Estudo de Caso - Estágio em Dados e IA

Aplicação prática de extração, classificação e consulta inteligente de PDFs em ambiente local.

Apresentado por:

Gustavo dos Santos Garcia

O Desafio

Entender o problema e os requisitos.

A Solução e Demo

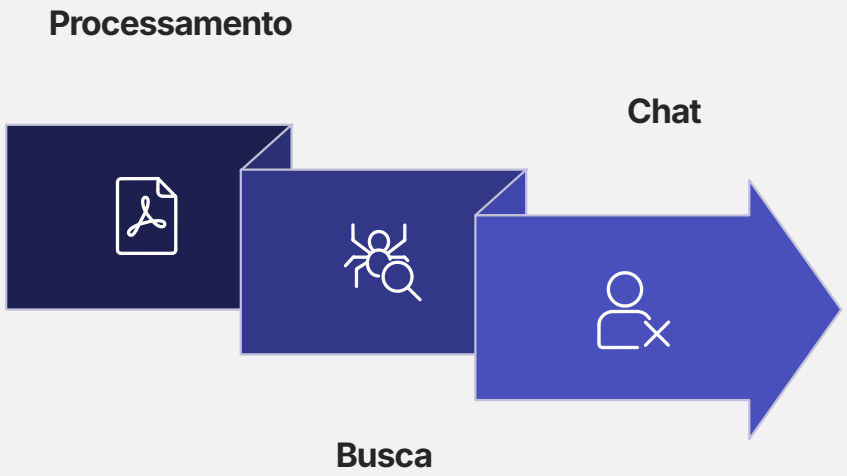
Arquitetura e demonstração ao vivo.

Conclusão

Escolhas técnicas e próximos passos.

Arquitetura da Solução: Do PDF à Resposta Inteligente

A solução foi estruturada para criar um sistema de Retrieval-Augmented Generation (RAG) em ambiente local, garantindo leveza e eficiência sem dependência de serviços em nuvem.



Explorando o Código: Os Três Pilares da Aplicação

Extração de Conteúdo

Usa PyMuPDF para lidar robustamente com diferentes formatos de PDF, extraindo o texto de forma limpa.

Escolha do Algoritmo

O TF-IDF foi escolhido por ser leve e eficaz, cumprindo o requisito de rodar localmente sem a necessidade de uma GPU dedicada.

Otimização Local (LLM)

A solução com Ollama e Gemma:2b permite rodar o Large Language Model em CPU, ideal para ambientes de desenvolvimento ou infraestrutura limitada.

processar.py (Extração e Classificação)

A classificação inicial é simples, lê os 10 PDFs e extrai o texto, estruturando-o em uma base de dados `documentos.json`. A classificação inicial dos documentos é realizada pelo prefixo do nome do arquivo (Resolução, Lei, Portaria), essencial para a contextualização futura.

buscar.py (Motor de Busca)

Implementa o algoritmo **TF-IDF** (Scikit-learn) para mapear termos da consulta às partes mais relevantes dos documentos, garantindo que apenas o contexto necessário seja passado para o LLM.

chat.py (Interface com LLM)

Recebe a pergunta do usuário, chama o motor de busca e envia o contexto relevante junto com a pergunta ao Ollama, configurado com o modelo Gemma:2b, para gerar a resposta final.

Escolhas Técnicas Estratégicas para Performance Local

A seleção das tecnologias foi guiada pela necessidade de uma solução robusta, mas que pudesse ser executada com eficiência em hardware local (CPU-only).

Busca: TF-IDF (Scikit-learn)

Técnica clássica, leve e extremamente rápida para busca por similaridade de termos. Evita a necessidade de infraestrutura pesada (como servidores de Embeddings) e cumpre o requisito de não depender de GPU.

LLM: Ollama + Gemma:2b

O Ollama simplifica o gerenciamento e a execução de modelos de linguagem localmente. O **Gemma:2b** é um modelo otimizado para CPU, entregando um excelente equilíbrio entre qualidade de resposta e tempo de inferência, crucial para uma demonstração local ágil.

Prioridade: A solução prioriza a portabilidade e a execução em qualquer máquina padrão, focando na lógica de **Geração Aumentada por Recuperação (RAG)** de forma eficiente.

Demonstração ao Vivo: Testando a Inteligência do Sistema

A demo ilustra o fluxo completo, desde o processamento inicial dos documentos até a interação final com o modelo de linguagem.

Passo 1: Processamento

Executamos `python processar.py`. O script lê os 10 PDFs na pasta `dados/` e cria a base de conhecimento.

Saída esperada: "Sucesso! 10 documentos processados."

Passo 2: Início do Chat

Iniciamos a interface de consulta com `python chat.py`.

O sistema está pronto para receber consultas em linguagem natural e buscar o contexto relevante.

Passo 3: Testes

Pergunta 1: O que a Resolução 909 define?
Pergunta 2: Faça um resumo da Portaria 44.
Pergunta 3: Qual a capital da França? (Deve falhar e dizer que não encontrou a informação).

Demonstração ao Vivo: Objetivos das perguntas

Teste 1: Busca e Contexto

Pergunta: O que a Resolução 909 define?

O sistema identifica o PDF correto usando TF-IDF e o LLM gera a resposta baseada exclusivamente nesse contexto.

Teste 2: Resumo e Extração

Pergunta: Faça um resumo da Portaria 44.

O modelo demonstra sua capacidade de sumarização e extração de informações-chave de documentos longos.

Teste 3: Teste de Limite (Evitando Alucinações)

Pergunta: Qual a capital da França?

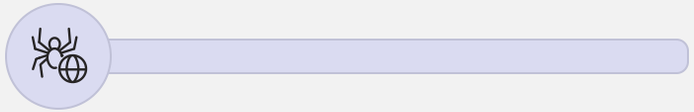
Resposta do Chat: "A informação não foi encontrada nos documentos..." Demonstra que o prompt está sendo obedecido (RAG).

Conclusão e Roteiro para o Futuro

A solução atual é um **MVP funcional** que resolve o desafio principal. No entanto, o projeto possui um claro roteiro de evolução técnica para aumentar a performance e a usabilidade.

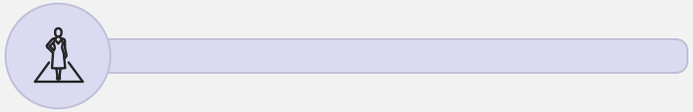
Melhoria da Busca

Migrar do TF-IDF para **Embeddings Vetoriais** (ex: Sentence Transformers com FAISS). Isso permitiria uma busca semântica mais rica, entendendo o *significado* da frase e não apenas as palavras-chave.



Interface Web Simples

Desenvolver uma interface de usuário via **Streamlit** ou **Flask**. Isso transformaria a ferramenta de linha de comando em uma aplicação interativa e mais acessível para o usuário final.



Otimização Contínua

Explorar modelos LLM mais robustos (ex: Llama 3) ou técnicas de quantização para manter a performance em CPU, enquanto se eleva a complexidade das respostas.

Obrigado pelo seu tempo e atenção. Fico à disposição para quaisquer perguntas!

Repositório do Projeto: github.com/Gustavod-Garcia/Desafio-Estagio-IA