



Imersão em GenAI para Desenvolvedores

Projeto 5: *Retrieval-Augmented Generation* (RAG)

Preâmbulo: Neste projeto, você irá implementar um sistema de armazenamento vetorial com ChromaDB para buscas semânticas em currículos, utilizar técnicas de chunking para melhorar a busca em PDFs, e criar uma aplicação web com Streamlit integrando RAG para análise de qualificações de candidatos.

Versão: 1.0

Sumário

I	Preâmbulo	2
II	Preâmbulo II	4
III	Instruções gerais	5
IV	Ex-01 – Configuração do ambiente Continue.dev	7
V	Ex00 – Embeddings para otimizar a busca de informações	9
V.1	Exercício	9
VI	Ex01 – Análise de codebase do ChromaDB	12
VII	Ex02 – Quebra estratégica de informações para uma busca ainda melhor	14
VII.1	Exercício	15
VIII	Ex03 – RAG: Busca semântica e LLMs entram em um bar	18
VIII.1	Exercício	19
IX	Entrega e Avaliação entre pares	21
IX.1	Processo de Entrega	21
IX.2	Avaliação entre pares	21
IX.3	Dicas para uma avaliação bem-sucedida	21

Capítulo I

Preâmbulo

Os avanços recentes em modelos de linguagem têm sido notáveis, mas ainda enfrentam desafios importantes. Lewis e seus colegas (2020) observaram que, apesar de sua capacidade de armazenar conhecimento factual, esses modelos ainda têm dificuldade em acessar e manipular esse conhecimento com precisão em tarefas que exigem informações extensas.

Para lidar com essa limitação, os autores propuseram uma nova abordagem chamada Retrieval-Augmented Generation (RAG). Embora possa ser traduzida literalmente como "Geração aumentada por recuperação", o termo em inglês é mais comumente usado na literatura. Esse método combina de forma inovadora dois tipos de memória:

- Uma memória paramétrica: um modelo linguístico pré-treinado
- Uma memória não paramétrica: um grande banco de dados de informações (neste caso, a Wikipédia)

O RAG permite que o modelo acesse dinamicamente informações externas durante o processo de geração de texto, resultando em respostas mais precisas e fundamentadas.

Os autores testaram duas variantes do RAG:

1. Uma que usa o mesmo conjunto de informações recuperadas para gerar toda a resposta
2. Outra que pode utilizar diferentes informações para cada parte da resposta

Ao aplicar essa técnica a diversas tarefas de processamento de linguagem natural, os pesquisadores alcançaram resultados impressionantes, superando abordagens anteriores em várias tarefas, incluindo perguntas e respostas de domínio aberto.

"Para tarefas de geração de linguagem, descobrimos que os modelos RAG geram linguagem mais específica, diversa e factual do que modelos anteriores."

— Baseado no trabalho de Lewis et al., 2020¹

¹Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020).

Capítulo II

Preâmbulo II

A integração de assistentes de IA no desenvolvimento de software promete uma transformação significativa na produtividade dos desenvolvedores. Estudos recentes mostram que essas ferramentas podem aumentar consideravelmente a eficiência, especialmente para desenvolvedores menos experientes.

Um estudo realizado com 4.867 desenvolvedores em três grandes empresas revelou que:

1. Aumento de produtividade:

- 26,08% de aumento no número de tarefas concluídas
- 13,55% de aumento no número de commits de código
- 38,38% de aumento no número de compilações de código

2. Impacto heterogêneo:

- Desenvolvedores juniores e com menor tempo de empresa apresentaram maiores ganhos de produtividade
- Maior taxa de adoção entre desenvolvedores menos experientes

Estes resultados indicam que:

Assistentes de IA, como o GitHub Copilot, têm o potencial de nivelar o campo de atuação, permitindo que desenvolvedores menos experientes alcancem níveis de produtividade mais próximos aos de seus colegas seniores.

Portanto, ao implementar assistentes de IA em ambientes de desenvolvimento, devemos:

Focar na integração efetiva e no treinamento, especialmente para desenvolvedores juniores e recém-contratados.

— Baseado no trabalho de Cui et al. (2024)¹

¹Cui, K. Z., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2024). The Effects of Generative AI on High Skilled Work: Evidence from Three Field Experiments with Software Developers.

Capítulo III


Instruções gerais

- Esta página é sua única referência oficial. Não confie em informações não verificadas.
- Os exercícios estão organizados em ordem crescente de complexidade. É essencial dominar cada exercício antes de prosseguir para o próximo.
- Preste atenção às permissões de seus arquivos e pastas.
- Siga rigorosamente o procedimento de entrega para todos os exercícios.
- Seus exercícios serão avaliados por colegas da Imersão.
- Para exercícios em Shell, utilize `/bin/zsh`.
- Mantenha em sua pasta apenas os arquivos explicitamente solicitados nos enunciados.
- Em caso de dúvidas, consulte seus colegas à direita ou à esquerda.
- Utilize recursos como Google, manuais online e a Internet como referência.
- Leia os exemplos com atenção. Eles podem conter requisitos não explicitamente mencionados no enunciado.
- Para exercícios em Python:
 - Use a versão do Python especificada no exercício de configuração do ambiente.
 - Utilize os modelos e provedores sugeridos para garantir tempos de resposta adequados e consistência nos testes.
- Esteja atento a erros em todos os exercícios. Eles raramente são tolerados durante a avaliação.
- **Aviso sobre o uso de ferramentas de AI (como ChatGPT):**
 - O uso de ferramentas como o ChatGPT não deve ser encarado como um substituto para seu próprio esforço e entendimento.
 - O aprendizado efetivo ocorre quando você interage ativamente com o conteúdo: pesquisando, refletindo e aplicando o que aprendeu.
 - Nas avaliações, serão feitas perguntas para avaliar sua compreensão real sobre o assunto.

- E durante as avaliações, seus colegas também avaliarão seu nível de conhecimento.

Capítulo IV

Ex-01 – Configuração do ambiente Continue.dev

	Exercício : -01
Instalar, configurar e explorar o Continue.dev para melhorar sua produtividade no desenvolvimento de software	
Pasta de entrega : <i>ex – 01/</i>	
Arquivos para entregar : Nenhum arquivo necessário	

Neste exercício, vamos configurar uma ferramenta para o uso da IA no desenvolvimento de software, o Continue.dev. Os assistentes de IA estão se tornando parte essencial do trabalho de muitos desenvolvedores, oferecendo sugestões de código, ajudando na manutenção e potencialmente aumentando a produtividade. É fundamental entender suas capacidades e limitações para usá-los de forma eficaz.

Durante este projeto, diferentemente de outros, encorajamos fortemente o uso de assistentes de IA. Você vai utilizar e experimentar com essas ferramentas na realização dos exercícios: este é um momento para explorar plenamente as capacidades desses assistentes e entender como eles podem ser integrados ao seu fluxo de trabalho.

No entanto, mesmo com esta liberdade, é crucial manter em mente:

- Embora o uso de IA seja incentivado, seu entendimento do processo e dos resultados continua sendo fundamental.
- O objetivo é aprender a trabalhar *com* a IA, não apenas deixá-la fazer todo o trabalho.
- Sua capacidade de explicar e justificar as soluções geradas ou sugeridas pela IA permanece fundamental para seu aprendizado.

Instruções

1. Instalação do Continue.dev:

- Instale a extensão Continue.dev no VSCode através do marketplace.
- Link: <https://marketplace.visualstudio.com/items?itemName=Continue.continue>

2. Configuração da API:

- Obtenha uma chave de API (Groq para o modelo llama3-8b-8192 ou gemini-1.5-flash da Google).
- Configure a extensão Continue.dev com a chave de API escolhida.

3. Exploração básica:

- Familiarize-se com a interface do Continue.dev no VSCode.
- Teste algumas funcionalidades básicas, como completar código ou gerar comentários.




Se encontrar dificuldades durante a instalação ou configuração, consulte a documentação oficial do Continue.dev ou peça ajuda a um colega.



A configuração e exploração inicial dessas ferramentas podem ser trabalhosas. Paciência e atenção aos detalhes são essenciais nesta fase. O tempo investido agora pode resultar em ganhos significativos de produtividade no futuro.

Capítulo V

Ex00 – Embeddings para otimizar a busca de informações

	Exercício : 00
Implementar um sistema de armazenamento vetorial usando ChromaDB para processar e realizar buscas semânticas eficientes em currículos, simulando um assistente de recrutamento baseado em IA	
Pasta de entrega : <i>ex00/</i>	
Arquivos para entregar : <code>resume_analyzer.py</code>	

Você vai criar um sistema de análise de currículos que busca por qualificações. Este sistema permitirá:

- Processar currículos em PDF, convertendo-os em representações vetoriais densas (conhecidos como *embeddings*¹).
- Armazenar e indexar eficientemente essas representações para busca rápida.
- Realizar consultas em linguagem natural sobre o conteúdo dos currículos.

Com esta ferramenta, você poderá analisar um grande volume de currículos, utilizando buscas semântica². Isso eliminará a necessidade de leitura individual de cada documento, permitindo uma triagem rápida e mais precisa dos candidatos mais adequados às suas necessidades.

V.1 Exercício

Instruções

1. Configurar o ChromaDB com persistência local.

¹<https://www.cloudflare.com/learning/ai/what-are-embeddings/>

²<https://www.elastic.co/what-is/semantic-search>

2. Implementar a função `process_pdf_directory` para ler e processar arquivos PDF.
3. Adicionar os documentos processados ao ChromaDB.
4. Criar a função `interactive_query_loop` para realizar consultas interativas.

Código base:

```
def main():
    persist_directory = "./chroma_data"
    pdf_directory = "./pdfs"

    # TODO: Configurar ChromaDB com persistência local
    chroma_client = ...

    # TODO: Criar embedding function
    # Dica: Experimente com diferentes modelos, como "paraphrase-multilingual-MiniLM-L12-v2"
    embedding_function = ...

    # TODO: Criar ou obter uma coleção existente
    collection = ...

    # TODO: Implementar a função process_pdf_directory
    process_pdf_directory(pdf_directory, collection)

    # TODO: Implementar a função interactive_query_loop
    interactive_query_loop(collection)

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

Dica para a função `interactive_query_loop`:

```
while True:
    query = input("\nConsulta: ")
    if query.lower() == 'sair':
        break
    results = ...
    print("\nResultados:")
    for document, metadata in ...:
        print(f"Documento: {metadata['source']}")
        print(f"Trecho: {document[:200]}..." # Apenas os 200 primeiros caracteres
        print()
```

Saída esperada:

```
Encontrados 50 arquivos PDF no diretório.
Processando PDF 1/50: curriculo_1.pdf
- Documento curriculo_1.pdf processado e armazenado.
Processando PDF 2/50: curriculo_2.pdf
- Documento curriculo_2.pdf processado e armazenado.
...
Processando PDF 50/50: curriculo_50.pdf
- Documento curriculo_50.pdf processado e armazenado.
Processamento de todos os PDFs concluído.
Processamento concluído. Iniciando modo de consulta.
Digite sua consulta ou 'sair' para encerrar.

Consulta: python

Resultados:
```

```
Documento: curriculo_12.pdf
Trecho: Maria Silva
São Paulo, SP | (11) 9XXXX-XXXX | maria.silva@42sp.org.br
Resumo Profissional
Desenvolvedora de Software Júnior com 2 anos de experiência em Python e frameworks web...

Documento: curriculo_35.pdf
Trecho: João Santos
São Paulo, SP | (11) 9XXXX-XXXX | joao.santos@42sp.org.br
Resumo Profissional
Engenheiro de Machine Learning com forte background em Python e bibliotecas de data science...

Documento: curriculo_7.pdf
Trecho: Ana Oliveira
São Paulo, SP | (11) 9XXXX-XXXX | ana.oliveira@42sp.org.br
Resumo Profissional
Analista de Dados com experiência em manipulação e visualização de dados utilizando Python...
```




Há currículos disponíveis para download na página principal do projeto.



Ainda não estamos usando modelos de linguagem para processamento ou geração de texto. O foco está na criação e consulta de *embeddings*.

Capítulo VI

Ex01 – Análise de codebase do ChromaDB

	Exercício : 01
Utilizar ferramentas de AI para analisar e compreender uma funcionalidade específica dentro da codebase do ChromaDB	
Pasta de entrega : <i>ex01/</i>	
Arquivos para entregar : <code>chromadb_analysis.md</code>	

Neste exercício, você vai exercitar suas habilidades de análise de código e compreensão de dependências em um projeto de código aberto. Você vai escolher uma funcionalidade específica dentro do ChromaDB e usar ferramentas de IA para analisar e documentar seu funcionamento.

Instruções

1. Seleção de funcionalidade:

- Escolha uma funcionalidade específica na codebase do [ChromaDB - \(documentação\)](#) para analisar (por exemplo, armazenamento de embeddings, indexação, busca ou persistência).
- Use a AI para identificar os arquivos e módulos relevantes para essa funcionalidade.

2. Análise de código:

- Utilize a ferramenta de IA para gerar uma explicação detalhada do fluxo de execução da funcionalidade escolhida.
- Peça à IA para identificar possíveis pontos de melhoria ou otimização no código.

3. Compreensão de dependências:

- Use a IA para mapear as dependências internas e externas da funcionalidade.
- Documente as bibliotecas e frameworks utilizados, com explicações geradas pela IA sobre seu propósito no projeto.

4. Relatório de análise:

- Crie um arquivo Markdown (chromadb_analysis.md) com o seguinte formato:

```
# Análise ChromaDB
```

```
## Funcionalidade Escolhida
```

```
[Nome da funcionalidade]
```

```
Breve descrição da funcionalidade e razão da escolha (6-10 frases)
```

```
## Análise do Código
```

```
- Principais arquivos/módulos envolvidos: [lista]
```

```
- Fluxo de execução resumido: [breve descrição]
```

```
- Pontos de melhoria identificados: [lista]
```

```
## Dependências
```

```
- Internas: [lista de módulos internos]
```

```
- Externas: [lista de bibliotecas/frameworks externos]
```


```
- Propósito principal de uma dependência chave: [breve explicação]
```



Utilize a IA para entender terminologias específicas relacionadas a sistemas de armazenamento vetorial e busca semântica.

Capítulo VII

Ex02 – Quebra estratégica de informações para uma busca ainda melhor

	Exercício : 02
Implementar um sistema de chunking para melhorar a granularidade e eficácia da busca semântica em documentos PDF, focando em currículos	
Pasta de entrega : <i>ex02/</i>	
Arquivos para entregar : <code>secure_resume_api.py</code>	

Chunking é o processo de dividir documentos longos em partes menores e mais gerenciáveis, chamadas de "chunks" ou fragmentos¹. No contexto de processamento de linguagem natural e sistemas de recuperação de informação, o *chunking* é uma etapa crucial por várias razões:

- **Gerenciamento de memória:** Documentos muito longos podem ser difíceis de processar de uma só vez devido a limitações de memória.
- **Granularidade na recuperação:** Permite recuperar trechos específicos e relevantes de um documento, em vez do documento inteiro.
- **Melhoria na assertividade:** Ajuda a focar a busca nas seções mais relevantes do texto.
- **Contextualização:** Mantém o contexto local dentro de cada trecho, o que é importante para entender o significado.

¹<https://www.pinecone.io/learn/chunking-strategies/>

Chunking simples vs. avançado

Chunking simples:

- Divide o texto em partes de tamanho fixo (por exemplo, a cada 1000 caracteres).
- Fácil de implementar, mas pode cortar frases ou parágrafos no meio.

Chunking avançado:

- Usa técnicas mais sofisticadas para dividir o texto de forma mais significativa.
- Pode considerar estruturas como parágrafos, frases ou até mesmo a semântica do conteúdo.

Como o chunking auxilia na busca semântica

A busca semântica visa entender a intenção e o contexto por trás de uma consulta, indo além da simples correspondência de palavras-chave. O chunking melhora a busca semântica de várias maneiras:

- **Relevância localizada:** Permite encontrar trechos específicos de um documento que são mais relevantes para uma consulta.
- **Contexto preservado:** Trechos bem definidos mantêm o contexto local, crucial para a compreensão do significado.
- **Eficiência:** Facilita a indexação e a busca em grandes volumes de texto.
- **Flexibilidade:** Permite ajustar o tamanho dos trechos para diferentes tipos de documentos ou necessidades de busca.

Desafios em realizar chunking

- **Tamanho do chunk:** Trechos muito pequenos podem perder o contexto, enquanto trechos muito grandes podem ser imprecisos.
- **Sobreposição:** Decidir quanta sobreposição ter entre trechos adjacentes para manter a continuidade.
- **Estrutura do documento:** Respeitar a estrutura natural do documento (parágrafos, seções) ao fazer a divisão em trechos.

VII.1 Exercício

Neste exercício, você implementará uma API segura usando Flask para processamento de currículos, utilizando chunking e incorporando medidas de segurança baseadas no OWASP API Security Top 10. Focaremos especificamente em:

- [API4:2023 - Unrestricted Resource Consumption](#)
- [API5:2023 - Broken Function Level Authorization](#)

As features implementadas devem levar em conta os cenários de teste dessas vulnerabilidades, conforme descritos pela OWASP, visando mitigar esses tipos de cenários na sua API.

Instruções

1. Implementar a função `chunk_text_recursive` usando `RecursiveCharacterTextSplitter`.
2. Criar uma API usando Flask com os seguintes endpoints:
 - `POST /upload_pdf`: para receber e processar um arquivo PDF de currículo
 - `GET /search`: para realizar buscas semânticas nos currículos processados
 - `DELETE /curriculum/<id>`: para excluir um currículo
3. Implementar um sistema de autenticação e autorização com três níveis de acesso:
 - Candidato: pode fazer upload de seu próprio currículo e buscar em currículos públicos
 - Recrutador: pode buscar em todos os currículos
 - Administrador: pode fazer tudo, incluindo excluir currículos
4. Implementar limites de taxa (rate limiting) para uploads de PDF e buscas, considerando os cenários de teste da OWASP para API4:2023.
5. Implementar controles de acesso baseados em função, levando em conta os cenários de teste da OWASP para API5:2023.
6. Integrar o armazenamento dos chunks processados no ChromaDB.
7. Implementar uma função de busca semântica que aproveite os chunks mais granulares.

Código base:

```
def chunk_text_recursive(text, chunk_size, chunk_overlap, separators):  
    pass  
  
def upload_pdf():  
    pass  
  
def search():  
    pass  
  
def delete_curriculum(id):  
    pass  
  
if __name__ == '__main__':  
    pass
```

Saída esperada:

```
POST /upload_pdf  
Response: {"message": "PDF processado com sucesso", "chunks_created": 5}  
  
GET /search?query=python  
Response: [  
  {  
    "document": "curriculo_12.pdf",  
    "chunk": 2,  
    "content": "Experiência profissional: Desenvolvedor Python Sênior na TechSolutions (2018-2022)..."  
  },  
  {  
    "document": "curriculo_35.pdf",  
    "chunk": 1,  
    "content": "Resumo Profissional: Engenheiro de Software com 6 anos de experiência em desenvolvimento Python..."  
  }  
]  
  
DELETE /curriculum/123 (com token de administrador)  
Response: {"message": "Currículo excluído com sucesso"}  
  
DELETE /curriculum/123 (com token de candidato)  
Response: {"error": "Acesso não autorizado"}  
  
POST /upload_pdf (excedendo o limite de taxa)  
Response: {"error": "Limite de taxa excedido. Tente novamente mais tarde."}
```




Considere implementar testes que simulem os cenários de ataque descritos pela OWASP para garantir que suas medidas de segurança sejam eficazes.



Ao implementar os controles de segurança, certifique-se de que eles não possam ser contornados por manipulação de parâmetros de solicitação ou cabeçalhos HTTP.

Capítulo VIII

Ex03 – RAG: Busca semântica e LLMs entram em um bar

	Exercício : 03
Criar uma aplicação web usando Streamlit que integre processamento de currículos, busca semântica e um modelo de linguagem para analisar e responder perguntas sobre as qualificações dos candidatos, utilizando <i>Retrieval-Augmented Generation</i> (RAG)	
Pasta de entrega : <i>ex03/</i>	
Arquivos para entregar : <code>resume_analyzer_app.py</code> , <code>project_explanation.md</code>	

Retrieval-Augmented Generation (RAG) é uma técnica poderosa que combina a precisão da busca semântica com a flexibilidade dos modelos de linguagem de grande escala (LLMs). Esta abordagem permite gerar respostas mais precisas e contextualizadas, especialmente útil em cenários como análise de currículos, onde informações específicas e detalhadas são cruciais.

Como RAG funciona

1. **Recuperação:** Utiliza busca semântica para encontrar informações relevantes em uma base de conhecimento (neste caso, currículos processados).
2. **Contextualização:** Fornece o contexto recuperado ao modelo de linguagem.
3. **Geração:** O LLM gera uma resposta baseada tanto na pergunta quanto no contexto fornecido.

VIII.1 Exercício

Instruções

1. Configurar o ambiente:
 - Configurar ChromaDB para armazenamento e busca de embeddings.
 - Preparar a integração com o Groq para utilizar o modelo llama3-8b-8192.
2. Implementar o processamento de currículos:
 - Utilizar a API de chunking do exercício anterior para processar os PDFs enviados.
 - Armazenar os chunks processados no ChromaDB.
3. Desenvolver a função de busca semântica utilizando ChromaDB.
4. Criar uma função para gerar respostas com o LLM via Groq, utilizando o contexto recuperado.
5. Construir a interface Streamlit com os seguintes componentes:
 - Opção para upload de múltiplos currículos.
 - Campo de entrada para perguntas sobre os candidatos.
 - Área para exibição das respostas geradas.
6. Gerar documentação automatizada do projeto usando Continue.dev:
 - Utilizar o Continue.dev para analisar o código do projeto e gerar uma documentação do tipo Explanation¹ em um único arquivo Markdown.
 - O arquivo de documentação (project_explanation.md) deve seguir este formato:

```
# Análise de currículos com RAG

## Visão geral do sistema
[Breve explicação do propósito e funcionamento geral do sistema]

## Componentes principais
1. [Nome do Componente 1]
   - [Breve explicação do componente e seu propósito]

2. [Nome do Componente 2]
```

¹<https://docs.divio.com/documentation-system/>

```
- [Breve explicação do componente e seu propósito]

...

## Conceitos principais
- [Conceito 1]: [Breve explicação]
- [Conceito 2]: [Breve explicação]
...

## Fluxo de funcionamento
[Explicação sucinta de como os componentes interagem]
```

- Revisar e ajustar a documentação gerada conforme necessário.

Código base:

```
import streamlit as st

# TODO: Adicione os imports necessários

# Interface Streamlit
st.title("Análise de currículos")

query = st.text_input("Faça uma pergunta sobre os candidatos:")
if query:
    # TODO: Implementar a lógica de busca e geração de resposta
    pass
```

Saída esperada:

Uma aplicação Streamlit funcional que permite o upload de currículos, realiza buscas semânticas e responde a perguntas sobre os candidatos usando um modelo linguístico.



Use `st.spinner()` para operações longas



Não deixe a sua chave de API no código submetido. Durante a avaliação, você precisa configurá-la adequadamente.

Capítulo IX

Entrega e Avaliação entre pares

IX.1 Processo de Entrega

- Submeta seu trabalho no repositório Git gerado na página principal do projeto.
- Certifique-se de que todos os arquivos necessários estejam incluídos e organizados conforme as instruções do projeto.
- Respeite o prazo de entrega estabelecido.

IX.2 Avaliação entre pares

- Seu projeto será avaliado por um dos seus colegas.
- A avaliação focará na qualidade do seu código e na aderência aos requisitos do projeto.
- Critérios de avaliação podem incluir:
 1. Funcionalidade: O código atende a todos os requisitos especificados?
 2. Legibilidade: O código é claro e bem estruturado?
 3. Eficiência: As soluções implementadas são otimizadas e seguem boas práticas?
 4. Organização: Os arquivos e estrutura do projeto estão bem organizados?
- Feedback detalhado é esperado, mas pode variar em extensão e detalhamento.

IX.3 Dicas para uma avaliação bem-sucedida

- Revise seu código antes da submissão final.

- Teste exaustivamente todas as funcionalidades implementadas.
- Se entender necessário para clareza, documente claramente qualquer decisão ou suposição feita.
- Esteja preparado para explicar suas escolhas de implementação.



A avaliação entre pares é uma oportunidade para aprendizado e crescimento pessoal e profissional. Esteja aberto ao feedback recebido e use-o para aprimorar suas habilidades.