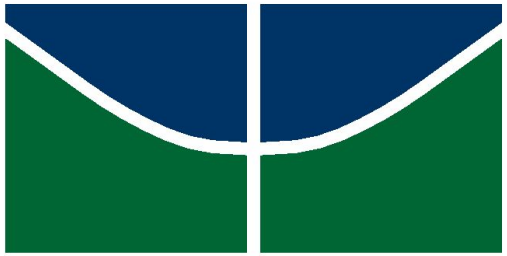
World Cup RAG Chatbot

Um Assistente Inteligente com RAG E Llama 3



Departamento de Ciência da Computação
Universidade de Brasília

Integrantes:

Ryan Reis

João Gomes

Marcus de Freitas

Victor Costa

Guilherme Matos

Albert Soares

Arthur Souza

Kaleb Pereira

Gabriel Castro

Tecnologia - Linguagens e Bibliotecas

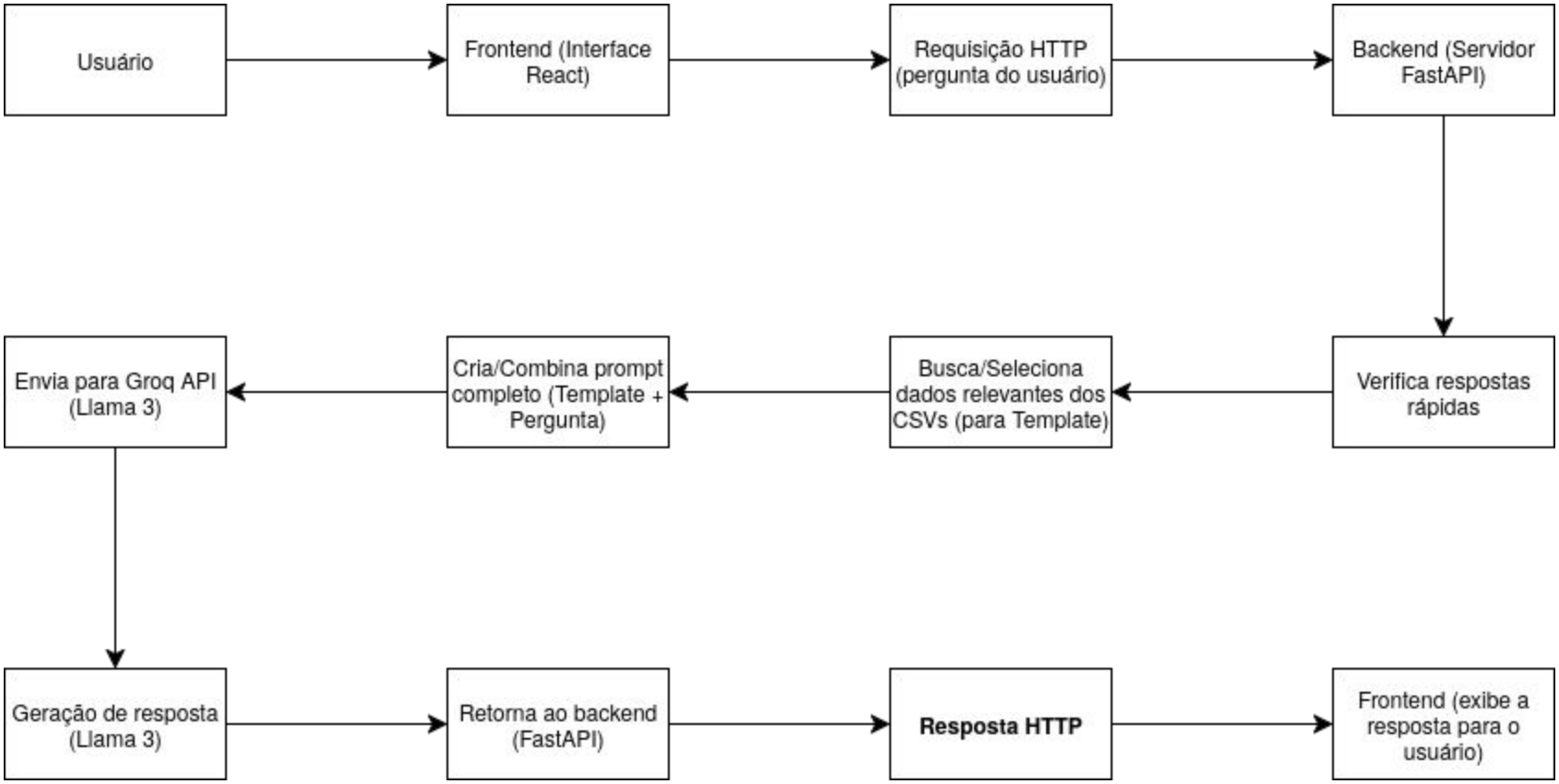
- Linguagens:
  - Python: Backend, LlamaIndex
  - JavaScript (React): Frontend
- Bibliotecas essenciais - Backend
  - FastAPI: Framework web assíncrono para API
  - Uvicorn: Servidor ASGI de alta performance
  - Pandas: Manipulação e análise de dados CSV
  - llama-index-llms-groq: Pacote para integração com LLMs da Groq
  - Pydantic: Validação de dados
  - python-dotenv: Gerenciamento de variáveis do ambiente

- Bibliotecas Essenciais - Frontend
  - Python: Backend, LlamaIndex
  - React, react-dom: Construção de interfaces de usuário
  - Vite, @vites/plugin-react: Ferramenta de build e suporte React
  - Tailwind CSS, Autoprefixe, PostCSS: Framework CSS utilitário
  - Lucide-react: Biblioteca de ícones
  - Tiptap: Editor de texto
  - ESLint: Ferramentas de linting

Plataforma e programas utilizados

- Servidor Backend: FastAPI servido por Uvicorn (Executado em ambiente Python)
- Frontend: Servido via Node.js/Vite (durante o desenvolvimento/localmente)
- Banco de Dados:
  - CSV: Arquivos com dados brutos da Copa do Mundo
  - Não utiliza banco de dados vetorial complexo na implementação atual, injeta dados diretamente no prompt)
- LLM (Modelo de Linguagem Grande): Llama 3 (acessado via API do Groq: llama3-70b-8192)

Funcionamento da Sistema



Dados: Aquisição e Tamanho

- Fonte: Dados históricos da Copa do Mundo FIFA (1930-2022)
- Conteúdo: Históricos de Copas, Resultados de Partidas, Estatísticas de Seleções e Jogadores e Ranking FIFA
- Formato: Arquivos CSV (e processados como strings para templates)
- Tamanho:
  - world_cup.csv: 22 edições (N) x 9 colunas (M)
  - matches_1930_2022.csv: Centenas de partidas (N) x Dezenas de colunas (M). Apenas 5 linhas processadas para o template inicial.

Pré-processamento e Complexidade

- Objetivo: Preparar os dados brutos para serem usados pelo LLM.
- O que será utilizado?
  - Pandas: para leitura, limpeza e mainpulação dos arquivos CSV.
  - LlamaIndex: Para carregar e estrutar os dados brutos de forma que possam ser processador na fase de indexação
- Etapas:
  - Carregamento de CSVs
  - Otimização/Seleão: processa world_cup.csv completamente e limita matches_1930_2022.csv a 5 linhas
  - Criação de Template de Texto
- Complexidade:
  - Leitura de CSVs: O (N x M) para cada arquivo (linhas x colunas)

Servidor e Formação da Resposta (Inferência)

- Objetivo: Processar a pergunta do usuário e gerar uma resposta.
- O que será utilizado:
  - FastAPI: Recebe a requisição HTTP.
  - LlamaIndex: Orquestra o fluxo RAG.
  - Groq (Llama 3): Gera a resposta final com base no contexto recuperado e na pergunta.
- Etapas
  - Recebe a pergunta do usuário
  - Verificação de Respostas rápidas: Busca em um dicionário por respostas pré-definidas (O(1))
  - Construção do Prompt Complexo: Concatena o template de dados pré-carregado com a pergunta
  - Chamada ao LLM
  - Recebe a resposta
  - Envia a resposta de volta ao frontend
- Complexidade
  - Construção de Prompt: $O(L_{template} + L_{question})$. Relativamente rápida, porque $L_{template}$ é fixo e $L_{question}$ pequena

Recursos Computacionais Disponíveis e Possibilidades

- Recursos:
  - Máquina local: [Notebook 16gb Ram, i5-1235 de 12ª geração]
  - Acesso à API Groq: Permite uso do Llama 3-70B sem hardware local.
- O que é possível fazer:
  - Rodar o backend e frontend localmente
  - Processar dados CSV e criar o template de texto localmente
  - Realizar inferência rápida do Llama 3 através do API Groq
- Limitação: Não é possível treinar/ajustar o Llama 3 localmente

Complexidade Dominante

O Llama 3 utiliza a arquitetura transformer [1], os LLMs que utilizam essa arquitetura tem sua complexidade dominada por $O(L^2 \cdot d)$, onde L é o comprimento da sequência e d é a dimensão do modelo (ou da chave/valor). Como d é uma constante para um modelo específico, a complexidade dominante é $O(L^2)$ [2].

Embora a complexidade algorítmica interna do modelo (para treinamento ou processamento batch em cenários específicos) seja $O(L^2)$, para uma inferência típica de geração, o tempo é linear em relação ao comprimento total da sequência gerada.

A operação mais custosa em termos de tempo de execução por interação do usuário é a chamada ao LLM no backend.

Mais especificamente, para o modelo `llama3-70b-8192` que utilizamos, o tempo de inferência é aproximadamente linear com o número total de tokens processados. Se `T_in` for o número de tokens de entrada e `T_out` for o número de tokens de saída, a complexidade é **O(T_in + T_out)**.

Fontes:

[1] LLAMA 3 Disponível em: `https://huggingface.co/meta-llama/Llama-3.1-8B` Acesso em: 05/07/2025.

[2] ASHISH, Vaswani. Attention is all you need. Advances in neural information processing systems, v. 30, p. I, 2017.

Tempo de execução por Fase

- **Inicialização:** O(Ncolunas total * Mlinhas max).
  - Como esta é uma operação de inicializacão única ela não afeta a complexidade por requisicão.
- **Backend:**
  - Complexidade Total por Requisicão: $O(T_{in} + T_{out})$
  - Complexidade Espacial: $O(N_{colunas} * M_{linhas} + L_{template})$
- **Frontend (desprezível):**
  - Complexidade Total por Requisicão: O(k), onde k = nº de mensagens no chat.
  - Complexidade Espacial: $O(K * L_{msg})$, onde L_{msg} = tamanho médio das mensagens no chat.

Tempo de "Treinamento" vs. Qualidade do Resultado

- Contexto RAG: Não há "treinamento" do Llama 3. O modelo já é pré-treinado e acessado via API.
- Impacto na Qualidade (para RAG):
  - Qualidade e Relevâcia dos Dados no Template: dados mais completos. limpos e relevantes resultam em respostas mais precisas
  - Tamanho do Template: Otimizar o tamanho do template para caber na janela de contexto LLM sem perder informações cruciais
  - Configuração do Prompt

Tempo de Inferência vs. Qualidade do Resultado

- Tempo de Inferência: Refere-se à velocidade com que o chatbot gera uma resposta.
- Impacto na Qualidade:
  - Direto: Um tempo de inferência muito longo deteriora a experiência do usuário (UX). Usuários esperam respostas rápidas.
  - Indireto: Não afeta diretamente a *precisão* da resposta do LLM, mas um gargalo na inferência pode levar a menos iterações de teste e otimização do sistema como um todo.
  - Nosso Caso (Groq): A baixa latência do Groq para o Llama 3 minimiza esse impacto, permitindo respostas quase instantâneas.

Obrigado!

FIM!