Using MCP Servers with other AI models or web search

Overview

In this lesson, we took a deep dive into how **MCP (Model Context Protocol) servers** can integrate multiple AI models to create a powerful application ecosystem. This allows us to leverage the unique strengths of different models, such as web search or complex computations, beyond the capabilities of a single large language model (LLM) like **Claude**.

Key Concepts

- **MCP Servers**: These servers enable our primary LLM, such as Claude, to connect and communicate with other specialized LLMs and services.
- **Integration of Models**: We focused on supplementing Claude with Perplexity's models, specifically **Sonar** and **Sonar Pro**, for enhanced web search functionalities.

Step-by-Step Process

Setting Up Perplexity

- **API Key**: First, we obtained an API key from Perplexity. This is crucial for enabling Claude to use Perplexity's web search via MCP.
- **Creating the Web Search Server**:
  - We designed a new server, `web_search.py`.
  - This server includes functions for web search capabilities using Perplexity's API.

Coding the Server Logic

- **Import Fast MCP**: We started by bringing in necessary modules with `from MCP server fast MP import fast MCP`.
- **Server Functionality**:
  - Created a function to handle search queries.
  - Set up a message system to pass queries to Perplexity and retrieve responses.
- **Standardization**: Emphasized the importance of using standard protocols similar to OpenAI, adjusting only the base URL.

Testing

- **Running the Server**: Initiated the server and verified connections with MCC commands to ensure everything functions seamlessly.

- **Web Search Example**: We successfully performed a search querying the Toronto Maple Leafs' status in the 2025 NHL playoffs.

Handling Errors

- **Timeout Issues**: Discussed potential timeouts when queries take longer than 30 seconds, and how to mitigate them by extending the timeout or using faster models.

Final Integration

- **Connecting to Claude**:
  - Updated `Claude_desktop_config.json` to include the new web search server.
  - Added new commands for the MCP server directory and specified the execution of `web_search.py`.

Conclusion

By integrating multiple AI models via MCP servers, we created a flexible and robust framework that significantly extends the capabilities of a single LLM. This approach enables Claude to perform tasks like web searches using Perplexity’s technologies, something it couldn't do natively.

We wrapped up by successfully setting up and verifying this integration, paving the way for creating versatile, multi-functional AI-driven applications.