VIII Simpósio de Iniciação Científica 2025

"Do conhecimento acadêmico à transformação sustentável: inovação com validação científica"

Assistente de estudos com IA

Guilherme Campos Silva[1] (EG), João Gabriel Miguêz da Silva[1] (EG), Júlia Pivato de Oliveira[1] (EG), Maria Fernanda Guimarães Soares[1] (EG), Egon Luiz Müller Júnior (PQ)[1]

[1]*Universidade Federal de Itajubá,*

Palavras-chave: Assistente de estudos; geração aumentada por recuperação; embeddings semânticos; indexação vetorial.

Introdução

O crescente volume de materiais de estudo em formato digital, como artigos e livros, impulsiona o desenvolvimento de ferramentas inteligentes que atuam como facilitadores no processo de aprendizado. Neste cenário, embora o acesso ao conhecimento seja amplo, a tarefa de localizar respostas para questões complexas, que exigem compreensão contextual, ainda pode ser demorada quando se utilizam métodos de busca tradicionais por palavra-chave. Visando otimizar essa busca, este trabalho tem como objetivo o desenvolvimento de um Assistente de Estudos baseado em Inteligência Artificial, projetado para fornecer respostas precisas e contextualizadas, fundamentadas exclusivamente em uma base de documentos fornecida pelo usuário. A justificativa para sua criação reside na necessidade de ferramentas que não apenas agilizem o estudo, mas também garantam a fidelidade das informações, mitigando o risco de desinformação ou "alucinações" por parte dos modelos de linguagem.

A metodologia empregada é o modelo de Geração Aumentada por Recuperação (Retrieval-Augmented Generation - RAG), sendo um modelo que vem ganhando destaque em problemas como resolução de questões (Karpukhin et al., 2020; Ram et al., 2021), modelagem de linguagens (Guu et al., 2018), outra vantagem é a redução de alucinações(Gao et al., 2023), muito comuns em modelos de linguagens generalistas como a GPT e Gemini (Zhang, 2025). O procedimento é dividido em duas etapas principais. Primeiramente, em uma fase de pré-processamento, os documentos em formato PDF são extraídos, segmentados em trechos menores (*chunks*), e cada trecho é convertido em um vetor numérico (*embedding*) através do modelo SentenceTransformer. Estes vetores são então armazenados em um índice FAISS, que permite buscas de similaridade semântica de alta eficiência. Posteriormente, na fase de interação, a pergunta do usuário é igualmente convertida em um vetor. O sistema utiliza o índice FAISS para recuperar os trechos de texto mais relevantes da base de conhecimento. Estes trechos, juntamente com a pergunta original, são submetidos como contexto a um Modelo de Linguagem Amplo (LLM), o Mistral, que gera uma resposta final coesa e estritamente fundamentada nos documentos originais. A interface do sistema foi desenvolvida com a biblioteca Streamlit para garantir uma experiência de uso interativa e intuitiva.

Metodologia

O desenvolvimento do Assistente de Estudos foi estruturado com base na arquitetura de Geração Aumentada por Recuperação (RAG), dividindo-se em duas etapas sequenciais: uma de pré-processamento e indexação, executada offline, e outra de recuperação e geração de resposta, que ocorre em tempo real durante a interação com o usuário. A organização dos métodos seguiu a ordem em que foram aplicados.

A primeira etapa consiste na construção da base de conhecimento. Inicialmente, o conteúdo textual é extraído de todos os documentos em formato PDF disponíveis, utilizando a biblioteca PyMuPDF (fitz). O texto contínuo extraído é então segmentado em trechos menores e de tamanho fixo, um processo essencial para a recuperação contextual. Na sequência, cada chunk de texto é submetido ao modelo de linguagem SentenceTransformer ('all-MiniLM-L6-v2') para ser convertido em um vetor numérico de alta dimensão (embedding). Por fim, esses vetores são organizados e armazenados em um índice do tipo IndexFlatL2 da biblioteca FAISS, que otimiza a busca por similaridade semântica. O índice gerado e os chunks de texto correspondentes são salvos em disco para serem consumidos pela aplicação.

A segunda etapa é iniciada quando o usuário submete uma pergunta através da interface gráfica, desenvolvida com o framework Streamlit. A pergunta é processada pelo mesmo modelo SentenceTransformer para gerar seu próprio vetor de embedding. Este vetor é então utilizado para consultar o índice FAISS, que

retorna os k trechos de texto mais semanticamente relevantes ao questionamento. Estes trechos recuperados (o contexto) são concatenados a um prompt estruturado, juntamente com a pergunta original, e enviados ao Modelo de Linguagem Amplo (LLM) Mistral, servido localmente via Ollama. O LLM é instruído a formular uma resposta coesa e precisa, utilizando exclusivamente o contexto fornecido, o que garante a fundamentação da resposta no material de estudo. A resposta final e suas fontes (os trechos utilizados) são então exibidas ao usuário na interface.

A interface em Streamlit operacionaliza o fluxo metodológico de ponta a ponta: o usuário insere a pergunta em um campo de entrada e aciona a busca; o aplicativo embebe a consulta, consulta o índice FAISS e monta o prompt com os trechos recuperados; em seguida, exibe a resposta do LLM acompanhada das fontes utilizadas. O layout organiza, em seções claras, (i) o campo de consulta, (ii) a resposta gerada e (iii) as citações/trechos recuperados, permitindo verificar rapidamente a origem de cada informação. Elementos de estado e mensagens de erro tratadas na própria página mantêm a experiência fluida, enquanto a inicialização dos recursos no startup do app evita recargas desnecessárias em interações subsequentes.

Figura 1 – Interface para as perguntar ao BOT

Figura 2 – Fontes utilizada pelo BOT para responder

Resultados e discussão

O desenvolvimento deste trabalho resultou em um software funcional que, com uma interface interativa, permite que o usuário submeta perguntas e receba respostas exibindo as fontes usadas, além de exibir detalhes como a quantidade de chunks e arquivos *.pdf* usados como referência.

A fase de pré-processamento e indexação que consiste das etapas de extração do conteúdo textual do documento, a divisão em chunks, o processo de *embedding* e por fim organização e armazenamento dos vetores resultantes demonstraram a eficácia das bibliotecas PyMuPDF e FAISS e do modelo de linguagem SentenceTransformer.

Quanto à fase de processamento, a conversão da pergunta do usuário em um vetor de *embedding* permitiu que o índice FAISS recuperasse, com sucesso, os trechos mais relevantes do material de estudo condizentes com a pergunta.

Embora o sistema tenha apresentado resultados promissores, possui limitações. A aplicação se restringe a documentos que possuem texto extraível não sendo aplicável a documentos escaneados nem a imagens. Além disso, a divisão em chunks de tamanho fixo pode, em alguns casos, segmentar sentenças ou ideias, gerando inconsistências.

Conclusões

O trabalho confirmou a viabilidade de um

assistente de estudos baseado em RAG capaz de entregar respostas coerentes, rastreáveis e alinhadas ao material-fonte. A solução construída — extração com PyMuPDF, segmentação em chunks, embeddings com SentenceTransformer (all-MiniLM-L6-v2), indexação FAISS (IndexFlatL2), geração com o LLM Mistral via Ollama e interface em Streamlit — demonstrou boa usabilidade: recupera trechos semanticamente relevantes, reduz o tempo de busca e mitiga "alucinações" ao sempre referenciar as fontes. Ao mesmo tempo, evidenciou limites práticos: depende de PDFs com texto extraível, pode fragmentar ideias por usar chunks fixos e seu desempenho está condicionado à cobertura e qualidade do acervo.

Para evoluir o sistema, recomendamos incorporar OCR e análise de layout para documentos digitalizados; adotar chunking adaptativo com sobreposição e um reranqueador (ex.: cross-encoder) para aumentar a precisão; incluir verificações de fidelidade (checar citação/trecho antes de responder) e métricas de avaliação (fidelidade, cobertura de citações, tempo de resposta); implementar cache e atualização incremental do índice com controles de acesso/privacidade; e ampliar o suporte a figuras, tabelas e múltiplos idiomas.

Agradecimentos

Os autores agradecem à Universidade Federal de Itajubá (Unifei) e ao Fundo Nacional de Desenvolvimento da Educação (FNDE) pelo apoio concedido para a realização deste trabalho.

Referências

LEWIS, Patrick; PEREZ, Ethan; PIKTUS, Aleksandra et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In: Advances in Neural Information Processing Systems, 2020.

DOUZE, Matthijs; GUZHVA, Alexandr; DENG, Chengqi et al. The Faiss library, 2024.

JOHNSON, Jeff; DOUZE, Matthijs; JÉGOU, Hervé. Billion-scale similarity search with GPUs, 2017.

REIMERS, Nils; GUREVYCH, Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019.

PYMUPDF. PyMuPDF: documentação. Read the Docs, 2025.

STREAMLIT. Streamlit documentation. [S. l.]: Streamlit, s.d.

ZHANG, Yue et al. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. Computational Linguistics, p. 1-46, 2025.

KARPUKHIN, Vladimir et al. Dense Passage Retrieval for Open-Domain Question Answering. In: EMNLP (1), 2020. p. 6769-6781.

GUU, Kelvin et al. Generating sentences by editing prototypes. Transactions of the Association for Computational Linguistics, v. 6, p. 437-450, 2018.

GAO, Yunfan et al. Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997, v. 2, n. 1, 2023.