# CAS in NLP

- Module 3 – Day 1 – MCP / Integrated LLMs

- Ahmad Alhineidi

## LLMs

- Good at generating text / language understanding
- Not great at math / calculation
- Relies on internal knowledge (Parametric)
- Not great at storing large data

# LLMs in a pipeline?

- Retrieval Augmented Generation (2020 paper link)

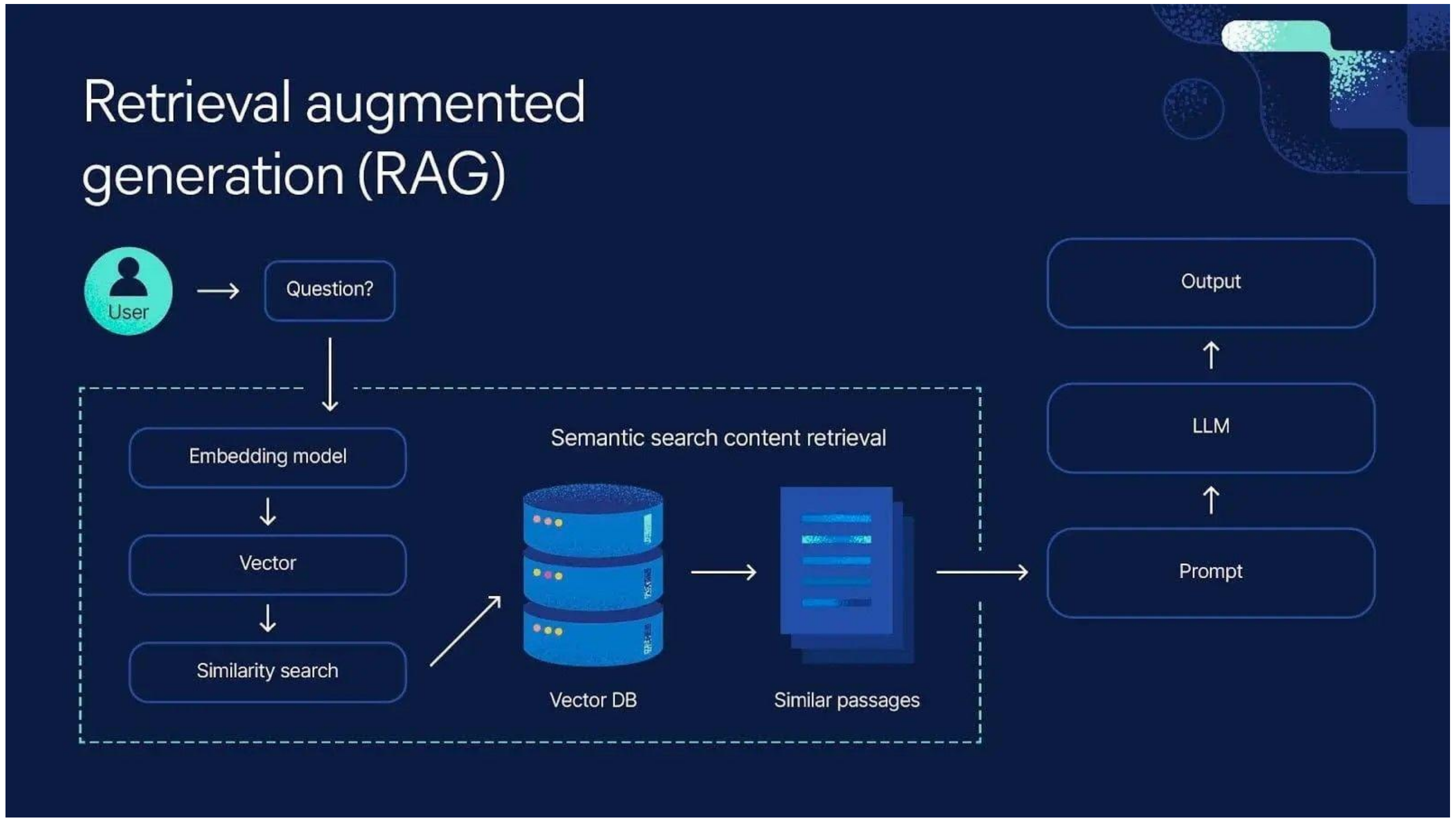


Image source: link

# LLMs in a pipeline?

- Vanilla RAG has limitation
- Introduce more functionalities in the system
- LLM decides which tool to use (data, functions, internet search)

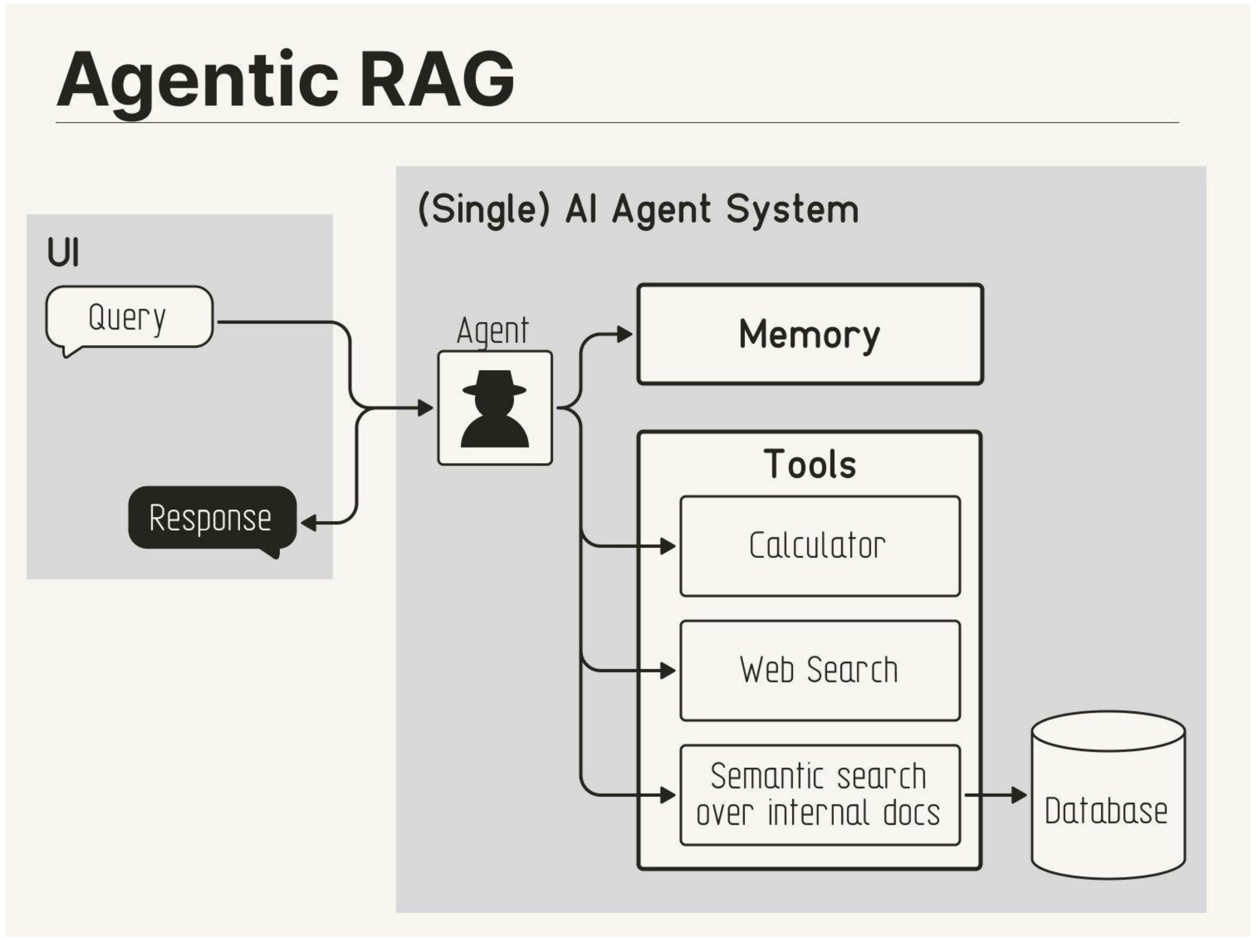


Image source: link

# Function calling

- Certain models can use external functions
- Example: tools in ollama
- Enhance LLMs in dealing with limitations

Ollama: Tool support

Image source: link

# Model Context Protocol (MCP)

- MCP open source
- Protocol == standard way of connecting LLMs into external applications
- Databases, file storages, functions, search engine, etc
- Client, server architecture

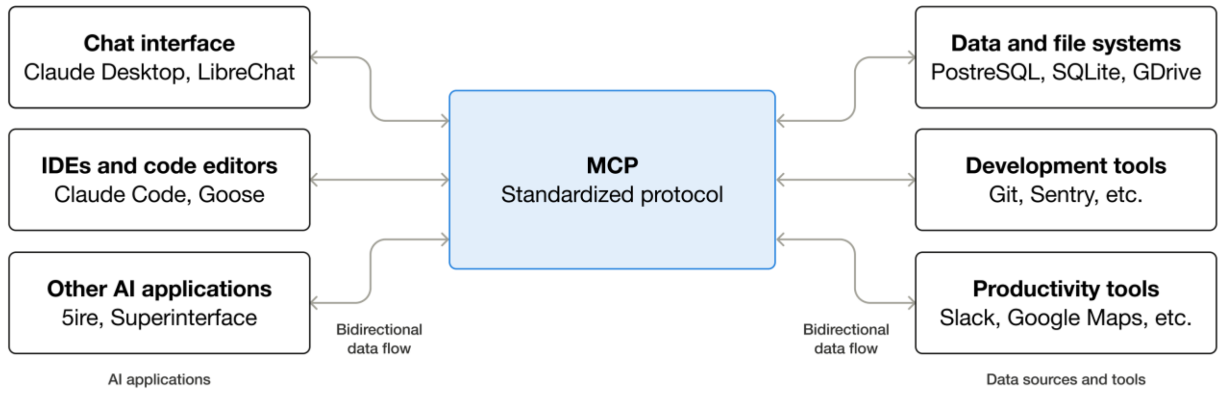


Image source: link

## MCP architecture and elements

Goal: help LLM & outside systems work together easily

Elements:
1- MCP host
2- MCP Client
3- MCP Server
4- Transport Layer

## MCP elements - Host

- Main AI application
- User interaction point
- Goal: Takes user input, deliver to LLMs
- Examples: VS code, Cursor, AI custom application, conversation application

# MCP elements - Client

- The translator & messenger of the LLM
- Lives also in the host
- Takes LLM request --> converts into standard MCP format (e.g. JSON)
- Example: “Model wants to use weather api function” → client translate *{"method": “get_weather", "params": {“city": “Bern"}}*

## MCP elements - Server

- External tools, promots, data, services
- It Receives standard input from the Client, send a standard respond
- From previous example:
  *{"method": “get_weather", “output": {“temp": “27“, “status”: “cloudy”}}*

## MCP elements – Transport layer

- *Language to communicate between the server & the Client*
- Standard is JSON-RPC 2.0 (**J**ava **S**cript **O**bject **N**otation – **R**emote **P**rocedure **C**all)
- Two transport methos:
  - Standard input/output (stdio)
  - Server-sent events (SSE)

## MCP elements – Transport layer – JSON – PRC 2.0

- JSON: human-machine readable structured/unstructured data with key – value format
- PRC: a procedure for one program to ask another program to run a function / tool / etc and get results back
- 2.0: the version of the protocol to use in MCP
- Example:

```
--> {"jsonrpc": "2.0", "method": "subtract", "params": [42, 23], "id": 1}
<-- {"jsonrpc": "2.0", "result": 19, "id": 1}

--> {"jsonrpc": "2.0", "method": "subtract", "params": [23, 42], "id": 2}
<-- {"jsonrpc": "2.0", "result": -19, "id": 2}
```

Moreon JSON – PRC 2.0: link

## MCP elements – Transport layer – stdio

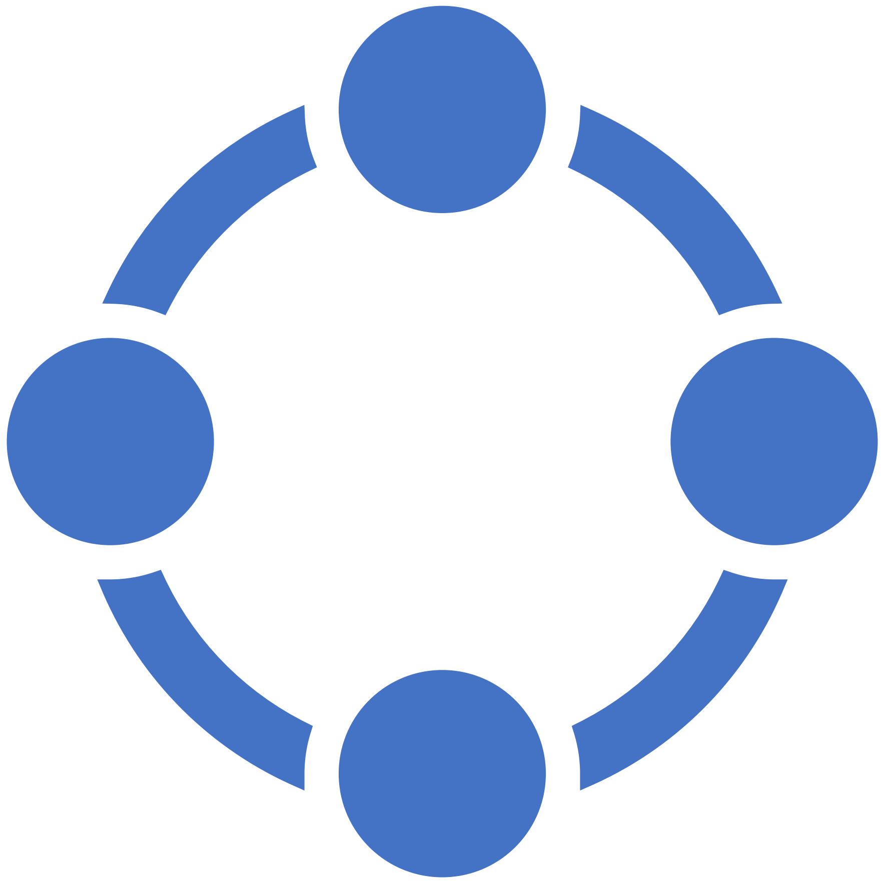
- Most basic way programs running on the same machine communicate
- Standard input – output  - error
- In most scenario in MCP where the host and the server running on the same machine
- Fast, synchronous connection

# MCP elements – Transport layer – SSE

- Communication between client and server over a network (MCP server - clients)
- Standard input – output  - error
- It allows efficient real time streaming of data
- Usually with HTTP request, server stays open for more requests rather than sending respond, active whenever there is new event

# Why use MCP?

- Development benefits: standard
- Development benefits: Creates larger eco-system
- Scalability: easier to add more tools/prompts/sources
- Reusability: components are independent
- User benefits: Increase LLMs useability



## Tutorial

- Ollama function calling
- Simple RAG
- Simple MCP

## Worksop

- Split into groups of 2
- Add more “useful” tools to the Ollama tools then into the MCP
- Optional: Add database (Example: All the data about the CAS NLP)
- Present findings on Wednesday evening (What you did?, Screenshot demo of the model performance with tools/DB) Here is a link to where to add your slides: link