OMNIABENCH: BENCHMARKING GENERAL AI AGENTS ACROSS DIVERSE SCENARIOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models are increasingly evolving from text generators into general agents capable of understanding user requests, invoking external tools, and completing complex tasks through interaction. However, existing agent benchmarks often focus on limited scenarios, tool ecosystems, or interaction formats, making it difficult to systematically characterize model capabilities across heterogeneous application settings. We introduce **OmniaBench**, a benchmark for evaluating general agents across diverse scenarios with explicit state spaces. We derive application-oriented scenario knowledge from app stores, product documents, industry resources, Web retrieval, and human refinement, forming a hierarchical taxonomy that spans ToC, ToB and ToE with 90 level-1 and 354 level-2 domains. Based on this taxonomy, we construct executable environments and synthesize single-turn and multi-turn tasks through four complementary routes: DAG, DAG-S, Solver, and Program. OmniaBench further introduces a ten-dimensional capability taxonomy and eight compositional atomic difficulty factors to support fine-grained evaluation and analysis. The resulting dataset contains 1,431 tasks, together with a challenging subset of 644 tasks designed to reduce evaluation cost and mitigate potential contamination of the full set after public release. The bench presents substantial challenges to current frontier models, with even Claude-Sonnet-5 and GPT-5.6-Sol achieving Overall Pass@1 scores of only 58.54 and 57.14, respectively. Further analyses reveal clear differences across domains and capabilities, as well as persistent limitations in planning, constraint maintenance, and adaptive correction. OmniaBench provides a broad and diagnostic benchmark for characterizing the capability boundaries of general agents.

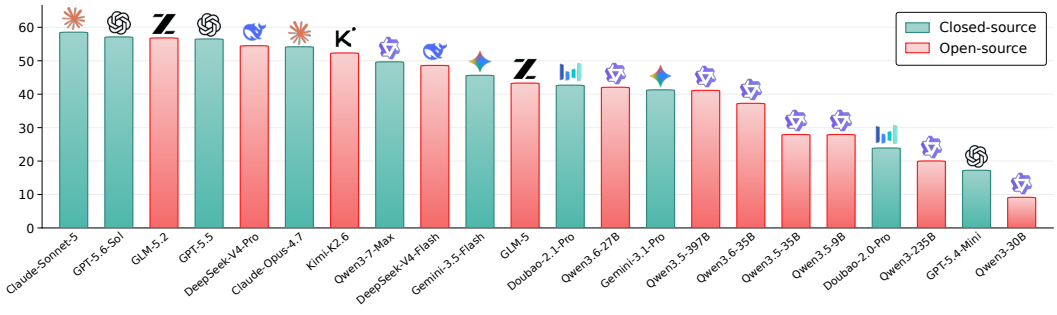


Figure 1: **OmniaBench leaderboard.** Overall performance of models on the challenging set.

1 INTRODUCTION

Recent advances in large language models (LLMs) have improved language understanding, reasoning, code generation, and multi-turn interaction (DeepSeek-AI et al., 2026; GLM-5-Team et al., 2026; Team et al., 2026), enabling a shift from text generation to agentic task execution (Liang et al., 2025; Dong et al., 2026). With function calling, retrieval, and code execution, LLM-based agents support software development, information seeking, office automation, business workflows, and personal assistance (Yao et al., 2024; Boisvert et al., 2024; Yang et al., 2024; Jimenez et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

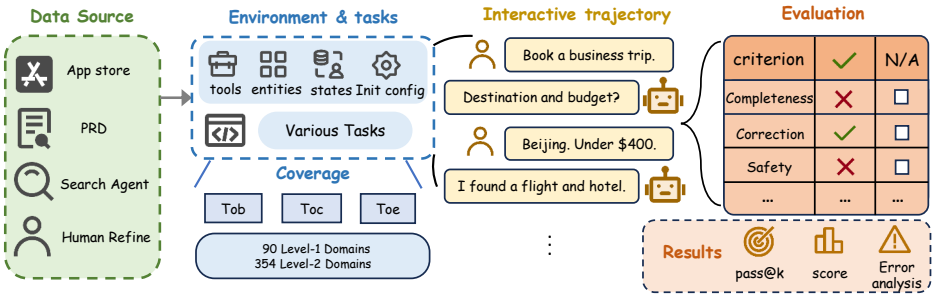


Figure 2: **Conceptual overview of OmniaBench.** Application resources inform executable environments and interactive tasks across ToC, ToB, and ToE domains. Rubric- and code-based evaluation supports aggregate scoring and capability analysis.

2024). Compared with conventional question answering or isolated reasoning (Cobbe et al., 2021; Wang et al., 2024), these tasks require agents to interpret user intents, plan multiple steps, invoke tools, and track evolving states through interaction.

Existing agent benchmarks evaluate complementary aspects of these abilities. Function-calling and tool-use benchmarks assess tool selection, argument generation, call composition, and execution constraints (Patil et al., 2025; Chen et al., 2025; Li et al., 2025; Shen et al., 2026). Interactive benchmarks introduce simulated users, dynamic states, and domain rules (Yao et al., 2024; Barres et al., 2025; He et al., 2025). Recent work extends evaluation to long-horizon planning, professional workflows, web and application environments, multimodal evidence grounding, and standardized tool ecosystems (Froger et al., 2026; Zhang et al., 2026; Patwardhan et al., 2025; Backlund & Petersson, 2025; Bandi et al., 2026; Meng et al., 2026). However, individual benchmarks often cover limited scenarios, tool ecosystems, or task formats. Evaluating general agents across heterogeneous applications and user needs therefore calls for broader scenario coverage, varied interaction modes, and diagnostic capability analysis.

We introduce **OmniaBench**, an interactive benchmark grounded in real-world application ecosystems. As illustrated in Figure 2, we organize scenario knowledge from app stores, product requirement documents (PRDs), web-search agents, and human refinement into a taxonomy spanning consumer-facing (ToC), business-facing (ToB), and employee-oriented (ToE) settings, with 90 level-1 and 354 level-2 domains. We instantiate these domains as executable environments with tools, entities, states, and initialization configurations, then construct diverse tasks requiring agents to complete user goals through tool use and interaction.

To support diagnosis beyond overall success rates, OmniaBench defines ten capability dimensions: Task Understanding, Information Gathering, Planning & Decision Making, State Management, Tool Use, Code & Programmatic Operations, Data Analysis, Office & Document Handling, Interactive Collaboration, and Reliability & Safety. Eight compositional atomic difficulty factors characterize challenges involving user intent, persona constraints, environment states, tool execution, interaction, and verification. Rubric checklists and *VerifyCode* assess task completion, supporting analysis across domains, capabilities, complexity, interaction turns, and error patterns.

Our contributions are summarized as follows:

- We introduce **OmniaBench**, a benchmark with broad scenario coverage and 1,431 tasks, including a challenging subset of 644 tasks. Claude-Sonnet-5 and GPT-5.6-Sol achieve Overall Pass@1 scores of only 58.54% and 57.14% on this subset, respectively.
- We develop a systematic evaluation framework combining a ten-dimensional **capability taxonomy** with eight compositional **atomic difficulty** factors, enabling fine-grained diagnosis of general-agent abilities and controlled construction of tasks with diverse challenges.
- We conduct a comprehensive evaluation of diverse models across domains, capabilities, task complexity, interaction turns, and error patterns, revealing current capability boundaries and evaluation stability.

2 Related Work

Agent evaluation has expanded from static question answering to tool use, environment interaction, and task execution. Foundational benchmarks such as BFCL v3 (Patil et al., 2024), BFCL v4 (Patil et al., 2025), ACEBench (Chen et al., 2025), and ToolSandbox (Lu et al., 2025) assess tool selection, argument generation, call composition, and execution control, covering different invocation patterns and stateful execution.

Table 1: Comparison with representative agent benchmarks. ✓indicates full support, ▲indicates partial support or a different formulation, and ✗indicates unsupported or not applicable. Avg. Tools / Env. is normalized by the number of environments, applications, domains, or servers when such grouping is available. Var. denotes task-dependent values, N/A denotes benchmarks without tool-based environments, and N/R denotes values not reliably reported in public sources.

Benchmark	Environment & Task			Domain & Capability		Interaction			
	Tasks	Avg. Tools / Env.	State Init.	Domains Taxonomy	Capability Taxonomy	Approx Turns	Persona	Interaction Mode	Workspace
Toolathlon	108	18.9	✓	▲ (32 apps)	▲	26.8	✗	Multi	✓
VitaBench	400	66	✓	✓ (3)	▲	50–100	✓	Multi	▲
GAIA2	800 / 1120	101	✓	✓ (10)	✓ (7)	22.5	▲	Multi	▲
ToolSandbox	1032	34.0	✓	✓ (11)	▲	13.9	▲	Both	✗
ACEBench	2000	66.7	▲	✓ (8/68)	▲	1.7	▲	Both	✗
τ^2-Bench	279	12.7	✓	✓ (3)	▲	30–80	▲	Multi	✗
τ-knowledge	97	51	✓	▲ (1)	▲	N/R	✓	Both	▲
BFCL v3	4441	Var.	✓	▲	▲	Var.	▲	Both	▲
BFCL v4	5088	Var.	✓	▲	▲	Var.	▲	Both	▲
MCP-Atlas	1000	6.1	▲	▲ (36 servers)	▲	3–6 calls	✗	Single	▲
DeepPlanning	360	12.0	✓	✓ (2)	✓ (3)	Var.	✗	Single	✗
GDPval	1320	N/A	✗	✓ (44/9)	✗	1	✗	Single	▲
Vending-Bench	1	10	✓	✗	✗	3k–6k msg.	▲	Multi	✗
Claw-Eval	300	Var.	✓	✓ (9)	✓ (3)	Var.	▲	Both	✓
ClawMark	100	5.0	✓	✓ (13)	▲	1–3 days	▲	Multi	✓
QwenClawBench	100	N/R	✓	✓ (8)	✗	N/R	✗	Single	✓
OmniaBench	**644 / 1431**	**65.32**	✓	✓ **(90/354)**	✓ **(10)**	100	✓	Both	✓

Recent benchmarks extend evaluation to richer applications and interaction settings. MCP-Atlas (Bandi et al., 2026) and Toolathlon (Li et al., 2025) evaluate task completion across realistic tools, services, and applications. τ^2-Bench (Barres et al., 2025) and τ^3-Bench (Shi et al., 2026) combine simulated users, domain rules, and stateful databases for multi-turn service tasks. VitaBench (He et al., 2025), GAIA2 (Froger et al., 2026), and DeepPlanning (Zhang et al., 2026) examine real-life services, open-ended tasks, long-horizon planning, and combinations of capabilities. GDPval (Patwardhan et al., 2025) and Vending-Bench (Backlund & Petersson, 2025) study professional tasks and long-running business decisions, respectively. Claw-Eval (Ye et al., 2026), ClawMark (Meng et al., 2026), and QwenClawBench (Qwen Team & Alibaba Data, 2026) focus on workspace or computer-use agents operating over files, documents, spreadsheets, code, and multi-service sandboxes.

OmniaBench builds on these efforts by combining broad domain coverage with multiple interaction modes and capability analysis (Table 1). Its single-turn tasks assess execution from an initial request, while multi-turn tasks additionally require clarification, feedback use, state maintenance, and iterative correction. Grounded in real-world application resources, OmniaBench integrates domain and capability taxonomies, stateful environments, and user personas. It unifies these interaction modes with workspace, coding, and tool-use tasks to diagnose agent capabilities across diverse scenarios.

3 OmniaBench: Evaluating General Agents in Diverse Scenarios

3.1 Formulation

We formulate a general-agent task as a partially observable Markov decision process (POMDP),

$$\mathcal{P} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \Omega, \rho_0),$$

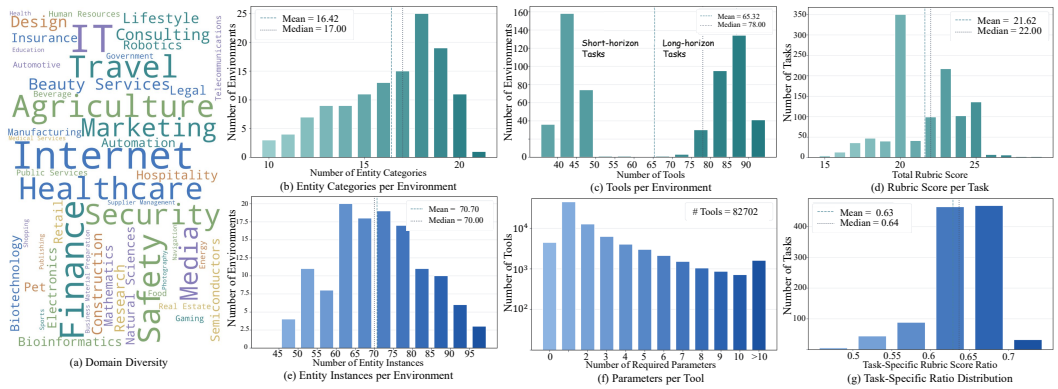


Figure 3: **Statistics of OmniaBench.** Comprehensive statistics of the benchmark, including domain diversity, environment complexity, tool availability, task difficulty, and rubric characteristics.

where $\mathcal{S}$, $\mathcal{O}$, and $\mathcal{A}$ denote the state, observation, and action spaces; $\mathcal{T}$ and Ω define state transitions and observations; and ρ_0 is the initial state distribution. The latent state includes user information, environment records, and workspace contents. The agent observes only user messages, tool outputs, and environment feedback, and selects actions from its interaction history:

$$h_t = (o_0, a_0, \ldots, o_t), \qquad a_t \sim \pi_\theta(\cdot \mid h_t),$$
$$s_{t+1} \sim \mathcal{T}(\cdot \mid s_t, a_t), \qquad o_{t+1} \sim \Omega(\cdot \mid s_{t+1}, a_t).$$

Actions comprise user-facing messages, tool calls, and final submissions. File operations and code execution use controlled tool interfaces, with results returned as observations.

Users form part of the environment dynamics. In multi-turn tasks, user feedback progressively reveals information and constraints, requiring clarification and plan revision; interaction ends with a user-side stop signal. Single-turn tasks provide the request initially and end with the agent's final submission. Both settings also enforce execution budgets and environment-level failure conditions.

For a trajectory $\Gamma = (o_0, a_0, \ldots, o_T, a_T)$, task goal g, and evaluation protocol $\mathcal{V}$,

$$\mathrm{Eval}_{\mathcal{V}}(\Gamma, g) \to (y, r),$$

where $y \in \{0, 1\}$ indicates success and r contains rubric scores when applicable. Evaluation checks task completion from the trajectory and final observation using rubrics or programmatic verification.

3.2 Taxonomy

OmniaBench connects application coverage, required capabilities, and sources of complexity through three taxonomy views. The domain view spans consumer-facing (ToC), business-facing (ToB), and employee-oriented (ToE) settings, with 90 level-1 and 354 level-2 domains. The capability view describes what agents must do through ten dimensions, while eight compositional difficulty factors describe why tasks are challenging. These views guide environment and task construction and support fine-grained analysis. Appendix Table 5 summarizes their sources and scales; Appendices B.1 and B.2 define the capabilities and difficulty factors.

3.3 Data Construction

Environment construction. As shown in Figure 4, a Web agent collects application functions, workflows, and user needs for each level-2 domain, followed by human refinement. This knowledge informs the entities, attributes, state variables, and constraints of executable Python environments. Each environment includes query and state-changing tools with implementations, schemas, and initialization configurations; filesystem and coding environments additionally expose sandboxed file tools and a Python executor. Syntax checks, tool-call rollouts, and state-consistency validation filter out non-executable or unstable environments.

Task synthesis. Domain knowledge also grounds task goals and operational constraints. We construct four complementary routes. *DAG* tasks sample executable chains from tool dependency graphs

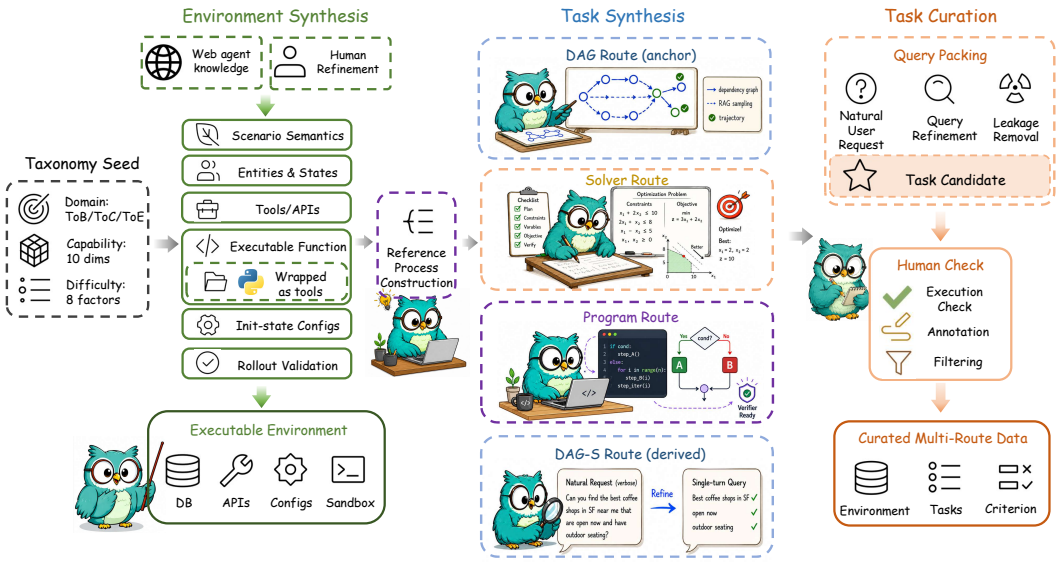


Figure 4: **Overview of the data construction pipeline.** Starting from the taxonomy, domain knowledge collected by a Web agent informs environment and task construction to produce four complementary data forms, subsequently curated through human annotation into the final benchmark.

to generate multi-turn requests, reference trajectories, and final states. *DAG-S* refines DAG requests into single-turn tasks. *Solver* uses implicit solver-guided or explicit solver-anchored synthesis for selection, scheduling, allocation, and optimization. *Program* targets branching and iterative procedures through task-program synthesis, execution debugging, and query refinement. Expanded states add candidates, distractors, cross-entity relations, and histories to challenge retrieval and aggregation. Execution, parameter repair, and trajectory–state consistency checks validate DAG tasks.

Evaluation signals and curation. DAG, DAG-S, and Solver tasks receive rubrics covering general completion quality and task-specific requirements, with checks on format, score allocation, and validity. Program tasks receive *VerifyCode* verifiers that determine success from trajectories and final observations. Human annotation and curation refine all artifacts for solvability, semantic consistency, evaluation validity, and reduced ambiguity.

Challenging set. From the full set of 1,431 tasks, we select 644 tasks to reduce evaluation cost and mitigate potential contamination after public release while preserving broad coverage. For DAG, we retain the task with the longest reference tool chain in each level-2 domain, yielding 354 tasks. After excluding low-complexity cases from the other routes, we retain 200 DAG-S, 60 Solver, and 30 Program tasks. Figure 3 summarizes benchmark characteristics; Appendix Table 8 details the challenging set's task, environment, and evaluation statistics, alongside full-set statistics in Appendix D.

3.4 Evaluation

Evaluation uses complete trajectories with route-specific success criteria. For DAG, DAG-S, and Solver, each rubric item receives its assigned score or zero, and success requires satisfying every item. Program tasks receive binary outcomes from *VerifyCode*. DAG uses persona-grounded user simulators whose preferences, communication styles, and disclosure strategies vary the interaction demands (Appendix F); the other routes use single-turn requests. Appendix E details these protocols. Each run produces an HTML report containing the dialogue, tool calls, environment feedback, final observation, scores, and failure reasons for case-level inspection.

4 Experiments

4.1 Experimental Setups

Models. We evaluate a collection of closed and open source models on OmniaBench. The closed source models include OpenAI's GPT-5.6-Sol, GPT-5.5 and GPT-5.4-Mini, Anthropic's Claude-

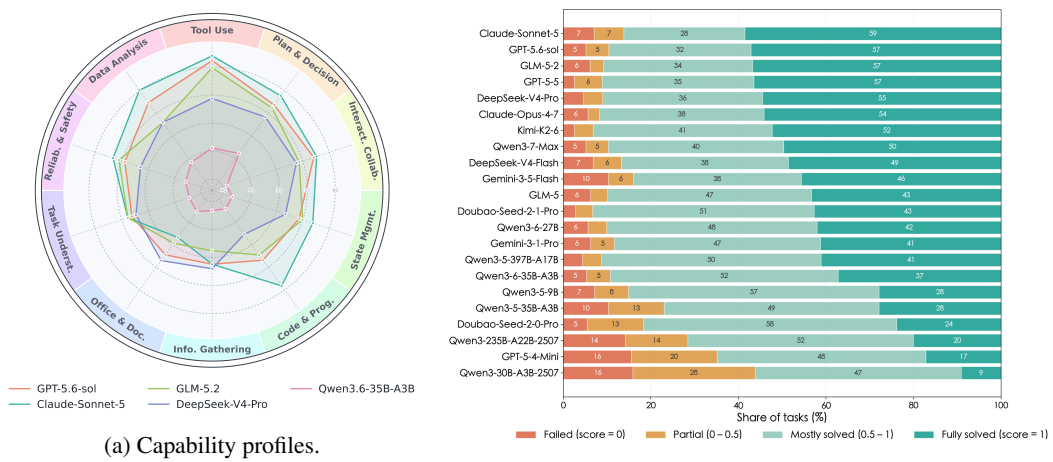
Table 2: **Benchmark results across different evaluation routes.** Pass@1 performance across four task construction routes: DAG, Solver, Program, and DAG-S. Overall denotes the Pass@1 score over all tasks from the four routes. The reasoning effort and other model parameters are described in Subsection 4.1.

Models	DAG		Solver	Program	DAG-S	ToB	ToC	ToE	ToolSteps	Overall
	Pass@1	UserTurns								
Closed-source Models										
Claude-Sonnet-5	**57.34**	9.97	56.67	63.33	60.50	**60.68**	**60.47**	48.11	64.16	**58.54**
GPT-5.6-sol	55.37	7.52	**65.00**	50.00	59.00	57.59	59.07	51.89	38.73	57.14
GPT-5.5	54.80	7.27	38.33	60.00	64.50	58.20	53.02	**58.49**	52.45	56.52
Claude-Opus-4.7	53.39	7.60	43.33	63.33	57.50	56.66	51.63	51.89	44.11	54.19
Qwen3.7-Max	48.59	6.49	51.67	**66.67**	48.50	48.30	48.84	55.66	47.96	49.69
Gemini-3.5-Flash	54.24	6.16	36.67	30.00	35.50	46.75	46.98	39.62	61.82	45.65
Doubao-Seed-2.1-Pro	54.80	6.85	25.00	36.67	27.50	43.03	40.93	45.28	44.83	42.70
Gemini-3.1-Pro	50.28	7.33	35.00	30.00	29.00	42.11	39.53	42.45	49.19	41.30
Doubao-Seed-2.0-Pro	33.90	8.14	21.67	36.67	5.00	22.60	25.12	25.47	29.76	23.91
GPT-5.4-mini	29.10	8.84	0.00	6.67	3.00	15.48	18.14	20.75	37.12	17.24
Open-source Models										
GLM-5.2	54.80	6.83	26.67	60.00	**69.00**	57.28	55.81	57.55	57.08	56.83
DeepSeek-V4-Pro	52.54	6.23	36.67	53.33	63.50	54.18	54.88	54.72	58.08	54.50
Kimi-K2.6	49.72	7.09	45.00	63.33	57.50	54.80	49.30	50.94	50.70	52.33
DeepSeek-V4-Flash	51.13	6.18	28.33	43.33	51.00	45.20	51.63	52.83	59.96	48.60
GLM-5	50.85	7.50	10.00	53.33	38.50	47.68	37.67	41.51	57.59	43.32
Qwen3.6-27B	48.02	7.66	28.33	46.67	35.00	43.65	38.60	44.34	46.62	42.08
Qwen3.5-397B-A17B	50.56	7.42	31.67	50.00	26.00	44.27	35.35	43.40	54.67	41.15
Qwen3.6-35B-A3B	43.22	7.46	20.00	36.67	32.00	41.80	31.63	34.91	53.28	37.27
Qwen3.5-35B-A3B	38.70	6.31	25.00	30.00	9.50	29.41	21.40	36.79	60.66	27.95
Qwen3.5-9B	41.53	7.06	16.67	23.33	8.00	30.96	20.93	33.02	66.19	27.95
Qwen3-235B-A22B	35.31	7.16	0.00	6.67	1.00	21.36	20.00	16.04	43.21	20.03
Qwen3-30B-A3B	16.67	9.13	0.00	0.00	0.00	9.91	6.51	12.26	29.63	9.16

Sonnet-5, Claude-Opus-4.7, Google's Gemini-3.5-Flash and Gemini-3.1-Pro-Preview, Qwen3.7-Max, Doubao-Seed-2.1-Pro-2026-06-28, and Doubao-Seed-2.0-Pro-2026-02-15. The open-source models include Kimi-K2.6, GLM-5.2, DeepSeek-V4-Pro, DeepSeek-V4-Flash, Qwen3.5-397B-A17B, Qwen3.6-27B, GLM-5, Qwen3.6-35B-A3B, Qwen3.5-35B-A3B, Qwen3.5-9B, Qwen3-235B-A22B-THINKING-2507, and Qwen3-30B-A3B-THINKING-2507.

Metrics. We use Pass@1 as the primary leaderboard metric and report task success rates on the DAG, Solver, Program, and DAG-S subsets. For rubric-based tasks, each task is associated with multiple rubric items weighted from 1 to 3 points. Each item receives either its full score or zero, and a task is considered successful only when all rubric items are satisfied. Program tasks are instead assigned binary outcomes directly by *VerifyCode*. The **Overall** score is computed as the Pass@1 over all four routes, obtained by dividing the total number of successfully completed tasks by the total number of tasks in the challenging set (644). The ToB, ToC, and ToE scores are computed by dividing the number of successfully completed tasks in each split by the total number of tasks in that split, with domain-level scores calculated analogously. **Capability** scores are computed as the average Pass@1 over tasks involving the corresponding capability. We additionally report **UserTurns**, the average number of user turns in DAG tasks, including the initial user request and both successful and unsuccessful trajectories, and **ToolSteps**, the average number of tool-call steps across all trajectories, including failed or prematurely terminated runs. We quantify user model robustness using Kendall's τ over model rankings and the average coefficient of variation (CV) of model scores across user simulators. Rubric-judge stability is evaluated separately using agreement-based statistics, as detailed in Appendix A.1. Separate experiments further report Pass@8 and Pass$\hat{}$8, measuring whether at least one or all eight independent runs succeed (Yao et al., 2024), respectively.

Implementation Details. We use Qwen3.5-397B-A17B for environment and task synthesis during data construction. For multi-turn evaluation, DeepSeek-V4-Pro with thinking disabled serves as both the user simulator and the rubric judge. The user simulator terminates an interaction by emitting the



(a) Capability profiles.

(b) Per-task score distribution.

Figure 5: **Capability and score analyses.** (a) Capability profiles of representative models across the ten capability dimensions. For each model and capability, the reported value is computed as the arithmetic mean of the model's normalized task scores over all tasks with the corresponding capability label. Details for capability dimensions are provided in Appendix B.1. (b) Distribution of normalized per-task scores for all models on the leaderboard.

special signal `###STOP###`, whereas single-turn tasks terminate when the evaluated model submits its final response. Temperature is set to 0 whenever supported; for the Claude and Gemini series, for which the official guidance recommends retaining the default setting, we use the default temperature of 1.0. GPT models use `high` reasoning effort, Claude series use the `max` reasoning budget, other models use their max effort mode if exist, and explicit thinking is enabled for all other models that support it. Each trajectory permits at most 200 tool-call steps, rather than 200 model-output turns. We allow up to 20 cumulative retries for unexpected environment-side tool exceptions; API-level failures are handled separately and do not count toward this limit. All tools follow the OpenAI function-calling schema, with consistent tool definitions across models.

4.2 Main Results

OmniaBench remains challenging for current frontier models. As reported in Table 2, even the frontier models Claude-Sonnet-5 and GPT-5.6-Sol, attains an Overall Pass@1 of only 58.54% and 57.14%, respectively, followed closely by GLM-5.2 and GPT-5.5 at 56.83% and 56.52%, respectively. The substantial performance spread across the remaining models further demonstrates the discriminative power of OmniaBench across different model families and capability levels. Figure 5b shows that, even among the leading models, only approximately half of the tasks are fully solved, while a considerable proportion are either partially completed or failed, indicating that models frequently make meaningful progress without satisfying all task requirements. Consistently, the ToolSteps statistics in Table 2 and Figure 8b show that evaluation trajectories often span dozens of tool calls and extended action sequences. These results highlight the sustained planning, information aggregation, state tracking, and cross-tool coordination required by OmniaBench.

Models exhibit substantial performance variation across both capability and domains. As shown in Figure 5a, performance across the ten capability dimensions is highly non-uniform: different models develop distinct strengths in Task Understanding, Information Gathering, Planning & Decision Making, State Management, and Tool Use. Consequently, similar overall scores may correspond to markedly different capability profiles. Clear variation is also observed across domains. Figure 6 shows that Claude-Sonnet-5, GPT-5.6-Sol, and GLM-5.2 achieve ToB scores of 60.68%, 57.59%, and 57.28%, respectively, substantially exceeding some of their results on the other domain splits, whereas GLM-5.2 attains its strongest split-level performance on ToE at 57.55%. Thus, even models with comparable Overall scores may exhibit substantially different relative strengths across ToB, ToC, and ToE, suggesting that a single aggregate metric cannot fully characterize their practical applicability. Figure 9 further corroborates this observation at the level-1 domain granularity, revealing additional shifts in model rankings across specific application scenarios.

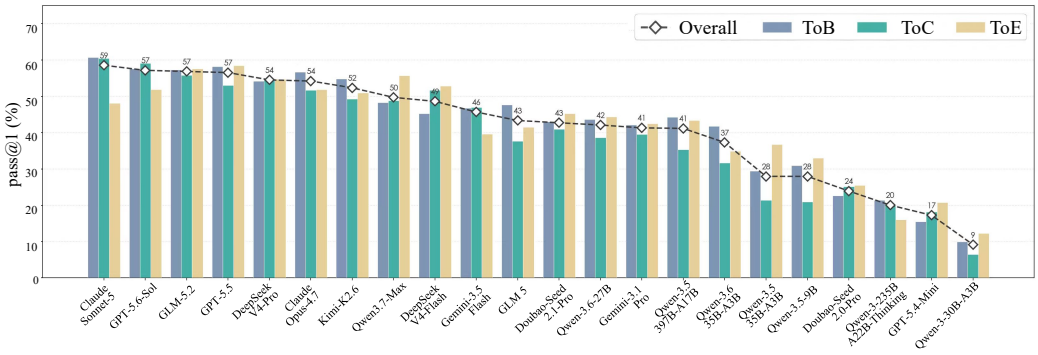


Figure 6: **Performance across domain splits.** Pass@1 results across ToB, ToC, and ToE splits for all evaluated models, with diamonds and the dashed line denoting overall performance. Models are ordered by their overall Pass@1 scores. The figure further illustrates how performance varies across domain splits for models at different overall capability levels.

Comparable overall scores can mask large differences in tool-use efficiency. As shown in Figure 8a, models with similar overall success rates can differ substantially in the length and structure of their execution trajectories. Some models complete tasks through compact, well-targeted sequences of tool calls, whereas others require markedly longer and more circuitous interactions to reach comparable outcomes. Such longer trajectories often reflect redundant exploration, repeated or ineffective calls, unnecessary replanning, or recovery from execution errors rather than more thorough task completion. More efficient models identify the required information and operations more precisely, avoid unnecessary interactions, and satisfy task requirements with fewer steps. OmniaBench therefore captures not only whether a task is completed, but also how efficiently tools are planned and orchestrated.

4.3 Discussion

Reliability of User Simulation and Repeated Exploration. As shown in Figure 8c, replacing the user simulator in multi-turn interactions with GPT-4.1, DeepSeek-V4-Flash, or DeepSeek-V4-Pro leaves the relative ranking of the three evaluated models unchanged, yielding a Kendall's τ of 1.0 and an average coefficient of variation of 7.24% across user models. Although different user simulators induce moderate shifts in absolute scores, the principal comparative conclusions remain stable. Figure 7 further shows that Pass@k increases with the number of independent runs, indicating that restarting from scratch provides additional opportunities to solve difficult tasks. At the same time, Pass^k declines rapidly, suggesting that such successes are not yet consistently reproducible across repeated executions. Repeated exploration therefore improves task coverage, but incurs additional execution cost and highlights the remaining gap toward reliable and efficient completion of complex tasks. The Pass@1 standard deviations of DeepSeek-V4-Flash and Qwen3.6-27B are only 1.07 and 2.68 percentage points, respectively, indicating that the aggregate results remain reproducible. We also conduct an independent stability analysis of the rubric judge and obtain consistent findings, as detailed in Appendix A.1.

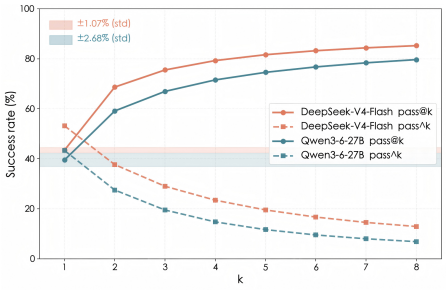
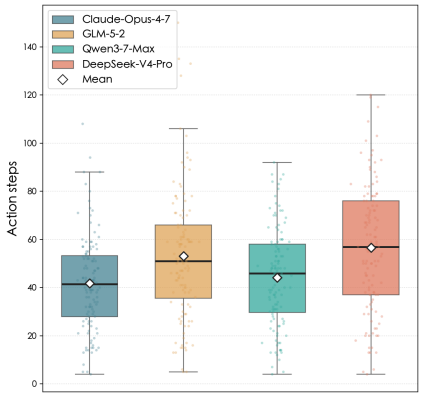
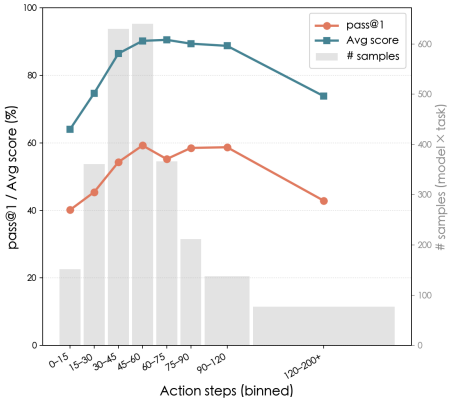


Figure 7: **Repeated-run performance.** Pass@k and Pass^k across two models.

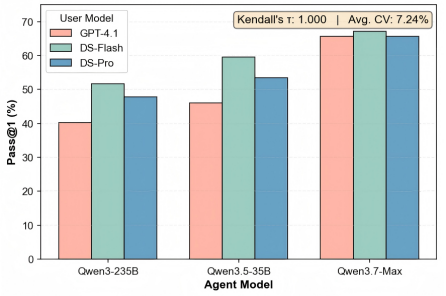
Distribution of Error Modes. As shown in Figure 8d, reasoning-related failures account for 53.8% of all observed errors, with planning and decomposition contributing 36.7% and constraint violations a further 16.5%. Meta-cognitive errors constitute another 31.0%, primarily arising from insufficient reflection and premature abandonment. These failures often emerge after several valid intermediate actions, suggesting that models struggle more with maintaining coherent long-horizon strategies than with individual tool calls. By comparison, direct tool-use errors, such as formatting mistakes,



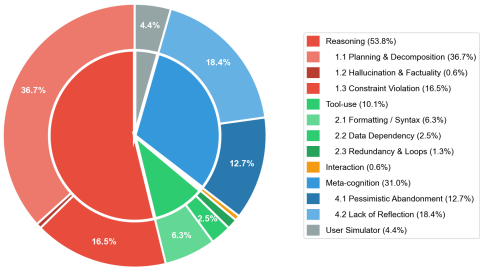
(a) Step efficiency on the commonly-solved set.



(b) Outcome vs. effort across action-step bins.



(c) Robustness across user models.



(d) Distribution of agent errors.

Figure 8: **Analysis of efficiency, task complexity, robustness, and errors.** (a) Distribution of action steps per model on the commonly-solved set. (b) Pass@1 and average score across action-step bins, with sample counts. (c) Pass@1 under different user simulators, showing stable model rankings. (d) Distribution of error categories observed during Claude-Opus-4.7 execution.

data-dependency failures, and redundant calls, account for a substantially smaller proportion. This distribution indicates that the main bottlenecks of current general-agent models lie less in basic tool invocation than in long-horizon planning, constraint maintenance, and adaptive correction based on execution feedback across extended and dynamically evolving trajectories.

Evaluation Priorities for Stronger General Agents. As overall task success continues to improve, a single aggregate score will become increasingly insufficient for characterizing differences among advanced models. Our results show that models with similar overall performance may still exhibit substantial variation across capability dimensions, domains, tool-use efficiency, and failure modes. Future evaluation should therefore place greater emphasis on structured analyses of specific abilities and domains, such as long-horizon planning, information integration, constraint adherence, and error recovery, rather than relying solely on task success. The domain, capability, and error-oriented analyses provided by OmniaBench offer a basis for systematically identifying finer-grained capability boundaries among increasingly strong general agents in realistic evaluation settings.

5 Conclusion

We present OmniaBench, a general agent benchmark covering diverse application scenarios. OmniaBench builds executable environments from a hierarchical ToB, ToC, and ToE taxonomy and constructs tasks through multiple interaction routes. Experiments on frontier models demonstrate that OmniaBench remains challenging while revealing distinct capability profiles, domain specialization, tool-use efficiency, and failure patterns. OmniaBench provides a systematic foundation for evaluating and improving general agents.

AI USE STATEMENT

Generative AI tools were used as part of the benchmark construction pipeline. In particular, Qwen3.5-397B-A17B was used to generate candidate environment and task specifications, as described in Section 3.3. AI-generated artifacts were subjected to syntax checking, executable rollouts, state-consistency validation, and human annotation and curation. The authors reviewed the resulting benchmark artifacts and manuscript and take responsibility for the final content, claims, code, and data released with this work.

ETHICS STATEMENT

OmniaBench evaluates agents in simulated, controlled environments and does not require agents to act on live user accounts or production systems. Nevertheless, the taxonomy and generated tasks may inherit biases or omissions from public application resources and generative models. Persona-grounded interactions may also imperfectly represent real users. We mitigate these risks through human review, executable validation, controlled tool interfaces, and explicit safety and reliability criteria. Benchmark results should be interpreted as evidence about performance in the included scenarios, not as a guarantee that a model is safe or effective in unrestricted deployment.

REPRODUCIBILITY STATEMENT

The benchmark construction process, task routes, evaluation metrics, model settings, and execution limits are described in Sections 3.3 and 4.1. Route-specific evaluation protocols, persona design, full-set statistics, and additional stability analyses are provided in the appendix. The accompanying environments, task specifications, rubrics, programmatic verifiers, and evaluation code provide the artifacts needed to reproduce the reported evaluation protocol.

REFERENCES

Axel Backlund and Lukas Petersson. Vending-bench: A benchmark for long-term coherence of autonomous agents. *arXiv preprint arXiv:2502.15840*, 2025.

Chaithanya Bandi, Razvan-Gabriel Dumitru, Ben Hertzberg, Divyansh Agarwal, Geobio Boo, Tejas Polakam, Sami Hassaan, Jeff Da, HiJae Kim, Vipul Gupta, et al. Mcp-atlas: A large-scale benchmark for tool-use competency with real mcp servers. *arXiv preprint arXiv:2602.00933*, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. *arXiv preprint arXiv:2506.07982*, 2025.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault L De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *Advances in Neural Information Processing Systems*, 37:5996–6051, 2024.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, et al. Acebench: Who wins the match point in tool usage? *arXiv preprint arXiv:2501.12851*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 9, 2021.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, Dongjie Ji, Erhang Li, Fang Wei, Fangyun Lin, Fangzhou Yuan, Feiyu Xia, Fucong Dai, Guangbo Hao, Guanting Chen, Guoai Cao, Guolai Meng, Guowei Li, Han Yu, Han Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoling Zhang, Haoming Luo, Haoran Wei, Haotian Yuan, Haowei Zhang, Haowen Luo, Haoyu Chen, Haozhe Ji, Hengqing Zhang, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, J Yang, JQ Zhu, Jia Luo, Jia Song, Jia Yu, Jialiang Huang, Jialu Cai, Jian Liang, Jiangting Zhou, Jiasheng Ye, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jieyu Yang, Jin Chen, Jin Yan, Jingchang Chen, Jingli Zhou, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jingzi Zhou, Jinhua Zhu, Jiping Yu, Joseph Sun, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junmin Zheng, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Leyi Xia, Li Zhang, Liang Zhao, Lihua Guo, Lingxiao Luo, Linwang Ma, Linyan Zhu, Litong Wang, Liyu Cai, Liyue Zhang, Longhao Chen, MS Di, MY Xu, Max Mei, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Mingxu Zhou, Minmin Han, Ning Wang, Panpan Huang, Panpan Wang, Peixin Cong, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Qiwei Jiang, Rui Tian, Ruifan Xu, Ruijie Lu, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqian Chen, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, Ruyi Chen, SH Liu, Shanghao Lu, Shangmian Sun, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoheng Nie, Shaoqing Wu, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Shuying Yu, Songyang Zhou, Tao Ni, Tao Yun, Tian Jin, Tian Pei, Tian Ye, Tianle Lin, Tianran Ji, Tianyi Cui, Tianyuan Yue, Tingting Yu, Tun Wang, W Zhang, WL Xiao, Wangding Zeng, Wei An, Weilin Zhao, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjing Yao, Wenjun Gao, Wenkai Yang, Wenlve Huang, Wenqing Hou, Wentao Zhang, Wenting Ma, Xi Gao, Xiang He, Xiangwen Wang, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingchen Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyu Zhang, Xu Chen, Xuanyu Wang, Xuecheng Su, Xueyin Chen, Xuheng Lin, Xuwei Fu, YC Yan, YQ Wang, YW Ma, Yanfeng Luo, Yang Zhang, Yanhong Xu, Yanru Ma, Yanwen Huang, Yao Li, Yao Li, Yao Xu, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Shao, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yijia Wu, Yiliang Xiong, Yiling Ma, Ying He, Ying Tang, Ying Zhou, Yingjia Luo, Yinmin Zhong, Yishi Piao, Yisong Wang,

Yixiang Zhang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, YuKun Li, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuanhao Li, Yuduan Wang, Yuehan Yang, Yuer Xu, Yuhan Wu, Yuhao Meng, Yuheng Zou, Yukun Zha, Yunfan Xiong, Yupeng Chen, Yuping Lin, Yuqian Cao, Yuqian Wang, Yushun Zhang, Yuting Yan, Yutong Lin, Yuxian Gu, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuxuan Zhou, Yuyang Zhou, Yuzhen Huang, ZF Wu, Zehao Wang, Zehua Zhao, Zehui Ren, Zekai Zhang, Zhangli Sha, Zhe Fu, Zhe Ju, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zheren Gao, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhixuan Chen, Zhiyu Wu, Zhizhou Ren, Zhongyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihua Qu, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Ziyi Wan, Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, et al. Gaia2: Benchmarking llm agents on dynamic and asynchronous environments. *arXiv preprint arXiv:2602.11964*, 2026.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, et al. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications. *arXiv preprint arXiv:2509.26490*, 2025.

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

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. *arXiv preprint arXiv:2510.25726*, 2025.

Hao Liang, Xiaochen Ma, Zhou Liu, Zhen Hao Wong, Zhengyang Zhao, Zimo Meng, Runming He, Chengyu Shen, Qifeng Cai, Zhaoyang Han, Meiyi Qiang, Yalin Feng, Tianyi Bai, Zewei Pan, Ziyi Guo, Yizhen Jiang, Jingwen Deng, Qijie You, Peichao Lai, Tianyu Guo, Chi Hsu Tsai, Hengyi Feng, Rui Hu, Wenkai Yu, Junbo Niu, Bohan Zeng, Ruichuan An, Lu Ma, Jihao Huang, Yaowei Zheng, Conghui He, Linpeng Tang, Bin Cui, Weinan E, and Wentao Zhang. Dataflow: An llm-driven framework for unified data preparation and workflow automation in the era of data-centric ai, 2025. URL https://arxiv.org/abs/2512.16676.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Fanqing Meng, Lingxiao Du, Zijian Wu, Guanzheng Chen, Xiangyan Liu, Jiaqi Liao, Chonghe Jiang, Zhenglin Wan, Jiawei Gu, Pengfei Zhou, et al. Clawmark: A living-world benchmark for multi-turn, multi-day, multimodal coworker agents. *arXiv preprint arXiv:2604.23781*, 2026.

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

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, et al. Gdpval: Evaluating ai model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025.

Qwen Team and Alibaba Data. QwenClawBench: Real-user-distribution benchmark for openclaw agents, April 2026. URL github.com/SKYLENAGE-AI/QwenClawBench.

Chengyu Shen, Yanheng Hou, Minghui Pan, Runming He, Zhen Hao Wong, Meiyi Qiang, Zhou Liu, Hao Liang, Peichao Lai, Zeang Sheng, et al. One-eval: An agentic system for automated and traceable llm evaluation. *arXiv preprint arXiv:2603.09821*, 2026.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. τ-Knowledge: Evaluating Conversational Agents over Unstructured Knowledge, 2026. URL https://arxiv.org/abs/2603.04370.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu

Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

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

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*, 2024.

Bowen Ye, Rang Li, Qibin Yang, Yuanxin Liu, Linli Yao, Hanglong Lv, Zhihui Xie, Chenxin An, Lei Li, Lingpeng Kong, et al. Claw-eval: Towards trustworthy evaluation of autonomous agents. *arXiv preprint arXiv:2604.06132*, 2026.

Yinger Zhang, Shutong Jiang, Renhao Li, Jianhong Tu, Yang Su, Lianghao Deng, Xudong Guo, Chenxu Lv, and Junyang Lin. Deepplanning: Benchmarking long-horizon agentic planning with verifiable constraints. *arXiv preprint arXiv:2601.18137*, 2026.

A RUBRIC-BASED JUDGMENT ROBUSTNESS

A.1 JUDGE MODEL ROBUSTNESS RESULTS

Since our benchmark relies on LLM-based judges to evaluate agent trajectories, we examine whether the resulting evaluation scores are overly sensitive to the choice of judge model. We re-evaluate the same set of trajectories using five different judge models under an identical rubric and judging protocol: `deepseek-v4-flash`, `gpt-4.1`, `deepseek-v4-pro`, `glm-5-thinking`, and `claude-opus-4-7-thinking`.

We evaluate judge-model robustness at both the trajectory and rubric-item levels. At the trajectory level, we measure absolute score agreement, ranking consistency, and normalized score deviation. At the rubric-item level, we examine agreement on point-valued item scores as well as agreement on normalized binary pass/fail decisions. Detailed metric definitions are provided in Appendix A.2.

Table 3: Robustness of benchmark scores under judge-model substitution.

Level	#Units	#Judges	ICC(A,1)	Mean Spearman	Min Spearman	NMPD	P95 NMPD
Total Score	1102	5	0.780	0.821	0.756	0.062	0.266
Weighted Item Score	9808	5	0.741	0.856	0.811	0.007	0.060

Note. Total Score evaluates trajectory-level score robustness. Weighted Item Score evaluates agreement on the original point-valued rubric-item scores. For Weighted Item Score, ICC and Spearman are computed from the raw item scores, while NMPD is normalized by the maximum total score of the corresponding trajectory to quantify the local contribution of item-level judge disagreement. #Units denotes evaluation units for which all five judges provide valid scores. NMPD denotes normalized mean pairwise deviation.

Results. As shown in Table 3, the trajectory-level scores are relatively stable across the evaluated judge models. The total-score $\mathrm{ICC}(A,1)$ is 0.780, showing that the judge models assign broadly similar absolute scores to the same trajectories. The mean pairwise Spearman correlation is 0.821, and the minimum pairwise correlation is 0.756, indicating that the relative ranking of trajectories is largely preserved across judge models.

The total-score NMPD is 0.062. In other words, the average absolute score difference between a pair of judge models corresponds to 6.2% of the maximum score of the corresponding trajectory. The P95 NMPD is 0.266, indicating that 95% of trajectories have an average pairwise judge difference no greater than 26.6% of their maximum score.

Table 4: Agreement statistics for rubric-item pass/fail decisions.

Level	#Items	#Judges	Fleiss' κ	Full Agree	One-off	Split
Item Decision	9808	5	0.573	86.3%	9.3%	4.4%

Note. Item scores are normalized by their point values into binary pass/fail decisions. Full Agree denotes unanimous agreement among all judges. One-off denotes a 4-vs-1 disagreement. Split denotes a near-even 3-vs-2 disagreement.

At the rubric-item decision level, Table 4 shows that 86.3% of trajectory-item units receive unanimous pass/fail decisions from all five judges. A further 9.3% exhibit a one-off 4-vs-1 disagreement, while only 4.4% exhibit a near-even 3-vs-2 split.

Fleiss' κ is 0.573, indicating moderate chance-corrected agreement among the evaluated judge models. Together, these results show that most rubric items elicit consistent binary decisions across the five judge models under the fixed rubric and judging protocol. However, this agreement does not by itself establish the correctness of the judgments or rule out systematic biases shared by multiple judge models.

The point-valued rubric-item scores also exhibit substantial cross-judge consistency. The weighted-item-score $\mathrm{ICC}(A,1)$ is 0.741, while the mean and minimum pairwise Spearman correlations are 0.856 and 0.811, respectively. These results indicate that the judges largely agree on both the absolute values and relative ordering of point-valued item scores.

The weighted-item-score NMPD is 0.007. This means that, for an average trajectory-item unit, the absolute score difference between a pair of judge models corresponds to approximately 0.7% of the maximum total score of the associated trajectory. The P95 NMPD is 0.060, indicating that the corresponding local item-level difference is no greater than 6.0% of the trajectory maximum score for 95% of trajectory-item units.

This statistic measures the average local effect of disagreement on a single rubric item. It should not be interpreted as the cumulative effect of all item-level disagreements on a trajectory's final score; cumulative score variation is directly captured by the trajectory-level Total Score NMPD.

Overall, the results provide evidence that aggregate benchmark scores are reasonably stable across the five evaluated judge models under the fixed rubric and judging protocol. The benchmark exhibits relatively high trajectory-level score agreement and ranking consistency, while binary item-level decisions show moderate chance-corrected agreement.

Agreement with human judgments. In addition to cross-judge robustness, we manually review a subset of rubric-based evaluations using the same task evidence and rubric criteria provided to the automatic judge. Human evaluators independently determine whether each rubric requirement is satisfied and compare their decisions with the automatic judgments. The agreement between human and automatic judgments exceeds 99%, providing complementary evidence that the rubric-based evaluation closely aligns with human assessment.

A.2 Judge Model Robustness Metrics

This appendix provides the formal definitions of the metrics reported in Tables 3 and 4.

Notation. Let N denote the number of evaluated trajectories and M denote the number of judge models. For trajectory $i \in \{1, \dots, N\}$ and judge model $j \in \{1, \dots, M\}$, let s_{ij} denote the total score assigned by judge j, and let P_i denote the maximum attainable total score of trajectory i.

Each trajectory i contains L_i rubric items. For item $\ell \in \{1, \dots, L_i\}$, let $z_{i\ell j}$ denote the raw item score assigned by judge j, and let $p_{i\ell}$ denote the maximum point value of that item.

Each rubric item follows an all-or-nothing scoring rule: a judge assigns either zero points or the full point value of the item, and partial credit is not allowed. Therefore,

$$z_{i\ell j} \in \{0, p_{i\ell}\}. \tag{1}$$

To remove differences in item point values, we normalize each item score into a binary pass/fail decision:

$$d_{i\ell j} = \frac{z_{i\ell j}}{p_{i\ell}} \in \{0, 1\}, \tag{2}$$

where $d_{i\ell j} = 1$ indicates that judge j considers item ℓ satisfied, and $d_{i\ell j} = 0$ indicates that the item is not satisfied.

Unless otherwise specified, all metrics are computed using complete-case evaluation units for which all M judge models provide valid scores.

Generic evaluation matrix. For each score-level analysis, we construct a score matrix

$$X \in \mathbb{R}^{n \times M}, \tag{3}$$

where rows correspond to evaluation units and columns correspond to judge models. Let x_{ij} denote the score assigned to evaluation unit i by judge model j.

For trajectory-level analysis, each evaluation unit is one trajectory. For rubric-item score analysis, each evaluation unit is one trajectory-item pair.

Absolute-agreement ICC. We report the absolute-agreement intraclass correlation coefficient, $\mathrm{ICC}(A, 1)$, to measure whether different judge models assign similar absolute scores to the same evaluation units.

Following the two-way random-effects, single-measure, absolute-agreement formulation, we compute:

$$\mathrm{ICC}(A,1) = \frac{MS_R - MS_E}{MS_R + (M-1)MS_E + \frac{M(MS_C - MS_E)}{n}}, \tag{4}$$

where MS_R, MS_C, and MS_E denote the ANOVA mean squares associated with evaluation units, judge models, and residual error, respectively. A higher $\mathrm{ICC}(A,1)$ indicates stronger absolute agreement across judge models.

Pairwise Spearman correlation. To measure ranking consistency, we compute the Spearman rank correlation for every pair of judge models (a, b):

$$\rho_{ab} = \mathrm{corr}\left(\mathrm{rank}(x_{\cdot a}), \mathrm{rank}(x_{\cdot b})\right). \tag{5}$$

We report both the mean and minimum Spearman correlation across all $\binom{M}{2}$ judge pairs:

$$\mathrm{MeanSpearman} = \frac{1}{\binom{M}{2}} \sum_{1 \le a < b \le M} \rho_{ab}, \tag{6}$$

and

$$\mathrm{MinSpearman} = \min_{1 \le a < b \le M} \rho_{ab}. \tag{7}$$

Mean Spearman reflects average ranking consistency across judge models, whereas Min Spearman captures the weakest-performing judge pair.

Normalized mean pairwise deviation. To quantify the magnitude of score variation across judge models, we compute the normalized mean pairwise deviation.

For each evaluation unit i, define:

$$\Delta_i = \frac{\frac{1}{\binom{M}{2}} \sum_{1 \le a < b \le M} |x_{ia} - x_{ib}|}{q_i}, \tag{8}$$

where q_i is an analysis-specific normalization denominator.

The normalized mean pairwise deviation is:

$$\mathrm{NMPD} = \frac{1}{n} \sum_{i=1}^{n} \Delta_i. \tag{9}$$

We additionally report the empirical 95th percentile of the unit-level normalized deviations:

$$\text{P95 NMPD} = Q_{0.95}\left(\{\Delta_i\}_{i=1}^{n}\right), \tag{10}$$

where $Q_{0.95}$ denotes the empirical 95th-percentile operator.

NMPD measures the average normalized pairwise score difference, whereas P95 NMPD characterizes instability in the upper tail of the evaluation units.

Metric instantiations. We instantiate the generic score matrix for two score-level analyses: trajectory-level total scores and point-valued rubric-item scores. Binary rubric-item decisions are analyzed separately using Fleiss' κ and raw agreement-pattern statistics.

TRAJECTORY-LEVEL TOTAL SCORE. For trajectory-level total scores, each evaluation unit corresponds to one trajectory, and we define:

$$x_{ij} = s_{ij}, \qquad q_i = P_i. \tag{11}$$

The corresponding unit-level normalized deviation is:

$$\Delta_i^{\mathrm{total}} = \frac{\frac{1}{\binom{M}{2}} \sum_{1 \le a < b \le M} |s_{ia} - s_{ib}|}{P_i}. \tag{12}$$

Thus, total-score NMPD measures the average pairwise judge difference in trajectory score, normalized by the maximum attainable score of that trajectory.

POINT-VALUED RUBRIC-ITEM SCORE. For point-valued rubric-item scores, each evaluation unit corresponds to a trajectory-item pair (i, ℓ), and we define:

$$x_{i\ell j} = z_{i\ell j}, \qquad q_{i\ell} = P_i. \tag{13}$$

ICC and Spearman correlation are computed directly from the raw point-valued rubric-item scores $z_{i\ell j}$. These scores jointly reflect the binary pass/fail decisions made by the judges and the predefined point values of the rubric items.

Let

$$\mathcal{U} = \{(i, \ell) : \text{all } M \text{ judges provide a valid score for item } \ell \text{ of trajectory } i\} \tag{14}$$

denote the complete-case set of valid trajectory-item units, and let

$$U = |\mathcal{U}| \tag{15}$$

denote its cardinality.

The item-score NMPD is:

$$\mathrm{NMPD}_{\mathrm{item}} = \frac{1}{U} \sum_{(i,\ell)\in\mathcal{U}} \frac{\frac{1}{\binom{M}{2}} \sum_{1\le a<b\le M} |z_{i\ell a} - z_{i\ell b}|}{P_i}. \tag{16}$$

The corresponding P95 NMPD is the empirical 95th percentile of:

$$\Delta_{i\ell}^{\mathrm{item}} = \frac{\frac{1}{\binom{M}{2}} \sum_{1\le a<b\le M} |z_{i\ell a} - z_{i\ell b}|}{P_i}, \qquad (i, \ell) \in \mathcal{U}. \tag{17}$$

This metric is a micro-average over trajectory-item units. Consequently, a trajectory containing more rubric items contributes more units to the metric. It measures the average local contribution of pairwise judge disagreement on a single rubric-item score relative to the maximum total score of the associated trajectory. It does not measure the cumulative effect of all item-level disagreements within an entire trajectory.

Fleiss' κ for binary item decisions. For normalized binary item decisions, we report Fleiss' κ to measure multi-judge agreement after correcting for agreement expected by chance.

Each complete-case trajectory-item pair is treated as one evaluation unit $u \in \{1, \ldots, U\}$. Let $n_{u,1}$ denote the number of judges assigning a pass decision and let $n_{u,0}$ denote the number assigning a fail decision. By definition,

$$n_{u,0} + n_{u,1} = M. \tag{18}$$

The observed agreement for unit u is:

$$P_u = \frac{1}{M(M-1)} \sum_{c\in\{0,1\}} n_{u,c}\big(n_{u,c} - 1\big). \tag{19}$$

The mean observed agreement across all item-decision units is:

$$\bar{P} = \frac{1}{U} \sum_{u=1}^{U} P_u. \tag{20}$$

The empirical marginal proportion of category $c \in \{0, 1\}$ is:

$$p_c = \frac{1}{UM} \sum_{u=1}^{U} n_{u,c}. \tag{21}$$

The expected agreement under the empirical marginal category distribution is:

$$\bar{P}_e = \sum_{c\in\{0,1\}} p_c^2. \tag{22}$$

Fleiss' κ is then defined as:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}. \tag{23}$$

A higher value of κ indicates stronger chance-corrected agreement among the judge models.

Table 5: Overview of the taxonomy structure.

View	Type	Source / Basis	Scale
Domain	ToC	Multiple app stores and consumer-facing app categories	22 L1 / 101 L2
	ToB	GDPval-style (Patwardhan et al., 2025) tasks and industry classifications	38 L1 / 186 L2
	ToE	Industry-general employee tasks from GDPval and industry templates	30 L1 / 67 L2
	Total	Normalized real-world scenario taxonomy	90 L1 / 354 L2
Capability	Dims.	General-agent execution requirements	10 dims.
Atomic Difficulty	Factors	Eight challenge factors across user, environment, tool use, and interaction	8 factors

Agreement patterns. In addition to chance-corrected agreement, we report three raw agreement-pattern statistics.

For each item-decision unit u, define:

$$r_u = \max(n_{u,0}, n_{u,1}), \tag{24}$$

where r_u is the number of judges supporting the majority decision.

The full-agreement ratio is:

$$\text{FullAgree} = \frac{1}{U}\sum_{u=1}^{U} \mathbf{1}[r_u = M]. \tag{25}$$

This measures the fraction of item-decision units for which all judge models produce the same binary decision.

The one-off disagreement ratio is:

$$\text{OneOff} = \frac{1}{U}\sum_{u=1}^{U} \mathbf{1}[r_u = M - 1]. \tag{26}$$

For $M = 5$, this corresponds to a 4-vs-1 disagreement.

The split ratio is:

$$\text{Split} = \frac{1}{U}\sum_{u=1}^{U} \mathbf{1}\left[\left\lceil \frac{M}{2} \right\rceil \le r_u \le M - 2\right]. \tag{27}$$

For $M = 5$, this corresponds to a 3-vs-2 split. Such cases represent the strongest possible disagreement pattern among an odd number of five judges and may indicate more ambiguous or difficult rubric-item decisions.

B Taxonomy Details

Table 5 summarizes the three taxonomy views used in Section 3.2. The domain taxonomy combines consumer app categories, GDPval-style tasks, industry classifications, and employee-task templates into 90 level-1 (L1) and 354 level-2 (L2) domains. ToC contributes 22 L1 and 101 L2 domains, ToB contributes 38 and 186, and ToE contributes 30 and 67. The capability and atomic difficulty views complement this application coverage with ten ability dimensions and eight compositional challenge factors, defined below.

Scenario collection and curation. The domain taxonomy is assembled from multiple application-oriented sources rather than a single predefined ontology. ToC scenarios are primarily informed by popular app stores and consumer-facing application categories; ToB scenarios incorporate industry classifications, professional tasks, and publicly available business resources; and ToE scenarios further cover general employee-oriented workflows. Web retrieval supplements these sources with

application functions, representative workflows, and user needs within individual domains. Product practitioners and domain experts further review the collected scenarios and task contexts, considering whether the user needs, operational processes, involved entities, and available tools form a coherent application setting. Candidates with weak scenario semantics, mismatched domain assignments, or implausible task contexts are revised or removed. OmniaBench is intended to provide broad and diverse scenario coverage rather than an exhaustive enumeration of real-world agent applications.

B.1 Capability Dimensions

The capability taxonomy provides a unified diagnostic view of the behaviors required across the heterogeneous domains and data routes in OmniaBench. As described in Section 3.3, the taxonomy is used to guide scenario selection, environment construction, task synthesis, and subsequent evaluation design. Rather than treating tool interaction as a single undifferentiated ability, we decompose general-agent competence into ten dimensions covering task interpretation, information acquisition, planning, execution, state tracking, artifact manipulation, interaction, and reliability.

A task may involve multiple capability dimensions simultaneously because realistic agent tasks often require several interdependent behaviors. We therefore associate each task with all applicable capability labels. For capability-level aggregation, a task contributes to every capability dimension with which it is annotated. The complete definitions of the ten capability dimensions are provided in Table 6.

Accordingly, a capability score characterizes performance on tasks involving the corresponding capability, rather than an isolated measurement of that capability alone.

Table 6: **Capability dimensions.** Each task may be annotated with multiple capability dimensions and contributes to the aggregation of every dimension with which it is associated.

Capability	**What it evaluates**
Task Understanding	Identifying user goals, implicit requirements, priorities, domain constraints, and expected outcomes.
Information Gathering	Locating, retrieving, and integrating relevant evidence from environment states, tools, files, databases, and external information sources.
Planning & Decision Making	Decomposing goals, selecting execution strategies, respecting dependencies and constraints, and revising plans as new observations become available.
State Management	Tracking intermediate progress and maintaining consistency across entities, environment states, and long or multi-turn trajectories.
Tool Use	Selecting appropriate tools, constructing valid arguments, interpreting outputs, and coordinating multiple tool calls.
Code & Programmatic Operations	Writing and executing code for computation, data transformation, automation, file manipulation, and programmatic task completion.
Data Analysis	Filtering, aggregating, comparing, reconciling, and reasoning over structured or semi-structured data.
Office & Document Handling	Reading, extracting, editing, merging, validating, and producing documents, spreadsheets, presentations, and other file-based artifacts.
Interactive Collaboration	Requesting missing information, clarifying ambiguous goals, confirming actions, incorporating user feedback, and coordinating across interaction turns.
Reliability & Safety	Detecting and recovering from failures, maintaining constraint compliance, avoiding unsafe or invalid actions, and completing tasks robustly under uncertainty.

B.2 Atomic Difficulty Factors

In addition to capability coverage, we characterize task complexity using a set of atomic difficulty factors. Capability dimensions describe *what* an agent is required to do, whereas difficulty factors describe *why* the corresponding task is challenging. This separation allows tasks that evaluate the same primary capability to instantiate different sources of difficulty, such as ambiguous re-

quirements, large structured states, long evidence contexts, dynamic dependencies, or inconsistent information.

The difficulty atoms are used as compositional design primitives during environment and task construction. A single task may combine multiple atoms, and the same atom may appear across different domains and data routes. For example, a DAG task may combine progressive information disclosure, state evolution, and dynamic multi-step planning, while a Solver task may combine structured-information complexity with conflicting constraints. These atoms are intended to support controlled task diversification and fine-grained error analysis rather than define mutually exclusive task categories. Table 7 summarizes the eight atomic difficulty factors and their typical instantiations.

Table 7: **Atomic difficulty dimensions.** Multiple atoms may be composed within a single task.

Difficulty Atom	Instantiation
Ambiguous Goal and Contextual Constraints	The request is underspecified, indirect, or conditioned on implicit preferences, policies, priorities, or professional constraints.
Tool and Parameter Grounding	The agent must distinguish similar or redundant tools, infer arguments from context, or request missing parameters before execution.
Structured-Information Complexity	The environment contains numerous structured entities, attributes, relations, or records that must be filtered, joined, compared, or reconciled.
Long-Context and Multi-Artifact Evidence	Relevant evidence is distributed across long tool outputs, documents, files, attachments, logs, or multiple heterogeneous artifacts.
Dynamic Multi-Step Planning	Completion requires long dependencies, conditional branches, intermediate decisions, or replanning after new observations.
Multi-Source Inconsistency	Information from users, tools, files, or environment states is incomplete, duplicated, outdated, or mutually conflicting.
Progressive Disclosure and State Evolution	Critical information or constraints are revealed gradually through user turns, tool results, approval stages, or state transitions.
Risk, Reliability, and Clarification	The task involves irreversible actions, insufficient evidence, conflicting instructions, or operations that require explicit confirmation, recovery handling, or refusal.

B.3 Representative Domain-Level Analysis

To complement the aggregate results across ToB, ToC, and ToE, we further compare four representative models over ten level-1 domains. As shown in Figure 9, model rankings vary substantially across application scenarios. GLM-5.2 performs best in Shopping and Healthcare, Claude-Opus-4.7 leads in Productivity, Lifestyle, and Navigation, while Qwen3.7-Max achieves the strongest result in IT & Internet. DeepSeek-V4-Pro performs particularly well in Travel, Banking & Finance, and Business, and matches Qwen3.7-Max on Personal Finance. These pronounced rank reversals indicate that no model consistently dominates across domains, and that similar aggregate scores may conceal substantially different scenario-specific strengths. The results further motivate evaluating general agents with broad domain coverage and reporting fine-grained performance beyond a single overall score.

C Data Annotation and Challenging-Set Statistics

C.1 Human Annotation

The formal annotation process involves 14 annotators, all of whom receive unified instructions and task-specific training before annotation. A pilot annotation stage is conducted to familiarize annotators with the criteria for task formulation, environment support, rubric design, capability assignment, and difficulty assessment, and to calibrate interpretations before full-scale annotation.

The 14 annotators are organized into seven pairs for double checking. Every retained instance is reviewed by at least two annotators. Disagreements are referred to designated reviewers for adjudication, while instances whose ambiguity or inconsistency cannot be reliably resolved are discarded. Human review covers the major components of the constructed data, including whether the task is

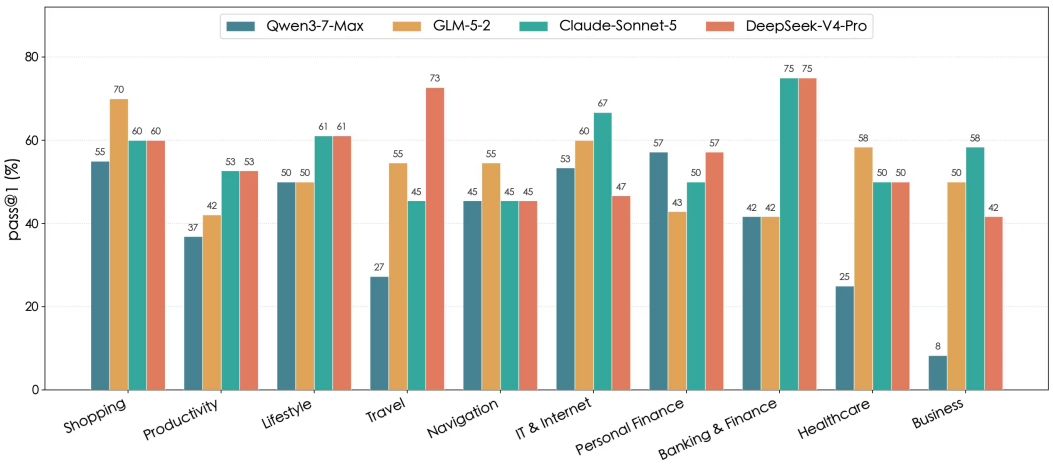


Figure 9: **Performance across representative agent scenarios.** Pass@1 results of four representative models over ten level-1 domains.

sufficiently specified and executable, whether the environment and tools support the required operations, whether the rubrics cover the essential task requirements, and whether capability and difficulty annotations are consistent with the task content. Capability and difficulty labels are generated with model assistance and subsequently reviewed and revised by annotators rather than adopted directly from model outputs.

The pilot annotation also shows consistent interpretations across annotators. Domain-related judgments reach 100% agreement. For task difficulty scored on a five-point scale, the mean pairwise absolute difference between annotators is approximately 0.33 points, and 94.3% of pairwise ratings differ by no more than one point. The relative difficulty rankings further achieve a Kendall's W of approximately 0.75. These results indicate that the annotation criteria lead to broadly consistent judgments across annotators after calibration.

C.2 Challenging-Set Data Statistics

Table 8 details the 644-task challenging set selected in Section 3.3: 354 DAG, 200 DAG-S, 60 Solver, and 30 Program tasks. The DAG subset preserves coverage of all 354 level-2 domains by retaining the longest reference tool chain in each domain; the other routes exclude low-complexity cases. The full-set statistics follow in Appendix D. The challenging set is intentionally constructed to retain instances with higher task complexity and sufficient discriminative power while preserving broad domain coverage. Its Pass@1 results therefore characterize performance under the challenging distribution defined by OmniaBench, rather than estimating success rates under the natural frequency distribution of real-world user requests.

The routes place different demands on agents. Program tasks have the longest descriptions (4,443.6 tokens on average) and the most instantiated entities (312.4), reflecting their procedural and state complexity. DAG tasks expose the most candidate tools (65.3), while DAG-S tasks combine relatively long requests (2,922.0 tokens) with dense states (205.7 entities). Solver tasks use fewer tools (8.2), with complexity arising from constraints and optimization objectives.

Evaluation requirements also vary: DAG-S has the most rubric items (13.0) and total item points (32.4), while Solver has the longest criterion descriptions (429.3). Program tasks instead use a binary *VerifyCode* verifier. These differences complement the aggregate benchmark characteristics in Figure 3 and help contextualize performance across routes.

D Full-Set Data Statistics

Table 9 summarizes the statistics of the full OmniaBench dataset, complementing the challenging-set statistics reported in the main text. The full set contains 1,431 tasks in total, including 1,102

Table 8: **Statistics of the challenging set across task categories.** All values are averaged over tasks within each category except #Tasks. Note that the Program route uses a binary *VerifyCode* verifier instead of multi-item textual rubrics. Statistics for the full set are reported in Appendix Table 9.

Metric	DAG	DAG-S	Solver	Program	Overall
Dataset Size					
#Tasks	354	200	60	30	**644**
Task					
Task length	953.2	2922.0	1450.5	4443.6	**1773.6**
Environment					
#Tools	65.3	56.9	8.2	52.2	**59.0**
#Entity tables	16.4	13.1	6.7	12.6	**13.7**
#Entities	70.7	205.7	111.1	312.4	**125.2**
Rubric or VerifyCode					
# Items	9.0	13.0	5.8	1	**10.0**
Item points	21.6	32.4	5.8	1	**23.6**
Description length	133.9	251.8	429.3	*VerifyCode*	**201.2**

DAG tasks, 222 DAG-S tasks, 60 Solver tasks, and 47 Program tasks. Although the four routes share a common environment and evaluation framework, their distinct construction objectives lead to substantially different task and environment characteristics.

Table 9: Overall statistics of different task categories in full-set data. All values are averaged over tasks within each category except #Tasks. The Program route uses a binary *VerifyCode* verifier instead of multi-item textual rubrics.

Metric	DAG	DAG-S	Solver	Program	Overall
Dataset Size					
#Tasks	1102	222	60	47	**1431**
Task					
Task length	845.6	2933.7	1450.5	4431.4	**1312.7**
Environment					
#Tools	82.6	56.4	8.2	51.7	**68.6**
#Entity tables	16.4	13.1	6.7	12.9	**14.9**
#Entities	70.7	208.1	111.0	308.2	**100.1**
Rubric or VerifyCode					
#Items	8.9	13.0	5.8	1	**9.4**
Item points	21.6	32.4	5.8	1	**22.6**
Description length	117.1	252.8	429.3	*VerifyCode*	**152.4**

Program tasks have the longest average task descriptions, reaching 4,431.4 tokens, reflecting their richer procedural constraints, branching structures, and execution requirements. DAG-S tasks are also relatively long, with an average length of 2,933.7, whereas DAG tasks are more concise on average. The routes additionally differ in environment scale. DAG tasks expose the largest tool space, with 82.6 candidate tools per task on average. In contrast, Program and DAG-S tasks contain denser state spaces, with 308.2 and 208.1 instantiated entities, respectively. Solver tasks involve fewer tools and entity tables, as their complexity primarily arises from structured constraints and optimization objectives rather than broad tool exploration.

The rubric statistics further reflect the differences among the four routes. DAG-S tasks contain the largest number of rubric items and rubric points, with averages of 13.0 and 32.4, respectively, indicating more fine-grained completion criteria. Solver tasks contain fewer rubric items, but their rubric descriptions are the longest on average, suggesting that their evaluation criteria require more detailed specification of feasibility, constraint satisfaction, and objective consistency. Across the full dataset except the program route, each task contains an average of 9.4 rubric items and 22.6 rubric points, providing structured and fine-grained supervision across diverse task formats.

E ROUTE-SPECIFIC EVALUATION PROTOCOLS

OmniaBench contains four data routes with different interaction and verification protocols. The DAG route adopts multi-turn interaction, whereas Solver, Program, and DAG-S are evaluated in single-turn settings.

E.1 DAG: MULTI-TURN INTERACTIVE EVALUATION

For DAG tasks, a persona-grounded user simulator initiates the conversation and progressively provides task-relevant information according to the interaction history. At each turn, the evaluated agent may respond to the user, invoke tools, or request clarification. The interaction terminates when the user simulator emits `###STOP###`, or when the predefined execution budget is exhausted.

DAG Evaluation Flow

1. The user simulator generates the initial request from the task specification and persona.
2. The agent interacts with the user and environment through messages and tool calls.
3. The user simulator reveals additional information or provides feedback when required.
4. Multiple rubric items assess whether all task objectives and constraints are satisfied.

This protocol evaluates not only task execution, but also clarification, constraint tracking, adaptation to user feedback, and state maintenance across multiple turns.

E.2 SOLVER AND DAG-S: SINGLE-TURN RUBRIC EVALUATION

Solver and DAG-S tasks provide the full user request at the beginning of the trajectory. The agent may invoke multiple tools, but no additional user feedback is introduced after the initial request. Execution terminates when the agent submits its final response.

Both routes are evaluated using multiple rubric items. Each item corresponds to a concrete task objective or constraint and receives either its assigned score or zero. A task is considered successful only when all rubric items are satisfied. Solver tasks additionally emphasize constraint satisfaction and the consistency of the produced decision with the underlying planning, allocation, scheduling, or optimization objective.

E.3 PROGRAM: SINGLE-TURN PROGRAMMATIC VERIFICATION

Program tasks also follow a single-turn protocol, but use programmatic verification instead of rubric-based judgment. After completing the required tool interactions, the agent submits its final response. The associated *VerifyCode* then determines the binary result directly from the complete trajectory and final observation.

Program Verification Flow

1. The agent receives the task request and available tool interfaces.
2. The agent gathers information and performs the required operations.
3. The agent submits its final response.
4. *VerifyCode* checks the trajectory and final observation and returns a binary pass/fail result.

F PERSONA DESIGN

Personas are used in DAG tasks to create diverse and realistic user behaviors. Each persona specifies three groups of attributes:

- **Core identity**: professional role, background, and life stage;
- **Interaction characteristics**: expression style, emotional tendency, information-disclosure strategy, decision preference, and patience;
- **Real-world constraints**: time, budget, privacy, and other scenario-dependent considerations.

These attributes control how the user expresses requests, reveals missing information, responds to clarification, and determines whether the interaction should continue.

F.1 Persona Example: School Alumna

Saint Saviour School Alumna

Core identity. A graduate familiar with the school's academic culture, learning pace, and campus traditions.

Interaction characteristics. She communicates in an organized and experience-grounded manner, actively corrects inaccurate descriptions, and prefers focused questions. She is willing to discuss representative school experiences while withholding unnecessary personal details.

Constraints and preferences. She values long-term educational impact over superficial reputation, avoids disclosing information about other individuals, and expects the conversation to respect the school's specific cultural and educational context.

F.2 Persona Example: Television Producer

Visionary Television Producer

Core identity. An experienced television producer interested in socially meaningful and underrepresented stories.

Interaction characteristics. The producer frames discussions around characters, conflicts, themes, and narrative structure. They engage deeply with insightful ideas but quickly reject generic suggestions or formulaic expressions.

Constraints and preferences. They balance authenticity, communication value, filming feasibility, production cost, and ethical boundaries. Information about unaired projects and real subjects is disclosed cautiously.

G User Prompt Composition

The initial prompt provided to the user simulator combines three components:

$$\text{User Prompt} = \text{Persona} \oplus \text{Environment Summary} \oplus \text{Task Description}. \tag{28}$$

User Prompt Template

```
[PERSONA CARD] + [ENVIRONMENT SUMMARY] + [TASK
                 DESCRIPTION]
```

Persona card. The persona card determines the simulated user's background, communication pattern, information-disclosure behavior, preferences, and practical constraints.

Environment summary. The environment summary briefly describes the application context represented by the available tools and state, such as a geographic information system connected to an asset-management database.

Task description. The task description specifies the user objective, operational constraints, expected outcome, and any information that may be disclosed progressively during interaction. It is written from the user's perspective and avoids exposing the intended tool sequence or reference trajectory.

For example, a maintenance-dispatch task may ask the agent to identify at most one suitable work order while jointly considering urgency, asset availability, ongoing repair status, and recent assignment changes. The agent must retrieve the relevant evidence, resolve competing constraints, and avoid unnecessary state modifications.

H Representative Cases

We next present two representative route1 tasks. For each case, we show both an environment-entity view and a candidate-tool view so that the task can be read from two complementary angles: the structured state that the agent must reason over and the representative operations that the task invokes.

H.1 Case 1

Case A: Litigation Matter Environment

Environment: `LawFirmLitigationMatterManagementSystem` **Global ID:** `803`
Domain: `Legal Industry → Attorney` **Task ID:** `env_22_task_001`

Task prompt (condensed).

> The task verifies the personal injury defense matter `LF-2024-PI-024` for Yunhe Foods. It confirms the client, responsible attorney, and opposing party, checks whether the archived investigation memorandum remains usable for linkage, reviews the evidence chain for completeness, and then adds a newly collected evidence item while updating the corresponding matter timeline.

Environment Entities snapshot.

Entity Groups

Matter Core:
`matters`
`clients`
`attorneys`
Document Layer:
`documents`
`document_versions`

Concrete Instance Slice

Matter profile

`matter_number`	`LF-2024-PI-024`
`client`	`Yunhe Foods Co.`
`attorney`	`Lin Qiaoxue`

Evidence-chain slice

`existing_evidence`	`EVD-004`
`link_id`	`LNK-003`

Candidate Tools snapshot.

Candidate Tool Groups

Matter Resolution:
`get_client_by_id`
`get_attorney_by_id`
Document & Version Checks:
`get_document`
`list_document_versions`

Concrete Tool Slice

Read-side checks

`get_client_by_id`	`CLT-002`
`get_attorney_by_id`	`ATT-002`

Write-side actions

`create_evidence`	store surveillance screenshots
`create_matter_event`	supplement filing event

Selected rubric criteria.

Rubric	Criterion excerpt
`G1`	Correctly identify the target matter as the Yunhe Foods personal injury defense matter with matter number `LF-2024-PI-024`.
`T1`	Perform an integrity/completeness review for the matter and detect the evidence-chain problem, specifically the self-referential evidence-link anomaly.
`T4`	Create a corresponding matter event for the supplement and ensure the matter activity timeline reflects the addition without changing the matter's stayed status.

H.2 CASE 2

Case B: Password Vault Environment

Environment: `PasswordVaultAppEnvironment` **Global ID:** `825`
Domain: `Tools → Password Management` **Task ID:** `env_3_task_001`

Task prompt (condensed).

The task performs a security cleanup for user `user_zheng_2048`: confirm accessible vaults and trusted devices, inspect a shared work login with unresolved breach risk, create a new work login and an encrypted note, verify passkey and note autofill behavior, and summarize export, reminder, and alert state across the vault.

Environment Entities snapshot.

Entity Groups

Vault State:
`user_vaults`
`vault_items`
`categories`
Device Surface:
`device_permissions`
`sync_states`
Credential Surface:
`login_credentials`
`passkey_credentials`
Security & Audit:
`share_records`
`export_jobs`
`import_batches`

Concrete Entity Slice

Vault and device state

`vault_id`	`vault_main_001`
`vault_status`	`active`
`device_id`	`DEV-ALPHA-01`
`device_name`	`LunaBook Pro`
`is_trusted`	`True`
`is_online`	`True`

Credential and alert state

`login_item`	`ITEM_LOGIN_001`
`passkey_item`	`ITEM_PASSKEY_002`
`secure_note`	`ITEM_NOTE_004`
`alert_id`	`ALERT_001`
`share_id`	`SHR_001`
`export_id`	`EXP_001`

Candidate Tools snapshot.

Candidate Tool Groups

Vault & Device Lookup:
`get_vaults`
`list_devices`
`get_device_permissions`
Credential Inspection:
`get_login_item`
`get_passkey_item`
`get_secure_note`
State-Changing Writes:
`create_credential`
`create_secure_note`
Security & Audit Checks:
`list_reminders`
`get_export_job`
`list_share_records`
`get_change_history`

Concrete Tool Slice

Read-side checks

`get_vaults`	`vault_main_001`
`list_devices`	`DEV-ALPHA-01`, `DEV-BETA-02`, `DEV-GAMMA-03`
`get_login_item`	`ITEM_LOGIN_001`
`get_passkey_item`	`ITEM_PASSKEY_002`
`get_secure_note`	`ITEM_NOTE_004`
`list_alerts`	`ALERT_001 (open)`

Write-side actions and audit

`create_credential`	`https://hr.atlas.example`
`create_secure_note`	incident follow-up note
`list_share_records`	`SHR_001`
`get_export_job`	`EXP_001`
`get_change_history`	new-login trace required
`list_reminders`	rotation / expiry reminders

Selected rubric criteria.

Rubric	Criterion excerpt
`G1`	Correctly identify the active vault owned by `user_zheng_2048` and the device best suited for sensitive actions as the trusted online laptop.
`G3`	Correctly summarize the vault security posture relevant to the review, including the compromised login, open alert, and due reminder horizon.
`T1--T2`	For the Aurora Mail login, verify lookup consistency, sharing state, and unresolved breach-alert status without inventing duplicate records.
`T5`	Aggregate export, import, reminder, and open-alert state and explicitly point out remaining inconsistencies or items needing human confirmation.

I TRAJECTORY CASE STUDY

I.1 CASE 1

Case Example: Litigation Matter Evidence Update

[1] Task Description *task excerpt*

Review litigation matter `LF-2024-PI-024` for Yunhe Foods: confirm client, responsible attorney, and opposing party; verify that the filed incident-investigation memo remains the current linkable version; run a completeness review with emphasis on evidence-chain anomalies; then add a new evidence item linked to `DOC-003`, and confirm the resulting trace record, activity timeline ...

[... additional description omitted ...]

[2] Tools *tool summary*

Matter lookup by number; client and attorney retrieval; full matter workspace lookup; document and version retrieval; matter-integrity validation; evidence-subgraph and neighbor traversal; evidence insertion; matter-event lookup; activity-timeline retrieval.

[... additional tools omitted ...]

[3] Agent Setup *prompt excerpts*

Tested agent "You are a helpful assistant. Your goal is to complete the user's request in an interactive environment by gradually calling the available tools step by step, and to proactively communicate with the user ...

User agent "You are role-playing as a user interacting with an agent. Your persona is written inside the `persona` tags. Your task is to gradually communicate the content in `instructions` to the agent ...

Judge model "You are a strict evaluator of task execution results. You will be given: 1. task: the task description 2. init_state: the initial state 3. final_state: the final state 4. final_assistant_response: the assistant's ...

[4] Demonstration *message excerpt*

User *Initial request*
Please review `LF-2024-PI-024`: first confirm the client, responsible attorney, and opposing party; then check whether the archived incident-investigation memorandum is still the current reliable version for linking; run a completeness review of the case file with special attention to the evidence chain; if the matter is stayed but not closed, add a new evidence item for store ...

Agent *Plan* Resolve the matter by number, verify party ownership, inspect `DOC-003` and its version history, validate integrity and traverse the evidence chain around `EVD-004`, then insert the new evidence item and re-check the post-update ...

Tool *Tool call* `validate_matter_integrity(MAT-2024-002)` → 1 violation; `EvidenceLink LNK-003 creates a self-loop.` `add_evidence(..., evidence_id=EVD-005)` → ID already exists; retry with `EVD-006` succeeds and links the new record to `DOC-003`.

Agent *Final response* The final workspace keeps the matter in `stayed` status, confirms `VER-004` as the current filed memo version, surfaces the self-loop anomaly on `LNK-003`, adds `EVD-006` with the requested metadata, and records the new filing ...

[... additional turns omitted ...]

[5] Rubrics and Judgment *selected rubric excerpts*

Overall pass@1: Pass (✓) **Rubric: 20/20** **Steps: 21**

Rubric T1 **3/3 pts**

Criterion

Perform an integrity review for the matter and detect the evidence-chain problem, specifically the self- ...

Judge reason

The trajectory correctly resolves `LF-2024-PI-024`, confirms Yunhe Foods / Lin Qiaoxue / He Shan, verifies `VER-004` as current, and flags the self-referential `LNK-003` link.

Rubric T4 **3/3 pts**

Criterion

Create a corresponding matter event for the supplement and ensure the activity timeline reflects the addition on ...

Judge reason

After one rejected ID attempt, the agent successfully adds `EVD-006`, links it to `DOC-003`, and confirms the resulting record through `event_11`, the matter timeline, and the full ....

I.2 CASE 2

Appendix Case Example: Expanded Litigation Matter Evidence Update

[1] Task Description *expanded task excerpt*

Please help verify the Yunhe Foods personal-injury defense matter `LF-2024-PI-024`. The request proceeds in stages: first confirm the client, responsible attorney, and opposing party; then check whether the archived accident-investigation memorandum remains the current usable filed version; then run a completeness review of the matter with particular attention to evidence-chain anomalies. If the matter is stayed but not closed, and the memorandum can be used for attachment/linkage, add a newly collected evidence item titled "store surveillance screenshots and incident timeline explanation" with source *store surveillance export*, collected on 2024-05-15, filed on 2024-05-16, authenticity pending verification, relevance high, custody secured, and a description tied to lighting, warning signage, and fall location, linked to `DOC-003`. Finally, reconfirm the trace record for this filing, inspect the full matter activity timeline, and return an updated workspace view that still exposes any unresolved completeness conflict.

[2] Tools *expanded tool summary*

`get_matter_by_number`, `get_client_by_id`, `get_attorney_by_id`, `get_matter_full_context`, `get_document_by_id`, `list_document_versions`, `validate_matter_integrity`, `get_evidence_chain_subgraph`, `get_evidence_chain_neighbors`, `get_evidence_by_id`, `add_evidence`, `list_matter_events`, and `get_matter_activity_timeline`. These tools jointly expose matter identity, party records, current document-version state, graph-structured evidence relations, state-changing evidence insertion, and post-update audit trails.

[3] Agent Setup *prompt excerpts*

Tested agent "You are a helpful assistant. Your goal is to complete the user's request in an interactive environment by gradually calling the available tools step by step, and to proactively communicate with the user when necessary until the user ends the conversation. At each step, you will receive one of two types of information: a user reply, or a tool-call result returned by the environment."

User agent "You are role-playing as a user interacting with an agent. Your persona is written inside the `persona` tags. Your task is to gradually communicate the content in `instructions` to the agent through a multi-turn conversation."

Judge model "You are a strict evaluator of task execution results. You will be given: 1. task: the task description 2. init_state: the initial state 3. final_state: the final state 4. final_assistant_response: the assistant's final response to the user, if any 5. trajectory_summary: a summary of the execution trajectory, including tool calls, parameters, and key observations 6. rubric_items: the list of rubric items to judge one by one. Your goal: Based on this evidence, make a binary judgment for each rubric item."

[4] Demonstration *expanded message excerpt*

User *Initial request*

Please review `LF-2024-PI-024`: confirm the client, responsible attorney, and opposing party; verify whether the archived incident-investigation memorandum is still the current filed version that can be relied upon for linking; run a completeness review of the full matter with special attention to evidence-chain anomalies; if the matter is stayed but not closed, add a new evidence item for store-surveillance screenshots and incident-timeline explanation, linked to `DOC-003`; then reconfirm the resulting trace record, inspect the full activity timeline, and return an updated workspace summary that still shows the unresolved integrity issue for later follow-up.

Agent *Plan* Resolve the matter from `LF-2024-PI-024`, pull the linked client / attorney / party records, inspect `DOC-003` and its version history, run an integrity check over the matter graph, traverse the evidence neighborhood around `EVD-004`, then decide whether the stated preconditions for evidence insertion hold before creating the supplemental record and validating the resulting event / timeline / workspace views.

Tool *Tool sequence* `get_matter_by_number(LF-2024-PI-024)` → `MAT-2024-002`, status `stayed`, attorney `ATT-002`, client `CLT-002`. `get_client_by_id(CLT-002)` → Yunhe Foods Co. `get_attorney_by_id(ATT-002)` → Lin Qiaoxue. `get_matter_full_context(MAT-2024-002)` → opposing party He Shan, plaintiff-side counsel Qin Yiran. `get_document_by_id(DOC-003)` and `list_document_versions(DOC-003)` → single filed current version `VER-004`.

Tool *Integrity review* `validate_matter_integrity(MAT-2024-002)` → one violation. `get_evidence_chain_subgraph(seed=EVD-004, depth=3)` → only `EVD-004` plus link `LNK-003`. `get_evidence_chain_neighbors(EVD-004)` → both inbound and outbound edges point back to `EVD-004`, confirming a self-referential loop explicitly marked as invalid.

Tool *State change and verification* `add_evidence(..., evidence_id=EVD-005)` → rejected because the identifier already exists. Retry with `EVD-006` → success; the new evidence is documentary, source *store surveillance export*, collected on 2024-05-15, submitted on 2024-05-16, authenticity pending, relevance high, custody secured, linked to `DOC-003`. `get_evidence_by_id(EVD-006)` confirms the inserted record. `list_matter_events(...)` returns `event_11`; `get_matter_activity_timeline(MAT-2024-002)` places the new filing after the 2024-05-14 stay and before the postponed 2024-05-16 status conference; `get_matter_full_context(MAT-2024-002)` returns the updated workspace with both evidence records and the original unresolved self-loop.

Agent *Final response* The assistant returns a consolidated workspace summary: matter `LF-2024-PI-024` remains `stayed`; party ownership and adverse-side records are consistent; `DOC-003` remains anchored to current filed version `VER-004`; `EVD-006` is added successfully with the requested metadata and memo linkage; a new evidence-added event `event_11` appears on 2024-05-16; and the prior completeness conflict persists as `LNK-003`, a self-loop on `EVD-004` that is preserved for later remediation rather than silently removed.

[5] Rubrics and Judgment *expanded rubric excerpts*

Overall pass@1: Pass (✓) **Rubric: 20/20** **Steps: 21**

Rubric G4 / T1 **5/5 pts**

Criterion

Check the current archival/document-version status for the incident-investigation memorandum, and perform an integrity review that detects the evidence-chain problem, specifically the self-referential link anomaly.

Judge reason

The trajectory retrieves `DOC-003`, inspects the version list, confirms that `VER-004` is the sole filed current version, then runs the integrity validator and evidence-chain traversal tools, all of which consistently identify `LNK-003` as a self-loop on `EVD-004`.

Rubric T2 / T3 / T4 **9/9 pts**

Criterion

Add the new evidence item with the required metadata and memo linkage, ensure the description captures lighting / signage / fall-location support, and confirm that the supplement is reflected in the matter event log and activity timeline without changing the matter's stayed status.

Judge reason

After one failed identifier attempt, the assistant retries with `EVD-006`, inserts the full evidence record, verifies the inserted fields directly, confirms the generated evidence-added event `event_11` on 2024-05-16, and returns a post-update workspace in which the matter remains `stayed` while the new filing is visible in the timeline.