RecToolBench: Benchmarking Recommendation-Specific Tool Orchestration under Fuzzy User Intent

Xiao Chen[1†*], **Yicheng Zhao**[2*], **Yingying Wu**[2], **Zhendong Chu**[3‡]
Changyi Ma[2‡], **Qingsong Wen**[3], **Xuan Song**[2]
[1]The Hong Kong Polytechnic University [2]Jilin University [3]Squirrel AI Learning
{zc9uy@virginia.edu,changyima@jlu.edu.cn}

Abstract

Recent advances in agentic recommender systems are shifting recommender systems from passive filtering engines to instruction-following agents that use external tools to resolve user intent. However, existing benchmarks often assume explicit user intent, simplified tool environments, or isolated function calls, leaving realistic tool orchestration for recommendation underexplored. To bridge this gap, we propose **RecToolBench**, a Model Context Protocol (MCP)-based benchmark for evaluating tool-using recommender agents under fuzzy user instructions. RecToolBench contains more than 1,200 executable tasks across three recommendation domains, 13 MCP servers, and 32 tools, spanning single-tool calls, parallel tool calls, sequential tool chains, and hybrid tool orchestration. We construct RecToolBench with a scalable *synthesize–fuzzify–judge* pipeline that generates executable fuzzy recommendation tasks, and evaluates agent trajectories using rule-based execution checks and rubric-based LLM evaluation. Experiments on representative LLMs show that syntactically valid tool calls do not guarantee successful recommendations. Models struggle with semantic parameter grounding, multi-step evidence integration, and grounded final recommendations, especially as orchestration complexity increases. Our results identify tool orchestration under fuzzy user intent as a major bottleneck for agentic recommender systems. Our data and code are available at https://github.com/ShawnChenn/RecToolBench.

1 Introduction

Agentic capabilities are reshaping recommender systems (RS) from *passive* filtering engines into *proactive*, instruction-following agents (Huang et al., 2025a; Maragheh and Deldjoo, 2025). A key paradigm in this transition is *tool-using recommender agents* (Zhao et al., 2024; Tang et al., 2026), which equip LLMs with external tools to iteratively clarify user intent and ground recommendations in retrieved evidence. Such capabilities are particularly well-suited to practical settings such as personalized e-commerce (Zhang et al., 2026; Chen et al., 2025b), where user requests are often open-ended and highly context-dependent.

Benchmark	# Tools	End-to-end Performance	Fuzzy Instruction	Graded Difficulty
Shopping MMLU (Jin et al., 2024)	-	×	×	×
ChineseEcomQA (Chen et al., 2025a)	-	×	×	×
τ-bench (Yao et al., 2024)	28	✓	×	×
ShoppingBench (Wang et al., 2026)	6	✓	×	○
Ours	32	✓	✓	✓

Table 1: Comparison with existing agent benchmarks that include recommendation scenarios. ✓: Fully addressed; ○: Partially addressed; ×: Not addressed.

Despite this progress, existing benchmarks remain limited in evaluating recommender agents under realistic tool-orchestration scenarios. General tool-use benchmarks typically assume explicit, well-specified instructions (Yao et al., 2024; Zhou et al., 2024), while recommendation-oriented benchmarks often focus on narrow settings (e.g., e-commerce QA) or isolated function calls (Jin et al., 2024; Chen et al., 2025a). As shown in Table 1, these works largely overlook a common real-world challenge in recommendations: user instructions are often **fuzzy** and **underspecified**. In such cases, agents must infer latent intent, coordinate appropriate tools, and compose tool outputs into grounded recommendations, which means that tool orchestration in recommendations requires more than syntactically valid tool calls.

Building a realistic benchmark for tool-using recommender agents requires addressing three key gaps. The first is *fuzzy intent resolution*. Real user instructions rarely specify the structured pa-

*Equal contribution. †Project leader. ‡Corresponding authors.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 23958–23983
October 24-29, 2026 ©2026 Association for Computational Linguistics

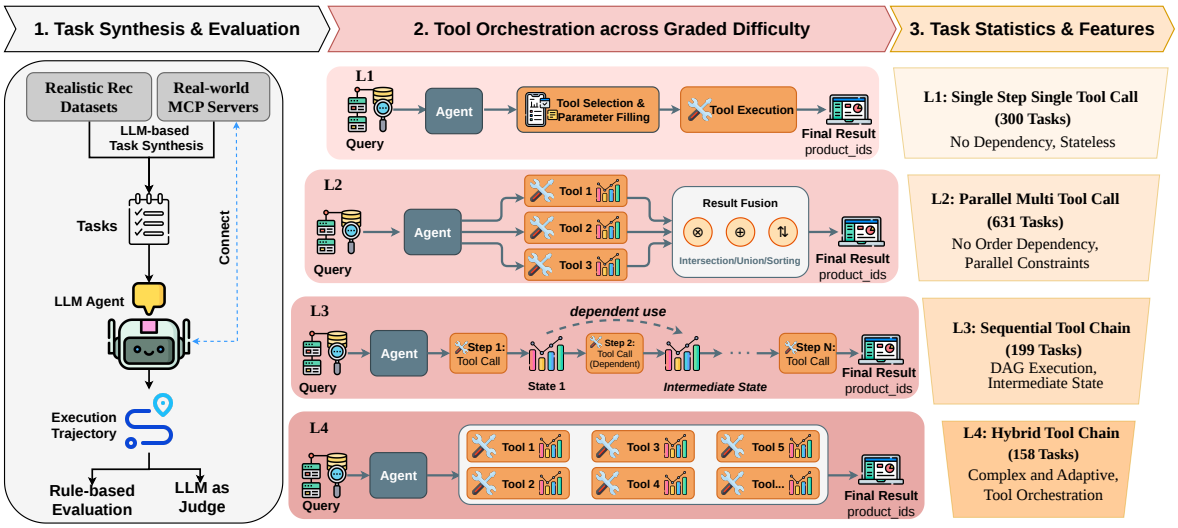


Figure 1: The overall data synthesis and agent evaluation pipeline, featuring a progressive four-level difficulty scheme. Tailored to address inherently fuzzy user instructions, the task complexity escalates from easy to hard: **L1** requires a single tool call; **L2** involves parallel tool invocations; **L3** demands serial dependency chains; and **L4** entails hybrid dependencies and tool orchestration.

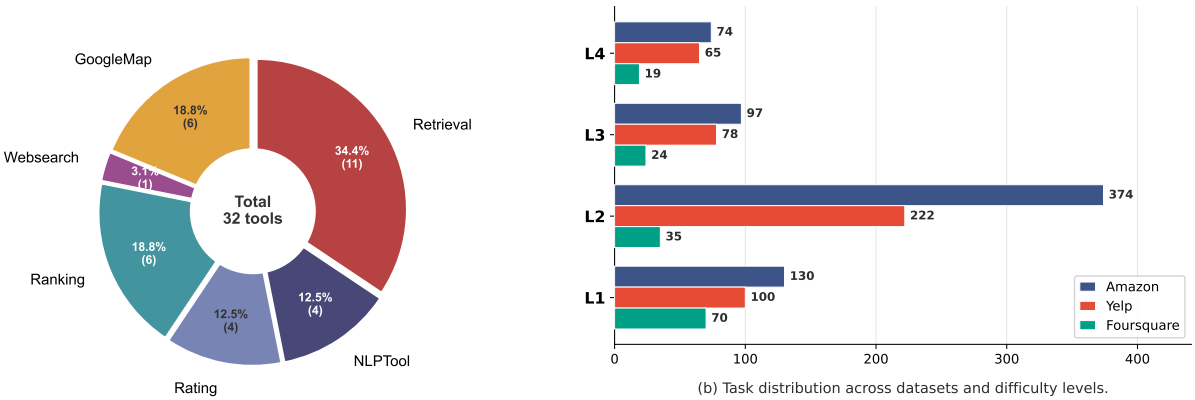
rameters (e.g., category, budget, time window) that downstream tools require, forcing agents to infer latent intent rather than execute pre-formed queries. The second is *compositional tool orchestration*. Recommendation tasks typically require parallel, serial, or cross-server tool dependencies, where intermediate outputs must be semantically propagated across steps, going well beyond isolated function calls. The third is the *syntactic–semantic evaluation gap*. A successfully executed tool call does not imply correct orchestration, since agents often produce syntactically valid invocations whose parameters are semantically misaligned with the user's fuzzy intent, making execution-only metrics systematically over-optimistic.

To bridge these gaps, we propose **RecToolBench**, the first benchmark dedicated to evaluating tool orchestration in recommender agents under fuzzy user instructions. Built on the *Model Context Protocol* (MCP) (Hou et al., 2025), RecToolBench provides standardized tool invocation schema across servers. Using a scalable LLM-driven "synthesize-fuzzify-judge" pipeline, we construct over 1,200 executable tasks spanning three recommendation domains and 13 MCP servers. Each server encapsulates a cohesive suite of complementary tools (32 in total). As illustrated in Figure 1, tasks are organized into a progressive four-level difficulty scheme that increases intent ambiguity and compositional dependency depth, ranging from single-tool calls to hybrid dependencies and orchestration.

Moreover, we conduct a comprehensive empirical study on Small Language Models (SLMs) and proprietary models. Each task is executed through single- or multi-turn MCP interactions and evaluated using rule-based checks for execution correctness and task completion, together with rubric-based LLM-as-a-Judge scoring for grounding and planning quality. Our evaluation reveals several key findings. First, *SLMs struggle with correct tool use*: in L1, SLMs often produce valid calls but fail to select the correct tool or parameters. Second, *orchestration failures become more severe as task complexity increases*: in L2 and L3, models all struggle with parameter grounding, parallel decomposition, and dependency tracking, even when their schema compliance remains high. Third, *hybrid tool orchestration exposes a persistent execution–outcome gap*: in L4, models achieve strong rule-based scores, but their Hit Ratios and grounding scores remain much lower. These findings show that agentic recommendation is bottlenecked by semantic parameter grounding, multi-step evidence integration, and grounded final recommendation.

Our contributions are summarized as follows:

- We introduce the **first MCP-based benchmark for fuzzy recommendation tool orchestration**, covering diverse workflows, compositional dependencies, and graded difficulty levels.
- We develop a **scalable task synthesis**



(a) Category distribution of MCP servers.

(b) Task distribution across datasets and difficulty levels.

Figure 2: Overview of MCP server and task distribution in the RecToolBench.

pipeline that generates executable fuzzy recommendation tasks grounded in real tool schemas, enabling agent trajectories to be evaluated through rule-based metrics and rubric-based LLM judging.

- We provide a **comprehensive empirical study** of SLMs and proprietary LLMs on these tasks. Our analysis shows that agent failures evolve with task complexity, from basic tool and parameter selection errors in SLMs to broader weaknesses in information grounding and dependency-aware orchestration in multi-step and hybrid tasks.

2 Benchmark Construction

2.1 Problem Formulation

We formalize each benchmark task as a tool-augmented partially observable Markov decision process (POMDP) (Yao et al., 2024). Given n MCP servers $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$, the available tool set is $\mathcal{T} = \bigcup_{j=1}^{n} \mathcal{T}_j$, where $\mathcal{T}_j$ is the set of tools exposed by server σ_j. Each task is represented as $\tau = (q, \mathcal{T}, \mathcal{C}, \mathcal{G})$, where q is an underspecified user instruction, $\mathcal{C} = \{c_i\}_{i=1}^{N}$ is the candidate set, and $\mathcal{G}$ serves as the ground truth for both recommendation results and tool use. Specifically, $\mathcal{G} = (c^\star, \mathcal{A}^\star)$, where $c^\star \in \mathcal{C}$ is the target item and $\mathcal{A}^\star$ denotes the expected tool calls.

The agent receives $(q, \mathcal{T})$ and interacts within a POMDP tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{U})$, denoting latent states $\mathcal{S}$, actions $\mathcal{A}$, observations $\mathcal{O}$, transition and observation models $\mathcal{P}$, rewards $\mathcal{R}$, and instructions $\mathcal{U}$. The action space consists of planning, tool invocation, and final recommendation: $\mathcal{A} = \mathcal{A}_{\text{plan}} \cup \mathcal{A}_{\text{tool}} \cup \mathcal{A}_{\text{final}}$. A tool action is formatted as $a_{\text{tool}} = \langle \sigma_j, t_k, \mathbf{p}_k \rangle$, representing the invocation of tool $t_k \in \mathcal{T}_j$ from server $\sigma_j \in \Sigma$ using arguments $\mathbf{p}_k$. The observation space $\mathcal{O} = \mathcal{O}_{\text{tool}} \cup \mathcal{O}_{\text{state}}$ combines external tool outputs ($\mathcal{O}_{\text{tool}}$) and state feedback ($\mathcal{O}_{\text{state}}$). At step t, the agent selects an action $a_t \in \mathcal{A}$ conditioned on q, $\mathcal{T}$, and past observations $o_{<t}$. The environment subsequently yields a new observation according to $\mathcal{P}$, continuing until a final recommendation is generated or the interaction budget is exhausted.

2.2 Task Synthesis

A challenge in benchmark construction is generating realistic, naturally phrased tasks, which are grounded in executable tool-use workflows across MCP servers. To this end, we ground task generation in user interaction sequences from three real-world datasets: Amazon Electronics[1], Yelp 2019[2], and Foursquare (Noulas et al., 2011), corresponding to different recommendation scenarios.

MCP Server Design. RecToolBench provides 13 domain-specific MCP servers across three recommendation domains, comprising 32 executable tools. These tools cover six core capabilities: retrieval, review and sentiment analysis, rating-based filtering, personalized recommendation, geospatial reasoning, and web search. Built on real-world recommendation datasets, they enable controlled yet realistic evaluation of tool-use recommender agents. Details of MCP servers are in Figure 2a and Appendix C.

Synthesis Pipeline. Accordingly, we propose *AmbiFlow Synthesis*, an execution-aware and

[1] http://jmcauley.ucsd.edu/data/amazon
[2] https://www.yelp.com/dataset/challenge

ambiguity-controlled pipeline for constructing realistic recommendation-oriented tool-use tasks. Given real-world user interaction sequences, AmbiFlow first identifies feasible tool-use workflows based on MCP tool schemas and user-item contexts. It then prompts a task synthesis LLM, i.e., GLM-5.1, with difficulty-level-specific requirements to generate explicit natural-language instructions along with the corresponding ground-truth tool sets. Next, AmbiFlow transforms each explicit instruction into a fuzzy one through a controlled fuzzification process, which is guided by five common ambiguity types: **underspecification**, **soft contradiction**, **vague quantifiers**, **preference uncertainty**, and **implicit constraints**. This process removes explicit tool names and procedural hints while introducing realistic uncertainty, yet preserves the underlying user intent. A final quality-filtering stage verifies the solvability, structural validity, and ambiguity reasonableness of each task. We further conduct human validation to assess instruction realism, workflow executability, and dependency correctness. We report these annotation details in Appendix D.1

Algorithm 1: Multi-turn Planning and Tool Execution for Benchmark Evaluation

Input: User instruction u, tool set $\mathcal{T}$, max rounds $T_{\max}$, token budget B
Output: Final recommendation $\hat{y}$, execution trajectory H

// Initialize S_p: planning state; $\mathcal{B}$: candidate bus; H: execution trajectory
$S_p \leftarrow \emptyset$; $\mathcal{B} \leftarrow \emptyset$; $H \leftarrow \emptyset$
for $t = 1$ **to** $T_{\max}$ **do**
  // Build planner prompt
  $z_t \leftarrow \textsc{RenderState}(S_p) \oplus \textsc{RenderBus}(\mathcal{B})$;
  // Generate current turn's tool plan
  $(\texttt{cont}_t, A_t) \leftarrow \pi_{\text{plan}}(u, z_t, \mathcal{T})$;
  if $\neg\,\texttt{cont}_t$ ***or*** $A_t = \emptyset$ **then break**
  // Parallel or serial tool execution
  $O_t^{\text{raw}} \leftarrow \pi_{\text{exec}}(A_t)$
  // Extract Item IDs from raw observation
  $O_t \leftarrow \texttt{Slim}(O_t^{\text{raw}})$
  // Update candidate bus status
  $\mathcal{B} \leftarrow \mathcal{B} \cup \textsc{ExtractBus}(O_t^{\text{raw}}, t)$
  // Update agent internal states
  $S_p \leftarrow \texttt{UpdateState}(S_p,\ A_t,\ O_t)$
  // LLM-based state compression
  if $|S_p| > B$ **then**
    $S_p \leftarrow \texttt{Compress}(S_p)$
  $H \leftarrow H \cup \{(A_t, O_t)\}$
$\hat{y} \leftarrow \texttt{Synthesize}(u,\ H)$
// LLM synthesizes final answer from trajectory
return $\pi_{\textit{final}}(u, H)$, H

Dataset Statistics The benchmark contains 1,288 synthesized tasks with grounded user intents and progressively increasing compositional complexity. Specifically, it includes 300 Level-1 tasks and 988 compositional tasks from Levels 2–4. Across datasets, the benchmark covers 675 Amazon, 465 Yelp, and 148 Foursquare tasks. Detailed statistics are provided in Fig 2b. We also present fuzzy query examples in Table 2.

3 Evaluation Protocol and Metrics

3.1 Evaluation Protocol

We evaluate agents using the multi-round plan–execute loop in Algorithm 1. Each rollout maintains three complementary states: a compact planning state S_p used in the planner prompt, a Candidate Bus $\mathcal{B}$ that stores candidate items and their metadata outside the prompt to reduce context load, and the full trajectory H used for final synthesis and evaluation. At each round, the planner conditions on the user instruction, tool schemas, the compressed planning history, and the candidate summary maintained in $\mathcal{B}$. It then either predicts the next tool plan A_t or terminates the rollout.

Tool outputs are stored in two views. Raw outputs O_t^{raw} keep complete tool outputs, while planning outputs O_t^{plan} retain only compact fields needed for subsequent planning, such as IDs. The Candidate Bus is updated from raw outputs to preserve reusable intermediate items without adding full payloads to the planner context. If the planner exceeds the context budget or returns an empty response, S_p is compressed while preserving tool names, round order, key parameters, and critical findings. This design keeps long execution traces available for evaluation while maintaining a compact context for multi-round planning.

3.2 Evaluation Metrics

Given an agent trajectory $\hat{\mathcal{G}} = \{(\hat{t}_j, \hat{\mathbf{p}}_j)\}_{j=1}^{n}$ and the available gold annotations, we evaluate tool use using two types of metrics. Rule-based evaluation focuses on the robustness of tool usage, while LLM-as-a-judge evaluation measures strategic quality with structured rubrics and prompt shuffling to mitigate positional bias.

Rule-based evaluation. We first check whether each executed call is valid and executable: *Tool Name Validity* verifies that the predicted tool exists in the available tool set; *Schema Compliance* checks whether its arguments satisfy the API schema; and *Execution Success* requires the call to

Level	Fuzzy User Query	Tool-Use Workflow	Ambiguity Types
L1	"I'm trying to find some camera stuff—maybe a hot shoe cover or bubble level for my Sony? I remember a brand called Foto&Tech or something similar."	`filter_items_by_attributes`	Underspecification
L2 (parallel)	"I've got [candidate item asins] saved but can't remember which one I wanted. I think around 4 stars? Not super strict. Budget maybe under $30 or $40?"	`filter_products` `filter_items_by_attributes` `query_sentiment`	Contradiction Underspecification
L3 (serial)	"Find accessories for [user_id]. Check their history first, I think it was a soundbar or speaker, maybe a keyboard? Get the details, then filter by that brand and price range."	`get_user_history` → `get_item_details` → `filter_items_by_attributes`	Implicit Constraints Underspecification
L4 (hybrid)	"I bought [item asin] and want accessory recommendations, peripherals maybe, not 100% sure. Search co-purchases and keywords at the same time. Then check reviews and sentiment. Finally get details on top picks."	`item2item_search` ‖ `search_keyword` → `search_reviews` ‖ `query_sentiment` → `get_item_details`	Preference Uncertainty Underspecification Contradiction

Table 2: Representative task examples from each difficulty level (Amazon Electronics). Examples L2–L4 are drawn from a query set with relatively high ambiguity.

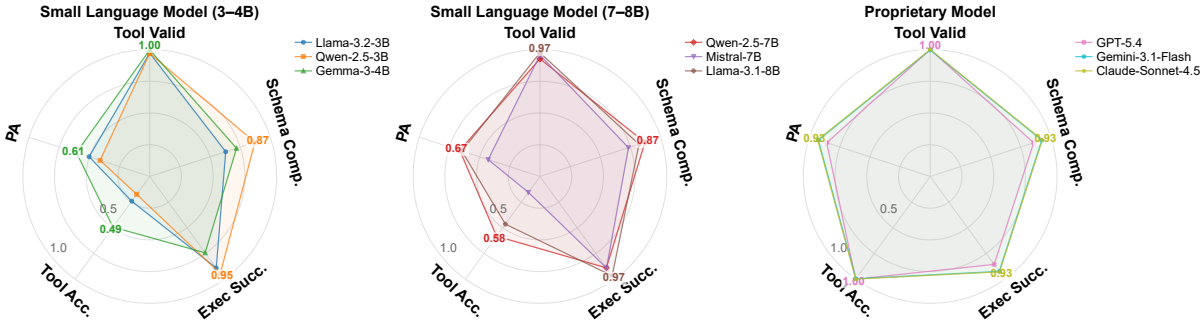


Figure 3: Evaluation results of representative LLM agents on atomic tool using abilities (L1).

finish without error and return a non-empty result.

Then, we compare predicted tool usage with gold annotations. Single-tool tasks are evaluated using *Tool Selection Accuracy*, while multi-tool tasks use *Tool Selection F1* over predicted and gold tool sets. We further report orchestration and outcome metrics: *Parallel Ratio* measures whether independent calls are issued early, *Efficiency* penalizes redundant calls, and *Hit Ratio* indicates whether the target item appears in the results.

LLM-as-a-Judge Evaluation. For L1 and L2, we report a reference-free *Parameter Appropriateness (PA)* score, where an independent judge (DeepSeek-V3.2) assesses whether the generated parameters are semantically appropriate for the fuzzy instruction. For longer serial and hybrid trajectories (L3-L4), exact matching cannot fully capture execution quality, as agents may select plausible tools while violating dependencies, unnecessarily serializing independent calls, or producing unsupported recommendations. We therefore evaluate the full execution trace using an LLM judge. Serial workflows are assessed by *Grounding* and *Dependency Awareness*, while hybrid workflows are evaluated by *Planning Coherence*, *Multi-round Efficiency*, *Parameter Appropriateness*, and *Information Grounding*. Each dimension is scored from 1 to 10, averaged, and normalized to $[0, 1]$. Detailed definitions and evaluation prompts are provided in Appendix E.

Remark. For L1–L3, we provide ground-truth workflows as evaluation references. **L1** contains a single annotated tool call. **L2** provides an unordered set of tools that can be invoked in parallel

within one round. Thus, all tool-set metrics are permutation-invariant. **L3** provides the efficient sequential trajectory, explicitly capturing the required dependencies among tool calls. For **L4**, we do not provide a reference workflow because hybrid tasks may admit multiple valid combinations of parallel and sequential execution.

4 Experiment Results

Baselines: We evaluate nine representative LLMs, including six small language models (SLMs)[3]: Llama-3.2-3b-it, Qwen-2.5-3b-it, Gemma-3-4b-it, Qwen-2.5-7b-it, Mistral-v0.3-7b-it, and Llama-3.1-8b-it; and three proprietary models: GPT-5.4, Gemini-3.1-Flash, and Claude-Sonnet-4.5.

Implementation Details: To reduce order sensitivity in LLM-based judging (Li et al., 2026), we apply prompt shuffling and score averaging. For each task, we randomly permute the evaluation axes and sub-dimensions within the scoring rubric, while keeping the rubric content unchanged. We run five shuffled evaluations per task, normalize each sub-dimension score to $[0, 1]$, compute axis-level averages, and take the mean across runs as the final score. Results are reported separately for each difficulty level (L1–L4), averaged over all tasks within that level, to isolate the effect of increasing orchestration complexity on model performance.

4.1 Main Results

4.1.1 Atomic Tool Using Abilities (L1)

We first report the L1 single-tool evaluation results in Figure 3. We have three key observations:

I. Proprietary models perform better in functional tool use. Proprietary models achieve 100% Tool Selection Accuracy, with near-perfect schema compliance and execution success. In contrast, the best SLM, Qwen-2.5-7B-it, reaches 58% Tool Selection Accuracy and 45% Parameter Accuracy, despite maintaining high format compliance. This gap indicates that current SLMs remain unreliable for tool use, even in the simplest single-tool setting.

II. Format validity does not imply semantic correctness. SLMs often produce structurally valid tool calls, while failing at functional correctness. For example, Gemma-3-4B obtains 99% Tool Name Valid Rate but only 49% Tool Selection Accuracy, while Qwen-2.5-3B achieves 95% Execution Success Rate but only 10% Parameter Accuracy. This indicates that format compliance alone is insufficient for semantically correct tool use.

III. Scaling up model size alone is not sufficient. Within the Qwen-2.5 family, the 7B variant significantly outperforms the 3B variant in Parameter Accuracy. However, cross-family comparisons reveal that this relationship is not monotonic: Llama-3.1-8B and Gemma-3-4B achieve comparable Tool Selection Accuracy (46% vs. 49%). This suggests that instruction-tuning quality and tool-use alignment also play a critical role in recommendation-oriented tool use.

IV. Reference-free PA reveals weakness in parameter grounding. To mitigate the single-reference concern for parameter evaluation, we report reference-free *Parameter Appropriateness (PA)* on L1, judged by DeepSeek-V3.2 from the fuzzy instruction. Figure 3 shows that even when tool selection is correct, weaker models still exhibit substantially lower PA.

4.1.2 Parallel Tool-Use Abilities (L2)

We report the L2 parallel tool-use results in Table 3. The results reveal several findings:

I. SLMs improve on schema-level execution but still lag in task completion. In L2, several SLMs are competitive in schema compliance and parallel planning, yet proprietary models achieve higher Hit Ratio. The gap mainly comes from tool-call effectiveness: Table 6 shows that some SLMs issue as many, or even more, calls per task than proprietary models, but many of these calls are repetitive or use incorrect parameter values, often yielding empty results. This also explains the divergence between Schema Compliance and Execution Success Rate. Proprietary models can achieve higher execution success than schema compliance because MCP servers often tolerate minor deviations.

II. Tool Selection F1 reflects different tool-use strategies. Proprietary models invoke more unique tools than the ground-truth plan, often adding auxiliary tools (e.g., `get_item_details`) to gather richer evidence. This lowers Tool Selection F1 but improves final recommendation result synthesis. In contrast, most SLMs call fewer tools and stay closer to the expected tool set, yielding higher F1 but lower task completion.

III. Parameter grounding is the central bottleneck under fuzzy intent (confirmed by reference-free PA). Across all models, rule-based Parameter Accuracy remains below 24%, far below proprietary models' L1 performance, suggesting that models struggle to

[3] Language Models with fewer than 10B parameters following (Belcak et al., 2025)

Model	Schema Understanding		Tool Usage		Planning		Completion
	Schema Compliance Rate	Execution Success Rate	Tool Selection F1 Score	Parameter Appropriateness	Parallel Rate	Efficiency Score	Hit Ratio @All
Small Language Models							
Llama-3.2-3b-instruct	97.77%	95.14%	41.35%	36.60%	33.57%	9.97%	17.54%
Qwen-2.5-3b-instruct	93.96%	90.12%	49.20%	55.20%	42.20%	45.38%	22.81%
Gemma-3-4b-instruct	94.34%	88.83%	42.27%	35.70%	34.69%	23.49%	10.53%
Qwen-2.5-7b-instruct	98.56%	96.45%	52.66%	68.50%	44.37%	30.17%	31.58%
Mistral-v0.3-7b-instruct	90.09%	85.67%	45.52%	44.20%	40.39%	17.82%	17.54%
Llama-3.1-8b-instruct	99.79%	98.41%	44.16%	58.80%	40.33%	10.74%	28.07%
Proprietary Models							
GPT-5.4	94.46%	100.00%	43.55%	93.00%	36.67%	34.07%	43.78%
Gemini-3.1-Flash	86.66%	100.00%	37.09%	85.90%	32.94%	16.81%	55.59%
Claude-Sonnet-4.5	96.58%	98.56%	42.42%	82.10%	39.37%	30.41%	47.29%

Table 3: Performance Results with Synthesized User Instructions — Parallel Tool Using Abilities (L2). The highest and second-highest scores are highlighted in green and blue, respectively.

Model	Rule-based					Rubric-based		Overall Score
	Schema Understanding		Tool Usage	Planning	Completion	Planning	Grounding	
	Schema Compliance Rate	Execution Success Rate	Tool Selection F1 Score	Efficiency Score	Hit Ratio	Dependency Awareness	Information Grounding	
Small Language Models								
Llama-3.2-3b-instruct	97.08%	86.66%	44.00%	67.24%	54.27%	0.200	0.162	0.461
Qwen-2.5-3b-instruct	96.62%	82.08%	53.34%	67.44%	52.76%	0.253	0.286	0.480
Gemma-3-4b-instruct	89.20%	35.95%	37.18%	41.75%	21.11%	0.341	0.473	0.289
Qwen-2.5-7b-instruct	96.96%	87.96%	60.18%	69.75%	52.76%	0.309	0.293	0.553
Mistral-v0.3-7b-instruct	92.69%	77.41%	53.07%	86.47%	65.83%	0.203	0.207	0.564
Llama-3.1-8b-instruct	97.01%	92.15%	57.14%	75.37%	59.30%	0.268	0.224	0.576
Proprietary Models								
GPT-5.4	93.65%	98.47%	50.43%	93.67%	96.67%	0.458	0.615	0.744
Gemini-3.1-Flash-Lite	88.27%	99.82%	44.77%	73.15%	93.33%	0.275	0.332	0.644
Claude-Sonnet-4.5	91.01%	99.95%	52.10%	98.03%	95.00%	0.437	0.548	0.806

Table 4: Performance Results with Synthesized User Instructions — Sequential Tool Using Abilities (L3). The highest and second-highest scores are highlighted in green and blue.

ground correct arguments from fuzzy user queries when multiple calls must be coordinated in parallel. To mitigate the generator–evaluator alignment and single-reference concerns, we additionally evaluate **reference-free Parameter Appropriateness (PA)** on L2 using DeepSeek-V3.2 as an independent judge (no access to the generated reference). The resulting model ordering is largely consistent with P_{acc} (Spearman $\rho = 0.88$), indicating that parameter grounding remains a robust bottleneck across evaluation protocols.

IV. High parallelism does not necessarily imply high efficiency. Several SLMs achieve high Parallel Rate yet low Efficiency Score, indicating that their parallel calls are frequently redundant rather than strategically decomposed. While efficient parallel tool use requires coordinated, task-relevant calls, rather than merely increasing the number of simultaneous tool invocations.

4.1.3 Sequential Tool-Use Abilities (L3)

We report the results on sequential tool chain tasks in Table 4. Key observations are as follows:

I. Sequential dependencies expose a clear gap between schema-level execution and true task completion. Several SLMs still achieve high Schema Compliance and Execution Success in L3. For example, Llama-3.1-8B reaches 97.01% Schema Compliance and 92.15% Execution Success. However, compared with proprietary models, SLMs remain weaker in Hit Ratio and rubric-based metrics, especially Dependency Awareness. This suggests that the main challenge in L3 is not producing syntactically valid tool calls, but correctly carrying intermediate information across dependent steps.

II. High tool selection quality does not guarantee successful sequential orchestration. Qwen-2.5-7B

Model	Rule-based				Rubric-based				Overall Score
	Schema Understanding		Tool Usage	Completion	Planning		Tool Grounding		
	Schema Compliance Rate	Execution Success Rate	Tool Selection F1 Score	Hit Ratio	Planning Coherence	Multi-round Efficiency	Parameter Appropriateness	Information Grounding	
Small Language Models									
Llama-3.2-3b-instruct	92.82%	85.61%	59.06%	20.27%	0.178	0.130	0.231	0.161	0.356
Qwen-2.5-3b-instruct	92.57%	88.75%	71.10%	20.27%	0.235	0.234	0.291	0.154	0.438
Gemma-3-4b-instruct	96.91%	90.06%	64.97%	20.27%	0.167	0.127	0.212	0.145	0.498
Qwen-2.5-7b-instruct	97.27%	89.59%	81.39%	25.68%	0.268	0.212	0.321	0.193	0.619
Mistral-v0.3-7b-instruct	87.18%	72.53%	68.03%	24.32%	0.182	0.142	0.181	0.153	0.125
Llama-3.1-8b-instruct	99.14%	91.81%	67.40%	28.38%	0.207	0.132	0.276	0.207	0.555
Proprietary Models									
GPT-5.4	100.00%	99.61%	78.42%	54.05%	0.408	0.370	0.500	0.418	0.932
Gemini-3.1-Flash-Lite	100.00%	100.00%	73.52%	39.19%	0.347	0.277	0.392	0.384	0.754
Claude-Sonnet-4.5	100.00%	100.00%	78.15%	44.59%	0.355	0.349	0.431	0.368	0.809

Table 5: Performance Results with Synthesized User Instructions — Hybrid Tool Using Abilities (L4). The highest and second-highest scores are highlighted in green and blue.

achieves the highest Tool Selection F1 among all SLMs at 60.18%. However, its Hit Ratio is only 52.76%, far below proprietary models. This gap indicates that identifying the right set of tools is not sufficient in L3: the agent must also preserve dependency structure, ground arguments from prior observations, and execute calls in a coherent multi-step workflow.

III. Proprietary models maintain high completion through stronger dependency tracking and grounding. GPT-5.4 and Claude-Sonnet-4.5 sustain very high Hit Ratios of 96.67% and 95.00%, respectively. Their advantage is matched by much stronger rubric-based scores, including Dependency Awareness and Information Grounding. This pattern suggests that proprietary models are better at tracking intermediate states and grounding later tool calls on earlier results.

IV. SLMs exhibit heterogeneous failure modes under sequential dependencies. Llama-3.1-8B achieves the highest overall score among SLMs (0.576). Mistral-v0.3-7B leads open models in Hit Ratio (65.83%) and Planning Efficiency, but remains weak in Dependency Awareness and Information Grounding. Gemma-3-4B shows relatively strong Information Grounding but low Hit Ratio (21.11%). This suggests that robust L3 performance requires multiple coupled abilities rather than isolated strengths.

4.1.4 Hybrid Tool-Use Abilities (L4)

We report the results on complex hybrid recommendation tasks in Table 5. The results reveal several critical observations.

I. L4 hybrid tasks widen the gap between tool execution and recommendation success. By combining structural complexity with semantic ambiguity, L4 requires agents to coordinate multi-round tool use while resolving underspecified intent. In this setting, task completion drops sharply: the best proprietary models reach only 54.05% and 44.59% Hit Ratio, while SLMs remain at 20%–28%. Thus, long-horizon hybrid recommendation remains difficult even when tool execution is largely reliable.

Model	L2		L3		L4	
	Calls	Rounds	Calls	Rounds	Calls	Rounds
Ground Truth (average)	**3.3**	**1.0**	**3.0**	**3.0**	**4.2**	**3.7**
Small Language Models						
Llama-3.2-3B	37.7	4.8	6.1	6.0	41.7	5.4
Qwen-2.5-3B	10.0	2.8	5.9	5.8	15.5	2.7
Gemma-3-4B	16.5	4.6	6.2	6.0	27.0	5.6
Qwen-2.5-7B	12.9	4.4	5.8	5.8	16.5	4.6
Mistral-7B	20.0	4.5	7.5	5.9	22.7	5.3
Llama-3.1-8B	32.5	5.0	6.1	6.0	68.2	6.0
Proprietary Models						
GPT-5.4	9.8	2.1	11.7	2.9	11.8	3.1
Claude-Sonnet-4.5	14.7	3.6	11.2	3.3	11.9	4.0
Gemini-3-Flash	18.8	4.4	23.7	5.3	19.0	4.3

Table 6: Average tool calls and rounds across models.

II. High rule-based scores do not ensure hybrid-task success. In L4, many models achieve high Schema Compliance, Execution Success, and Tool Selection F1, but their Hit Ratio remains substantially lower. This gap suggests that L4 task requires coherent multi-round coordination and evidence-grounded recommendation.

III. Hybrid tool-use and grounding remains challenging. Proprietary models generally achieve higher overall performance, suggesting stronger capabilities in planning and evidence grounding. SLMs show competitive results on several rule-based metrics, but their performance is less consistent on rubric-based dimensions and final recommendation accuracy. This discrepancy indicates that hybrid-task success requires more than correct

tool invocation; models must also integrate evidence across multiple steps and maintain grounding throughout the reasoning process.

4.2 Round and Tool Calls Across Models

Table 6 shows that proprietary models adhere most closely to the ground-truth execution budget, maintaining round counts that match the expected task structure despite utilizing more tool calls. In contrast, Llama models exhibit severe redundancy, often repeating identical calls rather than contributing new information. SLMs demonstrate a different inefficiency at L3, largely proceeding serially by issuing one call per round until the limit, instead of parallelizing independent steps. Collectively, these patterns indicate that efficient tool use hinges on redundancy control, dependency-aware parallelization, and reliable termination, rather than merely on the ability to invoke tools.

4.3 Error analysis

We further inspect GPT-5.4 trajectories in L4. The two dominant failure modes are dependency reasoning failures (54.1%) and target retrieval misses (53.6%), showing that strong proprietary models still struggle to maintain cross-step dependencies and propagate intermediate evidence into the final recommendation.

5 Related Work

5.1 Tool-Augmented Recommender Systems

Recent work augments LLM-based recommender systems with external tools. InteRecAgent (Huang et al., 2025b) and ToolRec (Zhao et al., 2024) explore tool orchestration and surrogate-user modeling, while RecGPT-V2 (Yi et al., 2025) and i^2Agent (Xu et al., 2025) extend tool use to multi-agent collaboration and safety enforcement. However, these systems differ in their tool sets, task formulations, and evaluation protocols, motivating a unified benchmark for systematic evaluation and fine-grained diagnosis of agents' tool-calling capabilities in recommendation scenarios (Zhang et al., 2025; Fan et al., 2022; Chen et al., 2023). Additional related work and system details are provided in the Appendix.

5.2 Related Benchmarks

Existing related benchmarks fall into two categories. **(i) Recommendation-specific benchmarks without tool calling** evaluate LLM knowledge and reasoning in e-commerce scenarios. Shopping MMLU (Jin et al., 2024) and ChineseEcomQA (Chen et al., 2025a) use static QA formats that reduce recommendation to isolated text generation, while WebShop (Yao et al., 2022) and WebArena (Zhou et al., 2024) enable interactive evaluation but focus on product-search and purchase tasks rather than broader recommendation workflows. **(ii) Recommendation-specific benchmarks with tool calling** are most closely related to our work but remain confined to narrow e-commerce settings. ShoppingBench (Wang et al., 2026) provides a sandbox with structured APIs, yet it offers no explicit difficulty hierarchy for diagnosing workflows with varying complexity. In contrast, our benchmark is *recommendation-native* and defines *explicit difficulty levels* for fine-grained evaluation.

6 Conclusion

We introduce **RecToolBench**, an MCP-based benchmark for evaluating tool-using agents under fuzzy recommendation instructions. It provides executable tasks with graded difficulty and diverse orchestration patterns, from single-tool use to hybrid workflows. Results show that current models often execute tools correctly but remain weak in parameter grounding, dependency tracking, evidence integration, and grounded recommendation, making complex tool orchestration a key bottleneck for reliable agentic recommender systems.

Limitations

Despite the expanded evaluation across L1–L4, this work still has several limitations. First, RecToolBench is evaluated in a controlled MCP environment, which supports reproducible execution but does not fully reflect the noise and latency of real-world industrial-level recommendation systems. Second, the current benchmark covers only a limited subset of recommendation scenarios, tool ecosystems, and user behaviors, given the diversity of industrial production settings. Third, while we now include both small language models and proprietary frontier models, our study still does not cover the full spectrum of agent designs due to cost constraints. Expanding along these dimensions is an important direction for future work.

References

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*.

Haibin Chen, Kangtao Lv, Chengwei Hu, Yanshi Li, Yujin Yuan, Yancheng He, Xingyao Zhang, Langming Liu, Shilei Liu, Wenbo Su, et al. 2025a. Chineseecomqa: A scalable e-commerce concept evaluation benchmark for large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5311–5321.

Xiao Chen, Wenqi Fan, Jingfan Chen, Haochen Liu, Zitao Liu, Zhaoxiang Zhang, and Qing Li. 2023. Fairly adaptive negative sampling for recommendations. In *Proceedings of the ACM Web Conference 2023*, pages 3723–3733.

Xiao Chen, Changyi Ma, Wenqi Fan, Zhaoxiang Zhang, and Li Qing. 2025b. C2kd: Cross-layer and cross-head knowledge distillation for small language model-based recommendation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17827–17838.

Wenqi Fan, Xiangyu Zhao, Xiao Chen, Jingran Su, Jingtong Gao, Lin Wang, Qidong Liu, Yiqi Wang, Han Xu, Lei Chen, et al. 2022. A comprehensive survey on trustworthy recommender systems. *arXiv preprint arXiv:2209.10117*.

Xuanqi Gao, Siyi Xie, Juan Zhai, Shiqing Ma, and Chao Shen. 2025. Mcp-radar: A multi-dimensional benchmark for evaluating tool use capabilities in large language models. *arXiv preprint arXiv:2505.16700*.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology*.

Chengkai Huang, Hongtao Huang, Tong Yu, Kaige Xie, Junda Wu, Shuai Zhang, Julian Mcauley, Dietmar Jannach, and Lina Yao. 2025a. A survey of foundation model-powered recommender systems: From feature-based, generative to agentic paradigms. *arXiv preprint arXiv:2504.16420*.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2025b. Recommender ai agent: Integrating large language models for interactive recommendations. *ACM Transactions on Information Systems*, 43(4):1–33.

Yilun Jin, Zheng Li, Chenwei Zhang, Tianyu Cao, Yifan Gao, Pratik Jayarao, Mao Li, Xin Liu, Ritesh Sarkhel, Xianfeng Tang, et al. 2024. Shopping mmlu: A massive multi-task online shopping benchmark for large language models. *Advances in Neural Information Processing Systems*, 37:18062–18089.

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

Qingquan Li, Shaoyu Dou, Kailai Shao, Chao Chen, and Haixiang Hu. 2026. Evaluating scoring bias in llm-as-a-judge. In *International Conference on Database Systems for Advanced Applications*, pages 19–34. Springer.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. 2025a. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Zhiwei Liu, Jielin Qiu, Shiyu Wang, Jianguo Zhang, Zuxin Liu, Roshan Ram, Haolin Chen, Weiran Yao, Shelby Heinecke, Silvio Savarese, et al. 2025b. Mcpeval: Automatic mcp-based deep evaluation for ai agent models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 373–402.

Reza Yousefi Maragheh and Yashar Deldjoo. 2025. The future is agentic: Definitions, perspectives, and open challenges of multi-agent recommender systems. *arXiv preprint arXiv:2507.02097*.

Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. 2011. An empirical study of geographic user activity patterns in foursquare. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, pages 570–573.

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

Jiakai Tang, Wen Chen, Yujie Luo, Xunke Xi, Yi Chao, Dian Chen, Zhujin Gao, Yang Li, Fei Sun, Xueyang Feng, et al. 2026. Interactive recommendation agent with active user commands. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 4580–4591.

Jiangyuan Wang, Kejun Xiao, Qi Sun, Huaipeng Zhao, Tao Luo, Jian Dong Zhang, and Xiaoyi Zeng. 2026. Shoppingbench: A real-world intent-grounded shopping benchmark for llm-based agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33521–33529.

Wujiang Xu, Yunxiao Shi, Zujie Liang, Xuying Ning, Kai Mei, Kun Wang, Xi Zhu, Min Xu, and Yongfeng Zhang. 2025. iagent: Llm agent as a shield between user and recommender systems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18056–18084.

Yunhe Yan, Shihe Wang, Jiajun Du, Yexuan Yang, Yuxuan Shan, Qichen Qiu, Xianqing Jia, Xinge Wang, Xin Yuan, Xu Han, et al. 2025. Mcpworld: A unified benchmarking testbed for api, gui, and hybrid computer use agents. *arXiv preprint arXiv:2506.07672*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, et al. 2025. Toolhop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2995–3021.

Chao Yi, Dian Chen, Gaoyang Guo, Jiakai Tang, Jian Wu, Jing Yu, Mao Zhang, Wen Chen, Wenjun Yang, Yujie Luo, et al. 2025. Recgpt-v2 technical report. *arXiv preprint arXiv:2512.14503*.

Peijie Yu, Wei Liu, Yifan Yang, Jinjian Li, Zelong Zhang, Xiao Feng, et al. 2026. Benchmarking llm tool-use in the wild. In *International Conference on Learning Representations*, volume 2026, pages 48265–48294.

Haobo Zhang, Yutao Zhu, Kelong Mao, Tianhao Li, and Zhicheng Dou. 2026. Recthinker: An agentic framework for tool-augmented reasoning in recommendation. *arXiv preprint arXiv:2603.09843*.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2025. Recommendation as instruction following: A large language model empowered recommendation approach. *ACM Transactions on Information Systems*, 43(5):1–37.

Yuyue Zhao, Jiancan Wu, Xiang Wang, Wei Tang, Dingxian Wang, and Maarten De Rijke. 2024. Let me do it for you: Towards llm empowered recommendation via tool learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1796–1806.

Lucen Zhong, Zhengxiao Du, Xiaohan Zhang, Haiyi Hu, and Jie Tang. 2025. Complexfuncbench: Exploring multi-step and constrained function calling under long-context scenario. *arXiv preprint arXiv:2501.10132*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

Appendix

A Author Contributions **12**

B Detailed Related Work **12**
- B.1 Tool-Augmented Recommender Systems 12
- B.2 Related Benchmark 12

C Details of Used MCP Servers **12**

D Benchmark Details **16**
- D.1 Synthesis Pipeline Details 16
- D.2 Prompts for Task Synthesis 16

E Evaluation Metric for each Level **16**
- E.1 L1 Metrics 16
- E.2 L2 Metrics 16
- E.3 L3 Metrics 17
- E.4 L4 Metrics 18

F Detailed Experiment Results **23**
- F.1 Level 1 23
- F.2 Level 2 23
- F.3 Error Analysis 23
- F.4 Robustness of LLM-as-a-judge evaluation 23

A Author Contributions

Xiao led the project, coordinated its overall development and execution, and took primary responsibility for drafting the manuscript. He designed and contributed to L1–L4 data synthesis, data verification, and analysis of the experimental results. YC contributed to L1–L2 data synthesis and partial experiments. YY contributed to L3 data synthesis and partial experiments. Zhendong provided valuable suggestions and guidance on idea initiation and experimental design throughout the project. CY contributed to figure refinement. QS and XS provided valuable support throughout the project.

B Detailed Related Work

B.1 Tool-Augmented Recommender Systems

The tool-use capabilities of LLMs have been extensively studied. Representative benchmarks include τ-Bench (Yao et al., 2024) and BFCL (Patil et al., 2025) for multi-turn tool-use evaluation, as well as ToolHop (Ye et al., 2025) for evaluating complex tool-dependent reasoning.

Building on these advances, integrating external tools into LLM-based recommender systems has emerged as an active research direction. InteRecAgent (Huang et al., 2025b) adopts an "LLM-as-brain, RS-as-hands" architecture in which the LLM coordinates recommendation tools. ToolRec (Zhao et al., 2024) introduces surrogate-user modeling and attribute-conditioned tool chains. RecGPT-V2 (Yi et al., 2025) scales to production via a multi-agent framework with specialized agents collaborating through tool-calling protocols. i^2Agent (Xu et al., 2025) further positions the agent as a protective intermediary that enforces safety constraints via external tools.

B.2 Related Benchmark

More broadly, general tool-calling benchmarks (Yu et al., 2026; Zhong et al., 2025; Li et al., 2025) emphasize multi-step function composition, and MCP-based benchmarks (Yan et al., 2025; Gao et al., 2025; Liu et al., 2025b) focus on standardized interaction protocols. Neither line captures the fuzzy intent grounding, item-centric retrieval, and heterogeneous tool compositions that characterize real-world recommendations.

C Details of Used MCP Servers

In Table 7, Table 8 and Table 9, we show the detailed descriptions for the involved MCP servers and the associated tools for the three individual tasks based on Amazon Electronics, Foursquare and Yelp data.

Server Name	Tool Name	Input	Output
Retrieval MCP Server	Get Item Details	Item ID (has fall back to fuzzy item title search)	JSON of full item metadata
	Filter Items by Attributes	Category	JSON of filtered items
		brand	
		min price	
		max price	
	Item2Item Search	Query Text (try match against item title)	JSON of referenced items based on similarity score
		Top-K	
	Search Keywords	Query (try match against item description)	JSON of filtered items by keywords, sort by hit rate
		Top-K	
NLPTool MCP Server	Query Sentiment	Min Positive Rate	JSON string with items matching the sentiment threshold
	Search Reviews	Keywords (List of keywords to search for in review)	JSON string with items whose reviews mention the specified aspects
		Search In	
Recommendation MCP Server	Rank Items	User ID	JSON of ranked items according to user id, contains id, score and rank
		Item IDs	
	Get User History	User ID	JSON with user id, histroy length, last 20 items
Rating MCP Server	Filter Products	Min Rating	JSON string with products matching the rating criteria, sorted by rating descending
		Max Rating	
		Min Reviews	
		Max Reviews	
		Category	
		Sort By	
		Limit (maximum number of returns, default 20)	
	Compare Ratings	Item IDs (list)	JSON string with rating comparison for the specified items
Websearch MCP Server	Web Search	Query	JSON string with search results including title, link, snippet, and optional knowledge graph
		Num Results (default 10)	

Table 7: Details of tools and descriptions in used Amazon Electronics MCP Servers.

Server Name	Tool Name	Input	Output
POI Retrieval MCP Foursquare Server	Get POI Details	Venue ID	JSON of venue item
	Search POIs	Query (try match against category name)	JSON of venues that met requirements (query like category name, or location within distance)
		Location (format: [lat, lng])	
		Distance (in meters)	
		Top-K	
	Filter POIs	Category	JSON of venues that exact match category name or location or min checkins
		Min Checkins	
		Location	
Recommendation MCP Foursquare Server	Rank Items	User ID	JSON of ranked venues by user ID; contains venue ID, score, and rank
		Item IDs	
	Get User History	User ID	JSON with user id, histroy length, last 20 items
GoogleMap MCP Server	Google Geo	Address	JSON of geo object
		City	
	Google ReGeo	LatLng	JSON of geo object
	Filter POIs	Origin (LatLng)	JSON of geo object with distance and travel time
		Destination (LatLng)	

Table 8: Details of tools and descriptions in used Foursquare MCP Servers.

Server Name	Tool Name	Input	Output
Retrieval MCP Yelp Server	Get Business Details	Query Text (of business name)	JSON of business item
	Search Businesses	Query (against name, address, categories)	JSON of businesses that met either query or around location, only top k
		Location	
		Top-K	
	Filter Businesses	Categories (exact category, list flatten)	JSON of businesses that met exact criteria
		Location	
		Open Timestamp (UTC Timestamp where business is open)	
	Recall Similar Items	Query Text (business name)	JSON of business frequently visited together
NLPTool MCP Server	Query Sentiment	Min Positive Rate	JSON string with items matching the sentiment threshold

Continued on next page

Table 9 —— Continued from previous page

Server Name	Tool Name	Input	Output
	Search Reviews	Keywords (List of keywords to search for in review)	JSON string with items whose reviews mention the specified aspects
		Search In	
Recommendation MCP Server	Rank Items	User ID	JSON of ranked items by user ID; contains item ID, score, and rank
		Item IDs	
	Get User History	User ID	JSON with user id, histroy length, last 20 items
Rating MCP Server	Filter Products	Min Rating	JSON string with products matching the rating criteria, sorted by rating descending
		Max Rating	
		Min Reviews	
		Max Reviews	
		Category	
		Business IDs (list)	
		Item IDs (list)	
		Sort By	
		Limit (maximum number of returns, default 20)	
	Compare Ratings	Item IDs (list)	JSON string with rating comparison for the specified items
GoogleMap MCP Server	Google Geo	Address	JSON of geo object
		City	
	Google ReGeo	LatLng	JSON of geo object
	Filter POIs	Origin (LatLng)	JSON of geo object with distance and travel time
		Destination (LatLng)	

Table 9: Details of tools and descriptions in used Yelp MCP Servers.

D Benchmark Details

D.1 Synthesis Pipeline Details

Ambiguity Fuzzification. We introduce ambiguity from five categories: *underspecification*, *soft contradiction*, *vague quantifiers*, *preference uncertainty*, and *implicit constraints*. Each task is assigned one of three intensity levels—low, moderate, and high—sampled with a 40%/40%/20% distribution.

Quality Filtering Criteria. Filtering criteria are tailored to each level: Level 1 verifies schema executability and parameter validity; Level 2 additionally checks parallel independence among required tools; Level 3 validates dependency-chain correctness; Level 4 verifies the hybrid parallel-serial structure and fan-in dependencies.

Human Validation of Query Quality. Synthesized query tasks from each domain and difficulty level are manually inspected by two annotators with experience in recommender systems and tool use. Annotators evaluate each instance along four dimensions: (i) whether the fuzzy instruction is natural and realistic, (ii) whether the intended recommendation goal is recoverable, (iii) whether the ground-truth tool workflow is executable under the MCP tool schemas, and (iv) whether dependency-chain annotations are correct for multi-step tasks. The two annotators show high agreement across these validation dimensions, with Cohen's κ of 0.78. Disagreements are resolved through discussion, and low-quality instances are removed or regenerated.

D.2 Prompts for Task Synthesis

We detail the prompts used in our task synthesis process below, including user query instruction generation, prompts used for fuzzy matching and data quality filtering.

E Evaluation Metric for each Level

E.1 L1 Metrics

In the Level 1 single-tool evaluation scenario, the task is relatively straightforward. We therefore use five **rule-based metrics** to assess the agent's schema understanding ability. Correct parameter specification, together with successful execution, indicates that the model has successfully resolved the recommendation task.

Here, let $i = (n, a)$ represent a single tool invocation produced by the agent for a given task, where n is the tool name and a is the argument object. Assume there are N independent tool invocation tasks in total. Each rule-based metric is defined as follows:

1. **Tool Name Valid Rate** (T_{valid}) calculates the fraction of tool invocations whose tool name belongs to the available tool set $\mathcal{T}_{\text{available}}$, denoted as $T_{\text{valid}} = \frac{1}{N}\sum_{j=1}^{N} \mathbf{1}\left[n_j \in \mathcal{T}_{\text{available}}\right]$.

2. **Schema Compliance Rate** (S_{acc}) assesses the fraction of tool invocations whose arguments strictly conform to the selected tool's input schema (i.e., required fields and data types).

3. **Execution Success Rate** (R_{success}) quantifies the success rate of executing the tool invocation without runtime errors and producing valid outputs, denoted as $R_{\text{success}} = \frac{1}{N}\sum_{j=1}^{N} \mathbf{1}\left[\text{exec}(j) = \text{Success}\right]$.

4. **Tool Selection Accuracy** (T_{acc}) measures the accuracy of tool choices compared to the ground-truth tool, denoted as $T_{\text{acc}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left[n_i = \hat{n}_i\right]$, where $\hat{n}_i$ is the ground-truth tool name for task i.

5. **Parameter Appropriateness** (S_{PA}) evaluates whether the generated arguments are semantically appropriate under fuzzy intent. We employ an independent LLM judge (DeepSeek-V3.2; different family from our synthesis model) to score appropriateness from the fuzzy user query, *without access to the generated reference*. Scores are normalized to $[0, 1]$.

E.2 L2 Metrics

In the Level-2 Multi-Tool Parallel scenario, the task is ideally completed in a single round. Beyond schema understanding and tool usage, we also evaluate planning effectiveness, and end-to-end task completion:

1. **Schema Understanding Ability** evaluates input schema adherence and execution validity. We adopt two metrics established in L1: *Schema Compliance Rate* (S_{acc}) and *Execution Success Rate* (R_{success}). In the L2 context, these metrics are aggregated across all tool invocations.

Servers & Tools	Task Description
Amazon L2 **Servers:** NLP Tool, Rating, Retrieval **Useful Tools (5):** compare_ratings, search_keyword, filter_items_by_attributes, search_reviews, item2item_search	Hey, so I've been looking at a few electronics items lately — got my eye on these specific ones: [*candidate item list*]. I'm kind of overwhelmed trying to figure out which one to actually go with. Basically I want something that's decent quality and won't drain my wallet — ideally under $100 for something in the electronics/accessories realm. I've seen some good reviews on a couple of these but haven't really dug into the details yet. Oh, and I was also curious about what people typically pair with the EVGA Power Booster — maybe that could help me decide? If you could take a look at these items and give me your thoughts on which one seems like the best pick, I'd really appreciate it. I'm not in a huge rush or anything, just trying to make a smart choice before I buy.
Yelp L3 **Servers:** Google Maps, Yelp NLP Tool, Yelp Rating, Yelp Rec, Yelp Retrieval **Useful Tools (5):** google_geo, search_businesses, google_distance, google_regeo, recall_similar_items	I'm looking for some dinner options near my location at 123 Main Street in San Francisco. I'm pretty hungry for pizza tonight and want to stay within walking distance—maybe around 2 kilometers or so. First, could you find pizza places in that area? Once you have those results, I'd like to see the top 3 highest-rated ones. Then, can you check how far each one is from my address and filter out anything that's more than 2000 meters away? If any pizza places are still in the running after that distance filter, I'd love to get their full addresses along with their ratings, review counts, and walking distances so I can make a decision. Oh, and just in case there aren't any good pizza options within that walking range, I have a backup plan. There's a place called Devon Donut & Bagel Company (business ID: qu0qoXXxVVDxMaof1SFzXg) that I really like. If needed, could you find me restaurants similar to that one from this list: [*candidate item list*]? Then check which of those similar places are within about 3000 meters from me and rank them by how similar they are to that Devon place.

Table 10: Examples of tasks in RecToolBench, selected from tasks that require parallel tool invocation or multi-turn sequential tool use.

2. **Tool Usage Quality** assesses the correctness of tool selection and parameter passing. *Tool Selection F1 Score* (T_{F1}) evaluates the appropriateness of tool selection by balancing precision (accuracy) and recall (completeness). *Parameter Accuracy* (P_{acc}) measures the per-call match rate of arguments against the ground truth, using F1 Score for keyword-search tools and Exact Match for others.

3. **Planning Effectiveness** evaluates the efficiency of the execution plan, specifically the agent's ability to minimize redundancy and exploit opportunities for parallel execution.

   Parallel Ratio (R_{para}) quantifies the alignment between the first-round invocations and the ground-truth plan: $R_{\text{para}} = \frac{|T_{\text{r1}} \cap T_{\text{exp}}|}{\max(|T_{\text{exp}}|, |T_{\text{r1}}|)}$. An optimal score ($R_{\text{para}} = 1.0$) indicates exact alignment with no redundant calls.

   Efficiency Score (S_{eff}) evaluates the execution efficiency relative to the ground-truth plan, defined as: $S_{\text{eff}} = \frac{N_{\text{exp}}}{\max(N_{\text{exp}}, N_{\text{total}})}$, where N_{total} denotes the total number of unique tools actually invoked across all rounds. The score reaches its optimum of 1.0 when the agent invokes exactly the expected tools without redundancy, and it decreases when redundant calls occur.

4. **Task Completion** measures the final recommendation quality. *Hit Ratio* (H_{ratio}) evaluates whether the final recommended item matches the ground-truth target.

E.3 L3 Metrics

In Level 3, given the fuzzy and complex nature of the tasks, we employ a comprehensive evaluation framework combining rule-based metrics and LLM-as-judge scoring.

- **Rule-based Evaluation**. Let $E = \{e_1, \ldots, e_k\}$ be the set of all tool invocations during execution. Since we have the ground-truth set of required tools for Level 3 tasks, we evaluate the agent's multi-turn tool usage trajectory along the following dimensions:

  1. **Schema Understanding Ability**: We adopt Schema Compliance (S_{acc}): $S_{\text{acc}} = \frac{|\{e \in E : \text{valid_schema}(e)\}|}{|E|}$, where $\text{valid_schema}(e)$ indicates parameter adherence to the schema; and Execution Success (R_{suc}): $R_{\text{suc}} = \frac{|\{e \in E : \text{success}(e)\}|}{|E|}$, where $\text{success}(e)$ denotes error-free execution.
  2. **Tool Usage Quality**: We adopt Tool Selection F1 (T_{F1}) and Parameter Accuracy (P_{acc}) from Level 2, calculated over the agent's entire trajectory E.
  3. **Planning Efficiency Score**: We define sequential efficiency score (S_{eff}): $S_{\text{eff}} = 1 - \frac{|\{e \in E : \text{is_duplicate}(e)\}|}{|E|}$, where $\text{is_duplicate}(e)$ indicates a redundant invocation with identical tool and arguments.
  4. **Task Completion**: We utilize the end-to-end metric *Hit Ratio* (R_{hit}), measuring the alignment between the final result and the Ground Truth.

- **Rubric-based LLM Evaluation.** We employ DeepSeek-V3.2 (Liu et al., 2025a) as LLM-as-a-Judge to evaluate two dimensions: **Information Grounding**, assessing whether the final answer is derived from tool execution results rather than hallucinations, and **Dependency Awareness**, verifying whether the agent correctly follows the dependency chain by using outputs from previous steps. Each dimension is scored on a 1–10 scale using a fine-grained rubric, then normalized to $[0, 1]$ for standardized comparison.

E.4 L4 Metrics

Level 4 evaluates hybrid workflows combining parallel and serial tool use. We combine rule-based metrics with rubric-based LLM-as-judge scoring for comprehensive evaluation.

- **Rule-based Evaluation.** We evaluate three dimensions following the metric definitions in Level 3: (1) **Schema Understanding** (S_{acc}, R_{suc}), (2) **Tool Usage Quality** via Tool Selection F1 (T_{F1}), Unlike lower-level tasks, Level 4 hybrid workflows usually do not have a unique reference parameterization: downstream arguments may depend on the agent's intermediate results. Therefore, exact-match rule-based Parameter Accuracy is not appropriate, and we instead evaluated it semantically through LLM-as-a-judge. and (3) **Task Completion** via Hit Ratio (R_{hit}).

- **Rubric-based LLM Evaluation.** We employ DeepSeek-V3.2 (Liu et al., 2025a) as judge, given the task specification, expected workflow structure, execution trace, and final answer. Four dimensions are scored on a 1–10 scale and normalized to $[0, 1]$: **Planning Coherence** (S_{PC}), whether the agent correctly decomposes the task, parallelizes independent tools, respects serial dependencies, and performs fan-in aggregation, **Multi-round Efficiency** (S_{ME}), whether the agent avoids unnecessary sequentialization, redundant calls, and excessive retries, **Parameter Appropriateness** (S_{PA}), whether parameters are semantically reasonable given task constraints, user intent, and upstream outputs, and **Information Grounding** (S_{IG}), whether the final response is faithfully grounded in tool outputs without hallucinated attributes or unsupported conclusions.

Data Quality Filter

Task Quality Assessment Prompt

System Role:
You are an expert evaluator for AI agents that execute serial tool chains. Score strictly based on evidence in the execution trace. Be critical: default to 4–5 unless strong evidence supports higher.
User:
You must assign scores **only based on evidence** from the task, solution, and tool usage. Your evaluation should be:

- Objective (avoid being influenced by language fluency or formatting)
- Justified (include specific reasons tied to each score)
- Robust against bias (ignore narrative style, verbosity, or formatting polish)

You are evaluating an AI agent that completed a task by calling tools **one at a time** in a sequential chain, where each step may depend on results from the previous step.
TASK DESCRIPTION: "{task}"
EXECUTION TRACE:
{execution_trace}
AGENT'S FINAL ANSWER: "{final_answer}"

Scoring Dimensions — score each on a scale of 1–10.
1. Grounding (G)
Does the agent's final answer faithfully reflect what the tools actually returned? Penalize unsupported claims or hallucinations. Score 9–10 if 90–100% of factual claims are directly grounded in tool outputs; 7–8 if 70–80% are grounded with minor unsupported details; 4–6 if 40–60% are grounded with noticeable hallucination; and 1–3 if less than 30% is grounded and the agent largely ignores tool results.
2. Dependency Awareness (D)
Does each step correctly use the output of the previous step? Penalize broken information flow. Score 9–10 if 90–100% of inter-step dependencies are correctly propagated; 7–8 if 70–80% are correct with one broken link; 4–6 if 40–60% are correct and the chain is partially broken; and 1–3 if less than 30% are correct and the agent treats steps as independent.
Return **only** the following JSON:

```
{
  "grounding_reasoning": "explain grounding...",
  "dependency_awareness_reasoning": "explain dependencies",
  "grounding": <int 1-10>, "dependency_awareness": <int 1-10>
}
```

Fuzzy Prompt

Fuzzing Prompt

Goal: Convert ONE detailed multi-tool sequential task into a NATURAL, CONVERSATIONAL user request. The original task involves a CHAIN of tool calls where later tools depend on earlier tools' outputs.

Input (detailed task):
{detailed_task}

Available tools: {len(tools)} tools (but don't mention them in the fuzzy version)

AMBIGUITY INJECTION:
{ambiguity_config}

[Sampled per task from: AMBIGUITY LEVEL in {LOW, MODERATE, HIGH} with AMBIGUITY TYPES drawn from {UNDERSPECIFICATION, SOFT_CONTRADICTION, VAGUE_QUANTIFIERS, PREFERENCE_UNCERTAINTY, IMPLICIT_CONSTRAINTS}]

CRITICAL RULES (Level3 sequential tasks):

- The task naturally involves MULTIPLE STEPS in sequence.
- You MUST preserve the sequential/dependency nature of the request (e.g., "find X first, then use X to get Y").
- Use natural sequential connectors: "first", "then", "after that", "once you have", "based on the results".
- Do NOT flatten the task into a single parallel request.
- Apply the specified ambiguity level and types, but ALWAYS keep the sequential flow inferable.

STYLE:

- Write like a real user describing a multi-step need.
- Use conversational language; do not sound like a benchmark task.
- Do NOT mention tool names or technical implementation details.
- Embed specific values naturally in conversation.

PRESERVE CRITICAL DATA:

- Keep exact numeric thresholds (ratings, prices, counts) when present.
- Keep ALL user IDs, item IDs (ASINs), and entity IDs EXACTLY as they appear. Do NOT replace them with generic terms like "my" or "the user".
- Keep all candidate item ASINs/IDs that appear in the task — these are INPUT data the agent needs, not target answers to hide.
- Use approximate language when appropriate: "around", "about", "somewhere between".
- ALWAYS USE relative time references (e.g., "past few weeks", "recently", "in the next month"; not specific dates).

HARD CONSTRAINTS:

- Never reveal the TARGET/ANSWER item identity directly (no exact target title, no target ASIN, no quoted seed target name).
- BUT you MUST keep all candidate item ASINs/IDs that appear in the task — these are input data, not answers.

Return ONLY the natural, conversational fuzzy description.

Data Quality Filter

Task Quality Assessment Prompt

```
Task Quality Assessment Prompt

Purpose: Evaluate task quality on solvability, utility, and consistency dimensions

Evaluate this task's quality on three dimensions:

Task Description:
{task_description}

Dependency Analysis (the serial chain logic and tool dependencies):
{dependency_analysis}

Fuzzy Description (what the agent sees):
{fuzzy_description}

Available Tools:
{tool_descriptions}

------------------------------------------------------------

EVALUATION CRITERIA:
1. SOLVABILITY (1-10):

10: All required data is provided, tools perfectly match needs, clear success criteria
8-9: Task is clearly solvable with the given tools, minor ambiguities acceptable
6-7: Mostly solvable but some steps may be challenging or unclear
4-5: Significant gaps in tool coverage or data requirements
1-3: Task cannot be meaningfully completed with available tools

Consider:

Are all necessary tools available?
Is all required data provided (no external dependencies)?
Can the agent achieve the stated goal with these tools based on the function and output of
    ↪ the tools?
Are success criteria clear and measurable?

2. UTILITY (1-10):

10: Perfectly simulates a real user recommendation scenario with rich personalization (budget,
    ↪  brand preference, feature requirements), multi-criteria decision-making (rating,
    ↪ review count, collaborative filtering), and clear practical value for shopping, dining
    ↪ , or location exploration
8-9: Strong recommendation scenario with meaningful user preferences and reasonable
    ↪ complexity; agent needs to synthesize signals from multiple tools to make justified
    ↪ recommendations
6-7: Basic recommendation task but lacks realistic personalization nuances or requires only
    ↪ simple single-tool queries; provides some value but could be more representative of
    ↪ real-world decision-making
4-5: Artificial or toy recommendation task; user intent is unrealistic, preferences are
    ↪ trivial or missing, and the task doesn't reflect actual shopping/exploration behavior
1-3: Completely contrived scenario with no resemblance to how users seek recommendations;
    ↪ fails to test agent's ability to balance multiple factors

Consider:

Does the task reflect a realistic user seeking recommendations (shopping, dining, places to
    ↪ visit)?
Are user preferences authentic (budget, brand affinity, functional needs, occasion context)?
Does the agent need to combine multiple signals (ratings, reviews, collaborative patterns,
```

```
    ↪ attributes) to make a well-justified recommendation?
Would the recommendations be genuinely useful for someone making a real decision?
Is the complexity aligned with typical user decision-making processes?


3. CONSISTENCY (1-10):

10: The fuzzy_description perfectly mirrors the dependency_analysis, same user intent, same
    ↪ key constraints (thresholds, categories, brands, keywords), same filtering/ranking
    ↪ logic, and the user's natural language request would naturally lead an agent to
    ↪ execute exactly the serial chain described
8-9: The fuzzy_description captures the core intent and most key parameters from the
    ↪ dependency_analysis; minor details (e.g., exact threshold numbers) may be softened but
    ↪  the overall workflow is fully recoverable
6-7: The fuzzy_description covers the main goal but misses or contradicts some important
    ↪ constraints, tools, or filtering steps described in the dependency_analysis
4-5: Significant misalignment, the fuzzy_description introduces unrelated requirements, drops
    ↪  critical steps, or implies a different workflow than what the dependency_analysis
    ↪ specifies
1-3: The fuzzy_description is essentially unrelated to the dependency_analysis, or actively
    ↪ contradicts the serial chain logic

Consider:

Does the fuzzy_description reflect the same user need and product domain as the
    ↪ dependency_analysis?
Are the key filtering criteria (category, brand, price range, rating thresholds, sentiment
    ↪ thresholds, keywords) from the dependency_analysis naturally embedded in the
    ↪ fuzzy_description?
Would an agent reading only the fuzzy_description be able to infer the correct serial tool
    ↪ chain described in the dependency_analysis?
Are there any extra requirements in the fuzzy_description that are absent from the
    ↪ dependency_analysis, or vice versa?


Output Format:
Provide scores and brief feedback in JSON format:

{{
  "solvability_score": <number 1-10>,
  "utility_score": <number 1-10>,
  "consistency_score": <number 1-10>,
  "solvability_feedback": "Brief explanation of solvability assessment",
  "utility_feedback": "Brief explanation of utility assessment",
  "consistency_feedback": "Brief explanation of consistency between dependency_analysis and
    ↪ fuzzy_description"
}}
```

F Detailed Experiment Results

F.1 Level 1

We provide the detailed results of Level 1 on each dataset in Tables 11, 12, and 13. The highest and second-highest scores are highlighted in green and blue.

F.2 Level 2

We provide the detailed results of Level 2 on each dataset in Tables 15, 16, and 17. The highest and second-highest scores are highlighted in green and blue.

F.3 Error Analysis

We conduct error analysis on GPT-5.4 trajectories in the L4 setting.

Dependency Reasoning Failure (54.1%). This is the most common failure mode. It occurs when the agent fails to infer or respect dependencies among tools, such as executing downstream tools before prerequisite information is available, skipping required intermediate checks, or breaking multi-step dependency chains. These errors indicate that L4 tasks require more than isolated tool invocation; agents must maintain a coherent execution graph across multiple rounds.

Target Retrieval Miss (53.6%). In these cases, the agent completes a plausible tool-use trajectory but fails to retrieve or identify the target item. This often happens when intermediate evidence is incomplete, overly broad, or not properly propagated to later search and ranking steps. The failure suggests that even when tools are called successfully, the agent may still struggle to translate observations into the correct recommendation target.

No Clear Failure / Correct Trajectory (18.5%). A subset of trajectories does not exhibit an obvious planning or execution error under manual inspection. These cases are either correct or fail due to factors outside the annotated failure taxonomy, such as ambiguous user intent, incomplete tool returns, or borderline candidate relevance. We keep this category to avoid over-attributing errors to planning failures.

Premature Termination (7.7%). Here the agent stops before collecting sufficient evidence or before completing the required tool sequence. Typical cases include producing a final recommendation after only partial retrieval, omitting a necessary verification step, or stopping once a plausible candidate is found. This behavior reflects difficulty in judging when enough evidence has been accumulated.

Unproductive Exploration (0.9%). This category captures trajectories where the agent spends additional steps exploring irrelevant tools or candidates without improving the final decision. Although rare, such failures increase latency and may distract the agent from the original goal.

F.4 Robustness of LLM-as-a-judge evaluation

We assess how well LLM-based judgments align with human preferences using annotations from three independent reviewers. Each reviewer scores the output of each judge pipeline on a 3-point scale, where 0, 1, and 2 denote disagreement, partial agreement, and full agreement, respectively. We report the average score across reviewers and tasks. With prompt shuffling and score averaging, the human agreement score increases from 1.37 to 1.52 out of 2, suggesting improved alignment with human-perceived evaluation quality.

Model	Format Compliance			Functional Accuracy	
	Tool Name Valid Rate	Schema Compliance Rate	Execution Success Rate	Tool Selection Accuracy	Parameter Accuracy
Small Language Models					
Llama-3.2-3b-instruct	94.94%	61.57%	80.94%	19.23%	10.00%
Qwen-2.5-3b-instruct	95.15%	90.84%	91.53%	32.91%	15.00%
Gemma-3-4b-instruct	99.68%	73.02%	73.33%	33.97%	16.19%
Qwen-2.5-7b-instruct	100.00%	96.08%	96.08%	40.78%	28.63%
Mistral-v0.3-7b-instruct	91.54%	88.15%	91.61%	19.74%	11.15%
Llama-3.1-8b-instruct	94.53%	81.73%	95.69%	54.11%	39.01%

Table 11: L1 Performance Results with Synthesized User Instructions on Amazon Electronics Dataset

Model	Format Compliance			Functional Accuracy	
	Tool Name Valid Rate	Schema Compliance Rate	Execution Success Rate	Tool Selection Accuracy	Parameter Accuracy
Small Language Models					
Llama-3.2-3b-instruct	100.00%	87.06%	90.47%	35.00%	33.57%
Qwen-2.5-3b-instruct	100.00%	98.50%	95.64%	6.43%	6.43%
Gemma-3-4b-instruct	100.00%	82.86%	78.57%	78.57%	64.29%
Qwen-2.5-7b-instruct	100.00%	95.71%	95.71%	100.00%	82.86%
Mistral-v0.3-7b-instruct	98.57%	85.90%	84.95%	11.43%	10.71%
Llama-3.1-8b-instruct	99.25%	100.00%	96.39%	37.86%	37.86%

Table 12: L1 Performance Results with Synthesized User Instructions on the Foursquare Dataset

Model	Format Compliance			Functional Accuracy	
	Tool Name Valid Rate	Schema Compliance Rate	Execution Success Rate	Tool Selection Accuracy	Parameter Accuracy
Small Language Models					
Llama-3.2-3b-instruct	96.61%	40.35%	96.15%	18.00%	12.67%
Qwen-2.5-3b-instruct	95.87%	72.15%	96.69%	13.00%	8.67%
Gemma-3-4b-instruct	99.62%	64.23%	71.15%	31.15%	19.62%
Qwen-2.5-7b-instruct	78.07%	69.23%	75.38%	33.85%	26.15%
Mistral-v0.3-7b-instruct	96.78%	46.03%	93.47%	15.33%	11.67%
Llama-3.1-8b-instruct	98.67%	64.39%	98.33%	47.33%	40.33%

Table 13: L1 Performance Results with Synthesized User Instructions on the Yelp2019 Dataset

Model	Format Compliance			Functional Accuracy	
	Tool Name Valid Rate	Schema Compliance Rate	Execution Success Rate	Tool Selection Accuracy	Parameter Accuracy
Small Language Models					
Llama-3.2-3b-instruct	97.18%	62.99%	89.19%	24.08%	18.75%
Qwen-2.5-3b-instruct	97.01%	87.16%	94.62%	17.45%	10.03%
Gemma-3-4b-instruct	99.77%	71.94%	74.35%	49.33%	33.37%
Qwen-2.5-7b-instruct	92.69%	87.01%	89.06%	58.21%	45.88%
Mistral-v0.3-7b-instruct	95.63%	73.36%	89.19%	15.50%	11.18%
Llama-3.1-8b-instruct	97.48%	82.04%	96.80%	46.43%	39.07%
Proprietary Models					
GPT-5.4	100.00%	85.70%	85.70%	100.00%	78.60%
Gemini-3.1-Flash	100.00%	92.86%	92.86%	100.00%	92.86%
Claude-Sonnet-4.5	100.00%	92.90%	92.90%	100.00%	92.90%

Table 14: Evaluation results of LLM agents on atomic tool using abilities (L1).

Model	Schema Understanding		Tool Usage		Planning		Completion
	Schema Compliance Rate	Execution Success Rate	Tool Selection F1 Score	Parameter Accuracy	Parallel Rate	Efficiency Score	Hit Ratio @All
Small Language Models							
Llama-3.2-3b-instruct	98.26%	94.61%	41.88%	14.61%	35.99%	10.52%	16.67%
Qwen-2.5-3b-instruct	94.25%	90.23%	50.88%	15.08%	45.40%	37.35%	8.33%
Gemma-3-4b-instruct	96.47%	92.83%	44.14%	10.99%	39.11%	22.17%	0.00%
Qwen-2.5-7b-instruct	98.93%	95.87%	50.56%	17.15%	43.90%	29.26%	33.33%
Mistral-v0.3-7b-instruct	92.50%	89.85%	46.82%	13.61%	42.93%	19.33%	0.00%
Llama-3.1-8b-instruct	99.93%	98.45%	43.27%	17.84%	42.54%	11.26%	58.33%

Table 15: L2 Performance Results with Synthesized User Instructions on Amazon Electronics

Model	Schema Understanding		Tool Usage		Planning		Completion
	Schema Compliance Rate	Execution Success Rate	Tool Selection F1 Score	Parameter Accuracy	Parallel Rate	Efficiency Score	Hit Ratio @All
Small Language Models							
Llama-3.2-3b-instruct	96.93%	91.95%	64.50%	9.19%	56.50%	12.64%	0.00%
Qwen-2.5-3b-instruct	86.95%	78.79%	47.38%	6.67%	33.33%	38.93%	0.00%
Gemma-3-4b-instruct	88.70%	88.05%	48.29%	2.57%	19.62%	44.81%	50.00%
Qwen-2.5-7b-instruct	95.43%	94.75%	59.53%	11.57%	41.86%	32.48%	0.00%
Mistral-v0.3-7b-instruct	91.41%	69.17%	63.53%	13.14%	51.54%	17.35%	100.00%
Llama-3.1-8b-instruct	100.00%	92.08%	61.06%	15.62%	45.43%	10.13%	0.00%

Table 16: L2 Performance Results with Synthesized User Instructions on Foursquare

Model	Schema Understanding		Tool Usage		Planning		Completion
	Schema Compliance Rate	Execution Success Rate	Tool Selection F1 Score	Parameter Accuracy	Parallel Rate	Efficiency Score	Hit Ratio @All
Small Language Models							
Llama-3.2-3b-instruct	97.08%	96.53%	36.80%	20.47%	25.87%	8.63%	18.60%
Qwen-2.5-3b-instruct	94.57%	91.73%	46.66%	16.40%	38.22%	59.91%	27.91%
Gemma-3-4b-instruct	91.64%	82.22%	38.16%	7.85%	29.63%	22.35%	11.63%
Qwen-2.5-7b-instruct	98.43%	97.68%	55.11%	25.80%	45.56%	31.34%	32.56%
Mistral-v0.3-7b-instruct	85.82%	81.24%	40.49%	17.79%	34.34%	15.36%	18.60%
Llama-3.1-8b-instruct	99.52%	99.33%	43.00%	30.47%	35.79%	9.96%	20.93%

Table 17: L2 Performance Results with Synthesized User Instructions on Yelp