In [4]:

```
# 📦 Install if not already done
# !pip install langchain openai faiss-cpu tiktoken python-dotenv

# 1️⃣ Imports
from langchain.agents import Tool, AgentType, initialize_agent
from langchain.chat_models import ChatOpenAI
from langchain.vectorstores import FAISS
from langchain.embeddings import OpenAIEmbeddings
from langchain.chains import RetrievalQA
from langchain.memory import ConversationBufferMemory

import os
from dotenv import load_dotenv

# 2️⃣ Load API keys
load_dotenv(".env")
os.environ["OPENAI_API_KEY"] = os.getenv("OPENAI_API_KEY")

# 3️⃣ Setup LLM
llm = ChatOpenAI(temperature=0)

# 4️⃣ Create Vector DB (Retriever)

with open("sample.txt", "r", encoding="utf-8") as f:
    text_data = f.read()

# 🧠 Split the text into smaller chunks
from langchain.text_splitter import CharacterTextSplitter
splitter = CharacterTextSplitter(separator="\n", chunk_size=300, chunk_overl
texts = splitter.split_text(text_data)
embedding = OpenAIEmbeddings()
vectorstore = FAISS.from_texts(texts, embedding)
retriever = vectorstore.as_retriever()

# 5️⃣ Create RetrievalQA Chain
qa_chain = RetrievalQA.from_chain_type(llm=llm, retriever=retriever)

# 6️⃣ Convert to LangChain Tool
qa_tool = Tool(
    name="LangChainRetriever",
    func=qa_chain.run,
    description="Use this to answer questions about LangChain framework, fea
)

# 7️⃣ Setup Memory (to support conversational context)
memory = ConversationBufferMemory(memory_key="chat_history", return_messages

# 8️⃣ Initialize Agent with Tool + Memory
agent = initialize_agent(
    tools=[qa_tool],
    llm=llm,
    agent=AgentType.CONVERSATIONAL_REACT_DESCRIPTION,  # ReAct agent
    memory=memory,
    verbose=True,
```

```
    handle_parsing_errors=True
)

# 9️⃣ Ask Questions (RAG-Style)
print("1️⃣ First Question")
res1 = agent.run("What is LangChain?")
print("Answer:", res1)

print("\n2️⃣ Follow-up")
res2 = agent.run("Who created it?")
print("Answer:", res2)

print("\n3️⃣ Combined Reasoning")
res3 = agent.run("Explain LangChain's use in AI workflows.")
print("Answer:", res3)
```

1️⃣ First Question

```
> Entering new AgentExecutor chain...
Thought: Do I need to use a tool? Yes
Action: LangChainRetriever
Action Input: What is LangChain?
Observation: LangChain is a framework created by Harrison Chase for building
applications with Large Language Models (LLMs). It supports various features
such as RAG, agents, memory, tools, and more. LangChain is commonly used in a
pplications like chatbots, document Q&A, and AI workflow.
Do I need to use a tool? No
AI: LangChain is a framework created by Harrison Chase for building applicati
ons with Large Language Models (LLMs). It supports various features such as R
AG, agents, memory, tools, and more. LangChain is commonly used in applicatio
ns like chatbots, document Q&A, and AI workflow.

> Finished chain.
Answer: LangChain is a framework created by Harrison Chase for building appli
cations with Large Language Models (LLMs). It supports various features such
as RAG, agents, memory, tools, and more. LangChain is commonly used in applic
ations like chatbots, document Q&A, and AI workflow.
```

2️⃣ Follow-up

```
> Entering new AgentExecutor chain...
Thought: Do I need to use a tool? Yes
Action: LangChainRetriever
Action Input: creator
Observation: LangChain was created by Harrison Chase.
Do I need to use a tool? No
AI: LangChain was created by Harrison Chase.

> Finished chain.
Answer: LangChain was created by Harrison Chase.
```

3️⃣ Combined Reasoning

```
> Entering new AgentExecutor chain...
Could not parse LLM output: `LangChain is a framework that is commonly used i
n AI workflows to build applications with Large Language Models (LLMs). In AI
workflows, LangChain can be used to create chatbots, document question-answer
ing systems, and other AI applications that require natural language processi
ng capabilities. By leveraging the features of LangChain, developers can easi
ly integrate LLMs into their workflows and create powerful AI systems that ca
n understand and generate human-like text. Overall, LangChain plays a crucial
role in enabling the development of advanced AI applications that rely on nat
ural language processing.`
For troubleshooting, visit: https://python.langchain.com/docs/troubleshootin
g/errors/OUTPUT_PARSING_FAILURE
Observation: Invalid or incomplete response
Do I need to use a tool? Yes
Action: LangChainRetriever
Action Input: Explain LangChain's use in AI workflows
```

```
Observation: LangChain is commonly used in AI workflows to facilitate the dev
elopment of applications using Large Language Models (LLMs). It provides supp
ort for various components such as RAG, agents, memory, and tools, which are
essential for building AI applications. By leveraging LangChain, developers c
an create sophisticated AI workflows that involve natural language processin
g, document Q&A, chatbots, and more. This framework streamlines the process o
f integrating LLMs into AI workflows, making it easier to build intelligent a
pplications that can understand and generate human-like text.
Do I need to use a tool? No
AI: LangChain is commonly used in AI workflows to facilitate the development
of applications using Large Language Models (LLMs). It provides support for v
arious components such as RAG, agents, memory, and tools, which are essential
for building AI applications. By leveraging LangChain, developers can create
sophisticated AI workflows that involve natural language processing, document
Q&A, chatbots, and more. This framework streamlines the process of integratin
g LLMs into AI workflows, making it easier to build intelligent applications
that can understand and generate human-like text.

> Finished chain.
Answer: LangChain is commonly used in AI workflows to facilitate the developm
ent of applications using Large Language Models (LLMs). It provides support f
or various components such as RAG, agents, memory, and tools, which are essen
tial for building AI applications. By leveraging LangChain, developers can cr
eate sophisticated AI workflows that involve natural language processing, doc
ument Q&A, chatbots, and more. This framework streamlines the process of inte
grating LLMs into AI workflows, making it easier to build intelligent applica
tions that can understand and generate human-like text.
```

In [ ]: