UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE EDUCAÇÃO SUPERIOR DO ALTO VALE DO ITAJAÍ – CEAVI
ENGENHARIA DE SOFTWARE – ESO

LUCAS PIETRO BIASI RAYZER

AVALIAÇÃO DO USO DE RAG EM LLM'S NO CONTEXTO DE INFORMAÇÕES INSTITUCIONAIS DA UDESC

IBIRAMA
2025

LUCAS PIETRO BIASI RAYZER

AVALIAÇÃO DO USO DE RAG EM LLM'S NO CONTEXTO DE INFORMAÇÕES INSTITUCIONAIS DA UDESC

Trabalho de conclusão apresentado ao curso de Engenharia de Software do Centro de Educação Superior do Alto Vale do Itajaí (CEAVI), da Universidade do Estado de Santa Catarina (UDESC), como requisito parcial para a obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Fernando dos Santos

IBIRAMA
2025

LUCAS PIETRO BIASI RAYZER

AVALIAÇÃO DO USO DE RAG EM LLM'S NO CONTEXTO DE INFORMAÇÕES INSTITUCIONAIS DA UDESC

Trabalho de conclusão apresentado ao curso de Engenharia de Software do Centro de Educação Superior do Alto Vale do Itajaí (CEAVI), da Universidade do Estado de Santa Catarina (UDESC), como requisito parcial para a obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Fernando dos Santos

BANCA EXAMINADORA:

Prof. Fernando dos Santos, Dr.
UDESC

Membros:

Prof. Carlos Alberto Barth, Me.
UDESC

Prof. Marcelo de Souza, Dr.
UDESC

Ibirama, 02 de dezembro de 2025

Aos estudantes da Universidade do Estado de Santa Catarina, pela inspiração de sempre!

AGRADECIMENTOS

Agradeço ao meu orientador, Prof. Dr. Fernando dos Santos, por aceitar conduzir este trabalho e por dedicar seu tempo ao projeto. Sua orientação atenciosa e prestativa foi essencial para o desenvolvimento desta pesquisa.

A todos os meus professores do curso de Engenharia de Software da Universidade do Estado de Santa Catarina (UDESC), minha gratidão pela excelência e qualidade técnica dos ensinamentos que moldaram minha formação.

Aos meus amigos e colegas, em especial — em especial André, Diego, Guilherme, João e Ramon — que caminharam ao meu lado nos momentos difíceis e alegres ao longo dos semestres. Agradeço pela parceria, pela constante troca de conhecimento e pelo crescimento mútuo que compartilhamos, tanto profissional quanto pessoal.

Agradeço também aos meus colegas de estágio. Em especial ao líder técnico Glauco, por sua dedicação às boas práticas de software e por compartilhar seu vasto conhecimento sobre as práticas essenciais para a construção de aplicações de alta qualidade.

Aos meus familiares, por todo o apoio durante minha trajetória. Em especial aos meus pais, Cassiano e Eliete, minha maior inspiração e motivação, que nunca mediram esforços e estiveram sempre ao meu lado. Sou profundamente grato por todas as oportunidades e por cada sacrifício feito para que eu pudesse chegar até aqui e abrir novas portas em minha vida.

“The best is yet to come.” (Scorpions)

RESUMO

A dispersão de documentos burocráticos e normativos na Universidade do Estado de Santa Catarina (UDESC) dificulta o acesso rápido à informação pela comunidade acadêmica, reduzindo a eficiência dos processos institucionais. Este trabalho teve como objetivo aplicar e avaliar um assistente institucional capaz de unificar e facilitar o acesso a tais documentos e procedimentos. A metodologia empregou uma técnica de geração de respostas apoiada pela geração aumentada por recuperação. Para isso, foi elaborado um fluxo de processamento composto pela coleta de 49 documentos institucionais, extração do conteúdo textual, normalização e segmentação. Os textos foram convertidos em representações numéricas e armazenados em uma base vetorial, cuja operação foi conduzida por uma estrutura própria para aplicações desse tipo. Dois modelos de linguagem foram comparados: o Meta Llama 3.1 8B Instruct, executado localmente, e o Google Gemini 2.5-pro, acessado remotamente. A análise quantitativa indicou que o principal empecilho do sistema foi a baixa precisão na recuperação dos trechos relevantes, frequentemente acompanhados de contextos irrelevantes. Nesse contexto, o modelo Gemini 2.5-pro apresentou desempenho superior, demonstrando maior fidelidade às informações recuperadas e maior correção nas respostas. Em contraste, o modelo Llama 3.1 8B mostrou maior sensibilidade ao ruído textual presente nos documentos. Conclui-se que o Gemini 2.5-pro se mostra mais adequado e confiável para uso institucional, destacando-se por sua capacidade de filtrar informações imprecisas e por evitar respostas sem respaldo factual, característica essencial para reduzir o risco de alucinações.

Palavras-chave: Agente. RAG. LLM. Processamento de Linguagem Natural. RAGAS.

ABSTRACT

The dispersion of bureaucratic and normative documents at the State University of Santa Catarina (UDESC) hinders quick access to information by the academic community, reducing the efficiency of institutional processes. This study aimed to apply and evaluate an institutional assistant capable of unifying and facilitating access to such documents and procedures. The methodology employed a response-generation technique supported by retrieval-augmented generation. A processing flow was developed, including the collection of 49 institutional documents, extraction of textual content, normalization, and segmentation. The texts were converted into numerical representations and stored in a vector database operated by a structure designed for this type of application. Two language models were compared: Meta Llama 3.1 8B Instruct, executed locally, and Google Gemini 2.5-pro, accessed remotely. Quantitative analysis indicated that the main limitation of the system was the low precision in retrieving relevant passages, which were often accompanied by irrelevant segments. In this scenario, the Gemini 2.5-pro model showed superior performance, demonstrating greater fidelity to the retrieved information and higher accuracy in its responses. In contrast, the Llama 3.1 8B model exhibited greater sensitivity to textual noise present in the documents. It is concluded that Gemini 2.5-pro is more suitable and reliable for institutional use, standing out for its ability to filter imprecise information and avoid responses lacking factual support, which is essential to reducing the risk of hallucinations.

Keywords: Agent. RAG. LLM. Natural Language Processing. RAGAS.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CAP	Câmara de Administração e Planejamento
CEAVI	Centro de Educação do Alto Vale do Itajaí
CECC	Câmara de Extensão, Cultura e Comunidade
CEG	Câmara de Ensino de Graduação
CONSEPE	Conselho de Ensino, Pesquisa e Extensão
CONSUNI	Conselho Universitário
CPPG	Câmara de Pesquisa e Pós-Graduação
FAISS	Facebook AI Similarity Search
IA	Inteligência Artificial
NLG	Natural Language Generation
NLP	Natural Language Process
NLU	Natural Language Understanding
PDF	Portable Document Format
PLM	Pretrained Language Model
RAG	Retrieval Augmented Generation
RAGAS	Retrieval Augmented Generation Assessment
RAIA	Rede de Avanço em Inteligência Artificial
SECON	Secretaria dos Conselhos Superiores
SETIC	Secretaria de Tecnologia da Informação e Comunicação
UDESC	Universidade do Estado de Santa Catarina
UFSM	Universidade Federal de Santa Maria
VM	Virtual Machine

LISTA DE SÍMBOLOS

^	Operador de versionamento. indica compatibilidade com atualizações de versão menor e correção dentro de uma mesma versão maior.

SUMÁRIO

1 INTRODUÇÃO

Atualmente, várias empresas estão buscando alternativas sustentáveis e tecnológicas, com fim de reduzir custos e aumentar o engajamento entre produto e consumidor. A empresa Xerox, por exemplo, deixou de mandar manuais junto com seus produtos, agora utiliza um chatbot, agente este, responsável por direcionar o usuário dentro do site, até chegar ao modelo que já adquirido no site da marca. Outro caso é o Chat Diário Oficial, desenvolvido pela RAIA - Rede de Acesso à Informação Aberta (2025), que visa agilizar o acesso diário dos servidores públicos a documentos do Diário Oficial do Município de Belo Horizonte (MG).

A gestão de documentos internos desempenha um papel fundamental no funcionamento das universidades, abrangendo desde processos administrativos simples até trâmites mais burocráticos. Essa organização é essencial para garantir que discentes, docentes e servidores consigam conduzir procedimentos institucionais de forma eficiente. No entanto, a dispersão dessas informações e o alto volume de processos dificultam o acesso ágil e preciso a documentos essenciais, comprometendo a transparência e a autonomia dos usuários.

Com o avanço de tecnologias digitais, os assistentes virtuais baseados em IA surgem como uma alternativa promissora para otimizar a comunicação e organização desses documentos. Já existem aplicações de IA voltadas ao atendimento acadêmico, como a de Oliveira (2024) que proporciona esclarecimento de dúvidas e suporte a alunos e servidores. No entanto, a maioria dessas soluções não foca na centralização de informações burocráticas, o que gera um vazio na acessibilidade e na eficiência do acesso a documentos institucionais.

Dessa forma, o desenvolvimento de um sistema inteligente capaz de fornecer respostas rápidas e precisas, é uma estratégia para estruturar e organizar os trâmites acadêmicos em um ambiente acessível. A implementação de um modelo de linguagem baseado em IA, aliado a técnicas de Recuperação de Informação via Geração Aumentada por Recuperação (RAG), pode representar um grande avanço. Nesse modelo, informações são recuperadas de uma base de dados antes de serem usadas na geração da resposta, tornando o sistema mais preciso e alinhado ao conteúdo institucional. Isso ajuda os usuários encontrem documentos de forma intuitiva, reduzindo a dependência de atendimentos presenciais e otimizando o fluxo interno de informações na universidade.

1.1 PROBLEMA

A UDESC ainda apresenta baixa clareza no que diz respeito ao fácil acesso à documentações burocráticas. Essa fragmentação na gestão documental, acaba prejudicando a eficiência dos processos institucionais e gera diversos impactos para a comunidade academica. Discentes enfrentam dificuldades para acessar informações essenciais, como regulamentos de cursos, prazos de matrículas e documentos administrativos, o que pode resultar em atrasos na conclusão de etapas burocráticas e até mesmo na perda de prazos importantes. Docentes, por sua vez, podem encontrar obstáculos na obtenção de normativas internas, diretrizes de pesquisa e procedimentos

administrativos, tornando mais complexa a gestão de suas atividades acadêmicas.

Além disso, os setores administrativos são diretamente afetados pela alta demanda por atendimento presencial ou por e-mail, sobrecarregando os servidores e aumentando o tempo de resposta para solicitações. A dispersão dos documentos e a falta de um mecanismo eficiente de recuperação de informações resultam em consultas repetitivas e inconsistências nos procedimentos, reduzindo a transparência e a confiabilidade do sistema institucional.

Outro reflexo é o impacto na evasão estudantil, que é destacada pelo fato de que menos de metade das matrículas no ensino superior resulta em diploma no Brasil, conforme reportagem da Ravagnani (2023). Muitos alunos, especialmente aqueles em situação de vulnerabilidade socioeconômica, desconhecem políticas institucionais de assistência estudantil, como auxílios financeiros, moradia estudantil e bolsas de permanência. Apesar das divulgações por e-mail e redes sociais, essas informações frequentemente se perdem na rotina acadêmica, sejam pela falta de tempo para acompanhar ou falta de alcance de postagens, logo, as informações acabam não chegando de forma clara aos estudantes que mais precisam. Se houvesse um canal unificado e acessível para consulta dessas políticas, os alunos poderiam encontrar com mais facilidade os auxílios que a universidade oferta.

1.2 OBJETIVOS

Esta seção aborda os objetivos geral e específicos do trabalho.

1.2.1 Objetivo Geral

O objetivo deste trabalho é realizar um estudo experimental em LLM's baseadas em geração de respostas com estratégia RAG, aplicado à centralização de documentos e procedimentos institucionais da UDESC, analisando seu desempenho e sua adequação ao uso institucional.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, são definidos os seguintes objetivos específicos:

a) Mapear os principais documentos e trâmites presentes no ambiente acadêmico administrativo da UDESC;

b) Implementar um protótipo de um agente utilizando a estratégia RAG, integrando um modelo de linguagem de grande porte a uma base vetorial construída a partir de documentos institucionais;

c) Realizar testes experimentais comparando diferentes modelos de linguagem, observando sua capacidade de recuperar e utilizar informações relevantes;

d) Avaliar o desempenho da ferramenta por meio de métricas automatizadas, considerando aspectos como correção das respostas, precisão na recuperação de contexto e coerência do conteúdo gerado.

1.3 JUSTIFICATIVA

A crescente digitalização, ao lado do grande volume de processos acadêmicos exige soluções inovadoras que acompanhem a evolução tecnológica e atendam às demandas da comunidade universitária da UDESC. A sobrecarga dos canais de atendimento tradicionais e a descentralização das informações dificultam o acesso rápido e eficiente a documentos institucionais, impactando diretamente a experiência de alunos, professores e servidores. Nesse contexto, torna-se necessário explorar novas abordagens que promovam autonomia, clareza e acessibilidade nos trâmites internos.

A utilização de agentes inteligentes baseados em modelos de linguagem natural de grande escala, associados à técnicas como RAG já mostraram-se opções inteligentes, sendo aplicados com sucesso em outros setores, como Saha e Saha (2024), ao trabalhar esta técnica para apoiar estudantes estrangeiros a se adpatarem à cultura local, ou Oliveira (2024), que aplicou para alimentar um modelo pré-treinado, com o objetivo de fornecer informações retiradas do site da UFSM. Essa tecnologia pode reduzir a dependência de atendimentos presenciais, otimizar o tempo de resposta e proporcionar um ambiente mais acessível e inclusivo para todos os usuários da universidade.

1.4 METODOLOGIA

Para que os objetivos fossem alcançados, tornou-se necessário avaliar quais modelos de linguagem seriam mais adequados ao contexto de aplicação, especificamente para tarefas de geração de texto a partir de documentos institucionais. Modelos de linguagem de grande porte são sistemas de inteligência artificial treinados em extensos conjuntos de dados textuais, capazes de interpretar e produzir linguagem humana de forma coerente. A análise buscou identificar qual modelo apresentava melhor desempenho ao responder dúvidas dos usuários, considerando critérios como qualidade das respostas, estabilidade diante de informações incompletas e capacidade de lidar com ruído textual.

Com esse propósito, o trabalho avaliou diferentes modelos de linguagem, comparando seu comportamento dentro de um ambiente RAG para identificar aquele que oferecesse as respostas mais corretas e confiáveis. Além dessa etapa, foram coletados os documentos referentes aos trâmites institucionais da UDESC, que serviram como base para alimentar o agente baseado em recuperação e geração de respostas. Por fim, a ferramenta foi avaliada por meio de métricas automatizadas específicas para esse tipo de aplicação, permitindo medir a precisão da recuperação, a fidelidade das respostas e a robustez do sistema frente a variações no conteúdo textual.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda os conceitos necessários para entendimento dos processos que envolvem uma universidade, além do funcionamento de um agente de conversação. Para isso, são apresentados os fundamentos que sustentam tal tecnologia, explorando desde os princípios dos agentes inteligentes, até técnicas mais avançadas com finalidade de demonstrar as vantagens e desvantages no contexto deste trabalho.

2.1 UDESC

A UDESC, é uma universidade pública e estadual no estado de Santa Catarina, sendo acessível e diversificada, oferecendo cursos de graduação, pós-graduação, mestrado e doutorado, além de ainda disponibilizar ensino a distância distribuídos em 12 centros espalhados pelo estado de Santa Catarina (Universidade do Estado de Santa Catarina, 2025). Essa estrutura, gera um volume de normas técnicas e necessidade de regulamentações, para garantir seu íntegro funcionamento. Para definir as normas de funcionamento da UDESC, foram criados os conselhos universitários, onde vários integrantes dos centros espalhados pelo estado de Santa Catarina, se reúnem em reuniões periódicas, com o intuito de discutir possíveis mudanças ou não sobre as normas em vigência, o que acaba gerando várias alterações. A tabela 1 demonstra a coleta de documentos ao longo dos três últimos anos, é possível ver a quantidade de normas só neste período.

Tabela 1 – Resoluções dos conselhos superiores nos últimos três anos

Conselho	**2025**	**2024**	**2023**
CONSUNI	12	97	105
Câmara CEG	7	10	32
Câmara CPPG	0	10	15
Câmara CECC	1	0	0
Câmara CAP	25	123	91

Fonte: Dados com base nas resoluções dos conselhos da UDESC (2023–2025).

Além dos conselhos gerais, há conselhos nos diversos centros de ensino que podem emitir outras normas, e cada uma pode ser diferente entre os centros acadêmicos. Isso acaba tornando ainda mais específico e de difícil acesso aos discentes e docentes, como por exemplo, dentro do escopo universitário, discentes podem perder provas por diversos motivos, o que leva ao dilema de onde encontrar instruções de o que fazer nesta situação.

Essa complexidade normativa, embora fundamental para garantir a coerência institucional, assegurando que todos os processos ocorram de forma justa e padronizada, acaba gerando um ambiente pesado ao que diz respeito à interação com o usuário, que reflete ao acesso e compreensão das normas. Muitos alunos e professores enfrentam dificuldades em localizar

rapidamente os documentos que regulamentam situações específicas. Por exemplo o CONSEPE (2015), em sua resolução n°039/2015, regulamenta as avaliações em segunda chamada, tal qual os prazos e instruções para solicitar a segunda via de avaliações.

Outro assunto de grande importância e complexo acesso, é a burocracia por trás da validação de créditos, onde grande parte dos cursos da graduação precisam concluir uma determinada carga horária, como o curso de Engenharia de Software ofertado pelo centro CEAVI, que tem em sua grade currícular antiga, a obrigatoriedade de concluir 17 créditos de atividades complementares, as quais podem incluir diversos tipos de atividades, tendo limite de créditos por categoria, as quais foram definidas pelo CONSEPE (2012), na resolução n°026/2012. Contudo, isso se encontra dentro do site da UDESC, com algumas rotas que não são claras, nem amplamente divulgadas dentro da universidade. Além disso, a linguagem utilizada nas resoluções costuma ser técnica e extensa, dificultando a compreensão por parte de discentes que não estão familiarizados com termos jurídicos ou administrativos.

2.2 AGENTES INTELIGENTES

De acordo com Russell e Norvig (2022), um agente é tudo que pode ser sensível ao ambiente que está, tendo a capacidade de absorver dados e ter a percepção de onde se encontra, afim de tomar decisões inteligentes com base no mundo, representado pelo local/ambiente que se encontra. Contudo, este trabalho foca em agentes de conversação, um tipo de agente computacional projetado para interagir com humanos por meio da linguagem natural, simulando a participação em um diálogo. Com finalidade de aprovar ou não a eficiência de um agente, foi definido por Turing (1950) o que ele chamou de teste de Turing. O objetivo é testar se a máquina programada para imitar o pensamento humano, consegue realizar tal tarefa com êxito. Para isso, uma pessoa tem uma conversa cega com o agente e após a conversa, deveria decidir se teve o debate com um ser humano ou com uma inteligência artificial, logo, quando aprovada no teste a máquina seria suficientemente inteligente a ponto de passar despercebida na conversação.

Os agentes de conversação, seguem duas principais vertentes, sendo eles baseados em regras ou em inteligência artificial (Oliveira, 2024).

2.2.1 Agentes baseados em regras

Agentes que possuem natureza baseada em regras recebem conjuntos de dados predefinidos para responderem perguntas do usuário com base nestas informações. Contudo, acabam sofrendo limitações, já que só conseguem responder perguntas associadas ao conjunto de dados fornecido. Geralmente usam fluxo de decisão ou buscas em árvores para guiar as conversar, explica Oliveira (2024). Ainda no viés desta natureza, Gabriel (2022) discorre sobre como o pensamento do modelo que é denominado simbolista. Agentes que seguem esta ideia acabam tentando imitar o funcionamento do pensamento humano, seguindo uma ordem lógica que interliga o pensamento para gerar soluções.

2.2.2 Agentes baseados em IA

Na segunda vertente, Oliveira (2024) define os agentes baseados em IA como um agente que busca ententer a linguagem humana de uma forma mais natural. Este tipo de agente utiliza *natural language processing* (NLP), o que possibilita maior evolução e aprendizado, logo, melhorando a interação com o usuário e tornando os diálogos mais naturais. Agentes que seguem tal método de aprendizado, podem ainda ser denominados conexionistas, já que são inspirados na forma do cérebro, não só em como ele funciona de forma lógica, mas também de forma fisiológica, utilizando do modelo de funcionamento das redes neurais. Nas redes neurais as respostas não são geradas por encadeamentos de ideias preestabelecidas, mas sim por camadas de neurônios artificiais que interagem e aprendem para responder da melhor maneira, isto é, as soluções surgem com base na tentativa e erro, sendo esta, evolucionista e tendo a capacidade de resolver problemas, não de seguir regras (Gabriel, 2022).

2.3 MODELOS DE LINGUAGEM DE GRANDE ESCALA (LLM)

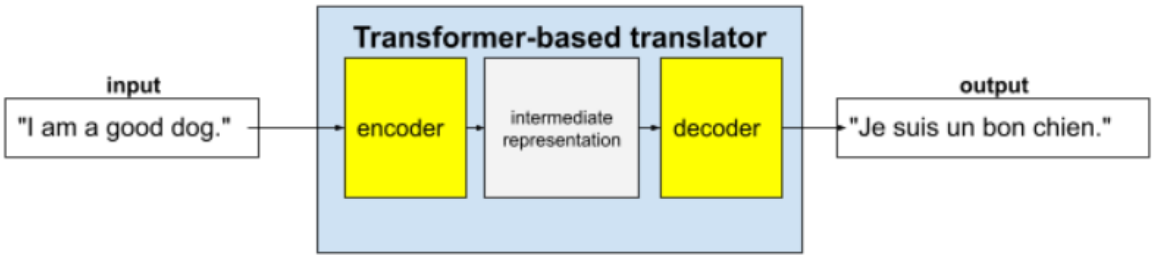
Os modelos de linguagem de grande escala, ou LLM, são amplamente estudados e vem evoluindo com rapidez, principalmente após o lançamento do ChatGPT em meados de 2022. De acordo com Minaee et al. (2024) os modelos de larga escala são também modelos estatísticos e pré treinados com base em redes neurais, que primordialmente eram chamados de modelos de linguagem estatísticos. Os LLM's são baseados em um tipo específico de rede neural, o *transformer*, onde podemos ver como é aplicado no exemplo da figura 1 para aplicativos de tradução.

Figura 1 – Aplicação baseada em Transformer para tradução do inglês para o francês

Fonte: Google (2025).

De acordo com Google (2025), a arquitetura *transformer* possui codificadores e decodificadores. O codificador é responsável pela conversão de textos de entrada para uma representação intermediária. Já o decodificador converte a representação preliminar em texto útil, onde ambos são redes neurais massivas. Seguindo na ideia de um tradutor, a figura 2 exibe onde o codificador processa uma mensagem em inglês e o decodificador converte em texto de saída, como no exemplo a frase foi traduzida para a língua francesa.

Figura 2 – Transformer completo para tradução do inglês para o francês



Fonte: Google (2025).

Com isso, os modelos de linguagem de grande porte utilizam vastas quantidades de dados textuais para aprender padrões linguísticos complexos, compreendendo relações entre palavras, frases e estruturas discursivas. Conforme destaca Minaee et al. (2024), esses modelos superam os tradicionais modelos de linguagem pré-treinados, sobretudo em função de sua escala, uma vez que podem conter centenas de bilhões de parâmetros. Essa característica lhes confere maior robustez, além de capacidades superiores de compreensão e geração de linguagem. No contexto deste trabalho, o papel central dos LLMs está na possibilidade de integrar técnicas RAG e atuar como agentes inteligentes capazes de interpretar perguntas e produzir respostas fundamentadas nos documentos institucionais da UDESC.

Além de seu tamanho e capacidade de processamento, os LLMs também apresentam habilidades emergentes, ou seja, competências que não estão explicitamente programadas, mas que surgem como resultado do treinamento em larga escala com dados diversos (Wei et al., 2022). Entre essas habilidades estão o *in-context learning*, onde o modelo aprende com poucos exemplos dados em tempo de inferência, o que permite que o modelo execute tarefas descritas em linguagem natural sem necessidade de ajustes adicionais no modelo.

No entanto, para que um LLM atue efetivamente como um agente inteligente e para que seja possível o aprimoramento de modelos existentes com conhecimento específico, diversas etapas e tecnologias de apoio são cruciais.

Primeiramente, é essencial compreender os fundamentos de um NLP responsável por detalhar como as máquinas processam e entendem a linguagem humana. Em seguida, para alimentar esses modelos com dados relevantes, muitas vezes é necessário realizar o processamento de arquivos, extraindo informações de fontes diversas. O texto extraído raramente está pronto para uso direto, exigindo uma normalização de texto para padronização e limpeza. Fundamentalmente, para que os LLM's possam processar esse texto, ele precisa ser convertido em representações numéricas, o que é feito através de *embeddings*.

Finalmente, para gerenciar e consultar eficientemente esses *embeddings* em larga escala, especialmente quando se busca construir uma base de conhecimento para o LLM, os bancos de dados vetoriais se tornam indispensáveis. As seções a seguir detalharão cada um desses componentes vitais para a construção de um sistema robusto e inteligente baseado em LLM.

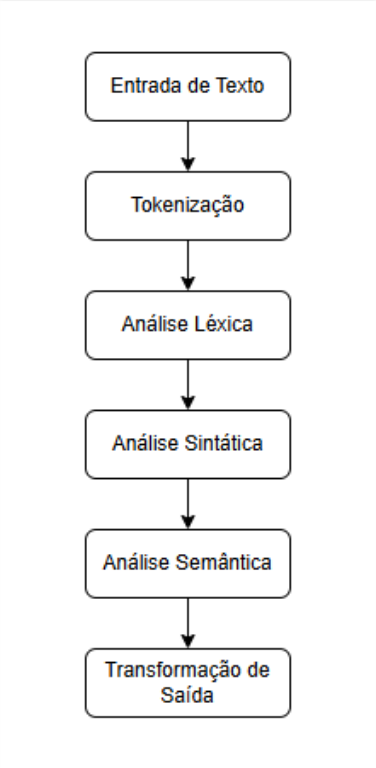
2.3.1 Natural Language Process

Ao interagir com máquinas computacionais, o usuário irá interagir apenas em sua língua de origem, logo, o computador precisa realizar a tradução pra linguagem de máquina e refazer o trabalho para voltar à linguagem solicitada pelo usuário. É justamente essa a área de atuação do NLP. De acordo com Caseli e Nunes (2024), o NLP busca resolver diversos problemas computacionais que envolvem o tratamento de uma linguagem falada pelo ser humano, seja ela escrita ou oral. Essa área se divide em duas vertentes principais: a NLU (*Natural Language Understanding*) e a NLG (*Natural Language Generation*).

No contexto de uma interação com um chatbot, a NLU é responsável por interpretar o que o usuário deseja realizar a partir do comando fornecido. A máquina tenta compreender os significados da linguagem humana e determinar se deve retornar uma resposta ou executar uma ação. Já a NLG cuida da geração da resposta, decidindo o que será respondido e como essa resposta será apresentada ao usuário. O NLP utiliza dados não estruturados e baseados em textos, integrando regras de linguagens, técnincas estatísticas e aprendizado de máquina para permitir que aplicações como assistentes virtuais gerem respostas que simulem as de um humano e compreendam o que é solicitado (Oliveira, 2024).

Para garantir o funcionamento de um NLP, Martins et al. (2020) afirma que após a entrada de texto fornecida pelo usuário, ele precisa seguir as etapas de tokenização, análise léxica, análise sintática, análise semântica e transformação de saída, como retratado na figura 3.

Figura 3 – Fluxo base para funcionamento de um NLP



Fonte: adaptado de Dale (2010).

Descrição das etapas de um NLP, conforme (Dale, 2010):

- **Entrada de texto:** É definido pelo método que o usuário fornece a informação, podendo ser escrito ou falado.
- **Tokenização:** A tokenização é responsável por dividir o texto de entrada em pequenas unidades, os tokens, que podem ser palavras, partes de palavras ou símbolos, o que facilita a análise posteriormente.
- **Análise Léxica:** Após a tokenização, cada token é identificado e classificado de acordo com sua função, como substantivo, verbo ou adjetivo. Aqui é onde o NLP reconhece a estrutura básica do texto.
- **Análise Sintática:** A análise sintática organiza os tokens de acordo com regras gramaticais, em estruturas de árvores sintáticas ou hierarquias para entender a relação entre as palavras formalmente.
- **Análise Semântica:** A análise semântica associa o significado às estruturas da etapa sintática, permitindo interpretar o contexto e o sentido das frases no texto.
- **Transformação de Saída:** Ao fim, a transformação de saída converte a compreensão semântica em uma ação ou resposta adequada, finalizando a geração de uma resposta contextualizada.

2.3.2 Processamento de Arquivos

No contexto de sistemas que empregam modelos de linguagem natural, o processamento de arquivos é uma etapa fundamental que envolve a extração e organização de informações contidas em documentos, com o objetivo de alimentar a base textual sobre a qual o modelo será treinado ou consultará posteriormente. Esses documentos geralmente estão em formatos não estruturados, como PDF, exigindo o uso de bibliotecas específicas para extração de conteúdo textual (Lee; Owens, 2021).

Além da extração, é comum realizar uma curadoria que inclui a eliminação de arquivos duplicados, o agrupamento de conteúdos por tema, data ou tipo de documento, e a organização em estruturas que facilitem futuras indexações ou buscas semânticas (Lee; Owens, 2021). Essa preparação garante que o conteúdo inserido no modelo seja confiável, relevante e útil para gerar respostas com qualidade.

2.3.3 Normalização de Texto

Após a extração dos textos, ocorre o processo de normalização, etapa essencial do pré-processamento textual que visa tornar o conteúdo mais uniforme e semanticamente consistente. A normalização envolve a remoção de ruídos textuais, como quebras de linha inconsistentes,

acentuações incorretas e caracteres especiais. Também são aplicadas técnicas de padronização linguística, como transformar todas as palavras para letras minúsculas, converter datas para um mesmo formato e substituir abreviações por formas completas (Dale; Moisl; Somers, 2000).

Esse processo permite reduzir variações linguísticas que podem confundir os modelos estatísticos e semânticos utilizados por sistemas de NLP. A normalização é particularmente relevante para garantir que o modelo compreenda o contexto e mantenha coerência nas respostas geradas, além de contribuir para a criação de representações vetoriais mais precisas e informativas (Jurafsky, 2000).

2.3.4 Embeddings

Embeddings são representações numéricas de dados como textos, imagens e áudios em vetores de dimensão fixa, que preservam relações semânticas ou estruturais. Essa conversão facilita o processamento computacional, permitindo que algoritmos operem de forma mais eficiente e precisa sobre dados complexos.

Quando um objeto é transformado em *embedding*, ele é representado por um vetor cujos valores capturam características específicas desse elemento. A quantidade de dimensões do vetor pode variar conforme a complexidade do dado, podendo alcançar ou ultrapassar milhares de dimensões.

Nesse espaço vetorial multidimensional, a proximidade entre dois vetores reflete o grau de similaridade semântica entre os elementos que representam. Essa semelhança é geralmente medida por métricas de distância, como a distância euclidiana ou cosseno. Quanto mais próximos os vetores, mais semanticamente relacionados são os elementos.

Graças a essa capacidade de representar significados de forma matemática, *embeddings* são amplamente utilizados em tarefas de aprendizado de máquina e análise de grandes volumes de dados (Oliveira, 2024).

Para que documentos extensos possam ser eficientemente indexados e recuperados em sistemas RAG, é necessário dividi-los em segmentos menores, conhecidos como *chunks*. O *chunking* consiste em particionar o texto em unidades de tamanho controlado, preservando coerência semântica e evitando perda de contexto. Cada chunk é posteriormente convertido em um *embedding*, permitindo que o modelo compare semanticamente trechos específicos em vez de analisar o documento inteiro. A definição adequada do tamanho dos *chunks* influencia diretamente a qualidade da recuperação, já que segmentos muito curtos podem perder significado, enquanto trechos excessivamente longos dificultam a identificação da informação relevante. Essa etapa é, portanto, fundamental para garantir que o sistema identifique e recupere com precisão o conteúdo mais pertinente durante o processo de busca semântica.

Além da definição do tamanho dos *chunks*, outra variável importante nos sistemas RAG é o parâmetro k, utilizado durante a etapa de recuperação. Esse parâmetro determina quantos *chunks* mais similares à consulta do usuário serão retornados pelo mecanismo de busca vetorial. Em geral, valores menores de k tendem a fornecer trechos mais específicos e diretamente

relacionados à consulta, enquanto valores maiores ampliam a quantidade de contexto recuperado, ao custo de potencialmente incluir informações irrelevantes. A escolha adequada de *k* impacta diretamente a qualidade das respostas, visto que os documentos retornados são posteriormente utilizados na etapa de enriquecimento do *prompt*.

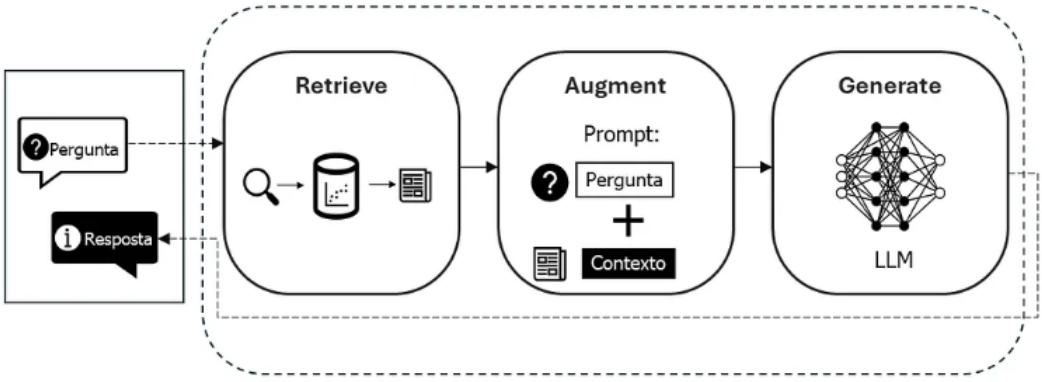
2.3.5 Banco de dados vetorias

Os bancos de dados vetoriais são um tipo especializado de bancos de dados. Estes foram desenvolvidos especificamente para armazenar vetores de alta dimensionalidade, onde cada dimensão corresponde a uma característica dos dados originais. Esses vetores são gerados por meio das técnicas de *embeddings*, que convertem dados não estruturados em representações numéricas que podem ser interpretadas por algoritmos de aprendizado de máquina. Uma das principais vantagens desse tipo de banco de dados é a eficiência na realização de buscas por similaridade. Ao armazenar informações como vetores, é possível utilizar algoritmos avançados, como a busca por vizinhos mais próximos, para identificar rapidamente itens com características semelhantes em grandes volumes de dados. Nesse cenário, vetores com propriedades parecidas são posicionados próximos uns dos outros em um espaço vetorial multidimensional, o que facilita a recuperação de informações relevantes. Esse mecanismo é fundamental em aplicações de inteligência artificial e ciência de dados, como reconhecimento de imagens, processamento de linguagem natural e sistemas de recomendação, onde a análise de similaridade tem maior importância do que correspondências exatas (Oliveira, 2024).

2.4 RETRIEVAL-AUGMENTED GENERATION (RAG)

Os LLMs aprendem a partir de um grande volume de dados estáticos. Isso significa que, sozinhos, eles não conseguem acessar informações novas ou em tempo real. Para superar essa limitação utiliza-se a abordagem RAG, que funciona com três principais etapas: *Retrieval*, *Augmentation* e *Generation*. A etapa de geração de uma resposta através do processamento de uma pergunta pode ser vista na figura 4.

Figura 4 – Estrutura RAG



Fonte: HACKERS (2023).

Como um todo, HACKERS (2023) explica que RAG é uma técnica que auxilia o modelo de geração de texto a recuperar as informações para alavancar performance e eficiência. A etapa de *Retrieve* é a base do sistema RAG. Sua eficácia determina a qualidade e a veracidade da resposta final. O objetivo é buscar e encontrar os trechos de informação mais relevantes para a pergunta do usuário dentro de uma base de conhecimento externa. Este processo se desdobra em duas fases, sendo elas a indexação e a recuperação.

A fase de indexação ocorre antes que qualquer consulta possa ser feita. Nessa etapa, a base de conhecimento, que é composta por documentos de texto, dados estruturados ou semiestruturados, precisa ser processada e organizada para possibilitar uma busca eficiente. Isso envolve:

- **Segmentação:** Os documentos são divididos em segmentos menores, ou *chunks*. A estratégia de segmentação define a granularidade da informação.
- **Criação de Embeddings:** Cada *chunck* de texto é transformado em um vetor numérico por meio de um modelo de linguagem específico para essa tarefa. Esses vetores capturam o significado semântico do texto.
- **Armazenamento Vetorial:** Os *embeddings* e seus *chunks* de texto correspondentes são armazenados em um banco de dados vetorial. Este banco de dados é otimizado para realizar buscas de similaridade semântica em alta velocidade.

De acordo com Oliveira (2024), a granularidade da segmentação é uma decisão de projeto crítica que envolve um importante balanço. Uma baixa granularidade corresponde a dividir os documentos em segmentos longos, o que preserva o contexto semântico, mas pode fazer com que a busca seja menos precisa. Por outro lado a alta granularidade corresponde a dividir os documentos em segmentos curtos, permitindo uma busca precisa que corresponde exatamente à dúvida do usuário, contudo, há um risco de perder o contexto global. O que Oliveira (2024) descreve como um prejuízo à "integridade semântica", ou seja, uma sentença isolada pode estar incompleta sem sentenças que à precedem ou sucedem.

A fase de recuperação tem início quando o usuário submete uma pergunta. O processo começa com a conversão da pergunta em uma *embedding* vetorial, aplicando o mesmo método de vetorização usado para os documentos na fase de indexação. Com este vetor representativo da pergunta, o sistema executa uma busca de similaridade no banco de dados vetorial. O objetivo é encontrar os *chunks* de texto cujas *embeddings* são matematicamente mais próximas à da pergunta. Como resultado, o sistema retorna um conjunto dos *chunks* mais relevantes, que servirão de base contextual para a próxima etapa do processo RAG.

Na fase de *Augmentation*, o sistema realiza o enriquecimento da pergunta original do usuário. De acordo com HACKERS (2023), essa etapa consiste em concatenar a pergunta inicial com os documentos relevantes recuperados na fase anterior, produzindo um *prompt* mais robusto e contextualizado. Esse *prompt* é estruturado para orientar o LLM sobre como proceder,

indicando claramente qual é a pergunta do usuário e quais informações devem servir de base para a formulação da resposta. Trata-se de uma etapa crucial para garantir que o modelo não apenas receba o conhecimento adicional, mas também compreenda como utilizá-lo adequadamente.

Com o *prompt* definido, a etapa final é a *Augmentation*. O prompt completo é submetido ao LLM, que agora tem a tarefa de sintetizar uma resposta coesa e precisa, utilizando exclusivamente as informações fornecidas no contexto. Conforme aponta Oliveira (2024), é importante que a etapa de *augmentation* já filtre redundâncias para não dificultar a identificação da informação central pelo modelo. Ao gerar a resposta com base no contexto fornecido, o LLM reduz drasticamente a probabilidade de inventar fatos. O agente, inclusive, pode citar as fontes consultadas, aumentando a transparência e a confiabilidade do sistema como um todo.

2.4.1 Avaliação de Sistemas RAG

Para análise dos resultados coletados, será utilizado a biblioteca RAGAS (ExplodingGradients, 2024). Com o avanço dos modelos LLM's, tornou-se comum a integração de mecanismos de RAG. Nessa abordagem, um componente de busca localiza informações relevantes em uma base de dados, enquanto o LLM utiliza esse contexto recuperado para gerar respostas fundamentadas e atualizadas.

A avaliação de sistemas RAG, contudo, apresenta maior complexidade do que a de modelos puramente generativos, pois envolve múltiplas dimensões de desempenho, desde a qualidade da recuperação até a fidelidade entre o contexto e a resposta. Nesse cenário, surge o RAGAS, uma ferramenta desenvolvida pela organização chamada Exploding Grandients para mensurar, de forma automática e estruturada, o desempenho de pipelines baseados em RAG (ExplodingGradients, 2024).

O RAGAS fornece métricas específicas que possibilitam uma análise modular do sistema, permitindo avaliar isoladamente as etapas de recuperação e de geração. Suas principais métricas incluem:

- ***Context Precision*:** Mede a proporção de contextos realmente relevantes dentre aqueles recuperados;
- ***Context Recall*:** Avalia o quanto do conhecimento relevante foi efetivamente recuperado;
- ***Faithfulness*:** Mensura a fidelidade da resposta em relação às informações do contexto;
- ***Answer Relevancy*:** Verifica o grau de adequação da resposta à pergunta do usuário;
- ***Answer Correctness*:** Indica o percentual de respostas consideradas corretas em relação a um gabarito ou resposta de referência.

Essas métricas variam de 0 a 1 e fornecem uma visão detalhada da eficácia do sistema, permitindo identificar gargalos tanto na busca de contexto quanto na geração textual. Por exemplo, uma alta taxa de *context recall* acompanhada de baixa *faithfulness* indica que o sistema recupera

adequadamente as informações, mas o LLM não as utiliza de forma consistente ao produzir a resposta.

Entre as métricas analisadas, *answer_correctness* e *faithfulness* são consideradas as mais críticas para a qualidade final do agente, pois avaliam diretamente a exatidão factual e a fidelidade da resposta ao conteúdo recuperado.

2.4.2 Cálculo das métricas do RAGAS

As métricas fornecidas pelo RAGAS permitem mensurar quantitativamente o desempenho de sistemas RAG em múltiplas dimensões. A seguir, são apresentadas como cada uma das métricas são calculadas, conforme a documentação oficial da ferramenta (ExplodingGradients, 2024).

2.4.2.1 Answer Correctness

Esta métrica avalia o quão correta é a resposta gerada pelo sistema em comparação com uma "resposta de referência" (*ground truth*) estabelecida. O cálculo não se limita a uma simples comparação de palavras. Ele é realizado em duas dimensões complementares:

- Correção Factual: O sistema analisa as afirmações (fatos) presentes na resposta gerada e as compara com as afirmações da resposta de referência. O objetivo é verificar se os fatos estão alinhados, penalizando tanto informações "inventadas" (presentes apenas na resposta gerada) quanto omissões (fatos da referência que não foram citados).
- Similaridade Semântica: A avaliação mede a proximidade de significado entre a resposta gerada e a resposta de referência. Isso permite que o sistema reconheça respostas como corretas mesmo que utilizem palavras ou estruturas de frase diferentes, desde que o sentido seja o mesmo.

2.4.2.2 Faithfulness

Esta métrica avalia o grau de fidelidade da resposta gerada em relação ao conteúdo presente no contexto recuperado. Seu objetivo principal é identificar e quantificar "alucinações"— informações presentes na resposta que não podem ser comprovadas pelas fontes recuperadas.

O RAGAS analisa cada sentença que compõe a resposta final e tenta validar em relação ao contexto fornecido. A pontuação é calculada como a proporção de sentenças que são factualmente consistentes com o contexto.

Por exemplo, se uma resposta gerada contiver três sentenças, mas apenas duas delas puderem ser diretamente verificadas no contexto recuperado, a pontuação de *faithfulness* será $\frac{2}{3}$, ou aproximadamente 0,67. Um valor de 1,0 indicaria que todas as afirmações contidas na resposta são sustentadas pelo contexto.

2.4.2.3 Context Precision

Esta métrica mede a capacidade do módulo de recuperação em classificar corretamente os *chunks* mais relevantes nas primeiras posições. Ela avalia se o sistema está apresentando conteúdo útil no topo de seus resultados de busca. A *Context Precision* foca na ordenação dos resultados. Ela verifica, posição por posição (de $k = 1$ até K), se o trecho recuperado é relevante ou irrelevante. A métrica final considera essa relevância ponderada pela posição.

Por exemplo, se o sistema for configurado para recuperar os 5 principais trechos (K=5) e, desses 5, apenas 3 forem considerados relevantes para responder à pergunta, a *Context Precision* seria $\frac{3}{5}$, ou 0,6.

2.4.2.4 Context Recall

Esta métrica também avalia o módulo de recuperação, mas sob a ótica da completude. Ela busca medir se o contexto recuperado contém todas as informações que seriam necessárias para compor a resposta de referência ideal.

O RAGAS compara as afirmações presentes na resposta de referência com o conjunto de informações disponíveis nos trechos de contexto recuperados. A pontuação é dada pela proporção de afirmações da referência que podem ser sustentadas pelo contexto.

Por exemplo, se a resposta de referência completa exigir quatro fatos distintos para ser considerada correta, mas o contexto recuperado pelo sistema contiver informações que sustentam apenas três desses fatos, a pontuação de *Context Recall* será $\frac{3}{4}$, ou 0,75.

2.4.2.5 Answer Relevancy

Esta métrica avalia o quão relevante a resposta gerada é em relação à pergunta original do usuário. Ela penaliza respostas que fogem do tópico, são incompletas ou incluem informações redundantes que não foram solicitadas.O método de cálculo desta métrica envolve um processo reverso:

- Primeiramente, o RAGAS analisa a resposta gerada e cria, artificialmente, um conjunto de perguntas que seriam bem respondidas por ela.
- Em seguida, essas perguntas artificiais são comparadas semanticamente com a pergunta original do usuário.

Se as perguntas artificiais (baseadas na resposta) forem muito similares em significado à pergunta original, entende-se que a resposta foi “direta ao ponto” e alinhada à intenção do usuário, resultando em uma pontuação alta.

2.5 TRABALHOS RELACIONADOS

Esta seção aborda os trabalhos relacionados, que de alguma forma, se assemelham com a proposta de desenvolvimento deste documento, enriquecendo a pesquisa e dando suporte ao desenvolvimento. O primeiro trabalho apresenta um desenvolvimento de um chatbot para facilitar acesso à informações gerais disponíveis no site da UFSM. O segundo trouxe a ideia da implementação de um guia para interações em um museu, no período da pandemia, para isso foi utilizado um NLP e um banco de dados contendo informações das obras de arte. O último trabalho é focado em trazer e centralizar a cultura de universidades internacionais, discorrendo sobre as dificuldades de estudantes intercâmbistas que muitas vezes ficam perdidos por conta de estarem em outro país, então tenta preencher esta lacuna com o chatbot que informa os discentes sobre a cultura.

2.5.1 Chatbot como Ferramenta de Apoio ao Acesso Às Informações Acadêmicas da UFSM

O trabalho de Oliveira (2024) tem como objetivo desenvolver um chatbot que simplifica o acesso às informações acadêmicas relacionadas a Universidade Federal de Santa Maria. Para isso, a autora desenvolveu uma ferramenta para raspagem de dados do site da universidade, utilizando a ferramenta SpiderAPI, que permite a extração de informações em formato JSON, incluindo HTML de cada página. Esses dados, então, foram processados para alimentar o modelo de linguagem escolhido, que foi o Gemini 1.5 Pro, produzido pelo Google (2024), um modelo de linguagem de larga escala gratuito.

Para treinar a ferramenta, foi utilizada a abordagem RAG, permitindo que o chatbot acesse dados externos, complementando a estrutura de pré-treinamento do modelo e garantindo respostas ágeis e dentro do contexto esperado. Para facilitar a integração entre a raspagem de dados e o modelo de linguagem, foi utilizado o *framework* LangChain, que oferece ferramentas para estruturar fluxos de trabalho e organizar dados de forma eficaz. O desenvolvimento do chatbot foi estruturado de forma a garantir uma interação intuitiva e eficiente, para facilitar o acesso às informações acadêmicas de maneira rápida e direta.

Com objetivo de validar a ferramenta foram realizados testes para avaliar o desempenho do chatbot. Ele respondeu com eficácia a perguntas variadas sobre a universidade, demonstrando precisão e relevância, mesmo com limitações de processamento. Em perguntas fora do escopo treinado, o sistema utilizou respostas padrão, orientando os usuários a consultar o site oficial. Ajustes nos parâmetros técnicos, como tamanho de *chunks* e buscas no banco vetorial, influenciaram positivamente a relevância e o tempo de resposta, evidenciando possibilidades de otimização do sistema.

2.5.2 Safer Museum Guide Interaction During a Pandemic and Further Using NLP in Human Interactive Museum Visits

Ceuca, Rednic e Chifu (2021) realizaram o desenvolvimento de um agente de conversação para museus, com o objetivo de oferecer uma alternativa segura de interação durante a pandemia. Para isso foi desenvolvido um chatbot baseado em NLP, sendo capaz de responder as perguntas relacionadas à aquele ambiente, as exposições artísticas ou aos artistas e aos movimentos da arte. O sistema foi implementado no Museu de Arte do Condado de Baia Mare, localizado na cidade de Baia Mare, na Romênia. A ferramenta se integra tanto a entradas de texto digitado, quanto entradas por voz, utilizando mecanismos de conversão de fala para texto e de texto para fala, fornecendo uma interação mais natural.

O banco de dados utilizado para treinar a ferramenta continha informações relevantes sobre os artistas, obras e seus contextos, permitindo que o agente respondesse de forma adequada ao tema. Para aferir a precisão, foram realizados testes práticos com usuários, obtendo uma variação nas respostas corretas de aproximadamente 40% a 80%, dependendo da formulação da pergunta pelo usuário. Esses resultados reforçam a viabilidade do uso de tecnologias de NLP em ambientes como museus, demonstrando que o chatbot pode enriquecer a experiência do visitante, principalmente em cenários de restrição à interação presencial.

2.5.3 Enhancing International Graduate Student Experience through AI-Driven Support Systems: A LLM and RAG-Based Approach

A aplicação desenvolvida por Saha e Saha (2024) tem como objetivo criar um chatbot de apoio personalizado para estudantes internacionais de pós-graduação, que enfrentam desafios como barreiras linguísticas, adaptações culturais e dificuldades acadêmicas. Para isso, foi utilizado o LLM da OpenAI (2023), o GPT-3.5 que utilizou da abordagem técnica do RAG, para oferecer respostas mais naturais e contextualizadas. Para a coleta de dados que iria alimentar o RAG, foram extraídos os 1000 melhores *posts* e comentários de cada comunidade do Reddit, uma plataforma onde usuários podem interagir em formato de fórum, além de que cada tópico poder ser promovido pelos usuários através de um sistema de votação, tornando mais relevante ou não.

Em seguida, os dados passaram por um processo de pré-processamento, que incluiu a organização, limpeza e remoção de informações sensíveis que pudessem comprometer a privacidade dos usuários. Os resultados obtidos indicaram que o modelo baseado em RAG apresentou desempenho superior em termos de precisão e relevância das respostas, quando comparado ao GPT-3.5 isoladamente. Avaliações utilizando métricas como RAGAS (qualidade de respostas em sistemas de recuperação aumentada) e BERTScore (similaridade semântica entre textos) evidenciaram a melhoria na qualidade das respostas geradas.

2.5.4 Considerações sobre trabalhos relacionados

Os trabalhos analisados nesta seção mostram o quão crescente é a aplicação de agentes inteligentes para apoio em alguma tarefa cotidiana no contexto informacional e educacional. Cada um dos estudos abordados contribui positivamente para o entendimento e desenvolvimento de sistemas que se baseiam em LLM's que combinam abordagem RAG.

O trabalho de Oliveira (2024) relaciona bem o desenvolvimento de um chatbot voltado à unificação de informações acadêmicas da UFSM. Utilizando Gemini 1.5 Pro, o framework LangChain e a raspagem de dados com SpiderAPI demonstrou um fluxo com boa estrutura de integração entre processos de extração, organização e geração de respostas. Esse estudo reforça a aplicabilidade da abordagem RAG em contextos institucionais, onde há uma vasta base documental que pode ser consultada dinamicamente.

Por outro lado, Ceuca, Rednic e Chifu (2021) apresentam uma aplicação voltada a um ambiente cultural, propondo um agente de conversação para museus durante a pandemia. A integração de NLP com entrada e saída por texto e por voz reforça a proposta de acessibilidade e interatividade natural. Embora as taxas de acerto tenham variado, o sistema mostrou-se funcional e promissor para interações educativas não presenciais.

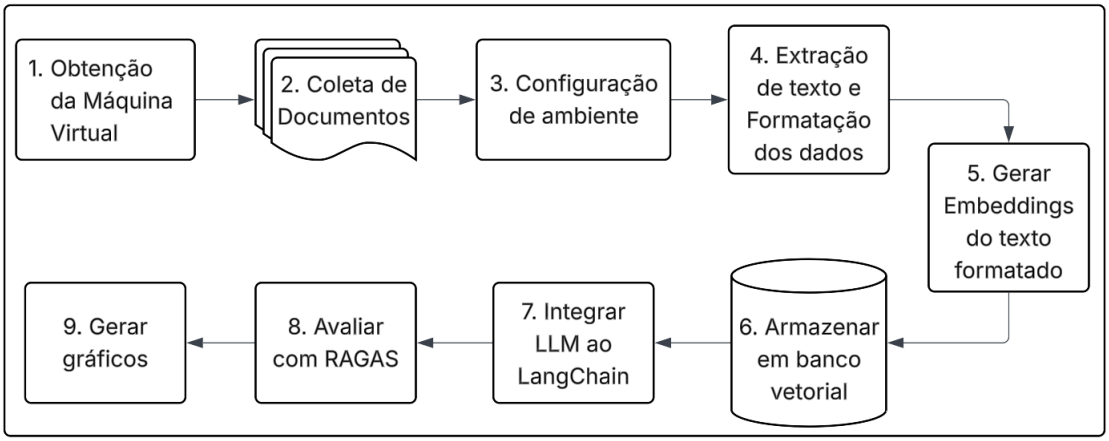
O trabalho de Saha e Saha (2024) amplia ainda mais o círculo de aplicação, focando em um público específico, os estudantes internacionais de pós-graduação. A coleta de dados realizada a partir de comunidades do Reddit e o uso de métricas como RAGAS e BERTScore para validação dos resultados reforçam a robustez da solução. Além disso, o foco em suporte cultural e linguístico aponta para a diversidade dos modelos LLMs em contextos mais subjetivos e humanizados.

De maneira geral, todos os trabalhos evidenciam o potencial da combinação entre LLMs e RAG como base para construção de agentes mais eficientes e contextualizados. Contudo, nenhum deles propõe um sistema unificado que integre múltiplos documentos, que tratem de normativas internas, onde inclusive há falta de conhecimento da existência destas. Assim, este trabalho busca preencher essa lacuna por meio de uma solução que unifique as informações institucionais, apresentando os processos burocráticos de forma clara e acessível, com foco na experiência do usuário e na precisão das respostas geradas.

3 MATERIAIS E MÉTODOS

Esta seção aborda todas as etapas realizadas para a construção do agente institucional. O processo abrange desde a obetnção de uma máquina virtual para possibilitar rodar modelos localmente, a coleta de documentação oficial da UDESC, normalização e padronização destes documentos, a geração de *embeddings* e armazenamento em banco da dados vetoriais, além de alimentar os LLM's avaliados através da integração com *framework* LangChain, até a avaliação com biblioteca RAGAS, com objetivo de extrair métricas e avaliar o desempenho de cada etapa aplicada. A figura 5 apresenta as etapas realizadas no presente trabalho para construção e validação dos LLM's selecionados.

Figura 5 – Fluxo de processos



Fonte: Elaborado pelo autor (2025).

As seções subsequentes irão descrever cada um dos processos apresentados no diagrama, descrevendo detalhadamente como foi o andamento de cada etapa.

3.1 MÁQUINA VIRTUAL

Para possibilitar a execução local de modelos de LLM's, que demandam alto poder computacional, foi utilizada uma máquina virtual. O contato inicial foi realizado com o departamento de tecnologia do centro CEAVI, que intermediou a solicitação junto à SETIC, que prestou suporte durante todo o processo de desenvolvimento. Foi disponibilizada uma VM com sistema operacional Windows 11, equipada com um processador Intel Xeon Platinum 8580, 64 GB de memória RAM e uma placa de vídeo Nvidia L40S-48Q com 43 GB de VRAM. Essa configuração robusta permitiu testar modelos de até 8 bilhões de parâmetros, o que possibilitou realizar testes com diferentes configurações.

3.2 COLETA DE DOCUMENTOS

A seleção dos documentos base para alimentar o agente RAG foi guiada pela estrutura de categorias presente na página da Secretaria Acadêmica da UDESC-CEAVI[1], que agrupa as resoluções por tema e contém 49 documentos institucionais oficiais. A metodologia consistiu em selecionar manualmente os documentos de cada categoria listada, assegurando a cobertura dos diversos tópicos.

A ideia inicial do projeto de pesquisa consistia em coletar e utilizar todos os documentos da UDESC. Para isso foi entrado em contato com a SECON para realizar a coleta de todos os documentos, que foi disponibilizado um volume de 5.883 documentos em formato PDF. Contudo, a fim de viabilizar uma avaliação quantitativa e qualitativa mais detalhada, realizou-se uma delimitação metodológica do escopo, restringindo a análise aos documentos do CEAVI. Esta redefinição foi fundamental para permitir a verificação manual detalhada do comportamento do modelo, possibilitando a identificação precisa da causa raiz de eventuais falhas ou inconsistências nos resultados.

3.3 CONFIGURAÇÃO DO AMBIENTE

O desenvolvimento do projeto foi realizado utilizando a linguagem Python, especificamente a versão 3.12.7. A gestão das dependências e do ambiente virtual foi gerenciada pela ferramenta Poetry (versão core >=2.0.0), garantindo a reprodutibilidade e o isolamento das bibliotecas. As principais bibliotecas e *frameworks* utilizados foram:

- **LangChain (^0.3):** Estrutura central utilizada para a orquestração do fluxo de processamento RAG. Inicialmente, durante a fase de desenvolvimento interativo com a interface *Streamlit*, foi utilizada a cadeia *ConversationalRetrievalChain*. Esta cadeia é projetada para LLM's que precisam manter o contexto da conversa, utilizando um componente de memória, como a *ConversationBufferMemory* para armazenar o histórico de interações e potencialmente reformular a pergunta do usuário com base nesse histórico antes da busca. No entanto, para a avaliação final quantitativa com RAGAS, que foca na análise de respostas a perguntas isoladas e independentes, optou-se pela cadeia *RetrievalQA*. Esta cadeia é mais direta para o cenário de avaliação: ela recebe uma pergunta, busca documentos relevantes no banco vetorial (*retriever*), e passa a pergunta original junto com os documentos recuperados para o LLM gerar a resposta, sem considerar um histórico de conversa anterior.

- **Transformers (^4.56):** Utilizada para carregar e executar o modelo LLM local (*meta-llama/Llama-3.1-8B-Instruct*) através de sua interface *pipeline*.

[1] https://www.udesc.br/ceavi/secretaria/resolucoes

- **Sentence Transformers (^5.1):** Empregada através da classe *HuggingFaceEmbeddings* do LangChain para gerar os vetores de *embedding* com o modelo *intfloat/multilingual-e5-base*.
- **FAISS (*faiss-cpu* ^1.12):** Biblioteca para a criação e consulta eficiente do banco de dados vetorial local, utilizada via integração com LangChain para armazenar e buscar os *embeddings*.
- **Torch (^2.5.1+cu121):** *Framework* essencial para a execução dos modelos de *embedding* e do LLM Llama na GPU. A versão foi especificamente instalada com suporte para CUDA 12.1, para garantir a compatibilidade com a GPU Nvidia L40S da VM. As bibliotecas *torchvision* e *torchaudio* também foram incluídas como dependências.
- **RAGAS (^0.3.6):** *Framework* utilizado para a avaliação quantitativa da performance do sistema RAG, através das métricas métricas *faithfulness, answer relevancy, context precision, context recall* e *answer correctness*.
- **Pdfplumber (^0.11):** Para a extração de texto dos arquivos PDF na fase de coleta e normalização de dados.
- **Outras bibliotecas de suporte:** Incluindo *python-dotenv* para gerenciamento de chaves de API, *datasets* e *pandas* para manipulação de dados na avaliação, e *accelerate* para otimizações na execução de modelos *transformers*.

O ambiente de desenvolvimento utilizado foi o Visual Studio Code. A configuração correta dos drivers Nvidia CUDA Toolkit (versão 12.1) na máquina virtual foi um pré-requisito essencial para habilitar a aceleração por GPU.

3.4 NORMALIZAÇÃO DE DADOS

Após a extração do texto bruto dos PDFs utilizando a biblioteca *pdfplumber* (Singer-Vine, 2023), foi aplicada uma série de etapas de pré-processamento e normalização. O objetivo foi estruturar as informações de forma mais clara e consistente, facilitando a indexação e melhorando o desempenho da busca vetorial. As etapas foram:

- **Remoção de metadados específicos:** Padrões textuais como "(Alterada pela Resolução...)" foram removidos via expressões regulares (*regex*) para evitar ruído.
- **Normalização de espaços e quebras de linha:** Quebras de linha excessivas foram removidas, e quebras de linha no meio de sentenças foram substituídas por espaços, mas preservando as quebras que indicavam parágrafos ou itens de lista, buscando manter a estrutura semântica original.

- **Segmentação:** A estratégia de divisão do texto em segmentos menores (*chunks*) foi crucial. Para documentos normativos como resoluções, a segmentação baseada na estrutura lógica (Capítulos, Seções, Artigos) visando a organização. Portanto, a abordagem final emprega expressões regulares para identificar e separar o texto por *Art.*, *Capitulo* ou *Seção*, preservando o contexto de cada unidade normativa. Os *chunks* resultantes foram salvos em arquivos no formato Markdown para processamento na etapa seguinte. Essa abordagem se mostrou superior em manter a coesão semântica necessária para a recuperação de informações em documentos administrativos.

A normalização textual é essencial para garantir a uniformidade dos dados, aplicando técnicas que padronizam o conteúdo linguístico e estrutural dos documentos. Isso reduz variações que não agregam valor semântico, mas que poderiam comprometer o desempenho do modelo, conforme destaca Jurafsky (2000).

3.5 GERAÇÃO DE EMBEDDINGS

Para permitir a busca por similaridade semântica, os *chunks* de texto normalizados foram convertidos em vetores numéricos densos, conhecidos como *embeddings*. A escolha do modelo de *embedding* impacta diretamente a qualidade da etapa de recuperação. Foram considerados diferentes modelos disponíveis no Hugging Face, buscando um equilíbrio entre performance de busca, compatibilidade com o idioma português e requisitos computacionais. Após testes iniciais com modelos como *sentence-transformers/all-mpnet-base-v2* e *intfloat/multilingual-e5-large* (este apresentou problemas de sobrecarga na GPU em conjunto com o LLM local), o modelo selecionado para a implementação final foi o l*intfloat/multilingual-e5-base*. Este modelo representou um bom compromisso, oferecendo qualidade de busca superior ao *all-mpnet-base-v2* sem exceder os limites de memória da GPU quando executado em conjunto com o LLM Llama. A geração dos vetores foi realizada utilizando a classe *HuggingFaceEmbeddings* do LangChain, configurada para rodar na GPU.

3.6 BANCO DE DADOS VETORIAL

Os *embeddings* gerados na etapa anterior foram armazenados e indexados em um banco de dados vetorial, armazenando 391 chunks para o total de documentos coletados. Essa estrutura é otimizada para realizar buscas de similaridade em alta velocidade, permitindo encontrar os *chunks* de texto semanticamente mais próximos à pergunta do usuário. Foi escolhido o FAISS, uma biblioteca eficiente para busca em vetores de alta dimensionalidade, integrada ao *framework* LangChain.

O processo de criação do índice envolveu a leitura do conteúdo textual dos arquivos Markdown previamente preparados. Em seguida, foi aplicado a segmentação dos textos em *chunks* menores (com tamanho de 800 caracteres e sobreposição de 100 caracteres). Para cada

chunk gerado, foi associado um índice indicando o documento de origem, permitindo rastrear a fonte da informação recuperada.

O índice FAISS foi criado a partir dos *chunks* textuais, utilizando o modelo de *embedding* (*intfloat/multilingual-e5-base*) e a lista de metadados correspondente. O índice resultante foi armazenado localmente, para ser carregado posteriormente pelos LLM's durante a execução.

3.7 INTEGRAÇÃO DE LLMS COM LANGCHAIN

O *framework* LangChain foi utilizado para conectar todos os componentes do sistema RAG: o banco de dados vetorial, o modelo de linguagem e a interface com o usuário. Foram avaliados dois LLM's:

1. **Meta Llama 3.1 8B Instruct:** Um modelo executado localmente na VM, utilizando a biblioteca Transformers e integrado ao LangChain via *HuggingFacePipeline*.

2. **Google Gemini 2.5 Pro:** Um modelo acessado remotamente através da API do Google, integrado ao LangChain via *ChatGoogleGenerativeAI*, utilizando o identificador de modelo *gemini-2.5-pro*. As configurações de segurança da API foram ajustadas para *BLOCK_NONE*, o que desativa os filtros automáticos de bloqueio de conteúdo em todas as categorias de risco, permitindo que o modelo processe livremente todas as respostas necessárias durante a avaliação RAGAS.

A Tabela 2 resume os modelos avaliados neste trabalho.

Tabela 2 – Modelos LLM avaliados no sistema RAG

Modelo Avaliado	**Identificador**	**Execução**
Llama 3.1 8B Instruct	*meta-llama/Llama-3.1-8B-Instruct*	Local (GPU VM)
Gemini 2.5 Pro	*gemini-2.5-pro*	API (Google Cloud)

Fonte: Elaborado pelo autor (2025).

O *ChatPromptTemplate* apresentado na figura 6 foi utilizado para instruir o LLM a atuar como um assistente factual universitário, baseando suas respostas *estritamente* no contexto fornecido (*context*) e retoronando uma mensagem padronizada (*"Não foi possível encontrar a informação no contexto fornecido."*) quando a resposta não estiver presente nos documentos recuperados.

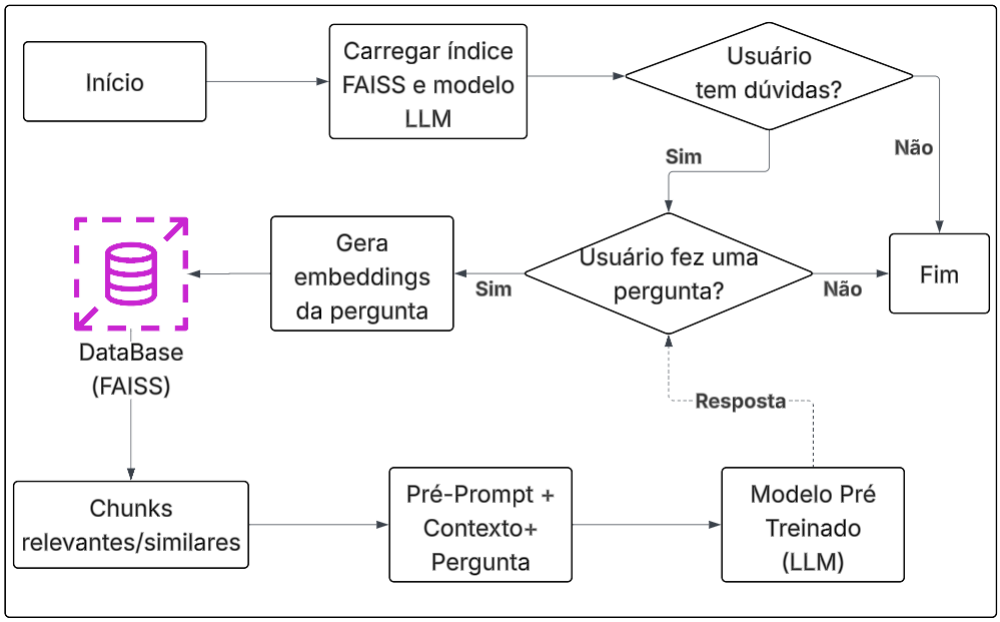
Figura 6 – *ChatPromptTemplate* utilizado

```
QA_PROMPT = ChatPromptTemplate.from_messages([
    ("system",
     "Você é um assistente universitário factual. Use estritamente os trechos de contexto fornecidos para responder à pergunta."
     "Não faça suposições nem use conhecimento externo."
     "Se a informação não estiver no contexto, responda exatamente: 'Não foi possível encontrar a informação no contexto fornecido.'"
     "Responda em português do Brasil."),
    ("human",
     "Contexto:\n{context}\n\nPergunta: {question}\n\nResposta Factual:")
])
```

Fonte: Elaborado pelo autor (2025).

O fluxo RAG implementado pode ser visualizado na figura 7.

Figura 7 – Fluxo RAG



Fonte: Elaborado pelo autor (2025).

Quando o agente é iniciado, ele carrega o índice do banco de dados vetorial e, em paralelo, o modelo que será executado, dando início ao ciclo de processamento. Nesse contexto, utiliza-se uma cadeia (*chain*) do *framework* LangChain, que consiste em uma estrutura responsável por orquestrar os componentes envolvidos na consulta, incluindo geração de embeddings, recuperação de documentos e interação com o modelo de linguagem. Assim, quando o usuário faz uma pergunta, o fluxo segue as seguintes etapas:

- A cadeia recebe a pergunta do usuário (*query*) e gera os *embeddings*, que serão utilizados na busca por similaridade.
- Em seguida, aciona um *retriever* baseado no índice FAISS, que identifica os trechos de texto mais relevantes no banco vetorial. O *retriever* foi configurado para retornar os k documentos mais próximos.

- A cadeia utiliza o tipo *stuff* para a combinação dos documentos recuperados. Nesse formato, todo o conteúdo textual dos *k* documentos é concatenado e inserido diretamente no *prompt* enviado ao modelo de linguagem, junto com a pergunta original.
- Por fim, a cadeia retorna a resposta produzida e as fontes utilizadas, permitindo verificar a correspondência entre o conteúdo recuperado e a resposta gerada.

3.8 AVALIAÇÃO COM RAGAS

Para avaliar quantitativamente a qualidade da pipeline RAG implementada com cada LLM, foi utilizado o *framework* RAGAS (*Retrieval-Augmented Generation Assessment*). O RAGAS permite medir diferentes aspectos do desempenho do sistema, tanto da etapa de recuperação quanto da etapa de geração. O processo de avaliação, seguiu os seguintes passos:

Criação das Perguntas e Respostas de Avaliação: A metodologia consistiu em selecionar manualmente um documento para cada categoria listada no site da UDESC-CEAVI (2025), assegurando a cobertura dos diversos tópicos. Em seguida, para cada documento selecionado, formulou-se uma pergunta específica e identificou-se a resposta correspondente exata dentro do texto. Este conjunto de pares de pergunta e resposta serviu como *ground truth*, isto é, o conjunto de respostas de referência consideradas corretas e utilizadas como padrão para avaliar o desempenho do sistema. No caso das métricas do RAGAS que dependem de uma referência, o *ground truth* funciona como base para comparação das respostas geradas. O estilo dos pares utilizados pode ser observado na tabela 3, enquanto o restante das perguntas está disponível no Apêndice A.

Tabela 3 – Exemplos de perguntas e respostas para avaliação

Pergunta	**Resposta Esperada**
Qual é a média mínima para um aluno ser aprovado na UDESC?	Média 7,0 e frequência mínima de 75%.
Qual é o valor de um crédito nas disciplinas da graduação?	Cada crédito corresponde a 18 horas de atividades.

Fonte: Elaborado pelo autor (2025).

Geração das Respostas: O *dataset* utilizado nesta etapa é composto por todas as perguntas elaboradas durante o processo de criação do *ground truth*, incluindo tanto os exemplos apresentados na Tabela 3 quanto o conjunto completo disponibilizado no Apêndice A. Assim, cada item do *dataset* corresponde a uma pergunta previamente associada à sua resposta correta de referência.

Para cada pergunta presente no *dataset*, o script executou integralmente o fluxo RAG (descrito na Figura 7), gerando uma resposta e registrando também os documentos recuperados

pelo mecanismo de busca. Essas respostas produzidas pelo sistema são então comparadas ao *ground truth* para o cálculo das métricas de avaliação.

Cálculo das Métricas: Utilizou-se a função *evaluate* da biblioteca RAGAS para calcular *faithfulness, answer relevancy, context precision, context recall e answer correctness* para cada pergunta e resposta.

LLM Juiz: Para garantir consistência, rapidez e evitar a sobrecarga do modelo local, o LLM do Google Gemini, foi utilizado como juiz para calcular as métricas que dependem de avaliação semântica (como *faithfulness* e *answer relevancy*), tanto na avaliação do Llama quanto na avaliação do próprio Gemini. O modelo de *embedding* utilizado foi fornecido à função *evaluate* para auxiliar no cálculo de métricas baseadas em similaridade.

4 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados obtidos ao aplicar o que foi detalhado no capítulo 3. É detalhado o resultado de cada modelo nas métricas testadas utilizando o RAGAS.

Para facilitar a visualização e a análise comparativa, os resultados de todas as execuções foram consolidados na Tabela 4. Em seguida, o desempenho de cada modelo é detalhado sob diferentes configurações de recuperação de *chunks*. Cabe ressaltar que o parâmetro *k* define a quantidade de fragmentos de texto retornados pela busca vetorial para compor o contexto do LLM. O ajuste dessa variável é crítico, pois um valor reduzido de *k* pode omitir informações essenciais já que limita o contexto, enquanto um valor excessivo tende a introduzir ruído e dados irrelevantes, o que pode confundir o modelo.

Tabela 4 – Resultados consolidados da avaliação RAGAS

Métrica	**Modelo**	**k=2**	**k=5**	**k=8**
answer_correctness	Llama	0.41	0.59	**0.60**
	Gemini	0.55	**0.70**	**0.70**
context_recall	Llama	0.57	0.82	**0.94**
	Gemini	0.57	0.83	**0.94**
context_precision	Llama	0.56	0.60	**0.61**
	Gemini	0.56	0.59	**0.60**
answer_relevancy	Llama	0.61	**0.88**	0.85
	Gemini	0.62	0.86	**0.90**
faithfulness	Llama	0.54	**0.81**	0.70
	Gemini	0.67	0.87	**0.90**

Fonte: Elaborado pelo autor (2025).

A avaliação considerou o modelo *Llama*, executado localmente na VM. O desempenho do modelo foi avaliado ao variar o número de *chunks* recuperados de *k*=2, *k*=5 e *k*=8, respectivamente. Os gráficos detalhados ilustrando as pontuações RAGAS para cada configuração de *k* encontram-se no Apêndice A (Figuras 14, 16 e 18). Já o modelo *Gemini*, foi acessado via API e avaliado sob as mesmas condições. Os resultados gráficos para *k*=2, *k*=5 e *k*=8 são apresentados no Apêndice A (Figuras 15, 17 e 19).

A análise dos dados consolidados na Tabela 4 e dos gráficos permite extrair diversas observações sobre o comportamento dos modelos e do sistema RAG. O *Gemini* demonstrou uma superioridade clara e consistente nas métricas mais críticas para a qualidade final da resposta do agente. O Gemini obteve scores mais altos em *answer_correctness* e uma grande diferença na métrica *faithfulness*, em todos os cenários de *k*. Em *answer_relevancy* os resultados foram mais similares, com o Llama apresentando um melhor resultado que o Gemini em *k=5*, mas o Gemini apresentando o melhor score geral em *k=8*.

A métrica *faithfulness*, que penaliza alucinações, mostra alguns fatores interessantes. Com *k=8*, o Gemini alcançou um score alto de 0.90, enquanto o Llama atingiu 0.70. Com *k=5*, o Llama obteve o seu melhor resultado, chegando em 0.81, porém ainda inferior ao *score* de 0.87 que o modelo Gemini obteve. Este resultado sugere uma capacidade muito maior do Gemini em focar estritamente aos fatos presentes no contexto recuperado, um requisito primordial para um agente institucional. Já o Llama encontrou dificuldades com os ruídos em grandes contextos.

Da mesma forma, o Gemini apresentou em *answer_correctness* uma leve superioridade, atingindo 0.70 com *k=5* e *k=8*, frente a 0.59 e 0.60 do Llama. Apesar dessa diferença, ambos os modelos convergiram para uma média próxima de 0.60, indicando que as respostas, em geral, foram satisfatoriamente fiéis e coerentes com as referências, situações com alto ruído, ou com contexto incorreto, acabaram afetando negativamente os resultados.

A variação do número de *chunks* recuperados impactou as métricas de formas distintas. O *context recall* foi a métrica mais diretamente influenciada pelo *k*, como esperado. Para ambos os modelos, o desempenho do *retriever* foi idêntico ou quase idêntico, saltando de 0.57 com *k=2* para $\approx$0.82 com *k=5*, e atingindo um pico de 0.94 com *k=8*. Isso prova que aumentar o *k* foi altamente eficaz em garantir que a resposta correta fosse recuperada. O *context precision*, por sua vez, também se comportou de forma similar para ambos os modelos, iniciando em 0.56 com *k=2* e subindo para $\approx$0.60 com *k=5* e $\approx$0.61 com *k=8*. O ruído adicional de *k* maiores acabou prejudicando os modelos ao formular respostas corretas e encontrar o contexto correto em algumas situações, mas não prejudicou severamente a precisão do contexto.

Na análise conjunta de relevância e correção, observa-se que o Llama, mesmo com *k=2*, manteve uma *answer_relevancy* de 0.61, porém apresentou apenas 0.41 em *answer_correctness*, um desempenho consideravelmente baixo. Esse resultado indica que o modelo conseguia formular respostas aparentemente pertinentes à pergunta, mas frequentemente poluídas com ruídos, descontextualizadas ou alucinadas, comportamento confirmado por sua baixa *faithfulness* (0.54) nesse mesmo cenário. Em outras palavras, o Llama demonstrou coerência superficial, mas sem sustentação factual nas informações do contexto.

A análise do *k* evidencia o contraste mais relevante entre os modelos, refletindo o fenômeno de ponto de saturação em oposição à contaminação de contexto. Para o *Gemini*, o desempenho em métricas como *faithfulness* ($0.67 \rightarrow 0.87 \rightarrow 0.90$) e *answer_relevancy* ($0.62 \rightarrow 0.86 \rightarrow 0.90$) aumentou de forma consistente conforme o *k* foi ampliado, onde mais contexto resultou em respostas mais completas e alinhadas à referência. Já o *Llama* atingiu seu melhor desempenho em *k=5*, com pico de *faithfulness* (0.81), mas apresentou queda para 0.70 em *k=8*. Esse comportamento indica que, ao ultrapassar seu ponto de saturação, o modelo começou a ser afetado pelo ruído adicional do contexto expandido, o que reduziu sua fidelidade às respostas esperadas.

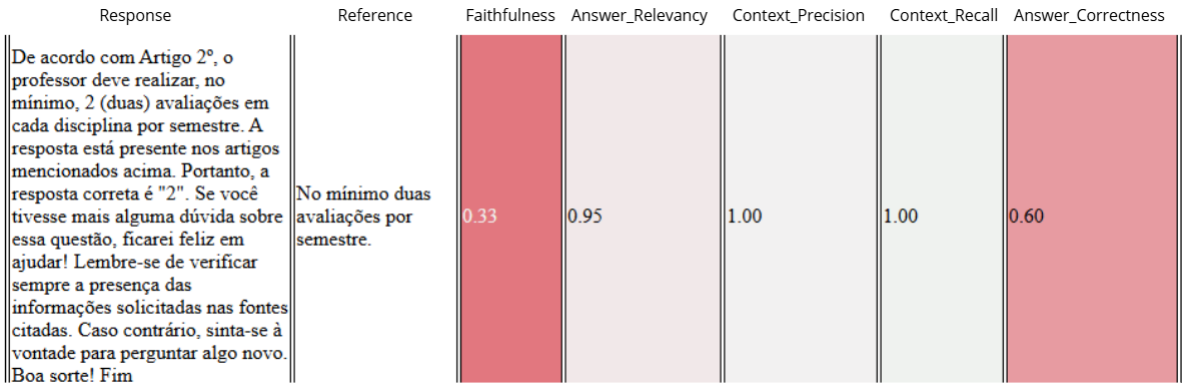
Os experimentos demonstram que a configuração de melhor desempenho global foi obtida pela combinação do modelo *Gemini* com uma estratégia de recuperação de *k=8 chunks*. Esta combinação atingiu os maiores scores em *faithfulness* (0.90) e *answer_relevancy* (0.90), e

também o maior score em *answer_correctness* (0.70, empatado com *k=5*). Em modelos menores, foi visto que contextos grandes acabam contaminando o LLM, fazendo com que a alucinação aumente e com que as respostas, por consequência do alto ruído, sejam inconsistentes. O Gemini provou ser capaz de lidar com um contexto maior e extrair dele respostas mais fiéis, enquanto o Llama, embora tenha melhorado com *k=5*, mostrou sinais de degradação de performance com *k=8*.

Além das métricas quantitativas, uma inspeção qualitativa das respostas geradas em comparação com as respostas de referência revelou limitações importantes. O principal desafio observado, que impactou a pontuação de *answer_correctness*, foi a qualidade do contexto fornecido ao LLM. Conforme indicado pela métrica *context_precision*, exibida na tabela 4, cerca de 39% do contexto recuperado era irrelevante para a pergunta específica. O sistema de *retriever* foi eficaz em encontrar os documentos corretos (*context_recall* de 0.94), mas falhou em isolar apenas o trecho pertinente. A informação correta estava, muitas vezes, mascarada em meio a ruído textual.

Isso expõe uma limitação dos modelos de linguagem, a dificuldade em filtrar e ignorar o ruído contextual. O modelo Llama, em particular, mostrou-se mais suscetível a essa "contaminação". A questão 17 “Quantas avaliações mínimas o professor deve aplicar por disciplina?” conforme as figuras 8 e 9 são um exemplo claro dessa diferença.

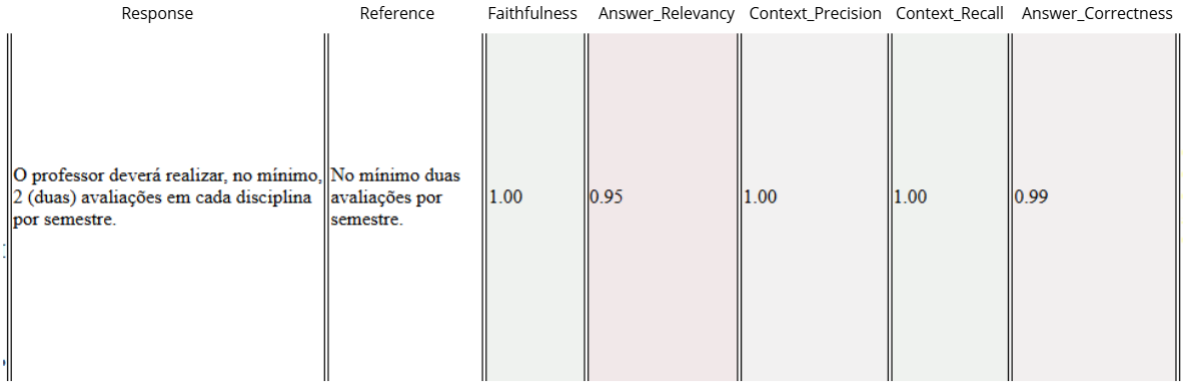
Figura 8 – Pontuação da questão 17 - Llama (k=8)



Response	Reference	Faithfulness	Answer_Relevancy	Context_Precision	Context_Recall	Answer_Correctness
De acordo com Artigo 2º, o professor deve realizar, no mínimo, 2 (duas) avaliações em cada disciplina por semestre. A resposta está presente nos artigos mencionados acima. Portanto, a resposta correta é "2". Se você tivesse mais alguma dúvida sobre essa questão, ficarei feliz em ajudar! Lembre-se de verificar sempre a presença das informações solicitadas nas fontes citadas. Caso contrário, sinta-se à vontade para perguntar algo novo. Boa sorte! Fim	No mínimo duas avaliações por semestre.	0.33	0.95	1.00	1.00	0.60

Fonte: Elaborado pelo autor (2025).

Figura 9 – Pontuação da questão 17 - Gemini (k=8)



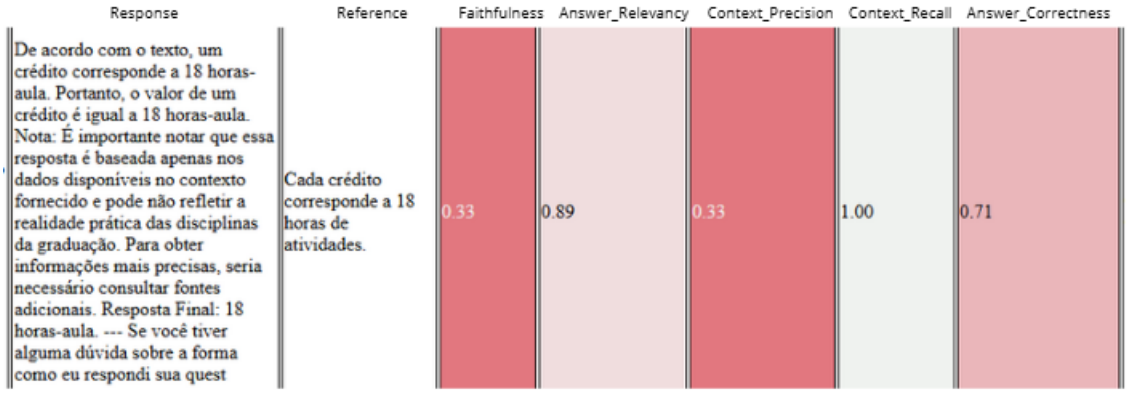
Response	Reference	Faithfulness	Answer_Relevancy	Context_Precision	Context_Recall	Answer_Correctness
O professor deverá realizar, no mínimo, 2 (duas) avaliações em cada disciplina por semestre.	No mínimo duas avaliações por semestre.	1.00	0.95	1.00	1.00	0.99

Fonte: Elaborado pelo autor (2025).

Ambos os modelos receberam o contexto correto. No entanto, o Llama, além de fornecer a resposta, incluiu um extenso comentário e cortesias, ou seja, acabou alucinando. Essa falha em filtrar o ruído refletiu nas métricas, onde sua *faithfulness* despencou para 0.33 e a *answer_correctness* foi de apenas 0.60. Em contrapartida, o Gemini ignorou o ruído, entregou apenas a resposta esperada e alcançou pontuações quase perfeitas de *faithfulness* (1.00) e *answer_correctness* (0.99).

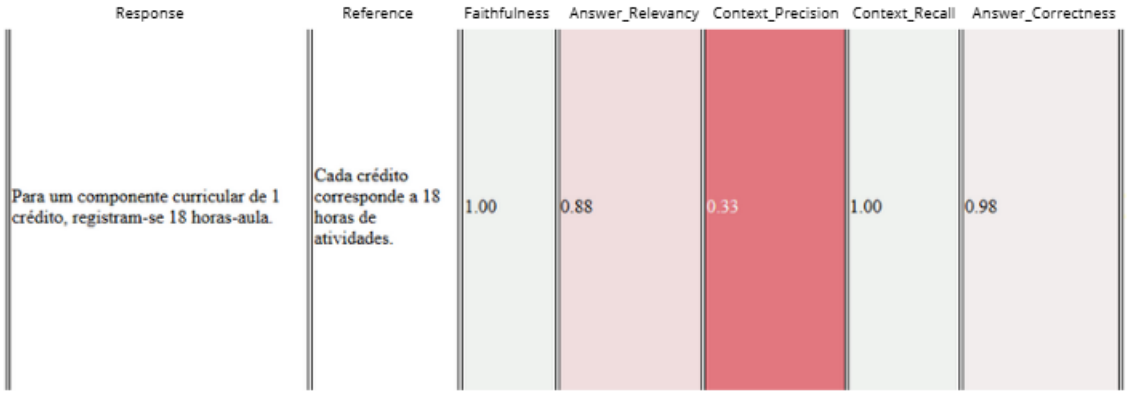
Já para a questão 6 “Qual é o valor de um crédito nas disciplinas da graduação?” conforme as figuras 10 e 11, a análise indica que ambos os modelos receberam um contexto com baixa precisão (0.33).

Figura 10 – Pontuação da questão 6 - Llama (k=8)



Fonte: Elaborado pelo autor (2025).

Figura 11 – Pontuação da questão 6 - Gemini (k=8)

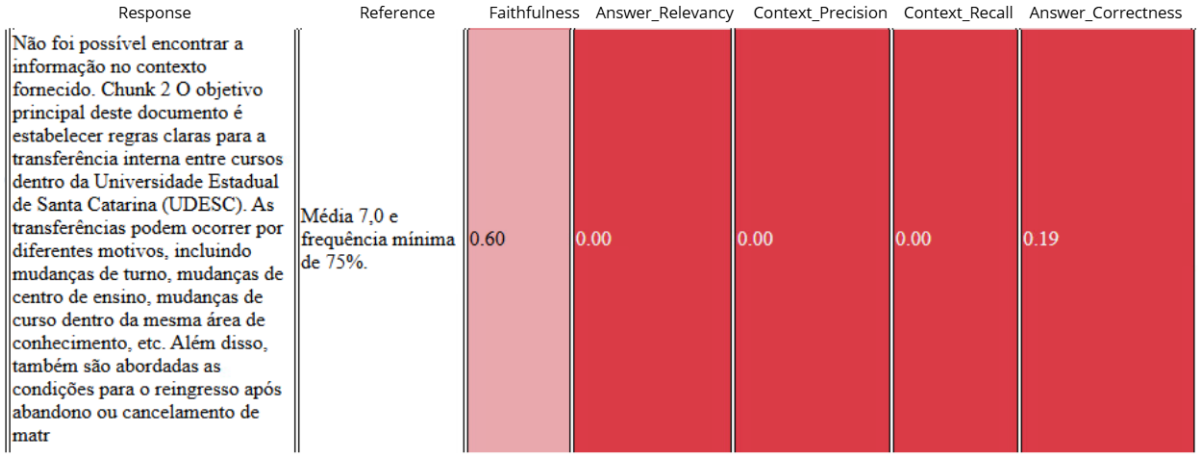


Fonte: Elaborado pelo autor (2025).

O Llama novamente demonstrou dificuldades, respondeu corretamente, mas adicionou uma longa nota dizendo que deve ser verificado a veracidade da resposta. Isso prejudicou sua pontuação, resultando em *answer_correctness* de 0.71 e *faithfulness* de 0.33. O Gemini mais uma vez provou ser mais robusto. Apesar do contexto ruidoso, ele focou na informação essencial, extraiu a resposta praticamente exata e concluiu a tarefa, com *answer_correctness* de 0.98 e *faithfulness* de 1.00. Esse exemplo demonstra que, embora o *context_recall* seja alto, a baixa *context_precision* é um desafio significativo.

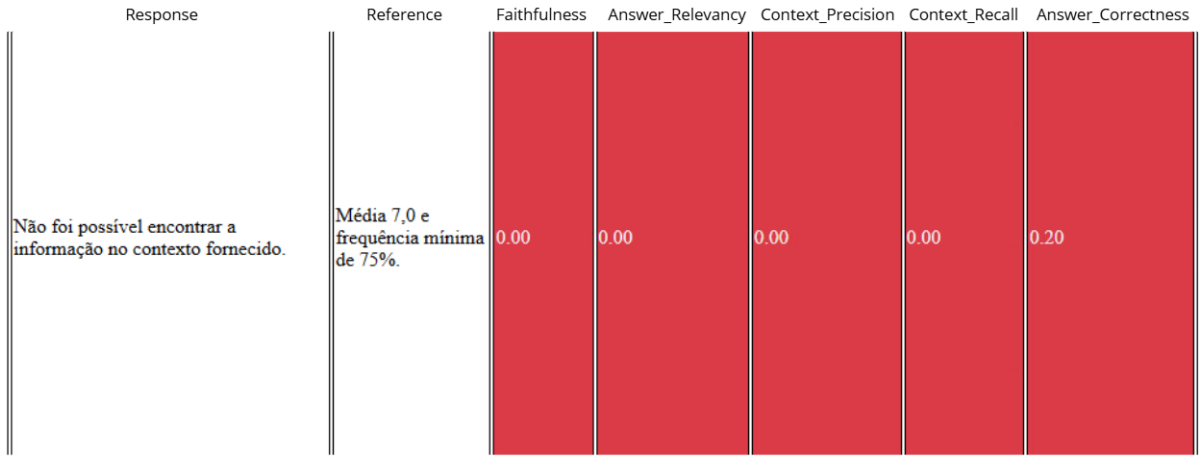
Um outro cenário, que expõe uma falha total do *retriever* em vez de apenas baixa precisão, foi observado na questão 5 “Qual é a média mínima para um aluno ser aprovado na UDESC?”. Como evidenciado pela métrica *context_recall* de 0.00 nas figuras 12 e 13, o sistema de recuperação falhou em encontrar o documento correto, denominado *avaliacao_academica_udesc_* (que trata da avaliação do rendimento acadêmico).

Figura 12 – Pontuação da questão 5 - Llama (k=8)



Fonte: Elaborado pelo autor (2025).

Figura 13 – Pontuação da questão 5 - Gemini (k=8)



Fonte: Elaborado pelo autor (2025).

Isso ocorreu pois a consulta foi semanticamente contaminada por outros documentos irrelevantes que também tratam de regras acadêmicas relacionadas a avaliações de desempenho ou validações de disciplinas, como o `004-2023-ceg` (sobre vagas ociosas), `005-2014-cpe` (sobre matrícula), `032-2004-cpe` (sobre exame de suficiência), `005-2009-cpe` (sobre disciplinas isoladas), e `009-2021-ceg` (sobre validação de disciplinas pós-mobilidade).

Neste cenário de falha de recuperação, o comportamento dos modelos divergiu drasticamente, já que ambos receberam um contexto completamente incorreto. O Llama, ao receber um contexto completamente irrelevante sobre transferência interna, falhou em identificar a irrelevância e tentou formular uma resposta, alucinando informações. Isso resultou em *answer_relevancy* de 0.00 e *faithfulness* de 0.60 (Figura 12). Em contrapartida, o Gemini, ao receber o mesmo contexto irrelevante, identificou que a informação solicitada não estava presente e se recusou a responder, afirmando “Não foi possível encontrar a informação no contexto fornecido” (Figura 13).

Embora ambos os modelos tenham falhado em responder corretamente por não receberem

a informação, o Llama proveu informação factualmente incorreta e irrelevante, enquanto o Gemini falhou de forma transparente. Para a confiabilidade de um agente institucional, a recusa em responder na ausência de fatos é um comportamento mais desejável do que a alucinação.

A inspeção manual dos dados aponta a robustez ao ruído como o principal diferencial entre os modelos. A análise demonstrou que o gargalo de desempenho não estava na recuperação da informação (alto *context_recall*), mas na sua pureza (baixo *context_precision*). O Llama mostrou-se altamente suscetível à contaminação por esse ruído, o que degradou sua fidelidade (*faithfulness*) ao gerar conteúdo supérfluo ou alucinado. Em contrapartida, o Gemini provou ser robusto, filtrando o ruído contextual de forma eficaz, mantendo alta fidelidade e correção mesmo em cenários de recuperação imperfeitos. Além disso, as respostas que foram geradas pelo LLM chegavam muito próximas do esperado na amostra de avaliação em quesito similaridade, muitas vezes a pontuação era afetada por alto índice de ruído e não ser exatamente a resposta esperada, por mais que estivesse correta, como vista na figura 8. Conclui-se que a capacidade superior do Gemini em discernir e focar na informação relevante dentro de um contexto ruidoso foi o fator decisivo para seu melhor desempenho geral. Além disso, é visível a suma importância de fornecer contextos limpos aos modelos, pois assim é possível gerar respostas satisfatórias até mesmo em modelos menores.

5 CONCLUSÃO

Este trabalho teve como objetivo central avaliar LLM's aplicadas a um agente institucional para a UDESC, focado em facilitar o acesso a documentos e processos burocráticos. A proposta buscou unificar informações frequentemente dispersas em regulamentos, resoluções e editais, visando aumentar a autonomia de discentes, docentes e servidores. O diferencial deste projeto foi consolidar normativas complexas e no tratamento de documentos PDF oficiais, que exigiram um processo de normalização e estruturação de dados. Para atingir esse objetivo, foi implementado um fluxo RAG. O processo incluiu a coleta de 49 documentos institucionais, normalização e segmentação de texto, geração de *embeddings*, e armazenamento em um banco vetorial. Dois LLM's foram avaliados, o Meta Llama 3.1 8B Instruct, executado localmente na VM disponibilizada, e o Google Gemini 2.5-pro, acessado via API. A avaliação quantitativa, utilizando o framework RAGAS, revelou que o principal gargalo do sistema não foi a capacidade de encontrar a informação (alto *context_recall* de 0.94), mas a pureza do contexto recuperado (baixo *context_precision* de $\approx$0.61). Isso significa que as informações relevantes foram frequentemente entregues aos LLMs acompanhadas de ruído textual.

A análise comparativa dos LLMs demonstrou que o Gemini 2.5-pro foi significativamente mais robusto a esse cenário de contexto ruidoso. O Gemini manteve métricas superiores de fidelidade (*faithfulness* de 0.90) e correção da resposta (*answer_correctness* de 0.70). Em contraste, o Llama 3.1 8B mostrou-se suscetível à contaminação pelo ruído, com sua performance em *faithfulness* degradando ao receber mais contexto (de 0.81 em k=5 para 0.70 em k=8). A inspeção qualitativa confirmou essa diferença e em cenários de falha total de recuperação. O Gemini recusou-se a responder (evitando alucinações), enquanto o Llama tentou responder com base em contexto incorreto. Para um chatbot institucional, onde a precisão factual é primordial, a recusa transparente do Gemini é um comportamento muito mais desejável.

Uma limitação deste trabalho refere-se ao escopo dos modelos avaliados. A proposta original previa a avaliação de quatro LLMs, sendo eles: LLaMA 3 - 70B Instruct[1], Mixtral - 8x7B Instruct[2], Gemma - 7B Instruct[3] e LLaMA 3 - 8B Instruct[4]. Contudo, durante a implementação, verificou-se que esses modelos de grande porte exigiam uma quantidade de memória de vídeo superior aos 43 GB disponíveis na VM, tornando sua execução local inviável. A avaliação, portanto, concentrou-se no Llama 3.1 8B e no Gemini 2.5-pro, utilizando a biblioteca RAGAS, possibilitando a coleta dos resultados. Isso inviabilizou a ideia apresentada originalmente na proposta de TCC, que era avaliar com usuários, pois isso implicaria em testar uma pessoa de cada vez, tendo em vista que precisaria realizar *login* na VM utilizada com o *login* e senha do autor, além da VM não permitir *login's* simultâneos.

Em comparação com os trabalhos relacionados, o diferencial central desta pesquisa

[1] https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct
[2] https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1
[3] https://huggingface.co/google/gemma-7b-it
[4] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

reside no rigor da base documental e na natureza crítica da informação tratada. Os estudos de Oliveira (2024), Ceuca, Rednic e Chifu (2021) e Saha e Saha (2024) demonstram a eficácia da abordagem RAG em domínios informacionais diversos, como páginas web gerais, guias de museu e fóruns de discussão. No entanto, o presente trabalho enfrentou o desafio específico de consolidar 49 documentos PDF oficiais da UDESC, incluindo regulamentos, resoluções e editais que são a fonte primária de normativas e processos burocráticos. O processo de normalização e estruturação desses PDFs institucionais, frequentemente complexos e dispersos, é um dos passos que o distingue. Além disso, a avaliação quantitativa e comparativa utilizando RAGAS ofereceu uma análise aprofundada, identificando o ruído no contexto recuperado como o principal gargalo do sistema e comprovando a maior robustez do Gemini 2.5-pro contra essa contaminação factual, o que é vital para um agente institucional onde a recusa em alucinar é o comportamento mais desejável.

Para trabalhos futuros, sugere-se explorar técnicas de *Re-ranking*, implementando um segundo passo após a recuperação inicial para reordenar os *chunks* retornados e priorizar aqueles mais relevantes, conforme discutido por (Sahin, 2024). Além disso, pode-se aplicar estratégias diversificadas de segmentação, testando métodos mais avançados que o *RecursiveCharacterTextSplitter*, como segmentação baseada em sentenças semânticas ou ajustes no tamanho e na sobreposição dos *chunks*, aplicando engenharia de contexto. Por fim, ainda é possível expandir a base de conhecimento com um volume maior e mais diversificado de documentos institucionais da UDESC, ampliando assim o escopo e a capacidade de resposta do LLM.

Em suma, este trabalho cumpriu seus objetivos ao construir um fluxo RAG funcional e avaliar seu desempenho por meio das métricas RAGAS. Os resultados mostraram que o Gemini se destaca como uma tecnologia robusta e viável para implementar um assistente institucional confiável na UDESC, embora apresente um custo elevado; por outro lado, o Llama mostrou-se uma opção mais leve que, quando recebe um contexto preciso e livre de contaminação, é capaz de atingir um desempenho satisfatório. Os resultados apresentados neste documento foram disponibilizados em um repositório do GitHub[5].

[5] https://github.com/LucasRayzer/TCC

REFERÊNCIAS

CASELI, Helena de Medeiros; NUNES, Maria das Graças Volpe. **Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português**. 2. ed. São Carlos: BPLN, 2024. Organizadoras. Licença CC BY-NC-ND 3.0 BR. ISBN 978-65-00-95750-1. Disponível em: https://brasileiraspln.com/livro-pln. Acesso em: 26 abr. 2025.

CEUCA, Laura; REDNIC, Ana; CHIFU, Emil Stefan. Safer museum guide interaction during a pandemic and further using nlp in human interactive museum visits : Museum guide chatbot. In: **2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP)**. [S.l.: s.n.], 2021. p. 313–318.

CONSEPE, Universidade do Estado de Santa Catarina. **Resolução n° 026/2012 – Dispõe sobre a regulamentação das atividades complementares nos cursos de graduação da UDESC**. 2012. https://www.udesc.br/conselho/consepe/resolucoes#2012. Acesso em: 3 jun. 2025.

CONSEPE, Universidade do Estado de Santa Catarina. **Resolução n° 039/2015 – Regulamenta a realização de avaliações de segunda chamada nos cursos de graduação da UDESC**. 2015. https://www.udesc.br/conselho/consepe/resolucoes#2015. Acesso em: 3 jun. 2025.

DALE, Robert. Classical approaches to natural language processing. In: ____. **Handbook of natural language processing, second edition**. United States: CRC Press, Taylor, Francis Group, 2010. p. 3–7.

DALE, R.; MOISL, H.; SOMERS, H. **Handbook of Natural Language Processing**. CRC Press, 2000. ISBN 9780824746346. Disponível em: https://books.google.com.br/books?id=MnEjBsMIxxsC.

EXPLODINGGRADIENTS. **Ragas: Supercharge Your LLM Application Evaluations**. 2024. Disponível em: https://github.com/explodinggradients/ragas. Acesso em: 10 nov. 2025.

GABRIEL, Martha. **Inteligência Artificial: Do Zero ao Metaverso**. Rio de Janeiro: Atlas, 2022. E-book. p.67. ISBN 9786559773336. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9786559773336/. Acesso em: 17 abr. 2025.

GOOGLE. **Gemini 1.5 Pro: modelo de linguagem multimodal**. 2024. Software. Disponível em: https://deepmind.google/technologies/gemini/.

Google. **Curso introdutório de aprendizado de máquina: Embeddings**. 2025. Disponível em: https://developers.google.com/machine-learning/crash-course/embeddings?hl=pt-br. Acesso em: 17 abr. 2025.

HACKERS, D. **Construindo aplicações personalizadas com LLM através de RAG (Retrieve Augmented Generation)**. 2023. Disponível em: https://medium.com/data-hackers/construindo-aplica%C3%A7%C3%B5es-personalizadas-com-llm-atrav%C3%A9s-de-rag-retrieve-augmented-generation-6f3a3df7b6de. Acesso em: 02 mai. 2025.

JURAFSKY, Dan. **Speech & language processing**. [S.l.]: Pearson Education India, 2000.

LEE, Benjamin Charles Germain; OWENS, Trevor. **Grappling with the Scale of Born-Digital Government Publications: Toward Pipelines for Processing and Searching Millions of PDFs**. 2021. Disponível em: https://arxiv.org/abs/2112.02471.

MARTINS, C. O. et al. **Processamentos de Linguagem Natural**. Porto Alegre: SAGAH, 2020. E-book. p.15. ISBN 9786556900575. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9786556900575/. Acesso em: 24 mai. 2025.

MINAEE, Shervin et al. Large language models: A survey. **ArXiv**, abs/2402.06196, 2024. Disponível em: https://api.semanticscholar.org/CorpusID:267617032.

OLIVEIRA, Luana Ferreira. **Chatbot como ferramenta de apoio ao acesso às informações acadêmicas da UFSM**. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) — Universidade Federal de Santa Maria (UFSM), 2024.

OPENAI. **GPT-3.5: modelo de linguagem da OpenAI**. 2023. Software. Disponível em: https://openai.com.

RAIA - Rede de Acesso à Informação Aberta. **Chat Diário Oficial**. 2025. Disponível em: https://grupo-raia.org/projects/project/1. Acesso em: 03 jun. 2025.

RAVAGNANI, Fernanda. **Dificuldade financeira, ensino médio fraco e desilusão causam desistência em universidades**. 2023. Disponível em: https://ruf.folha.uol.com.br/2023/noticias/dificuldade-financeira-ensino-medio-fraco-e-desilusao-causam-desistencia-em-universidades.shtml. Acesso em: 19 nov. 2025.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência Artificial: Uma Abordagem Moderna**. 4. ed. Rio de Janeiro: GEN LTC, 2022. E-book. p.33. ISBN 9788595159495. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788595159495/. Acesso em: 17 abr. 2025.

SAHA, Binita; SAHA, Utsha. Enhancing international graduate student experience through ai-driven support systems: A llm and rag-based approach. In: IEEE. **2024 International Conference on Data Science and Its Applications (ICoDSA)**. [S.l.], 2024. p. 300–304.

SAHIN, Samia. **What is Re-ranking in Retrieval Augmented Generation (RAG)?** 2024. Disponível em: https://medium.com/@sahin.samia/what-is-reranking-in-retrieval-augmented-generation-rag-ee3dd93540ee. Acesso em: 17 nov. 2025.

SINGER-VINE, Jeremy. **pdfplumber**. 2023. Disponível em: https://github.com/jsvine/pdfplumber. Acesso em: 10 nov. 2025.

TURING, A. M. I.—computing machinery and intelligence. **Mind**, LIX, n. 236, p. 433–460, 10 1950.

UDESC-CEAVI. **Resoluções - Secretaria CEAVI**. 2025. Disponível em: https://www.udesc.br/ceavi/secretaria/resolucoes. Acesso em: 27 out. 2025.

Universidade do Estado de Santa Catarina. **A UDESC**. 2025. Universidade pública estadual de Santa Catarina. Disponível em: https://www.udesc.br. Acesso em: 03 jun. 2025.

WEI, Jason et al. Emergent abilities of large language models. **arXiv preprint arXiv:2206.07682**, 2022.

APÊNDICES

APÊNDICE A – PERGUNTAS E RESPOSTAS PARA AVALIAÇÃO RAGAS

1. **Pergunta:** Quem precisa assinar a autodeclaração dizendo que pertence ao grupo racial negro no vestibular da UDESC?
   Resposta: Os candidatos classificados nas vagas destinadas a pessoas negras precisam assinar a autodeclaração.

2. **Pergunta:** Quem pode pedir o regime especial de atendimento domiciliar na UDESC?
   Resposta: Podem solicitar estudantes em licença maternidade, em situações excepcionais com atestado médico, ou acadêmicos com condições de saúde que impeçam a frequência às aulas, como tratamentos, infecções, traumatismos ou outras afecções comprovadas por profissional de saúde.

3. **Pergunta:** Que tipos de atividades podem contar como Atividades Complementares na UDESC?
   Resposta: Podem ser atividades de ensino, pesquisa, extensão, administração universitária ou mistas que integrem teoria e prática, conforme previsto na regulamentação.

4. **Pergunta:** Como o aluno valida uma atividade de extensão na UDESC?
   Resposta: Entregando o formulário assinado pelo coordenador da ação à Coordenação de UCE, que registra os créditos.

5. **Pergunta:** Qual é a média mínima para um aluno ser aprovado na UDESC?
   Resposta: Média 7,0 e frequência mínima de 75%.

6. **Pergunta:** Qual é o valor de um crédito nas disciplinas da graduação?
   Resposta: Cada crédito corresponde a 18 horas de atividades.

7. **Pergunta:** Quantas disciplinas isoladas podem ser cursadas por semestre na UDESC?
   Resposta: Cada solicitante pode se matricular em no máximo duas disciplinas isoladas por semestre, salvo exceções previstas, como disciplinas orientadas ou casos de alunos estrangeiros que necessitam validar o diploma.

8. **Pergunta:** Quais disciplinas podem ser oferecidas em caráter de Estudo Dirigido?
   Resposta: Podem ser oferecidas disciplinas da matriz curricular da UDESC que foram extintas, estão em extinção ou possuem equivalência parcial com as novas matrizes.

9. **Pergunta:** Quais condições o aluno deve atender para se inscrever no Exame de Suficiência?
   Resposta: O aluno deve cumprir pré-requisitos, não ter sido reprovado, não ter feito exame antes, não ter faltado previamente e apresentar documentos que comprovem conhecimento ou habilidade.

10. **Pergunta:** Como funciona a avaliação do extraordinário aproveitamento?
    Resposta: A avaliação é feita por Banca Examinadora de 3 professores, com provas sobre todo o conteúdo da disciplina; a nota é a média aritmética, e mínimo 9,0 garante aprovação sem exame final.

11. **Pergunta:** É permitido ao aluno concluir o curso em menos tempo que o mínimo previsto?
    Resposta: Não, a conclusão do curso não pode ser inferior ao prazo mínimo estabelecido para integralização do currículo.

12. **Pergunta:** É permitido ao aluno da UDESC matricular-se em dois ou mais cursos de graduação ao mesmo tempo?
    Resposta: Não, é vedada a matrícula e a frequência simultânea em dois ou mais cursos.

13. **Pergunta:** É possível desmatricular-se de disciplinas fora do período do calendário acadêmico?
    Resposta: Sim, até 15 dias antes do encerramento do período letivo, mediante solicitação justificada e comprovada pelo Sistema Acadêmico, a ser analisada pela Chefia do Departamento.

14. **Pergunta:** O que acontece com as disciplinas do currículo em extinção que não têm equivalência na nova matriz curricular?
    Resposta: Essas disciplinas devem permanecer no histórico escolar do(a) acadêmico(a) e podem ser oferecidas conforme as normativas da universidade, se necessário.

15. **Pergunta:** Quem pode solicitar a inclusão do nome social nos registros acadêmicos da UDESC?
    Resposta: Pessoas transgênero, travestis e transexuais podem requerer a inclusão do nome social, sendo que acadêmicos maiores de 18 anos podem solicitar a qualquer tempo, enquanto menores precisam de autorização por escrito dos pais ou responsáveis legais.

16. **Pergunta:** Quem está habilitado a participar da solenidade de Outorga de Grau?
    Resposta: Somente o aluno que concluiu o currículo completo do curso, incluindo estágios e/ou Trabalho de Conclusão de Curso (TCC), conforme aprovação do colegiado. Alunos que concluíram apenas uma nova habilitação do curso não participam da solenidade.

17. **Pergunta:** Quantas avaliações mínimas o professor deve aplicar por disciplina?
    Resposta: No mínimo duas avaliações por semestre.

18. **Pergunta:** Quem pode solicitar a segunda chamada de uma avaliação?
    Resposta: O acadêmico regularmente matriculado que deixou de comparecer à avaliação nas datas fixadas pelo professor, mediante requerimento assinado e entregue na Secretaria de Ensino de Graduação ou Secretaria do Departamento.

19. **Pergunta:** Como funciona o abono de faltas por crença religiosa?
    Resposta: Acadêmicos que, por força de crença religiosa, deixarem de comparecer às aulas sextas-feiras após 18h até o pôr do sol de sábado, têm essas faltas registradas como dispensa. O acadêmico deve comprovar a participação na congregação através de atestados com firma reconhecida. Essas faltas não contam para o cálculo de

20. **Pergunta:** Quais normas a UDESC adota nos processos de revalidação e reconhecimento de diplomas estrangeiros?
    Resposta: A UDESC adota a Resolução nº 3, de 22 de junho de 2016, da Câmara de Educação Superior do CNE/MEC, e a Portaria Normativa nº 22, de 13 de dezembro de 2016, do Ministro de Estado da Educação.

21. **Pergunta:** Qual é o prazo para solicitar a revisão de nota após a divulgação do resultado da avaliação?
    Resposta: O aluno deve apresentar a solicitação à Secretaria Acadêmica do Centro em até 10 dias após a publicação do resultado da avaliação.

22. **Pergunta:** Quantos alunos são necessários para a abertura de uma turma de disciplina optativa?
    Resposta: O número mínimo de acadêmicos necessários para a realização de cada turma/disciplina optativa é igual a 10 (dez).

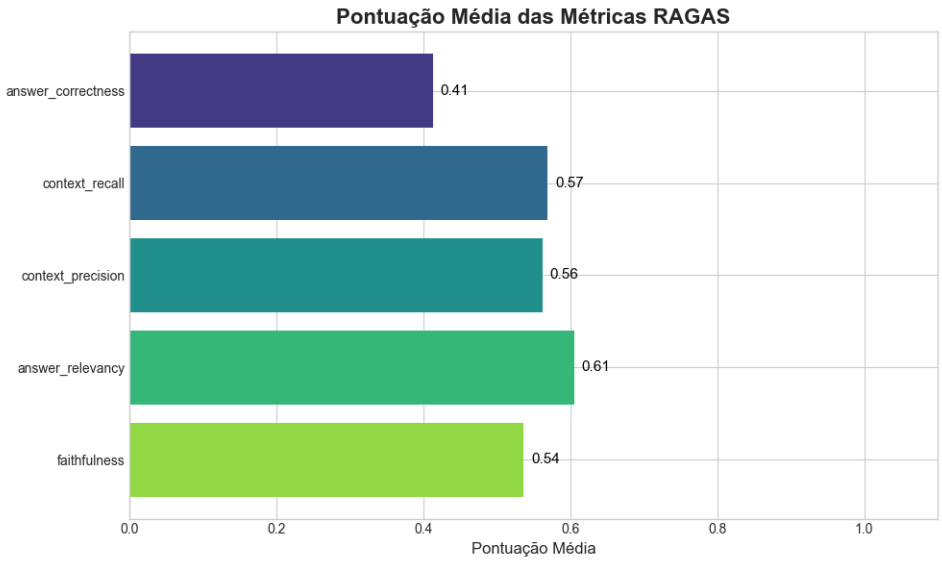
23. **Pergunta:** Quais são as modalidades de ingresso para ocupação de vagas ociosas nos cursos de graduação da UDESC?
    Resposta: As modalidades de ingresso para ocupação de vagas ociosas são: Transferência Interna, Transferência Externa, Reingresso após Abandono, Reingresso após Cancelamento de Matrícula pelo(a) estudante, e Retorno ao Portador de Diploma de Graduação.

24. **Pergunta:** Qual é o critério de equivalência mínima para que a disciplina seja validada?
    Resposta: O programa da disciplina cursada deve corresponder a, no mínimo, 75% do conteúdo e da carga horária da disciplina que o(a) acadêmico(a) deveria cumprir na UDESC.

ANEXOS

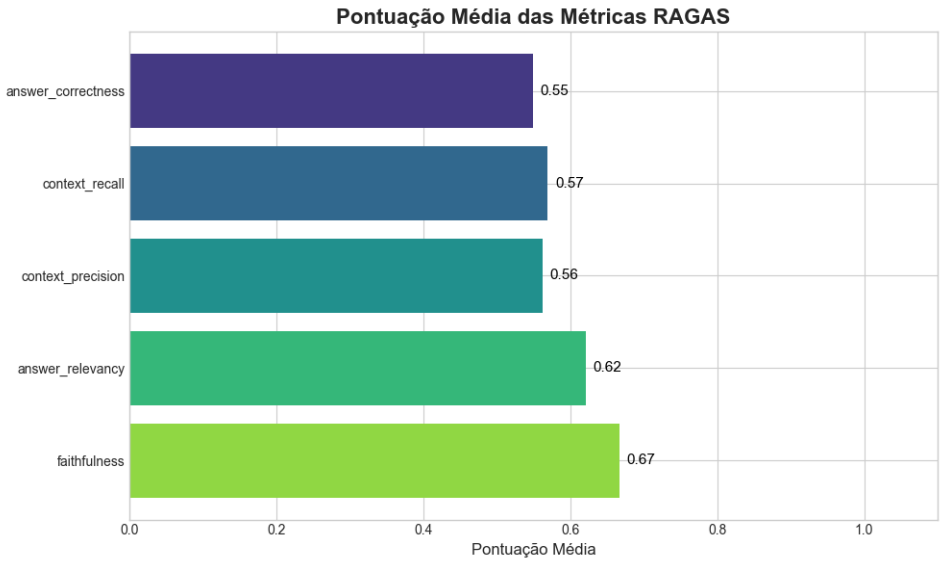
ANEXO A – GRÁFICOS DOS RESULTADOS OBTIDOS

Figura 14 – Pontuação Média RAGAS - Llama (k=2)



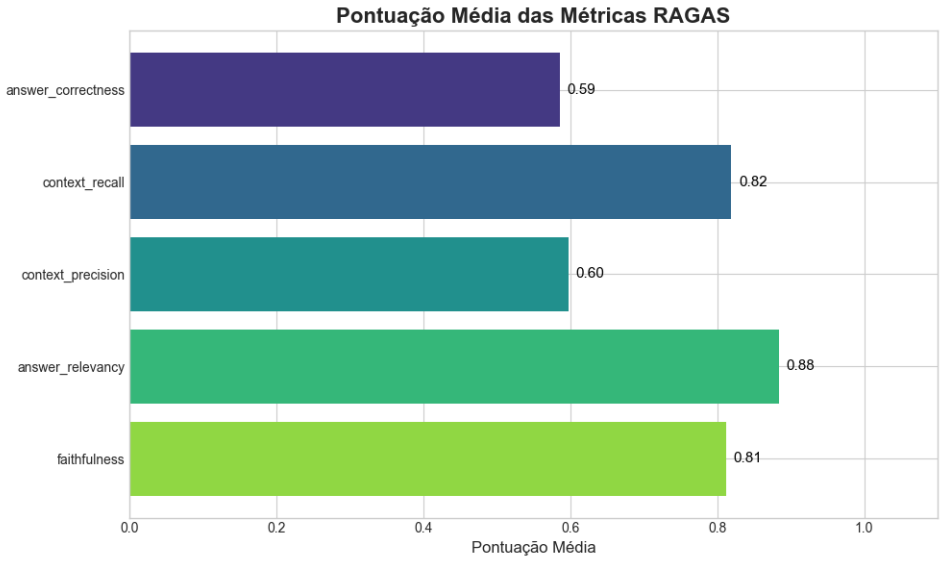
Fonte: Elaborado pelo autor.

Figura 15 – Pontuação Média RAGAS - Gemini (k=2)



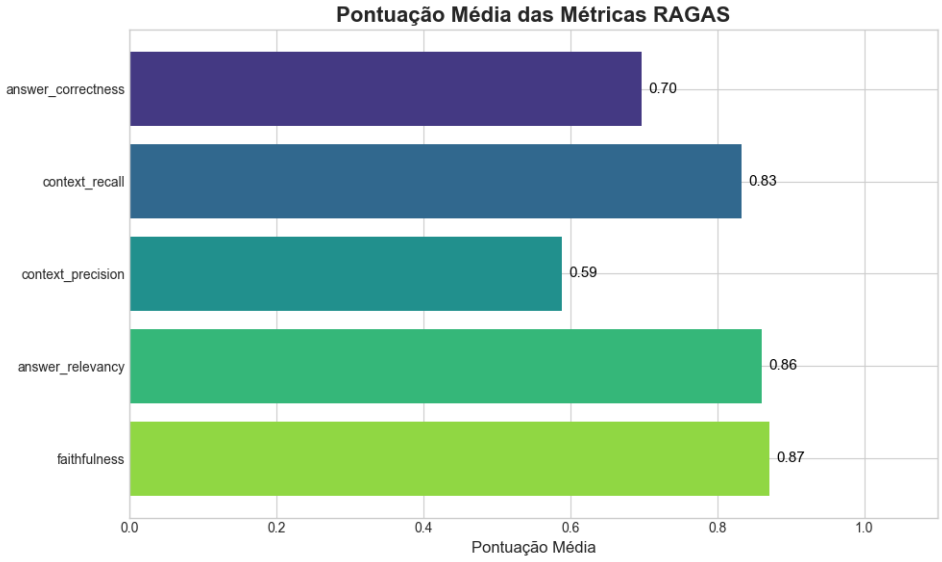
Fonte: Elaborado pelo autor.

Figura 16 – Pontuação Média RAGAS - Llama (k=5)



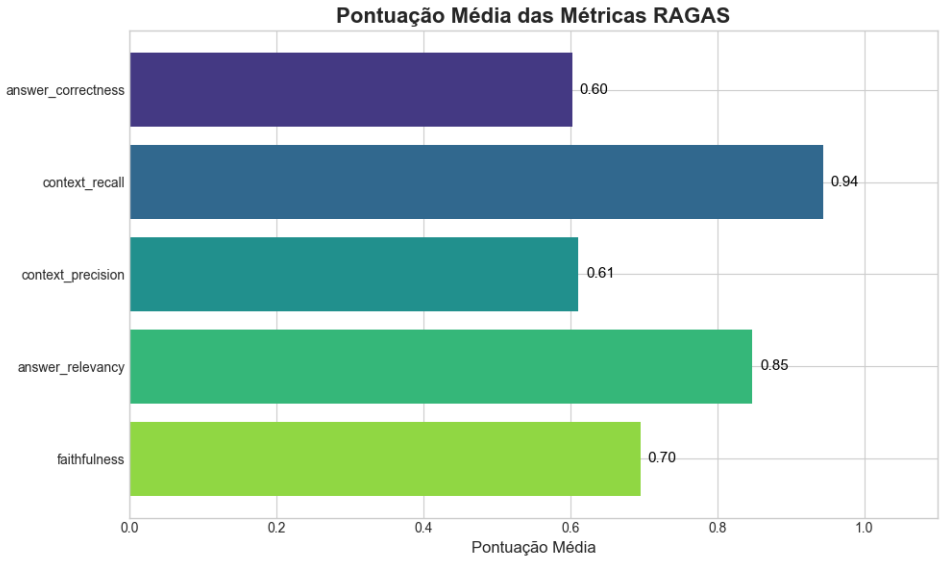
Fonte: Elaborado pelo autor.

Figura 17 – Pontuação Média RAGAS - Gemini (k=5)



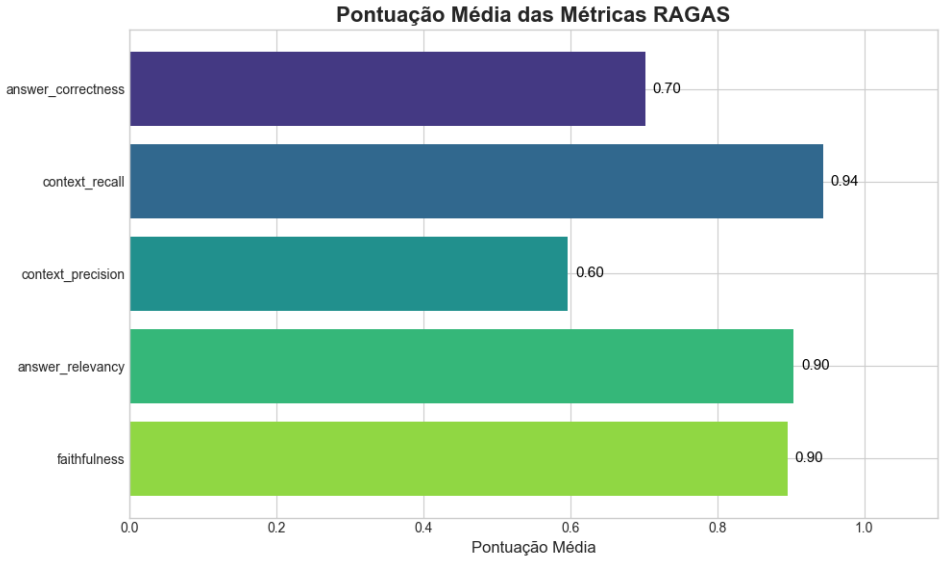
Fonte: Elaborado pelo autor.

Figura 18 – Pontuação Média RAGAS - Llama (k=8)



Fonte: Elaborado pelo autor.

Figura 19 – Pontuação Média RAGAS - Gemini (k=8)



Fonte: Elaborado pelo autor.