Help or Hurdle? Rethinking Model Context Protocol-Augmented Large Language Models

Anonymous ACL submission

Abstract

The Model Context Protocol (MCP) enables large language models (LLMs) to access external resources on demand. While commonly assumed to enhance performance, how LLMs actually leverage this capability remains poorly understood. We introduce MCPGAUGE, the first comprehensive evaluation framework for probing LLM–MCP interactions along four key dimensions: proactivity (self-initiated tool use), compliance (adherence to tool-use instructions), effectiveness (task performance post-integration), and overhead (computational cost incurred). MCPGAUGE comprises a 160-prompt suite and 25 datasets spanning knowledge comprehension, general reasoning, and code generation. Our large-scale evaluation spans six commercial LLMs, 30 MCP tool suites, and both one- and two-turn interaction settings, comprising around 20,000 API calls. This comprehensive study reveals four key findings that challenge prevailing assumptions about the effectiveness of MCP integration. These insights highlight critical limitations in current AI–tool integration and position MCPGAUGE as a principled benchmark for advancing controllable, tool-augmented LLMs.

1 Introduction

The prospect of autonomous AI agents that can seamlessly access diverse tools and data sources has gained considerable traction. However, this landscape remains fragmented. Each tool requires bespoke interface definitions, authentication handling, and execution logic, and function-calling APIs differ across platforms (Hou et al., 2025; Krishnan, 2025). As a result, AI agents are often constrained by static, hard-wired workflows rather than dynamically discovering tools at runtime.

To address these issues, Anthropic released the Model Context Protocol (MCP) in late 2024 (Anthropic, 2024). MCP aims to streamline AI development and enhance flexibility in managing intricate workflows by standardizing interfaces and enabling agents to dynamically discover, select, and coordinate external services. Since its release, MCP has transformed from a protocol into a fundamental building block of AI agents, supported by a vibrant community ecosystem of MCP tools that provide connectivity to web search engines, structured databases, file systems, and custom APIs. Instead of requiring LLMs to internalize every piece of knowledge within their parameters, MCP separates retrieval and execution from generation: the LLM issues a "tool call" (e.g., a web search), receives back structured snippets, and continues reasoning with that injected context, aiming to improve accuracy and strengthen reasoning.

Research Gap. While MCP provides promising infrastructure for tool integration, a significant gap persists between its theoretical benefits and practical usefulness. The reason is that the final performance on various tasks depends on not only the extra context provided by MCP but also LLMs' capacity to recognize when external tools are needed, execute MCP calls appropriately, and effectively utilize the retrieved information. Although recent studies have examined MCP's architecture (Hou et al., 2025; Singh et al., 2025; Ray, 2025), security concerns (Radosevich and Halloran, 2025; Narajala and Habler, 2025), and MCP tools' efficiency of requesting resources (Luo et al., 2025), a critical gap remains in understanding how LLMs engage with MCP. Existing benchmarks like MCP-RADAR (Gao et al., 2025) evaluate only task performance outcomes without examining the behavioral aspects of how LLMs recognize tool needs, execute calls, and integrate retrieved information.

Research Questions. Given this gap, we examine LLM-MCP interaction through four key RQs:

RQ1: Do LLMs proactively invoke tools provided by MCP when such actions could improve task performance, without explicit user instructions?

RQ2: To what extent do LLMs follow explicit user instructions to use MCP tools?

RQ3: How does external context retrieved via MCP tools affect LLM task performance?

RQ4: What is the computational overhead, measured in input token increase, associated with MCP tool integration?

Challenges. Answering these research questions presents two key challenges. First, the community lacks a clear set of axes for measuring how well LLMs use MCP tools. We therefore define four complementary dimensions, including **proactivity**, **compliance**, **effectiveness**, and **overhead**, to measure how well a LLM can proactively invoke a tool (RQ1), how well it obeys explicit directives (RQ2), to what extent the external context helps (RQ3), and how much it costs (RQ4). Second, existing LLM benchmarks were designed for stand-alone models and seldom require MCP tool use. We build a 160-prompt suite to assess tool recognition for proactivity and compliance, and adapt established LLM benchmarks on knowledge comprehension, general reasoning, and code generation with explicit MCP instructions to evaluate effectiveness.

By addressing the challenges, we introduce MCPGAUGE, the first systematic evaluation framework for LLM–MCP interaction. MCPGAUGE consists of a newly-designed suite of 160 prompts and 25 well-established task datasets.

We used MCPGAUGE to evaluate six commercial LLMs equipped with 30 MCP tool suites across both one-turn and two-turn dialogue settings. Our analysis revealed four surprising and insightful findings that challenge common assumptions about the effectiveness of MCP integration: (1) Most models exhibit minimal proactive use of MCP tools in the first turn, but tool-invocation rates increase substantially in two-turn dialogues. (2) Instruction compliance improves only when tool-use directives are embedded within incremental dialogue, indicating a limited ability to follow single-shot commands. (3) Contrary to expectations, under default zero-shot integration, automated MCP access by LLMs reduces accuracy by an average of 9.5% across the six LLMs on three core task categories, revealing non-trivial friction between retrieved context and the model's internal reasoning. (4) MCP integration introduces substantial computational overhead: input-token volume increases by $3.25\times$ to $236.5\times$ across models and tasks. These findings highlight critical bottlenecks in current LLM–MCP interactions, offering valuable guidance for future research and practical AI system design, particularly for developing more efficient and controllable MCP-augmented LLM agents.

We make the following contributions:

- We propose MCPGAUGE, the first comprehensive framework for empirically evaluating LLM interactions with MCP tools. MCPGAUGE includes a 160-prompt suite and 25 datasets covering knowledge comprehension, general reasoning, and code generation tasks.
- We introduce four evaluation dimensions, including proactivity, compliance, effectiveness, and overhead, to systematically assess an MCP-enabled LLM's ability to initiate tool use, follow explicit instructions, utilize retrieved context to improve performance, and manage the computational cost of tool integration.
- Our large-scale evaluation (around 20,000 LLM API calls, over USD 6,000) yields several key insights into current LLM–MCP integration: proactive tool use usually requires a brief conversational context; multi-turn interaction improves instruction compliance; misaligned retrieved context can impair effectiveness; and tool use incurs substantial input-token overhead.

To facilitate open science and future research, we provide the code and raw experiment data as supplementary materials and will open-source them upon paper acceptance, including the full 160-prompt diagnostic suite, all augmented benchmark prompts, MCP server configurations, and evaluation scripts.

2 Background and Related Work

2.1 Model Context Protocol

Model Context Protocol (MCP) is a communication standard that bridges AI models and external resources through a tripartite architecture of hosts, clients, and servers. The *host* is the AI application environment (e.g., Claude Desktop, Cursor IDE) that runs the client; the *client* bridges the host and servers, managing requests and coordinating tool invocations; and the *server* exposes external capabilities through tools (external operations and API calls), resources (structured and unstructured data), and prompts (reusable workflow templates). In a typical workflow, the client receives a user prompt, collaborates with the server to identify and access relevant tools, and returns the processed results to the user.

2.2 Related Work

Multiple surveys (Hou et al., 2025; Singh et al., 2025; Ray, 2025) have examined the MCP and its emerging ecosystem. Hou et al. (2025) provide a broad landscape analysis that examines MCP's architecture and applications while highlighting the absence of systematic evaluation frameworks. Singh et al. (2025) focus on standardization efforts for LLM enhancement, while Ray (2025) examine MCP's current applications and challenges.

As MCP adoption has expanded, security implications have emerged as a critical concern. Radosevich and Halloran (2025) conduct a comprehensive safety audit revealing major security exploits possible when LLMs interact through MCP, demonstrating that even well-aligned models can be compromised in MCP environments. This work underscores the importance of systematic evaluation frameworks like ours for understanding LLM behavior in tool-augmented settings. Building on these security concerns, Narajala and Habler (2025) propose enterprise-grade security frameworks and mitigation strategies for MCP implementations, addressing the challenges of deploying MCP in production environments while maintaining security standards. Finally, Song et al. (2025) provide a detailed analysis of specific attack vectors and vulnerability exploitation methods in MCP systems.

In parallel with these security investigations, the evaluation of LLMs with the MCP paradigm has taken several distinct directions. Luo et al. (2025) introduce MCPBench, a systematic evaluation framework for assessing MCP server performance, but does not comprehensively evaluate LLM capabilities in MCP utilization. Gao et al. (2025) introduce MCP-RADAR, a benchmark solely focused on measuring performance outcomes through evaluating LLM tool-use capabilities across answer accuracy, tool selection efficiency, computational resource efficiency, parameter construction accuracy, and execution speed. In contrast to MCP-RADAR, which targets tool-use *outcomes* and pipeline behavior (e.g., tool selection efficiency, parameter correctness, and execution speed), MCPGauge decomposes LLM–MCP interaction into four orthogonal *behavioral* dimensions—proactivity, compliance, effectiveness, and overhead—thereby isolating how LLMs recognize tool necessity and integrate retrieved context rather than the quality of the tool-use pipeline itself.

Although prior research has investigated MCP's architecture, security, and server performance, they only focus on the MCP side. A systematic analysis of interactions between LLMs and MCP is lacking. This work addresses this gap by introducing four new dimensions for evaluating LLM–MCP interactions and applying them to assess widely used LLMs and MCP tools. The results reveal key bottlenecks in current MCP-augmented LLM systems and offer valuable insights for future research.

3 MCPGauge Design

MCPGauge systematically assesses LLM-MCP interactions through a tailored 160-prompt suite and 25 datasets covering knowledge comprehension, general reasoning, and code generation tasks, across four MCP-specific dimensions. Figure 1 demonstrates an overview of MCPGauge.

3.1 Evaluation Dimensions

We introduce four MCP-specific evaluation dimensions, **proactivity**, **compliance**, **effectiveness**, and **overhead** to examine LLM-MCP interactions:

Proactivity is quantified as the proportion of task scenarios where LLMs correctly identify the need for external tools and successfully invoke them. It measures whether LLMs demonstrate self-directed awareness of their knowledge limitations, for example, needing real-time web search for current events, and proactively initiate appropriate MCP tool calls without explicit instruction (RQ1).

Compliance measures LLMs' adherence to explicit user instructions regarding MCP tool usage. When users instruct LLMs to use MCP tools by mentioning MCP generally (e.g., "use MCP to get today's weather") or by specifying particular tools (e.g., "use web search tool to find current stock prices"), we evaluate whether LLMs follow instructions to execute the requested tool calls (RQ2).

Effectiveness quantifies the improvement in output quality when LLMs utilize MCP-retrieved context. We assess how external context impacts performance by comparing outputs with and without MCP integration (RQ3).

Overhead captures the computational cost introduced by MCP integration. Concretely, we compare the total number of input tokens that LLMs must process with and without MCP, thereby quantifying the additional resource footprint that MCP-augmented reasoning imposes (RQ4).

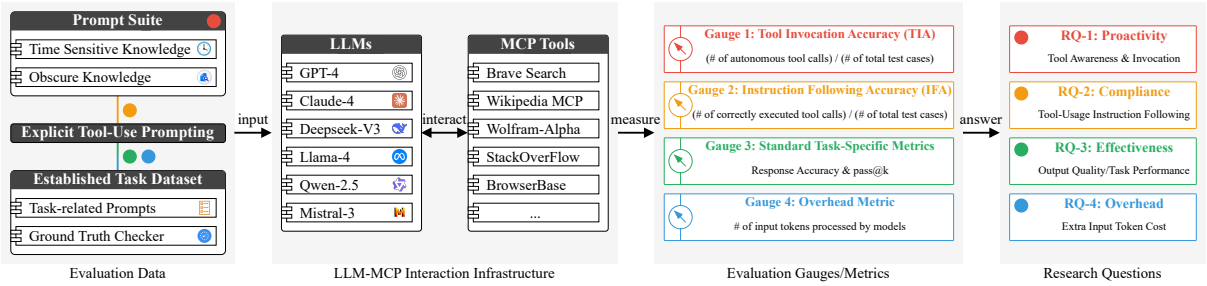

Figure 1: Framework of MCPGauge.

3.2 Design Rationale

3.2.1 Dimension-specific Design

For proactivity, we construct scenarios where MCP calls are necessary but not explicitly requested, crafting queries that require time-sensitive knowledge (e.g., current weather, recent news) or obscure knowledge (e.g., small companies), so that the answer cannot come from the LLM's parametric knowledge and external context is needed (Figures 2 and 4). For compliance, we augment prompts with explicit MCP usage instructions, such as "use MCP calls to find/search/retrieve ..." (Figures 3 and 5). For effectiveness, we compare performance on identical tasks with and without MCP; for overhead, we tally the full input-token workload (user prompt, system instructions, and retrieved context) each LLM processes during inference.

3.2.2 Conversation-depth Settings

We evaluate proactivity and compliance under two dialogue depths. The *one-turn* setting records the model's first response, mirroring the single-shot scenario where users expect an immediate answer; the *two-turn* setting appends a follow-up query that mimics real-world dialogue refinement. A preliminary study showed marginal gains beyond two turns, so we focus on one- and two-turn conversations and leave deeper interactions to future work.

3.3 Evaluation Datasets

To evaluate LLM–MCP interactions, we construct a 160-prompt suite for assessing proactivity and compliance, and employ 25 benchmark datasets to measure effectiveness and overhead.

For proactivity, existing LLM benchmarks (Liu et al., 2023; Zhong et al., 2023) are inadequate because they include task specifications that do not require tools. We therefore construct a 160-prompt suite that deliberately withholds any tool mention yet requires MCP calls for task completion according to the design rationale discussed previously.

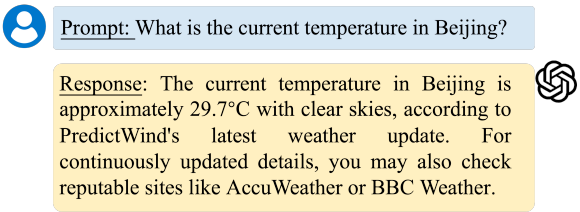

Figure 2: Example of a one-turn conversation without explicit MCP call invocation.

For compliance, we reuse the same 160 prompts but insert explicit instructions for invoking MCP calls into the prompts, as illustrated in Figure 3.

For effectiveness and overhead, we draw on 25 existing well-established LLM benchmarks. The reason is that they provide ground-truth for performance evaluation. The benchmark datasets cover three task domains: knowledge comprehension, general reasoning, and code generation. For knowledge comprehension (KC), we utilize two datasets from HellaSwag (Zellers et al., 2019): ActivityNet, which provides video captions, and WikiHow, which consists of how-to articles for completion tasks. For general reasoning (GR), we incorporate 21 datasets from AGI-Eval (Zhong et al., 2023), including test datasets (SAT Math, SAT English, LSAT Analytical Reasoning, LSAT Logical Reasoning, LSAT Reading Comprehension), mathematical reasoning tasks (AQUA-RAT, Math), logical reasoning datasets (LogiQA-EN, LogiQA-ZH), comprehensive academic assessments from Gaokao covering various subjects (Chinese, English, Geography, History, Biology, Chemistry, Physics, Math QA, Math Cloze), and legal reasoning tasks (JEC-QA-KD, JEC-QA-CA). For code generation (CG), we include two datasets from EvalPlus (Liu et al., 2023): HumanEval and MBPP, which focus on Python programming tasks such as algorithm implementation and code debugging.

It is worth noting that all 25 datasets have their own user prompts and queries. Therefore, the 160-prompt suite is not used here. Instead, similar to

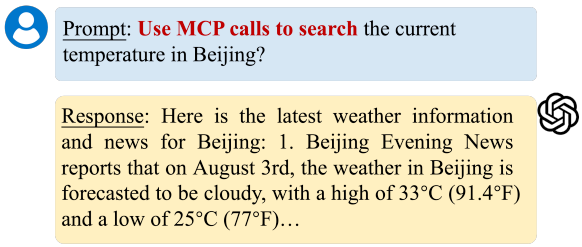

Figure 3: Example of a one-turn conversation with explicit MCP call invocation.

how we modify the prompts for evaluating compliance, we insert explicit instructions for calling MCP tools into the prompts in these benchmarks to increase the frequency of LLM-MCP interactions during the task-solving process. Concretely, we prepend a single instruction sentence, "Use MCP tools to answer the query.", to the original prompt while leaving the task content unchanged; for example, an original prompt "Akira came directly, breaking all tradition. . . " becomes "Use MCP tools to answer the query. Akira came directly, breaking all tradition. . . ". We do not modify answer formats, evaluation scripts, dataset structure, or ground-truth labels. Moreover, since LLMs may not choose to use MCP tools even when explicitly instructed, we only collect and analyze the task-solving instances where MCP calls are made by LLMs for the purpose of studying the impact of MCP calls on performance and overhead. Consequently, the effectiveness (Table 1) and overhead (Table 3) results are *conditional* analyses computed over instances with successfully executed MCP calls, while the corresponding tool-invocation frequencies under explicit instruction are characterized by the compliance (IFA) results in Section 4.2.

3.4 Evaluation Metrics

We define formal metrics to quantitatively evaluate LLM–MCP interactions along the four proposed dimensions:

Tool Invocation Accuracy (TIA): Measures the proportion of test cases where the LLM autonomously invokes an appropriate MCP tool without being explicitly instructed. Formally,

$$\text{TIA} = \frac{N_{\text{autonomous}}}{N_{\text{total}}}$$

where $N_{\text{autonomous}}$ is the number of test cases in which the LLM proactively initiates a correct MCP tool call, and N_{total} is the total number of tool-dependent test cases.

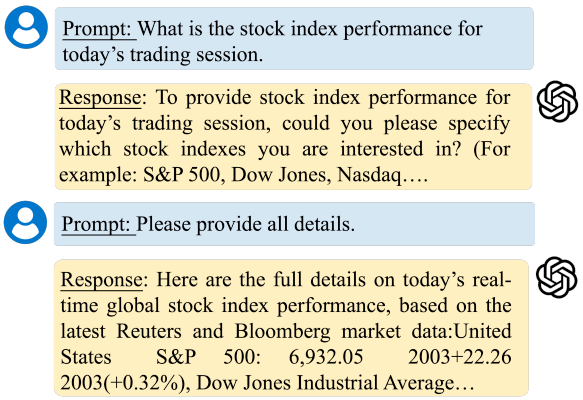

Figure 4: Example of a two-turn conversation without explicit MCP call invocation.

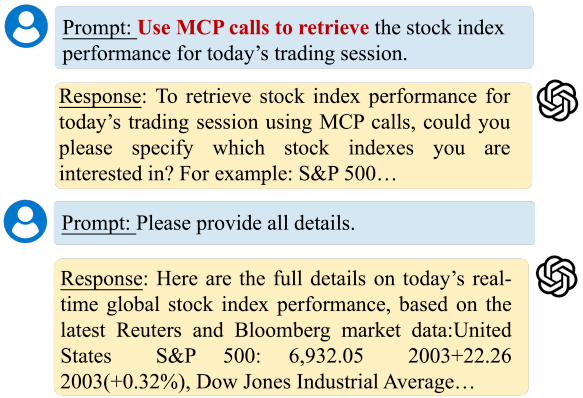

Figure 5: Example of a two-turn conversation with explicit MCP call invocation.

Instruction Following Accuracy (IFA): Measures the LLM's adherence to explicit tool-use instructions embedded in prompts. Formally,

$$\text{IFA} = \frac{N_{\text{compliant}}}{N_{\text{total}}}$$

where $N_{\text{compliant}}$ denotes the number of test cases where the LLM correctly executes the instructed MCP tool call, and N_{total} is the total number of instruction-containing prompts.

Effectiveness (Acc, pass@k): Assesses the improvement in task performance when MCP-retrieved context is integrated. We use domain-appropriate metrics:

For knowledge and reasoning tasks:

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}}$$

where N_{correct} is the number of task responses matching ground-truth answers.

For code generation tasks, we adopt the unbiased pass@k metric as defined in (Chen et al., 2021).

Overhead (Token Cost): Quantifies the computational cost introduced by MCP integration in terms of input token volume. Formally,

$$\text{Overhead Ratio} = \frac{T_{\text{with-MCP}}}{T_{\text{without-MCP}}}$$

where $T_{\text{with-MCP}}$ and $T_{\text{without-MCP}}$ represent the total input tokens (including user prompt, instructions, and retrieved context) with and without MCP, respectively.

4 Experiments

4.1 Experiment Setup

Large Language Models. We evaluate six commercial LLMs: GPT-4 (GPT-4.1-2025-04-14), Claude-4 (Claude-4-sonnet-2025-05-14), DeepSeek-V3, Llama-3.3 (Llama-3.3-70B-Instruct-Turbo), Qwen-2.5 (Qwen2.5-VL-72B-Instruct), and Mistral-3 (Mistral-medium-latest). **All models are accessed via their official APIs with function (tool) calling enabled, using the default configurations provided by the corresponding vendors.**

MCP Tools. We integrate 30 MCP tool suites, providing diverse information retrieval domains, including general web search MCP tools (e.g., `brave_web_search`), general knowledge-focused retrieval MCP services (e.g., `wikipedia_mcp`), mathematical problems-related searching MCP tools (e.g., `wolfram-alpha`), and programming questions-related retrieval MCP tools (e.g., `stackoverflow_mcp`). We manually verify that the 30 MCP tools are enough for covering all the need for the tasks in the 160-prompt suite and the 25 benchmark datasets. A list of the used MCP tools are provided in the appendix (Section A.1).

Computation Platform. All LLM inference is performed through the official vendor APIs; no model is run locally. A single NVIDIA RTX 3090 GPU is used only for local evaluation scripts, metric computation, prompt preprocessing, and hosting certain MCP tool backends, and therefore does not affect model inference or any reported result.

4.2 Main Results

Proactivity. We evaluate six commercial LLMs' autonomous MCP tool recognition through the tailored 160-prompt suite designed to necessitate real-time information access, e.g., current weather, without explicit MCP tool usage instructions. We measure **Tool Invocation Accuracy (TIA)**, the proportion of cases where models successfully recognize tool necessity and initiate appropriate MCP calls across one-turn and two-turn dialogue settings.

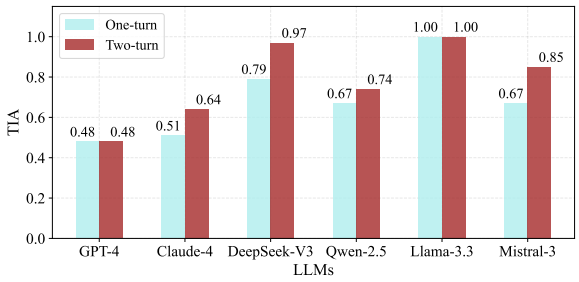

Figure 6: **Proactivity (RQ1).** Autonomous invocation of tools by six LLMs on tool-dependent prompts, compared between one-turn and two-turn dialogue settings.

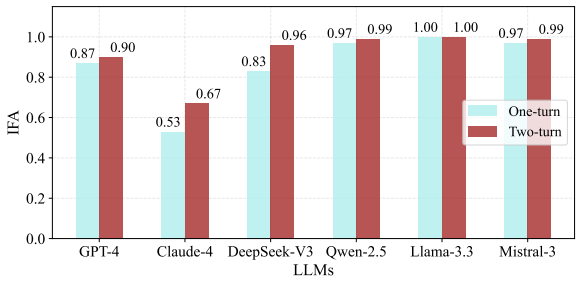

Figure 7: **Compliance (RQ2).** Adherence of six LLMs to explicit MCP tool-use instructions in prompts, compared between one-turn and two-turn dialogue settings.

> **Finding** ☞ Multi-turn interaction substantially increases MCP tool-invocation rates: most LLMs show limited proactive tool use in one-turn scenarios while achieving substantially improved proactivity in two-turn interactions.

As shown in Figure 6, most LLMs exhibit limited autonomous tool recognition in one-turn settings (e.g., TIA of 0.48 for GPT-4 and 0.51 for Claude-4), but a follow-up turn yields remarkable gains: Claude-4, DeepSeek-V3, Qwen-2.5, and Mistral-3 improve by 25.5%, 22.8%, 10.4%, and 26.9%, respectively, while Llama-3.3 sustains a perfect TIA of 1.0 across both settings and GPT-4 remains unchanged. These results indicate that a second conversational turn substantially increases tool-invocation rates, turning a mostly unused capacity into proactive behaviour for nearly all LLMs.

Compliance. We next assess these six LLMs' instruction-following capabilities for explicit MCP tool usage through a tailored 160-prompt suite but augmented with direct tool usage directives. We measure **Instruction Following Accuracy (IFA)** (Section 3.4), the proportion of cases where LLMs follow explicit tool usage instructions across one-turn and two-turn conversation scenarios.

Model	Setting	KC(Acc↑)		GR(Acc↑)							CG(pass@k↑)	
		ActivityNet	WikiHow	SAT	LSAT	Gaokao	LogiQA	JEC	AQUA	MATH	HumanEval	MBPP
GPT-4	w/o MCP	0.80	0.94	0.90	0.87	0.77	0.77	0.60	0.88	0.39	0.92	0.78
	w/ MCP	0.79	0.94	0.89	0.79	0.79	0.74	0.65	0.88	0.65	0.91	0.80
Claude-4	w/o MCP	0.86	0.99	0.87	0.87	0.85	0.87	0.70	0.90	0.70	0.92	0.79
	w/ MCP	0.82	0.97	0.92	0.82	0.79	0.78	0.69	0.85	0.60	0.86	0.67
DeepSeek-V3	w/o MCP	0.77	0.97	0.91	0.85	0.85	0.81	0.80	0.84	0.58	0.88	0.74
	w/ MCP	0.78	0.92	0.89	0.80	0.78	0.76	0.65	0.69	0.59	0.86	0.72
Qwen-2.5	w/o MCP	0.76	0.97	0.85	0.73	0.80	0.71	0.70	0.75	0.60	0.82	0.79
	w/ MCP	0.70	0.92	0.79	0.66	0.62	0.72	0.60	0.73	0.40	0.61	0.38
Llama-3.3	w/o MCP	0.72	0.91	0.81	0.73	0.64	0.66	0.48	0.82	0.72	0.79	0.75
	w/ MCP	0.73	0.89	0.80	0.65	0.59	0.60	0.48	0.71	0.55	0.59	0.62
Mistral-3	w/o MCP	0.74	0.93	0.88	0.78	0.81	0.70	0.57	0.87	0.58	0.86	0.64
	w/ MCP	0.81	0.91	0.75	0.73	0.65	0.72	0.46	0.88	0.59	0.81	0.53

Table 1: **Effectiveness (RQ3).** Effectiveness evaluation of the six LLMs with and without MCP integration across three task domains: Knowledge Comprehension (KC), General Reasoning (GR), and Code Generation (CG).

Finding ☞ While most LLMs exhibit a baseline level of instruction-following in one-turn settings, robust MCP tool usage compliance typically emerges through conversational context-building, with deeper interactions further strengthening execution reliability.

As illustrated in Figure 7, a single follow-up turn improves instruction following for nearly all models—by 3.4%, 26.4%, 15.7%, 2.1%, and 2.1% for GPT-4, Claude-4, DeepSeek-V3, Qwen-2.5, and Mistral-3, respectively—while Llama-3.3 sustains perfect IFA of 1.0 in both settings. One-turn compliance is weakest for GPT-4 (IFA of 0.11) and strongest for Qwen-2.5, Mistral-3, and Llama-3.3 (0.87, 0.97, and 1.0). Thus, brief conversational scaffolding transforms explicit MCP directives from largely ignored hints into reliably executed actions.

Effectiveness. Beyond proactivity and compliance, we also evaluate MCP's impact on LLM's effectiveness using well-established LLM benchmarks. We conduct comparative experiments between MCP-augmented and standalone LLM configurations, measuring performance using standard accuracy (Acc) (Zhong et al., 2023) for knowledge comprehension and reasoning tasks, and pass@k (Chen et al., 2021) for code generation tasks. Note that only successful MCP calls are included in the MCP-augmented LLM effectiveness evaluation.

Finding ☞ Contrary to expectations, under default zero-shot integration, MCP integration results in performance degradation across three major task domains, rather than yielding improvements. This indicates that, without MCP-aware adaptation, current LLMs do not yet effectively leverage external information retrieved via MCP tools in an autonomous manner; it characterizes the currently deployed zero-shot setting rather than the inherent limits of MCP-based systems.

As shown in Table 1 (more results are provide in Tables 4 – 9 in the appendix (Section A.2)), instead of enhancing model performance, integration of MCP can degrade LLM effectiveness across the three task domains, averaging 9.5% performance decline across the six LLMs, when MCP tools are employed compared to standalone operation.

Degradation is mild on knowledge comprehension (KC; 1.4% average decline across LLM-dataset pairs), larger on reasoning (10.2% average), and most severe on code generation (17.0% average in pass@k). GPT-4 is the main exception, modestly improving on reasoning (+4.2%) and code (+1.4%), whereas the largest single drop is Qwen-2.5 on code generation (48.1%).

Inspecting the retrieved MCP contexts, we observe recurring low-quality patterns that correlate with these drops, including irrelevant or promotional web content and MCP execution error/diagnostic messages injected into the model input. These observations suggest rather than defini-

Model	Proactivity (TIA↑)		CG (pass@k↑)	
	One-turn	Two-turn	w/o MCP	w/ MCP
Claude-Sonnet-4	0.58	0.87	0.86	0.77
Qwen3	0.31	0.67	0.87	0.86

Table 2: **Newer tool-capable models.** Proactivity (Tool Invocation Accuracy, TIA) under one- and two-turn settings, and effectiveness (pass@k on EvalPlus) with and without MCP, for two recent models with stronger tool-calling alignment.

tively establish that retrieved context may introduce noise or conflicting signals that interfere with models' internal reasoning, instead of providing beneficial context.

Overhead. To quantify the computational cost of MCP integration, we measure the overhead incurred by each LLM. For each model and task domain, we record the total number of input tokens processed with and without MCP calls. This comparison captures the increase in prompt length, and by extension, the additional computational burden introduced by context retrieved via MCP.

> **Finding** ☞ MCP integration imposes a substantial computational overhead—input-token volume grows by $4.25\times$ to $236.5\times$ across the six LLMs and three task domains.

As demonstrated in Table 3, integrating MCP context expands the input-token volume by $4.25\times$ to $236.5\times$ across the six models and three domains, showing that MCP tool integration carries a steep computational cost. The increase spans 4.25–$236.5\times$ on knowledge comprehension, 22.5–$81.4\times$ on general reasoning, and 53.9–$212.7\times$ on code generation.

4.3 Additional Analysis

Robustness on newer tool-capable models. To check that our findings are not artifacts of earlier model generations, we additionally evaluate two recent models with stronger tool-calling alignment, Claude-Sonnet-4 and Qwen3. As shown in Table 2, both the multi-turn proactivity improvement and the effectiveness degradation under zero-shot MCP integration (on EvalPlus) persist, indicating that the observed phenomena are not limited to earlier-generation models.

Effect of tool-set size. A natural hypothesis is that the degradation and overhead stem from exposing too many tools. To test this, we conduct a controlled ablation on EvalPlus with GPT-4, reducing the exposed tools from 9 to 3 via a retrieval-style tool-ranking step (using the `Alibaba-NLP/gte-base-en-v1.5` embedding model). Reducing the tool count changes neither effectiveness (pass@k stays at 0.84, versus 0.85 without MCP) nor overhead (input tokens drop only marginally from 4.30M to 4.23M). This indicates that tool-set size alone is unlikely to explain either the performance degradation or the token overhead: the overhead is driven primarily by injected retrieved content rather than tool count, and the degradation reflects integration-level challenges rather than tool-selection confusion.

Model	Setting	KC	GR	CG
GPT-4	w/o MCP	0.04	0.60	0.06
	w/ MCP	0.17	14.45	8.20
Claude-4	w/o MCP	0.04	0.70	0.07
	w/ MCP	9.46	57.00	14.89
DeepSeek-V3	w/o MCP	0.04	0.53	0.06
	w/ MCP	0.98	20.34	7.05
Qwen-2.5	w/o MCP	0.04	0.58	0.07
	w/ MCP	2.13	13.07	3.77
Llama-3.3	w/o MCP	0.05	0.66	0.08
	w/ MCP	2.69	37.86	7.69
Mistral-3	w/o MCP	0.04	0.62	0.06
	w/ MCP	0.92	13.99	6.65

Table 3: **Overhead (RQ4).** Total input tokens (in millions) processed by the six LLMs with and without MCP tool calls across three task domains, knowledge comprehension (KC), general reasoning (GR), and code generation (CG).

5 Conclusion

We present MCPGauge, an end-to-end framework for systematically evaluating how LLMs interact with the MCP across realistic tool-augmented settings. By combining a 160-prompt diagnostic suite with 25 established benchmarks, MCPGauge enables fine-grained analysis along four critical dimensions: proactivity, compliance, effectiveness, and overhead. Our evaluation of six commercial LLMs reveals consistent limitations in current zero-shot LLM–MCP integration. Proactive tool usage can be enhanced with two-turn conversations, and injected tool context often degrades task accuracy while increasing inference cost by up to $236.5\times$ in input-token overhead. These findings diagnose current zero-shot deployment practice rather than the MCP protocol itself, and establish MCPGauge as a robust benchmark for developing more reliable and cost-efficient tool-augmented LLMs.

6 Limitations

Despite MCPGAUGE's broad coverage, our evaluation has several limitations that merit discussion.

First, all experiments are conducted on off-the-shelf LLMs without any MCP-specific fine-tuning. Our findings therefore reflect how current general-purpose models interact with MCP tools under default deployment settings. It remains an open question whether targeted fine-tuning, reinforcement learning, or tool-use–aware alignment could substantially improve proactivity, instruction compliance, or the effective integration of retrieved context. Second, MCPGAUGE focuses on observable interaction behaviors and end-task outcomes, rather than internal reasoning mechanisms. Although we identify phenomena such as delayed proactivity, instruction-following improvements in multi-turn settings, and performance degradation after context injection, our evaluation does not directly reveal why models fail to exploit MCP effectively. A deeper analysis of internal representations, attention patterns, or reasoning trajectories could provide more mechanistic insights, but is beyond the scope of this work. Third, our evaluation is limited to a fixed set of MCP tools and task domains. While MCPGAUGE covers 30 MCP tool suites and 25 benchmark datasets across knowledge comprehension, general reasoning, and code generation, real-world MCP deployments may involve different tool designs, tool qualities, or domain-specific APIs. Performance and overhead characteristics may therefore vary as the MCP ecosystem evolves.

Despite these limitations, MCPGAUGE establishes a necessary empirical foundation for understanding LLM–MCP interactions. We hope it will motivate future work on MCP-aware training, better tool selection policies, and more efficient integration mechanisms for tool-augmented LLMs.

7 Ethical Consideration

This work systematically evaluates how large language models (LLMs) interact with the Model Context Protocol (MCP), with the aim of understanding the reliability, cost, and failure modes of tool-augmented LLMs rather than enabling misuse. All experiments are conducted using APIs of commercial LLMs under default provider configurations, without circumventing safeguards, rate limits, or usage policies, and in full compliance with their terms of service.

References

Anthropic. 2024. Model Context Protocol (MCP). https://www.anthropic.com/news/model-context-protocol. Published: Nov. 25, 2024; Accessed: 2025-07-06.

Apify. 2025. Model context protocol (mcp) server for apify actors. https://github.com/apify/actors-mcp-server. Online; accessed 10 July 2025.

Apify Console. 2025. Actors MCP Server. https://console.apify.com/. Online; accessed 10 July 2025.

Brave Software, Inc. 2025. Brave Web Search MCP Tool. https://mcpservers.org/servers/brave_web_search. Online; accessed 10 July 2025.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model context protocol (MCP): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*.

Naveen Krishnan. 2025. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. *arXiv preprint arXiv:2504.21030*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572.

Zhiling Luo, Xiaorong Shi, Xuanrui Lin, and Jinyang Gao. 2025. Evaluation report on MCP servers. *arXiv preprint arXiv:2504.11094*.

Model Context Protocol Community. 2025. Model context protocol servers (reference implementations). https://github.com/modelcontextprotocol/servers. Online; accessed 10 July 2025.

Vineeth Sai Narajala and Idan Habler. 2025. Enterprise-grade security for the model context protocol (MCP): Frameworks and mitigation strategies. *arXiv preprint arXiv:2504.08623*.

Brandon Radosevich and John Halloran. 2025. MCP safety audit: Llms with the model context protocol allow major security exploits. *arXiv preprint arXiv:2504.03767*.

Partha Pratim Ray. 2025. A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions. *Authorea Preprints*.

Search1 API. 2025. Search1 API Dashboard. https://console.apify.com/. Online; accessed 10 July 2025.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. A survey of the model context protocol (MCP): Standardizing context to enhance large language models (LLMs).

Hao Song, Yiming Shen, Wenxuan Luo, Leixin Guo, Ting Chen, Jiashui Wang, Beibei Li, Xiaosong Zhang, and Jiachi Chen. 2025. Beyond the protocol: Unveiling attack vectors in the model context protocol ecosystem. *arXiv preprint arXiv:2506.02040*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence?

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

A Appendix

A.1 MCP Server Detail

MCPGAUGE integrates 30 MCP tool suites from three MCP servers: Apify (Apify Console, 2025), Brave-search (Brave Software, Inc., 2025) and Search1api (Search1 API, 2025). We select the web search-based MCP servers from the official Model-Context-Protocol repository (Model Context Protocol Community, 2025). The complete list of tools is provided below:

- **Apify** (Apify offers over 3,000 pre-built cloud tools designed for tasks such as web data extraction. Depending on the task requirements, it dynamically selects and integrates appropriate tools for each AI agent (Apify, 2025). Below, we provide the list of tools utilized for each dataset.):
  - HellaSwag (#6):
    - `get-actor-details`
    - `search-actors`
    - `search-apify-docs`
    - `fetch-apify-docs`
    - `add-actor`
    - `apify-slash-rag-web-browser`
  - AGIEval (#16):
    - `abort-actor-run`
    - `get-actor-details`
    - `get-actor`
    - `get-actor-log`
    - `get-actor-run`
    - `get-dataset`
    - `get-dataset-items`
    - `get-key-value-store`
    - `get-key-value-store-keys`
    - `get-key-value-store-record`
    - `get-actor-run-list`
    - `get-dataset-list`
    - `get-key-value-store-list`
    - `apify-actor-help-tool`
    - `search-actors`
    - `apify-slash-rag-web-browser`
  - EvalPlus (#18):
    - `abort-actor-run`
    - `get-actor-details`
    - `get-actor`
    - `get-actor-log`
    - `get-actor-run`
    - `get-dataset`
    - `get-dataset-items`
    - `get-key-value-store`
    - `get-key-value-store-keys`
    - `get-key-value-store-record`
    - `get-actor-run-list`
    - `get-dataset-list`
    - `get-key-value-store-list`
    - `apify-actor-help-tool`
    - `search-actors`
    - `add-actor`
    - `remove-actor`
    - `apify-slash-rag-web-browser`
- **Brave-Search** (#2):
  - `brave_web_search`
  - `brave_local_search`
- **Search1api** (#6):
  - `search`
  - `news`
  - `crawl`
  - `sitemap`
  - `trending`
  - `search1api_information`

A.2 Detailed Results

We provide six commercial LLMs and their performance with three MCP integration servers for each dataset, respectively, in Tables 4 – 9.

Benchmark	Dataset	GPT-4	GPT-4 with Apify	GPT-4 with Brave-Search	GPT-4 with Search1api
HellaSwag (KC)	ActivityNet	0.80	0.77	0.79	0.81
	WikiHow	0.94	0.95	0.96	0.92
	Average	0.87	0.86	0.88	0.87
AGIEval (GR)	AQUA-rat	0.88	0.87	0.89	0.89
	Math	0.39	0.64	0.64	0.66
	LogiQA-en	0.74	0.61	0.68	0.76
	LogiQA-zh	0.80	0.80	0.78	0.80
	Jec-QA-kd	0.58	0.66	0.64	0.72
	Jec-QA-ca	0.61	0.62	0.61	0.63
	LSAT-ar	0.79	0.67	0.54	0.59
	LSAT-lr	0.89	0.86	0.85	0.87
	LSAT-rc	0.94	0.95	0.90	0.92
	SAT-math	0.99	0.98	0.99	0.99
	SAT-en	0.94	0.93	0.92	0.93
	SAT-en (w/o Psg.)	0.76	0.72	0.76	0.78
	Gaokao-ch	0.73	0.71	0.71	0.66
	Gaokao-en	0.89	0.89	0.90	0.87
	Gaokao-ge	0.83	0.85	0.88	0.86
	Gaokao-hi	0.83	0.87	0.86	0.89
	Gaokao-bio	0.89	0.87	0.87	0.87
	Gaokao-che	0.72	0.76	0.78	0.77
	Gaokao-ph	0.94	0.90	0.89	0.91
	Gaokao-mq	0.87	0.82	0.82	0.81
	Gaokao-mc	0.19	0.24	0.22	0.23
	Average	0.76	0.76	0.76	0.77
EvalPlus (CG)	HumanEval	0.92	0.91	0.92	0.90
	MBPP	0.78	0.84	0.78	0.79
	Average	0.85	0.87	0.85	0.85

Table 4: **Effectiveness (RQ3) for GPT-4.** Effectiveness evaluation of the GPT-4 and GPT-4 with three MCP integration servers: Apify, Brave-Search and Searchapi, across three datasets representing three critical task domains: Knowledge Comprehension (KC), General Reasoning (GR), and Code Generation (CG), respectively.

Benchmark	Dataset	Claude-4	Claude-4 with Apify	Claude-4 with Brave-Search	Claude-4 with Search1api
HellaSwag (KC)	ActivityNet	0.86	0.79	0.80	0.87
	WikiHow	0.99	0.97	0.98	0.96
	Average	0.93	0.88	0.89	0.92
AGIEval (GR)	AQUA-rat	0.90	0.84	0.86	0.86
	Math	0.70	0.50	0.59	0.72
	LogiQA-en	0.87	0.76	0.76	0.77
	LogiQA-zh	0.87	0.82	0.79	0.76
	Jec-QA-kd	0.70	0.76	0.66	0.76
	Jec-QA-ca	0.70	0.68	0.57	0.69
	LSAT-ar	0.70	0.53	0.59	0.61
	LSAT-lr	0.95	0.97	0.94	0.94
	LSAT-rc	0.95	0.92	0.92	0.92
	SAT-math	0.99	0.98	0.98	1.00
	SAT-en	0.98	0.96	0.95	0.97
	SAT-en (w/o Psg.)	0.63	0.82	0.77	0.85
	Gaokao-ch	0.82	0.77	0.77	0.86
	Gaokao-en	0.96	0.96	0.94	0.92
	Gaokao-ge	0.93	0.93	0.79	0.91
	Gaokao-hi	0.95	0.93	0.84	0.91
	Gaokao-bio	0.96	0.93	0.77	0.89
	Gaokao-che	0.95	0.84	0.80	0.85
	Gaokao-ph	0.92	0.89	0.72	0.93
	Gaokao-mq	0.88	0.82	0.77	0.80
	Gaokao-mc	0.26	0.23	0.23	0.20
	Average	0.82	0.79	0.75	0.81
EvalPlus (CG)	HumanEval	0.92	0.90	0.87	0.81
	MBPP	0.79	0.44	0.77	0.79
	Average	0.85	0.67	0.82	0.80

Table 5: **Effectiveness (RQ3) for Claude-4.** Effectiveness evaluation of the Claude-4 and Claude-4 with three MCP integration servers: Apify, Brave-Search and Searchapi, across three datasets representing three critical task domains: Knowledge Comprehension (KC), General Reasoning (GR), and Code Generation (CG), respectively.

Benchmark	Dataset	DeepSeek-V3	DeepSeek-V3 with Apify	DeepSeek-V3 with Brave-Search	DeepSeek-V3 with Search1api
HellaSwag (KC)	ActivityNet	0.77	0.77	0.78	0.78
	WikiHow	0.97	0.92	0.93	0.90
	Average	0.83	0.87	0.85	0.84
AGIEval (GR)	AQUA-rat	0.84	0.32	0.88	0.86
	Math	0.58	0.65	0.61	0.59
	LogiQA-en	0.72	0.71	0.67	0.73
	LogiQA-zh	0.90	0.83	0.83	0.78
	Jec-QA-kd	0.82	0.72	0.65	0.68
	Jec-QA-ca	0.77	0.68	0.59	0.59
	LSAT-ar	0.74	0.69	0.62	0.65
	LSAT-lr	0.87	0.82	0.85	0.85
	LSAT-rc	0.93	0.92	0.92	0. 91
	SAT-math	1.00	0.99	1.00	0.98
	SAT-en	0.95	0.95	0.92	0.94
	SAT-en (w/o Psg.)	0.79	0.67	0.77	0.79
	Gaokao-ch	0.93	0.81	0.76	0.89
	Gaokao-en	0.89	0.91	0.86	0.91
	Gaokao-ge	0.95	0.90	0.83	0.75
	Gaokao-hi	0.96	0.86	0.86	0.78
	Gaokao-bio	0.95	0.91	0.82	0.83
	Gaokao-che	0.93	0.86	0.85	0.83
	Gaokao-ph	0.98	0.95	0.87	0.92
	Gaokao-mq	0.87	0.88	0.85	0.86
	Gaokao-mc	0.23	0.21	0.20	0.17
	Average	0.83	0.76	0.76	0.77
EvalPlus (CG)	HumanEval	0.88	0.76	0.91	0.86
	MBPP	0.74	0.73	0.73	0.70
	Average	0.81	0.75	0.82	0.80

Table 6: **Effectiveness (RQ3) for DeepSeek-V3.** Effectiveness evaluation of the DeepSeek-V3 and DeepSeek-V3 with three MCP integration servers: Apify, Brave-Search and Searchapi, across three datasets representing three critical task domains: Knowledge Comprehension (KC), General Reasoning (GR), and Code Generation (CG), respectively.

Benchmark	Dataset	Qwen-2.5	Qwen-2.5 with Apify	Qwen-2.5 with Brave-Search	Qwen-2.5 with Search1api
HellaSwag (KC)	ActivityNet	0.76	0.62	0.75	0.73
	WikiHow	0.97	0.91	0.93	0.93
	Average	0.87	0.77	0.84	0.83
AGIEval (GR)	AQUA-rat	0.75	0.71	0.70	0.79
	Math	0.60	0.44	0.39	0.37
	LogiQA-en	0.68	0.59	0.60	0.54
	LogiQA-zh	0.73	0.64	0.69	0.68
	Jec-QA-kd	0.76	0.46	0.68	0.62
	Jec-QA-ca	0.63	0.63	0.60	0.62
	LSAT-ar	0.43	0.39	0.33	0.36
	LSAT-lr	0.88	0.67	0.82	0.83
	LSAT-rc	0.87	0.79	0.89	0.90
	SAT-math	0.95	0.89	0.89	0.91
	SAT-en	0.91	0.89	0.92	0.90
	SAT-en (w/o Psg.)	0.70	0.60	0.58	0.57
	Gaokao-ch	0.90	0.77	0.75	0.80
	Gaokao-en	0.92	0.84	0.86	0.88
	Gaokao-ge	0.91	0.69	0.82	0.84
	Gaokao-hi	0.90	0.79	0.86	0.85
	Gaokao-bio	0.93	0.75	0.84	0.84
	Gaokao-che	0.80	0.72	0.82	0.83
	Gaokao-ph	0.87	0.70	0.78	0.77
	Gaokao-mq	0.76	0.71	0.72	0.67
	Gaokao-mc	0.25	0.22	0.16	0.14
	Average	0.76	0.65	0.69	0.69
EvalPlus (CG)	HumanEval	0.82	0.65	0.55	0.64
	MBPP	0.79	0.56	0.17	0.43
	Average	0.81	0.60	0.36	0.54

Table 7: **Effectiveness (RQ3) for Qwen-2.5.** Effectiveness evaluation of the Qwen-2.5 and Qwen-2.5 with three MCP integration servers: Apify, Brave-Search and Searchapi, across three datasets representing three critical task domains: Knowledge Comprehension (KC), General Reasoning (GR), and Code Generation (CG), respectively.

Benchmark	Dataset	Llama-3.3	Llama-3.3 with Apify	Llama-3.3 with Brave-Search	Llama-3.3 with Search1api
HellaSwag (KC)	ActivityNet	0.72	0.68	0.75	0.76
	WikiHow	0.91	0.84	0.93	0.89
	Average	0.82	0.76	0.84	0.83
AGIEval (GR)	AQUA-rat	0.82	0.74	0.68	0.72
	Math	0.72	0.60	0.48	0.57
	LogiQA-en	0.63	0.51	0.62	0.63
	LogiQA-zh	0.69	0.56	0.62	0.65
	Jec-QA-kd	0.51	0.53	0.47	0.52
	Jec-QA-ca	0.45	0.41	0.48	0.46
	LSAT-ar	0.43	0.38	0.35	0.33
	LSAT-lr	0.84	0.72	0.83	0.86
	LSAT-rc	0.93	0.85	0.61	0.92
	SAT-math	0.89	0.89	0.84	0.87
	SAT-en	0.89	0.88	0.83	0.91
	SAT-en (w/o Psg.)	0.65	0.65	0.72	0.60
	Gaokao-ch	0.65	0.54	0.56	0.61
	Gaokao-en	0.82	0.78	0.82	0.83
	Gaokao-ge	0.83	0.74	0.80	0.83
	Gaokao-hi	0.82	0.68	0.71	0.74
	Gaokao-bio	0.81	0.69	0.76	0.79
	Gaokao-che	0.66	0.55	0.60	0.58
	Gaokao-ph	0.59	0.52	0.53	0.58
	Gaokao-mq	0.45	0.38	0.41	0.49
	Gaokao-mc	0.12	0.20	0.20	0.12
	Average	0.67	0.60	0.61	0.64
EvalPlus (CG)	HumanEval	0.79	0.58	0.64	0.56
	MBPP	0.75	0.61	0.68	0.56
	Average	0.77	0.59	0.66	0.56

Table 8: **Effectiveness (RQ3) for Llama-3.3.** Effectiveness evaluation of the Llama-3.3 and Llama-3.3 with three MCP integration servers: Apify, Brave-Search and Searchapi, across three datasets representing three critical task domains: Knowledge Comprehension (KC), General Reasoning (GR), and Code Generation (CG), respectively.

Benchmark	Dataset	Mistral-3	Mistral-3 with Apify	Mistral-3 with Brave-Search	Mistral-3 with Search1api
HellaSwag (KC)	ActivityNet	0.74	0.77	0.76	0.91
	WikiHow	0.93	0.90	0.93	0.91
	Average	0.84	0.84	0.85	0.91
AGIEval (GR)	AQUA-rat	0.87	0.88	0.90	0.87
	Math	0.58	0.54	0.58	0.64
	LogiQA-en	0.66	0.73	0.72	0.66
	LogiQA-zh	0.74	0.73	0.77	0.73
	Jec-QA-kd	0.59	0.47	0.48	0.53
	Jec-QA-ca	0.55	0.48	0.26	0.53
	LSAT-ar	0.54	0.44	0.49	0.49
	LSAT-lr	0.88	0.79	0.79	0.81
	LSAT-rc	0.92	0.92	0.92	0.89
	SAT-math	0.99	0.97	1.00	0.99
	SAT-en	0.93	0.91	0.91	0.90
	SAT-en (w/o Psg.)	0.72	0.58	0.24	0.27
	Gaokao-ch	0.82	0.71	0.72	0.77
	Gaokao-en	0.88	0.88	0.92	0.89
	Gaokao-ge	0.88	0.80	0.82	0.86
	Gaokao-hi	0.91	0.82	0.37	0.87
	Gaokao-bio	0.89	0.84	0.93	0.88
	Gaokao-che	0.88	0.68	0.76	0.80
	Gaokao-ph	0.90	0.81	0.88	0.86
	Gaokao-mq	0.83	0.83	0.80	0.85
	Gaokao-mc	0.29	0.25	0.29	0.27
	Average	0.76	0.71	0.68	0.72
EvalPlus (CG)	HumanEval	0.86	0.76	0.84	0.84
	MBPP	0.64	0.43	0.57	0.60
	Average	0.75	0.59	0.71	0.72

Table 9: **Effectiveness (RQ3) for Mistral-3.** Effectiveness evaluation of the Mistral-3 and Mistral-3 with three MCP integration servers: Apify, Brave-Search and Searchapi, across three datasets representing three critical task domains: Knowledge Comprehension (KC), General Reasoning (GR), and Code Generation (CG), respectively.