

warning "Atualização de backend (outubro de 2026)" A inferência do GitHub Models foi descontinuada em 30 de julho de 2026. As referências abaixo são mantidas apenas como contexto histórico enquanto este lab é migrado. Use o Lab 013 ? Backends de LLM gratuitos para configurar o Foundry Local ou o Ollama antes de continuar.

Lab 027: Agent Memory Patterns

Nível: L200 Trilha: Pro Code / SK Tempo: ~35 min ? Custo: Gratuito (local)

O que Você Vai Aprender

Os 4 tipos de memória de agente e quando usar cada um Memória em contexto (gerenciamento do histórico de conversas) Memória externa (vector store + SQLite para dados estruturados)

Memória episódica (o que aconteceu em sessões anteriores) Construindo um agente com memória que lembra as preferências do usuário

Introdução

Sem memória, toda conversa com o agente começa do zero. Um agente de atendimento ao cliente que esquece o nome do cliente no meio da conversa, ou um assistente que não consegue lembrar o que você decidiu na semana passada, cria uma experiência frustrante.

Os quatro tipos de memória:

Tipo | Armazenamento | Duração | Exemplo | Em contexto | Janela de contexto do LLM |
Requisição única | Histórico de conversas | Semântica externa | Banco de dados vetorial |
Persistente | Tickets de suporte anteriores | Estruturada externa | Banco de dados SQL |
Persistente | Preferências do usuário, histórico de pedidos | Episódica | Chave-valor /
arquivo | Sessão ou persistente | "Na última sessão discutimos X" |

Pré-requisitos

Python 3.11+ pip install openai chromadb GITHUB_TOKEN set

tip "Dados de exemplo incluídos" This lab uses orders.csv and products.csv from the repo.

[diagram or code sample]

Exercício do Lab

Passo 1: Memória em contexto (histórico de conversas)

A memória mais simples ? apenas mantenha o histórico de mensagens. O desafio: janelas de contexto têm limites.

[diagram or code sample]

Passo 2: Memória estruturada com SQLite

Para fatos que precisam de precisão (preferências, pedidos, configurações), use um banco de dados, não a janela de contexto.

[diagram or code sample]

Passo 3: Memória episódica semântica com ChromaDB

ChromaDB é um banco de dados vetorial gratuito e local ? sem necessidade de Docker.

[diagram or code sample]

Passo 4: Agente com memória

Combine os três tipos de memória em um único agente:

[diagram or code sample]

Guia de Decisão de Padrões de Memória

[diagram or code sample]

Próximos Passos

RAG agêntico usa memória de recuperação: ? Lab 026 ? Agentic RAG SK possui abstrações de memória integradas: ? Lab 023 ? SK Plugins & Memory

Kit de apoio

O build do site gera downloads estáveis em PDF e ZIP. O ZIP contém o guia do lab e todos os arquivos de apoio declarados para este lab.

Baixar PDF gerado Baixar ZIP gerado

Supporting file: CHEATSHEET.md

Arquivos de apoio: Lab 027: Agent Memory Patterns

Comece pelo caminho local ou gratuito documentado. Registre premissas e evidências de validação. Mantenha serviços hospedados opcionais, salvo quando o lab os exigir.

Supporting file: README.md

Lab 027: Agent Memory Patterns

Use estes arquivos com o guia do lab e o kit gerado.