MCP-AgentBench: Evaluating Real-World Language Agent Performance with MCP-Mediated Tools

Zikang Guo[1*], Benfeng Xu[1], Chiwei Zhu[1], Wentao Hong[2], Xiaorui Wang[2], Zhendong Mao[1,3†]

[1]University of Science and Technology of China
[2]Metastone Technology, Beijing, China
[3]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
{gzk170401, benfeng}@mail.ustc.edu.cn, zdmao@ustc.edu.cn

Abstract

The Model Context Protocol (MCP) is rapidly emerging as a pivotal open standard, designed to enhance agent-tool integration and interoperability, and is positioned to unlock a new era of powerful, interconnected, and genuinely utilitarian agentic AI. However, despite MCP's growing adoption, existing benchmarks often fail to capture real-world agent performance within this new paradigm, leading to a distorted perception of their true operational value and an inability to reliably differentiate proficiencies. To bridge this critical evaluation gap, we introduce MCP-AgentBench—a comprehensive benchmark specifically engineered to rigorously assess language agent capabilities in MCP-mediated tool interactions. Core contributions of MCP-AgentBench include: the establishment of a robust MCP testbed comprising 33 operational servers with 188 distinct tools; the development of a benchmark featuring 600 systematically designed queries distributed across 6 distinct categories of varying interaction complexity; and the introduction of MCP-Eval, a novel outcome-oriented evaluation methodology prioritizing real-world task success. Through extensive empirical evaluation of leading language agents, we provide foundational insights. MCP-AgentBench aims to equip the research community with a standardized and reliable framework to build, validate, and advance agents capable of fully leveraging MCP's transformative benefits, thereby accelerating progress toward truly capable and interoperable AI systems.

Introduction

Language agents built on large language models (LLMs) are moving beyond single-turn dialogue toward autonomous tool use across the web, operating systems, and software ecosystems (Sumers et al. 2023; Wang et al. 2024; Guo et al. 2024; Su et al. 2024; Zhou et al. 2023; Pan et al. 2024; Xue et al. 2025; Xie et al. 2024; Niu et al. 2024; Anthropic 2024; Yang et al. 2024; Liu et al. 2024b). This shift promises goal-directed interaction in real environments, but it also exposes a central obstacle: achieving scalable interaction with heterogeneous tools and systems.

Beyond agent-level competence in multi-step tool workflows, including intent grounding, orchestration, and recovery (Patil et al. 2024; Qin et al. 2023; Yang et al. 2024), practical deployments encounter an ecosystem-level M×N integration problem: M agents must interface with N disparate tools, driving pairwise adapters, duplicated schemas, and inconsistent error semantics that undermine interoperability and scalability.

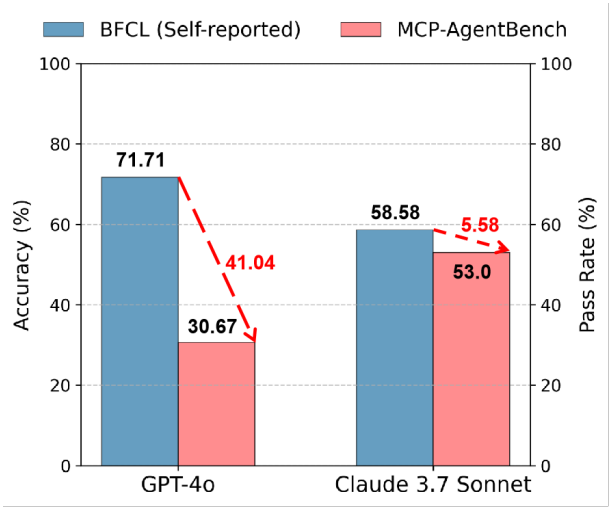

Figure 1: BFCL (Yan et al. 2024) vs. MCP-AgentBench

The Model Context Protocol (MCP) (Anthropic 2025b) addresses these challenges by providing an open, protocol-level interface between agents and tool servers. Compared with traditional function calling paradigms (Qin et al. 2023; Yao et al. 2024), MCP (1) abstracts interaction at the server level, enabling multi-operation exchanges that may maintain context per protocol specification; (2) standardizes environment feedback, yielding richer and more actionable responses necessary for robust autonomous loops; and (3) supports more dynamic capability discovery, reducing bespoke schema engineering and improving adaptability across tools. These properties make MCP a promising foundation for interoperable, protocol-driven agent systems (Hou et al. 2025).

However, the emergence of MCP calls for a rethinking of agent evaluation. Existing function-calling benchmarks (Yan et al. 2024; Qin et al. 2023; Yao et al. 2024; Zhong et al. 2025) were not designed for standardized, protocol-level in-

*Work done during the internship at Metastone.
†Corresponding author.

teractions and therefore are a poor fit for MCP-mediated activities. By emphasizing per-call correctness against fixed JSON schemas, they undermeasure end-to-end effectiveness and fail to reliably differentiate capabilities, creating a critical evaluation gap that slows both MCP-based system development and systematic progress. For example, scores on BFCL (Yan et al. 2024), which targets a narrower scope of tool interactions (Figure 1), may misrepresent practical efficacy on standardized server workflows. Reports from real-world usage, including OpenRouter (OpenRouter 2025), indicate that models such as Claude 3.7 Sonnet (Anthropic 2025a) can outperform GPT-4o (OpenAI 2025a) on complex tool-assisted tasks, yet their BFCL rankings can diverge substantially. These discrepancies highlight the limitations of current benchmarks for protocol-driven evaluation and risk mischaracterizing model capabilities.

To bridge this critical evaluation gap, we introduce MCP-AgentBench, a comprehensive benchmark for assessing language agents in MCP-mediated tool use. It is built on a unified MCP server testbed that provides a realistic and standardized environment, comprising 33 MCP-compliant servers exposing 188 tools. Servers are selected for executability, initial statelessness to support reproducibility, and text-based interaction.

MCP-AgentBench features systematically categorized queries spanning a spectrum of interaction complexities, from single-server operations to multi-server sequential workflows requiring sophisticated planning and information synthesis. Data construction follows a three-stage pipeline: (1) curate and deploy the MCP server testbed, (2) generate diverse queries using a structured categorization scheme that varies the number of servers, dependency patterns, and (3) create reference answers through an LLM-assisted, human-verified annotation process.

To measure performance, we introduce MCP-Eval, an LLM-as-a-judge methodology (Zheng et al. 2023) tailored to MCP-AgentBench. The evaluation prioritizes real-world effectiveness by scoring end-to-end task success under MCP semantics rather than exact action traces, acknowledging that complex problems admit multiple valid solution paths.

Our contributions are threefold:

1. We introduce a unified server testbed that integrates diverse MCP-compliant servers. This provides a standardized and robust environment for reproducible MCP research and agent development.
2. We present **MCP-AgentBench**, a comprehensive benchmark built upon our testbed, featuring 600 queries across 6 distinct categories. These tasks are designed to rigorously probe complex agent-tool interaction patterns. To measure performance, we introduce **MCP-Eval**, an LLM-as-a-judge protocol that scores end-to-end task success under MCP semantics.
3. We provide a detailed evaluation of leading language agents using MCP-AgentBench. Our analysis yields foundational insights into current agent capabilities, highlights critical limitations, and identifies key challenges, thereby charting clear directions for future research in this domain.

MCP-AgentBench

This section details the development of MCP-AgentBench, our novel benchmark for evaluating agents in MCP environments. We first describe the comprehensive data construction pipeline, which encompasses the establishment of the MCP server testbed, the methodology for query generation, and the process for reference answer annotation. Subsequently, we present key statistics of the resulting dataset and conclude by introducing MCP-Eval, our automated framework for evaluating agent performance on this benchmark.

Data Construction Methodology

The construction of MCP-AgentBench followed a rigorous, multi-stage methodology, illustrated in Figure 2. This process involved three primary stages: (1) establishing the MCP server testbed, (2) generating diverse user queries, and (3) annotating corresponding reference answers. An LLM (specifically, the Claude 3.7 Sonnet model (Anthropic 2025a)) provided automated assistance throughout these procedures. To ensure high data quality, executability, and overall reliability, each stage of this process—from server curation and query generation to answer annotation—incorporated a meticulous human-in-the-loop verification procedure. Human annotators validated key aspects and outputs, thereby establishing a robust foundation for the benchmark and subsequent evaluations.

Establishing the MCP Server Testbed: Rigorous Curation, Deployment, and Integration The foundational phase in developing MCP-AgentBench was the meticulous construction of a robust and diverse MCP server testbed. This process began with a rigorous curation procedure, involving an extensive survey of official MCP registries and community resources that initially identified 369 candidate MCP servers. From this large pool, a stringent manual filtering process retained only servers that met our strict criteria of being readily executable and stable, stateless in their operational paradigm, and primarily reliant on text-based input and output as defined by the MCP. The focus on text-based interactions stemmed from the complexities associated with the quantitative evaluation of non-textual modalities. Furthermore, stateful servers, which require persistent state across independent benchmark queries, were deemed unsuitable for our framework due to the intricate requirements for managing initial states, tracking temporal changes, and handling state resets, all of which impede reproducible assessment. This meticulous screening, accomplished by a team of three people over seven days, yielded 33 servers that met our stringent criteria.

Following curation, the 33 selected servers underwent careful deployment and configuration in strict accordance with their official operational guidelines. Subsequently, for seamless and standardized agent interaction, all deployed servers were consolidated under a unified invocation interface. This was achieved by leveraging mcprouter (ChatMCP Team 2025), a tool that abstracts server-specific operational idiosyncrasies and facilitates uniform communication protocols. This comprehensive process culminated in an operational testbed comprising 33 distinct MCP servers, which

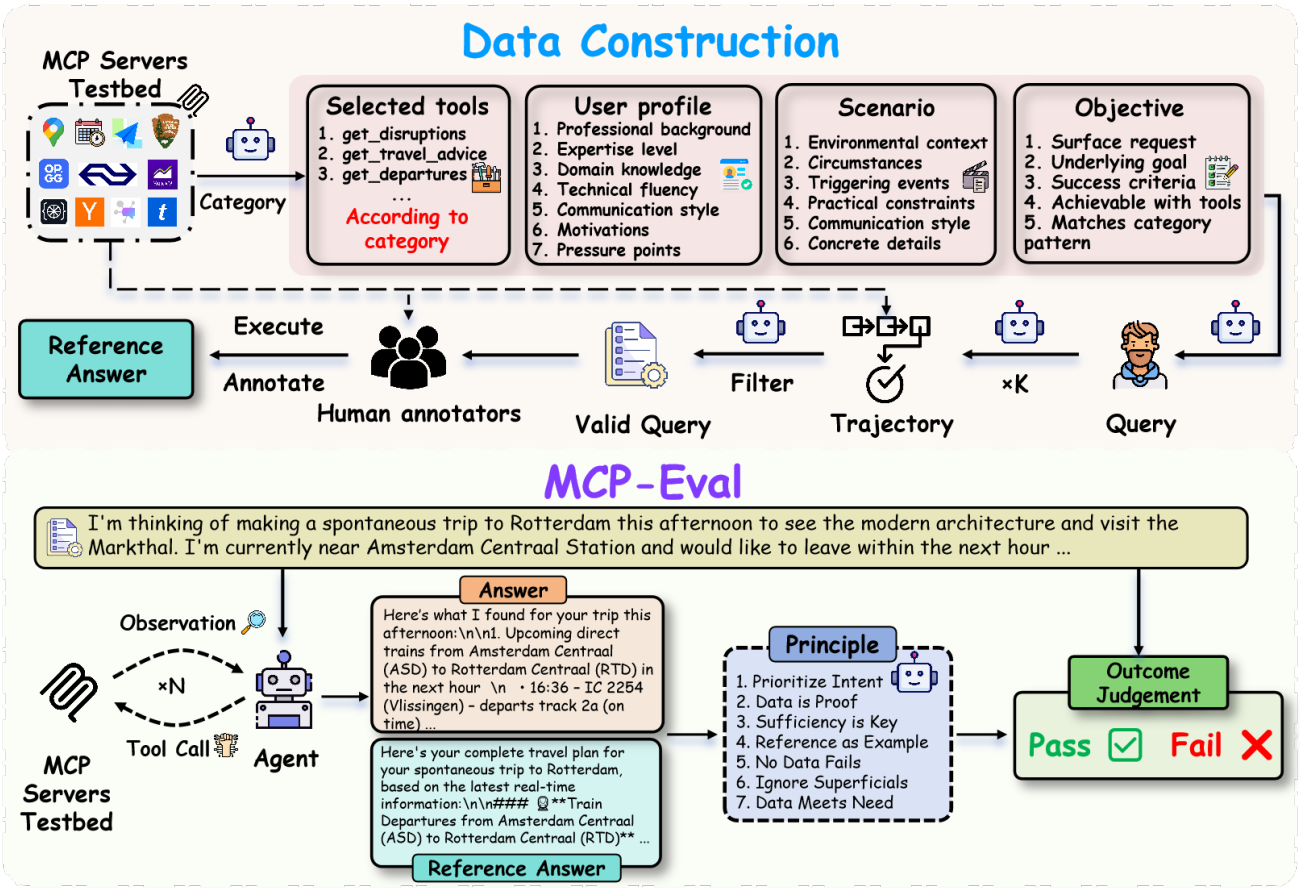

Figure 2: The construction and evaluation workflow of MCP-AgentBench. Top (Data Construction): (1) A unified MCP server testbed is established. (2) Diverse, categorized queries are generated with LLM assistance and human verification. (3) Reference answers are produced via an expert-in-the-loop framework. Bottom (MCP-Eval): An LLM-as-a-judge performs an outcome-oriented evaluation of the agent's final answer, yielding a pass/fail score based on task completion.

collectively provide access to 188 unique tools, as detailed in Figure 3.

Query Generation To address the limitations of existing benchmarks in controlling complexity and ensuring realism, we developed an advanced query generation methodology. This methodology employs an LLM to construct test queries that are valid, verifiable, realistic, and diverse. The process unfolds in three core phases. **First**, we establish a structured categorization framework to systematically vary interaction complexity. **Second**, we leverage the LLM, guided by a detailed set of instructions, to generate a coherent set of contextual components, including necessary tools, user profiles, background scenarios, and explicit objectives. **Finally**, these components are synthesized into naturalistic user queries by the LLM, followed by a rigorous human verification process to produce the final, validated queries for our dataset.

Category Definition The objective of the first phase is to systematically control and vary the complexity of interaction patterns within our test queries. To this end, we establish a categorization framework defined by two primary dimensions: **server scope** and **call dependency**. Server scope distinguishes whether task resolution is confined to a single server or requires coordination across multiple servers. Call dependency differentiates tasks based on whether they require a single tool invocation, concurrent invocations of multiple tools (parallel calls), or a sequence of dependent invocations (sequential calls). The combination of these dimensions yields six concrete categories: *Single-Server-Single-Call*, *Single-Server-Parallel-Call*, *Single-Server-Sequential-Call*, and their *Multi-Server* counterparts, which together span a spectrum from simple requests to complex, multi-step workflows.

Contextual Component Generation In the second phase, the LLM generates a set of contextual components tailored to a specific category from our framework. Guided by a comprehensive prompt, the model produces a single, structured output containing four internally consistent elements:

Tool Selection The model identifies the specific set of tools from the testbed required to fulfill the user's objective, ensuring the chosen tools are sufficient and produce deterministic, verifiable outputs aligned with the category's complexity.

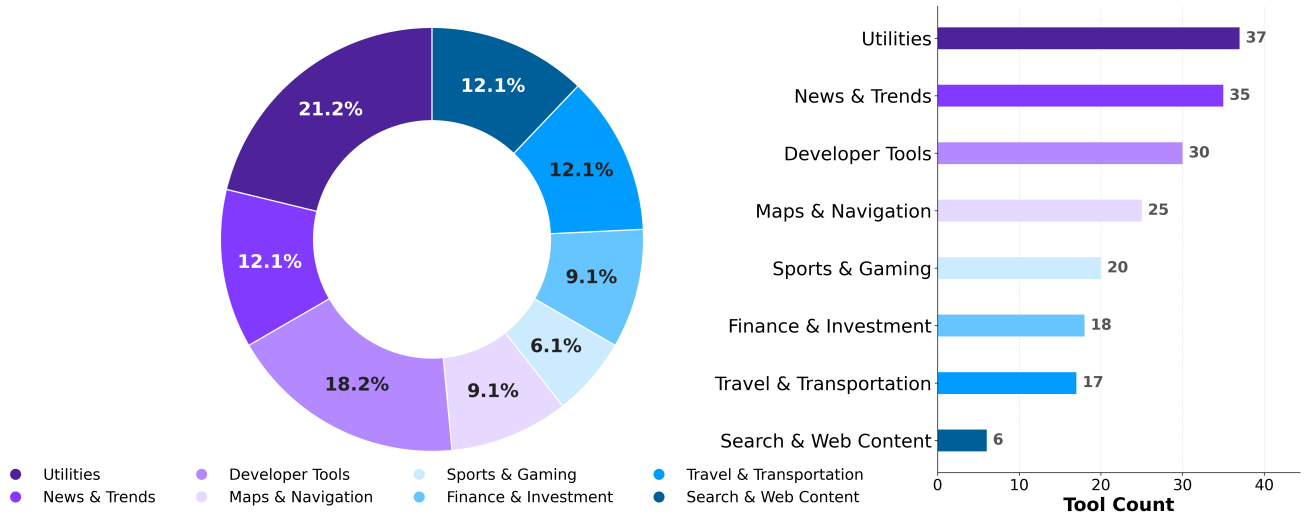

Figure 3: Characterization of the MCP server distribution in our testbed. Left: topic-based distribution of the MCP servers; Right: number of tools available per topic across these servers.

User Profile The model generates a detailed user persona, including professional background and communication style, to ground the query in a realistic context.

Scenario Description The model establishes a narrative context that motivates the user's need. Scenarios are calibrated for complexity and include concrete details that naturally yield the necessary parameters for the selected tools.

User Objective The model explicitly defines the user's intended goal and verifiable success criteria. A critical constraint is that each objective must be fully achievable using only the tools selected in the same generation step. This unified generation process ensures that all contextual components are coherent and aligned with the complexity defined by the chosen category.

Final Query Synthesis In this phase, the structured contextual components are synthesized into a naturalistic user query. This is accomplished by leveraging the LLM as a *User Simulator*, guided by a comprehensive prompt that orchestrates the generation process. The prompt instructs the LLM to generate an utterance that is authentic to the specified user's persona in communication style and vocabulary, and is also carefully constructed to be informationally self-contained for single-turn resolution. Furthermore, the query's structure must naturally imply the required tool interaction pattern (e.g., parallel or sequential calls) of its designated category. Crucially, the generated query remains strictly goal-oriented, focusing on the user's objective while avoiding any reference to the underlying tools or the process of task execution. The resulting output is a high-fidelity simulated user query, which then proceeds to the reference answer annotation stage.

Reference Answer Annotation The final stage in the construction of our benchmark is the annotation of a gold-standard reference answer for each query. Recognizing that the manual annotation of complex, multi-step agent tasks is both labor-intensive and requires significant domain expertise, we devised a hybrid, expert-in-the-loop framework. This methodology is engineered to maximize annotation quality and scalability by strategically directing human effort toward the most challenging cases, ensuring the overall robustness of the resulting benchmark.

Our annotation process unfolds in a two-tiered sequence, beginning with a broad automated assessment. First, for every query, an LLM is leveraged to generate K distinct execution trajectories. These trajectories are then programmatically evaluated by a separate LLM-based judge to compute an aggregate pass rate. Queries that fall below a predetermined success threshold are automatically isolated and escalated for manual intervention. This initial screening efficiently filters out straightforward cases and pinpoints queries that are ambiguous, inherently difficult, or expose model limitations.

In the second stage, these flagged, low-pass-rate queries are subjected to a meticulous review by human experts. The expert's role extends beyond simple verification; they perform a diagnostic analysis to determine the root cause of failure. Based on their findings, the experts undertake one of several corrective actions: (1) revising the source query itself to resolve ambiguity or logical flaws, (2) refining a syntactically correct but semantically suboptimal LLM-generated answer, or (3) authoring a definitive reference answer from scratch. This targeted approach ensures that the most complex and nuanced instances within MCP-AgentBench are backed by human-validated, high-quality reference solutions. The final benchmark is thus composed of these curated query-answer pairs. We deliberately abstain from annotating execution trajectories as ground truth, as multiple valid solution paths often exist for a given task. For any queries involving time-sensitive information, the reference answer reflects a correct outcome as captured at the time of annotation.

Data Statistics

The resulting MCP-AgentBench dataset comprises **600** unique queries, uniformly distributed across the six interaction categories defined in our methodology, with 100 queries allocated to each category. This balanced design ensures comprehensive coverage of diverse interaction patterns and complexity levels. Furthermore, given that the tool-calling interface of most LLMs typically supports at most 128 tools, we employ randomized server selection within the multi-server categories so that the overall number of instantiated tools remains close to this practical limit, even though not all candidate servers are necessarily utilized.

MCP-Eval

To facilitate a scalable and objective assessment of agent performance on MCP-AgentBench, we introduce MCP-Eval, an automated evaluation framework. MCP-Eval deliberately prioritizes the correctness of the final outcome over the intermediate execution trajectory, a design choice that acknowledges an agent's capacity for self-correction and the existence of multiple valid solution paths. The primary performance metric is the overall **Pass Rate**, which measures the fraction of queries an agent, M_{eval}, successfully resolves across the benchmark $\mathcal{B}$:

$$\text{Pass Rate} = \frac{\sum_{i=1}^{N} \mathbb{I}(J^{(i)} = \text{Pass})}{N}$$

where $N = |\mathcal{B}|$, $J^{(i)}$ is the binary judgment for test instance i, and $\mathbb{I}(\cdot)$ is the indicator function. We employ a binary pass/fail metric to provide a clear and verifiable signal of success, aligning with how real-world tasks naturally resolve into discrete outcomes. Fine-grained metrics, by contrast, depend on subjective evaluations expressed in natural language and lack objective verification. In open-ended task environments, such scores can be unreliable and are more susceptible to manipulation.

Our framework implements this outcome-oriented evaluation using an LLM-as-a-judge paradigm (Zheng et al. 2023), employing a designated LLM, L_{judge} (in our case, *o3-mini* (OpenAI 2025b)). We choose o3-mini as the judge for its demonstrated robustness and its growing adoption as an automatic evaluator in recent benchmarks such as Humanity's Last Exam and PaperBench (Phan et al. 2025; Starace et al. 2025), which suggests that its behavior is relatively stable and well-understood in evaluation settings. For each test instance, the judge assesses the model's generated answer $A^{(i)}_{model}$ against the user query $Q^{(i)}_{final}$ and the reference answer $A^{(i)}_{ref}$. This process is formalized as:

$$J^{(i)} = L_{judge}(Q^{(i)}_{final}, A^{(i)}_{ref}, A^{(i)}_{model}, \mathcal{I}_{judge})$$

The judge's decision is guided by a comprehensive prompt, $\mathcal{I}_{judge}$, which operationalizes a set of core principles to ensure objective and realistic assessment by prioritizing functional success over procedural mimicry. Key tenets include: **prioritizing the user's core intent** to define success; treating the **presence of specific, external data as conclusive evidence** of tool use, making its absence a primary failure criterion; emphasizing **sufficiency**, where meeting the core need is valued over exhaustive detail; and **disregarding superficial aspects** such as formatting or verbosity when comparing against the reference answer. By adhering to these instructions, L_{judge} provides a scalable and consistent method for measuring practical agent effectiveness within the flexible and potentially non-deterministic context of complex tool interactions.

Experiments

Experimental Setup

We conducted a comprehensive evaluation of 10 representative LLMs on MCP-AgentBench, encompassing both state-of-the-art proprietary systems and prominent open-source architectures. All models evaluated in this study were accessed uniformly through their respective official APIs. These include proprietary systems such as Anthropic (*claude-3.7-sonnet*, *claude-4-sonnet*) (Anthropic 2025a), OpenAI (*o3-mini-high*, *gpt-4o-2024-11-20*) (Achiam et al. 2023), and Google (*gemini-2.5-flash*, *gemini-2.5-pro*) (Comanici et al. 2025), alongside leading open-source models including Qwen (*qwen3-235b-a22b-thinking-2507*) (Team 2025), MoonshotAI (*kimi-k2*) (Team et al. 2025), and DeepSeek (*deepseek-v3-0324*, *deepseek-r1-0528*) (Liu et al. 2024a; Guo et al. 2025).

The ReAct framework served as the primary mechanism for enabling agent interaction. Additionally, native tool calling (TC) was employed for models possessing this inherent capability. However, Gemini's tool calling mode was not evaluated due to a formatting incompatibility between the tool descriptions on our deployed servers and the model's parser, which prevented successful execution. Agent activity for each query was constrained to a maximum of 30 actions, which could comprise either tool calls or textual responses. The interaction loop concludes and the agent formulates its final answer when a model output contains only text and no further tool calls, signaling that all necessary tool-assisted operations are considered complete. We evaluate the deep reasoning capabilities of advanced LLMs, such as Claude and Gemini, by configuring their thinking budget to 8192.

Main Results

Model Comparison The results in Table 1 reveal a surprising and significant trend: the leading open-source models demonstrate exceptional capabilities, rivaling and even surpassing their proprietary counterparts. Most notably, Qwen3-235B-A22B, using the ReAct framework, achieved the highest overall score in the entire benchmark, challenging the prevailing narrative of proprietary model dominance. Among other open-source models, Kimi K2 also delivered robust results, particularly excelling in TC mode. Within the proprietary category, Anthropic's Claude 4 Sonnet emerged as the top performer, showing a clear advantage when using its native TC capabilities. OpenAI's o3-mini also proved to be a consistent performer across both modes. In stark contrast, GPT-4o underperformed significantly in all tested scenarios, indicating potential limitations in its agentic reasoning for these tasks.

Type	Method	Model	Single Server			Multi Server			Avg.
			Single	Parallel	Sequential	Single	Parallel	Sequential	
Proprietary	ReAct	GPT-4o	47	35	30	33	13	9	27.8
		o3-mini	56	**63**	41	48	**50**	**49**	**51.2**
		Claude 3.7 Sonnet	60	52	51	58	32	39	48.7
		Claude 4 Sonnet	**62**	50	**57**	**66**	27	33	49.2
		Gemini 2.5 Flash	54	54	42	63	30	34	46.2
		Gemini 2.5 Pro	58	53	43	58	38	44	49.0
	TC	GPT-4o	53	41	25	38	12	15	30.7
		o3-mini	**64**	52	41	66	33	44	50.0
		Claude 3.7 Sonnet	63	48	47	69	39	52	53.0
		Claude 4 Sonnet	61	**54**	**52**	**74**	**53**	**54**	**58.0**
Open-source	ReAct	Kimi K2	70	64	**69**	63	52	41	59.8
		DeepSeek V3	59	58	55	61	36	45	52.3
		DeepSeek R1	75	57	53	58	42	40	54.2
		Qwen3-235B-A22B	**78**	**69**	**69**	**68**	**54**	**50**	**64.7**
	TC	Kimi K2	63	**61**	**65**	**70**	**48**	**59**	**61.0**
		DeepSeek V3	59	38	35	58	36	26	42.0
		Qwen3-235B-A22B	**68**	57	**65**	20	14	17	40.2

Table 1: Main results of MCP-AgentBench (hierarchical structure). **Bold** denotes the best score within each sub-group. **Avg.** reflects the overall performance.

Method Comparison A comparison between the ReAct and TC modes highlights that model performance is highly dependent on the interaction framework, with no universally superior option. The most striking instance is Qwen3-235B-A22B, which leads the benchmark with ReAct but suffers a drastic performance collapse in TC mode. This failure often stems from the model not generating a tool call when one is required, leading to premature termination and an incorrect final answer. Conversely, models like Claude 4 Sonnet show a marked improvement with TC over ReAct, indicating its architecture is finely tuned for this interaction style. This variance underscores the critical importance of selecting a model-appropriate framework to unlock an agent's maximum potential.

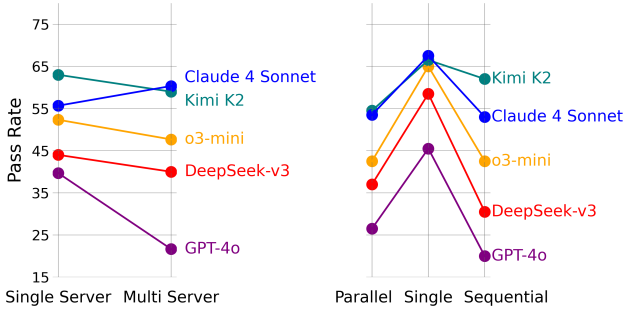

Figure 4: Model pass rates under different server scopes and call dependencies, illustrating varying task difficulty.

Analysis

Subsequent analysis in this section will exclusively utilize results from the native TC mode for models supporting it.

Demonstrating Varying Task Difficulty via Server Scope and Call Dependency MCP-AgentBench's graduated difficulty is evident across two dimensions. First, performance generally declines as tasks transition from Single Server to Multi Server scopes. Second, a similar drop occurs as call dependency increases from simple Single to complex Sequential calls. Claude 4 Sonnet, however, is a notable exception to the first trend, showing an improved pass rate on more challenging multi-server tasks. Our analysis indicates this anomaly occurs because the model's tendency to fail on simpler tasks by incorrectly relying on its parametric knowledge is mitigated when greater complexity compels reliable tool engagement. This hierarchy of challenges confirms MCP-AgentBench's ability to test a wide spectrum of agentic capabilities, as shown in Figure 4.

Token Efficiency Figure 5 illustrates the trade-off between model performance and token consumption. It reveals that the models with the highest pass rates, Kimi K2 (61.0%) and Claude 4 Sonnet (58.0%), are also by far the most costly, consuming 101.7k and 140.3k tokens per query, respectively. This higher consumption is largely attributable to their use of a thinking mode for internal reasoning before generating a response. In contrast, o3-mini demonstrates exceptional efficiency. It achieves a strong 50.0% pass rate with a token cost of only 36.5k, which is comparable to models with significantly lower performance, such as GPT-4o and DeepSeek-v3. This disparity highlights that while higher performance often correlates with increased token usage, some models like o3-mini offer a significantly more cost-effective balance.

Consistency between MCP-Eval and Human Evaluation We assessed the consistency between MCP-Eval outputs and human evaluations, alongside inter-rater reliability. Three

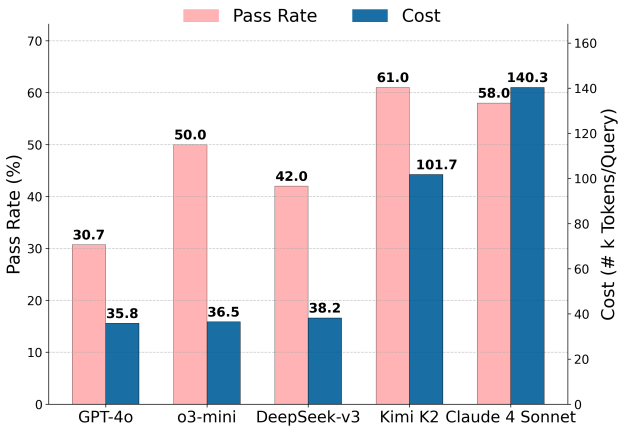

Figure 5: Model performance vs. token consumption.

human experts annotated 60 items, comprising 10 randomly selected items from each of six categories derived from Claude 3.7 Sonnet outputs. This annotation task required each expert to dedicate approximately 2.5 hours (150 minutes) in total, averaging 2.5 minutes per item. The percentage agreement between MCP-Eval and the aggregated human majority vote reached 91.67%, with a Cohen's Kappa of 0.734. For inter-rater reliability on pass/fail decisions, Fleiss' Kappa among the experts was 0.671. The overall three-way percentage agreement for all valid expert judgments stood at 86.67%. This strong concordance validates MCP-Eval's alignment with human evaluative standards and confirms the robustness of our human-annotated dataset.

Error Analysis Based on our evaluation of agent performance on MCP-AgentBench, we identified several recurring categories of errors. This analysis provides insight into the common failure modes of LLM agents in protocol-driven, tool-use scenarios.

- **Misinterpretation of Query**: The agent fails to accurately capture the user's primary objective or misconstrues critical semantic details and constraints embedded within the query.
- **Refusal to Use Tool**: The agent improperly defaults to its parametric knowledge, providing a response from its internal memory when the task explicitly necessitates tool invocation for accessing external, dynamic, or proprietary data sources.
- **Omission of Key Information**: The agent delivers an incomplete response, either by failing to address all components of the user's request or by neglecting to synthesize and incorporate essential information obtained from tool outputs in previous steps of a multi-turn interaction.
- **Hallucination**: The agent fabricates information, presenting factually incorrect details or assertions that are either unsupported by or directly contradict the evidence provided by the tool outputs.

Related Work

LLM Agents and Standardized Interaction The proliferation of sophisticated LLM-based agents, capable of complex reasoning and planning (Huang and Chang 2022; Huang et al. 2024), has unlocked new capabilities in domains like web navigation, software engineering, and GUI automation (Wang et al. 2024; Guo et al. 2024; Zhang et al. 2024). A key enabler for these agents is their ability to interact with external tools (Qin et al. 2023; Patil et al. 2024), a process facilitated by frameworks like AutoGen (Wu et al. 2023) and MetaGPT (Hong et al. 2024). However, scaling these interactions presents a significant M×N integration problem. In response, standardized communication frameworks like the Model Context Protocol (MCP) (Anthropic 2025b) have been developed to provide a universal layer between agents and tools. The adoption of such protocols necessitates new benchmarks specifically designed to evaluate agent performance within these structured environments.

Benchmarking Protocol-Aware Tool Use Existing tool-use benchmarks, such as ToolBench (Qin et al. 2023) and API-Bank (Li et al. 2023), have been instrumental in assessing general agent capabilities. Yet, they were not designed to measure agent effectiveness within the context of a standardized protocol, a critical aspect for real-world deployment. While evaluations for the MCP ecosystem are emerging, they have key limitations. MCP-Bench (Luo et al. 2025), for instance, evaluates the performance of MCP servers, not the proficiency of agents. Others, such as MCPEval (Liu et al. 2025) and MCP-RADAR (Gao et al. 2025), are constrained by a limited scope of servers and specific domains, which compromises their comprehensiveness. This leaves a critical gap for a large-scale benchmark that assesses agents across a diverse and operational MCP infrastructure. To address this, we introduce MCP-AgentBench, a comprehensive benchmark designed to rigorously evaluate agent performance in such environments.

Conclusion

This work addresses the evaluation gap for language agents within the Model Context Protocol (MCP) ecosystem through three main contributions: (1) MCP server testbed, a large-scale operational infrastructure (33 servers, 188 tools) that offers a realistic, standardized environment; (2) MCP-AgentBench, a benchmark of 600 diverse queries for rigorous assessment on this testbed; and (3) MCP-Eval, an outcome-oriented LLM-as-a-judge methodology focused on tangible task success. Together, these components provide a unified framework and empirical insights for building capable, interoperable MCP-based systems. Nonetheless, MCP-AgentBench has limitations: MCP-Eval still depends on LLM judgments, which may introduce residual biases, and the current testbed centers on stateless, text-based servers, omitting long-lived, stateful, and multimodal workflows common in practice. Extending MCP-AgentBench to cover stateful servers and multimodal tools is a key direction for more comprehensive evaluation of agentic capabilities.

Acknowledgments

We thank the reviewers for their insightful comments and constructive suggestions, which have substantially improved the quality of this paper. We are also grateful to Jinshuai Zhang and An Liu for their invaluable support with the infrastructure. This research is supported by Artificial Intelligence-National Science and Technology Major Project 2023ZD0121200 and National Natural Science Foundation of China under Grant 62222212.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. https://www.anthropic.com/news/3-5-models-and-computer-use.

Anthropic. 2025a. Claude 3.7 Sonnet. https://www.anthropic.com/claude/sonnet.

Anthropic. 2025b. Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol.

ChatMCP Team. 2025. MCPRouter: API Router for MCP Servers. https://github.com/chatmcp/mcprouter.

Comanici, G.; Bieber, E.; Schaekermann, M.; Pasupat, I.; Sachdeva, N.; Dhillon, I.; Blistein, M.; Ram, O.; Zhang, D.; Rosen, E.; et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Gao, X.; Xie, S.; Zhai, J.; Ma, S.; and Shen, C. 2025. MCP-RADAR: A Multi-Dimensional Benchmark for Evaluating Tool Use Capabilities in Large Language Models. *arXiv preprint arXiv:2505.16700*.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Guo, T.; Chen, X.; Wang, Y.; Chang, R.; Pei, S.; Chawla, N. V.; Wiest, O.; and Zhang, X. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S. K. S.; Lin, Z.; Zhou, L.; Ran, C.; Xiao, L.; Wu, C.; and Schmidhuber, J. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *The Twelfth International Conference on Learning Representations*.

Hou, X.; Zhao, Y.; Wang, S.; and Wang, H. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*.

Huang, J.; and Chang, K. C.-C. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Huang, X.; Liu, W.; Chen, X.; Wang, X.; Wang, H.; Lian, D.; Wang, Y.; Tang, R.; and Chen, E. 2024. Understanding the planning of LLM agents: A survey. *arXiv preprint arXiv:2402.02716*.

Li, M.; Zhao, Y.; Yu, B.; Song, F.; Li, H.; Yu, H.; Li, Z.; Huang, F.; and Li, Y. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. arXiv:2304.08244.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Liu, J.; Wang, K.; Chen, Y.; Peng, X.; Chen, Z.; Zhang, L.; and Lou, Y. 2024b. Large language model-based agents for software engineering: A survey. *arXiv preprint arXiv:2409.02977*.

Liu, Z.; Qiu, J.; Wang, S.; Zhang, J.; Liu, Z.; Ram, R.; Chen, H.; Yao, W.; Wang, H.; Heinecke, S.; et al. 2025. MCPEval: Automatic MCP-based Deep Evaluation for AI Agent Models. *arXiv preprint arXiv:2507.12806*.

Luo, Z.; Shi, X.; Lin, X.; and Gao, J. 2025. Evaluation report on mcp servers. *arXiv preprint arXiv:2504.11094*.

Niu, R.; Li, J.; Wang, S.; Fu, Y.; Hu, X.; Leng, X.; Kong, H.; Chang, Y.; and Wang, Q. 2024. Screenagent: A vision language model-driven computer control agent. *arXiv preprint arXiv:2402.07945*.

OpenAI. 2025a. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/.

OpenAI. 2025b. OpenAI o3-mini. https://openai.com/index/openai-o3-mini/.

OpenRouter. 2025. LLM Rankings. https://openrouter.ai/rankings.

Pan, Y.; Kong, D.; Zhou, S.; Cui, C.; Leng, Y.; Jiang, B.; Liu, H.; Shang, Y.; Zhou, S.; Wu, T.; et al. 2024. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*.

Patil, S. G.; Zhang, T.; Wang, X.; and Gonzalez, J. E. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565.

Phan, L.; Gatti, A.; Han, Z.; Li, N.; Hu, J.; Zhang, H.; Zhang, C. B. C.; Shaaban, M.; Ling, J.; Shi, S.; et al. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Starace, G.; Jaffe, O.; Sherburn, D.; Aung, J.; Chan, J. S.; Maksin, L.; Dias, R.; Mays, E.; Kinsella, B.; Thompson, W.; et al. 2025. PaperBench: Evaluating AI's Ability to Replicate AI Research. *arXiv preprint arXiv:2504.01848*.

Su, Y.; Yang, D.; Yao, S.; and Yu, T. 2024. Language Agents: Foundations, Prospects, and Risks. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, 17–24.

Sumers, T.; Yao, S.; Narasimhan, K.; and Griffiths, T. 2023. Cognitive architectures for language agents. *Transactions on Machine Learning Research*.

Team, K.; Bai, Y.; Bao, Y.; Chen, G.; Chen, J.; Chen, N.; Chen, R.; Chen, Y.; Chen, Y.; Chen, Y.; et al. 2025. Kimi K2: Open Agentic Intelligence. *arXiv preprint arXiv:2507.20534*.

Team, Q. 2025. Qwen3 Technical Report. arXiv:2505.09388.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; Awadallah, A. H.; White, R. W.; Burger, D.; and Wang, C. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv:2308.08155.

Xie, T.; Zhang, D.; Chen, J.; Li, X.; Zhao, S.; Cao, R.; Hua, T. J.; Cheng, Z.; Shin, D.; Lei, F.; et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37: 52040–52094.

Xue, T.; Qi, W.; Shi, T.; Song, C. H.; Gou, B.; Song, D.; Sun, H.; and Su, Y. 2025. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*.

Yan, F.; Mao, H.; Ji, C. C.-J.; Zhang, T.; Patil, S. G.; Stoica, I.; and Gonzalez, J. E. 2024. Berkeley Function Calling Leaderboard. https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html.

Yang, J.; Jimenez, C.; Wettig, A.; Lieret, K.; Yao, S.; Narasimhan, K.; and Press, O. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37: 50528–50652.

Yao, S.; Shinn, N.; Razavi, P.; and Narasimhan, K. 2024. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint arXiv:2406.12045*.

Zhang, C.; He, S.; Qian, J.; Li, B.; Li, L.; Qin, S.; Kang, Y.; Ma, M.; Liu, G.; Lin, Q.; et al. 2024. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36: 46595–46623.

Zhong, L.; Du, Z.; Zhang, X.; Hu, H.; and Tang, J. 2025. ComplexFuncBench: Exploring Multi-Step and Constrained Function Calling under Long-Context Scenario. arXiv:2501.10132.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.