Lecture 13: Autonomous Research Agents

🎯 Learning Objectives

By the end of this lecture, you should be able to:

- Understand the architecture of research-oriented agents.
- Combine retrieval-augmented generation (RAG) with reasoning and memory.
- Build an agent that can read, analyze, and summarize documents.
- Apply autonomous loops to extract insights from multi-source information.

🧩 Key Concepts

What Is a Research Agent?

- An LLM-powered agent capable of:
  - Searching and retrieving relevant documents
  - Extracting insights or answering questions
  - Summarizing long or complex content
  - Operating across multiple iterations if needed

Key Techniques

- **RAG (Retrieval-Augmented Generation)**:
  - Retrieve relevant data from a corpus
  - Feed it into the prompt context before generation
- **Multi-hop Reasoning**:
  - Chain insights across documents or queries
- **Memory + Tools**:
  - Combine long-term memory, embeddings, and tools like search or summarizers

🛠️ Required Tools/Libraries

- LangChain
- FAISS or ChromaDB
- OpenAI / Hugging Face models
- PyPDF or Docx reader (for ingesting documents)

  pip install langchain faiss-cpu openai pypdf

🔬 Hands-on Exercise: Build a Research Assistant Agent

Goal: Build an agent that answers questions about uploaded documents.

Step 1: Load and Chunk PDF

```
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader("sample_paper.pdf")
docs = loader.load_and_split()
```

Step 2: Embed and Store in FAISS

```
from langchain.vectorstores import FAISS
from langchain.embeddings.openai import OpenAIEmbeddings

embeddings = OpenAIEmbeddings()
vectorstore = FAISS.from_documents(docs, embeddings)
```

Step 3: Create a Retrieval QA Chain

```
from langchain.chains import RetrievalQA
from langchain.llms import OpenAI

retriever = vectorstore.as_retriever()
chain = RetrievalQA.from_chain_type(llm=OpenAI(), retriever=retriever)

response = chain.run("What are the key contributions of the paper?")
print(response)
```

Bonus:

- Enable multi-turn conversations with memory across follow-up questions.
- Add summarization as a post-processing step.
- Try ingesting multiple sources (PDFs, websites, Wikipedia) into a single agent.
