Lecture 06: Memory for Agents

🎯 Learning Objectives

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

- Understand the role of memory in agentic systems.
- Differentiate between short-term and long-term memory.
- Use vector stores to store and retrieve semantic information.
- Implement a memory-augmented agent for contextual conversations.

🧩 Key Concepts

Why Memory Matters

- Without memory, agents can't:
  - Recall past interactions.
  - Track progress across multi-step tasks.
  - Personalize their behavior.
  - Reflect on or learn from previous mistakes.

Types of Memory

- **Short-Term Memory**:
  - Context window of the current interaction.
  - Typically limited by token constraints.
- **Long-Term Memory**:
  - Persisted across sessions or time.
  - Can be stored in databases or vector stores.

Vector Stores for Semantic Memory

- Stores chunks of information as embeddings (numerical vectors).
- Retrieval is based on similarity rather than exact match.
- Common tools: FAISS, ChromaDB, Weaviate, Pinecone

🛠️ Required Tools/Libraries

- FAISS or ChromaDB
- Hugging Face Transformers or OpenAI Embeddings
- LangChain (optional for memory modules)
- Python

🔬 Hands-on Exercise: Memory-Augmented QA Agent

Goal: Build an agent that remembers and reuses information using a vector store.

Step 1: Install FAISS or Chroma

```
pip install faiss-cpu
pip install sentence-transformers
```

Step 2: Load and Embed Documents

```
from sentence_transformers import SentenceTransformer
import faiss
import numpy as np

docs = ["Paris is the capital of France.", "The Eiffel Tower is in Paris."]
model = SentenceTransformer('all-MiniLM-L6-v2')
embeddings = model.encode(docs)

index = faiss.IndexFlatL2(len(embeddings[0]))
index.add(np.array(embeddings))
```

Step 3: Query from Memory

```
query = "Where is the Eiffel Tower?"
q_emb = model.encode([query])
D, I = index.search(np.array(q_emb), k=1)
print("Retrieved:", docs[I[0][0]])
```

Step 4: Combine with LLM Response

```
- Provide retrieved documents as context to the LLM.
- Ask the LLM to answer the question using the memory.
```

Bonus:

- Implement a long-term memory buffer that saves new facts after each interaction.
- Test the agent across multiple turns and see if it recalls prior knowledge.
