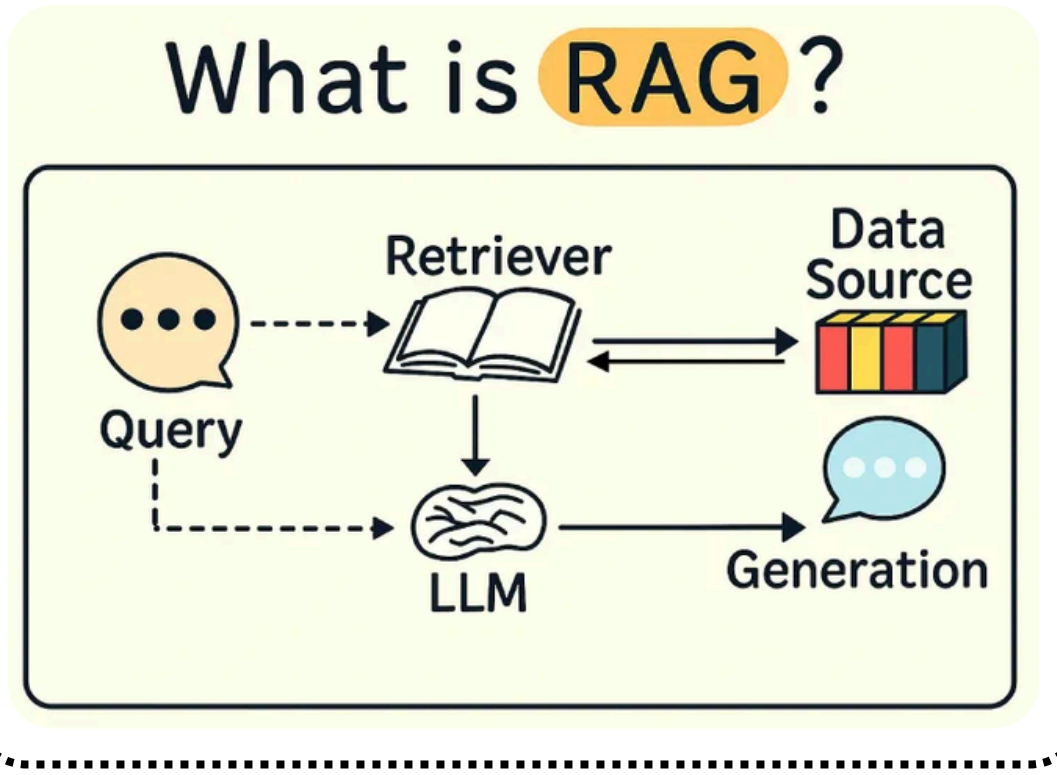
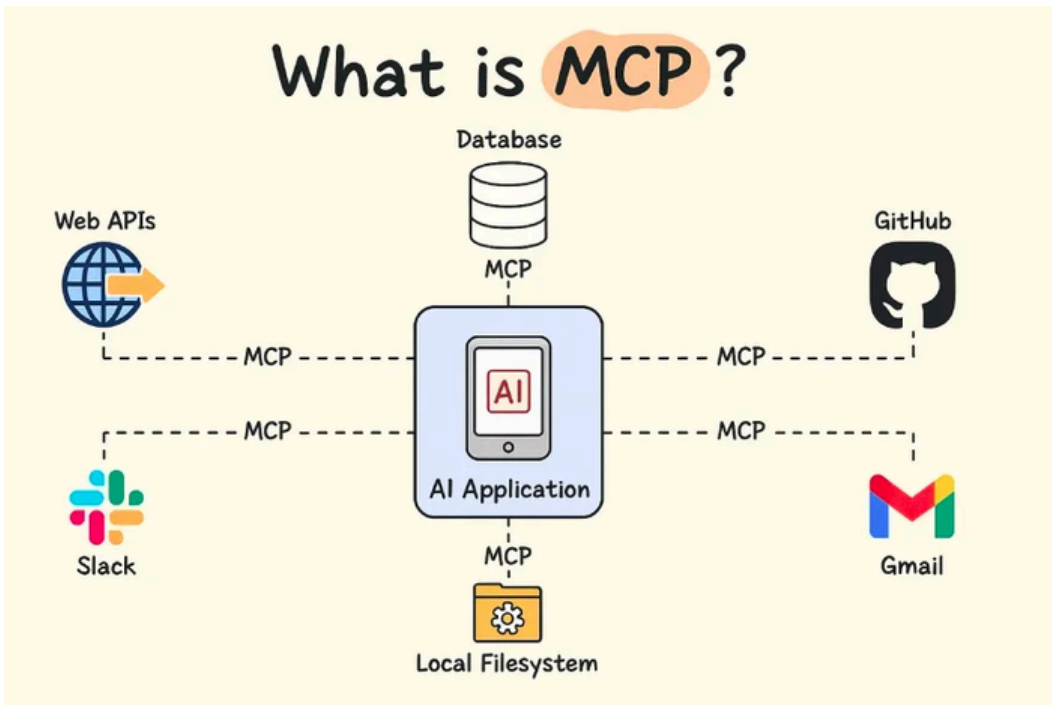
Analytics Vidhya

How to Perform RAG using MCP?





RAG is an AI framework that combines the strengths of traditional information retrieval systems (such as search and database) with the capabilities of AI models that are outstanding at natural language generation. Its benefits include real-time and factual responses, reduced hallucinations, and context-aware answers. RAG is like asking a librarian about the information before writing a detailed report.

MCP acts as a bridge between your AI assistant and external tools. It's an open protocol that lets LLMs access real-world tools, APIs, or datasets accurately and efficiently. Traditional APIs and tools require custom code for integrating them with AI models, but MCP provides a generic way to connect tools to LLMs in the simplest way possible. It provides plug-and-play tools.

Steps for Performing RAG with MCP

Now, we are going to implement RAG with MCP in a detailed manner. Follow these steps to create your first MCP server performing RAG. Let’s dive into implementation now:
Firstly, we will set up our RAG MCP server.

Step 1: Installing the dependencies

```
pip install langchain>=0.1.0 \
            langchain-community>=0.0.5 \
            langchain-groq>=0.0.2 \
            mcp>=1.9.1 \
            chromadb>=0.4.22 \
            huggingface-hub>=0.20.3 \
            transformers>=4.38.0 \
            sentence-transformers>=2.2.2
```

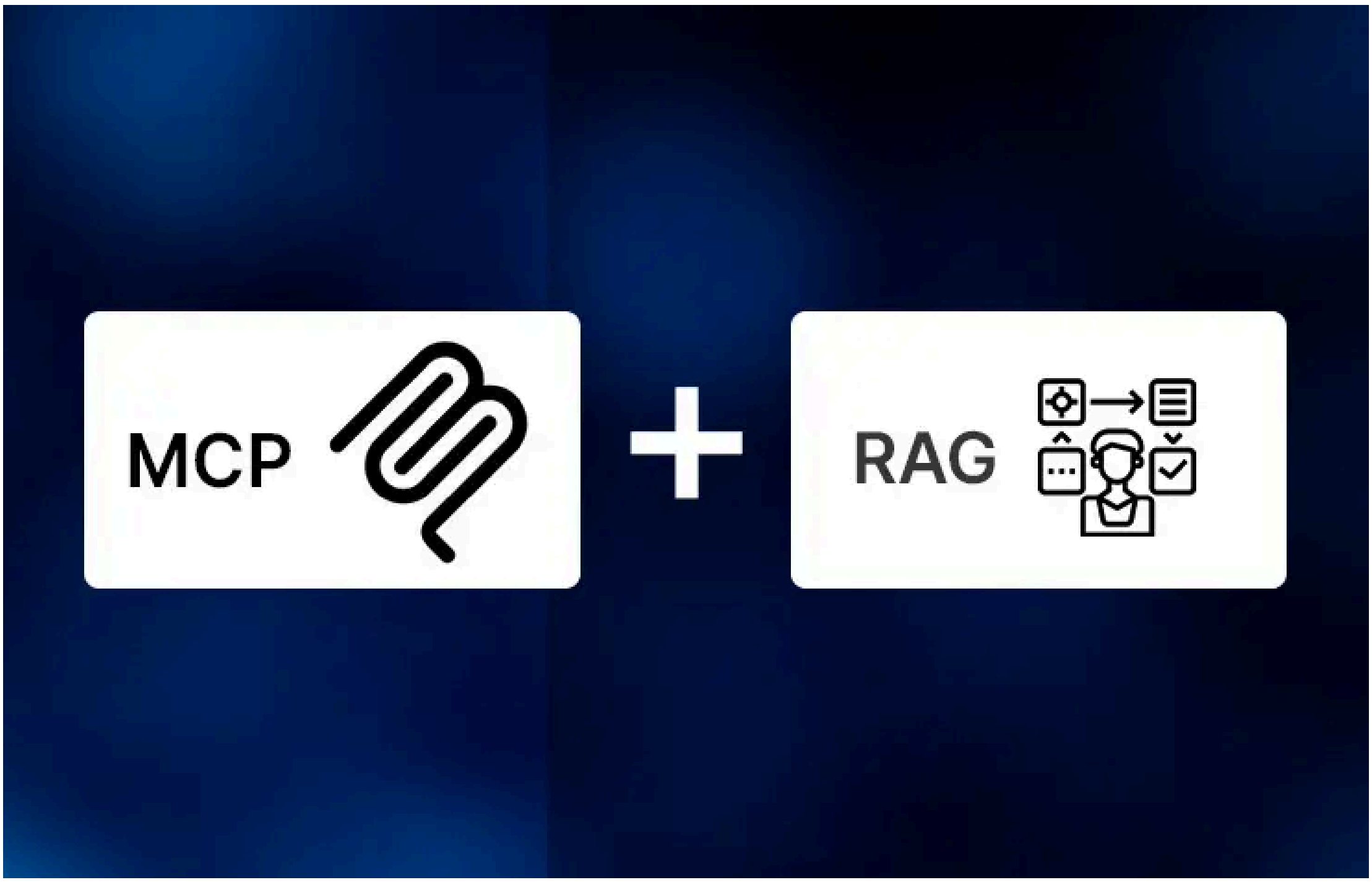
This step will install all the required libraries in your system.

Step 2: Creating server.py

Now, we are defining the RAG MCP server in the server.py file. Following is the code for it. It contains a simple RAG code with an MCP connection to it.

```python
from mcp.server.fastmcp import FastMCP
from langchain.chains import RetrievalQA
from langchain.document_loaders import TextLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain_community.vectorstores import Chroma
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_groq import ChatGroq  # Groq LLM


# Create an MCP server
mcp = FastMCP("RAG")


# Set up embeddings (You can pick a different Hugging Face model if preferred)
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")

# Set up Groq LLM
model = ChatGroq(
   model_name="llama3-8b-8192",  # or another Groq-supported model
   groq_api_key="YOUR_GROQ_API"  # Required if not set via environment variable
)


# Load documents
loader = TextLoader("dummy.txt")
data = loader.load()


# Document splitting
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
texts = text_splitter.split_documents(data)


# Vector DB
docsearch = Chroma.from_documents(texts, embeddings)


# Retriever chain
qa = RetrievalQA.from_chain_type(llm=model, retriever=docsearch.as_retriever())


@mcp.tool()
def retrieve(prompt: str) -> str:
   """Get information using RAG"""
   return qa.invoke(prompt)


if __name__ == "__main__":
   mcp.run()
```

For more information, kindly visit this article



AI Agents Intermediate RAG

How to Perform RAG using MCP?

Explore how to integrate RAG with MCP to enhance your AI a