



Universidade Federal de Sergipe


DISCIPLINA: TÓPICOS AVANÇADOS EM ENGENHARIA DE SOFTWARE E SISTEMAS DE
INFORMAÇÃO I (PROCC0111)

Professor: Glauco de Figueiredo Carneiro
Aluno: Marcelo Moreira West



Agente Médico

Implementação de um assistente IA para recuperação de informações médicas



Introdução ao Agente Médico





Definição do Agente Médico

O agente médico é um sistema inteligente projetado para oferecer respostas a perguntas relacionadas à saúde, utilizando técnicas de inteligência artificial. Ele opera com base em informações extraídas de documentos PDF e utiliza modelos como GPT-4 para formular respostas adequadas e contextualmente relevantes.





Objetivos do Agente Médico


O principal objetivo do agente médico é fornecer assistência precisa e confiável em consultas médicas. Ele busca melhorar a eficiência na recuperação de informações, permitindo que os usuários acessem informações médicas relevantes de forma rápida e fácil, assim como apoiar decisões clínicas baseadas em evidências acadêmicas extraídas de múltiplas fontes documentais.





Utilização de GPT-4 e RAG

O GPT-4 é uma poderosa ferramenta de geração de linguagem natural que, quando combinada com técnicas de Recuperação Aumentada de Geração (RAG), permite ao agente médico fornecer respostas mais contextualizadas. Ele utiliza informações extraídas de documentos para fundamentar suas respostas, oferecendo um atendimento mais preciso e baseado em dados.





Instalação de Bibliotecas

Para implementar o agente médico, é necessário instalar bibliotecas essenciais, como OpenAI, Gradio e PyMuPDF. A instalação pode ser realizada em ambientes como o Google Colab ou Visual Studio Code, permitindo uma configuração rápida e eficaz. Estas bibliotecas fornecem as funcionalidades necessárias para a interação com a API da OpenAI e a manipulação de arquivos PDF.




Implementação do Código





Arquitetura do Código


A arquitetura do código é projetada para ser modular e escalável, permitindo fácil manutenção e atualizações. O agente é composto por várias classes e métodos que cuidam das funções, desde a extração de texto e geração de embeddings até a interface do usuário. Essa estrutura permite que o agente opere de forma eficiente e responda a perguntas com rapidez.





Funções principais do Agente Médico

O agente médico possui várias funções principais: extrair texto de arquivos PDF, dividir o texto em partes menores (chunks) para facilitar a recuperação, gerar embeddings para esses chunks e indexá-los usando FAISS. Ele também permite a interação do usuário através de uma interface Gradio, onde perguntas podem ser feitas e as respostas são geradas com base nas informações recuperadas.



Código Fonte



1. CÓDIGO DO AGENTE MÉDICO

Instalar bibliotecas (executar no Google Colab)

```
!pip install --upgrade openai gradio PyMuPDF --quiet
```

```
!pip install --upgrade openai gradio PyMuPDF faiss-cpu --quiet
```

```
import os
import openai
import gradio as gr
import fitz # PyMuPDF
import faiss
import numpy as np
```


```
class MedicalAssistantAgent:
```

```
    """
```

Um agente médico que utiliza GPT-4o e RAG (Retrieval-Augmented Generation)
para responder perguntas

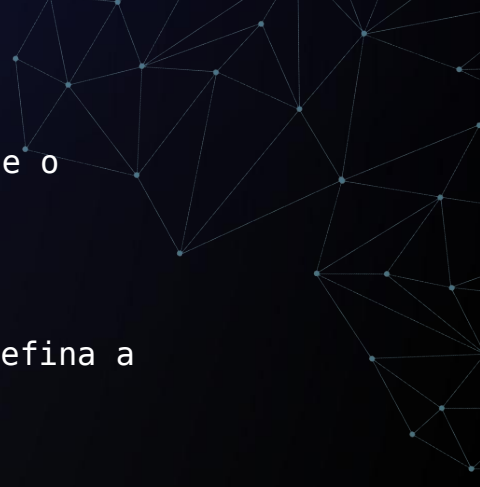
com base em informações extraídas de PDFs.

```
    """
```




```
self.def __init__(self, caminho_pdfs):
    """
    Inicializa o agente, configurando a chave da API da OpenAI e o
    caminho para os PDFs.
    """
    self.api_key = os.getenv("OPENAI_API_KEY")
    if not self.api_key:
        raise ValueError("A chave da API não foi encontrada. Defina a
variável de ambiente 'OPENAI_API_KEY'.")
    openai.api_key = self.api_key
    self.caminho_pdfs = caminho_pdfs

def extrair_texto_pdf(self, caminho_pdf):
    """
    Extrai texto de um arquivo PDF.
    """
    texto = ""
    with fitz.open(caminho_pdf) as doc:
        for pagina in doc:
            texto += pagina.get_text()
    return texto
```



```
self.def dividir_em_chunks(self, texto, tamanho=500, sobreposicao=100):
    """
    Divide o texto em chunks de tamanho fixo com sobreposição.
    """
    palavras = texto.split()
    chunks = []
    for i in range(0, len(palavras), tamanho - sobreposicao):
        chunk = " ".join(palavras[i:i + tamanho])
        if chunk:
            chunks.append(chunk)
    return chunks

def gerar_embedding(self, texto):
    """
    Gera embeddings para um texto usando a API da OpenAI.
    """
    response = openai.Embedding.create(
        input=texto,
        model="text-embedding-ada-002"
    )
    return np.array(response['data'][0]['embedding'], dtype=np.float32)
```





```
selfdef indexar_pdf(self, caminho_pdf):
    """
    Indexa os chunks de um PDF em um banco vetorial FAISS.
    """
    texto = self.extrair_texto_pdf(caminho_pdf)
    chunks = self.dividir_em_chunks(texto)
    embeddings = [self.gerar_embedding(chunk) for chunk in chunks]

    index = faiss.IndexFlatL2(len(embeddings[0]))
    index.add(np.array(embeddings))

    return index, chunks
```

```
def recuperar_chunks(self, pergunta, index, chunks, k=3):
    """
    Recupera os k chunks mais relevantes para uma pergunta.
    """
    pergunta_emb = self.gerar_embedding(pergunta)
    _, indices = index.search(np.array([pergunta_emb]), k)
    return [chunks[i] for i in indices[0]]
```






```
self def consultar_gpt_rag(self, pergunta, nome_pdf):
    """
    Consulta o modelo GPT-4o com informações recuperadas de um PDF.
    """
    caminho_pdf = os.path.join(self.caminho_pdfs, nome_pdf)
    index, chunks = self.indexar_pdf(caminho_pdf)
    trechos_relevantes = self.recuperar_chunks(pergunta, index, chunks)


    contexto = "\n---\n".join(trechos_relevantes)

    prompt = f"""
        Você é um assistente médico especializado em doenças
        respiratórias.

        Use as informações abaixo extraídas de documentos para
        responder a pergunta do usuário:

        {contexto}
```






```
self Pergunta: {pergunta}
"""
```

```
try:
```

```
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[
            {"role": "system", "content": "Você é um assistente médico
confiável, que usa informações de contexto com
responsabilidade."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.5,
        max_tokens=1000
    )
    return response['choices'][0]['message']['content']
except Exception as e:
    return f"Erro: {str(e)}"
```



```
def listar_pdfs(self):
    """
    Lista os arquivos PDF disponíveis no diretório configurado.
    """
    return [f for f in os.listdir(self.caminho_pdfs) if
            f.endswith(".pdf")]

def iniciar_interface(self):
    """
    Inicia a interface Gradio para interação com o agente.
    """
    interface = gr.Interface(
        fn=self.consultar_gpt_rag,
        inputs=[
            gr.Textbox(lines=4, label="Digite sua pergunta"),
            gr.Dropdown(choices=self.listar_pdfs(), label="Selecione um PDF")
        ],
        outputs="text",
        title="RAG Médico com GPT-4o e PDF",
        description="Faça perguntas sobre doenças respiratórias. O modelo buscará as respostas mais relevantes nos arquivos PDF usando RAG (Retrieval-Augmented Generation).")
    interface.launch(share=True)
```



```
# Executa o agente
```

```
if __name__ == "__main__":
```

```
    try:
```


```
        caminho_pdfs = "/content/" # Substitua pelo caminho onde os  
        PDFs estão armazenados
```

```
        agente = MedicalAssistantAgent(caminho_pdfs)
```

```
        agente.iniciar_interface()
```

```
    except ValueError as e:
```

```
        print(e)
```





2. DESCRIÇÃO DA IMPLEMENTAÇÃO

2.1. Encapsulamento em uma classe:

Foi criada a classe 'MedicalAssistantAgent' com os Métodos: 'extrair_texto_pdf', 'dividir_em_chunks', 'gerar_embedding', 'indexar_pdf', 'recuperar_chunks' e 'consultar_gpt_rag'.

2.2. Interface Gradio:

A interface Gradio foi integrada no método 'iniciar_interface' para permitir interação com o agente.

2.3. Modularidade:

O código é modular e reutilizável, permitindo fácil manutenção e expansão.

2.4. Configuração do caminho dos PDFs:

O caminho dos PDFs é configurado no construtor da classe e usado em todos os métodos relacionados.





3. INSTRUÇÕES DE EXECUÇÃO

3.1. Certifique-se de que os PDFs estão no diretório configurado (``/content/`` ou outro caminho que você definir).

3.2. Instale as dependências necessárias:


```
pip install openai gradio pymupdf faiss-cpu numpy
```

3.3. Configure a variável de ambiente 'OPENAI_API_KEY' com sua chave da API:

```
export OPENAI_API_KEY="sua-chave-aqui"
```

3.4. Execute o programa:

```
python3 medical_assistant_agent.py
```



4. RESULTADO DA EXECUÇÃO:

```
(venv) marcelo-west@serverwest:~/UFBA/agent$ python3 chatbot_agent.py
Bem-vindo ao chatbot GPT-4o! Digite 'sair' para encerrar.
Você: quem é você?
Chatbot: Eu sou um modelo de linguagem desenvolvido pela OpenAI, conhecido como ChatGPT. Estou aqui para ajudar a responder perguntas, fornecer informações e interagir com você sobre uma variedade de tópicos. Como posso ajudar hoje?
Você: quais são os sintomas da covid-19?
Chatbot: Os sintomas da COVID-19 podem variar amplamente entre os indivíduos, mas os mais comuns incluem:

1. Febre ou calafrios
2. Tosse seca
3. Fadiga
4. Dificuldade para respirar ou falta de ar
5. Dores musculares ou corporais
6. Dor de cabeça
7. Perda de paladar ou olfato
8. Dor de garganta
9. Congestão ou coriza
10. Náusea ou vômito
11. Diarreia


Esses sintomas podem aparecer de 2 a 14 dias após a exposição ao vírus. Além disso, algumas pessoas infectadas com o coronavírus podem ser assintomáticas, ou seja, não apresentam sintomas, mas ainda podem transmitir o vírus para outras pessoas.

É importante observar que a gravidade dos sintomas pode variar de leve a grave, e algumas pessoas, especialmente aquelas com condições médicas preexistentes ou idosos, podem ter um risco maior de desenvolver complicações graves. Se você ou alguém que você conhece estiver apresentando sintomas graves, como dificuldade respiratória, dor ou pressão persistente no peito, confusão, incapacidade de acordar ou permanecer acordado, ou coloração azulada nos lábios ou rosto, deve procurar atendimento médico de emergência imediatamente.
```



Conclusões

A implementação de um agente médico utilizando GPT-4 e RAG representa um avanço significativo na recuperação de informações médicas. A capacidade de extrair e processar dados de documentos PDF garante que os usuários tenham acesso a respostas informadas, aumentando a eficiência no processamento de informações e contribuindo para decisões mais embasadas na área da saúde.





Obrigado!

Marcelo West

westmarcelo@gmail.com

