Evaluation Report on MCP Servers

2025-04-13

Executive Summary

1. There are significant differences in effectiveness and efficiency among MCP servers; using MCPs does not demonstrate a noticeable improvement compared to the usage of non-MCP tools.
2. The effectiveness of the MCP server can be enhanced by optimizing the parameters that need to be constructed by the LLM.

Introduction

To study the effectiveness and efficiency of MCP servers, we selected several widely used MCP servers and conducted an experiment to evaluate them using MCPBench based on their accuracy, latency, and token usage. We focused on two tasks: web search and database search. The former involves searching the internet to answer questions, while the latter entails fetching data from a database. All MCP servers were compared using the same LLM and prompt in a controlled environment. We aimed to answer the following questions:

- Question 1: Are MCP servers effective and efficient in practice?
- Question 2: Does using MCP provide higher accuracy compared to function calls?
- Question 3: Can we optimize performance?

Tasks and Dataset

The evaluation report involves two tasks:

1) Web Search

This task takes a question as input. The LLM rewrites it into keywords or short sentences, involving a tool that typically searches the internet and returns results to the LLM. Here is an example involving the brave_web_search MCP:

Input	**Tool**	**Results**
What is the middle name of Barack Obama	brave_web_search {"query":"middle name of Barack Obama"}	Hussein

We utilized a hybrid dataset from the following sources:

Datasource	**Details**	**Volume**	**Case**

Frames	Open-source dataset	100	"Prompt": "As of August 1, 2024, which country hosted the FIFA World Cup the last time the UEFA Champions League was won by a club from London?", "Answer": "France"
News	Collected by cleaning data from daily Xinwen Lianbo transcripts over the past three months and processing it using reverse engineering techniques.	260	"Prompt": "In which city did Tesla's first energy storage super factory outside of the United States officially start production on February 11, 2025?", "Answer": "Shanghai"
Knowledge	Collected by cleaning data from knowledge-intensive websites like Wikipedia and science and technology reports, and processing it using reverse engineering techniques.	200	"Prompt": "What type of fish might the family Cyprinidae and the family Dace represent as their primitive types?", "Answer": "armored fish"

2) Database Search

This task similarly takes a question as input. The LLM retrieves data from the database through a database MCP server. Here's an example involving the MySQL MCP:

Input	**Tool**	**Results**
Fetch the sales of Tesla Model S since 2025-01	execute_sql {"query":"Select sum(sales) from sales where series='Tesla Model S' and datetime¿ 2025-01"}	13402

We gathered datasets from the following sources:

Datasource	**Details**	**Volume**	**Case**
Car_bi	A synthetic dataset from an automobile manufacturer datasource	74	"Prompt": "What is the total number of orders in the South China region in February 2025?", "Answer": "0"
SQL_EVAL	Sampled from SQL_EVAL[1]. It is based off the schema from the Spider, but with a new set of hand-selected questions and queries grouped by query category.	256	"Prompt": Which authors have written publications in both the domain "Machine Learning" and the domain "Data Science"? "Answer": "Ashish Vaswani"

Overview of MCP Servers

We collected the MCP servers from GitHub and Smithary.AI. Due to limitations of time and cost, we selected those with high call records.

[1]https://github.com/defog-ai/sql-eval

Web Search Related MCP

- **Brave Search:** A web and local search utilizing Brave's Search API.
  Source: `https://github.com/modelcontextprotocol/servers/tree/main/src/brave-search`
  Tool Name: brave_web_search
  Developer: erdnax123

- **DuckDuckGo Search Server:** A Model Context Protocol (MCP) server providing web search capabilities through DuckDuckGo, with additional features for content fetching and parsing.
  Source: `https://github.com/nickclyde/duckduckgo-mcp-server`
  Tool Name: search
  Developer: nickclyde

- **Tavily MCP Server:** A search engine for AI agents (search + extract) powered by Tavily.
  Source: `https://github.com/tavily-ai/tavily-mcp`
  Tool Name: tavily-search
  Developer: tavily-ai

- **Exa Search:** A search engine designed for AI by Exa.
  Source: `https://github.com/exa-labs/exa-mcp-server`
  Tool Name: web_search
  Developer: exa-labs

- **Fire Crawl Search:** Extracts web data using Firecrawl.
  Source: `https://github.com/mendableai/firecrawl-mcp-server`
  Tool Name: firecrawl_search
  Developer: mendableai

- **Bing Web Search:** A Model Context Protocol (MCP) server for Microsoft Bing Search API integration, enabling AI assistants to conduct web, news, and image searches.
  Source: `https://github.com/leehanchung/bing-search-mcp`
  Tool Name: bing_web_search
  Developer: leehanchung

- **BochaAI:** A search engine for AI that provides access to high-quality global knowledge from nearly ten billion web pages and ecological content sources across various fields, including weather, news, encyclopedias, healthcare, and travel.
  Source: Alibaba Cloud BaiLian Platform
  Tool Name: bocha_web_search
  Developer: Alibaba Cloud

Non-MCP for Web Search

For comparison, we included some non-MCP servers:

- **Qwen Web Search:** Uses the API provided by Qwen-Max-0125 to enable online search with `extra_body="enable_search": True`.

- **Quark Search:** A search engine that is particularly useful for searching unknown information such as weather, exchange rates, and current events.
  Source: Official Quark Search Plugin provided by Alibaba Cloud BaiLian Platform
  Tool Name: quark_search
  Developer: Alibaba Cloud

Database Search Related MCP

- **XiYan MCP Server:** An MCP server that supports data retrieval from a database using natural language queries, powered by XiyanSQL as the text-to-SQL LLM.
  Source: `https://github.com/XGenerationLab/xiyan_mcp_server`
  Tool Name: get_data
  Developer: XGenerationLab

- **MySQL MCP Server:** An implementation that facilitates secure interaction with MySQL databases.
  Source: `https://github.com/designcomputer/mysql_mcp_server`
  Tool Name: execute_sql
  Developer: designcomputer

Criteria for Evaluation

Accuracy

Accuracy is evaluated to determine the correctness of the answer. It is assessed by an LLM-based grader. We use DeepSeek-v3 as the grader. The prompt is:

> For the following question: {question}
> Please judge whether the predicted answer is correct. It is considered correct if it addresses the key information:
> Predicted Answer: {prediction}
> Correct Answer: {ground_truth}
> Just return True or False.

The accuracy of each sample is 1 if the grader replies True, and 0 otherwise. The overall accuracy is the average of all samples. Due to the instability of the network environment, we consider another criterion: accuracy of valid samples. Valid samples are those returned within a limited time, while invalid samples are those that timed out.

Time Consumption

We collect the end-to-end time consumption, which includes the latency of both the LLM and the MCP server.

Token Consumption

We record the pre-fill and completion tokens used during the experiment.

Other Setups

The experiment is executed on a server in Singapore with a dual-core CPU and 2GB RAM. The evaluation framework used is MCPBench. All MCP servers (except DuckDuckGo) are launched on the server in SSE mode. The timeout is set to 30 seconds. The system prompt is as follows: We collect the query from `<WebSearch></WebSearch>` and send it to the MCP, which handles questions that may require web searching.

- Input contains:
  - The question that needs to be answered.
  - Past search steps and their results.
- Output can be either:
  - If more search is needed: output in the format `<WebSearch>search_query</WebSearch>`
  - If the question can be answered: direct answer.

Example:

> Input question: "Who is the current President of the United States?"
> - If no search has been conducted: `<WebSearch>current President of United States</WebSearch>`
> - If sufficient information is available: "Joe Biden is the current President of the United States."

Comparative Analysis

Question 1: Are MCP Servers Effective and Efficient in Practice?

MCP Server	Accuracy (%) ↑	Time Consumption (s) ↓	Pre-fill Token Consumption ↓	Completion Token Consumption ↓
Brave Search	46.6	13.98	5802.35	236.26
DuckDuckGo Search Server	13.62	64.17	1718.84	162.25
Tavily MCP Server	47.99	95.52	2441.73	196.03
Exa Search	15.02	231.24	2475.24	190.49
BoChaAI Search	20	35.54	1642.71	189.13
Fire Crawl Search	58.33	15.44	1727.17	179.61
Bing Web Search	64.33	12.4	4060.34	206.87

Table 1 shows the experiment results of the MCP servers. Here are the observations:

1. The differences in effectiveness are significant. Based on the accuracy of valid samples, the highest accuracy is observed with Bing Web Search (64%), while DuckDuckGo has the lowest at just 10%, representing a difference of 54 percentage points.

2. The differences in efficiency are even more pronounced; regarding the average time consumed for valid samples, the fastest are Bing Web Search and Brave Search, which require less than 15 seconds, while the slowest, Exa Search, takes 231 seconds (note that valid samples are cases of normal returns without timeouts, so this value is unaffected by timeouts).

3. Token consumption is similar; based on the number of output tokens for valid samples, consumption generally falls between 150 and 250 tokens, indicating that the model consistently provides concise answers without unnecessary elaboration on its MCP usage.

Question 2: Does MCP Provide Higher Accuracy Compared to Function Calls?

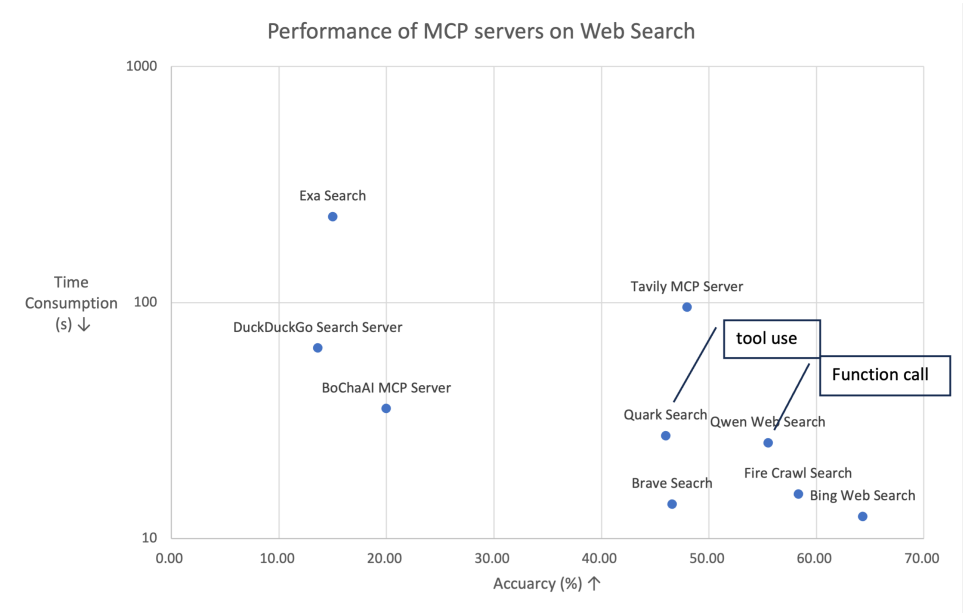

Figure 1: The performance of MCP servers in Web search

We compared the performance of non-MCP servers with the above MCP servers. The results are shown in Figure 1. We observe that both function calls (Qwen Web Search) and tool usage (Quark

Search) exhibit competitive accuracy and time consumption. The accuracy of Qwen Web Search is 55.52%, surpassing that of Exa Search, DuckDuckGo, Tavily, and Brave Search.

Question 3: Can We Optimize Performance?

MCP Server	Accuracy (%) ↑	Time Consumption (s) ↓	Pre-fill Token Consumption ↓	Completion Token Consumption ↓
MySQL MCP Server	77.78	3.75	2782.76	58.88
MySQL MCP Server + Text-2-SQL model	83.33	5.29	386.68	44.49

MCP Server	Accuracy (%) ↑	Time Consumption (s) ↓	Pre-fill Token Consumption ↓	Completion Token Consumption ↓
PostgreSQL MCP Server	58.5	5.85	6896.51	103.22
PostgreSQL MCP Server + Text-2-SQL model	80.08	12.87	434.86	97.57

To address this question, we used the database search task. The MySQL MCP server implements a straightforward encapsulation of the database connection. After configuring the database account, password, and other information, these MCPs establish a persistent connection to the database and expose the execute_sql tool interface. When calling this tool, the model must construct a query parameter that must be an executable SQL statement. Although this simple encapsulation functions as an MCP, it assigns the most challenging part of the process—constructing the SQL query statement—to the LLM. Consequently, the success of the entire tool call heavily relies on the LLM's ability to construct SQL statements.

The XiYan MCP server is an optimized version that allows for parameter input using natural language instead of SQL. It utilizes XiyanSQL-qwencoder-32B to convert natural language queries into SQL before executing comparable to the MySQL MCP server. The experiment results are shown in Table 2. By adding a text-to-SQL model to the MCP server, it improves accuracy by 5 percentage points. In the PostgreSQL experiment, the optimization results in a 22-point increase.

In summary, tool developers can optimize their MCPs by analyzing the challenges faced by LLMs during parameter construction, thereby enhancing the accuracy of MCP usage.

Conclusion

The evaluation of various Model Context Protocol (MCP) servers highlights significant differences in both effectiveness and efficiency. While MCPs offer distinct advantages in structuring tool usage, they do not consistently demonstrate marked improvements over non-MCP approaches, such as function calls. Our experiments showed that the most effective MCP, Bing Web Search, achieved an accuracy of 64%, whereas DuckDuckGo lagged at just 10%. Furthermore, performance varied widely in terms of time consumption, with top performers like Bing and Brave Search executing tasks in under 15 seconds, contrasted with significantly slower alternatives like Exa Search.

Importantly, we found that the accuracy of MCP servers can be substantially enhanced by optimizing the parameters that LLMs must construct. For instance, transitioning from SQL-based queries to natural language processing in the XiYan MCP server resulted in a noteworthy increase in accuracy, demonstrating that incorporating a text-to-SQL model can lead to a 5 percentage point improvement.

Overall, while MCPs provide a structured means for AI tools to interact with data, there remains considerable potential for optimization. By addressing the challenges LLMs encounter in parameter construction and enhancing the user-friendliness of tool interfaces, developers can significantly improve the performance and reliability of MCP servers. This research paves the way for further investigations into optimized MCP implementations, ultimately leading to better AI-driven search and data retrieval solutions.