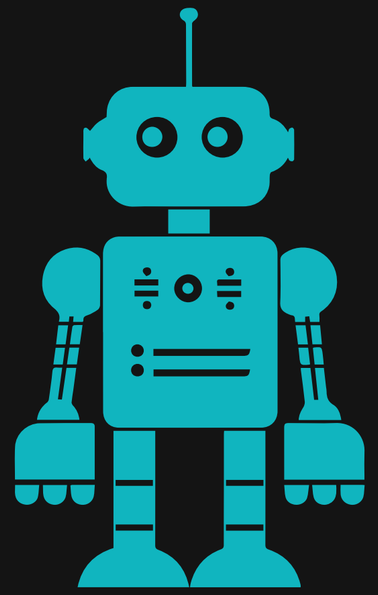
UNIVEL START 25/10

Desenvolvimento de Agentes Inteligentes com Python e OpenAI



Marcos Agnes, Especialista em Arquitetura de Software, Professor, Head de IA e CTO

O que Vamos Aprender Hoje?

Conceitos Fundamentais: O que são LLMs?

O "Cérebro" do Agente: LLMs como Motor de Raciocínio.

Dando Conhecimento: RAG (Retrieval-Augmented Generation) com Qdrant.

Dando Poderes: Ferramentas Externas e Memória.

Orquestração: O que são Agentes? (Langchain vs. LangGraph).

O que é "MCP"?

O que são Agentes Inteligentes?

Definição de Agente Inteligente (Pré-LLM)

- Em sua definição clássica (popularizada por Stuart Russell e Peter Norvig em "Inteligência Artificial: Uma Abordagem Moderna"), um agente inteligente (AI) é qualquer entidade que:
  - Percebe seu ambiente através de sensores.
  - Atua sobre esse ambiente através de atuadores.
  - Atua de forma autônoma e racional (ou seja, age para maximizar uma medida de desempenho ou "função de utilidade").

O que são Agentes Inteligentes?

Definição de Agente Inteligente (Contexto LLM)

- No contexto dos LLMs, um agente inteligente é um sistema que utiliza um Large Language Model (LLM) como seu núcleo de raciocínio (ou "cérebro") para decompor um objetivo complexo em uma sequência de etapas executáveis.
- Diferente dos agentes clássicos, ele não depende de regras codificadas (if-then). Em vez disso, ele opera em um ciclo de feedback (loop) para decidir, de forma autônoma, qual a próxima ação necessária para atingir seu objetivo.

O que são Agentes Inteligentes?

Definição: Um agente usa um LLM para tomar decisões, planejar e executar ações para atingir um objetivo. O Ciclo Básico de um Agente (ReAct):

PENSAMENTO (THOUGHT)

O LLM analisa o objetivo e decide o que fazer.

AÇÃO (ACTION)

O agente usa uma "Ferramenta" (ex: buscar no Google).

OBSERVAÇÃO

O agente recebe o resultado da ferramenta.

(LOOP)

O agente repete o ciclo com a nova informação.

O que é um LLM?

LLM é a sigla para Large Language Model (Modelo de Linguagem de Grande Porte)

- Em essência, é um tipo avançado de Inteligência Artificial (especificamente, uma rede neural massiva) que foi treinado em uma quantidade gigantesca de texto e dados (livros, artigos, código e a maior parte da internet).
- A sua função mais fundamental é surpreendentemente simples: prever estatisticamente a próxima "palavra" (ou token) em uma sequência.

O que é um LLM?

Os 3 Pilares de um LLM

- Large (Grande): Isso se refere a dois aspectos:
  - Tamanho do Modelo: A rede neural possui bilhões (ou até trilhões) de "parâmetros". Pense em parâmetros como as "**sinapses**" em um cérebro, que armazenam o conhecimento aprendido. (Ex: GPT-3 tem 175 bilhões de parâmetros).
  - Tamanho dos Dados: Foi treinado em Terabytes (ou Petabytes) de dados de texto.
- Language (Linguagem): É o seu domínio. O modelo opera sobre linguagem humana (texto). Ele entende a gramática, o contexto, o sentimento, a semântica e as relações entre as palavras.
- Model (Modelo): É um modelo estatístico. Isso é crucial: um LLM não "**entende**" o mundo, não "**sabe**" fatos, nem "**pensa**" como um humano. Ele calcula a probabilidade de qual palavra deve vir a seguir com base em todo o texto que ele "**leu**" durante o treinamento.

O que é um LLM?

A Analogia Perfeita

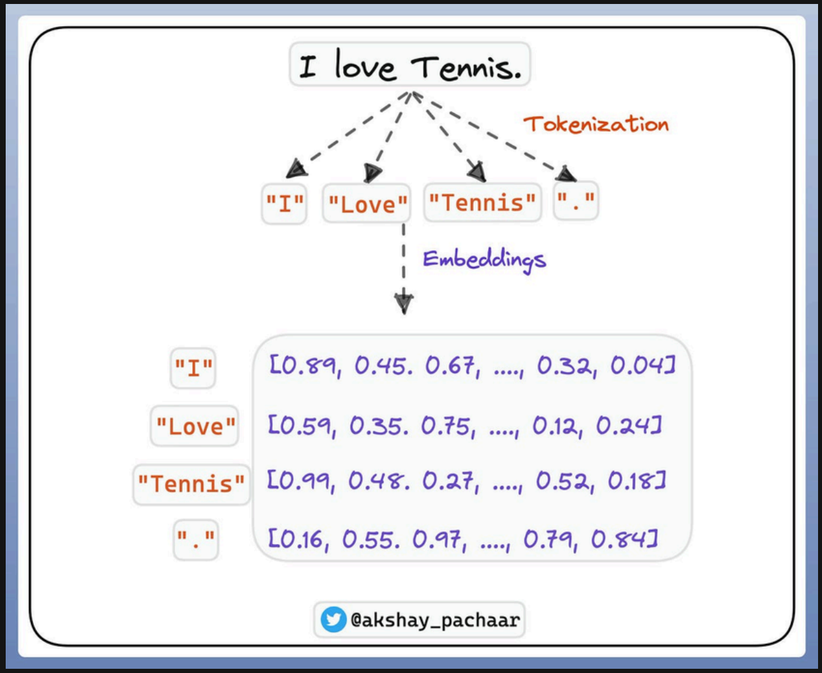
- Pense em um LLM como o sistema de "autocompletar" mais avançado do mundo.
  - O autocompletar do seu celular sugere a próxima palavra.
  - Um LLM sugere a próxima palavra, depois a próxima, e a próxima... e faz isso tão bem que, no final, ele escreveu um parágrafo inteiro, um poema, um código em Python ou um plano de ação.

O que é um LLM?

A Limitação

- A maior limitação de um LLM é que ele é um "cérebro em um pote".
- Não tem conhecimento do mundo real: Seu conhecimento é "congelado" na data em que seu treinamento terminou.
- Não pode executar ações: Ele só pode gerar texto. Ele não pode navegar na internet, consultar um banco de dados ou enviar um e-mail.

O que são LLMs (Large Language Models)?



Os LLMs (ex: GPT-4, Llama 3) são o "cérebro" de raciocínio do agente.

Como funcionam: Previsão probabilística de tokens (palavras).

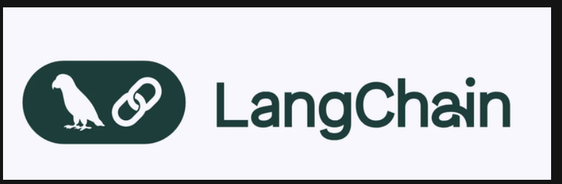
A importância do Prompt Engineering: O LLM é um motor que responde a instruções.

Limitações Principais:

- Conhecimento estático (limitado à data de corte do treino).
- Alucinações (inventar fatos).
- Falta de acesso ao "mundo real" (internet, bancos de dados, etc.).

Orquestração de Agentes (Parte 1): Langchain

- O "Canivete Suíço" para aplicações com LLMs.
- Fornece os "blocos de construção" (componentes) para agentes.
- Conceitos-chave: Chains, Tools, Memory, Agents (Executors).
- Foco em chains lineares (passo A -> passo B -> passo C).
- Ótimo para agentes mais simples e fluxos de trabalho diretos.

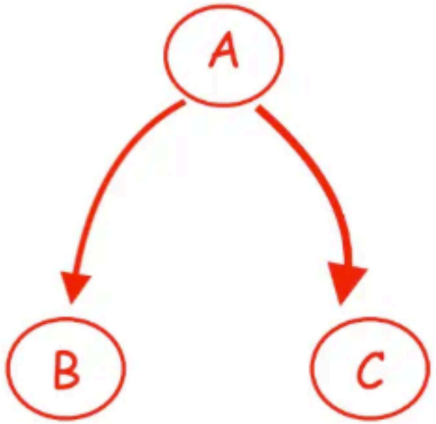


Orquestração de Agentes (Parte 2): LangGraph

- A evolução do Langchain para agentes complexos.
- Trata o fluxo do agente como um Grafo de Estados (State Machine).
- Por que usar? Permite Ciclos (Loops), que são essenciais para agentes.
  - (Ex: "Se a busca falhar, tente uma busca diferente" -> isso é um ciclo).
- Ideal para: Multi-agentes (agentes conversando entre si), planejamento robusto e fluxos de decisão condicionais.

LangChain

LangGraph



Persistindo contexto e aprendizado contínuo

- Tipos de memória (short-term/conversacional, long-term, vetorial)

Memória de Curto Prazo (ou Conversacional)

- O que é: A "RAM" do chatbot. A capacidade de lembrar o que foi dito dentro da mesma sessão.
- Tecnologia: Janela de Contexto (Context Window).
- Como Funciona:
  - Todo o histórico do chat atual é enviado de volta ao modelo a cada nova pergunta.
  - Permite que o modelo mantenha a coerência e siga o fluxo da conversa.
- Limitação:
  - Volátil: É perdida quando a sessão é encerrada.
  - Limitada: Conversas muito longas "empurram" as mensagens mais antigas para fora da janela, causando "esquecimento".

Memória de Longo Prazo

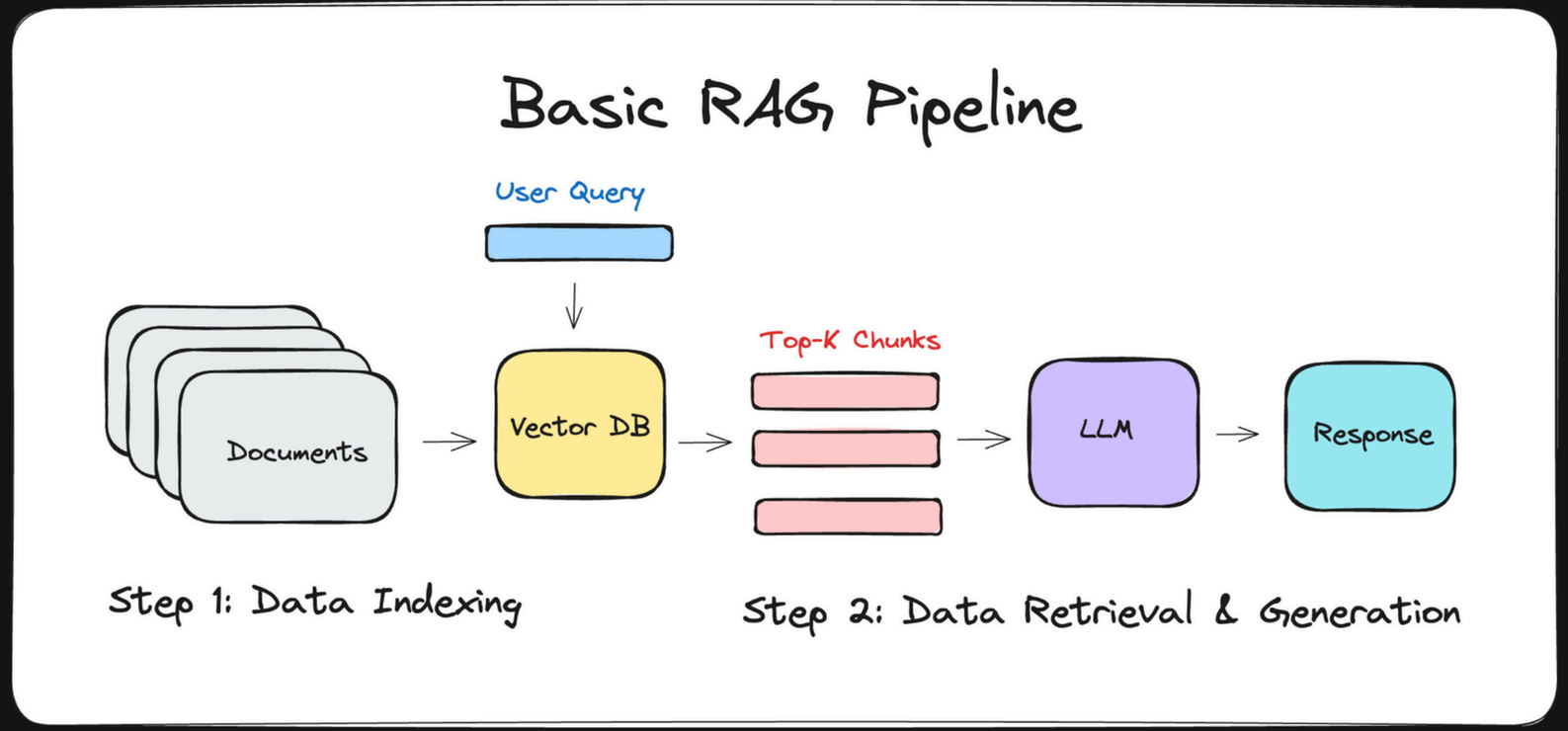
- O que é: O "HD" do sistema de IA. A capacidade de reter informações entre sessões.
- Conceito-Chave: Não é uma característica nativa do LLM, mas uma arquitetura construída ao redor dele.
- Objetivo:
  - Personalização (lembrar preferências do usuário).
  - Acesso a bases de conhecimento externas (documentos, manuais, etc.).
- Principal Implementação: Memória Vetorial (RAG).

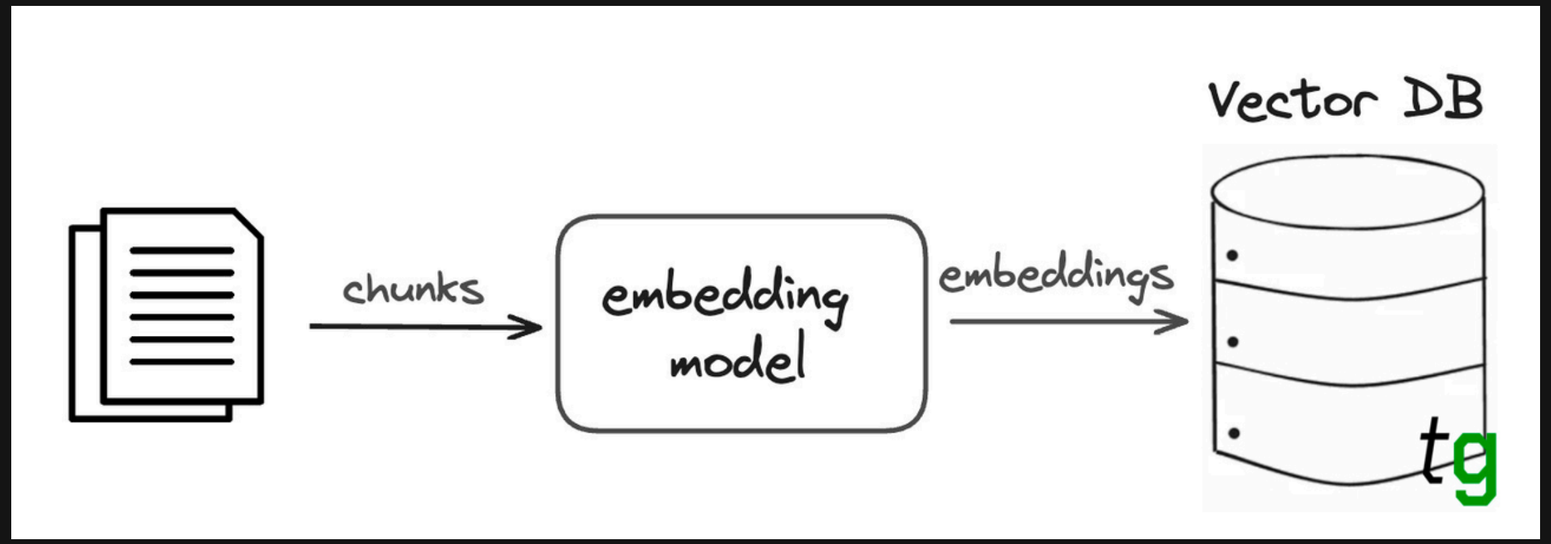
Memória de Longo Prazo

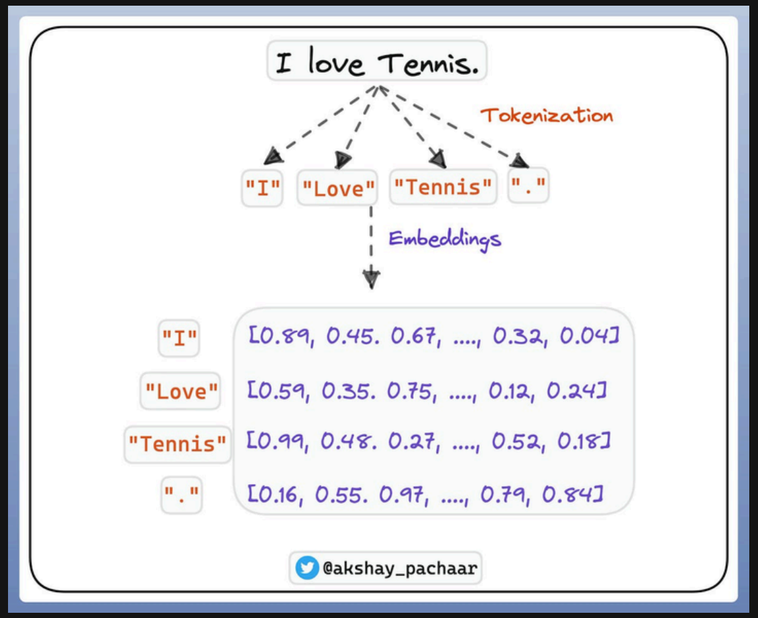
- Tecnologias-Chave:
  - Embeddings (Vetores): Textos são convertidos em representações matemáticas (vetores) que capturam seu significado.
  - Banco de Dados Vetorial (Vector DB): Armazena e pesquisa esses vetores por similaridade.
- Processo (RAG - Retrieval-Augmented Generation):
  - A pergunta do usuário é convertida em um vetor.
  - O sistema busca no Vector DB os vetores (textos) mais "próximos" (relevantes).
  - Os textos encontrados são injetados na memória de curto prazo (janela de contexto) para o LLM usar na resposta.

Dando Conhecimento: RAG (Retrieval-Augmented Generation)

- Problema: Como o LLM responde sobre meus dados privados (PDFs, docs, etc.)?
  - Solução: RAG!
    - Passo 1 (Indexação): "Quebrar" documentos em pedaços (chunks) e convertê-los em vetores (embeddings).
    - Passo 2 (Recuperação): Buscar os pedaços mais relevantes para a pergunta do usuário.
    - Passo 3 (Geração): Enviar ao LLM: [Pergunta] + [Pedaços Relevantes] = Resposta contextualizada.


Basic RAG Pipeline
User Query
Documents
Vector DB
Top-K Chunks
LLM
Response
Step 1: Data Indexing
Step 2: Data Retrieval & Generation


Vector DB
chunks
embedding model
embeddings
tg


I love Tennis.
Tokenization
"I"
"Love"
"Tennis"
"."
Embeddings
"I"
[0.89, 0.45. 0.67, ..., 0.32, 0.04]
"Love"
[0.59, 0.35. 0.75, ..., 0.12, 0.24]
"Tennis"
[0.99, 0.48. 0.27, ..., 0.52, 0.18]
"."
[0.16, 0.55. 0.97, ..., 0.79, 0.84]
@akshay_pachaar

MCP – Protocolos para Ferramentas Inteligentes

- MCP e seu papel nos novos ecossistemas
- Integração segura e modular entre agentes e ferramentas
- Exemplos: conectar API, DB, scraper, automação local

Prática!

https://github.com/marcos14/workshop_univel_out_2025

Obrigado!

(45) 98823-8608