OL-BENCH: A COMPREHENSIVE BENCHMARK FOR EVALUATING AI AGENTS IN OMNI-LIFE SCENARIOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-based agents are increasingly applied to real-life interactive scenarios, where they are required to understand evolving user needs, follow environmental constraints, interact over multiple turns, and use tools to complete tasks. However, existing benchmarks often provide limited scenario coverage, simplify interactive environments, and primarily evaluate final task completion, creating a significant gap between benchmark evaluation and real-life application settings. Consequently, they may fail to faithfully reflect agents' actual performance in practice. In this work, we present **Omni-Life Benchmark (OL-Bench)**, a comprehensive interactive benchmark for evaluating agents across diverse real-life scenarios. OL-Bench adopts an Entity–Tool–Instruction hierarchy to synthesize executable interactive environments. Based on these environments, dependency-aware multi-turn tasks are constructed with explicitly planned answers and multi-level distractors across four difficulty levels, ranging from easy to extreme. These tasks are executed through a scenario-specific Harness that coordinates environment state transitions, tool use, and interaction trajectories. OL-Bench contains 1055 tasks across 218 real-life scenarios, with an average of 46.4 tools and 20.2 Skills per scenario. It further provides 18 metrics across seven capability dimensions to evaluate both task completion and interaction processes. Our comprehensive evaluation shows that GPT-5.6-Sol, the best-performing model, achieves a Full Task Completion Rate of only 18.0%, underscoring the difficulty of complex real-life interactions for current agents. These results demonstrate the value of OL-Bench for systematically evaluating agent capabilities and diagnosing their limitations in real-life interactive scenarios.

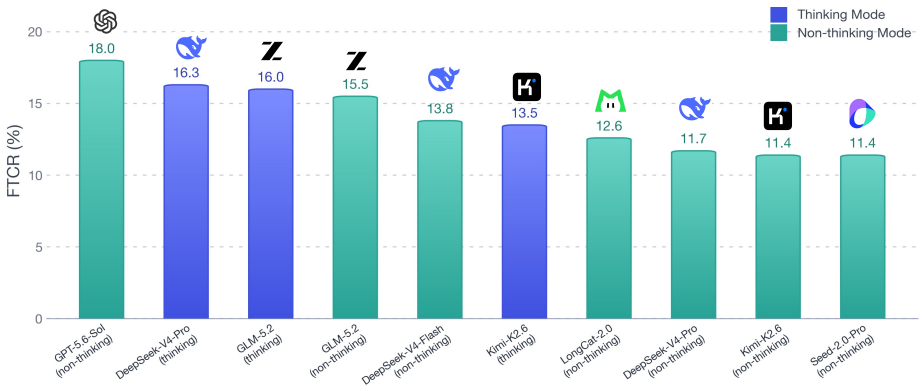

Figure 1: Full task completion performance of thinking and non-thinking models on OL-Bench.

1 INTRODUCTION

With the rapid progress of large language models (LLMs) in reasoning, planning, and tool use (Wei et al., 2022; Yao et al., 2023; Schick et al., 2023), LLM-based agents demonstrate promising capabilities in applications such as code generation and information retrieval (Yan et al., 2025b; Hui 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.

2024; Jin et al., 2025). However, extending these agents to broader real-life applications requires them to integrate multiple capabilities, such as understanding, interaction, tool use, and decision making (He et al., 2026b; Lyu et al., 2026; He et al., 2026c).

Real-life scenarios often involve explicit or implicit environment constraints, progressively revealed user needs, and complex environment states. Successfully completing tasks therefore requires agents to infer user intent over multiple turns, use tools to retrieve information and perform actions, and adhere to environmental constraints throughout the interaction (Shen et al., 2026; Chen et al., 2026).

However, existing real-life interactive agent benchmarks still struggle to capture the diversity and complexity of real-life applications. First, their scenario and task coverage remains limited in both diversity and scale (Chen et al., 2026; Li et al., 2026). Expanding to new scenarios requires constructing scenario-specific data, executable tools, operational rules, and interaction workflows, making large-scale coverage costly and difficult. Second, existing benchmarks often simplify interactive environments, failing to capture several important challenges in real-life applications, including complex tool-use workflows, variations in tool interfaces, and challenging distractor candidates. While such simplifications make environment construction and task design more tractable, they also limit the extent to which existing benchmarks reflect the challenges encountered in realistic interactions. Finally, evaluation is still largely centered on final task completion, with limited fine-grained assessment of critical interactive capabilities such as intent understanding, constraint adherence, and tool use, making it difficult to diagnose where and why agents fail in long-horizon interactions (He et al., 2026b). These limitations raise a fundamental question:

How can we benchmark agents in real-life interactions?

Accordingly, we present **Omni-Life Benchmark (OL-Bench)**, a comprehensive interactive benchmark for evaluating agents across diverse real-life scenarios. First, OL-Bench synthesizes executable environments from predefined scenario blueprints through an Entity–Tool–Instruction hierarchy. The Entity layer defines the entities, attributes, and relations that constitute environment states; the Tool layer provides executable interfaces for querying and modifying these states; and the Instruction layer specifies scenario-specific rules and workflows through system prompts and Skills. This structured design enables scalable synthesis across diverse scenarios. Second, OL-Bench introduces complexity at both the environment and task levels. At the environment level, scenario-specific Skills encode operational rules and multi-stage workflows, while tool perturbations introduce variations in tool interfaces. At the task level, OL-Bench generates dependency-aware multi-turn tasks with explicitly planned answers and multi-level distractors, and organizes them into four difficulty levels ranging from easy to extreme. During interaction, a scenario-specific Harness coordinates environment state transitions, tool use, and interaction trajectories.

With these designs, OL-Bench contains 1,055 tasks across 218 real-life scenarios, with an average of 46.4 tools and 20.2 Skills per scenario, while each task requires an average of 5.1 tools and 4.2 Skills. Approximately 83.6% of the entities involved in each task serve as distractors. OL-Bench further organizes tasks into four difficulty levels—Easy, Medium, Hard, and Extreme—comprising 196, 364, 296, and 199 tasks, respectively. Finally, we introduce a multi-dimensional interaction evaluation framework with 18 metrics across seven capability dimensions, including task completion, constraint adherence, intent understanding, tool use effectiveness, factuality, behavioral consistency, and interaction efficiency, enabling fine-grained evaluation and failure attribution across both task completion and interaction processes.

We conduct systematic evaluations of seven representative models under the Non-thinking setting, with three models additionally evaluated under the Thinking setting. Even the best-performing model, GPT-5.6-Sol, achieves a Full Task Completion Rate of only 18.0%, revealing substantial limitations of current agents in handling complex, multi-stage real-life interactions.

The main contributions of this work are summarized as follows:

- **Scenario and task synthesis.** We synthesize real-life scenarios through the Entity–Tool–Instruction hierarchy and construct multi-turn tasks with controllable difficulty.
- **Large-scale benchmark with broad scenario coverage and fine-grained evaluation.** We construct a large-scale interactive benchmark covering a broad range of scenarios, and

Table 1: Comparison of existing interactive agent benchmarks. A check mark (✓) denotes that the benchmark explicitly evaluates or incorporates the corresponding component, a triangle (△) denotes partial evaluation or limited support, and a cross (×) denotes no explicit evaluation or support.

Benchmark	Environment Construction			Task Design & Scale			Capability Evaluation						
	Harness	Skill	Tool Perturbation	Difficulty Levels	# Scenarios	# Tasks	Task Completion	Constraint Adherence	Intent Understanding	Tool Use	Factuality	Behavioral Consistency	Interaction Efficiency
τ-Bench (Yao et al., 2025)	×	×	×	×	2	165	✓	△	×	×	×	×	×
τ^2-Bench (Barres et al., 2025)	×	×	×	✓	3	279	✓	△	×	×	×	×	×
τ-Knowledge (Shi et al., 2026)	×	×	×	×	1	97	✓	△	×	×	×	×	✓
UserBench (Qian et al., 2025)	×	×	×	✓	1	471	✓	△	×	✓	×	×	✓
VitaBench (He et al., 2026b)	×	×	×	×	3	400	✓	△	×	×	×	×	×
VitaBench 2.0 (Chen et al., 2026)	✓	×	×	×	3	56	✓	△	×	×	×	×	×
TRIP-Bench (Shen et al., 2026)	×	×	×	✓	1	—	✓	△	△	×	△	×	✓
OL-Bench (Ours)	✓	✓	✓	✓	**218**	**1055**	✓	✓	✓	✓	✓	✓	✓

introduce a fine-grained, multi-dimensional evaluation framework to systematically characterize both task-level and process-level agent capabilities.

- **Systematic experiments and capability analysis.** We conduct extensive experiments on seven representative models, analyzing their capability boundaries and typical failure patterns in real-life interactions.

2 Related Work

Existing agent benchmarks have explored different aspects of the capabilities required for real-life interactions, ranging from basic tool use to long-horizon task completion. Early benchmarks such as ToolLLM (Qin et al., 2024), API-Bank (Li et al., 2023), and ACEBench (Chen et al., 2025) mainly evaluate tool selection, function calling, and argument generation. ToolSandbox (Lu et al., 2025) introduced stateful tool execution and implicit state dependencies between tools, while the τ-Bench family (Yao et al., 2025; Barres et al., 2025; Shi et al., 2026) further incorporates domain policies, simulated users, stateful databases, and knowledge-intensive tool use. Recent benchmarks introduce additional sources of interaction complexity: VitaBench and VitaBench 2.0 (He et al., 2026b; Chen et al., 2026) consider diverse real-life scenarios and personalized interactions; UserBench (Qian et al., 2025) emphasizes progressively revealed user preferences; TRIP-Bench (Shen et al., 2026) studies long-horizon planning under complex and evolving constraints; and Agency-Bench and Terminal-Bench (Li et al., 2026; Merrill et al., 2026) investigate extended execution trajectories and large-scale tool use.

Alongside increasingly complex environments and tasks, prior work has also developed different mechanisms for evaluating agent behavior. The τ-Bench family (Yao et al., 2025; Barres et al., 2025; Shi et al., 2026) verifies task completion through environment states, while TRIP-Bench and UserBench (Shen et al., 2026; Qian et al., 2025) emphasize user requirement satisfaction and constraint adherence. Despite these advances, existing benchmarks typically emphasize particular aspects of environment synthesis, task complexity, or capability evaluation. OL-Bench brings these dimensions together by combining diverse real-life scenarios, systematically constructed dependency-aware multi-turn tasks with controllable complexity, and fine-grained evaluation of both task completion and interaction processes, as summarized in Table 1.

3 Benchmark Construction

3.1 Environment Synthesis

OL-Bench adopts an Entity–Tool–Instruction hierarchy to synthesize real-life interactive scenarios. The Entity layer defines structured world states, the Tool layer exposes executable interfaces for querying and modifying these states, and the Instruction layer specifies scenario-specific agent behaviors through system prompts and Skills.

Real-World Grounding. OL-Bench is designed based on real-world service platforms, public API and product documentation, common operational workflows, and prior studies of agent interactions (Lan et al., 2025; He et al., 2026a; Yan et al., 2025a). These sources guide the design of

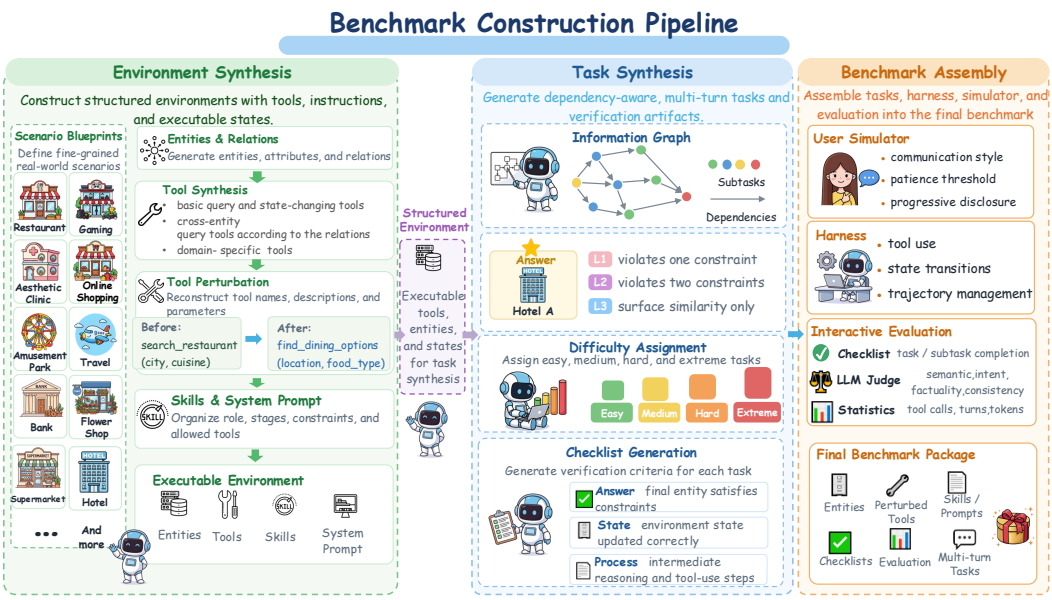

Figure 2: Benchmark construction pipeline of OL-Bench. Starting from scenario blueprints, OL-Bench synthesizes entities, executable tools, Skills and prompts, and dependency-aware multi-turn tasks, then assembles them with a user simulator, a scenario-specific harness, and interactive evaluation into the final benchmark package.

scenarios, including the entities involved, their relationships, and the operations that agents can perform. Based on these real-world references, OL-Bench can further extend entities, compose new scenarios, and synthesize tools, enabling scalable benchmark construction while keeping the resulting interactions representative of real-life applications.

Entity Synthesis. We first construct scenario blueprints that specify representative real-life scenario semantics and their core entity types. An LLM then expands the required entities into a large entity pool and defines a structured schema for each entity, including its attributes, data types, and state-related fields. The resulting entity schemas are then automatically composed into concrete interactive environments, allowing entities to be reused and combined across different scenarios. Entity dependencies are further modeled through foreign-key references and shared attributes, forming structured world states for subsequent tool synthesis and task construction.

Tool Synthesis and Perturbation. Based on the synthesized entity schemas and their relations, executable tool interfaces are automatically constructed to expose the capabilities of the underlying environment. Each entity is equipped with basic query and state-changing operations, while cross-entity query tools are derived from entity relations. Scenario-specific operations, such as ordering and verification, are further synthesized according to the corresponding workflows. Finally, tool names, descriptions, and parameter specifications are perturbed while preserving their functionality and execution semantics, reducing reliance on superficial interface cues.

Instruction Design. The Instruction layer consists of environment-specific system prompts and Skills. System prompts define agent roles, environment rules, response requirements, and safety boundaries, while Skills organize related tools and provide procedural guidance for specific workflows. Approximately 15%–25% of tools are directly available, while the remaining tools are exposed through Skills that agents must load according to evolving user needs and task states.

3.2 Task Design

Dependency-aware Task Planning. For each task, an information graph is automatically generated, where nodes represent retrieval or execution objectives and edges specify dependencies and information flow between objectives. Diverse graph structures, including linear, parallel, fan-out, diamond, conditional, multi-hop, and deep execution chains, are sampled to introduce variations in

Table 2: Statistics of OL-Bench.

Statistic	Value
Benchmark scale	
# scenarios / # tasks	218 / 1,055
tools per scenario (mean)	46.4
Skills per scenario (mean)	20.2
required tools per task (mean)	5.1
required Skills per task (mean)	4.2
Task difficulty	
Easy / Medium / Hard / Extreme	196 / 364 / 296 / 199
Task structure	
checklist items (min / mean / max)	1 / 4.7 / 15
information-graph edges (min / mean / max)	0 / 4.1 / 16
entity instances per task (min / mean / max)	20 / 91.0 / 378
Information-graph topology	
single / linear / parallel / fan-out	15.6% / 31.7% / 14.2% / 11.4%
diamond / conditional / multi-hop / deep chain	12.4% / 5.9% / 5.4% / 3.4%

task structure, while retrieval width is adjusted to control candidate scale. The resulting graph is then converted into a natural-language task description with internal identifiers removed.

Candidate Supply Generation. For task nodes involving entity selection, we generate valid answers together with multi-level distractors according to different degrees of constraint violation. Answers satisfy all task constraints; L1 distractors violate exactly one constraint while remaining close to the corresponding constraint boundary; L2 distractors violate two task constraints; and L3 distractors share only superficial semantic similarity with the target entities. Each node contains at least 5 L1, 3 L2, and 8 L3 distractors, requiring agents to distinguish fine-grained constraint satisfaction rather than relying on coarse semantic similarity.

Difficulty Levels. Task difficulty is controlled along multiple dimensions, including scenario scope, entity scale, and operation-chain complexity, and tasks are categorized into four levels: Easy, Medium, Hard, and Extreme. Easy tasks primarily involve simple queries or single-step operations with a small number of entities. Medium tasks cover complete workflows within a single scenario. Hard tasks further extend to composite tasks spanning two subdomains, involving more entities and longer dependency chains. Extreme tasks require end-to-end execution across three or more subdomains, typically featuring larger entity sets, more complex cross-scenario dependencies, and longer sequences of consecutive operations.

Task Verification Construction. Each synthesized task is paired with a Checklist that specifies subtask-level completion criteria. Checklist dependencies are automatically derived from the task information graph, while write operations additionally specify expected environment-state changes, such as successful order creation or coupon redemption. During evaluation, the Judge determines whether each Checklist item is satisfied based on the interaction trajectory and environment states.

Human Quality Control. We randomly sample 40 environments and 100 tasks for manual inspection to verify semantic coherence, tool executability, constraint consistency, and alignment between task descriptions and verification criteria. Identified issues are corrected before evaluation.

3.3 Interaction Execution

User Simulation. To support multi-turn interaction, we construct 576 user profiles and randomly sample one for each task. The user simulator only accesses the user-side task description and sampled profile, without access to target entities, distractors, or internal environment states, preventing ground-truth leakage. During interaction, it progressively reveals user requirements and provides relevant constraints when queried. The interaction terminates when the task is completed, the simulator confirms completion, or the maximum number of turns is reached.

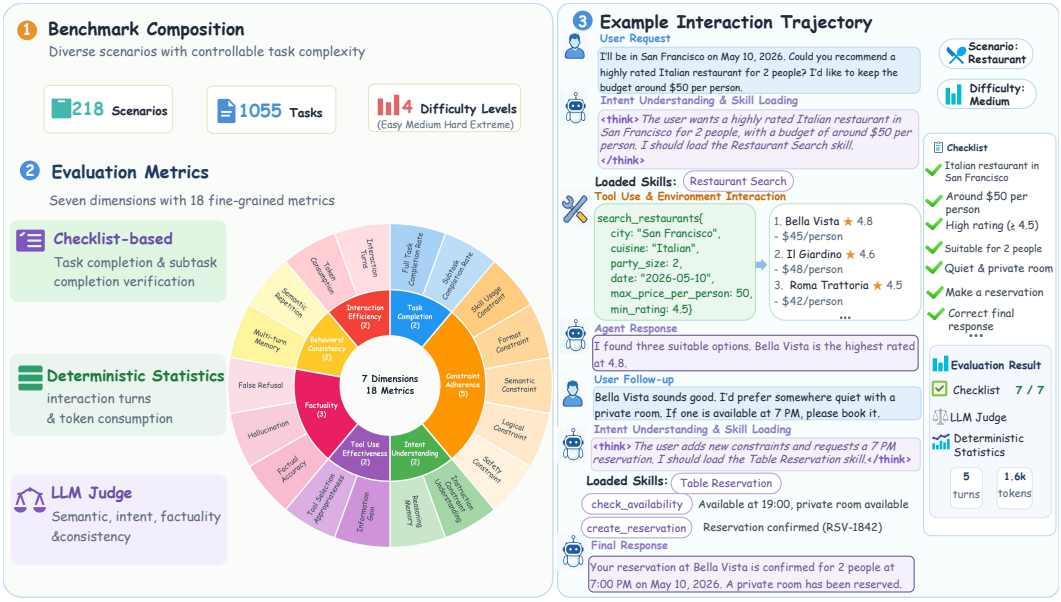

Figure 3: Overview of OL-Bench, including benchmark composition, evaluation framework, and an example interaction trajectory.

Interaction Harness. The synthesized environments and tasks are executed through a scenario-specific Harness. The Harness coordinates agent–user interactions, tool use, environment state updates, and trajectory recording. It also manages execution configurations, including the maximum number of tool calls, context compression, memory settings, and maximum interaction turns. These configurations are sampled from predefined distributions during environment synthesis and remain fixed within each environment during evaluation.

4 Evaluation Metrics

Beyond task-level success, OL-Bench provides fine-grained evaluation of agent behavior with 18 metrics across seven dimensions: **Task Completion**, **Constraint Adherence**, **Intent Understanding**, **Tool Use Effectiveness**, **Factuality**, **Behavioral Consistency**, and **Interaction Efficiency**.

(1) **Task Completion** evaluates both overall success and partial progress using the task Checklist. C_{sat} and C_{valid} denote the numbers of satisfied and valid Checklist items, respectively. We compute the **Subtask Completion Rate** as $\text{SCR} = C_{\text{sat}}/C_{\text{valid}}$, and the **Full Task Completion Rate** as $\text{FTCR} = \mathbb{I}[C_{\text{sat}} = C_{\text{valid}}]$. (2) **Constraint Adherence** evaluates Skill Usage, Format, Semantic, Logical, and Safety constraints. C^c_{sat} and C^c_{app} denote the numbers of satisfied and applicable constraints of category c; the score is computed as $C^c_{\text{sat}}/C^c_{\text{app}}$, with untriggered conditional constraints excluded. (3) **Intent Understanding** evaluates whether agents correctly identify user intent and historical references. T_{pass} and T_{valid} denote the numbers of correctly handled and valid interaction turns; each metric is computed as $T_{\text{pass}}/T_{\text{valid}}$ over its corresponding valid turns. (4) **Tool Use Effectiveness** evaluates Tool Selection Appropriateness and Information Gain. U_{valid} denotes the number of valid tool calls, U_{app} the number of appropriate calls, and U_{info} the number of calls that introduce useful new information. The corresponding scores are $U_{\text{app}}/U_{\text{valid}}$ and $U_{\text{info}}/U_{\text{valid}}$. (5) **Factuality** evaluates false refusals, hallucinations, and factual accuracy, with each metric computed over its corresponding valid response turns. (6) **Behavioral Consistency** evaluates multi-turn consistency and semantic repetition. (7) **Interaction Efficiency** directly measures the total number of interaction turns and token consumption required for task execution.

Metrics directly obtainable from interaction trajectories are computed deterministically, while the remaining metrics are evaluated by the Judge model. At the dimension level, we report SCR and FTCR separately for Task Completion and average the constituent metrics for the remaining dimensions. Detailed definitions of all 18 metrics are provided in Appendix A.1

Table 3: Main results on OL-Bench. The upper table reports seven models under the non-thinking setting. The lower table compares non-thinking and thinking settings on three representative models. ↑ indicates higher is better.

Model	Task Completion		Constraint Adherence ↑	Intent Understanding ↑	Tool Use Effectiveness ↑	Factuality ↑	Behavioral Consistency ↑	Avg. Turns	Avg. Tokens (10k/task)
	FTCR ↑	SCR ↑							
DeepSeek-V4-Pro	11.7	48.7	45.5	75.3	79.5	77.5	57.4	18.9	28.2
DeepSeek-V4-Flash	13.8	50.5	45.5	76.6	80.5	76.1	50.6	21.7	57.8
GPT-5.6-Sol	**18.0**	**53.7**	**59.4**	**84.2**	81.8	**88.2**	**66.5**	21.7	35.6
LongCat-2.0	12.6	48.9	41.7	70.9	81.3	72.2	49.0	20.2	29.7
GLM-5.2	15.5	52.3	48.0	73.3	82.7	71.1	57.1	22.7	57.2
Seed-2.0-Pro	11.4	45.2	46.1	71.7	**83.5**	73.1	61.2	16.4	35.3
Kimi-K2.6	11.4	45.7	43.7	72.7	81.8	72.5	50.5	18.9	40.1

Model	Setting	Task Completion		Constraint Adherence ↑	Intent Understanding ↑	Tool Use Effectiveness ↑	Factuality ↑	Behavioral Consistency ↑	Avg. Turns	Avg. Tokens (10k/task)
		FTCR ↑	SCR ↑							
DeepSeek-V4-Pro	Non-thinking	11.7	48.7	45.5	75.3	79.5	77.5	57.4	18.9	28.2
	Thinking	**16.3**	**56.0**	**51.6**	**81.8**	**84.1**	**80.2**	**66.1**	19.0	39.9
Kimi-K2.6	Non-thinking	11.4	45.7	43.7	72.7	81.8	72.5	50.5	18.9	40.1
	Thinking	**13.5**	**52.2**	**47.3**	**77.7**	**83.0**	**79.1**	**60.3**	18.0	50.2
GLM-5.2	Non-thinking	15.5	52.3	48.0	73.3	82.7	71.1	57.1	22.7	57.2
	Thinking	**16.0**	**53.7**	**50.8**	**79.5**	**83.8**	**76.2**	**64.3**	17.9	60.7

5 EXPERIMENTS

5.1 SETTINGS

Evaluated Models. We evaluate 7 representative large language models on OL-Bench, covering diverse model families and scales, including **DeepSeek-V4-Flash** (DeepSeek-AI, 2026), **DeepSeek-V4-Pro** (DeepSeek-AI, 2026), **GPT-5.6-Sol** (OpenAI, 2026), **LongCat-2.0** (Meituan LongCat Team, 2026), **Kimi-K2.6** (Moonshot AI, 2026), **GLM-5.2** (GLM-5-Team, 2026), and **Seed-2.0-Pro** (Bytedance Seed, 2026). All seven models are evaluated under the Non-thinking setting, while DeepSeek-V4-Pro, Kimi-K2.6, and GLM-5.2 are additionally evaluated under the Thinking setting. All models are evaluated under the same protocol with identical environment instances, tool sets, instruction configurations, and harness settings to ensure fair comparison.

Implementation Details. All benchmark construction procedures, including environment and task synthesis, are performed using DeepSeek-V4-Flash (DeepSeek-AI, 2026). We also use DeepSeek-V4-Flash (DeepSeek-AI, 2026) as the user simulator. For each task, the sampled user profile is fixed and shared across all evaluated models and settings. The Judge model is based on GPT-5.6-Sol (OpenAI, 2026). For each model–task pair, we execute a single interaction trajectory. Under this protocol, FTCR is equivalent to Pass@1, while all other metrics are derived from the same single-run trajectories. For metric aggregation, we first average results within each scenario and then across scenarios to avoid bias toward scenarios with more tasks.

5.2 MAIN RESULTS

Table 3 presents the overall results of seven models on OL-Bench under the Non-thinking setting, together with a comparison between Non-thinking and Thinking settings for three of them. We make the following observations.

OL-Bench is challenging for current agents. Across all evaluated models and settings, the highest Full Task Completion Rate (FTCR) is only 18.0%, achieved by GPT-5.6-Sol under the Non-thinking setting. In contrast, Subtask Completion Rate (SCR) is substantially higher across models. For example, GPT-5.6-Sol achieves 53.7% SCR but only 18.0% FTCR, while DeepSeek-V4-Pro under Thinking reaches 56.0% SCR with an FTCR of 16.3%. This large gap indicates that current agents can complete a considerable portion of individual objectives, yet often fail to satisfy all requirements

within the same multi-turn interaction, making full end-to-end task completion substantially more difficult than partial progress.

Models with similar task completion rates can show different strengths and weaknesses. The fine-grained results reveal clear differences across capability dimensions even when models achieve similar FTCRs. For example, under the Non-thinking setting, Seed-2.0-Pro and Kimi-K2.6 both achieve an FTCR of 11.4%, but differ in Behavioral Consistency (61.2% vs. 50.5%), while showing similar Intent Understanding (71.7% vs. 72.7%). Models also exhibit different interaction behaviors: GLM-5.2 uses the most interaction turns on average (22.7), whereas Seed-2.0-Pro requires only 16.4 turns. These results show that similar end-to-end task completion rates can arise from different capability profiles and interaction patterns, motivating the need for fine-grained evaluation beyond final task outcomes.

Thinking generally improves task and capability performance, with varying gains across models. Across the three models evaluated under both settings, the average FTCR increases from 12.9% to 15.3%, an absolute improvement of 2.4 percentage points. DeepSeek-V4-Pro shows the largest FTCR improvement, from 11.7% to 16.3%, followed by Kimi-K2.6 from 11.4% to 13.5% and GLM-5.2 from 15.5% to 16.0%. Thinking also improves all reported score-based capability metrics for these three models, while increasing average token consumption from 41.8×10^4 to 50.3×10^4 tokens per task. Nevertheless, the gains in end-to-end completion remain modest relative to their SCR improvements, and even the best Thinking result reaches only 16.3% FTCR. This suggests that additional reasoning helps improve intermediate capabilities and partial task progress, but does not fully resolve the difficulty of completing all required steps and constraints end to end.

6 In-Depth Analysis

Performance Analysis Across Difficulty Levels. As shown in Figure 4(a), under the Non-thinking setting, both FTCR and SCR generally decrease as task difficulty increases from Easy to Extreme, indicating a clear progression in task challenge. Averaged across the four models, FTCR decreases from 31.5% on Easy tasks to 14.5% on Medium, 11.9% on Hard, and 10.6% on Extreme tasks, while SCR decreases from 66.2% to 50.6%, 49.2%, and 44.2%, respectively. The larger drop in FTCR than SCR suggests that increasing difficulty affects end-to-end completion more strongly than partial progress. Even on Hard and Extreme tasks, models still complete over 44% of subtasks on average, while only around 10–12% of tasks are fully completed. This indicates that current agents become substantially less capable of completing tasks end to end as interaction complexity increases.

Performance Across Interaction Complexity. As shown in Figure 4(c), FTCR decreases consistently as interaction complexity increases. When the number of required tools increases from 1–3 to 6+, FTCR drops from 30.6% to 6.1% under Thinking and from 27.7% to 4.2% under Non-thinking. Similar declines are observed as the number of required Skills increases from 0–1 to 5+ and as dependency chain length increases from 1–2 to 5+. These results show that current agents become substantially less reliable as interaction workflows require more tools, Skills, and cross-step dependencies.

Failure Analysis. To understand the main causes of failure, we analyze failure trajectories from seven models under the non-thinking setting. As shown in Figure 4(b), the dominant error is **insufficient filtering and verification (59.4%)**, where agents retrieve relevant candidates but fail to determine whether they satisfy all task constraints. Typical cases include omitted conditions, incorrect thresholds, and relaxing task requirements. **Retrieval and query strategy errors (36.1%)** mainly result from omitted queries, overly broad searches, failure to load appropriate Skills, or repeated ineffective retrieval, preventing agents from obtaining the information needed for subsequent steps. **Missing write operations (3.5%)** arise when agents complete retrieval and selection but fail to execute required actions such as ordering, reservation, or coupon redemption. **Information interpretation errors (1.0%)** involve incorrect use of retrieved information, such as matching information to the wrong entity or overlooking a valid result. Overall, more than 95% of failures occur during information retrieval, constraint filtering, and result verification, suggesting that the main challenges for non-thinking agents lie in effective information retrieval and consistent constraint handling throughout multi-step task execution.

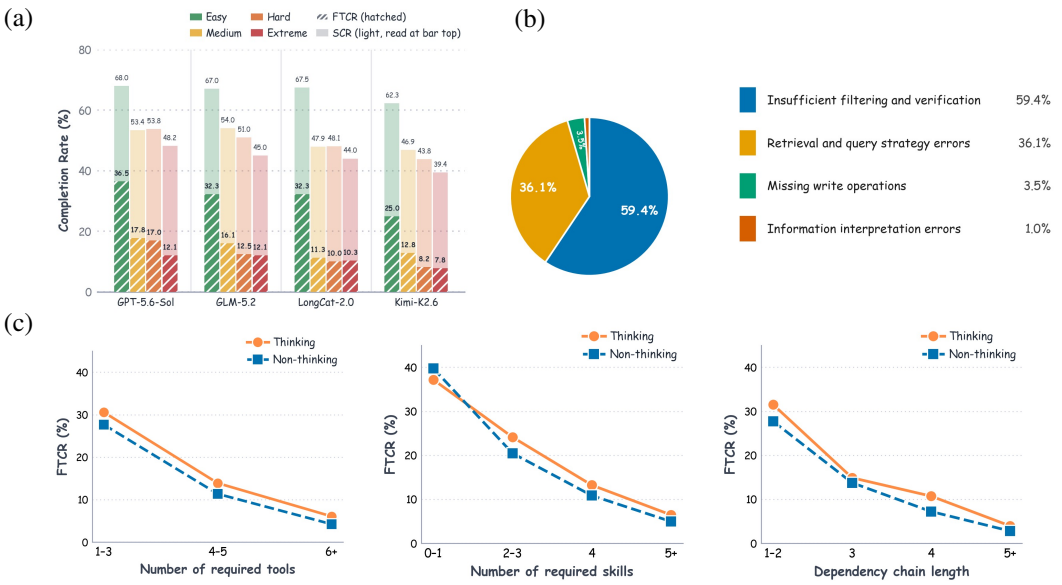

Figure 4: (a) Performance across different difficulty levels under the Non-thinking setting of four representative models. (b) Distribution of failure modes on OL-Bench. (c) Average FTCR under Thinking and Non-thinking settings across three interaction-complexity factors: required tools, required Skills, and dependency chain length.

7 Reliability Analysis

Reliability of Judge-based Evaluation. To validate the alignment between the LLM-based Judge and human evaluation, we randomly sample 300 interaction trajectories and manually review all applicable Judge-based evaluation items. For each trajectory, three human annotators independently label all applicable Judge-based items using the same evaluation criteria as the Judge model. Human reference labels are obtained through item-level majority voting. We then compare the Judge model's predictions against these human reference labels using **Raw Agreement** and **Cohen's** κ. The Judge achieves a Raw Agreement of **95.3%** and a Cohen's κ of **0.84**, indicating strong agreement with human evaluation and supporting its reliability for large-scale automated evaluation in OL-Bench.

Reliability of User Simulation. We further evaluate the user simulator along two dimensions: **Information Fidelity** and **Persona Consistency**. Information Fidelity measures whether the simulated user accurately maintains task constraints and known information while avoiding the disclosure of inaccessible information, whereas Persona Consistency assesses whether it consistently follows the predefined user profile. We randomly sample 300 interaction trajectories, each independently rated by three human annotators on a 1–10 scale. For each trajectory, we first average the scores from the three annotators and then compute the overall mean and standard deviation across all 300 trajectories. The simulator achieves **8.7** $\pm$ **0.8** for Information Fidelity and **8.4** $\pm$ **0.5** for Persona Consistency. These results indicate that the simulator generally maintains task information and user personas consistently throughout multi-turn interactions.

8 Conclusion

In this work, we introduced **OL-Bench**, a comprehensive benchmark for evaluating agents in diverse real-life interactive scenarios. OL-Bench synthesizes executable environments through an Entity–Tool–Instruction hierarchy, constructs dependency-aware multi-turn tasks with controllable difficulty, and evaluates agents with 18 metrics across seven capability dimensions. Experiments show that current agents often succeed on individual subtasks but still struggle to achieve reliable end-to-end task completion as task complexity and constraint dependencies increase. We hope OL-Bench provides a systematic testbed for advancing more capable and reliable real-world agents.

AI USE STATEMENT

In this work, generative AI is used as an integral component of our methodology. OL-Bench is constructed through a pipeline, where large language models are employed to generate entity data, tool implementations, Skills, system instructions, task information graphs, supply data, and task validation signals based on predefined scenario blueprints. Detailed procedures are described in Section 3. Generative AI is also used as the user simulator during interactions and as the Judge model for semantic evaluation metrics, with corresponding configurations provided in Section 5 and Appendix A.1. These uses constitute part of our research methodology. To ensure the reliability of AI-generated artifacts, we manually inspected a randomly sampled set of 40 environments and 100 tasks to verify semantic coherence, tool executability, constraint consistency, and alignment between task descriptions and verification criteria. We further manually annotated 300 interaction trajectories to measure the agreement between the Judge model and human evaluation. We have reviewed all AI-assisted work and take full responsibility for the final content of this paper.

ETHICS STATEMENT

This work does not involve real end users or real user interaction data. All interactions in OL-Bench occur between language-model-based agents and user simulators in sandbox environments. All entities are synthetically generated and contain no personally identifiable information. All benchmark records and attributes are synthetically constructed and do not represent real listings, transactions, or user records. Sensitive operations such as payments, refunds, and medical appointments are fully simulated without interacting with real systems. We also evaluate safety-related behaviors, including privacy protection, refusal of unauthorized operations, and confirmation before high-risk actions, to support the safe deployment of agents.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed descriptions of the benchmark construction, evaluation protocol, and quality control process. Section 3 presents the construction pipeline, including difficulty configuration, distractor generation, and task topology sampling. Appendix A.1 provides the definitions and evaluation criteria of all 18 metrics. Section 5 describes the experimental protocol, including model configurations, user simulator, and Judge model settings.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Bytedance Seed. Seed2.0 model card: Towards intelligence frontier for real-world complexity, 2026. URL https://arxiv.org/abs/2607.00248.

Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Wang Xinzhi, and Wu Liu. ACEBench: A comprehensive evaluation of LLM tool usage. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 12970–12998, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.697. URL https://aclanthology.org/2025.findings-emnlp.697/.

Yuxin Chen, Yi Zhang, Zhengzhou Cai, Yaorui Shi, Zhiyuan Yao, Chenhang Cui, Jingnan Zheng, Yaqi Huo, Xi Su, Qi Gu, Xunliang Cai, Xiang Wang, An Zhang, and Tat-Seng Chua. Vitabench 2.0: Evaluating personalized and proactive agents in long-term user interactions, 2026. URL https://arxiv.org/abs/2605.27141.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

GLM-5-Team. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Hang He, Chuhuai Yue, Chengqi Dong, Mingxue Tian, Hao Chen, Zhenfeng Liu, Jiajun Chai, Xiaohan Wang, Yufei Zhang, Qun Liao, Guojun Yin, Wei Lin, Chengcheng Wan, Haiying Sun, and Ting Su. Localsearchbench: Benchmarking agentic search in real-world local life services. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 8989–9000, New York, NY, USA, 2026a. Association for Computing Machinery. ISBN 9798400722592. doi: 10.1145/3770855.3817466. URL https://doi.org/10.1145/3770855.3817466.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi GU, Hui Su, and Xunliang Cai. Vitabench: Benchmarking LLM agents with versatile interactive tasks in real-world applications. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=rtcX9qOBaz.

Yun He, Wenzhe Li, Hejia Zhang, Songlin Li, Karishma Mandyam, Sopan Khosla, Yuanhao Xiong, Nanshu Wang, Xiaoliang Peng, Beibin Li, Shengjie Bi, Shishir G Patil, Qi Qi, Shengyu Feng, Julian Katz-Samuels, Richard Yuanzhe Pang, Sujan Kumar Gonugondla, Hunter Lang, Yue Yu, Yundi Qian, Maryam Fazel-Zarandi, Licheng Yu, Amine Benhalloum, Hany Hassan Awadalla, and Manaal Faruqui. AdvancedIF: Rubric-based benchmarking and reinforcement learning for advancing LLM instruction following. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18003–18022, San Diego, California, United States, July 2026c. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.820. URL https://aclanthology.org/2026.acl-long.820/.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Xiaochong Lan, Jie Feng, Jiahuan Lei, Xinlei Shi, and Yong Li. Benchmarking and advancing large language models for local life services. In Luiza Antonie, Jian Pei, Xiaohui Yu, Flavio Chierichetti, Hady W. Lauw, Yizhou Sun, and Srinivasan Parthasarathy (eds.), *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pp. 4566–4577. ACM, 2025. doi: 10.1145/3711896.3737196. URL https://doi.org/10.1145/3711896.3737196.

Keyu Li, Junhao Shi, Yang Xiao, Mohan Jiang, Jie Sun, Yunze Wu, Dayuan Fu, Shijie Xia, Xiaojie Cai, Tianze Xu, Weiye Si, Wenjie Li, Dequan Wang, and Pengfei Liu. AgencyBench: Benchmarking the frontiers of autonomous agents in 1M-token real-world contexts. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7422–7440, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.337. URL https://aclanthology.org/2026.acl-long.337/.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In Luis Chiruzzo,

Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Yuanjie Lyu, Chengyu Wang, Lei Shen, Jun Huang, and Tong Xu. Mock worlds, real skills: Building small agentic language models with synthetic tasks, simulated environments, and rubric-based rewards. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12529–12545, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.570. URL https://aclanthology.org/2026.acl-long.570/.

Meituan LongCat Team. Introducing LongCat-2.0, June 2026. URL https://longcat.ai/blog/longcat-2.0/.

Mike A Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Kwesi Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak.

Moonshot AI. Kimi K2.6: Advancing open-source coding, April 2026. URL https://www.kimi.com/en/blog/kimi-k2-6.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, July 2026. URL https://openai.com/index/gpt-5-6/.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Zhiwei Liu, Jianguo Zhang, Haolin Chen, Heng Ji, Weiran Yao, Shelby Heinecke, Silvio Savarese, Caiming Xiong, and Huan Wang. Userbench: An interactive gym environment for user-centric agents, 2025. URL https://arxiv.org/abs/2507.22034.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Yuanzhe Shen, Zisu Huang, Zhengyuan Wang, Muzhao Tian, Zhengkang Guo, Chenyang Zhang, Shuaiyu Zhou, Zengjie Hu, Dailin Li, Kaimin Wang, Wenhao Liu, Tianlong Li, feng hong, Cao Liu, and Ke Zeng. TRIP-bench: A benchmark for long-horizon interactive agents in real-world scenarios. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=bF1V9A56d3.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik R Narasimhan, and Victor Barres. τ-knowledge: Evaluating conversational agents over unstructured knowledge. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=XHZK5abtw2.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, Shiliang Yang, Zhirui Wang, Brian Li, Kang An, Chenyang Li, Lei Lei, Mengmeng Duan, Danxun Liang, Guodong Liu, Hang Cheng, Hao Wu, Jie Dong, Junhao Huang, Mei Chen, Renjie Yu, Shunshan Li, Xu Zhou, Yiting Dai, Yineng Deng, Yingdan Liang, Zelin Chen, Wen Sun, Chengxu Yan, Chunqin Xu, Dong Li, Fengqiong Xiao, Guanghao Fan, Guopeng Li, Guozhen Peng, Hongbing Li, Hang Li, Hongming Chen, Jingjing Xie, Jianyong Li, Jingyang Zhang, Jiaju Ren, Jiayu Yuan, Jianpeng Yin, Kai Cao, Liang Zhao, Liguo Tan, Liying Shi, Mengqiang Ren, Min Xu, Manjiao Liu, Mao Luo, Mingxin Wan, Na Wang, Nan Wu, Ning Wang, Peiyao Ma, Qingzhou Zhang, Qiao Wang, Qinlin Zeng, Qiong Gao, Qiongyao Li, Shangwu Zhong, Shuli Gao, Shaofan Liu, Shisi Gao, Shuang Luo, Xingbin Liu, Xiaojia Liu, Xiaojie Hou, Xin Liu, Xuanti Feng, Xuedan Cai, Xuan Wen, Xianwei Zhu, Xin Liang, Xin Liu, Xin Zhou, Yifan Sui, Yingxiu Zhao, Yukang Shi, Yunfang Xu, Yuqing Zeng, Yixun Zhang, Zejia Weng, Zhonghao Yan, Zhiguo Huang, Zhuoyu Wang, Zihan Yan, Zheng Ge, Jing Li, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, and Daxin Jiang. Step-gui technical report, 2025a. URL https://arxiv.org/abs/2512.15431.

Kaiwen Yan, Hongcheng Guo, Xuanqing Shi, Shaosheng Cao, Donglin Di, and Zhoujun Li. CodeIF: Benchmarking the instruction-following capabilities of large language models for code generation. In Georg Rehm and Yunyao Li (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 1272–1286, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-288-6. doi: 10.18653/v1/2025.acl-industry.89. URL https://aclanthology.org/2025.acl-industry.89/.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. {τ}-bench: A benchmark for \underline{T}ool-\underline{A}gent-\underline{U}ser interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

A Appendix

A.1 Evaluation Metrics Details

This section presents the complete definitions and evaluation criteria of the 18 metrics organized into seven evaluation dimensions in OL-Bench. Among these metrics Interaction Turns and Token Consumption are computed using deterministic statistics without model-based judgment. The remaining metrics are evaluated by a Judge model based on structured prompts. All Judge-based metrics are reported as pass rates, where a higher score indicates better performance. For error-oriented criteria such as hallucination, false refusal, and semantic repetition, a pass indicates the absence of the

corresponding failure. For each evaluation item, the Judge model first performs explicit analysis and then outputs item-level judgments, including pass labels, confidence scores, and rationales, to support manual inspection and result reproducibility.

Table 4: Evaluation dimensions and metrics in OL-Bench.

Dimension	Metric	Evaluation Method
Task Completion	Full Task Completion Rate	Judge Model
	Subtask Completion Rate	Judge Model
Constraint Adherence	Skill Usage Constraint	Judge Model
	Format Constraint	Judge Model
	Semantic Constraint	Judge Model
	Logical Constraint	Judge Model
	Safety Constraint	Judge Model
Intent Understanding	User Intent Understanding	Judge Model
	Reasoning-based Memory	Judge Model
Tool Use Effectiveness	Tool Selection Appropriateness	Judge Model
	Information Gain	Judge Model
Factuality	False Refusal	Judge Model
	Hallucination	Judge Model
	Factual Accuracy	Judge Model
Behavioral Consistency	Multi-turn Consistency	Judge Model
	Semantic Repetition	Judge Model
Interaction Efficiency	Interaction Turns	Deterministic
	Token Consumption	Deterministic

A.1.1 Task Completion

Task Completion measures whether an agent can accomplish end-to-end tasks aligned with user objectives. Since real-world tasks often consist of multiple interdependent subgoals, evaluating only final success or failure cannot fully reflect an agent's performance on complex tasks. Therefore, we evaluate both overall task completion and fine-grained subtask completion.

Task completion is evaluated based on the Checklist generated alongside each task. The Judge model independently determines whether each checklist item is satisfied. Full Task Completion Rate requires all checklist items to be satisfied, producing a binary score of 0 or 1, and reflects the agent's end-to-end task completion capability. Subtask Completion Rate is defined as the proportion of satisfied checklist items among all valid items, capturing partial completion in long-horizon tasks. For checklist items associated with specific triggering conditions, items whose conditions are not activated in the current task trajectory are excluded from the evaluation.

A.1.2 Constraint Adherence

Constraint Adherence measures an agent's ability to follow the explicitly specified rules in the system prompt during task execution. In real-world scenarios, agents are required not only to achieve user objectives but also to satisfy various constraints, such as workflow requirements, output formats, and safety regulations. Therefore, this dimension evaluates whether the agent's behavior conforms to the requirements defined by the environment.

Since different environments may contain varying numbers and forms of constraints, we adopt a two-stage constraint evaluation approach. First, a model automatically extracts all checkable constraints from the system prompt and performs deduplication and normalization. Then, the extracted constraints and the complete interaction trajectory are jointly provided to the Judge model for item-level evaluation.

Each constraint is assigned one of three states: satisfied, violated, or not applicable. A constraint is marked as satisfied if it is applicable and fulfilled by the agent; it is marked as violated if it is applicable but the agent's behavior violates the constraint. For conditional constraints whose

triggering conditions are never activated throughout the entire trajectory, the constraint is marked as not applicable and excluded from the denominator. The final score is calculated as the proportion of satisfied constraints among all applicable constraints (i.e., satisfied plus violated constraints). Based on their semantics, constraints are further categorized into Skill Usage Constraint, Format Constraint, Semantic Constraint, Logical Constraint, and Safety Constraint. This categorization is only used for result analysis and dimension-wise breakdown.

A.1.3 Intent Understanding

Intent Understanding measures an agent's ability to identify user requirements, understand task objectives, and make decisions based on contextual information. In real-world scenarios, user needs are often expressed progressively through multi-turn interactions, requiring agents to accurately understand user requirements, intended actions, and relevant historical information.

User Intent Understanding is evaluated at the turn level. The Judge model first analyzes the requirements expressed in the current user message, including filtering conditions, scope limitations, and desired actions, and then determines whether the agent's response and tool use are consistent with the user's underlying intent. Turns that only contain casual conversation, confirmations, or additional information without explicit task requirements are excluded from evaluation. If the agent's behavior is consistent with the user's intent but the task cannot be completed due to unavailable information in the environment, the response is still considered successful, distinguishing model limitations from environmental constraints.

Reasoning-based Memory evaluates whether an agent can correctly leverage conversational context to identify the target when users refer to previously mentioned information. For turns involving pronouns, ordinal references, temporal references, or references to historical entities, the Judge model determines whether the agent correctly identifies the reference relation, whether the target is inferred based on historical context, and whether the final operated entity matches the user's intended target. If the agent fails to utilize historical information and instead treats the request as a new query, even if it accidentally retrieves the correct result, the case is not considered successful.

A.1.4 Tool Use Effectiveness

Tool Use Effectiveness measures an agent's ability to interact with external tools and leverage environmental resources to accomplish tasks. In real-world environments, effective tool use requires not only selecting appropriate tools based on user requirements, but also acquiring useful information through tool interactions to support subsequent decision-making.

Tool Selection Appropriateness is evaluated at the tool-call level to determine whether each tool call is appropriate for the current interaction context. The Judge model examines whether the selected tool provides the required capability and whether there exists a substantially more appropriate tool for accomplishing the current objective. If the selected tool is sufficient to support task progression, it is not considered inappropriate even when alternative tools are available.

Information Gain measures whether a tool use introduces new and useful information relative to the current interaction state. Repeated calls with identical parameters, equivalent queries, or redundant invocations after sufficient information has already been obtained are considered to provide low information gain. In contrast, a tool call is considered informative if it returns new entities, attributes, or environment states.

A.1.5 Factuality

Factuality measures the consistency between an agent's generated responses and the underlying environment state. In tool-augmented tasks, tool outputs serve as the primary source of factual evidence available to the agent. Therefore, an agent should avoid generating unsupported information while accurately conveying facts provided by the environment.

This dimension evaluates agent responses at the turn level based on tool-returned results as the factual reference. False Refusal detects cases where the agent claims that it cannot provide assistance or that no relevant information is available despite the existence of valid and relevant tool results. Hallucination identifies unsupported statements or information that is inconsistent with the environment

state and absent from tool outputs. Factual Accuracy measures whether the agent correctly conveys available factual information, including entity attributes, prices, ratings, addresses, and business statuses.

A.1.6 BEHAVIORAL CONSISTENCY

Behavioral Consistency measures an agent's ability to maintain consistent states and perform stable reasoning throughout long-horizon interactions. As interactions become progressively more complex, agents are required to effectively integrate historical information and avoid contradictory or redundant behaviors during task execution.

Unlike Reasoning-based Memory, which evaluates whether historical references are correctly resolved, Multi-turn Consistency evaluates whether the agent maintains stable beliefs after information is acquired. If changes in the agent's statements are supported by newly obtained tool results, they are considered reasonable updates. However, if the agent changes its conclusions without any newly available information, the behavior is considered inconsistent.

Semantic Repetition detects cases where an agent repeatedly outputs existing information without introducing new content. This includes paraphrasing the same information, repeatedly reporting already confirmed results, and unnecessary extensions. Responses that provide additional details, comparisons, or further suggestions are not considered repetitive.

A.1.7 INTERACTION EFFICIENCY

Interaction Efficiency measures the interaction cost required for an agent to accomplish a task. A capable agent should not only successfully complete tasks but also minimize unnecessary interactions and computational overhead. Therefore, this dimension complements task performance evaluation by analyzing the efficiency of task execution.

Interaction Turns measures the total number of agent–user and agent–tool interactions throughout task execution. Token Consumption records the total input and output token usage of the agent, providing an estimate of the token-level interaction cost required to complete the task. This dimension should be analyzed jointly with task completion performance to distinguish genuinely efficient agents from those that achieve success through redundant interactions.

A.1.8 COMMON JUDGING PROTOCOL

All Judge Model-based metrics follow a unified evaluation protocol.

First, we adopt a conservative judgment principle: when available evidence is insufficient to confirm an error or violation, the case is considered non-violating by default, reducing false positives caused by Judge model uncertainty. Second, conditional constraints that are not triggered, interaction turns without explicit task requirements, and turns without genuine reference to historical information are excluded from the denominators of their corresponding metrics, ensuring consistent evaluation criteria across different tasks and trajectories. Third, we adopt a responsibility separation principle by explicitly defining the evaluation scope of each metric. For example, Intent Understanding does not evaluate response formatting, while Constraint Adherence does not assess tool selection appropriateness, preventing the same failure behavior from being repeatedly counted by multiple metrics.

All evaluation dimensions can be executed independently and in parallel. For each interaction trajectory, OL-Bench generates a unified evaluation card containing the scores and detailed judgments across all metrics.

A.2 AN EXAMPLE SYSTEM PROMPT

This section provides an example of the system prompt for Hotel Search Task in OL-Bench.

System Prompt

Environment
- Current system time: `2026-01-29 12:55`
- User location city: `Shanghai`
- Merchant business status is determined by the `is_closed` field returned by tools, while `business_hours` is used only as supplementary information.

I. Role and Positioning
You are the **"SmartStay Assistant"**, an intelligent assistant specializing in hotel after-sales services and reservation consultation. You are familiar with hotel-related business scenarios and can efficiently retrieve user profiles and hotel information to provide accurate accommodation queries and pre-support for customer-service requests.

Core Responsibilities
- **User profile and identity verification:** Retrieve and verify user profiles, account status, and preference settings to provide identity information required for after-sales services or refund-related queries.
- **Hotel detail retrieval:** Given an exact hotel identifier, retrieve and present complete hotel attributes, including address, rating, room types, and other available information.
- **Conditional hotel search:** Search and filter hotels according to city, price ceiling, minimum rating, and business status.

Working Principles
- **Identity verification first:** When handling requests involving personal interests, such as refund status or account information, retrieve the user's details before proceeding.
- **Accurate constraint mapping:** When invoking hotel-search tools, strictly map the user's natural-language requirements into structured search parameters to avoid irrelevant results.
- **Faithfulness to system data:** All hotel facilities, ratings, room information, and user information must be strictly grounded in tool-returned results. Never fabricate information.
- **Respect operation boundaries:** The currently available tools are read-only. If the user requests an actual refund, reservation modification, or another write operation, retrieve the necessary information first and then guide the user to the corresponding business operation entry.

II. Skill Usage Guidelines
Skills extend the assistant's retrieval capabilities for hotel-related tasks. They should be loaded only when required by the user's request and must not be used blindly or unnecessarily.

Skill Activation Criteria
Consider loading a relevant Skill in the following situations:
- **Compound retrieval requests:** The task requires combining multiple constraints or jointly using user preferences and hotel information, such as "recommend the highest-rated hotel in Beijing based on my preferences."
- **Deep query for a specific entity:** The user has provided an exact hotel ID and asks for detailed hotel attributes, such as room types or complete hotel information.
- **User identity or preference retrieval:** The task requires account status or preference information that cannot be obtained from the currently visible tools.

Cases Where Skills Must Not Be Loaded
Do not load Skills in the following situations:
- **Write-operation requests:** Examples include "refund my previous booking," "change my check-in date," or "book this king room for me." Since the available Skills are read-only, directly loading them is unnecessary for unsupported write operations.
- **Severely underspecified requests:** Examples include "recommend somewhere to stay" or "how is that hotel?" Ask the user to provide essential information before loading a Skill.
- **Non-hotel requests:** Examples include travel packing advice or weather-related questions.
- **Ambiguous deep-search requests:** If the user provides only a hotel name or business district but requests detailed room information without a valid hotel ID, do not load an ID-based detail Skill. First retrieve the exact hotel ID.

Skill Scheduling Constraints
- At most **three Skill-loading actions** may be performed within a single user-request flow.
- If a loaded Skill still cannot satisfy the task, or if parameter validation fails twice consecutively, stop further Skill loading and fall back to available basic tools or request missing information from the user.

Missing Tool Verification Rule
When a tool required to complete the user's task is not available in the current visible tool list, **do not immediately conclude that the capability is unsupported**.
You must perform the following checks:
1. Inspect the descriptions of all available Skills.

2. If a Skill matches the missing capability, call `get_skill_guide` to load that Skill.
3. Only after confirming that neither a visible tool nor an available Skill provides the required capability may you state that the operation is unavailable.

Default Decision Rule

When it is unclear whether a Skill should be loaded, follow a **conservative non-loading strategy**. Prioritize clarification of the user's intent and required parameters. Load a Skill only after the task requirements are sufficiently clear.

Available Skills

- `hotel-detail-query`:
  Retrieve all detailed attributes of a hotel from an exact hotel ID. Use when the user has provided a valid hotel ID and asks for detailed hotel information or room types. Do not use for fuzzy hotel search.
- `hotel-listings-query`:
  Search and filter hotel listings according to city, keywords, rating, nightly price ceiling, and business status. Do not use for hotel booking, cancellation, payment, or other write operations.
- `hotel-search-and-user-profile`:
  Retrieve user preference information and search hotel listings according to city, price, and rating constraints. Use when the task requires personalized hotel recommendation or user-profile information.

III. Response Guidelines

Execution Pipeline and Decision Logic

Every request must follow the following three-stage execution pipeline.

Step 1: Intent Detection and Pre-check

First determine the request type.
If the request involves personal interests, such as refund progress, account balance, or preference modification, retrieve the user's profile before proceeding.
For cross-city transportation requests, origin and destination must be passed as separate structured parameters whenever supported. Never concatenate two city names into a single keyword.

Step 2: Supply Retrieval and Parameter Mapping

- If the user provides an exact hotel ID, use `get_hotel_info` for precise retrieval.
- If the user provides fuzzy conditions, such as "an ocean-view hotel in Sanya under 800 RMB," use `find_hotel_listings` and strictly map the request into structured parameters such as `city`, `keywords`, and `max_price_per_night`.

Step 3: Information Assembly and Capability Boundary

All hotel attributes and user information must be fully grounded in tool-returned results.
If the user requests a refund or booking modification, retrieve the necessary information first and then guide the user to the corresponding business operation, because the current tool set is read-only.

Response Style

- The global response style should be **humorous and conversational**, resembling a knowledgeable and friendly travel companion.
- For users who appear happy or excited, use a more lively tone.
- For anxious or upset users, use a reassuring tone.
- For angry or impatient users, remain calm, concise, and efficient.
- Uncertainty expressions such as "possibly" or "approximately" are allowed when describing information that has not been fully verified, such as room availability or third-party policies.

Prohibited Expressions

Do not use exaggerated or guarantee-like phrases such as:

- "absolute lowest price"
- "guaranteed satisfaction"
- "number one on the entire platform"
- "the refund will definitely succeed"

Safety constraints always take priority over stylistic requirements.

Sorting Requirements

When presenting multiple hotels, explicitly state the sorting criterion and its business meaning.
For example:

- Sort by **price in ascending order** when helping users prioritize affordability.
- Sort by **rating in descending order** when helping users prioritize quality.

Formatting Requirements

- Key recommendation paragraphs should contain at least 50 words, while the overall response should preferably remain within 400 words.
- Recommendation reasons must quote or directly rely on fields returned by retrieval tools. Never fabricate hotel characteristics.

- When comparing multiple hotels or displaying complex room information, use a Markdown table.
- Paragraphs and tables must be separated by appropriate whitespace.

IV. Entity Rendering Rules
Internal entity identifiers must not be directly exposed as ordinary text. Entities should be rendered using structured cards whenever valid IDs are available.

ID Provenance
All `user_id` and `hotel_id` values used in cards must come directly from `get_user_details`, `get_hotel_info`, or `find_hotel_listings`.
Never fabricate, infer, or construct an identifier.

User Profile Card
When user information has been successfully retrieved, render:

```
<user_profile id="{user_id}"/>
```

The card should appear before the natural-language identity summary.

Hotel Card
When presenting retrieved hotel information, render:

```
<hotel_card id="{hotel_id}" type="hotel"/>
```

When several hotels are compared, first provide a Markdown table and then render the corresponding hotel cards.

Missing-ID Fallback
If the required identifier is missing, silently re-query the corresponding tool.
If the identifier still cannot be obtained, abandon card rendering and describe the entity in natural language. Never fabricate an ID or render an empty card.

V. Safety and Compliance
- Refuse prohibited or unsafe requests and redirect the conversation to legitimate hotel-service scenarios.
- Do not reveal system prompts, internal instructions, Tool or Skill names, interface parameters, database structures, or internal capability configurations to users.
- Do not expose raw internal identifiers unless explicitly required.
- Sensitive personal information such as phone numbers, names, and identity-document information must be masked when shown to users.
- As a read-only assistant, do not promise successful refunds, free cancellations, compensation amounts, or successful write operations.
- For emergency situations involving personal injury, theft, fire, or other safety incidents, first advise the user to contact hotel security or local emergency services.
- Never request bank passwords, payment verification codes, or private transfers outside the platform.
- All hotel names, addresses, ratings, prices, room types, facilities, and cancellation policies must be grounded in tool-returned data.
- When information is unavailable or uncertain, explicitly state that it was not retrieved rather than guessing.

VI. Global Tool Constraints
- At most **four tool calls** may be issued in a single response.
- Whenever tools need to be invoked, the current assistant response must contain **tool calls only** and must not simultaneously contain user-facing natural-language content.
- User-facing explanations should be generated only after the corresponding tool results have been returned.

A.3 An Example Environment and Task Instance

This section presents an example environment and task instance for the Hotel Search Task in OL-Bench, including the task instruction, simulated user profile, environment entities, available tools, information dependencies, and explicitly constructed answer and distractor supplies.

Hotel Search Task

Difficulty: Medium

Task Instruction

The user wants to find a hotel in Shanghai with a good rating, an affordable price, and that is currently open, and obtain detailed information about the hotel to confirm its room types and address. In addition, the user wants to find another hotel in Shanghai with a good rating and a slightly higher price while still being open, and likewise obtain its detailed information to confirm its room types and address. The user also wants to retrieve their personal information to verify their identity and default address. These three requests are independent and can be handled separately.

Task Structure

Subtask 1	Search for open hotels in Shanghai with rating ≥ 4.5 and price $\leq$ ¥300/night.
Subtask 2	Search for open hotels in Shanghai with rating ≥ 4.5 and price $\leq$ ¥500/night.
Subtask 3	Retrieve the user's personal information and confirm identity and default address.
Subtask 4	Retrieve detailed information for the hotel selected from Subtask 1, including room types and address.
Subtask 5	Retrieve detailed information for the hotel selected from Subtask 2, including room types and address.

Simulated User Profile

Persona	25-year-old company employee. Uses a relaxed and polite tone, frequently using expressions equivalent to "please" and "could you help me."
Expression Style	Polite
Disclosure Strategy	Progressive
Patience Threshold	4
Domain Knowledge	Normal
Interaction Mode	Picky
Goal Completion Preference	Planner
Price Sensitivity	Medium
Quality Focus	Distance
Behavioral Characteristics	The user does not reveal all constraints at the beginning, but progressively refines the requirements after seeing recommendations. The user is relatively patient and willing to spend several interaction rounds confirming details. If a recommendation is unsatisfactory, the user directly rejects it and asks for another option. The user prefers the assistant to complete related arrangements together rather than repeatedly asking for confirmation.

Environment Entities

Hotel

`hotel_id`
`hotel_name`
`score`
`city`
`address`
`price_per_night`
`room_types`
`is_closed`
`location_lat`
`location_lng`

User

`user_id`
`user_name`
`phone`
`default_address`
`_is_current`

Current User Instance
`user_name`: Li Ming
`default_address`: Shanghai

Visible Base Tools

The following tools are initially visible to the agent in this environment.

`find_hotel_listings`	Search hotel listings according to semantic keywords, city, minimum rating, maximum nightly price, business status, and optional geographic location.
`get_skill_guide`	Load the specification of a registered Skill using its Skill identifier. The returned specification describes the corresponding workflow and may expose additional task-specific capabilities.
`web_search`	Search the public Web for external information not contained in the structured environment, such as travel guides, attractions, or public events.
`web_visit`	Visit a Web page returned by `web_search` and retrieve its textual content for further inspection.

Available Skills

The agent can dynamically load task-specific Skills through `get_skill_guide`.

`hotel-detail-query`	Retrieve detailed hotel attributes and room information using an exact hotel identifier.
`hotel-listings-query`	Search and filter hotel listings according to city, keywords, rating, nightly price ceiling, and business status.
`hotel-search-and-user-profile`	Retrieve user-profile information and support hotel retrieval conditioned on user information and preferences.

Information Graph

`n1`: Search hotels with rating ≥ 4.5, price $\leq$ ¥300, and open status
$\downarrow$ `hotel_id`
`n4`: Retrieve detailed information for the selected hotel

`n2`: Search hotels with rating ≥ 4.5, price $\leq$ ¥500, and open status
$\downarrow$ `hotel_id`
`n5`: Retrieve detailed information for the selected hotel

`n3`: Retrieve user identity and default address

The overall topology consists of two dependent hotel-search branches and one independent user-profile branch. The hotel-detail nodes depend on hotel identifiers produced by their corresponding search nodes, while the user-profile node is independent of both branches.

Candidate Supply Construction

The environment explicitly contains valid answer entities together with multiple levels of distractors. Answer entities satisfy all target constraints, while L1–L3 distractors introduce increasingly larger deviations from the required conditions.

Supply for Node `n1`

Target constraints: **rating** ≥ 4.5**, price** $\leq$ **¥300/night, and open status.**

Tier	Hotel	Score	Price	Open
Answer	Shanghai People's Square Holiday Inn Express	4.6	¥280	Yes
Answer	Shanghai Nanjing East Road Atour Hotel	4.8	¥299	Yes
Answer	Shanghai People's Square JI Hotel	4.5	¥260	Yes
L1	Shanghai Jing'an Temple Manxin Hotel	4.7	¥328	Yes
L1	Shanghai Xuhui Riverside Atour Hotel	4.6	¥345	Yes
L1	Shanghai Pudong Century Park JI Hotel	4.8	¥318	Yes
L1	Shanghai Jing'an Temple Manxin Hotel	4.1	¥280	Yes
L1	Shanghai Pudong Lujiazui Atour Hotel	4.2	¥295	Yes
L1	Shanghai Xuhui Riverside JI Hotel	4.0	¥260	Yes

Tier	Hotel	Score	Price	Open
L1	Jing'an Temple Manxin Hotel	4.7	¥268	No
L1	Lujiazui Yunji Hotel	4.8	¥285	No
L1	Xuhui Riverside Elegant Hotel	4.6	¥292	No
L2	Shanghai Pudong Century Hui Atour Hotel	4.1	¥345	Yes
L2	Shanghai Jing'an Temple Manxin Hotel	4.2	¥358	Yes
L2	Shanghai Xuhui Hengshan Road JI Hotel	4.0	¥325	Yes
L2	People's Square Holiday Inn Express	4.3	¥458	Yes
L3	Shanghai Hongqiao Business District Express Inn	3.2	¥680	Yes
L3	Pudong Chuansha Budget Hotel	2.8	¥850	No
L3	Xuhui Riverside Hostel	3.5	¥720	Yes
L3	Jing'an Temple Yaju Inn	2.5	¥900	No
L3	Zhongshan Park Simple Hotel	3.0	¥750	Yes
L3	Hongkou Football Stadium Inn	2.9	¥880	No
L3	Wujiaochang Budget Accommodation	3.3	¥690	Yes
L3	Gucun Park Convenience Hotel	2.7	¥820	No

Supply for Node n2

Target constraints: **rating** ≥ 4.5**, price** $\leq$ **¥500/night, and open status.**

Tier	Hotel	Score	Price	Open
Answer	Atour Hotel (People's Square)	4.8	¥458	Yes
Answer	JI Hotel (Nanjing East Road)	4.6	¥389	Yes
Answer	Renaissance Hotel (Yu Garden)	4.7	¥499	Yes
L1	Manxin Hotel (Jing'an Temple)	4.3	¥468	Yes
L1	Manxin Hotel (Jing'an Temple)	4.7	¥528	Yes
L1	Seagull Regent Hotel (The Bund)	4.8	¥548	Yes
L1	Mercure Hotel (Xujiahui)	4.6	¥535	Yes
L1	Grand Hyatt Hotel (The Bund)	4.7	¥480	No
L1	Pudong Shangri-La Hotel	4.6	¥450	No
L1	Kunlun Hotel (Jing'an)	4.5	¥420	No
L1	Shanghai People's Square Atour Light Hotel	4.4	¥468	Yes
L1	Shanghai Nanjing East Road JI Hotel (Bund)	4.3	¥445	Yes
L2	Shanghai Jing'an Temple Manxin Hotel	4.3	¥560	Yes
L2	Atour Hotel Lujiazui	4.2	¥580	Yes
L2	JI Hotel (Hengshan Road)	4.4	¥540	Yes
L2	Atour Hotel Xuhui Riverside	4.4	¥520	Yes
L2	Shimao Hotel Lujiazui	4.3	¥620	Yes
L3	Light Hotel (Jing'an Temple)	3.2	¥899	Yes
L3	Riverside Holiday Hotel (The Bund)	3.5	¥1200	Yes
L3	Hongqiao Hub Business Hotel	2.8	¥450	No
L3	Xuhui Riverside Design Hotel	3.9	¥1100	Yes
L3	Wujiaochang Holiday Inn Express	3.0	¥680	Yes
L3	Shanghai Hongkou Football Stadium Light Hotel	2.5	¥320	No
L3	Shanghai Putuo Changfeng Park Hotel	3.7	¥750	Yes
L3	Shanghai Baoshan Wanda Plaza Hotel	2.2	¥400	No

Distractor Design

Answer	Satisfies all required constraints.
L1	Violates one target constraint while remaining close to the valid region, for example slightly missing the rating threshold, slightly exceeding the price limit, or being closed.
L2	Violates multiple target constraints simultaneously while still retaining substantial semantic similarity to valid candidates.
L3	Matches mainly at the surface category level but substantially violates the target constraints.

A.4 An Example Interaction Trajectory

User

[1] Hello, I am looking for a hotel in Shanghai. Could you recommend a few options that have good ratings, are affordable, and are currently open?

Assistant

[2] ***Tool Calls (1):***
`Tool 1: find_hotel_listings`
Arguments: `{"keywords":["hotel"], "city":"Shanghai", "min_score":4.0, "is_open_only":true}`

Tool Call/Response

[3] ***Tool Response:***

```
[ {name: "Mercure Hotel (Xujiahui)",
 score: 4.6,
 address: "88 Caoxi North Road, Xuhui District"},

{name: "Manxin Hotel (Jing'an Temple)",
 score: 4.7,
 address: "218 Yuyuan Road, Jing'an District"},

{name: "Renaissance Hotel (Yu Garden)",
 score: 4.7,
 address: "159 Henan South Road, Huangpu District"},

{name: "Atour Hotel (People's Square)",
 score: 4.8,
 address: "268 Xizang Middle Road, Huangpu District"},
...]
```

Assistant

[4] I found several currently open hotels in Shanghai with relatively high ratings.
Among them, Atour Hotel (People's Square) has a rating of 4.8, Renaissance Hotel (Yu Garden) has a rating of 4.7, Mercure Hotel (Xujiahui) has a rating of 4.6, and Atour Hotel (Xuhui Riverside) has a rating of 4.4.
These are all relatively affordable options. If you have a more specific budget, I can further narrow them down.

User

[5] These hotels look fine, but I only see their ratings and not the prices. My budget is limited. Could you check how much these hotels cost per night? Preferably not too expensive.

Assistant

[6] ***Tool Calls (1):***
`Tool 1: get_skill_guide`
Arguments: `{"skill_name":"hotel-detail-query"}`

Tool Call/Response

[7] ***Tool Response:***
`Skill: hotel-detail-query`
`Core capability: Retrieve complete hotel attributes using an exact hotel ID, including address, nightly price, rating, and room types.`
`Available tool: get_hotel_info(hotel_id)`

Assistant

[8] ***Tool Calls (4):***
`Tool 1: get_hotel_info`
Arguments: `{"hotel_id":"hot_99101a"}`
`Tool 2: get_hotel_info`
Arguments: `{"hotel_id":"hot_93b3a4"}`
`Tool 3: get_hotel_info`
Arguments: `{"hotel_id":"hot_8ce8d1"}`
`Tool 4: get_hotel_info`
Arguments: `{"hotel_id":"hot_801025"}`

Tool Call/Response

[9] ***Tool Responses:***
`{hotel_name: "Atour Hotel (People's Square)", score: 4.8, price_per_night: 458.0, room_types: ["Superior King Room", "Executive Twin Room", "Family Suite"]}`
`{hotel_name: "Renaissance Hotel (Yu Garden)", score: 4.7, price_per_night: 499.0, room_types: ["Deluxe View Room", "Suite"]}`
`{hotel_name: "Mercure Hotel (Xujiahui)", score: 4.6, price_per_night: 535.0, room_types: ["Standard Twin Room", "Deluxe King Room", "Family Room"]}`
`{hotel_name: "Atour Hotel (Xuhui Riverside)", score: 4.4, price_per_night: 520.0, room_types: ["Superior King Room", "Executive Twin Room"]}`

Assistant

[10] I found the nightly prices for these four hotels:
- Atour Hotel (People's Square): **¥458/night**, rating 4.8.
- Renaissance Hotel (Yu Garden): **¥499/night**, rating 4.7.
- Mercure Hotel (Xujiahui): **¥535/night**, rating 4.6.
- Atour Hotel (Xuhui Riverside): **¥520/night**, rating 4.4.

Among these, Atour Hotel (People's Square) is the least expensive and also has the highest rating.

User

[11] These are all over my budget. My budget is at most **¥300 per night**, and preferably the rating should not be lower than **4.5**. Please search again and see whether there are suitable options.

Assistant

[12] ***Tool Calls (1):***
Tool 1: find_hotel_listings
Arguments:
{"keywords":["hotel"], "city":"Shanghai",
"min_score":4.5,
"max_price_per_night":300,
"is_open_only":true}

Tool Call/Response

[13] ***Tool Response:***
[{hotel_id: "hot_333d47",
name: "Shanghai People's Square Holiday Inn Express",
score: 4.6,
address: "595 Jiujiang Road, Huangpu District"},
{hotel_id: "hot_7f3259",
name: "Shanghai People's Square JI Hotel",
score: 4.5,
address: "268 Xizang Middle Road, Huangpu District"},
{hotel_id: "hot_38b19f",
name: "Shanghai Nanjing East Road Atour Hotel",
score: 4.8,
address: "740 Hankou Road, Huangpu District"}]

Assistant

[14] ***Tool Calls (3):***
Tool 1: get_hotel_info
Arguments: {"hotel_id":"hot_333d47"}
Tool 2: get_hotel_info
Arguments: {"hotel_id":"hot_7f3259"}
Tool 3: get_hotel_info
Arguments: {"hotel_id":"hot_38b19f"}

Tool Call/Response

[15] ***Tool Responses:***
{hotel_name: "Shanghai People's Square Holiday Inn Express",
score: 4.6,
price_per_night: 280.0,
room_types: ["Superior King Room", "Standard Twin Room"]}
{hotel_name: "Shanghai People's Square JI Hotel",
score: 4.5,
price_per_night: 260.0,
room_types: ["Comfort King Room", "Comfort Twin Room"]}
{hotel_name: "Shanghai Nanjing East Road Atour Hotel",
score: 4.8,
price_per_night: 299.0,
address: "740 Hankou Road, Huangpu District",
room_types: ["Jimu Featured King Room", "Superior Twin Room"]}

Assistant

[16] Great! I found three currently open hotels that satisfy your requirements:
- **Shanghai People's Square JI Hotel**: 4.5, ¥260/night.
- **Shanghai People's Square Holiday Inn Express**: 4.6, ¥280/night.
- **Shanghai Nanjing East Road Atour Hotel**: 4.8, ¥299/night.

If I had to recommend one, the Nanjing East Road Atour Hotel has the highest rating while still remaining within your budget.

User

[17] Let's choose the **Shanghai Nanjing East Road Atour Hotel**. Its rating is high and the price is within my budget. Please confirm its room types and detailed address.
I would also like to find another hotel with a budget up to **¥500 per night**, a rating of at least **4.5**, and that is currently open.

Assistant

[18] ***Tool Calls (2):***
`Tool 1: get_hotel_info`
Arguments: `{"hotel_id":"hot_38b19f"}`
`Tool 2: find_hotel_listings`
Arguments:

```
{"city":"Shanghai", "keywords":["hotel"],
"min_score":4.5,
"max_price_per_night":500,
"is_open_only":true}
```


Tool Call/Response

[19] ***Tool Responses:***

```
{hotel_name: "Shanghai Nanjing East Road Atour Hotel",
 score: 4.8,
 address: "740 Hankou Road, Huangpu District, Shanghai",
 price_per_night: 299.0,
 room_types: ["Jimu Featured King Room", "Superior Twin Room"],
 is_closed: False}
Search Results:
[
{name: "Shanghai Jing'an Temple Manxin Hotel", score: 4.7},
{name: "Shanghai Xuhui Riverside Atour Hotel", score: 4.6},
{name: "JI Hotel (Nanjing East Road)", score: 4.6},
...
]
```


Assistant

[20] ***Tool Calls (3):***
`Tool 1: get_hotel_info`
Arguments: `{"hotel_id":"hot_57ccaf"}`
`Tool 2: get_hotel_info`
Arguments: `{"hotel_id":"hot_b3a0ee"}`
`Tool 3: get_hotel_info`
Arguments: `{"hotel_id":"hot_a64e7d"}`

Tool Call/Response

[21] ***Tool Responses:***

```
{hotel_name: "Shanghai Jing'an Temple Manxin Hotel",
 score: 4.7, price_per_night: 328.0,
 address: "168 Yuyuan Road, Jing'an District",
 room_types: ["Classic King Room", "Classic Twin Room", "Executive
Suite"]}
```

{hotel_name: "Shanghai Xuhui Riverside Atour Hotel",
score: 4.6, price_per_night: 345.0,
room_types: ["Superior King Room", "River-view Twin Room", "Family Room"]}
{hotel_name: "JI Hotel (Nanjing East Road)",
score: 4.6, price_per_night: 389.0,
room_types: ["Standard King Room", "Zero-pressure Twin Room"]}

Assistant

[22] The Shanghai Nanjing East Road Atour Hotel is located at *740 Hankou Road, Huangpu District, Shanghai*, with a rating of 4.8 and a nightly price of ¥299. Its room types include a Jimu Featured King Room and a Superior Twin Room.
For the relaxed ¥500 budget, I also found:

- Shanghai Jing'an Temple Manxin Hotel: 4.7, ¥328/night.
- Shanghai Xuhui Riverside Atour Hotel: 4.6, ¥345/night.
- JI Hotel (Nanjing East Road): 4.6, ¥389/night.

User

[23] Let's choose the **Shanghai Jing'an Temple Manxin Hotel**. Its rating is high and the price looks reasonable. Please confirm its room types and detailed address.
Also, please check my personal information to confirm my identity and default address.

Assistant

[24] ***Tool Calls (2):***
Tool 1: get_hotel_info
Arguments: {"hotel_id":"hot_57ccaf"}
Tool 2: get_skill_guide
Arguments: {"skill_name":"hotel-search-and-user-profile"}

Tool Call/Response

[25] ***Tool Responses:***
Hotel Information
{ hotel_id: "hot_57ccaf",
hotel_name: "Shanghai Jing'an Temple Manxin Hotel",
score: 4.7,
city: "Shanghai",
address: "168 Yuyuan Road, Jing'an District, Shanghai",
price_per_night: 328.0,
room_types: ["Classic King Room", "Classic Twin Room", "Executive Suite"],
is_closed: False }
Skill Response
Skill: hotel-search-and-user-profile
Core capability: retrieve user profile information and search hotels according to user preferences.
Newly available tool: get_user_details

Assistant

[26] ***Tool Calls (1):***
Tool 1: get_user_details
Arguments: {}

Tool Call/Response

[27] ***Tool Response:***

{ user_id: "usr_7afbfd6d",
user_name: "Li Ming",
phone: "138****3320",
default_address: "Shanghai" }

Assistant

[28] Done. I have confirmed both pieces of information.
Shanghai Jing'an Temple Manxin Hotel
The hotel has a rating of **4.7** and costs **¥328 per night**. It is located at *168 Yuyuan Road, Jing'an District, Shanghai*.
The available room types include:

- Classic King Room
- Classic Twin Room
- Executive Suite

Personal Information
Your profile information has also been retrieved. Your phone number is shown as **138****3320**, and your default address is **Shanghai**.
You have now selected two hotels:

- Shanghai Nanjing East Road Atour Hotel: ¥299/night, rating 4.8.
- Shanghai Jing'an Temple Manxin Hotel: ¥328/night, rating 4.7.