XLVII International Sodebras Congress
04 a 06 de Setembro de 2025
Campinas - SP

INTELIGÊNCIA ARTIFICIAL APLICADA À GESTÃO DO CONHECIMENTO: CONSTRUÇÃO DE CHATBOTS COM LLMS E RAG

ARTIFICIAL INTELLIGENCE APPLIED TO KNOWLEDGE MANAGEMENT: BUILDING CHATBOTS WITH LLMS AND RAG

Daniel Schmidt[1]
Rodrigo Pruenca de Souza[2]
Miguel Afonso Sellitto[3]
Leandro Gauss[4]

Resumo – Este artigo apresenta o desenvolvimento de um software de código aberto voltado à criação de chatbots de Perguntas e Respostas (QA) com base em modelos de linguagem de larga escala (LLMs) e no método de Recuperação Aumentada (RAG). A solução permite a ingestão de documentos em diversos formatos, a conversão dos conteúdos em vetores semânticos com modelos de embedding, e a recuperação de trechos relevantes a partir de consultas em linguagem natural. Utilizando a biblioteca LangChain, armazenamento vetorial com ChromaDB, interface com Streamlit e ajuste automatizado com Optuna, o sistema permite a personalização da cadeia de inferência e avaliação com base em um conjunto de perguntas e respostas esperadas. A aplicação foi testada em ambiente local e demonstrou resultados satisfatórios na geração de respostas técnicas fundamentadas. A arquitetura modular e extensível permite sua aplicação em contextos organizacionais diversos, com potencial para gestão do conhecimento, suporte técnico e automação de consultas documentais.

Palavras-chave: Inteligência artificial. LLM. Recuperação aumentada.

Abstract - This paper presents the development of an open-source software designed for building Question Answering (QA) chatbots based on Large Language Models (LLMs) and the Retrieval-Augmented Generation (RAG) method. The solution supports document ingestion in multiple formats, semantic vectorization using embedding models, and retrieval of relevant

[1] Doutorando em Engenharia de Produção e Sistemas pela UNISINOS, Mestre em Engenharia de Produção e Sistemas pela UNISINOS e Bacharel em Ciência da Computação. Contato: daniel.panambi@gmail.com.
[2] Mestre em Engenharia Mecânica pela UFRGS e Bacharel em Engenharia Mecânica pela UFRGS. Contato: rodrigo.souza@tmsa.ind.br.
[3] Professor e pesquisador do PPG em Engenharia de Produção e Sistemas. Possui graduação em Engenharia Eletrônica, com Mestrado e Doutorado em Engenharia de Produção, membro do corpo docente internacional da UNIMORE, Itália. Contato: sellitto@unisinos.br.
[4] Doutor, mestre e especialista em Engenharia de Produção e Sistemas, com graduação em Engenharia Mecânica e Design de Produtos. É pesquisador do GMAP, professor e coordenador do curso de Engenharia de Produção na UNISINOS. Contato: lgauss@unisinos.br.

content from natural language queries. Built with LangChain, ChromaDB, Streamlit, and automated tuning via Optuna, the system enables the customization of the inference pipeline and evaluation using a predefined QA test set. The application was tested in a local environment and showed promising results in generating technically grounded answers. Its modular and extensible architecture supports deployment in various organizational contexts, with potential applications in knowledge management, technical support, and document-driven query automation.

Keywords: Artificial intelligence. LLM. Retrieval-augmented generation.

I. INTRODUÇÃO

O avanço da inteligência artificial e do Processamento de Linguagem Natural (PLN) tem impulsionado o desenvolvimento de chatbots inteligentes. Esses sistemas de Perguntas e Respostas (Question Answering – QA) são capazes de gerar respostas em linguagem natural, a partir de um conjunto de documentos ou mesmo sem contexto, como nas tarefas de closed-book QA. Historicamente, os primeiros chatbots utilizavam regras fixas, como o ELIZA, desenvolvido por Weizenbaum em 1966 (Weizenbaum, 1966), que operava com scripts estáticos. Posteriormente, surgiram modelos baseados em recuperação, como o ALICE, criado por Wallace em 2007, que selecionavam respostas a partir de conjuntos pré-definidos (Wallace, 2007).

Com a introdução dos modelos Sequence-to-Sequence (Seq2Seq), propostos por Sutskever, Vinyals e Le (2014), e do modelo Transformer, por Vaswani et al. (2017), tornou-se possível criar sistemas generativos capazes de produzir respostas inéditas e contextualmente apropriadas. Esses avanços deram origem aos Modelos de Linguagem de Larga Escala (LLMs), como GPT-3.5, GPT-4, LLaMA-2 e Mistral, treinados com grandes volumes de texto usando a arquitetura Transformer. LLMs exigem conjuntos de dados de alta qualidade para contextualizar as entradas e gerar respostas eficazes (Birhane et al., 2023). Neste projeto, os LLMs são utilizados como núcleo de um sistema QA, aprimorado pelo método de Retrieval-Augmented Generation (RAG), que combina conhecimento pré-treinado com informações recuperadas dinamicamente de documentos técnicos organizacionais. Essa abordagem gera respostas mais precisas e aderentes ao domínio aplicado.

A área de QA é um campo interdisciplinar que integra Recuperação da Informação (IR), Extração de Informação e PLN. Com o uso crescente de aprendizado profundo (DL), o PLN tornou-se essencial para a interação entre humanos e sistemas computacionais, permitindo que agentes compreendam nuances semânticas complexas (Otter et al., 2021). Os sistemas QA, nesse contexto, tornam-se instrumentos centrais na Interação Humano-Agente (HAI), promovendo comunicações naturais e contextualizadas. Estudos apontam que o uso de PLN com DL aumenta a aceitação e a eficácia desses sistemas, especialmente quando alinhados a princípios centrados no usuário (Kopp et al., 2021).

Neste projeto, os sistemas QA são concebidos como mecanismos fundamentais para facilitar a interação entre humanos e agentes digitais, apoiando a construção de interfaces orientadas ao conhecimento técnico especializado. Estudos recentes confirmam que modelos baseados em redes neurais são eficazes em QA, principalmente com a abordagem sequence-to-sequence (Yin et al., 2016; Su et al., 2018; Vinyals e Le, 2015). A Figura 1 ilustra a estrutura do método RAG aplicado ao desenvolvimento do chatbot, incluindo etapas como ingestão de dados, vetorização, recuperação de documentos e inferência com LLMs. Embora os modelos possam ser ajustados para tarefas específicas, muitos LLMs são capazes de generalizar sem ajustes adicionais.

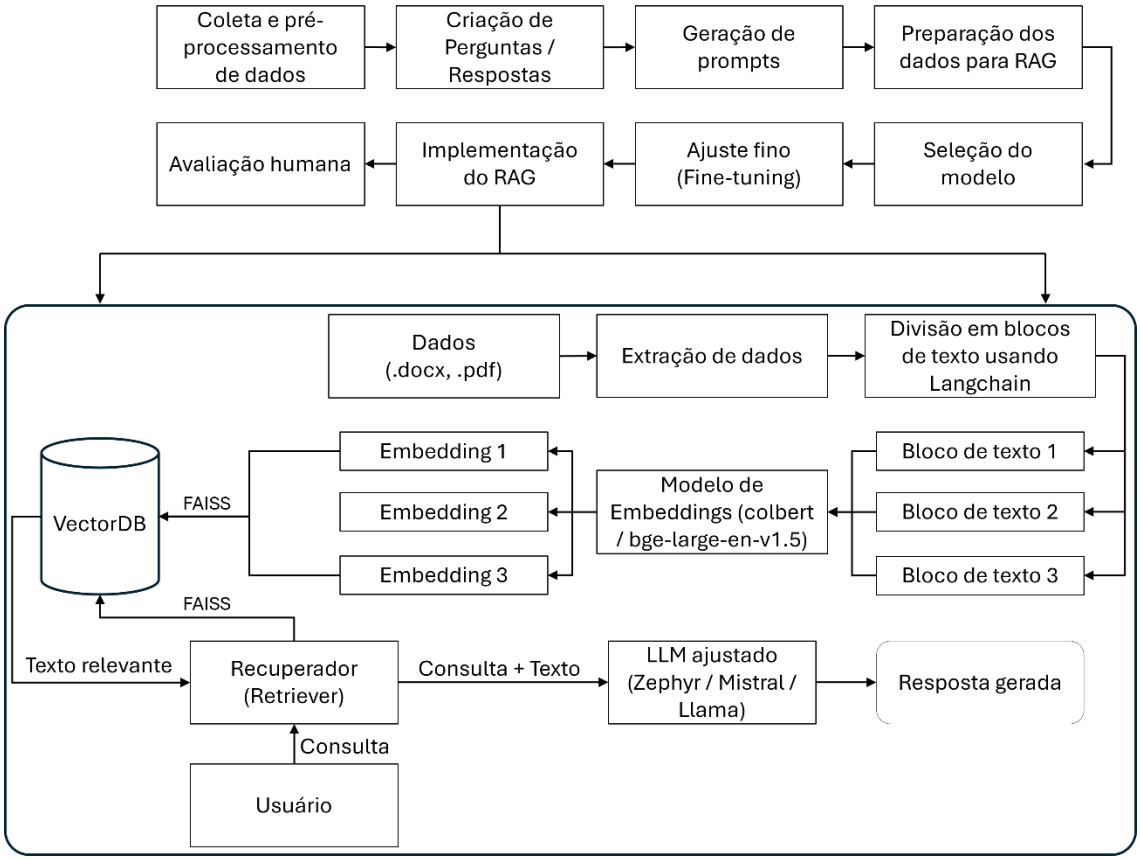

Figura 1. Pipeline de desenvolvimento do chatbot LLM QA. Salim et al., 2025.

A construção de sistemas QA com LLMs envolve etapas como coleta e curadoria de dados, ajuste fino, testes, inferência e implantação. No entanto, como observado por Salim et al. (2025), ainda não há softwares gratuitos e de código aberto que ofereçam suporte acessível a usuários técnicos e não técnicos. O ajuste fino e a estruturação da base de conhecimento demandam esforços significativos em pré-processamento.

Além disso, as métricas atuais de avaliação nem sempre são eficazes para medir o desempenho de LLMs. Nesse cenário, a avaliação conduzida por humanos é considerada o método mais confiável (Stureborg et al., 2024). A adoção de sistemas QA com LLMs pode transformar a gestão do conhecimento nas organizações, automatizando o acesso a documentos institucionais como normas técnicas, editais e procedimentos operacionais. Essa automação facilita consultas, acelera a aprendizagem interna e apoia a tomada de decisão com base em conhecimento estruturado.

Este projeto propõe uma aplicação que permite a criação de chatbots QA com LLMs sem necessidade de programação pelo usuário. A plataforma, baseada em código aberto, integra módulos para ingestão de dados, vetorização com ChromaDB, recuperação via RAG e ajuste automático de parâmetros com base em um conjunto de QA de referência. O sistema aceita múltiplos formatos de documentos, permite controle dinâmico da inferência e visualização de fontes, demonstrando sua viabilidade técnica para diversos domínios organizacionais.

II. DESCRIÇÃO DO SOFTWARE

A descrição geral do software é apresentada na Seção 2.1. Em seguida, a Seção 2.2 detalha a implementação de cada funcionalidade.

2.1 Arquitetura do Software

O software desenvolvido contempla seis funcionalidades principais: ingestão de dados, ajuste fino de parâmetros, geração de dados de teste, customização do processo RAG, avaliação automatizada, inferência e implantação. Essas etapas são descritas na Seção 2.2. O sistema foi implementado com diferentes modelos de linguagem de código aberto, modelos de embedding e bibliotecas Python, buscando equilíbrio entre qualidade de resposta e desempenho computacional.

Foram incluídas funcionalidades para ingestão de documentos fornecidos pelos usuários, com indexação automática via mecanismo RAG. O sistema suporta modelos recentes e robustos, como Mistral, Zephyr e Llama-3 (7 a 8 bilhões de parâmetros), além de modelos leves, como Phi-3 e Flan-T5, apropriados para dispositivos com menor capacidade. Também foi incorporada a opção de ajuste fino dos parâmetros de embedding.

A aplicação permite a personalização dos parâmetros de inferência, viabilizando a adaptação a futuras versões de LLMs. Um módulo de avaliação automatizada compara as respostas geradas com um conjunto de referência, permitindo medir o desempenho segundo o modelo utilizado, a segmentação dos dados (chunking) e a estratégia de embedding. A funcionalidade de implantação permite a execução do modelo treinado em ambiente local.

A Figura 2 ilustra a arquitetura modular da solução. O processo tem início com a ingestão de dados, realizada por meio de upload de arquivos ou inserção manual de texto. Os conteúdos são segmentados e vetorizados com o uso de modelos de embedding; neste projeto, os vetores gerados apresentam dimensão de 1.536, correspondente ao modelo OpenAIEmbeddings adotado. Esses vetores são então armazenados em um repositório vetorial implementado com a biblioteca ChromaDB. Na sequência, a cadeia de QA é construída com o ConversationalRetrievalChain, utilizando o modelo GPT-3.5-turbo com prompt técnico estruturado. A interface permite consultas, visualização das respostas e fontes utilizadas, além do ajuste de parâmetros. Um módulo adicional realiza a avaliação automática com dados de teste e otimização via Optuna. A interface, construída com Streamlit, permite execução local e futura adaptação para ambientes de produção.

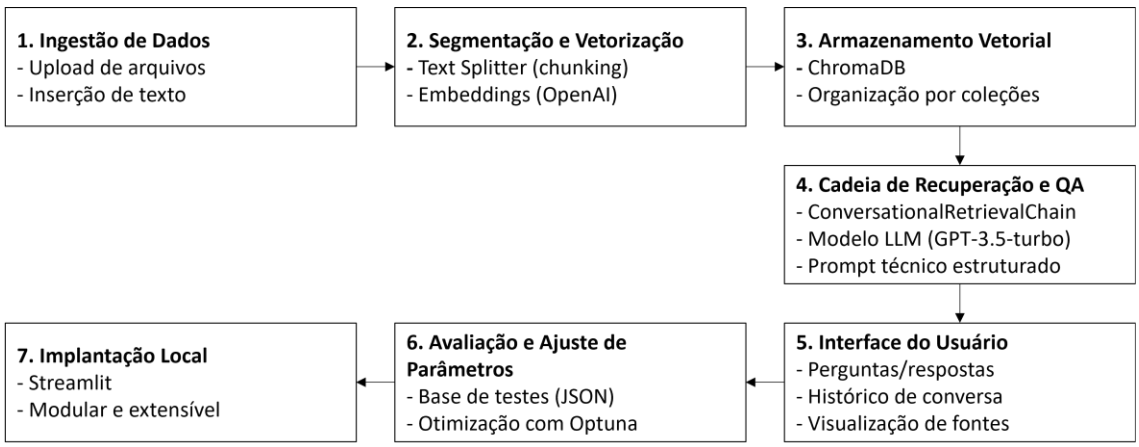

Figura 2. Arquitetura do Software para Desenvolvimento de Chatbots QA com LLMs

2.2 Funcionalidades do Software

O software desenvolvido apresenta um conjunto de funcionalidades integradas que viabilizam o desenvolvimento, ajuste, avaliação e implantação de chatbots baseados em LLMs para sistemas QA. A figura 3 exibe a interface do sistema e suas funcionalidades.

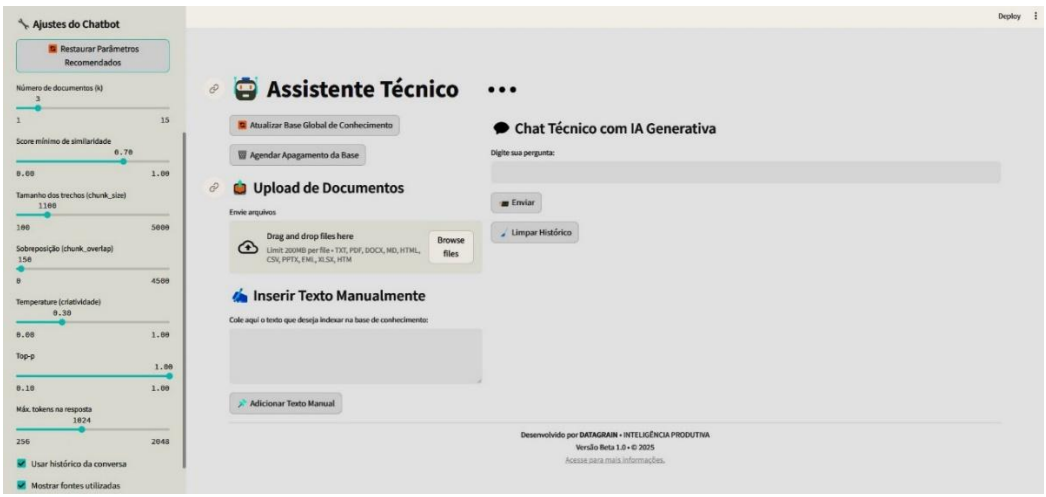

Figura 3. Interace do Software

A aplicação realiza a ingestão de documentos em formatos como .txt, .pdf, .docx, .md, .html, .csv, .pptx, .eml e .xlsx, por meio de upload na interface ou inserção manual de texto. Os arquivos são processados com segmentação de conteúdo (chunking) utilizando o algoritmo RecursiveCharacterTextSplitter, com parâmetros ajustáveis como tamanho e sobreposição dos trechos. Os segmentos são vetorizados com OpenAIEmbeddings e armazenados localmente por meio da biblioteca ChromaDB.

O sistema permite o ajuste dinâmico de hiperparâmetros que impactam o desempenho do QA, incluindo: número de documentos recuperados (k), limiar de similaridade (score_threshold), tamanho e sobreposição dos trechos (chunk_size e chunk_overlap), temperatura, top-p e limite de tokens. Esses parâmetros podem ser definidos manualmente ou restaurados a uma configuração recomendada com base na avaliação automatizada. Alterações nos parâmetros reinicializam a cadeia de QA, garantindo consistência. Novos documentos adicionados são automaticamente integrados à base vetorial existente.

A cadeia de inferência consiste na recuperação de trechos com maior similaridade vetorial, seguida pela geração de respostas condicionada a um prompt técnico estruturado, utilizando o modelo GPT-3.5-turbo (OpenAI). O sistema permite manter o histórico de conversas, viabilizando interações com contexto. Cada resposta pode vir acompanhada, de forma opcional, da fonte documental mais relevante, com nome do arquivo e trecho correspondente, promovendo transparência e explicabilidade.

Adicionalmente, foi implementado um módulo complementar (tuning_chatbot_original.py) que avalia o desempenho do sistema com base em um conjunto de perguntas e respostas de referência. O Optuna é utilizado para automatizar a busca por parâmetros ideais de chunking e recuperação, conforme a acurácia frente ao gabarito. Essa abordagem permite calibrar o sistema segundo o tipo de conteúdo indexado. O chatbot pode ser executado localmente via interface em Streamlit. A arquitetura modular e extensível permite integração com servidores, APIs e futuras versões com modelos quantizados ou alternativos de código aberto.

III. METODOLOGIA

A avaliação do sistema foi conduzida para verificar sua eficácia na geração de respostas técnicas a partir de documentos previamente indexados. A metodologia incluiu preparação da base de conhecimento, definição do conjunto de testes, execução da cadeia QA e análise dos resultados.

Documentos técnicos representativos de um contexto industrial real foram selecionados, contendo informações sobre escopo de fornecimento, especificações de equipamentos, normas técnicas e orientações de montagem. Esses arquivos foram processados pelo módulo de ingestão, segmentados com o algoritmo RecursiveCharacterTextSplitter e vetorizados com embeddings da OpenAI. As representações foram armazenadas localmente via ChromaDB.

Para avaliar a acurácia, foi construído um conjunto de perguntas e respostas de referência no formato JSON (perguntas_respostas.json), contemplando temas como materiais exigidos, número de itens, padrões de pintura e critérios técnicos. As respostas esperadas foram formuladas com base no conteúdo explícito dos documentos. A fim de otimizar a precisão, foi realizada uma busca automatizada de hiperparâmetros com a biblioteca Optuna, testando diferentes valores para chunk_size, chunk_overlap, score_threshold e k. O critério de avaliação adotado foi a proporção de respostas que continham claramente a informação esperada.

Nos testes, o sistema respondeu às perguntas com base apenas na base indexada. A cadeia QA foi executada com o ConversationalRetrievalChain, utilizando o modelo GPT-3.5-turbo e prompt técnico restritivo, evitando extrapolações. As respostas foram comparadas com as esperadas por meio de análise de similaridade semântica. Respostas incompletas, genéricas ou incompatíveis com os documentos foram consideradas incorretas. Os testes foram realizados localmente com execução via Streamlit, simulando um ambiente de uso real. O sistema permitiu carregamento incremental de documentos, ajuste dinâmico de parâmetros e visualização das fontes utilizadas. A configuração foi mantida constante para garantir a reprodutibilidade dos resultados.

IV. RESULTADOS E DISCUSSÃO

O sistema foi avaliado em um contexto técnico real, voltado à automação do acesso a informações extraídas de documentos técnicos utilizados por engenheiros e analistas da indústria de equipamentos para manuseio e armazenagem de granéis sólidos. Esse domínio envolve documentos como escopos de fornecimento, especificações construtivas, normas técnicas e instruções operacionais, geralmente estruturados em arquivos extensos, com vocabulário especializado e alta densidade informacional. Tal cenário representa um desafio típico de engenharia de projetos industriais, nos quais a rápida e precisa localização de informações específicas é essencial para decisões técnicas e comerciais.

Os testes realizados permitiram avaliar a capacidade do sistema em recuperar informações relevantes a partir desses documentos e em gerar respostas alinhadas às expectativas do domínio. A seguir são apresentados os principais resultados obtidos com base no conjunto de avaliação e no processo de otimização de parâmetros.

Inicialmente, o sistema foi executado com os parâmetros recomendados por literatura e práticas comuns em projetos baseados em recuperação aumentada: chunk_size = 1000, chunk_overlap = 100, score_threshold = 0.5, k = 4. Nessa configuração, o sistema atingiu uma taxa de acerto de 62,5%, considerando a proporção de respostas que apresentaram, de forma completa e precisa, as informações esperadas em relação às oito perguntas da base de teste.

As respostas incorretas foram, em sua maioria, decorrentes de limitações na segmentação dos documentos, que fragmentaram informações relevantes entre diferentes trechos. Em outros casos, a recuperação de documentos com baixa similaridade semântica reduziu a qualidade da resposta final gerada.

A aplicação da rotina de otimização de parâmetros, conforme o script tuning_chatbot_original.py, resultou na identificação de uma combinação mais eficaz

para o conjunto de dados avaliado. Os melhores resultados foram obtidos com os seguintes parâmetros: chunk_size = 1100, chunk_overlap = 150, score_threshold = 0.7 e k = 3. Com essa configuração, o sistema atingiu uma taxa de acerto de 87,5%, demonstrando melhora significativa em relação à configuração padrão. A maior granularidade dos trechos e o aumento do limiar de similaridade contribuíram para a recuperação de documentos mais relevantes e semanticamente densos, resultando em respostas mais precisas e completas.

A análise qualitativa indicou que, nos casos em que o sistema recuperou corretamente os documentos relevantes, o modelo LLM foi capaz de sintetizar as informações de forma técnica e objetiva, respeitando as diretrizes do prompt configurado. A exibição do trecho e do nome do documento utilizado na resposta facilitou a verificação da confiabilidade das respostas, agregando transparência ao processo.

Por outro lado, observou-se que a ausência de contexto suficiente ou a segmentação inadequada compromete diretamente a performance do modelo. Esses fatores reforçam a importância de mecanismos de ajuste fino e validação contínua da base de conhecimento, especialmente em domínios técnicos com vocabulário especializado e alta densidade informacional.

Embora os resultados tenham sido satisfatórios, algumas limitações foram observadas. A dependência de modelos hospedados na nuvem (como o modelo da família GPT-3.5-turbo (OpenAI) pode restringir o uso da ferramenta em ambientes com requisitos de privacidade ou disponibilidade offline. Além disso, o desempenho do sistema está diretamente vinculado à qualidade dos dados fornecidos e à estrutura documental original, exigindo etapas rigorosas de curadoria e normalização.

Mesmo que a versão atual implementada dependa da utilização de modelos via API da OpenAI, foram realizados testes exploratórios com modelos quantizados em 4-bit e 8-bit, como Mistral 7B, Phi-3 e TinyLLaMA, com o objetivo de viabilizar a execução local da solução e eliminar custos relacionados ao uso de API's com a OpenAI. No entanto, os resultados não atenderam aos critérios mínimos de precisão e completude exigidos para respostas fundamentadas em documentos técnicos, indicando que essas alternativas, embora promissoras, ainda apresentam limitações em domínios especializados.

O uso de computadores de alto desempenho, com pelo menos 48 GB de memória RAM, pode permitir a execução de modelos mais robustos localmente — como o Mistral 8x7B ou outros modelos open-source de maior capacidade, conforme demonstrado por Philipp et al. (2025) —, potencialmente elevando a qualidade das respostas geradas sem a necessidade de dependência de serviços em nuvem.

V. IMPACTOS

O software desenvolvido representa uma contribuição relevante para a construção de QA com base em LLMs, especialmente em contextos de pesquisa aplicada, engenharia e gestão do conhecimento técnico. Sua arquitetura modular, combinada com uma interface acessível e recursos de personalização, permite que diferentes perfis de usuários explorem soluções baseadas em RAG sem necessidade de conhecimentos avançados em programação.

A ferramenta também se destaca pela incorporação de um módulo de ajuste automático de parâmetros com Optuna, que permite calibrar a cadeia de QA conforme a estrutura dos documentos e o tipo de consulta. Isso contribui para adaptar a solução a diferentes domínios e casos de uso, promovendo maior precisão e eficiência.

Ao oferecer recursos de ingestão documental, controle de inferência, visualização de fontes e ajuste de desempenho, o sistema se posiciona como uma plataforma integrada

e flexível para a construção de chatbots técnicos especializados, com potencial para aplicações em ambientes acadêmicos, industriais e corporativos.

VI. CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento de um software voltado à criação de chatbots de Perguntas e Respostas com base em LLMs e recuperação aumentada. A solução foi projetada para oferecer uma experiência completa, abrangendo desde a ingestão de documentos até a geração de respostas, passando por etapas de ajuste fino e validação com base em perguntas e respostas de referência.

A ferramenta demonstrou viabilidade técnica e flexibilidade de uso, evidenciando sua aplicabilidade em diferentes contextos organizacionais. Sua arquitetura está preparada para futuras evoluções, incluindo a integração de modelos mais eficientes para execução local e a ampliação de mecanismos de avaliação contínua. A proposta contribui para ampliar o acesso a tecnologias de linguagem natural aplicada à gestão do conhecimento, com potencial de impacto em setores que lidam com documentação técnica, processos operacionais e suporte à decisão.

Como continuidade deste trabalho, propõem-se diversas frentes de evolução da ferramenta. A primeira refere-se à integração de modelos com arquitetura Mixture of Experts (MoE) mais avançados, capazes de serem executados localmente com desempenho satisfatório, especialmente em ambientes com maior capacidade computacional. Isso permitirá reduzir a dependência de APIs externas e promover maior autonomia no uso do sistema.

Outra linha de desenvolvimento contempla a integração de modelos de embedding alternativos, mais adequados a domínios específicos, e o aprimoramento do pipeline de ingestão documental. Isso inclui a adoção de ferramentas para extração de dados semiestruturados (como tabelas e formulários), mecanismos de segmentação inteligente baseados em estrutura semântica, e técnicas de enriquecimento contextual para melhorar a representatividade dos trechos utilizados na inferência.

Propõe-se também a inclusão de um módulo de avaliação contínua com base em feedback humano. Após cada interação, o sistema poderá coletar a percepção do usuário quanto à precisão e utilidade da resposta, armazenando esses dados para análise posterior. A partir dessa análise, será possível identificar padrões de erro, ajustar os parâmetros de recuperação, reindexar trechos mal segmentados e, futuramente, adaptar a base vetorial com técnicas de aprendizado supervisionado leve. Esse mecanismo visa garantir que a lógica de recuperação evolua continuamente, acompanhando mudanças nos documentos e nas demandas dos usuários.

Por fim, pretende-se tornar o software acessível como um pacote de código aberto, com documentação técnica detalhada, tutoriais e possibilidade de execução em diferentes ambientes, incentivando sua adoção por comunidades acadêmicas e industriais interessadas no desenvolvimento de sistemas de QA baseados em documentos.

VII. REFERÊNCIAS

ARNOLD, P. G. et al. Performance of large language models for CAD-RADS 2.0 classification derived from cardiac CT reports. **Journal of Cardiovascular Computed Tomography**, 2025. ISSN 1934-5925. DOI: 10.1016/j.jcct.2025.03.007.

BIRHANE, A. et al. Science in the age of large language models. **Nature Reviews Physics**, v. 5, p. 277–280, 2023. DOI: 10.1038/s42254-023-00581-.

KOPP, T.; BAUMGARTNER, M.; KINKEL, S. Success factors for introducing industrial human-robot interaction in practice: an empirically driven framework.

International Journal of Advanced Manufacturing Technology, v. 112, p. 685–704, 2021. DOI: 10.1007/s00170-020-06398-0.

OTTER, D. W.; MEDINA, J. R.; KALITA, J. K. A survey of the usages of deep learning for natural language processing. **IEEE Transactions on Neural Networks and Learning Systems**, v. 32, n. 2, p. 604–624, fev. 2021. DOI: 10.1109/TNNLS.2020.2979670.

SALIM, M. S. et al. LLM based QA chatbot builder: a generative AI-based chatbot builder for question answering. **SoftwareX**, v. 29, 2025. Artigo 102029. DOI: 10.1016/j.softx.2024.102029.

STUREBORG, R.; ALIKANIOTIS, D.; SUHARA, Y. Large language models are inconsistent and biased evaluators. **arXiv preprint**, 2024. DOI: 10.48550/arXiv.2405.01724.

SU, P.-H. et al. Deep learning for conversational AI. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: **Tutorial Abstracts**. New Orleans: ACL, 2018. p. 27–32. DOI: 10.18653/v1/N18-6006.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. **In: Proceedings** of the 27th International Conference on Neural Information Processing Systems – Volume 2 (NIPS'14). MIT Press, 2014. p. 3104–3112.

VASWANI, A. et al. Attention is all you need. **Advances in Neural Information Processing Systems**, v. 30, 2017.

VINYALS, O.; LE, Q. A neural conversational model. **arXiv preprint**, 2015. DOI: 10.48550/arXiv.1506.05869.

WALLACE, R. S. The Anatomy of A.L.I.C.E. In: Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer. **Dordrecht: Springer**, 2007. p. 181–210. DOI: 10.1007/978-1-4020-6710-5_13.

WEIZENBAUM, J. ELIZA—a computer program for the study of natural language communication between man and machine. **Communications of the ACM**, v. 9, n. 1, p. 36–45, jan. 1966. DOI: 10.1145/365153.365168.

YIN, J. et al. Neural generative question answering. In: Workshop on Human-Computer Question Answering. **Association for Computational Linguistics (ACL)**, 2016. p. 36–42. DOI: 10.18653/v1/W16-0106.

VIII. COPYRIGHT