Langchain + ChatGPT

Geração Aumentada de Recuperação (RAG)

Fundamentos e Contextualização

- Biblioteca| FrameWork
- Simplificar o desenvolvimento de IA
  - Correntes
  - Links
- Componentes do LangChain
  - Interface LLM
  - Modelos de prompt
  - Módulos de recuperação
  - Agentes
  - Memória

Geração Aumentada de Recuperação [RAG]

Aplicações

- Agentes Inteligentes
- Sistemas de Perguntas e Respostas (Q&A)
- Chatbots Personalizados
- Aplicações de Resumo (Summarization)
- Automação de Fluxos de Trabalho
- Geração de Dados Sintéticos
- **Geração Aumentada de Recuperação (RAG, Retrieval-Augmented Generation)**

Geração Aumentada de Recuperação [RAG]

Motivação:

- Os modelos de LLM não conhecem seus dados
- Os aplicativos de IA devem aproveitar dados personalizados para serem eficazes

O que é:

- Abordagem de arquitetura tenta melhorar a eficácia das aplicações de grandes modelo de linguagem (LLM) usando dados personalizados. Isso é feito recuperando dados/documentos relevantes para uma pergunta ou tarefa e fornecendo-os como contexto para o LLM.

Geração Aumentada de Recuperação [RAG]

Que problemas resolve:

- Os modelos de LLM não conhecem seus dados
- Os aplicativos de IA devem aproveitar dados personalizados para serem eficazes
- Limitações de Conhecimento Estático
- Redução de Alucinações

Geração Aumentada de Recuperação [RAG]

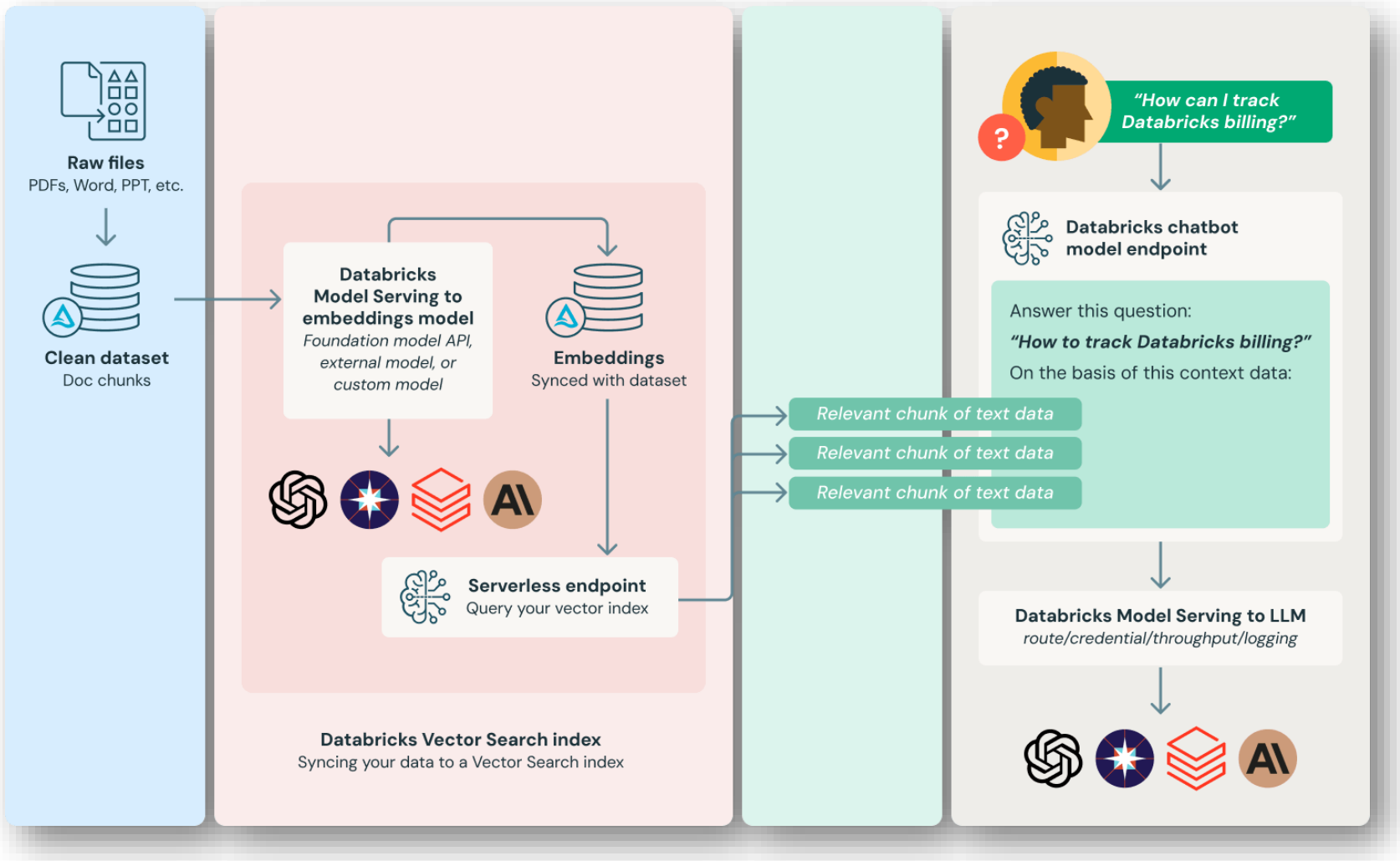
Quais são os casos de uso da RAG :

- Assistentes de suporte ao cliente
- Análise de documentos corporativos
- Gestão de conhecimento empresarial
- Educação e treinamento

- Monitoramento e manutenção preditiva
- Melhorias em processos logísticos
- Q&A para novos funcionarios

Geração Aumentada de Recuperação [RAG]

Arquitetura



Geração Aumentada de Recuperação [RAG]

Código | Armazenando os dados

```python
import argparse
import os
import shutil
from langchain_community.document_loaders import PyPDFDirectoryLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
from langchain.schema.document import Document
from get_embedding_function import get_embedding_function
from langchain_chroma import Chroma
from dotenv import load_dotenv
import openai

# Carrega as variáveis de ambiente. Assume que o projeto contém um arquivo .env com as chaves da API.
load_dotenv()
# ---- Define a chave da API da OpenAI
# Altere o nome da variável de ambiente de "OPENAI_API_KEY" para o nome especificado no
# seu arquivo .env.
openai.api_key = os.environ["OPENAI_API_KEY"]

CHROMA_PATH = "chroma"
DATA_PATH = "data"


def main():
    # Verifica se o banco de dados deve ser limpo (usando o parâmetro --reset).
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--reset", action="store_true", help="Reinicia o banco de dados."
    )
    args = parser.parse_args()
    if args.reset:
        print("✨ Limpando o banco de dados")
        clear_database()

    # Cria (ou atualiza) o banco de dados.
    documents = load_documents()
    chunks = split_documents(documents)
    add_to_chroma(chunks)
```

```python
# from langchain_community.embeddings.ollama import OllamaEmbeddings
# from langchain_community.embeddings.bedrock import BedrockEmbeddings
from langchain_openai import OpenAIEmbeddings
from dotenv import load_dotenv
import openai
import os

# Carrega as variáveis de ambiente. Assume que o projeto contém um arquivo .env com as chaves da API.
load_dotenv()
# ---- Define a chave da API da OpenAI
# Altere o nome da variável de ambiente de "OPENAI_API_KEY" para o nome especificado no
# seu arquivo .env.
openai.api_key = os.environ["OPENAI_API_KEY"]


def get_embedding_function():
    # ---- Modelo da Amazon
    # embeddings = BedrockEmbeddings(credentials_profile_name="default", region_name="us-east-1")

    # ---- Modelos locais
    # embeddings = OllamaEmbeddings(model="nomic-embed-text")

    embeddings = OpenAIEmbeddings(model="text-embedding-3-large")
    return embeddings

```

Geração Aumentada de Recuperação [RAG]

Código | Armazenando os dados

```python
def add_to_chroma(chunks: list[Document]):
    # Carrega o banco de dados existente.
    db = Chroma(
        persist_directory=CHROMA_PATH, embedding_function=get_embedding_function()
    )

    # Calcula os IDs das páginas.
    chunks_with_ids = calculate_chunk_ids(chunks)

    # Adiciona ou atualiza os documentos.
    existing_items = db.get(include=[])  # IDs são sempre incluídos por padrão
    existing_ids = set(existing_items["ids"])
    print(f"Número de documentos existentes no banco de dados: {len(existing_ids)}")

    # Adiciona apenas documentos que não existem no banco de dados.
    new_chunks = []
    for chunk in chunks_with_ids:
        if chunk.metadata["id"] not in existing_ids:
            new_chunks.append(chunk)

    if len(new_chunks):
        print(f"👉 Adicionando novos documentos: {len(new_chunks)}")
        new_chunk_ids = [chunk.metadata["id"] for chunk in new_chunks]
        db.add_documents(new_chunks, ids=new_chunk_ids)
        db.persist()
    else:
        print("✅ Nenhum novo documento para adicionar")
```

```python
def calculate_chunk_ids(chunks):
    # Isso criará IDs como "data/monopoly.pdf:6:2"
    # Fonte da Página : Número da Página : Índice do Fragmento

    last_page_id = None
    current_chunk_index = 0

    for chunk in chunks:
        source = chunk.metadata.get("source")
        page = chunk.metadata.get("page")
        current_page_id = f"{source}:{page}"

        # Se o ID da página for o mesmo que o último, incrementa o índice.
        if current_page_id == last_page_id:
            current_chunk_index += 1
        else:
            current_chunk_index = 0

        # Calcula o ID do fragmento.
        chunk_id = f"{current_page_id}:{current_chunk_index}"
        last_page_id = current_page_id

        # Adiciona o ID aos metadados do fragmento.
        chunk.metadata["id"] = chunk_id

    return chunks
```

Geração Aumentada de Recuperação [RAG]

Código |Consulta dos dados

```python
import argparse
from langchain_chroma import Chroma
from langchain.prompts import ChatPromptTemplate

# from langchain_community.llms.ollama import Ollama
from langchain_openai import ChatOpenAI

from get_embedding_function import get_embedding_function

CHROMA_PATH = "chroma"

PROMPT_TEMPLATE = """
Responda à pergunta com base apenas no seguinte contexto:

{context}

---

Responda à pergunta com base no contexto acima: {question}
"""


def main():
    # Cria a interface de linha de comando (CLI).
    parser = argparse.ArgumentParser()
    parser.add_argument("query_text", type=str, help="O texto da consulta.")
    args = parser.parse_args()
    query_text = args.query_text
    query_rag(query_text)
```

```python
def query_rag(query_text: str):
    # Prepara o banco de dados.
    embedding_function = get_embedding_function()
    db = Chroma(persist_directory=CHROMA_PATH, embedding_function=embedding_function)

    # Pesquisa no banco de dados.
    results = db.similarity_search_with_score(query_text, k=5)

    context_text = "\n\n---\n\n".join([doc.page_content for doc, _score in results])
    prompt_template = ChatPromptTemplate.from_template(PROMPT_TEMPLATE)
    prompt = prompt_template.format(context=context_text, question=query_text)
    # print(prompt, "\n\n---\n\n")

    model = ChatOpenAI(model="gpt-4o")
    response_text = model.invoke(prompt)

    sources = [doc.metadata.get("id", None) for doc, _score in results]
    formatted_response = f"Resposta: {response_text.content}\nFontes: {sources}"
    print(formatted_response)
    return response_text


if __name__ == "__main__":
    main()
```

Geração Aumentada de Recuperação [RAG]

Resultado

```
>>> python query_data.py "De que inteligencias artificiais esse trabalho fala?"

Resposta: O trabalho mencionado fala sobre inteligências artificiais generativas,
como o ChatGPT e MidJourney, além de abordar conceitos como Aprendizado de Máquina
(Machine Learning) e Deep Learning. Além disso, menciona as Redes Neurais
Generativas Adversativas (GANs).

---

Fontes: ['data/IMPACTO DA INTELIGENCIA ARTIFICIAL NA ENGENHARIA.pdf:27:1',
         'data/IMPACTO DA INTELIGENCIA ARTIFICIAL NA ENGENHARIA.pdf:11:1',
         'data/IMPACTO DA INTELIGENCIA ARTIFICIAL NA ENGENHARIA.pdf:47:0',
         'data/IMPACTO DA INTELIGENCIA ARTIFICIAL NA ENGENHARIA.pdf:6:0',
         'data/IMPACTO DA INTELIGENCIA ARTIFICIAL NA ENGENHARIA.pdf:21:0']
```