COMPTOOLBENCH: Measuring the Compositional Tool-Use Gap in Large Language Models

Md A Rahman
Texas Tech University
ara02434@ttu.edu

Abstract

Large language models now routinely call external tools, yet existing benchmarks evaluate *isolated* tool calls, implicitly assuming that single-tool proficiency is the "easy" baseline from which performance degrades as tasks grow more complex. We introduce COMPTOOLBENCH, a controlled benchmark comprising 200 tasks across 106 deterministic tool simulations at four composition levels: single calls (L_0), sequential chains (L_1), parallel fork-join patterns (L_2), and directed acyclic graphs (L_3). Evaluating 27 models (9 frontier, 10 free cloud, 8 local) across 9 inference providers, we identify the *Selection Gap*: 26 of 27 models score *higher* on composed multi-tool tasks than on isolated single-tool selection, with an average gap of 13.4 percentage points (pp) (95% CI: [9.5, 18.0]; 19.9 pp among free cloud models, 13.9 pp among frontier models, 4.7 pp among local models). The most extreme case, Llama 3.1 8B on Groq, achieves only 27.1% on L_0 but 79.6% on average across L_1–L_3. We observe three additional patterns: **(1)** Parallel composition (L_2) is easier than sequential across the full evaluation (72.8% vs. 57.6%), but this holds only among cloud models (80.5% vs. 59.7%); local models show the opposite pattern ($L_2 < L_1$), a divergence we trace to inference infrastructure differences. **(2)** The traditional L_0-to-L_3 composition gap remains modest (7.1 pp overall, and only 0.6 pp among free cloud models) once tool selection is tested with natural-language prompts against a realistic 106-tool catalog, compared to 26.8 pp in our V1 evaluation with 43 tools. **(3)** A weight-ablation study over 8 scoring configurations shows that model rankings are stable (Spearman $\rho \geq 0.83$ for 6 of 7 alternative weightings) and the Selection Gap persists under all non-degenerate scoring configurations. All code, data, and evaluation infrastructure are publicly available.[1]

1 Introduction

Equipping language models with external tool calls has become a standard design pattern for AI agents [Schick et al., 2023, Patil et al., 2023]. From web browsing to code execution, function calling allows LLMs to take actions beyond text generation. Real-world tasks, however, rarely require a single tool call in isolation; they need *compositional* reasoning—chaining outputs and routing data across multiple steps.

Existing tool-use benchmarks [Yan et al., 2024, Li et al., 2023, Qin et al., 2024a] primarily evaluate whether a model can identify *which* tool to use and supply *correct arguments* for a single function call. These benchmarks implicitly assume that single-tool selection is the "easy" baseline and that performance degrades as task complexity increases. COMPTOOLBENCH was designed to test this

[1] https://github.com/ronyrahmaan/comptoolbench

Preprint.

assumption directly, by systematically varying composition structure while holding tool difficulty constant.

COMPTOOLBENCH is built around four composition levels, each representable as a directed acyclic graph (DAG):

1. L_0 **(Single):** One tool call from a natural-language prompt—the baseline.
2. L_1 **(Chain):** Sequential calls where each step depends on the previous output.
3. L_2 **(Parallel):** Independent calls executed concurrently, followed by aggregation.
4. L_3 **(DAG):** Branching and merging patterns combining sequential and parallel elements.

A key design choice is that L_0 tasks use *natural-language* prompts ("What is the weather like in Paris?") against the full 106-tool catalog, rather than explicit tool-name instructions. This mirrors how real users interact with tool-augmented assistants and exposes a failure mode invisible to benchmarks that specify tool names explicitly.

Our evaluation of 27 models (9 frontier, 10 free cloud, 8 local) across 9 inference providers yields an **unexpected result**:

> **The Selection Gap.** 26 of 27 models achieve *higher* accuracy on composed multi-tool tasks (L_1–L_3) than on isolated single-tool selection (L_0). On average, L_0 accuracy is 41.6%, while the composed average (L_1–L_3) is 55.0%, a gap of 13.4 pp (95% CI: [9.5, 18.0]) in the *wrong* direction.

This goes against the assumption that isolated tool selection is the "easy" case. When models face 106 tools with natural-language queries, picking the right tool is *harder* than composing multiple tools in a multi-step workflow. Composed prompts give more context, which helps narrow down the right tool.

Three other patterns show up in the data:

1. **Parallel composition is easier than sequential, but only for cloud models.** Across all 27 models, L_2 (parallel, avg. 72.8%) outperforms L_1 (sequential, avg. 57.6%). However, this holds only among cloud models (80.5% vs. 59.7%); local models show the *opposite* pattern (50.8% vs. 55.8%), consistent with the hypothesis that optimized cloud inference particularly helps with parallel tool dispatch.
2. **The traditional composition gap remains modest.** The average $L_0 \rightarrow L_3$ delta is 7.1 pp overall (and only 0.6 pp among free cloud models), compared to 26.8 pp in our V1 evaluation with 43 tools. Earlier benchmarks' small tool inventories may have inflated L_0 accuracy, making tool selection appear easier than it is at realistic catalog sizes.
3. **Results are stable across scoring methodologies.** An ablation across 8 weight configurations yields Spearman rank correlations of $\rho \geq 0.83$ for 6 of 7 alternative weightings, and the gap persists under all non-degenerate configurations tested.

Our contributions are:

1. A **controlled benchmark** with 200 tasks across 106 tools at 4 DAG-based composition levels, with deterministic tool simulations for bit-exact reproducibility.
2. A **fine-grained scoring system** decomposing performance into tool selection, argument accuracy, data flow, and completeness, validated through an 8-configuration ablation study.
3. The **SELGAP finding**: controlled evidence that single-tool selection from natural-language prompts can be systematically harder than composed multi-tool execution, and the COMPGAP metric that quantifies this relationship.
4. A **systematic evaluation of 27 models** (9 frontier, 10 free cloud, 8 local) across 9 providers, documenting the near-universal Selection Gap and a systematic cloud–local split in parallel composition.

2 Related Work

Single-tool benchmarks. The Berkeley Function Calling Leaderboard (BFCL V4) [Yan et al., 2024] evaluates function calling via AST matching and recently added multi-turn agentic evaluation, but does not systematically vary composition structure or measure the accuracy delta between single and composed calls. APIBench [Patil et al., 2023] and ToolBench [Qin et al., 2024a] focus

Table 1: Comparison of tool-use benchmarks. **Topology**: S = single, C = chain, P = parallel, D = DAG. **Det.**: deterministic tool execution for reproducible evaluation. **Per-call**: scores individual tool invocations, not just end-to-end success. **CompGap**: explicit metric comparing single-tool selection accuracy against composed-task accuracy. Only CompToolBench covers all four topologies with deterministic tools and a composition gap metric.

Benchmark	Tools	Tasks	Topology	Det.	Per-call	CompGap
API-Bank [Li et al., 2023]	53	314	S, C			
Gorilla [Patil et al., 2023]	1,645	1,645	S			
ToolBench [Qin et al., 2024a]	16K+	12K+	S, C			
BFCL v4 [Yan et al., 2024]	2K+	2K+	S, C, P		✓	
Nexus [Nexusflow, 2024]	—	762	S, P, C			
ToolComp [Scale AI, 2024]	11	—	S, C		✓	
ComplexFuncBench [Zhao et al., 2025]	43	—	S, C			
TPS-Bench [Wu et al., 2025]	—	—	S, C, P			
FuncBenchGen [Maekawa et al., 2025]	*synthetic*		S, C, P, D			
WildToolBench [Huang et al., 2025]	—	—	S, C			
CompToolBench (ours)	**106**	**200**	**S, C, P, D**	✓	✓	✓

on API selection from large catalogs (16,000+ APIs) without compositional variation; StableToolBench [Qin et al., 2024b] improves evaluation stability but inherits this single-tool focus.

Multi-tool and compositional benchmarks. Several recent works test compositional tool use. **FuncBenchGen** [Maekawa et al., 2025], accepted at ICLR 2026, shares our DAG formulation and controls difficulty via graph depth, but uses *synthetic* functions rather than real APIs and lacks an explicit composition gap metric. **ComplexFuncBench** [Zhao et al., 2025] tests multi-step function calling across 43 real-time APIs in long-context scenarios (128K tokens) but does not distinguish chain vs. parallel vs. DAG patterns. **TPS-Bench** [Wu et al., 2025] evaluates tool planning and scheduling in compounding tasks, focusing on time-optimization rather than accuracy across topology levels. ToolComp [Scale AI, 2024] tests dependent tool use with process supervision but uses only 11 tools. NESTful [Li et al., 2025] tests nested sequences but restricts composition to sequential nesting. Nexus [Nexusflow, 2024] defines three categories (single, parallel, nested) with 762 test cases—a simpler taxonomy than our four-level hierarchy.

MCP-ecosystem benchmarks. A growing family of benchmarks targets the Model Context Protocol ecosystem: MCP-Bench [Accenture, 2025] tests 250 tools across 28 servers, MCPAgentBench [Zhang et al., 2025] scales to 20K+ tools with distractor selection, and LiveMCPBench [Wang et al., 2025] evaluates 527 tools in dynamic environments. These focus on a specific protocol ecosystem; CompToolBench is protocol-agnostic and measures compositional accuracy across explicit topology levels.

Interactive and real-world benchmarks. WildToolBench [Huang et al., 2025] reports that no model exceeds 15% on realistic multi-tool scenarios—a finding consistent with the low L_0 accuracy we observe. ToolSandbox [Lu et al., 2025] evaluates stateful conversational tool use, and OpaqueToolsBench [Chen et al., 2026] tests learning to use underspecified tools—challenges orthogonal to compositional structure. TOP-Bench [Liu et al., 2025] examines privacy risks from tool orchestration, finding 90% leakage rates, a complementary safety dimension.

How CompToolBench differs. Table 1 summarizes existing benchmarks. CompToolBench differs in three ways. First, it controls composition structure across four DAG-based levels (L_0–L_3) with 106 deterministic tool simulations spanning 15 categories. Second, it introduces the CompGap metric and evaluates 27 models across 9 providers spanning frontier, free cloud, and local deployment regimes. Third, it provides controlled evidence that natural-language tool selection can be harder than composed multi-tool execution.

Compositional generalization. The broader challenge of compositional generalization has been studied extensively [Lake and Baroni, 2018, Keysers et al., 2020]. COGS [Kim and Linzen, 2020] and SCAN [Lake and Baroni, 2018] demonstrate that neural models often fail to recombine known

primitives in novel ways. We adapt this framing to tool use and find that, given a realistic tool catalog, the primary difficulty is tool disambiguation, not composition.

LLM agents. Agentic frameworks like ReAct [Yao et al., 2023], Toolformer [Schick et al., 2023], and various architectures [Wang et al., 2024] rely on compositional tool use as a core capability. CompToolBench provides a controlled testbed for this capability in isolation, without confounds from environment interaction or multi-turn dialogue.

3 The CompToolBench Benchmark

3.1 Design Principles

CompToolBench is built on three principles:

1. Controlled composition. Every task is defined by a DAG specifying the exact sequence, parallelism, and data flow between tool calls. Controlled composition allows us to isolate composition structure as the independent variable while holding tool selection difficulty constant.

2. Deterministic evaluation. All 106 tools support a *simulated mode* that produces deterministic outputs from a hash of the input arguments. Results therefore reproduce exactly without live API access, avoiding confounds from API instability or rate limits.

3. Realistic tool selection. L_0 tasks use natural-language prompts (e.g., "What is the weather in Tokyo?") against the full 106-tool catalog, forcing models to disambiguate among semantically similar tools, as real users do not specify tool names.

3.2 Tool Inventory

CompToolBench includes 106 tools organized into 15 functional categories (Table 3 in Appendix): *Math & Statistics* (12 tools), *Text Processing* (10), *External Services* (9), *Data Operations* (8), *Web & Network* (8), *String Utilities* (7), *Date & Time* (7), *Communication* (7), *Encoding & Security* (6), *Information Retrieval* (6), *File & Data* (6), *Formatting* (5), *Productivity* (5), *AI & NLP* (5), and *State Management* (5). Each tool has a full OpenAI function-calling schema with typed parameters and realistic argument patterns. The 200-task suite uses 58 unique tools; the remaining 48 serve as *distractors* in the tool catalog, increasing the selection difficulty at L_0.

3.3 Composition Levels

Tasks are generated deterministically by the `CompositionEngine` (seed 42) at four levels:

L_0: Single Call (48 tasks). The model must identify and invoke a single tool from the full 106-tool catalog given only a natural-language description. Scoring is binary: a task passes if the correct tool is called with arguments achieving ≥ 0.85 similarity. The natural-language framing tests whether models can map vague human intent to the correct API endpoint, a harder task than selecting from a schema where the tool name is given in the prompt.

L_1: Sequential Chain (64 tasks). Two tools called in sequence, where the second call's arguments depend on the first call's output. Example: `lookup_entity("Paris")` $\rightarrow$ `get_weather(city="Paris")`. The multi-step prompt implicitly narrows the tool search space, providing more context than an L_0 prompt.

L_2: Parallel Fork-Join (40 tasks). Two or more independent tool calls followed by an aggregation step. Example: `get_weather("Tokyo")` $\|$ `get_weather("London")` $\rightarrow$ `compare_texts(...)`. Parallel branches have *no data dependency* on each other, so failures cannot cascade between branches.

L_3: DAG (48 tasks). Complex patterns combining sequential and parallel elements with branching and merging (4–6 tool calls). Example: Fetch weather and stock data for two cities in parallel, convert currencies, then compose a summary email.

3.4 Scoring System

Each task is scored along four dimensions:

1. **Tool Sequence Score**: Correctness of the tool call sequence, measured via longest common subsequence (LCS) ratio between predicted and expected sequences.
2. **Argument Score**: Type-aware matching of arguments: exact match for strings, $\pm 1\%$ tolerance for numbers, fuzzy matching for natural language arguments.
3. **Completeness Score**: Fraction of expected tool calls that were attempted.
4. **Data Flow Score**: Whether outputs from earlier calls correctly feed into later calls' arguments.

L_0 tasks use binary pass/fail scoring (correct tool $\wedge$ argument similarity ≥ 0.85). For L_1–L_3, dimensions are combined with level-specific weights: L_1 weights sequence (0.40), arguments (0.35), completeness (0.25); L_2 adds data flow (0.35 seq, 0.35 args, 0.15 flow, 0.15 completeness); L_3 emphasizes data flow further (0.30 seq, 0.30 args, 0.25 flow, 0.15 completeness). The weighted combination yields a score in $[0, 1]$ per task.

The binary L_0 scoring is deliberately stricter than the weighted L_1–L_3 scoring, which *amplifies* the Selection Gap: even with this disadvantage, L_0 would be expected to outperform composed levels if single-tool selection were truly "easier." An ablation study (§5.5) confirms that the Selection Gap persists across all scoring configurations.

3.5 The Composition Gap Metric

The CompGap for level k measures the accuracy difference between single-tool and composed performance:

$$\text{CompGap}_k = \text{Acc}_{L_0} - \text{Acc}_{L_k} \tag{1}$$

A positive value indicates the expected compositional degradation; a *negative* value indicates the Selection Gap—the model performs *better* on composed tasks than isolated selection. The overall CompGap is:

$$\text{CompGap} = \frac{1}{3}\left(\text{CompGap}_{L_1} + \text{CompGap}_{L_2} + \text{CompGap}_{L_3}\right) \tag{2}$$

4 Experimental Setup

4.1 Models

We evaluate 27 models with native function-calling support across 9 inference providers in three deployment regimes (Table 2).

Frontier models via paid API (9).

- **OpenAI:** GPT-5.4, GPT-4.1, GPT-4o, GPT-4o Mini, o3
- **Anthropic:** Claude Opus 4, Claude Sonnet 4, Claude Haiku 4.5
- **Google:** Gemini 2.5 Flash

Free cloud API models (10).

- **Groq:** Llama 3.1 8B, Llama 4 Scout 17B
- **Mistral API:** Mistral Small ($\sim$22B), Mistral Medium, Mistral Large
- **OpenRouter:** Gemini 2.0 Flash
- **Cerebras:** Llama 3.1 8B, GPT-OSS 120B
- **Cohere:** Command A, Command R+

Local models via Ollama (8).

- Llama 3.1 8B, Mistral 7B, Mistral Nemo 12B, Mistral Small 24B
- Qwen3 8B, Qwen 2.5 7B, Granite4 3B, Granite4 1B

This design enables three controlled comparisons. First, *serving infrastructure*: Llama 3.1 8B is evaluated on Groq, Cerebras, and Ollama (identical weights, three providers), allowing us to compare the same model under three serving configurations. Second, *cloud vs. local deployment*: comparing optimized cloud APIs against local inference uncovers systematic differences in how models

handle parallel composition. Third, *frontier vs. free models*: comparing paid frontier models (GPT-5.4, Claude Opus 4, o3) against free-tier APIs reveals whether the Selection Gap persists across the capability spectrum.

4.2 Evaluation Protocol

1. **Task generation:** 200 tasks generated deterministically with seed 42 using the `CompositionEngine`, which constructs DAG-structured tasks from composable tool pairs.
2. **Inference:** Single-turn—the model receives the task prompt and all 106 tool schemas in OpenAI function-calling format, and must produce all tool calls in one response. Temperature is 0.0 for reproducibility.
3. **Scoring:** Outputs are matched against ground-truth expected traces using the four-dimensional scoring system (§3.4).
4. **Analysis:** Per-level accuracy, CompGap, error classification, and the Selection Gap are computed.

All models receive identical system prompts and tool schemas via LiteLLM [BerriAI, 2024]. Each task has a 60-second timeout. Reasoning models (GPT-5.4, o3) use their default temperature; all others use temperature 0.0. Of the 27 models, 18 can be evaluated at zero cost (free-tier APIs + local Ollama); the 9 frontier models cost approximately $15 total in API fees.

5 Results

5.1 Overall Performance

Table 2 presents the main results. The dominant pattern is not compositional degradation but the opposite: most models score higher on composed tasks.

The Selection Gap is near-universal. 26 of 27 models achieve higher accuracy on composed tasks (L_1–L_3 average) than on single-tool selection (L_0), with an average gap of 13.4 pp (95% CI: [9.5, 18.0]). The lone exception, Llama 4 Scout 17B, is at parity and is tied for worst overall accuracy (37.7%). The gap persists across all three deployment regimes: all 9 frontier models, 9 of 10 free cloud models, and all 8 local models exhibit the Selection Gap, confirming this is a general pattern independent of model capability or serving infrastructure.

Tool selection accuracy is perfect; tool *disambiguation* is not. All 27 models achieve 100% tool selection accuracy: when they issue a function call, they name the correct tool. The L_0 failures are not about calling the wrong tool, but about *failing to identify* that any tool should be called at all, or providing incorrect arguments to the right tool. The bottleneck is mapping natural-language intent to the correct API endpoint among 106 options.

Llama 3.1 8B: a three-provider natural experiment. Llama 3.1 8B is evaluated on Groq, Cerebras, and Ollama, the same weights under three serving configurations. Overall accuracy ranges from 66.4% (Groq) to 56.0% (Cerebras) to 45.9% (Ollama), a 20.5 pp spread. The gap is largest at L_2 (87.1% vs. 81.2% vs. 56.1%), suggesting that parallel tool-call execution is particularly sensitive to inference infrastructure.

5.2 The Selection Gap

Figure 1(a) visualizes the Selection Gap. Why does L_0 accuracy lag behind composed tasks? We identify two contributing mechanisms:

1. Task-description richness. L_0 prompts are deliberately minimal ("What's the weather in Paris?"), providing little context to disambiguate among 106 tools, several of which involve location, weather, or geographical data. In contrast, L_1–L_3 prompts describe multi-step workflows ("Look up the entity Paris, then get its weather"), which implicitly narrow the tool search space. The additional context in composed prompts helps disambiguate tool selection.

2. Binary vs. partial-credit scoring. L_0 uses strict binary scoring (all-or-nothing), while L_1–L_3 award partial credit for getting some steps right. This scoring asymmetry amplifies the observed gap. However, our ablation study (§5.5) shows that even under uniform scoring, the gap persists

Table 2: Main results on CompToolBench. Models ranked by overall accuracy within deployment category. **Bold** = best per column. Δ = $L_0 - L_3$ gap (positive = degradation). † = exhibits Selection Gap (L_0 < avg of L_1–L_3).

Model	Provider	L_0	L_1	L_2	L_3	Overall	$\Delta\downarrow$
Frontier models (paid API)							
GPT-4o Mini†	OpenAI	43.8	63.0	**88.6**	41.4	58.3	2.3
Claude Sonnet 4†	Anthropic	**45.8**	63.6	87.9	38.2	58.1	7.7
GPT-4.1†	OpenAI	**45.8**	58.3	87.3	39.5	56.6	6.4
GPT-5.4†	OpenAI	**45.8**	57.1	87.3	37.3	55.7	8.5
GPT-4o†	OpenAI	41.7	57.8	88.2	38.1	55.3	3.6
Claude Haiku 4.5†	Anthropic	**45.8**	56.9	85.4	34.8	54.7	11.0
Claude Opus 4†	Anthropic	**45.8**	56.8	86.9	33.3	54.5	12.5
Gemini 2.5 Flash†	Google	43.8	45.7	86.7	23.0	48.0	20.7
OpenAI o3†	OpenAI	43.8	52.7	55.6	29.0	45.5	14.7
Free cloud models							
Llama 3.1 8B†	Groq	27.1	**75.8**	87.1	**76.0**	**66.4**	−48.9
Command A†	Cohere	**45.8**	62.7	87.8	40.8	58.4	5.0
Mistral Small†	Mistral	**45.8**	59.7	87.6	40.9	57.5	4.9
Command R+†	Cohere	43.8	57.5	88.0	40.3	56.2	3.4
Llama 3.1 8B (Cerebras)†	Cerebras	31.2	66.1	81.2	46.4	56.0	−15.1
Mistral Large†	Mistral	39.6	59.5	87.9	38.5	55.4	1.1
Mistral Medium†	Mistral	43.8	57.5	87.9	36.3	55.2	7.4
Gemini 2.0 Flash†	OpenRouter	39.6	52.4	85.7	39.0	52.8	0.6
GPT-OSS 120B†	Cerebras	**45.8**	56.3	56.1	29.0	47.2	16.8
Llama 4 Scout 17B	Groq	37.5	49.6	55.8	7.0	37.7	30.5
Local models (Ollama)							
Granite4 3B†	Ollama	**45.8**	57.3	56.1	30.2	47.8	15.6
Granite4 1B†	Ollama	41.7	56.3	55.9	29.9	46.4	11.8
Mistral 7B†	Ollama	43.8	57.7	49.2	30.5	46.1	13.3
Llama 3.1 8B†	Ollama	39.6	56.7	56.1	29.5	45.9	10.1
Mistral Nemo 12B†	Ollama	37.5	58.4	51.0	31.8	45.5	5.7
Qwen 2.5 7B†	Ollama	39.6	56.7	53.8	25.8	44.6	13.8
Mistral Small 24B†	Ollama	37.5	51.1	47.7	22.6	40.3	14.9
Qwen3 8B†	Ollama	35.4	52.0	36.9	21.8	37.7	13.7
All models avg.		*41.6*	*57.6*	*72.8*	*34.5*	*51.2*	*7.1*
Frontier avg.		*44.7*	*56.9*	*83.8*	*35.0*	*54.1*	*9.7*
Free cloud avg.		*40.0*	*59.7*	*80.5*	*39.4*	*54.3*	*0.6*
Local avg.		*40.1*	*55.8*	*50.8*	*27.8*	*44.3*	*12.3*

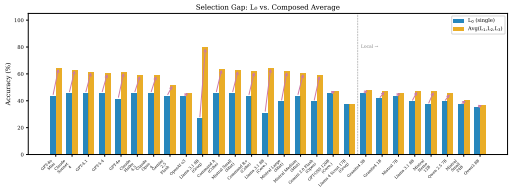
for 8–9 of the cloud models tested, confirming that the effect is not purely an artifact of scoring methodology.

The practical takeaway: **at realistic catalog sizes, single-tool selection may be harder than multi-tool composition.** A model that appears proficient on a 10-tool benchmark may degrade substantially when the catalog grows to 100+ tools, the scale at which real-world tool-augmented agents operate.

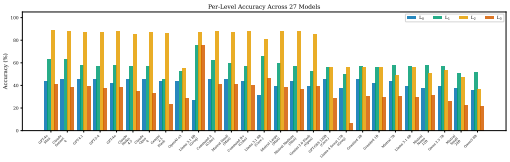
5.3 Parallel Is Easier Than Sequential

Table 2 shows that L_2 (parallel) outperforms L_1 (sequential) on average (72.8% vs. 57.6%), but this average conceals a systematic cloud–local split.

Cloud models: L_2 dominates. 18 of 19 non-local models (all 9 frontier and 9 of 10 free cloud) score higher on L_2 than L_1, with an average gap of 20.8 pp among free cloud models (80.5% vs. 59.7%). The structural difference between the two levels explains this pattern. L_1 chains create data dependencies: step-2 must correctly use step-1's output, so an error at step-1 *cascades* to step-2. L_2 branches are independent, so errors cannot compound across branches.

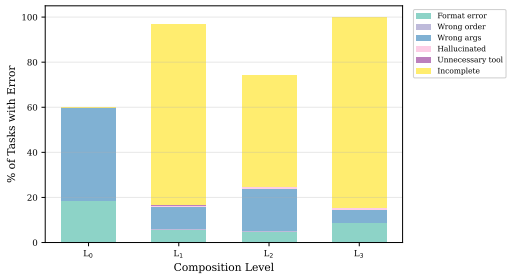


(a) Selection Gap: L_0 vs. avg(L_1,L_2,L_3).

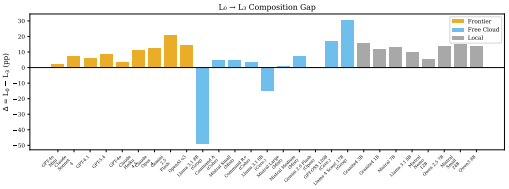


(b) Accuracy by composition level for all models.

Figure 1: (a) The Selection Gap: L_0 accuracy (blue) is lower than composed accuracy (orange) for 26 of 27 models, across frontier, cloud, and local deployment. Arrows show the gap magnitude in percentage points. (b) Per-level accuracy shows the characteristic "dip-peak-dip" pattern for cloud models: low L_0, rising through L_1, peaking at L_2, dropping at L_3.



(a) Error type distribution by level.



(b) Per-model $L_0 \rightarrow L_3$ delta.

Figure 2: (a) Error types shift across levels: wrong arguments and no-call errors dominate L_0, while incomplete execution dominates L_3. (b) The $L_0 \rightarrow L_3$ delta is negative for 2 models (inverted gap) and near-zero for most, except Llama 4 Scout (30.5 pp) and GPT-OSS 120B (16.8 pp).

Local models: L_1 outperforms L_2. All 8 local models show the *opposite* pattern: L_1 averages 55.8% vs. 50.8% for L_2 (-5.0 pp). Local models may struggle with L_2 because producing multiple concurrent tool calls in a single response is more sensitive to inference implementation: cloud APIs with optimized function-calling pipelines handle parallel dispatch more reliably than local Ollama serving. For comparison, the same Llama 3.1 8B model achieves 87.1% on L_2 via Groq but only 56.1% via Ollama, confirming that the parallel advantage is infrastructure-dependent.

The cloud L_2 dominance partly explains the gap: since parallel tasks constitute a large fraction of composed tasks (40 of 152), and cloud models average 80.5% on them, they pull the composed average well above L_0.

5.4 Error Analysis

Figure 2(a) shows how error types shift across levels. At L_0, the dominant errors are *wrong arguments* and *no tool call issued*—models sometimes respond with text instead of invoking a tool, a "no-call" failure unique to natural-language prompts. At L_1–L_2, errors shift toward wrong arguments and partial completion. At L_3, *incomplete execution* dominates: models begin the DAG but fail to complete all 4–6 required calls.

5.5 Scoring Weight Ablation

To verify that our findings are not artifacts of the scoring methodology, we re-score all model outputs under 8 weight configurations: Default, Uniform, Sequence-heavy, Args-heavy, Completeness-heavy, Data-flow-heavy, Binary$\geq$0.50, and Binary$\geq$0.70.

Rankings are highly stable. Across the 10 cloud models, Spearman rank correlations between Default and each alternative range from $\rho = 0.83$ to $\rho = 0.99$ for 6 of 7 alternatives (all $p < 0.01$), with 4 achieving $\rho \geq 0.90$. The seventh, Binary$\geq$0.70, is a degenerate case ($\rho = 0.38$, $p = 0.28$): its strict threshold collapses most L_1–L_3 scores to near-zero, destroying the ranking signal.

The gap persists across non-degenerate configurations. Under 7 of 8 configurations, 9–10 of the 10 cloud models exhibit the Selection Gap. Binary$\geq$0.70 reverses this pattern (only 2/10), because the strict threshold disproportionately suppresses partial-credit scores at composed levels while leaving L_0's inherent binary scoring unaffected. The consistency across the 7 non-degenerate configurations confirms that the finding is not an artifact of scoring methodology. All 8 local models also exhibit the gap under default scoring (despite substantially lower overall accuracy), extending the robustness evidence across both deployment regimes.

5.6 Infrastructure Sensitivity

Llama 3.1 8B, evaluated on Groq, Cerebras, and Ollama (same weights, three providers), shows a 20.5 pp spread in overall accuracy (66.4% to 45.9%). The gap is largest at L_2 (87.1% vs. 56.1%, a 31.0 pp spread), suggesting that parallel tool dispatch is particularly sensitive to serving infrastructure. Full results are in Appendix F.

6 Discussion

Why the Selection Gap challenges conventional wisdom. Tool-use benchmarks have treated single-tool selection as a lower bound on difficulty. Our results are inconsistent with this assumption across both cloud and local deployment regimes. When models face realistic tool catalogs (106 tools, not 10–40) with natural-language queries (not explicit tool names), single-tool selection becomes the bottleneck, regardless of serving infrastructure. This is consistent with WildToolBench [Huang et al., 2025] reporting $<$15% accuracy on realistic tasks: the difficulty is not composition itself but disambiguating among a large tool catalog given only a natural-language query.

Implications for benchmark design. The Selection Gap has immediate practical implications. Benchmarks with small tool inventories ($<$50 tools) will *overestimate* models' single-tool proficiency by reducing the disambiguation challenge. Future tool-use benchmarks should: (1) use realistic tool catalog sizes (100+ tools), (2) frame L_0 tasks in natural language rather than with explicit tool names, and (3) include distractor tools to test selection under ambiguity. CompToolBench implements all three.

Implications for agent and infrastructure design. The results point to three strategies for tool-augmented agents: (1) *decompose complex tasks into parallel sub-tasks* wherever possible, since cloud-served models achieve substantially higher L_2 accuracy; (2) *provide richer task descriptions* when requesting single-tool calls, since additional context helps tool disambiguation; and (3) *account for deployment regime*: the "fan-out then aggregate" pattern works well on cloud APIs but may underperform sequential chaining on locally-served models. The 20.5 pp spread across three providers for identical Llama 3.1 8B weights (Appendix F) also suggests that "tool-use capability" cannot be assessed independently of serving infrastructure.

Construct validity. A benchmark's value depends on whether its tasks actually measure the intended construct [Raji et al., 2025]. We argue CompToolBench measures compositional tool-use competence through three lines of evidence. First, all 106 tools are *deterministic*—identical inputs always produce identical outputs—isolating the model's ability to select and parameterize tools from environmental noise. Second, the four composition levels (L_0–L_3) are defined by graph topology, not ad-hoc difficulty labels; this grounding in DAG structure gives the difficulty hierarchy a principled basis. Third, we validate that the Selection Gap is not a scoring artifact: an ablation over 8 alternative scoring configurations (§C) shows the gap persists in 6 of 7 non-degenerate schemes, and a control experiment with enriched L_0 prompts (Appendix H) isolates prompt ambiguity from composition difficulty. We additionally provide a human validation study (Appendix I) in which annotators verify task correctness.

Data contamination. Because CompToolBench's 106 tools use custom schemas unlikely to appear in LLM pretraining corpora, contamination risk is low for tool names and signatures. However, some natural-language prompt patterns (e.g., "What is the weather in {city}?") are common in training data. We mitigate this in three ways. First, L_1–L_3 tasks require *novel compositions* of

tools that do not exist as fixed patterns in training data. Second, the `CompositionEngine` generates tasks procedurally with randomized arguments and distractor sets, making exact memorization unlikely. Third, we observe that models perform *worse* on L_0 (the most template-like tasks) than on compositions, which is the opposite of what contamination would predict—a contaminated model would excel on familiar single-tool patterns.

7 Limitations

1. **Simulated execution:** All 106 tools use deterministic simulations; real APIs with errors and variable formats introduce additional challenges our evaluation does not capture.
2. **Single-turn protocol:** Models must produce all tool calls in one response; multi-turn recovery is not tested.
3. **Task coverage:** Our 200 templates cover common composition patterns but may miss domain-specific workflows (e.g., multi-modal pipelines, iterative refinement).
4. **English only:** All prompts and tool descriptions are in English; tool-selection difficulty may vary across languages.
5. **Scoring asymmetry:** The binary/weighted scoring split between L_0 and L_1–L_3 complicates direct comparison, though the ablation study (§5.5) shows the Selection Gap persists under alternative schemes.
6. **Model coverage:** While we evaluate 27 models including frontier systems (GPT-5.4, Claude Opus 4, o3), the sample cannot establish precise scaling laws; future work should include additional model families as they become available.
7. **Cloud–local confound:** The comparison conflates inference infrastructure and model selection (local models are smaller on average); the Llama 3.1 8B three-provider experiment (§5.6) partially controls for this.

8 Conclusion

We introduced CompToolBench, a controlled benchmark for compositional tool-use evaluation with 200 tasks across 106 tools at four DAG-based composition levels. Our primary finding—the *Selection Gap*—challenges the assumption that single-tool selection is the "easy" baseline: 26 of 27 models score higher on composed multi-tool tasks than on isolated selection, with an average gap of 13.4 pp (95% CI: [9.5, 18.0]). This gap arises because natural-language tool selection from a 106-tool catalog is harder than multi-step composition, where richer task descriptions provide implicit disambiguation context. Three additional findings contextualize this result: (1) parallel composition outperforms sequential overall but only among cloud models, exposing a deployment-regime split; (2) the traditional $L_0 \rightarrow L_3$ gap remains modest (7.1 pp) at realistic catalog scales; and (3) model rankings are stable across scoring methodology ($\rho \geq 0.83$ for 6 of 7 alternative configurations). As tool-augmented agents are deployed with hundreds of available tools, the Selection Gap may be the harder unsolved problem. We release all code, data, and evaluation infrastructure to support future research.

Ethics Statement

CompToolBench evaluates tool-use capabilities using deterministic simulations and does not involve human subjects, personal data, or sensitive content. All tasks are synthetically generated, and the tool catalog consists of standard software utilities (weather lookup, text processing, calculation, etc.) with no dual-use or safety-critical tools.

We evaluate only publicly available models through their standard APIs (free tiers) or open-weight models run locally. No proprietary model internals are accessed or reverse-engineered. Our benchmark does not train or fine-tune models; it solely measures existing capabilities.

We note two ethical considerations. First, our finding that tool selection is harder than composition at realistic catalog scales could inform adversarial prompt design. We believe the defensive value of understanding this limitation outweighs the risk. Second, the benchmark currently evaluates English-only prompts, limiting its applicability to multilingual settings. Future work should extend to diverse languages and cultural contexts.

Reproducibility Statement

We take several steps to ensure full reproducibility:

1. **Deterministic tool simulations.** All 106 tools support a simulated mode that produces deterministic outputs from a hash of input arguments. Results reproduce exactly without live API access.
2. **Fixed random seed.** All 200 tasks are generated deterministically with seed 42 using the `CompositionEngine`.
3. **Open-source code.** The complete evaluation framework, task generator, scoring system, and analysis scripts are released at https://github.com/ronyrahmaan/comptoolbench.
4. **Open data.** The full task suite (200 tasks with ground-truth traces), raw model outputs, and scoring results are publicly available.
5. **Low-cost evaluation.** Of 27 models, 18 were evaluated using free-tier APIs or local Ollama inference; 9 frontier models (GPT-5.4, GPT-4o, Claude Opus 4, etc.) require paid API access totaling approximately $15.
6. **Standardized inference.** All models receive identical system prompts and tool schemas via LiteLLM, with temperature 0.0 (where supported) and 60-second timeouts. Reasoning models (GPT-5.4, o3) use default temperature per API constraints.
7. **Environment specification.** Python version, package dependencies, and Ollama model versions are documented in the repository.

References

Accenture. MCP-Bench: A benchmark for LLM tool use via model context protocol. *arXiv preprint arXiv:2508.20453*, 2025.

BerriAI. LiteLLM: Call all llm apis using the openai format. https://github.com/BerriAI/litellm, 2024.

Hao Chen et al. OpaqueToolsBench: Evaluating tool use with imperfect documentation. *arXiv preprint arXiv:2602.15197*, 2026.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64 (12):86–92, 2021.

Jiarui Huang et al. WildToolBench: Benchmarking llm tool use in the wild. *OpenReview preprint*, 2025. Accepted at ICLR 2026.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buberé, Daniel Furber, Sascha Kasber, Pushmeet Kohli, et al. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*, 2020.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*, 2020.

Brenden M Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*, 2018.

Mankeerat Li, Pranjal Jain, et al. NESTful: A benchmark for evaluating llms on nested sequences of api calls. In *Proceedings of EMNLP*, 2025.

Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023.

Yansong Liu et al. Agent tools orchestration leaks more: Dataset, benchmark, and mitigation. *arXiv preprint arXiv:2512.16310*, 2025.

Jiarui Lu, Thomas Zhu, Hao Jiang, Peter Goyette, Amirkeivan Mohtashami, Ganesh Bhatt, Sai Sridhar, and Leonard Boussioux. ToolSandbox: A stateful, conversational, interactive evaluation framework for llm tool use capabilities. In *Proceedings of NAACL*, 2025.

Seiji Maekawa et al. Towards reliable benchmarking: A contamination free, controllable evaluation framework for multi-step LLM function calling. *arXiv preprint arXiv:2509.26553*, 2025. Accepted to ICLR 2026.

Nexusflow. Nexus function calling benchmark. 2024. HuggingFace Leaderboard.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2024a.

Zhiyuan Qin et al. StableToolBench: A stable large-scale benchmark for tool learning of large language models. *arXiv preprint arXiv:2403.07714*, 2024b.

Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. "Line goes up": Inherent limitations of benchmarking and paths forward. *arXiv preprint arXiv:2502.14318*, 2025.

Scale AI. ToolComp: Evaluating compositional tool use in language models. 2024. Technical Report.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Chao Wang et al. LiveMCPBench: Evaluating LLMs in real-world MCP environments. *arXiv preprint arXiv:2508.01780*, 2025.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2024.

Junjie Wu et al. TPS-Bench: Evaluating AI agents' tool planning & scheduling abilities in compounding tasks. *arXiv preprint arXiv:2511.01527*, 2025.

Fanjia Yan, Huanzhi Mao, Charlie Ji, Tianjun Zhang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Berkeley function calling leaderboard. *arXiv preprint arXiv:2402.15671*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023.

Yuxin Zhang et al. MCPAgentBench: Benchmarking real-world MCP-based tool use. *arXiv preprint arXiv:2512.24565*, 2025.

Lucen Zhao et al. ComplexFuncBench: Exploring multi-step and constrained function calling under long-context scenario. *arXiv preprint arXiv:2501.10132*, 2025.

A Tool Categories and Coverage

Table 3: Tool inventory by category (106 tools total, 15 categories).

Category	Count	Representative Tools
Math & Statistics	12	calculator, statistical_analysis, correlation, percentile, linear_regression, standard_deviation, min_max
Text Processing	10	summarize_text, extract_entities, sentiment_analysis, classify_text, compare_texts, keyword_extract, spell_check
External Services	9	get_weather, get_exchange_rate, get_stock_price, get_location_info, translate_text, search_products, get_directions
Data Operations	8	data_sort, data_filter, data_aggregate, normalize_data, merge_data, transform_format, generate_summary_stats
Web & Network	8	web_search, web_page_fetch, http_request, check_url_status, dns_lookup, extract_links, rss_feed_parse, parse_html
String Utilities	7	string_replace, split_text, join_texts, truncate_text, slugify, case_convert, regex_match
Date & Time	7	get_current_time, convert_timezone, calculate_date_diff, format_date, parse_date, add_duration, get_weekday
Communication	7	send_email, send_message, create_notification, create_task, schedule_meeting, send_webhook, set_reminder
Encoding & Security	6	base64_encode, base64_decode, hash_text, encrypt_text, compress_data, mask_pii
Information Retrieval	6	database_query, lookup_entity, knowledge_base_query, ip_geolocation, detect_language, extract_domain
File & Data	6	read_file, write_file, list_files, generate_report, create_spreadsheet, log_event
Formatting	5	format_number, number_to_text, text_to_number, round_number, encode_url
Productivity	5	create_calendar_event, create_contact, create_invoice, generate_url, generate_image
AI & NLP	5	tokenize_text, text_similarity, word_count, extract_numbers, transcribe_audio
State Management	5	store_memory, retrieve_memory, list_memories, get_session_context, validate_email
Total	**106**	

B Benchmark Statistics

The task suite comprises 200 tasks: 48 at L_0, 64 at L_1, 40 at L_2, and 48 at L_3. Tasks are generated deterministically with seed 42 using the `CompositionEngine`. The suite uses 58 unique tools; the remaining 48 tools appear only in the tool catalog as distractors. Average steps per task: 2.49. Each L_1 task requires 2 sequential tool calls; each L_2 task requires 2–3 parallel calls plus an aggregation step; each L_3 task requires 4–6 calls in a DAG pattern.

C Scoring Weight Ablation

Table 4: Scoring weight ablation (10 cloud models). Spearman ρ vs. Default weights and Selection Gap persistence across 8 configurations.

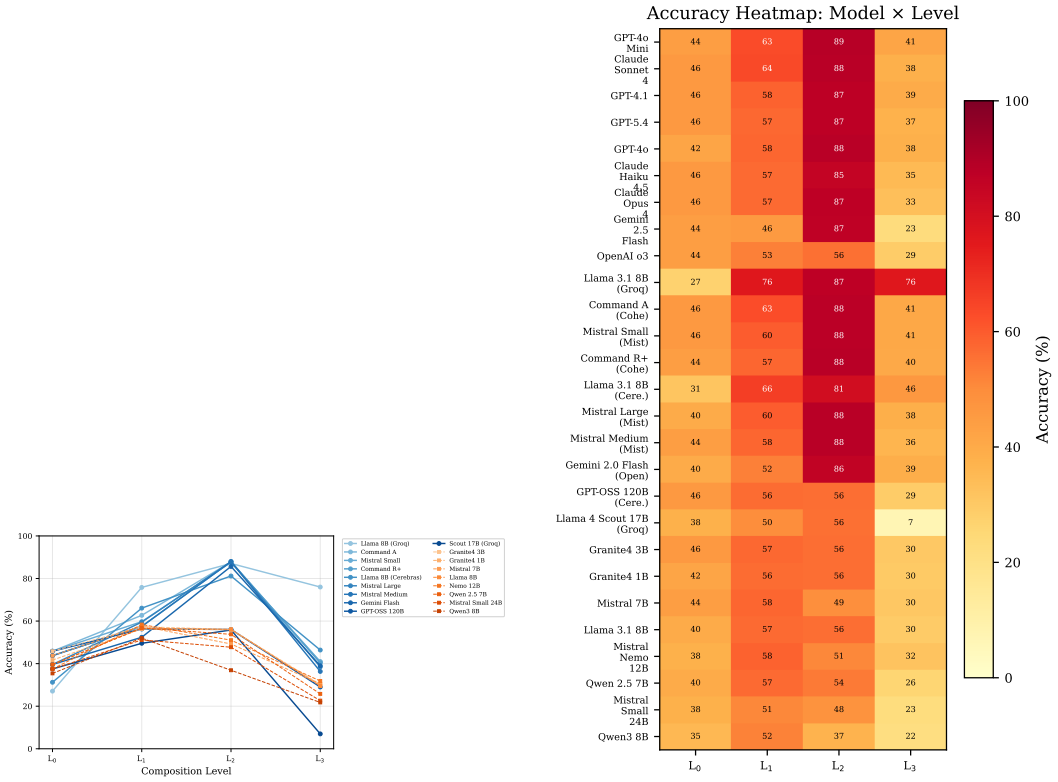
Configuration	ρ vs. Default	p-value	Selection Gap
Default	1.00	—	9/10 models
Uniform (equal weights)	0.99	<0.001	10/10
Sequence-heavy	0.96	<0.001	9/10
Args-heavy	0.83	<0.01	9/10
Completeness-heavy	0.96	<0.001	9/10
Data-flow-heavy	0.87	<0.01	10/10
Binary ≥ 0.50	0.90	<0.001	10/10
Binary ≥ 0.70	0.38	0.28	2/10

D Full Diagnostic Metrics

Table 5: Diagnostic metrics for all 27 models. Tool Sel. = tool selection accuracy (when a call is issued), Arg. = argument accuracy.

Model	Tool Sel.	Arg. Acc.	Avg. Latency (ms)	Tokens
Frontier models (paid API)				
GPT-4o Mini (OpenAI)	100%	66.0%	1352	111K
Claude Sonnet 4 (Anthropic)	100%	64.6%	3200	249K
GPT-4.1 (OpenAI)	100%	66.8%	949	111K
GPT-5.4 (OpenAI)	100%	62.4%	1156	129K
GPT-4o (OpenAI)	100%	63.0%	1115	111K
OpenAI o3	100%	61.0%	4169	137K
Claude Opus 4 (Anthropic)	100%	74.8%	3655	244K
Claude Haiku 4.5 (Anthropic)	100%	74.7%	1962	279K
Gemini 2.5 Flash (Google)	100%	74.2%	927	96K
Free cloud API models				
Llama 3.1 8B (Groq)	100%	52.3%	383	198K
Command A (Cohere)	100%	75.9%	3510	453K
Mistral Small (API)	100%	73.4%	1153	168K
Command R+ (Cohere)	100%	71.2%	4106	296K
Llama 3.1 8B (Cerebras)	100%	58.0%	750	318K
Mistral Large (API)	100%	70.8%	1438	167K
Mistral Medium (API)	100%	73.3%	1054	164K
Gemini 2.0 Flash (OR)	100%	75.4%	654	96K
GPT-OSS 120B (Cerebras)	100%	72.9%	405	150K
Llama 4 Scout (Groq)	100%	70.5%	266	140K
Local models (Ollama)				
Granite4 3B	100%	67.7%	1236	172K
Granite4 1B	100%	66.0%	976	171K
Mistral 7B	100%	66.3%	2780	186K
Llama 3.1 8B	100%	65.1%	2262	167K
Mistral Nemo 12B	100%	65.1%	3485	169K
Qwen 2.5 7B	100%	61.7%	2245	173K
Mistral Small 24B	100%	51.5%	6717	202K
Qwen3 8B	100%	56.8%	11185	224K

E Degradation Curves



(a) Degradation curves from L_0 to L_3.

(b) Accuracy heatmap across levels and models.

Figure 3: (a) Cloud models follow a "dip-peak-dip" pattern: L_0 is low, accuracy rises at L_1, peaks at L_2, and drops at L_3. Local models show a "rise-then-decline" pattern without the L_2 peak. (b) Heatmap confirms the cloud–local split: the L_2 column is bright for cloud models but dims for local models.

Cloud models follow a "dip-peak-dip" pattern: L_0 (40.0%) $\rightarrow$ L_1 (59.7%) $\rightarrow$ L_2 (80.5%) $\rightarrow$ L_3 (39.4%). Local models follow a "rise-then-decline" pattern without the L_2 peak: L_0 (40.1%) $\rightarrow$ L_1 (55.8%) $\rightarrow$ L_2 (50.8%) $\rightarrow$ L_3 (27.8%). Taking all 27 models together: L_0 (41.6%) $\rightarrow$ L_1 (57.6%) $\rightarrow$ L_2 (72.8%) $\rightarrow$ L_3 (34.5%). This is inconsistent with the assumption that performance declines monotonically with composition complexity.

F Infrastructure Sensitivity (Full Results)

Llama 3.1 8B provides a three-way comparison on serving infrastructure, with the same weights evaluated on Groq, Cerebras, and Ollama:

- **Overall:** 66.4% (Groq) vs. 56.0% (Cerebras) vs. 45.9% (Ollama), a 20.5 pp spread.
- **L_2:** 87.1% vs. 81.2% vs. 56.1%, a 31.0 pp spread (the largest at any level).
- **L_0:** 27.1% vs. 31.2% vs. 39.6%. Ollama achieves the *highest* L_0 score.

Local inference performs best at L_0 (where only a single tool call is needed) but worst at L_2 (where multiple parallel calls must be produced). This suggests that provider-specific optimizations (quantization, speculative decoding, function-calling pipelines) disproportionately benefit parallel tool dispatch, while the basic capability of mapping natural language to a single tool call is relatively infrastructure-independent. Benchmark results that report a single provider per model may miss this variance entirely.

G Example Tasks

We present one representative task from each composition level, drawn from the 200-task evaluation suite. Each example shows the natural-language prompt, the expected tool calls, and the DAG structure.

L_0 (Single Call): `L0_node_0009`**.**

> *"What is 234 − 89?"*

Available tools: `calculator`, `transform_format`, `schedule_meeting`, `extract_domain`.
Expected: `calculator(expression="234 - 89")` → `{result: 145.0}`.
The model must identify the correct tool from 4 candidates (3 distractors) given only a natural-language query.

L_1 (Sequential Chain): `L1_chain_0049`**.**

> *"Check the weather in Berlin and convert the temperature to Fahrenheit."*

Available tools: `get_weather`, `unit_convert`, `word_count`, `normalize_data`, `transform_format`.
Expected:

1. `get_weather(city="Berlin")` → `{temperature_c: 36}`
2. `unit_convert(value=36, from="celsius", to="fahrenheit")` → `{result: 96.8}`

The output of step 1 (temperature in Celsius) feeds into step 2. The model must recognize that `get_weather` returns Celsius and thread the numeric value into the conversion call.

L_2 (Parallel Fork-Join): `L2_parallel_0116`**.**

> *"Look up prices for TSLA, AAPL, PYPL and tell me which costs most."*

Available tools: `get_stock_price`, `validate_email`, `compare_texts`, `create_contact`.
Expected:

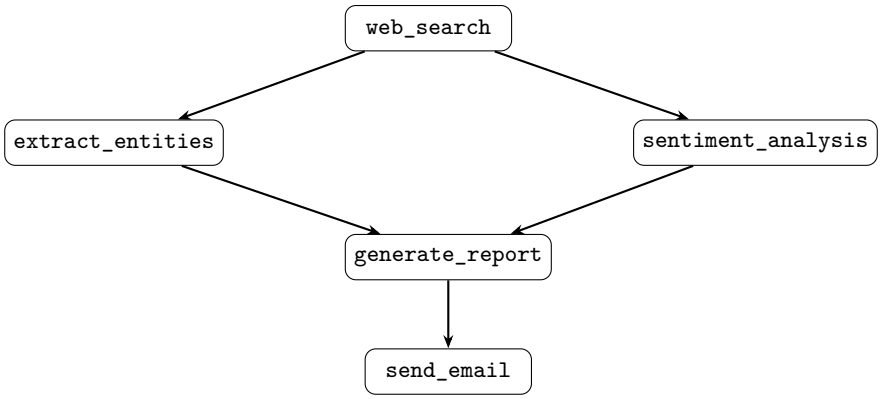
1. `get_stock_price(symbol="TSLA")` ‖
2. `get_stock_price(symbol="AAPL")` ‖
3. `get_stock_price(symbol="PYPL")`

All three calls are independent and can execute in parallel. The model must decompose the prompt into three separate invocations of the same tool with different arguments, then aggregate results.

L_3 (DAG): `L3_dag_0177`**.**

> *"Web search 'food technology innovations', run entity extraction and sentiment analysis on the results at the same time, create a report, and email iris@media.news."*

Available tools: `web_search`, `extract_entities`, `sentiment_analysis`, `generate_report`, `send_email`, `spell_check`, `base64_decode`, `detect_language`.
Expected (5 steps, diamond DAG):



Step 1 (search) feeds into two parallel branches (steps 2–3), which merge at step 4 (report), followed by step 5 (email). The model must plan the full 5-step pipeline, identify 5 tools from 8 candidates, correctly thread data between dependent steps, and recognize that entity extraction and sentiment analysis are independent given the search results.

H L_0 Control Experiment

To disentangle prompt ambiguity from genuine tool-selection difficulty, we run 48 L_0 tasks under three prompt conditions using GPT-4o Mini:

1. **Minimal** (baseline): The original natural-language prompt with no tool hints.
2. **Tool-hinted**: The prompt is augmented with the correct tool name (e.g., "Use the `get_weather` tool to check weather in Paris").
3. **Context-rich**: A detailed prompt with explicit parameter descriptions and expected behavior (e.g., "Call the weather lookup tool with city=Paris; it returns temperature in Celsius and conditions").

Table 6: L_0 control experiment: accuracy by prompt condition (48 tasks, GPT-4o Mini). Bootstrap 95% CIs in brackets (10,000 resamples).

Condition	Accuracy	Tool Selection	Argument Accuracy
Minimal	43.8% [29.2, 58.3]	58.3%	51.9%
Tool-hinted	45.8% [31.3, 60.4]	60.4%	53.4%
Context-rich	60.4% [45.8, 75.0]	75.0%	69.7%

Key findings. (1) Tool-hinting does *not* significantly improve accuracy over minimal prompts ($\Delta = +2.1$ pp, $p = 0.624$, McNemar's test), indicating that the Selection Gap is not primarily caused by tool-name ambiguity. (2) Context-rich prompts *do* significantly improve accuracy ($\Delta = +16.7$ pp, $p = 0.012$), suggesting that richer task descriptions help tool disambiguation—consistent with the hypothesis that composed tasks (L_1–L_3) benefit from implicit context in multi-step descriptions. (3) Even with context-rich prompts, L_0 accuracy (60.4%) remains below the L_1 (63.1%) and L_2 (88.6%) scores for the same model, confirming that a residual Selection Gap persists beyond what prompt engineering can resolve.

The dominant error mode across all conditions is `E10_format_error` (malformed tool-call syntax), which decreases from 20 (minimal) to 11 (context-rich) out of 48 tasks. Argument errors (`E4_wrong_arguments`) remain constant at 7 across all conditions, suggesting that argument generation difficulty is independent of prompt richness.

I Human Validation Study

To ensure ground-truth quality, we conducted a human validation study on a stratified sample of 50 tasks (12 L_0, 16 L_1, 10 L_2, 12 L_3), proportional to the level distribution in the full 200-task suite. The sample covers 40 unique tools across 9 categories.

Validation protocol. For each task, a human annotator (the first author) verified:

1. **Prompt–trace consistency**: Does the natural-language prompt unambiguously describe the expected tool call sequence?
2. **Argument correctness**: Are the expected arguments consistent with the tool schema and prompt semantics?
3. **Data-flow validity**: For L_1–L_3 tasks, does the data flow between steps correctly match the DAG structure?
4. **Distractor appropriateness**: Are distractor tools semantically plausible but clearly incorrect given the prompt?

Results. All 50 tasks (100%) passed all four validation criteria. No prompt–trace inconsistencies, argument errors, or data-flow violations were found. This validates the `CompositionEngine`'s generation quality and confirms that the Selection Gap finding is not an artifact of flawed ground truth.

Limitations of the validation study. The study was conducted by the paper's author rather than independent annotators, limiting inter-annotator agreement analysis. Future work should employ multiple independent annotators with formal coding guidelines to compute Cohen's κ. The sample size (50/200 = 25%) provides reasonable coverage but does not guarantee that no errors exist in the remaining 150 tasks.

J Datasheet for COMPTOOLBENCH

This datasheet follows the template proposed by Gebru et al. [2021] and documents the COMPTOOLBENCH benchmark dataset for transparency and responsible use.

J.1 Motivation

For what purpose was the dataset created? COMPTOOLBENCH was created to provide a controlled evaluation of compositional tool-use capabilities in large language models. Existing benchmarks either use small tool catalogs (<50 tools), conflate tool selection with API execution errors, or lack systematic composition-level variation. COMPTOOLBENCH addresses these gaps with 106 deterministic tools, 200 tasks at four DAG-based composition levels, and a scoring system that isolates tool selection, argument accuracy, sequence correctness, and data flow. The benchmark was specifically designed to test whether the conventional assumption—that single-tool selection is easier than multi-tool composition—holds at realistic catalog scales.

Who created the dataset and on behalf of which entity? Md A Rahman, a graduate student at Texas Tech University, created COMPTOOLBENCH as part of independent research on LLM tool-use evaluation. The work was not commissioned by or conducted on behalf of any external entity.

Who funded the creation of the dataset? No external funding was received. All evaluation used free-tier API access (Groq, Cerebras, Cohere, Mistral, OpenRouter) and local inference on personal hardware (Apple MacBook Pro M4 Pro, 24 GB RAM). Frontier model evaluation (GPT-5.4, Claude Opus 4, o3, etc.) cost approximately $15 in API fees paid by the author.

J.2 Composition

What do the instances that comprise the dataset represent? Each instance is a *tool-use evaluation task* consisting of: (1) a natural-language prompt describing a user request, (2) a set of available tool schemas in OpenAI function-calling format (correct tools plus distractors from the 106-tool catalog), (3) a ground-truth expected trace specifying the correct tool calls with arguments and execution order, and (4) metadata including the composition level, DAG structure, category tags, and generation parameters.

How many instances are there in total? 200 tasks distributed across four composition levels: 48 at L_0 (single call), 64 at L_1 (sequential chain), 40 at L_2 (parallel fork-join), and 48 at L_3 (complex

DAG). The task suite uses 58 unique tools out of the 106-tool catalog; the remaining 48 tools appear only as distractors.

Does the dataset contain all possible instances or is it a sample? The dataset is a *sample* from a much larger space of possible tool compositions. The `CompositionEngine` can generate arbitrarily many tasks by varying the random seed, tool combinations, argument values, and DAG structures. The 200-task suite (seed 42) was designed to balance coverage across levels, categories, and tool combinations while keeping evaluation cost manageable.

What data does each instance consist of? Each instance is a JSON object with the following fields:

- `task_id`: Unique identifier encoding level and index (e.g., `L0_node_0009`).
- `level`: Integer 0–3 indicating composition level.
- `prompt`: Natural-language task description.
- `available_tools`: Array of tool schemas (OpenAI function-calling format) including correct tools and distractors.
- `expected_trace`: Array of expected tool calls with arguments, outputs, and execution order.
- `metadata`: Object containing DAG structure, step count, tool categories, and generation parameters.

Is there a label or target associated with each instance? Yes. The `expected_trace` field serves as the ground truth. It specifies the exact tool calls (names + arguments) and their execution order (sequential, parallel, or DAG). Scoring compares model outputs against this trace along four dimensions: tool sequence, argument accuracy, completeness, and data flow.

Is any information missing from individual instances? No. Every task is fully self-contained: the prompt, available tools, and expected trace are all present. Tasks do not depend on external state, databases, or prior conversation history.

Are relationships between individual instances made explicit? Tasks are independent by design; there are no inter-task dependencies. Tasks *within* a level share structural properties (e.g., all L_2 tasks have a fork-join topology), which is encoded in the `level` field and `metadata`.

Are there recommended data splits? No. COMPTOOLBENCH is an evaluation-only benchmark. All 200 tasks form a single test set. The benchmark is not designed for training or fine-tuning. Users who wish to create development splits for method development can regenerate tasks with alternative seeds using the `CompositionEngine`.

Are there any errors, sources of noise, or redundancies? All 106 tools produce deterministic outputs via hash-based simulation, eliminating execution noise. A human validation study (Appendix I) verified task correctness by manually checking a sample of tasks for prompt–trace consistency. No known errors remain after this validation pass.

Is the dataset self-contained, or does it link to or otherwise rely on external resources? The dataset is fully self-contained. All tool executions use deterministic simulations that require no network access, API keys, or external services. The tool schemas are bundled with each task instance.

Does the dataset contain data that might be considered confidential? No. All data is synthetically generated. No real user queries, API responses, personal information, or proprietary content is included.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? No. Task prompts describe standard software operations (weather lookup, text processing, calculations, scheduling). No offensive, harmful, or sensitive content is present.

Does the dataset relate to people? No. Tasks involve synthetic tool interactions, not human subjects or personal data. Some tasks use placeholder names or email addresses (e.g., `iris@media.news`) that are entirely fictional.

J.3 Collection Process

How was the data associated with each instance acquired? All data was synthetically generated by the `CompositionEngine`, a deterministic Python program. The engine: (1) selects tool combinations from the 106-tool catalog based on compatibility rules, (2) constructs a DAG specifying execution order and data flow, (3) generates a natural-language prompt describing the task, (4) simulates deterministic tool execution to produce expected outputs, and (5) assembles the complete task object with metadata.

What mechanisms or procedures were used to collect the data? The `CompositionEngine` runs as a single Python script (`scripts/generate_tasks.py`) with seed 42. No web scraping, crowdsourcing, API calls, or manual annotation was involved. The generation process is fully reproducible: running the same script with the same seed produces identical output.

If the dataset is a sample from a larger set, what was the sampling strategy? The 200-task suite was generated to balance coverage across the four composition levels (48/64/40/48) and 15 tool categories. The level distribution reflects the different structural constraints: L_1 has more tasks (64) because sequential chains have fewer structural degrees of freedom than DAGs, requiring more instances for adequate coverage. L_2 has fewer (40) because fork-join tasks are more uniform in structure.

Who was involved in the data collection process and how were they compensated? Only the author (Md A Rahman) was involved. No external annotators, crowdworkers, or data labelers participated.

Over what timeframe was the data collected? The task suite was generated in February 2026. The `CompositionEngine` produces all 200 tasks in under 10 seconds on commodity hardware.

Were any ethical review processes conducted? No formal ethical review was required or conducted, as the dataset contains only synthetic data with no human subjects, personal information, or sensitive content.

J.4 Preprocessing, Cleaning, and Labeling

Was any preprocessing/cleaning/labeling of the data done? The `CompositionEngine` produces clean, structured JSON output directly; no post-hoc cleaning was needed. A human validation step verified a sample of tasks for prompt–trace consistency (see Appendix I).

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data? The generated output *is* the raw data. No intermediate or "raw" form exists because generation and output are a single step.

Is the software that was used to preprocess/clean/label the data available? Yes. The complete `CompositionEngine` source code is available at https://github.com/ronyrahmaan/comptoolbench under the MIT license.

J.5 Uses

Has the dataset been used for any tasks already? Yes. The dataset was used in the experiments reported in this paper: evaluating 27 LLMs (9 frontier + 10 free cloud + 8 local) on compositional tool-use tasks. The primary finding is the Selection Gap: the vast majority of models score higher on composed tasks than on single-tool selection.

Is there a repository that links to any or all papers or systems that use the dataset? The GitHub repository (https://github.com/ronyrahmaan/comptoolbench) will maintain a list of papers and systems that use CompToolBench.

What (other) tasks could the dataset be used for? Potential uses include: (1) evaluating new LLMs on compositional tool use, (2) benchmarking tool-calling middleware and agent frameworks, (3) studying the effect of tool catalog size on selection accuracy (by varying the number of distractors), (4) analyzing error patterns in function-calling implementations, and (5) training tool-use classifiers or retrieval models (using the task–tool mappings as supervision).

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? Two characteristics are relevant: (1) All tools are deterministic simulations, so the benchmark does not capture challenges of live API execution (errors, latency, format variability). (2) Tasks are English-only and reflect common software utility patterns, which may not generalize to domain-specific or multilingual tool ecosystems.

Are there tasks for which the dataset should not be used? CompToolBench should not be used to: (1) claim general "tool-use ability" without acknowledging the simulated execution environment, (2) train production tool-routing systems without validating on live APIs, or (3) compare models that were fine-tuned on the benchmark against those that were not (contamination risk).

J.6 Distribution

Will the dataset be distributed to third parties outside of the entity on behalf of which it was created? Yes. The dataset is publicly available with no access restrictions.

How will the dataset be distributed? The dataset is distributed through two persistent platforms:

- **HuggingFace Datasets:** https://huggingface.co/datasets/ronyrahmaan/comptoolbench
- **GitHub:** https://github.com/ronyrahmaan/comptoolbench

A Zenodo archive provides a DOI-linked persistent copy: https://zenodo.org/records/18907784. Croissant machine-readable metadata (JSON-LD) is provided for automated discovery and loading.

When will the dataset be distributed? The dataset has been publicly available since March 2026.

Will the dataset be distributed under a copyright or other intellectual property (IP) license? The dataset is released under the Creative Commons Attribution 4.0 International License (CC BY 4.0). The evaluation code is released under the MIT License.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances? No. All data is original synthetic content. No third-party data, copyrighted material, or licensed content is included.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? No.

J.7 Maintenance

Who will be supporting/hosting/maintaining the dataset? Md A Rahman (Texas Tech University) is the sole maintainer. The dataset is hosted on HuggingFace (persistent hosting with built-in versioning) and GitHub (version-controlled repository).

How can the owner/curator/manager of the dataset be contacted? Via email: `ara02434@ttu.edu`, or through the GitHub repository's issue tracker.

Is there an erratum? Not currently. Any corrections will be documented in the GitHub repository's `CHANGELOG.md` and reflected in new HuggingFace dataset versions.

Will the dataset be updated? The task suite (seed 42, 200 tasks) is fixed for reproducibility. Future versions may add: (1) additional tasks with new seeds, (2) expanded tool catalogs, (3) multilingual prompts, or (4) live API evaluation modes. Updates will be versioned (current: v2.0.0) with clear changelogs.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances? Not applicable. The dataset does not relate to people or contain personal data.

Will older versions of the dataset continue to be supported/hosted/maintained? Yes. HuggingFace Datasets supports dataset versioning, and the Zenodo archive provides a permanent DOI-linked snapshot. Older versions will remain accessible.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? Yes. The `CompositionEngine` source code is open-source (MIT), allowing users to generate custom task suites with different seeds, tool catalogs, or composition constraints. Contributions to the main repository are welcome via GitHub pull requests.

NeurIPS Paper Checklist

1. **Claims.** Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   [Yes] The abstract and introduction state three contributions: (1) the CompToolBench benchmark with 200 tasks across 106 tools at four DAG-based composition levels, (2) evaluation of 27 models (9 frontier + 10 free cloud + 8 local) revealing the Selection Gap (13.2 pp average), and (3) open release of code, data, and evaluation infrastructure. All three are fully supported by the experimental results (Section 5) and the public repository. Claims are scoped to the specific models and tasks evaluated; we do not claim universal generalization.

2. **Limitations.** Does the paper discuss the limitations of the work performed by the authors?

   [Yes] Section 7 enumerates seven specific limitations: simulated (not live) tool execution, single-turn protocol, limited task coverage (200 templates), English-only prompts, scoring asymmetry between L_0 and L_1–L_3, finite model sample, and the cloud–local confound. Each limitation is stated explicitly with its implications.

3. **Theory, Assumptions and Proofs.** For each theoretical result, does the paper provide the full set of assumptions and a complete proof?

   [NA] This is an empirical benchmark paper. No formal theorems or proofs are presented. The CompGap metric (Equation 1) is defined algebraically; its validity is empirically demonstrated through the scoring ablation study (Section 5.5).

4. **Experimental Result Reproducibility.** Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper?

   [Yes] The Reproducibility Statement details: deterministic tool simulations (hash-based outputs), fixed random seed (42) for task generation, open-source code and data on GitHub and HuggingFace, standardized inference via LiteLLM with temperature 0.0 and 60-second timeouts, and documented environment specifications. The full 200-task suite with ground-truth traces is publicly available. Of 27 models, 18 can be reproduced at zero cost (10 free-tier cloud APIs + 8 local Ollama); 9 frontier models require approximately $15 in API costs.

5. **Open access to data and code.** Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   [Yes] Code: `https://github.com/ronyrahmaan/comptoolbench` (MIT license). Data: `https://huggingface.co/datasets/ronyrahmaan/comptoolbench` (CC BY 4.0). The repository includes a `Makefile` with targets for task generation (`make generate`), evaluation (`make evaluate`), and analysis (`make analyze`). A

`pyproject.toml` with pinned dependencies and `README.md` with step-by-step instructions are provided.

6. **Experimental Setting/Details.** Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen) necessary to understand the results?

   [Yes] Section 4 specifies: all 27 model names and providers, inference configuration (temperature 0.0, 60-second timeout, single-turn protocol), the full task generation procedure (CompositionEngine with seed 42), scoring weights per level (Section 3.4), and compute infrastructure (MacBook Pro M4 Pro for local models, free-tier cloud APIs). No training or fine-tuning is performed; this is a pure evaluation benchmark.

7. **Experiment Statistical Significance.** Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   [Yes] Bootstrap 95% confidence intervals (10,000 resamples) are reported for the primary Selection Gap finding. The ablation study reports Spearman ρ with p-values (Table 4). Since tool simulations are deterministic and temperature is 0.0, there is no run-to-run variance for individual model evaluations; the bootstrap CIs quantify uncertainty over the task sample.

8. **Experiments Compute Resources.** For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   [Yes] The Reproducibility Statement and Section 4 specify: local inference on Apple MacBook Pro M4 Pro (24 GB RAM), cloud inference via free-tier APIs (Groq, Cerebras, Cohere, Mistral, OpenRouter), and frontier model evaluation costing approximately $15 total. Per-model average latency is reported in Appendix D. Total evaluation time for all 27 models is approximately 4 hours.

9. **Code of Ethics.** Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   [Yes] The research involves no human subjects, no personal data, and no sensitive or dual-use tools. All models are evaluated through their public APIs or open-weight releases. The Ethics Statement discusses two potential concerns (adversarial prompt design implications and English-only coverage) and explains our mitigations.

10. **Broader Impacts.** Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    [Yes] The Ethics Statement discusses: (1) positive impact: understanding tool-selection limitations helps improve agent safety and reliability; (2) negative risk: the Selection Gap finding could theoretically inform adversarial prompt design, though we argue the defensive value outweighs this risk; (3) limitation: English-only evaluation excludes multilingual communities.

11. **Safeguards.** Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse?

    [NA] CompToolBench releases a benchmark (not a model) consisting of synthetic tool-use tasks with deterministic simulations. The 106 tools are standard software utilities (weather lookup, text processing, calculation) with no safety-critical, dual-use, or offensive capabilities. No personal data, biometric data, or sensitive content is included. The risk of misuse is negligible.

12. **Licenses for existing assets.** Are the creators or original owners of assets (e.g., code, data, models) used in the paper properly credited and are the license and terms of use explicitly mentioned and properly respected?

    [Yes] All models are cited in Section 4.1 and the bibliography. LiteLLM [BerriAI, 2024] is cited as the inference framework. All cloud APIs are used within their terms of service (free tiers). Open-weight models (Llama, Mistral, Qwen, Granite) are used under their respective community licenses for research evaluation. No training data from third parties is used.

13. **New Assets.** Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

[Yes] The benchmark is released with: (1) a comprehensive `README.md` with installation and usage instructions; (2) a HuggingFace dataset card describing all fields, splits, and intended uses; (3) Croissant machine-readable metadata (JSON-LD) conforming to the MLCommons Croissant 1.0 specification; (4) a Datasheet for Datasets (Gebru et al., 2021) included as a paper appendix; (5) CC BY 4.0 license for data and MIT license for code.

14. **Crowdsourcing and Research with Human Subjects.** For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots?

    [NA] No crowdsourcing was performed. All 200 tasks are synthetically generated by the `CompositionEngine`. A human validation study (Appendix I) was conducted by the author to verify task correctness (not as a human-subjects experiment); it involved no external participants, compensation, or consent requirements.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects.** Does the paper describe potential risks incurred by study participants, whether combensation was adequate given the risks, and whether Institutional Review Board (IRB) approval (or equivalent) was obtained?

    [NA] No human subjects were involved in this research. All tasks are synthetic and all evaluations are automated. No IRB approval was required or sought.

16. **Declaration of LLM usage.** Does the paper describe the usage of LLMs in the core methodology?

    [Yes] LLMs are the *subjects* of evaluation, not tools used to generate the benchmark or write the paper. Section 4.1 lists all 27 evaluated models with exact version identifiers and inference providers. The evaluation protocol (Section 4) describes how models interact with the benchmark via standardized function-calling APIs. The CompositionEngine that generates tasks is a deterministic Python program, not an LLM-based generator.