Laboratório 17: O RAG Feito à Mão

Objetivo do Laboratório

Neste laboratório, você construirá o pipeline completo da arquitetura RAG (Retrieval-Augmented Generation) do zero, usando Python. O script fará uma busca num banco ChromaDB, extrairá contextos e injetará o texto em um prompt rígido, forçando o LLM a atuar como um sumarizador fiel dos seus dados locais.

Atividade Prática

1. Pré-Requisitos e Setup

Certifique-se de que as bibliotecas necessárias estão instaladas. Usaremos o `chromadb` para a busca semântica e o provedor generativo (ex: `google-generativeai`).

```
pip install chromadb google-generativeai python-dotenv
```

Crie um arquivo chamado `rag_manual.py`.

2. O Código: Construindo a Arquitetura

Este script pressupõe que você já tenha populado um diretório do ChromaDB (conforme visto na Aula 12 de Embeddings).

```
import os
import chromadb
import google.generativeai as genai
from dotenv import load_dotenv

# 1. Carregamento e Configuração
load_dotenv()
genai.configure(api_key=os.getenv("API_KEY"))

# 2. Conectando ao Banco de Dados Vetorial (ChromaDB)
# Ajuste o path para a pasta do seu banco populado
```

```
chroma_client = chromadb.PersistentClient(path="./meu_banco_chroma")
colecao = chroma_client.get_collection(name="documentos_empresa")

# 3. A Pergunta do Usuário
pergunta_usuario = "Quais são as regras para home-office na empresa?"

# 4. Passo 1 do RAG: Recuperação (Retrieval)
print("Buscando no banco de dados...")
resultados_busca = colecao.query(
    query_texts=[pergunta_usuario],
    n_results=2 # Retorna os 2 parágrafos mais relevantes
)

# Extrai os textos relevantes encontrados (o 'Contexto')
contextos_encontrados = resultados_busca['documents'][0]
contexto_concatenado = "\n".join(contextos_encontrados)

# 5. Passo 2 do RAG: Construção do Prompt Aumentado
prompt_rag = f"""
Você é um assistente corporativo. Responda à pergunta do usuário usando APENAS
o contexto fornecido abaixo. Não invente informações.
Se o contexto não contiver a resposta, diga "Desculpe, não encontrei a informação".

CONTEXTO:
{contexto_concatenado}

PERGUNTA DO USUÁRIO:
{pergunta_usuario}
"""

# 6. Passo 3 do RAG: Geração (Generation)
print("Invocando a IA Generativa...\n")
modelo = genai.GenerativeModel('gemini-1.5-flash')
resposta = modelo.generate_content(prompt_rag)

print("--- Resposta do RAG ---")
print(resposta.text)
```

3. Entrega via Git

1 Execute o script e faça testes. Tente fazer uma pergunta que SABEMOS estar no banco, e comprove que o LLM respondeu perfeitamente com os dados injetados.

2 Faça uma pergunta fora do contexto da empresa (ex: "Quem ganhou a copa de 2002?"). O modelo DEVE se recusar a responder, graças à sua instrução restrita.

3 Comite as modificações com `git commit -m "Implementa arquitetura RAG pura`

`em Python"` e envie ao GitLab.

Critério de Avaliação

O script deve demonstrar claramente os três passos: Conexão com VectorDB para busca, Template de Interpolação (f-string) misturando Contexto e Pergunta, e Chamada da API em Nuvem com esse Prompt Aumentado.