CENTRO UNIVERSITÁRIO DE BRASÍLIA – CEUB

FELIPE YOSHIDA

JOÃO PEDRO BORGES

JOSÉ MULLER

MATEUS BITAR

MATHEUS ALVES

RAFAEL MARTINS

NASCENTIA: ARQUITETURA DUAL PARA SISTEMAS DE RAG ESPECIALIZADOS EM SAÚDE MATERNA COM MULTI-MODELOS E ÍNDICES SEGREGADOS

BRASÍLIA

2025

FELIPE YOSHIDA

JOÃO PEDRO BORGES

JOSÉ MULLER

MATEUS BITAR

MATHEUS ALVES

RAFAEL MARTINS

NASCENTIA: ARQUITETURA DUAL PARA SISTEMAS DE RAG ESPECIALIZADOS EM SAÚDE MATERNA COM MULTI-MODELOS E ÍNDICES SEGREGADOS

Projeto Integrador apresentado como requisito parcial para conclusão do curso de Ciência de Dados e Machine Learning do Centro Universitário de Brasília – CEUB.

Orientador: Prof. Weslley Rodrigues

BRASÍLIA

2025

RESUMO

O presente projeto descreve o desenvolvimento do NascentIA, um sistema inteligente de perguntas e respostas focado no domínio de parto, pré-natal e pós-parto, utilizando a arquitetura de Geração Aumentada por Recuperação (RAG). O objetivo principal foi solucionar o problema de alucinações em Modelos de Linguagem de Grande Escala (LLMs) em contextos médicos sensíveis, garantindo rastreabilidade via citação de fontes documentais (PDFs). A inovação do projeto reside na implementação de uma arquitetura de software dual: uma versão de produção (chatbot.py), otimizada para estabilidade com o modelo GPT-4o-mini; e uma versão de desenvolvimento (chatbot (dev).py), projetada para experimentação técnica com múltiplos modelos open-source (famílias Qwen, Phi-3, Gemma e Mistral). A metodologia empregou estratégias de chunking adaptativas — 1500 caracteres para modelos robustos e 400 caracteres para modelos locais — e índices vetoriais FAISS segregados para evitar conflitos de dimensionalidade entre diferentes embeddings (text-embedding-3-small e multilingual-e5-large). Os resultados validaram a eficácia da versão de produção na interação com o usuário final, enquanto a versão de desenvolvimento demonstrou a viabilidade técnica de modelos locais em infraestrutura limitada.

Palavras-chave: Chatbot. RAG. Saúde Materna. GPT-4o. Modelos Open-Source. FAISS.

SUMÁRIO

Sumário

1 INTRODUÇÃO

O avanço dos Modelos de Linguagem de Grande Escala (LLMs) transformou a interação humano-computador, permitindo interfaces conversacionais fluidas. No entanto, em domínios de conhecimento sensíveis, como a saúde materna, a propensão desses modelos a gerar informações factualmente incorretas — fenômeno conhecido como "alucinação" — representa um risco crítico.

O Chatbot Nascentia surge como uma solução tecnológica para mitigar esse risco, aplicando a arquitetura RAG (Retrieval-Augmented Generation). O sistema restringe as respostas do modelo a um corpus documental validado (arquivos PDF sobre parto humanizado e pré-natal), garantindo que as orientações fornecidas sejam baseadas em evidências.

Diferentemente de abordagens genéricas, este projeto implementou um ecossistema de software dividido em dois ambientes: produção e desenvolvimento. Essa separação permite, simultaneamente, oferecer uma ferramenta estável para o usuário final e um laboratório de testes para a equipe técnica avaliar novos modelos de IA, como os recentes Small Language Models (SLMs).

1.1 OBJETIVOS

O objetivo geral é desenvolver e validar um sistema de perguntas e respostas especializado em saúde materna, capaz de operar em duas modalidades distintas (produção e desenvolvimento).

Os objetivos específicos são:

a) Implementar uma versão de produção robusta utilizando a API da OpenAI (GPT-4o-mini);

b) Criar um ambiente de desenvolvimento flexível para testes comparativos com modelos open-source (HuggingFace);

c) Desenvolver estratégias de indexação vetorial segregadas para suportar múltiplos modelos de embedding simultaneamente;

d) Avaliar o impacto do tamanho dos fragmentos de texto (chunks) na qualidade das respostas geradas por modelos de diferentes portes.

2 REFERENCIAL TEÓRICO

Esta seção apresenta o embasamento teórico necessário para a compreensão da arquitetura dual e do funcionamento do Chatbot Nascentia.

2.1 MODELOS DE LINGUAGEM: DA ESCALA MASSIVA AOS MODELOS EFICIENTES (SLMs)

O Processamento de Linguagem Natural (PLN) passou por uma mudança de paradigma com a introdução da arquitetura Transformer. Atualmente, o estado da arte divide-se em duas categorias principais utilizadas neste projeto:

Large Language Models (LLMs) Proprietários: Modelos como o GPT-4o-mini caracterizam-se pelo vasto número de parâmetros e treinamento em corpora fechados. Eles oferecem alta capacidade de generalização e seguimento de instruções complexas.

Open-Source e Small Language Models (SLMs): Recentemente, houve a emergência de modelos menores e abertos, focados em eficiência. O projeto explora famílias como Qwen2.5, Phi-3 e Mistral. Estes modelos, variando de 0.5B a 7B de parâmetros, democratizam o acesso à IA, permitindo execução local ou em infraestrutura controlada (YANG et al., 2024; JIANG et al., 2023).

2.2 GERAÇÃO AUMENTADA POR RECUPERAÇÃO (RAG)

A arquitetura RAG (Retrieval-Augmented Generation) conecta o modelo a uma base de conhecimento externa. O processo é dividido em: indexação (vetorização dos documentos), recuperação (busca semântica) e geração (injeção do contexto no prompt).

Neste trabalho, a teoria de RAG é expandida para uma abordagem modular, onde diferentes estratégias de chunking são aplicadas para otimizar a janela de contexto de cada arquitetura.

2.3 REPRESENTAÇÃO VETORIAL (EMBEDDINGS)

A eficácia do RAG depende da qualidade dos embeddings. O projeto adota uma estratégia híbrida:

Embeddings Proprietários: text-embedding-3-small da OpenAI.

Embeddings Open-Source (SOTA): intfloat/multilingual-e5-large-instruct. Este modelo pertence à família E5, que utiliza instruções de tarefa para melhorar a qualidade da representação semântica em múltiplos idiomas (WANG et al., 2024).

2.4 INDEXAÇÃO VETORIAL E ALGORITMOS DE BUSCA (FAISS)

Para a recuperação eficiente, utiliza-se o Facebook AI Similarity Search (FAISS). Neste projeto, a utilização do FAISS é crítica para a implementação de índices segregados. Devido à incompatibilidade dimensional entre os vetores gerados pela OpenAI e pelo modelo E5, a teoria de espaços vetoriais dita que estes devem ser armazenados em estruturas de dados distintas.

3 METODOLOGIA

Esta seção detalha a arquitetura de software e as estratégias de processamento adotadas.

3.1 ARQUITETURA DE VERSÕES E PROPÓSITOS

O sistema foi bifurcado em dois artefatos de execução principais:

a) Versão Final (chatbot.py)

Foco: Usabilidade e Produção.

Modelo: Fixo no OpenAI GPT-4o-mini.

Processamento: Varredura automática do diretório data/ para ingestão de PDFs.

Interface: Tema personalizado Nascentia, otimizado para clareza visual.

b) Versão de Desenvolvimento (chatbot (dev).py)

Foco: Teste A/B de modelos e depuração.

Seletor de Modelos: Permite alternar entre OpenAI e modelos hospedados no HuggingFace.

Ferramentas: Visualização detalhada dos chunks indexados e estatísticas do índice FAISS.

3.2 ESTRATÉGIAS DE VETORIZAÇÃO E CHUNKING

Um dos diferenciais técnicos deste projeto é a aplicação de parametrizações distintas de processamento de texto, conforme detalhado na Tabela 1.

Tabela 1 – Parâmetros de Processamento por Arquitetura

Fonte: Elaborado pelos autores (2025).

A redução do tamanho do chunk para 400 caracteres no pipeline open-source é estratégica para modelos menores (Small Language Models), que possuem janelas de atenção mais restritas e se beneficiam de contextos mais granulares.

3.3 ESTRUTURA DE DIRETÓRIOS E ORGANIZAÇÃO DO CÓDIGO

A refatoração do projeto resultou na seguinte estrutura organizacional, essencial para a manutenção dos índices segregados:

src/models/: Contém as classes lógicas (openai_chatbot.py, huggingface_chatbot.py) que herdam de uma classe base.

src/utils/: Módulos auxiliares, incluindo document_processor.py.

faiss_index/: Diretório particionado em subpastas (openai/ e huggingface/) para evitar conflitos de dimensão vetorial.

4 DESENVOLVIMENTO

4.1 FLUXO DE INGESTÃO E RECUPERAÇÃO

O sistema opera sob a lógica de Lazy Loading para os índices vetoriais. Ao iniciar, o sistema verifica a existência de índices em faiss_index/{modelo_ativo}. Se o índice existe, é carregado na RAM. Caso contrário, o processador de documentos é acionado para ler os PDFs da pasta data/, segmentar o texto e criar os vetores.

4.2 INTERFACE E EXPERIÊNCIA DO USUÁRIO

A interface foi desenvolvida em Streamlit. Ambas as versões apresentam:

Chat Interativo: Histórico de conversa mantido durante a sessão.

Citações Automáticas: Ao final de cada resposta, o sistema lista as fontes utilizadas (nome do arquivo PDF), garantindo a confiabilidade da informação.

Visualização de Chunks (Versão Dev): Permite aos desenvolvedores inspecionar exatamente quais trechos de texto foram recuperados pelo algoritmo de busca, facilitando o ajuste fino dos parâmetros de chunking.

5 RESULTADOS

A implementação da arquitetura dual permitiu atingir resultados significativos em duas frentes:

Experiência de Uso (Versão Final): O uso do gpt-4o-mini com chunks maiores (1500 caracteres) provou-se ideal para a síntese de informações complexas sobre parto, oferecendo respostas fluídas e bem fundamentadas com latência reduzida.

Capacidade Técnica (Versão Dev): A capacidade de alternar para modelos como Qwen2.5-7B ou Mistral-7B demonstrou que é possível executar soluções de RAG com custos de API reduzidos. A separação física dos índices FAISS resolveu conflitos de dimensionalidade vetorial que ocorriam nas versões anteriores do projeto, estabilizando a troca de contextos.

6 CONCLUSÃO

O projeto NascentIA evoluiu de uma prova de conceito simples para uma aplicação estruturada em camadas de desenvolvimento e produção. A adoção de uma estratégia híbrida — utilizando a robustez da OpenAI para o usuário final e a flexibilidade do HuggingFace para pesquisa — validou a viabilidade técnica do uso de RAG em domínios sensíveis como a saúde materna.

A implementação de índices segregados e estratégias de chunking diferenciadas demonstrou maturidade na engenharia de software aplicada à Ciência de Dados, garantindo que o sistema seja escalável e adaptável a novos modelos de linguagem que venham a surgir no futuro.

REFERÊNCIAS

JIANG, Albert Q. et al. Mistral 7B. arXiv preprint arXiv:2310.06825, 2023.

WANG, Liang et al. Multilingual E5 Text Embeddings: A Technical Report. arXiv preprint arXiv:2402.05672, 2024.

YANG, An et al. Qwen2.5 Technical Report. arXiv preprint arXiv:2412.15115, 2024.