii) Interactive Task Execution*.

2.2.1 Agent Setup and Tool Access

The setup phase specifies which tools the agent can use and how they are exposed.

Toolbox Interface. Agents access the standardized tool representations defined in Workflow 1 through the native OPENTOOLS interface or the Model Context Protocol. MCP supports discovery, inspection, evaluation, and invocation of explicitly allowlisted tools, enabling integration with external applications without tool-specific adapters.

Agent Policy Selection. OPENTOOLS includes several standard agent policies (e.g., ReAct (Yao et al., 2023b), OctoTools (Lu et al., 2026), and MultiAgent) and supports user-defined agents via custom controllers and workflows. The policy module determines action selection, and policies are treated as plug-and-play components, decoupling *tool access* from *decision logic*.

2.2.2 Interactive Task Execution

During execution, the agent iteratively updates the task state, invokes tools, and produces the final answer. An execution manager mediates between the policy and toolbox: it validates arguments against schemas, executes tool calls, returns results as observations, and records failures (e.g., exceptions, rate limits, timeouts). OPENTOOLS logs a structured trace of tool invocations, inputs/outputs, and error states, enabling reproducibility and attribution of failures to either policy errors (tool-use) or tool-side issues (intrinsic reliability), and linking behavior to Workflow 1 reliability signals. The system outputs the answer and trace for interactive inspection and offline analysis.

2.3 Web Demonstration Interface

OPENTOOLS provides a public web demonstration interface[3] that exposes both workflows without requiring local installation. Users can inspect tool cards and recorded evaluations, run supported tools and predefined agents, contribute test cases, submit open-source tools for review. Additional interface details are provided in Appendix A.1.

Tool Accuracy and Contribution. For Workflow 1, users can inspect schemas and usage guidance, run supported tools on custom inputs, and view the latest recorded evaluation. Individual or bulk test cases can be submitted as candidate regression cases. The *Contribute Tools* interface applies the conversion and non-executing inspection pipeline described above, presents its risk and scanner evidence, and optionally requests advisory LLM review of sanitized evidence. It then creates a pending-review bundle; uploaded programs are neither executed nor automatically published.

Agentic Workflow Interaction. For Workflow 2, the demo provides an end-to-end sandbox where users can input a task query, optionally restrict the tool subset, and run a *predefined* agent configuration (e.g., ReAct (Yao et al., 2023b), OctoTools (Lu et al., 2026)). The interface returns both the final response and a structured execution trace, including the sequence of tool calls, tool arguments, intermediate observations, and errors if any. Users can provide feedback on both tool-level outputs and the overall agent response, facilitating diagnosis of whether failures arise from tool-use decisions or from intrinsic tool issues.

3 Experiments

We evaluate OPENTOOLS under the *agentic workflow* to measure its end-to-end utility as an integrated, community-maintained toolbox. For each configuration, we compare agents equipped with OPENTOOLS-T, including community-contributed tools and tests, against the strong, curated OctoTools-T baseline while holding the base LLM, agent policy, benchmark instances, and evaluation protocol fixed. This system-level comparison examines whether replacing an existing toolbox with OPENTOOLS-T improves agent performance in practice. Our evaluation targets both breadth, spanning multiple benchmarks and domains, and generality, covering several agent policies.

3.1 Experimental Setup

3.1.1 Benchmarks.

Following the OctoTools testbed protocol (Lu et al., 2026), we evaluate a diverse suite of benchmarks spanning multiple domains and reasoning types, grouped into five task categories: **VQA/Puzzle** (AlgoPuzzleVQA (Ghosal et al., 2025), Hallusion-VD (Guan et al., 2024), PuzzleVQA (Chia et al., 2024), VQAv2 (Goyal et al., 2017)), **Math/Reasoning** (Game of 24 (Lile, 2024), Omni-MATH (Gao et al., 2024), CLEVR-Math (Lindström and Abraham, 2022), MathVista (Lu et al., 2023)), **Scientific** (GPQA (Rein et al., 2024), MMLU-Pro (Wang et al., 2024b), SciFiBench (Roberts et al., 2024)), **Medical** (MedQA (Jin et al., 2021), PathVQA (He et al., 2020), SLAKE (Liu et al., 2021)), and **Agent** (GAIA-Text (Mialon et al., 2023)).

[3] https://huggingface.co/spaces/opentools/opentools

	Prompting	ReAct		OctoTools		MultiAgent	
Task	-	OctoTools-T	*OpenTools-T*	OctoTools-T	*OpenTools-T*	OctoTools-T	*OpenTools-T*
VQA/Puzzle	45.38 / 74.36	55.52 / 73.13	**68.95 / 79.92**	52.13 / 69.75	**67.01 / 80.48**	72.40	**75.28**
Math/Reasoning	43.77 / 83.58	48.71 / 83.67	**59.03 / 85.33**	42.93 / 78.23	**55.54 / 81.28**	81.25	**83.15**
Scientific	46.54 / 80.20	50.67 / 75.67	**53.38 / 79.89**	47.87 / 76.11	**52.97 / 77.11**	69.48	**73.78**
Medical	43.56 / 69.56	52.00 / 69.11	**57.59 / 71.52**	53.00 / 66.78	**55.48 / 68.50**	66.05	**68.67**
Agent	5.51 / 21.41	14.17 / 51.97	**28.50 / 55.91**	4.72/ 46.03	**24.41 /53.54**	53.92	**66.14**
Average	42.16 / 73.49	49.27 / 74.23	**58.22 / 78.08**	45.84 / 71.10	**56.00 / 75.83**	71.67	**75.15**

Table 1: Task-group performance for tool-free Prompting and three agent approaches equipped with OctoTools-T or OPENTOOLS-T. Scores are reported as gpt-4o-mini / gpt-5-mini for Prompting, ReAct, and the OctoTools agent; MultiAgent reports gpt-5-mini only. For each toolbox comparison, the base model, agent policy, benchmark instances, and evaluation protocol are fixed. OPENTOOLS-T improves the overall average across every evaluated tool-using configuration. Higher is better.

	OctoTools-T	OPENTOOLS-T
Total tools	13	42
Program-based	1	16
API-based	8	19
Prompting-based	4	7

Table 2: Composition of the evaluated toolboxes by primary execution mechanism.

3.1.2 Baselines

Toolboxes We compare two experimental toolboxes. **OctoTools-T** is the 13-tool collection used by the OctoTools testbed (Lu et al., 2026), including a general solution generator and domain tools for web search, image understanding, code generation, and classification. **OPENTOOLS-T** contains 42 tools drawn from three sources: general-purpose tools such as search and calculation, curated domain tools for areas such as vision and chemistry, and community-contributed tools such as specialized maze solvers. All OPENTOOLS-T tools follow the common interface described in Section 2 and can be reevaluated through the Tool Accuracy and Maintenance Loop.

We categorize tools by their primary execution mechanism. **Program-based** tools perform local computation without relying on remote services. **API-based** tools wrap externally hosted services, which may require user-provided credentials. **Prompting-based** tools implement model-mediated functions through fixed prompt templates. Table 2 summarizes the resulting composition.

The OPENTOOLS-T implementations include newly developed tools and open-source wrappers adapted from ecosystems such as OctoTools (Lu et al., 2026) and AWorld (Yu et al., 2025). Adapted tools are refactored to use consistent schemas across agent policies. This broader composition allows us to examine how tool coverage and intrinsic tool quality affect end-to-end agent performance.

Agent Frameworks. We evaluate our systems under multiple agent frameworks: **(i) Prompting:** the LLM answers directly without tool use. We experiment with both zero-shot and chain of thought prompting (CoT), which are step-by-step reasoning without external tools). In preliminary experiments, zero-shot performs better overall for both gpt-4o-mini and gpt-5-mini; therefore, we report zero-shot as *Prompting* in the main results and provide the full zero-shot/CoT comparison in Appendix A.2. **(ii) ReAct Agent (Yao et al., 2023b):** an iterative reason-and-act agent that interleaves natural-language reasoning with tool calls. **(iii) OctoTools Agent (Lu et al., 2026):** the planner–executor style agent as defined in the OctoTools framework. **(iv) MultiAgent Framework:** Inspired by LemonAgent (Jiang et al., 2026), we use a hierarchical pipeline that decomposes a query and routes subproblems to specialized sub-agents. Unlike LemonAgent, we (i) perform explicit reasoning-based decomposition before routing and (ii) use a *Verifier*/guardrail to store only validated intermediate results in shared memory. A *Planner* decomposes the query, a *Generator* issues tool calls, tools are executed and verified, validated results are written to memory, and a *Composer* generates the final answer with a brief memory-based summary.

For tool-using agents, we run each framework with (i) the OctoTools toolbox (OctoTools-T) and (ii) our OPENTOOLS toolbox (OpenTools-T), enabling a controlled comparison of toolsets under the same decision logic. Following OctoTools (Lu et al., 2026), we perform greedy tool selection for OpenTools-T to choose task-appropriate tool sub-

sets. For OctoTools-T, we use the tool sets reported in the OctoTools paper that achieve the best performance for each task.

4 Results & Analysis

Table 1 compares OctoTools (OctoTools-T) and our OpenTools toolbox (OPENTOOLS-T) under agent frameworks. We report task-group accuracy in Table 1 using the same sampled instance identifiers as OctoTools for direct comparability. We run three independent trials per setting and report per-benchmark mean $\pm$ std. in Appendix A.

Tool Augmentation Effect. For the weaker model (gpt-4o-mini), tool augmentation consistently outperforms tool-free prompting. With OctoTools-T, ReAct and OctoTools gain **+16.86%** and **+8.70%** (relative) over Prompting; with OPENTOOLS-T, the gains rise to **+38.09%** and **+32.83%**. This indicates that tools substantially boost smaller LLMs. For the stronger model (gpt-5-mini), tool-free prompting is already strong on knowledge/reasoning-heavy tasks (VQA/Puzzle, Math/Reasoning, Scientific, Medical), so tool gains are smaller and can even turn negative with weaker toolbox. With OctoTools-T, **ReAct** improves over Prompting by only **+1%** on average, and **OctoTools** can underperform Prompting. In contrast, with OPENTOOLS-T, both **ReAct** and **OctoTools** outperform **Prompting** (78.08 and 75.83 vs. 73.49 on average), indicating that *tool reliability and coverage* still matter even for strong base models.

Tools matter most on the hardest Agent task, where external actions are required. Compared to **Prompting**, **ReAct** with OctoTools-T/OPENTOOLS-T improves by **143%/161%** (relative), and **OctoTools** with OctoTools-T/OPENTOOLS-T improves by **115%/150%**.

OPENTOOLS-T Provides Consistent End-to-End Gains. Replacing OctoTools-T with OPENTOOLS-T improves average performance across every evaluated agent framework and task category. The relative gains are **18.16%/5.19%** for **ReAct** and **22.16%/6.65%** for the **OctoTools** agent (gpt-4o-mini / gpt-5-mini), while **MultiAgent** improves by **4.86%**. Overall, OPENTOOLS-T yields relative improvements of approximately **4.9%–22.2%** across the evaluated tool-using configurations. The consistency of these gains across models, agent policies, and domains demonstrates the practical end-to-end value of a broader, standardized, and maintainable toolbox.

The largest improvements concentrate on the **Agent** task, where tool breadth and robustness are crucial. With the stronger model, OPENTOOLS-T improves over OctoTools-T by **+7.58%** (**ReAct**), **+16.32%** (**OctoTools**), and **+22.67%** (**MultiAgent**) on Agent. **MultiAgent** with OPENTOOLS-T achieves the best Agent score overall (66.14), but we also observe that complex agent policies can hurt on simpler tasks when paired with a less reliable toolbox (e.g., **MultiAgent** with OctoTools-T underperforms **Prompting** on average). Overall, OPENTOOLS-T provides the most consistent gains, especially for tool-intensive end-goal tasks.

5 Related Work

Tool use and agentic LLMs. Prior work on tool use and agentic LLMs focuses on how models plan, and invoke external tools through training or prompting, including Toolformer-style learning (Schick et al., 2023), ReAct reasoning-and-acting (Yao et al., 2023b), and agent controllers that add reflection/search and multi-step tool execution (Patil et al., 2024; Dang et al., 2025). These lines of work primarily improve tool-use accuracy, while often assuming the underlying tools are correct and stable.

In parallel, tool libraries and evaluations (e.g., LangChain[4], AutoGen[5]) broaden tool access and provide domain-specific suites (e.g., chemistry (Wu et al., 2025), medical (Li et al., 2024a), vision (Hu et al., 2024)). Existing benchmarks assess end-to-end task success and tool-calling competence (Huang et al., 2023; Guo et al., 2024), but frequently conflate model mistakes with tool-side failures and seldom measure tool drift or regressions over time. OPENTOOLS addresses this infrastructure gap by standardizing JSON-schema tool interfaces, providing community-maintained wrappers, and enabling intrinsic tool evaluation with reliability signals and regression tracking—supporting plug-and-play use across agent frameworks while separating tool-use from tool reliability.

6 Conclusion and Future Work

We introduced OPENTOOLS, a community-driven framework for discovering, contributing, evaluating, and using open-source tools with LLM agents.

[4]https://github.com/langchain-ai/langchain
[5]https://github.com/microsoft/autogen

The framework distinguishes *tool-use accuracy* from *intrinsic tool accuracy* and combines standardized interfaces, automated risk and reliability evidence, regression testing, and maintainer verification in a review-oriented lifecycle. Its public web interface supports community participation, while MCP enables integration with external applications. Across 15 benchmarks, replacing OctoTools-T with OPENTOOLS-T under otherwise fixed evaluation settings improves average performance across every evaluated tool-using configuration, with relative gains of approximately **6%–22%**. These results demonstrate the practical value of a broader, task-relevant, standardized, and maintainable tool ecosystem.

Future Work. Beyond supporting current agents, OPENTOOLS provides a shared foundation for tool-reliability research. We encourage the community to contribute matched implementations, failure cases, and regression tests, enabling controlled comparisons, repair analysis, fault injection, and longitudinal evaluation of tool drift. Future work will also expand domain coverage and study community participation and maintenance at scale.

7 Limitations

Our evaluation measures the end-to-end utility of OPENTOOLS. With models, agent policies, benchmark instances, and selection protocols fixed, replacing OctoTools-T with OPENTOOLS-T improves average performance across all evaluated tool-using configurations. Because the toolboxes differ in size and coverage, the results do not isolate the contribution of each factor. Nevertheless, broader, task-relevant coverage is an intended property of a practical toolbox, and the consistent gains demonstrate the value of the integrated OPENTOOLS ecosystem. We have not yet evaluated community adoption, maintainer effort, or long-term maintenance costs at scale. Moreover, static inspection, secret detection, testing, and advisory LLM review reduce risk but cannot guarantee correctness or safety. Although uploaded tools are neither executed nor automatically published, deployed tools may access networks, files, credentials, or user data. Deployments therefore require appropriate isolation, least-privilege access, resource limits, and human oversight.

Acknowledgements

This work was partially supported by NSF IIS-2119531, IIS-2137396, IIS-2142827, IIS-2234058, and Coefficient Giving. We also appreciate the support from the Foundation Models and Applications Lab of Lucy Institute and ND-IBM Tech Ethics Lab.

References

Sören Arlt, Xuemei Gu, and Mario Krenn. 2025. Towards autonomous quantum physics research using llm agents with access to intelligent tools. *arXiv preprint arXiv:2511.11752*.

Yew Ken Chia, Vernon Toh, Deepanway Ghosal, Lidong Bing, and Soujanya Poria. 2024. Puzzlevqa: Diagnosing multimodal reasoning challenges of language models with abstract visual patterns. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16259–16273.

Hy Dang, Tianyi Liu, Zhuofeng Wu, Jingfeng Yang, Haoming Jiang, Tao Yang, Pei Chen, Zhengyang Wang, Helen Wang, Huasheng Li, Bing Yin, and Meng Jiang. 2025. Improving large language models function calling and interpretability via guided-structured templates. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24437–24453, Suzhou, China. Association for Computational Linguistics.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, and 1 others. 2024. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2.

Deepanway Ghosal, Vernon Toh, Yew Ken Chia, and Soujanya Poria. 2025. Algopuzzlevqa: Diagnosing multimodal reasoning challenges of language models with algorithmic multimodal puzzles. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9615–9632.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, and 1 others. 2024. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14375–14385.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11143–11156, Bangkok, Thailand. Association for Computational Linguistics.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. 2020. Pathvqa: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. 2024. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *Advances in Neural Information Processing Systems*, 37:139348–139379.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and Lichao Sun. 2023. Metatool benchmark: Deciding whether to use tools and which to use. *arXiv preprint arXiv: 2310.03128*.

Dongsuk Jang, Ziyao Shangguan, Kyle Tegtmeyer, Anurag Gupta, Jan T Czerminski, Sophie Chheang, and Arman Cohan. 2025. Medtutor: A retrieval-augmented llm system for case-based medical education. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 319–353.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843.

Haipeng Jiang, Kailong Ren, Zimo Yin, Zhetao Sun, Xin Gan, Guangyi Lv, Ming He, Peng Wang, Congli Yin, Hong Pan, and 1 others. 2026. Lemon agent technical report. *arXiv preprint arXiv:2602.07092*.

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. *arXiv preprint arXiv:2503.09516*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Binxu Li, Tiankai Yan, Yuanting Pan, Jie Luo, Ruiyang Ji, Jiayuan Ding, Zhe Xu, Shilong Liu, Haoyu Dong, Zihao Lin, and 1 others. 2024a. Mmedagent: Learning to use medical tools with multi-modal agent. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8745–8760.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024b. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1.

Nathan Lile. 2024. Math twenty four (24s game) dataset. https://huggingface.co/datasets/nlile/24-game. Hugging Face Datasets.

Adam Dahlgren Lindström and Savitha Sam Abraham. 2022. Clevr-math: A dataset for compositional language, visual and mathematical reasoning. *arXiv preprint arXiv:2208.05358*.

Bo Liu, Li-Ming Zhan, Li Xu, Lin Ma, Yan Yang, and Xiao-Ming Wu. 2021. Slake: A semantically-labeled knowledge-enhanced dataset for medical visual question answering. In *2021 IEEE 18th international symposium on biomedical imaging (ISBI)*, pages 1650–1654. IEEE.

Qiang Liu, Xinlong Chen, Yue Ding, Bowen Song, Weiqiang Wang, Shu Wu, and Liang Wang. 2025. Attention-guided self-reflection for zero-shot hallucination detection in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21016–21032, Suzhou, China. Association for Computational Linguistics.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. 2026. OctoTools: A multi-agent framework with extensible tools for complex reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–86, San Diego, California, United States. Association for Computational Linguistics.

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*.

Ivan Milev, Mislav Balunović, Maximilian Baader, and Martin Vechev. 2025. Toolfuzz–automated agent tool testing. *arXiv preprint arXiv:2503.04479*.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

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.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Jonathan Roberts, Kai Han, Neil Houlsby, and Samuel Albanie. 2024. Scifibench: Benchmarking large multimodal models for scientific figure interpretation. *Advances in Neural Information Processing Systems*, 37:18695–18728.

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.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Cailin Winston and René Just. 2025. A taxonomy of failures in tool-augmented llms. In *2025 IEEE/ACM International Conference on Automation of Software Test (AST)*, pages 125–135. IEEE.

Mengsong Wu, YaFei Wang, Yidong Ming, Yuqi An, Yuwei Wan, Wenliang Chen, Binbin Lin, Yuqiang Li, Tong Xie, and Dongzhan Zhou. 2025. Chemagent: Enhancing llms for chemistry and materials science through tree-search based tool learning. *arXiv preprint arXiv:2506.07551*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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

Chengyue Yu, Siyuan Lu, Chenyi Zhuang, Dong Wang, Qintong Wu, Zongyue Li, Runsheng Gan, Chunfeng Wang, Siqi Hou, Gaochi Huang, and 1 others. 2025. Aworld: Orchestrating the training recipe for agentic ai. *arXiv preprint arXiv:2508.20404*.

Xiang Zhang and Dujian Ding. 2024. Supervised chain of thought. *arXiv preprint arXiv:2410.14198*.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. 2025. KnowAgent: Knowledge-augmented planning for LLM-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3709–3732, Albuquerque, New Mexico. Association for Computational Linguistics.

A Appendix

A.1 Web Demonstration Interface

Tool Contribution. As described in Section 2.3, users can contribute an open-source tool by uploading Python and README files, providing tool metadata, and optionally requesting LLM review. The interface applies deterministic conversion and non-executing inspection, then returns a pending-review bundle containing the standardized wrapper, tool card, and review evidence. The contributor-facing report presents conversion status, overall risk, scanner coverage, detailed findings, recommended actions, and the optional LLM recommendation. Unavailable scanners are reported explicitly rather than treated as successful checks. Uploaded code is not executed or automatically published; maintainers review the bundle and run functional tests separately before acceptance. Figure 2 shows the submission and reporting interfaces.

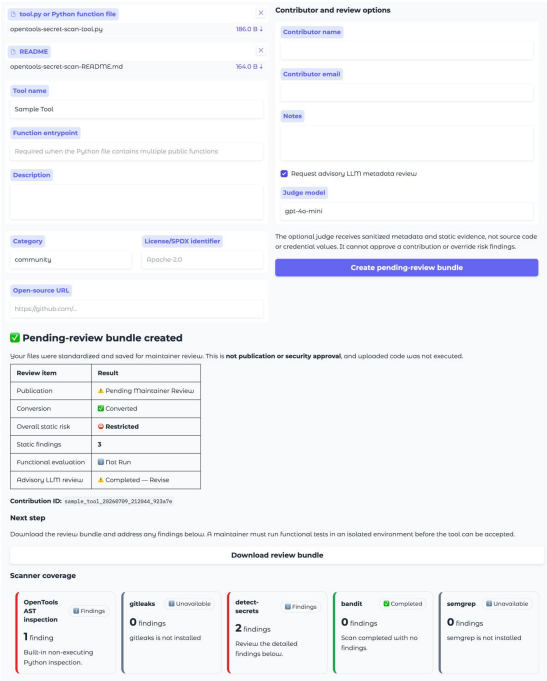

Figure 2: OpenTools tool-contribution interface. Contributors upload a Python implementation and README, provide metadata, and may request advisory LLM review. The interface performs deterministic conversion and non-executing inspection, then reports conversion status, static risk, scanner coverage and findings, and the LLM recommendation. Uploaded code remains unexecuted and unpublished, and the resulting pending-review bundle is returned for contributor revision and maintainer review.

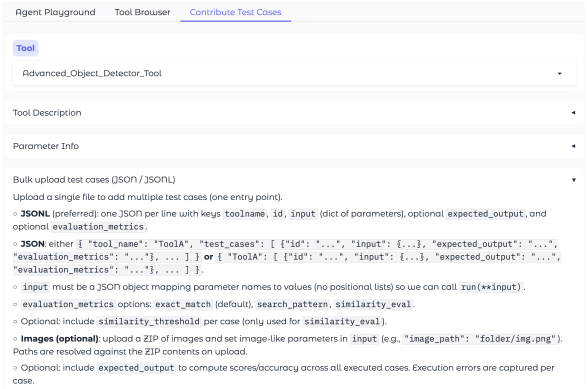

Figure 3: Test-case contribution UI: submission instructions and supported formats for adding new tool tests.

Test Case Contribution As described in Section 2.3, the web interface allows users to contribute test cases for tool evaluation. Submitted cases are reviewed by our team before being added to the shared OpenTools test suite. Figure 3 shows the submission instructions and supported formats. Figure 4 shows the authoring interface, where users can run tools on submitted cases and store outputs in a bulk-results table. The evaluation uses metrics specified in the submission or selected from a predefined list on the site.

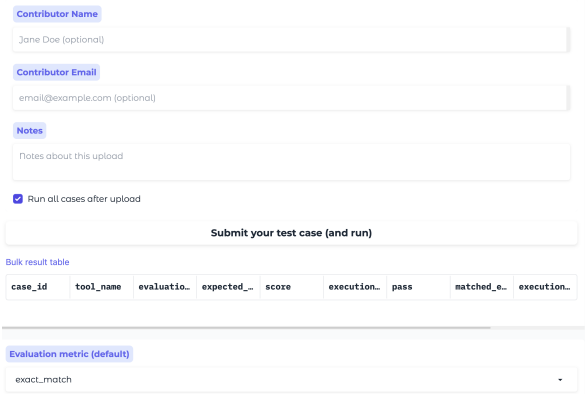

Figure 4: Test-case authoring/execution UI: run tools on submitted cases and store outputs in a bulk-results table with selectable evaluation metrics.

Agent Workflow Figure 5 shows the Agent Playground in our web demo, where users select a predefined agent and a base LLM. Users can optionally enable and configure tool retrieval to filter tools before exposing them to the agent. The interface displays agent responses and detailed execution logs in the Agent Answer boxes, and users may also provide feedback on the workflow.

Comparison between Prompting Approaches We compare two tool-free baselines: Zero-shot

Agent	ReAcT		OctoTools		MultiAgent	
Datasets	**OctoTools-T**	**OpenTools-T**	**OctoTools-T**	**OpenTools-T**	**OctoTools-T**	**OpenTools-T**
VQA / Puzzle						
AlgoPuzzleVQA	39.66±2.39 / 71.66±0.84	77.11±0.95 / 85.22±1.64	38.56±2.04 / 55.33±2.04	73.00±0.72 / 86.00±1.54	70.25±3.03	75.67±2.02
Hallusion-VQA	65.56±2.53 / 67.50±0.87	68.00±1.64 / 70.33±0.15	62.78±1.36 / 69.00±1.96	66.00±1.18 / 75.33±2.10	66.55±1.83	67.33±0.82
PuzzleVQA	36.89±1.75 / 72.01±1.93	41.03±1.28 / 73.79±1.85	48.05±1.06 / 70.68±2.57	46.83±1.75 / 73.91±1.13	71.56±2.97	73.45±1.93
VQA 2.0	79.96±1.82 / 81.33±0.99	89.64±0.68 / 90.33±1.39	59.11±1.64 / 84.00±1.64	82.22±1.13 / 86.66±1.27	81.25±1.48	84.67±1.19
Math / Reasoning						
Game of 24	76.66±1.32 / 100.00±0.00	98.67±0.98 / 100.00±0.00	40.89±1.54 / 98.33±1.32	91.22±0.68 / 100.00±0.00	100.00±0.84	100.00±0.00
Omni-MATH	27.50±0.37 / 74.67±0.65	33.00±2.05 / 76.78±0.41	30.55±1.59 / 71.57±1.87	29.33±1.15 / 73.24±2.56	70.13±1.54	71.91±1.45
CLEVR-Math	38.66±1.56 / 85.00±1.76	48.77±1.50 / 87.22±1.22	47.44±1.25 / 72.00±1.87	48.03±1.44 / 78.20±1.86	84.62±1.73	89.67±2.87
MathVista	52.00±3.58 / 75.00±2.87	55.67±2.18 / 77.33±1.49	52.84±1.91 / 71.00±1.74	53.56±0.41 / 73.67±3.56	70.23±3.51	71.00±2.34
Knowledge / QA						
GPQA	35.67±2.69 / 70.33±3.37	39.22±4.79 / 75.33±0.56	37.17±2.86 / 72.00±3.38	38.01±0.68 / 73.00±1.86	69.38±3.64	73.67±0.79
MMLU-Pro	54.33±3.46 / 74.67±2.56	56.69±0.68 / 80.00±0.97	40.67±0.47 / 72.33±1.28	57.33±1.02 / 74.67±0.96	73.38±2.37	78.00±1.34
SciFIBench	62.00±0.85 / 82.00±1.07	64.22±0.87 / 84.33±1.18	65.78±0.56 / 84.00±0.56	63.56±0.95 / 83.67±0.63	65.67±0.84	69.67±1.28
Medical / Multimodal						
MedQA	76.67±1.09 / 92.00±1.27	79.33±1.73 / 94.89±0.15	77.00±0.40 / 92.67±0.43	76.44±0.47 / 94.33±1.68	90.11±1.25	93.00±0.23
PathVQA	31.33±3.63 / 45.33±0.68	39.45±1.36 / 47.89±0.81	35.50±0.81 / 42.00±1.37	36.66±1.02 / 43.50±0.36	44.83±1.18	46.33±0.49
SLAKE	48.00±0.25 / 70.00±1.35	54.00±0.47 / 71.78±0.83	46.50±1.77 / 65.66±2.37	53.33±0.26 / 67.67±0.39	63.21±2.55	66.67±1.65
General / Agent						
GAIA-Text	14.17±2.11 / 51.97±3.48	28.5±2.15 / 55.91±1.61	4.72±0.64 / 46.03±2.53	24.41±1.56 / 53.54±1.75	53.92±3.37	66.14±2.37
Average (%)	49.27 / 74.23	**58.22 / 78.08**	45.84 / 71.1	**56.00 / 75.83**	71.67	**75.15**

Table 3: Per-benchmark results for 15 benchmarks with GPT-4o-mini and GPT-5-mini as base models. We report mean ± std over 3 runs; each entry is formatted as GPT-4o-mini / GPT-5-mini. MultiAgent reports a single gpt-5-mini score. OpenTools-T improves the overall average across all frameworks, demonstrating that higher intrinsic tool quality can translate into better end-to-end reliability. Higher means better.

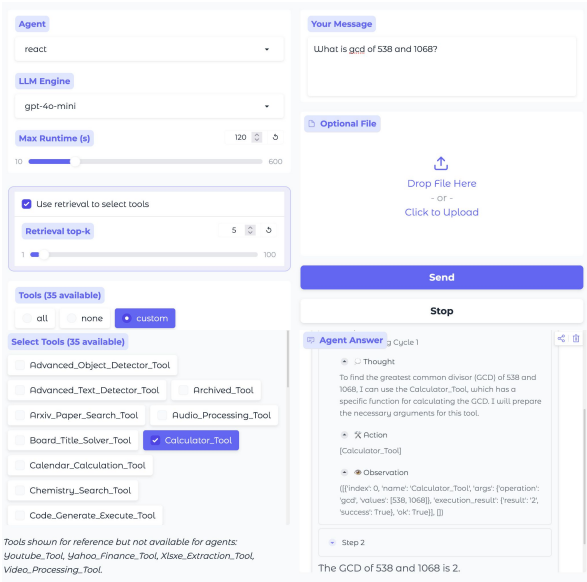

Figure 5: Agent Playground UI: select an agent and base LLM, optionally filter tools via retrieval, and inspect execution logs, responses

prompting and Chain-of-Thought (CoT) prompting. CoT elicits explicit step-by-step reasoning using the template in Table 4. Overall, the two approaches perform similarly; notably, the stronger model (gpt-5-mini) performs better in zero-shot than in CoT (73.49 vs. 71.45), suggesting improved intrinsic reasoning without additional instructions.

	ZeroShot	CoT
VQA/Puzzle	45.38 / **74.36**	**46.95** / 74.20
Math/Reasoning	**43.77** / **83.58**	41.38 / 76.16
Scientific	46.54 / 80.20	**48.89** / **80.74**
Medical	43.56 / **69.56**	**45.33** / 69.33
Agent	**5.51** / **21.41**	4.46 / 20.01
Average	42.16 / **73.49**	**42.70** / 71.45

Table 4: Task-group performance for Prompting without tools between ZeroShot and Chain of Thought (CoT). Scores are reported as gpt-4o-mini / gpt-5-mini.

A.2 Experiment Results

Comparison Between Toolboxes. Table 3 reports per-benchmark mean ± std. over three trials for each agent framework under two tool settings, OctoTools-T and OpenTools-T. Consistent with the main results, swapping OctoTools-T for OpenTools-T improves performance across frameworks and domains, indicating that higher tool reliability and coverage translate into better end-to-end agent outcomes. The gains are most pronounced on the **Agent** benchmark, where robust external actions are required, and they are especially important for the weaker base model (gpt-4o-mini); for the stronger model (gpt-5-mini), improvements remain consistent, though smaller on knowledge/reasoning-heavy tasks.