Inteligência Artificial 2

Aula 27: Hackathon do Projeto Final (Backend e RAG)

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Inteligência Artificial 2

Sumário

Onde Paramos?

Ontem vocês montaram a **infraestrutura base** do Hackathon:

- Repositório GitLab configurado com `.gitignore` e `.env`.
- Esqueleto do projeto rodando localmente (Hello World).
- Issues/Kanban com a divisão de tarefas da equipe.

A Missão de Hoje

Construir o **cérebro**: ingestão de dados,
VectorStore populado e endpoint RAG respondendo via Postman.

Objetivos do Dia

Ao final desta sessão, a equipe deve ter:

1. **Scripts de ingestão** funcionais (PDF $\rightarrow$ chunks $\rightarrow$ VectorStore).
2. **Banco vetorial** populado com dados reais do domínio escolhido.
3. **Endpoint** `POST /api/chat` retornando respostas fundamentadas.
4. **System Prompt** refinado com persona, tom e guardrails.
5. **Validação** via Postman/Swagger (sem front-end ainda).

O Papel Crítico do Backend em Aplicações de IA

O *Backend* não é apenas um "repassador de requisições" para a API do ChatGPT. Ele é o verdadeiro **Cérebro Especialista** do seu produto.

- **Orquestração (Routing):** Decide se a pergunta do usuário exige uma busca no banco vetorial (RAG) ou se é apenas conversa trivial (Saudação).
- **Gestão de Contexto:** LLMs não têm memória. O *Backend* busca as últimas 5 mensagens da sessão no SQL e as injeta na requisição HTTP.
- **Segurança:** Garante que o Usuário A nunca acesse o contexto vetorial dos PDFs enviados pelo Usuário B (Isolamento *Multi-Tenant*).

Definição de Rotas API (Design RESTful)

Para o Hackathon, seu serviço base precisará expor no mínimo três rotas cruciais:

1. `POST /api/documents`: Rota de ingestão. Recebe um arquivo `.pdf` ou texto bruto, processa (Chunking), gera os Embeddings e insere no Banco Vetorial. Retorna `201 Created`.
2. `POST /api/chat`: A artéria principal. Recebe `{''query'': ''...''}`, realiza a similaridade semântica, injeta no prompt e retorna o texto completo. Retorna `200 OK`.
3. `GET /api/chat/{session_id}`: Retorna o histórico de mensagens formatadas para que o *Frontend* renderize a interface de balões de conversa.

Validando a Entrada (Pydantic / DTOs)

Não confie na string bruta. Tanto em Python (FastAPI) quanto em Java (Spring Boot), valide o JSON de entrada para barrar anomalias ou *Payloads* gigantes.

```
from pydantic import BaseModel, Field

class ChatRequest(BaseModel):
    session_id: str = Field(..., description=‘‘ID unico da conversa’’)
    query: str = Field(..., max_length=1000,
                       description=‘‘A pergunta do usuario’’)
    temperature: float = Field(default=0.7, ge=0.0, le=1.0)

# O FastAPI barra requisicoes sem session_id automaticamente
app.post(‘‘/api/chat’’)async def process_chat(req: ChatRequest):return ‘‘reply’’:
    orquestrar_ia(req.query, req.session_id)
```

O Desafio da Extração (Document Loaders)

O pipeline começa com a extração bruta. Ler texto puro (TXT, Markdown) é fácil, mas o mundo corporativo opera em PDFs escaneados e planilhas confusas.

- **PDFs Nativos vs Scans:** Um extrator comum falha em scans. É necessário integrar motores de OCR (Optical Character Recognition) como `Tesseract`.
- **O Abismo das Tabelas:** Ao extrair o texto "*linha a linha*", colunas de tabelas se fundem numa *string* sem sentido. O vetor resultante será inútil para o LLM raciocinar. Soluções empresariais utilizam LLMs multimodais para primeiro converter tabelas em Markdown ou JSON.

Se o documento contiver muitos gráficos ou diagramas vitais:

- A extração puramente textual via *PyPDF* os ignorará.
- **A solução Multimodal:** Converte-se a página inteira do PDF em uma imagem JPG. Envia-se para o `gpt-4o` (com capacidade de visão) com a instrução: *"Traduza tudo que vê nesta imagem para texto Markdown de alta fidelidade"*.
- Essa tática eleva absurdamente o custo do ETL, mas entrega RAG de altíssima qualidade corporativa.

A Matemática do Chunking

Modelos de Embedding possuem limite restrito de entrada (ex: 8192 tokens no `text-embedding-3-small` ou apenas 512 no `all-MiniLM-L6-v2` local). Se o seu documento tem 50.000 palavras, precisamos quebrar em blocos (*Chunks*). Mas quebrar onde?

- **Quebra por Caractere Fixo:** Quebra exatamente a cada 1000 letras. *Risco:* Cortar uma palavra ao meio ou separar o sujeito do predicado. O vetor resultante perderá a semântica da frase.
- **Quebra Recursiva (`RecursiveCharacterTextSplitter`):** Tenta quebrar em parágrafos (`\n\n`). Se o parágrafo for maior que o limite, tenta quebrar em frases (`.`). Se a frase for maior, quebra em vírgulas. **Preserva blocos lógicos**.

Sobreposição (Chunk Overlap)

Mesmo com a quebra lógica, a informação de uma frase pode depender fortemente da frase anterior, que ficou no *Chunk* passado. **A Solução: Overlap (Deslizamento)** O *Chunk 2* começa pegando as últimas 100 palavras do *Chunk 1*.

- *Chunk 1:* "O lucro líquido do Itaú no terceiro trimestre foi de"
- *Chunk 2 (Com overlap):* "no terceiro trimestre foi de 9 bilhões de reais, impulsionado por..."

Dessa forma, ambos os vetores conterão contexto parcial para atrair a busca por "lucro".

Chunking Semântico

A abordagem final é o **Chunking Semântico**:

- Em vez de quebrar por regras ortográficas, calculamos o Embedding de cada frase individualmente.
- Medimos a Distância de Cosseno entre a *Frase A* e a *Frase B*.
- Se forem semanticamente similares, elas formam um único bloco. Quando o assunto muda abruptamente (distância aumenta), nós dividimos o bloco ali.
- Isso garante *Chunks* perfeitamente coesos, mas custa muito mais processamento durante a Ingestão.

A Diferença entre Bancos: Local vs Cloud

As equipes do Hackathon devem decidir a infraestrutura vetorial:

Local (ChromaDB / FAISS)

- Roda na memória da aplicação ou salva em arquivos locais (`.sqlite3`).
- Excelente para protótipos rápidos e privacidade total.
- *Desvantagem:* Se sua API for replicada para lidar com tráfego pesado (Múltiplos *Workers*), os bancos locais se dessincronizam e corrompem arquivos.

Gerenciado (Pinecone / Qdrant Cloud)

- Servidor autônomo. Sua API viaja pela rede para consultá-lo via REST.
- Permite bilhões de vetores, escalabilidade horizontal e monitoramento real.
- *Desvantagem:* Introduz latência de rede adicional e depende de chaves de API externas.

Como o Banco Vetorial acha seu Texto? (HNSW)

Se temos 1 Milhão de vetores $v \in \mathbb{R}^{1536}$, calcular a similaridade de Cosseno da pergunta com *todos* eles (Busca Exata/Flat) levaria segundos ou minutos. A mágica está no índice **HNSW**

(Hierarchical Navigable Small World):

- Cria um "grafo social" multicamadas dos vetores.
- Na camada superior, há poucos vetores espaçados (ex: um vetor falando de Política, outro de Esportes).
- A pergunta (vetor) entra pela camada superior, vai para a vizinhança mais próxima (ex: Esportes).
- O algoritmo desce de camada (zoom in) buscando amigos mais próximos daquele ponto, até chegar na resposta exata. Busca **aproximada** logarítmica $\mathcal{O}(\log N)$, feita em milissegundos.

A Matemática das Camadas do HNSW

O HNSW baseia-se na teoria probabilística *Skip-List*.

- Cada nó (Vetor) inserido no banco sorteia uma camada aleatória ℓ. Camadas mais altas têm menos nós (distribuição exponencial negativa).
- As conexões horizontais na camada ℓ garantem que vizinhos de longo alcance sejam percorridos com saltos imensos em microssegundos (os *navigable small worlds*).
- A busca afunila verticalmente até a camada basal ($\ell = 0$), que contém todos os documentos, mas num raio já focado.

Similaridade: Cosseno vs Produto Interno

Ao configurar seu banco, você deverá escolher a métrica:

- **Distância Euclidiana (L2):** Mede a distância física ponta-a-ponta entre os vetores. Altamente sensível ao *tamanho/magnitude* do vetor.
- **Similaridade de Cosseno:** Mede o **ângulo** θ entre os vetores, ignorando o comprimento. É a métrica **padrão-ouro** para semântica de textos.

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

- **Produto Interno (Dot Product):** Mais barato computacionalmente. Se os modelos de Embedding normalizarem o vetor ($\|A\| = 1$), o Cosseno será matematicamente igual ao Produto Interno. Modelos modernos da OpenAI retornam vetores normalizados, permitindo buscas em frações de microssegundos.

Roteamento e Prompting Dinâmico

No seu *endpoint* /api/chat, após recuperar os top-K contextos via HNSW, você injeta o conhecimento em um *Prompt Template*.

```
prompt_template = ''''''Você é um analista jurídico restrito.
Utilize APENAS os trechos fornecidos em [CONTEXTO] para responder.
Caso a resposta não esteja nos trechos, diga: ''Não possuo esta
   informação.''

[CONTEXTO]
{contextos_agrupados}

[PERGUNTA DO USUÁRIO]
{query}
''''''

prompt_formatado = prompt_template.format(
    contextos_agrupados=''\n''.join([doc.page_content for doc in
       top_k_docs]),
```

Manutenção de Estado (Stateful vs Stateless)

A API Web é **Stateless** (não lembra de requisições anteriores). Para que o chat funcione, o estado deve ser persistido e injetado:

1. **Estratégia Client-Side:** O Frontend envia não apenas a nova pergunta, mas toda a lista de mensagens anteriores a cada `POST`. Gasta mais banda de rede e tokens.
2. **Estratégia Server-Side (Bancos SQL/Redis):** O Frontend envia a pergunta e o `SessionID`. O Backend faz um `SELECT * FROM messages WHERE session_id=X`. Empacota tudo em formato *ChatML* (System, User, Assistant, User) e envia ao LLM. Mantém o cliente leve.

O Problema do RAG Ingênuo (Naive RAG)

Ao desenvolverem hoje, a primeira barreira lógica será a busca de "continuação".

- *Usuário (Turno 1):* "O que o artigo 5 do regulamento diz sobre demissões?"
- A IA responde baseada nos vetores corretos.
- *Usuário (Turno 2):* "Mas e se eu fizer isso amanhã?"

Se você mandar "Mas e se eu fizer isso amanhã?" para o Banco de Vetores (ChromaDB), ele procurará documentos que falam "amanhã" e ignorará "demissões". A busca será destruída.

Reescrita de Consulta (Query Rewriting)

A solução em *Advanced RAG* para conversas longas:

- O Backend usa um LLM rápido (Llama 3 8B) com a instrução: *"Com base na conversa anterior e na nova pergunta curta, formule uma pergunta isolada completa."*
- A LLM transforma "Mas e se eu fizer isso amanhã?" em $\rightarrow$ *"Como o artigo 5 do regulamento de demissões lida com desligamentos realizados no dia seguinte?"*
- Esta string estendida é o que você passa para o Modelo de Embeddings e para a busca de Cosseno.

Tolerância a Falhas em Produção

A Internet cai. A API da OpenAI atinge o limite de taxa de requisições (*HTTP 429 Too Many Requests*) ou trava por sobrecarga global (*HTTP 502/503*). O seu Backend **nunca** pode

"craschar" internamente e deixar o Frontend carregando infinitamente.

- **Retentativa Exponencial (Exponential Backoff):** Usando bibliotecas como `Tenacity` (Python), o Backend deve tentar chamar o LLM novamente após 1s, depois 2s, depois 4s, antes de desistir.
- **Fallbacks:** Se a API da OpenAI falhar todas as tentativas, a aplicação (LangChain) pode chavear automaticamente a chamada para a API do Google Gemini, garantindo a entrega do texto ao usuário sem que ele perceba a queda.

Controle de Versão: O Caos do Hackathon

Trabalhar em equipe demanda organização rigorosa no Git.

Não desenvolvam isolados na branch `main`

- **Feature Branches:** O aluno focado no RAG cria a branch `feature/rag-engine`. O aluno do chat cria `feature/chat-route`.
- **Pull Requests (PR):** Ao terminar, peça ao colega para revisar o código antes de jogar na branch principal (*Merge*).
- O erro de um desenvolvedor *crashando* o servidor na branch principal derruba toda a equipe e atrasa o dia inteiro do Hackathon.

Testes Locais (Postman e Swagger)

Não integrem com o Frontend de imediato.

- Usem o painel **Swagger UI** que o FastAPI gera na rota `/docs`.
- Enviem o JSON de payload pelo Swagger e observem o terminal.
- Vejam se a resposta JSON demora o esperado, ou se a formatação (ex: aspas escapadas \") está pronta para o React engolir.

Laboratório: O Backend Especialista

Missões da Aula:

1. Criar os scripts isolados para Ler os dados do seu Domínio (PDFs, sites) e inseri-los no Banco Vetorial.
2. Validar as distâncias vetoriais com scripts de teste.
3. Configurar a estrutura do `FastAPI` (ou `Spring`) com as rotas reais baseadas no OpenAPI.
4. **Meta Ouro:** Finalizar o dia podendo mandar um `POST` via Postman ou Swagger e receber o texto gerado pelo LLM com contexto!

Chamem o Tech Lead para desatar os nós arquiteturais!

Dúvidas?

Email: alessio@cefetmg.br
Aula: Segunda e Quarta-Feira, 7h as 8:40h (Sala 39)
WhatsApp: Dúvidas rápidas

Lembrete

Confira sua Issue no GitLab!

Próxima aula: **Capstone Sprint 3**

100% da Aula 27 Concluída!