Documento: projeto-paciente

1. Introdução

Este projeto tem como objetivo demonstrar conhecimento técnico, a linha de pensamento adotada e o processo de tomada de decisão diante de um desafio técnico proposto. Além disso, busca compreender melhor os desafios enfrentados pela empresa.

2. Contexto e Problema

O desafio aborda um problema técnico relacionado a computação, que é auxiliar a medicina através de soluções de aprendizado de máquina e com isso tornar cada vez mais rápido e preciso os diagnósticos médicos.

A chamada do desafio traz o seguinte texto:

"Em anexo você recebeu as seguintes bases de dados:

1. client.csv: Contém informações de clientes
2. chat_history.csv: Contém histórico de mensagens de alguns clientes
3. client_conditions.csv: Tabela de exemplo de possível output do modelo com a identificação de alguns clientes e suas condições
4. seed_*.csv: Tabelas estáticas que utilizamos para mapear

O desafio é desenvolver um modelo que retorne as condições de saúde de cada cliente baseado em suas mensagens, por exemplo:

- Obesidade;
- Tabagismo;
- Diabetes;
- Hipertensão e outros.

Esse modelo deve avaliar todas as mensagens, extrair as condições de saúde e guardá-las em um banco de dados, esse banco de dados pode ser um simples arquivo CSV que será consultado pela API.

O projeto deve ser conter uma API, que vai receber um ID de cliente e retorna todas as condições de saúde dele como uma lista, se o cliente não existe na base devemos retornar um erro de 404 Not Found. "

Os desafios aqui são, em primeiro lugar, explorar a base de dados e entender um pouco mais sobre ela para em um segundo momento desenvolver um solução que possa ajudar na solução deste problema.

Além disso, vai ser necessário o desenvolvimento de um sistema com banco de dados que disponibilize os dados processados através de uma API.

Seria interessante também, pensando um pouco no lado do engenheiro de machine learning, pensar em uma solução que facilmente vai ser disponibilizada para outros sistemas, então vamos pensar em um ambiente de containers para esse caso.

3. Premissas (Assumptions)

Estrutura dos dados dos chats

A estrutura dos dados dos chats é simples, mas os registros estão fragmentados entre diferentes tabelas.

Consolidar essas informações pode facilitar a análise e melhorar a eficácia do modelo.

Quantidade limitada de dados

A base de dados disponível é pequena, o que inviabiliza o treinamento de grandes modelos de aprendizado de máquina do zero.

Como alternativa, a abordagem adotada será o uso de modelos pré-treinados e ajustados ao contexto do problema.

Uso de LLMs para análise de texto

Modelos especializados em processamento de linguagem natural (NLP), como LLMs, são mais adequados para a tarefa de compreensão de mensagens dos chats.

A escolha inicial será o uso de APIs de modelos generativos, pois elas reduzem a necessidade de infraestrutura computacional pesada e aceleram a implementação da solução.

4. Decisões Técnicas e Justificativas

Para a implementação computacional, utilizaremos Python, uma das linguagens mais adequadas para soluções envolvendo modelos de aprendizado de máquina.

Dentro do ecossistema Python, empregaremos a biblioteca OpenAI, que oferece flexibilidade na escolha de diferentes modelos de LLMs para resolver tarefas de inferência.

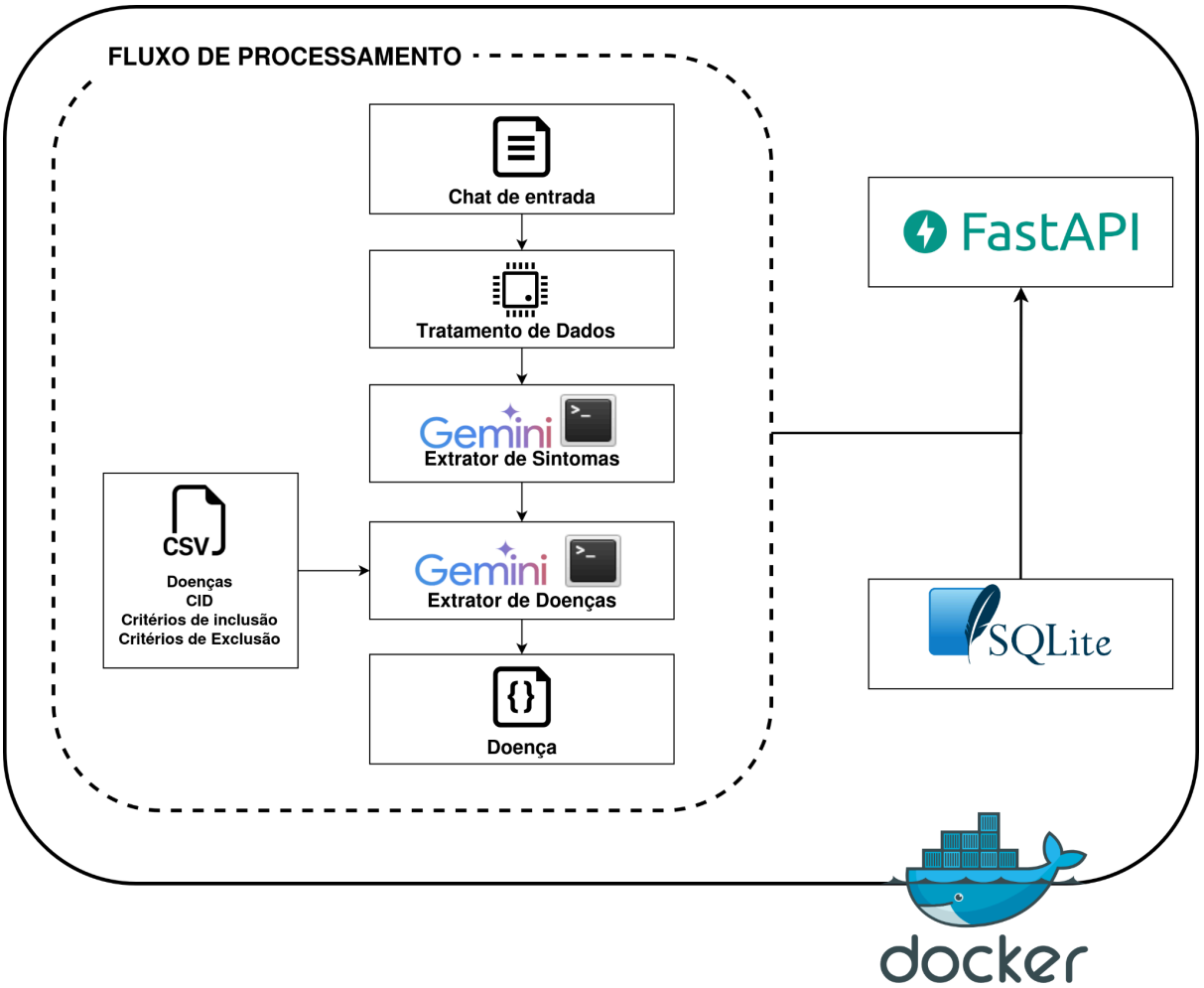
Para a construção da API, adotaremos o FastAPI, uma opção leve e eficiente para desenvolvimento de serviços web em Python.

No que diz respeito ao banco de dados, utilizaremos SQLite, uma solução simples e eficaz para a fase inicial do projeto. Sua escolha elimina a necessidade de configurar serviços adicionais, tornando a implementação mais ágil.

Como modelo inicial, utilizaremos o Gemini-2.0-Flash, pois se trata de uma solução sem custos no momento, permitindo viabilizar a abordagem sem comprometer recursos financeiros.

Por fim, para garantir a portabilidade e facilitar a replicabilidade da solução, utilizaremos Docker e Docker Compose para a conteinerização

5. Arquitetura da Solução



Como mostrado no diagrama, temos um pipeline de processamento para a extração de dados que se inicia com a entrada das informações. Em seguida, ocorre uma etapa de tratamento, onde são removidas pontuações, espaços e palavras desnecessárias.

Com os dados devidamente preparados, realizamos uma chamada à API da OpenAI para extrair os sintomas mencionados na conversa entre o cliente e o atendente. A partir desses sintomas, combinados com informações sobre a doença, critérios de inclusão e exclusão, utilizamos novamente a IA generativa para identificar a provável condição de saúde associada.

A informação resultante, juntamente com o ID do cliente, é armazenada no banco de dados SQLite. Além disso, disponibilizamos uma API desenvolvida com FastAPI para consultar essas informações e registrar/processar novos chats.

6. Riscos e Mitigações

O primeiro ponto de risco a ser considerado é o tamanho reduzido da base de dados disponível. Isso impossibilita, no momento, o desenvolvimento de métricas confiáveis para avaliar os resultados. Com um conjunto de dados maior, poderemos definir com mais precisão métricas de classificação, como acurácia, precisão, F1-score e recall.

Outro fator importante é a limitação de uso do modelo gratuito do Gemini. Dessa forma, é essencial avaliar com mais cuidado a viabilidade desse sistema antes de considerá-lo para produção.

Por fim, em termos de aprimoramento do modelo, há espaço para melhorias significativas, incluindo técnicas de engenharia de prompt, fine-tuning e abordagens multi-agente.

7. Conclusão e Próximos Passos

Acreditamos que o sistema está funcional para uma primeira versão, mas ainda há muitos passos a serem dados para aprimorá-lo.

Para melhorar o modelo, podemos explorar diferentes abordagens de engenharia de prompt, como one-shot e few-shot learning. Além disso, a implementação de técnicas Retrieval-Augmented Generation (RAG) permitiria a extração de informações adicionais sobre doenças a partir da internet, enriquecendo a base de conhecimento do sistema.

Outra possibilidade é testar diferentes modelos, incluindo opções pagas ou especializados na área da saúde, como o BERT-Based Medical Chatbot. Além disso, o uso do inglês pode ser uma estratégia relevante, já que alguns modelos apresentam melhor desempenho nesse idioma.

Por fim, um aprimoramento significativo seria a adoção de uma arquitetura multi-agent, onde uma LLM gerenciadora coordenaria o fluxo de informações—extraindo sintomas, utilizando múltiplos modelos para alcançar um consenso sobre a doença e aplicando RAG para obter dados complementares da internet, entre outras estratégias.