João Pedro Freire Cabral

Chatbot Inteligente para Acesso a Regulamentos Acadêmicos: Um Sistema de Recuperação de Informações Baseado em RAG

Natal – RN

Janeiro de 2025

João Pedro Freire Cabral

Chatbot Inteligente para Acesso a Regulamentos Acadêmicos: Um Sistema de Recuperação de Informações Baseado em RAG

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Jean Mario Moreira de Lima

Universidade Federal do Rio Grande do Norte – UFRN

Departamento de Engenharia de Computação e Automação – DCA

Curso de Engenharia de Computação

Natal – RN

Janeiro de 2025

João Pedro Freire Cabral

Chatbot Inteligente para Acesso a Regulamentos Acadêmicos: Um Sistema de Recuperação de Informações Baseado em RAG

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Jean Mario Moreira de Lima

Trabalho aprovado. Natal – RN, 21 de Janeiro de 2025:

Prof. Dr. Jean Mario Moreira de Lima - Orientador
UFRN

Prof. Dr. André Morais Gurgel - Examinador
UFRN

Prof. Dr. Sérgio Natan Silva - Externo
UFCG

Natal – RN
Janeiro de 2025

Para todos aqueles que, de alguma forma, poderão tirar proveito deste trabalho e, sobretudo, Àquele que é a origem de tudo e se fez carne.

AGRADECIMENTOS

A realização deste trabalho só foi possível graças ao apoio, incentivo e orientação de diversas pessoas e instituições, às quais expresso minha sincera gratidão.

Agradeço primeiramente à minha família, por seu apoio incondicional, compreensão e incentivo em cada etapa desta jornada acadêmica. Sua presença foi fundamental para que eu pudesse enfrentar os desafios e seguir em frente com determinação.

Aos meus professores e orientadores, que, com paciência e dedicação, compartilharam seus conhecimentos e ofereceram valiosas orientações ao longo do desenvolvimento deste trabalho. Suas contribuições foram essenciais para aprimorar a qualidade e profundidade desta pesquisa.

Aos colegas e amigos, pelo apoio, pelas trocas de ideias e pelo companheirismo ao longo dessa trajetória, que tornaram este percurso mais leve e motivador.

À instituição de ensino e a todos os colaboradores que, direta ou indiretamente, contribuíram para a realização deste estudo, oferecendo infraestrutura, suporte técnico e um ambiente propício ao aprendizado e desenvolvimento acadêmico.

Por fim, expresso minha gratidão a todos aqueles que, de alguma forma, participaram deste processo e que, com suas palavras de incentivo e apoio moral, me ajudaram a concluir mais esta etapa da minha vida.

“Se quisermos buscar um propósito cósmico,
então que encontremos um objetivo digno
do nosso vasto universo."
(Carl Sagan)

RESUMO

O mercado de chatbots no Brasil encontra-se em franca expansão, evidenciado pelo aumento expressivo tanto no número de bots em atividade quanto no volume de mensagens processadas. Esses sistemas apresentam grande potencial para modernizar serviços, sobretudo no setor público, ao facilitar a disseminação de regulamentações complexas e ampliar a acessibilidade para cidadãos e servidores. Em paralelo, cresce o interesse acadêmico por modelos de linguagem de grande escala, refletido na quantidade de publicações científicas que destacam sua capacidade de compreender e gerar linguagem natural de forma mais contextualizada e confiável. Diante desse cenário, este trabalho tem como objetivo desenvolver um chatbot inteligente baseado em modelos de linguagem de grande escala para facilitar o acesso às normas acadêmicas da UFRN, promovendo maior acessibilidade e precisão na obtenção de informações. Para contornar limitações como necessidade de dados atualizados e respostas mais contextualizadas, emprega-se o método Retrieval-Augmented Generation, que integra geração de texto e recuperação de fontes externas. Ao longo do desenvolvimento, a estruturação do dataset com o framework RAGAS e a escolha criteriosa da segmentação semântica dos regulamentos mostraram-se cruciais para a consistência das respostas. Os testes realizados evidenciaram que a abordagem de recuperação tradicional combinada com reranking proporcionou os melhores resultados, enquanto métodos como consultas múltiplas e geração de documentos hipotéticos não apresentaram desempenho satisfatório. O modelo Rerank 1.0 da Amazon foi determinante para filtrar documentos irrelevantes, garantindo maior precisão e confiabilidade. Em ambiente de produção, o chatbot atingiu bom desempenho, com consumo médio de memória de 241 MB e tempo de resposta aproximado de 4,5 segundos, indicando escalabilidade e robustez da solução. Apesar desses avanços, desafios como custo computacional e ajustes contínuos permanecem, reforçando a necessidade de aperfeiçoamento. Em síntese, a proposta apresentada fornece uma abordagem sólida e replicável para implementação de chatbots baseados em Retrieval-Augmented Generation, com aplicações em contextos acadêmicos e administrativos que exijam alta precisão e contextualização das informações.

Palavras-chaves: chatbot; normas; modelos de linguagem de grande escala; acesso a informação.

ABSTRACT

The Brazilian chatbot market is experiencing rapid growth, marked by a notable increase in both the number of active bots and the volume of processed messages. These systems hold significant potential for modernizing services, particularly in the public sector, by streamlining the dissemination of complex regulations and enhancing accessibility for citizens and public servants. In parallel, there has been a surge in academic interest in large language models, evidenced by the growing volume of scientific publications that highlight their ability to understand and generate natural language with greater contextual accuracy. Against this backdrop, the present study aims to develop an intelligent chatbot based on large language models to facilitate access to the academic regulations of UFRN, thereby promoting improved accessibility and information accuracy. To address challenges such as the need for up-to-date data and more contextualized responses, we adopt the Retrieval-Augmented Generation approach, which combines text generation with external data retrieval. During the development phase, the structured dataset created with the RAGAS framework, alongside a well-considered semantic segmentation of the regulations, proved essential for ensuring consistent responses. Test results indicated that a traditional retrieval method enhanced by reranking yielded the most satisfactory outcomes, while techniques such as multiple queries and hypothetical document generation did not demonstrate satisfactory performance. The Rerank 1.0 model from Amazon played a key role in filtering irrelevant documents, thereby improving the precision and reliability of the responses. In a production environment, the chatbot showed robust scalability, consuming an average of 241 MB of memory and achieving a response time of approximately 4.5 seconds. Nevertheless, challenges remain, including computational costs and the need for continual adjustments. Overall, this study provides a solid and replicable approach to implementing Retrieval-Augmented Generation chatbots, with potential applications in both academic and administrative contexts that demand high precision and contextualization of information.

Keywords: chatbot; regulations; large-scale language models; information access.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFRN	*Universidade Federal do Rio Grande do Norte*
PROGRAD	*Pró-Reitoria de Graduação*
LLM	*Large Language Model*
PLN	*Processamento de Linguagem Natural*
RNN	*Redes Neurais Recorrentes*
NLMs	*Modelos de Linguagem Neurais*
RAG	*Geração Aumentada de Recuperação*
HyDE	*Hypothetical Document Embedding*
SIA	*Secretaria de Inclusão e Acessibilidade*
DPR	*Dense Passage Retrieva*

SUMÁRIO

1 INTRODUÇÃO

No Brasil, o mercado de chatbots continua em expansão, destacando-se como uma ferramenta essencial em diversos setores. De acordo com o Mapa do Ecossistema Brasileiro de Bots (PAIVA, 2022), o número de bots desenvolvidos aumentou 47% entre 2021 e 2022, atingindo um total de 317 mil robôs de conversação, enquanto os bots em atividade cresceram 23%, totalizando 58 mil unidades. Esses robôs trafegam mensalmente cerca de 4,5 bilhões de mensagens, representando um aumento de 60% no tráfego em comparação ao ano anterior. No contexto do setor público, embora a adoção ainda seja limitada, chatbots apresentam grande potencial para modernizar serviços, especialmente em áreas que envolvem regulamentações complexas, como nas instituições públicas. Aplicações voltadas para a interpretação e disseminação de normas, por exemplo, podem não apenas aumentar a eficiência administrativa, mas também promover maior transparência e acessibilidade para cidadãos e servidores.

A crescente adoção de chatbots no Brasil reflete a demanda por soluções automatizadas capazes de lidar com grandes volumes de interações, sejam elas no setor privado ou público. Nesse cenário, os Modelos de Linguagem de Grande Escala (LLMs) despontam como tecnologias fundamentais para aprimorar as capacidades dos chatbots, possibilitando interações mais naturais e eficientes. Enquanto os chatbots tradicionais têm mostrado resultados promissores no atendimento ao cliente e na disseminação de informações básicas (PAIVA, 2022), os LLMs, como o GPT-3 e o PaLM, oferecem uma evolução significativa ao permitir a interpretação de regulamentações complexas e a personalização das respostas, características essenciais para aplicações mais sofisticadas. Essa convergência tecnológica destaca o papel dos LLMs como catalisadores da próxima geração de soluções conversacionais, alinhando-se às necessidades do mercado e aos desafios enfrentados pelas instituições públicas e privadas.

Os modelos de LLM também têm atraído crescente atenção da comunidade científica devido ao impacto significativo no desempenho de tarefas complexas. Estudos demonstram que a escalabilidade desses modelos, seja no tamanho do modelo ou na quantidade de dados de treinamento, segue a chamada lei de escala (KAPLAN et al., 2020), resultando em uma maior capacidade dos modelos para resolver tarefas avançadas. Exemplos notáveis incluem o GPT-3, com 175 bilhões de parâmetros, e o PaLM, com 540 bilhões de parâmetros. Esses modelos superam suas versões menores, como o GPT-2 e o BERT, apresentando habilidades emergentes, como a aprendizagem contextualizada (*few-shot learning*), que não são observadas em modelos de menor escala (WEI et al., 2022). A relevância dos LLMs foi amplificada com o lançamento de aplicativos como o ChatGPT, que demonstram

capacidades de diálogo impressionantes, levando a um aumento acentuado no número de publicações acadêmicas relacionadas a LLMs, conforme ilustrado na Figura 1.

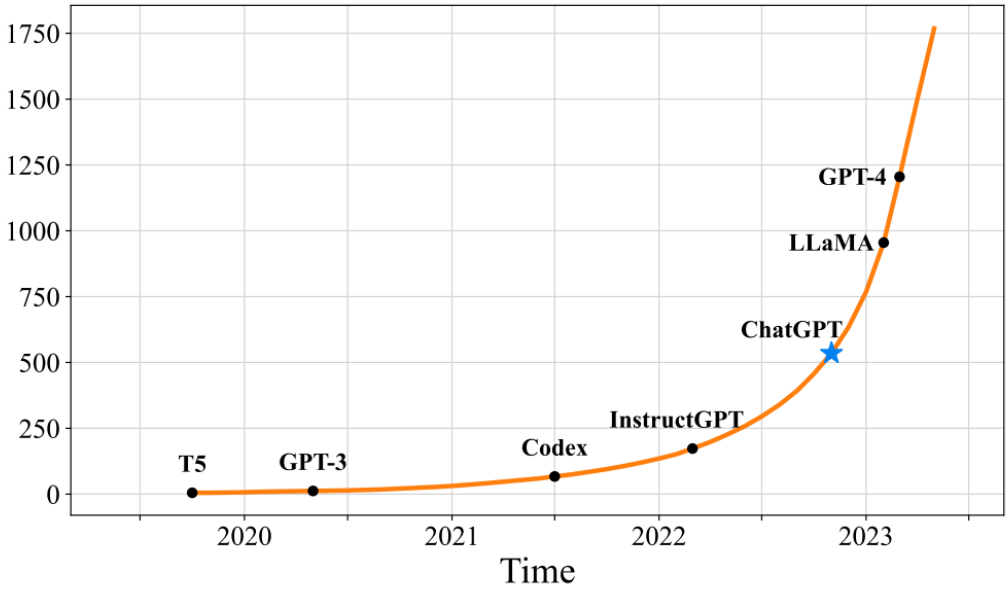

Figura 1 – Publicações no Arxiv com query = Large Language Models.

Fonte: Zhao et al. (2024, p. 2).

Atualmente, o número de publicações no arXiv relacionadas a Modelos de Linguagem de Grande Escala ultrapassou recentemente a marca de 20 mil artigos, demonstrando o enorme crescimento no interesse acadêmico e na relevância desses modelos na pesquisa atual. Esse aumento exponencial é reflexo não apenas das inovações tecnológicas trazidas pelos LLMs, mas também do impacto significativo que eles têm gerado em diversas áreas, como processamento de linguagem natural, geração de texto, e resolução de problemas complexos em geral. Esses LLMs também têm transformado o setor privado ao impulsionar avanços no processamento e geração de linguagem natural. Soluções amplamente utilizadas, como o ChatGPT, desenvolvido pela OpenAI, e o Google Bard, destacam-se por automatizar interações em linguagem natural, aprimorando serviços como atendimento ao cliente, geração de conteúdo e análise de dados, o que resulta em maior eficiência operacional e personalização de experiências para os consumidores (DATACAMP, 2023).

As principais aplicações dos LLMs no mercado privado abrangem:

- **Atendimento ao Cliente:** Implementação de agentes de IA para responder automaticamente a perguntas frequentes, melhorando a eficiência e a satisfação do cliente (SKIMAI, 2023).
- **Análise de Dados:** Conversão de linguagem natural em consultas SQL para facilitar a análise de dados empresariais, permitindo insights mais profundos e decisões informadas (SKIMAI, 2023).

- **Criação de Conteúdo:** Assistência na geração de conteúdos para blogs e redes sociais, otimizando estratégias de marketing e engajamento com o público (SKIMAI, 2023).

Já setor público, os LLMs têm sido integrados para aprimorar serviços governamentais. Por exemplo, o governo brasileiro iniciou projetos-piloto utilizando IA para interpretar vastas quantidades de documentos relacionados à gestão de pessoas no executivo (SERPRO, 2024). Além disso, iniciativas como o INACIA demonstram a aplicação de LLMs em tribunais de contas, automatizando etapas de análise processual (PEREIRA et al., 2024).

Dado esse cenário, as universidades emergem como ambientes propícios para a adoção crescente de LLMs. A integração dessa tecnologia pode transformar processos acadêmicos e administrativos, promovendo eficiência e inovação. LLMs podem auxiliar na análise de grandes volumes de dados de pesquisa, na automação de atendimentos aos estudantes e na melhoria da comunicação institucional. A adoção de LLMs nas universidades não apenas moderniza suas operações, mas também prepara os alunos para interagir com tecnologias de ponta, alinhando-se às demandas do mercado e contribuindo para a formação de profissionais capacitados para lidar com as inovações tecnológicas emergentes.

1.1 Motivação

A implementação de um chatbot para facilitar o acesso a documentos acadêmicos em uma universidade apresenta diversas motivações fundamentadas. Primeiramente, os chatbots podem fornecer suporte personalizado em grande escala, disponibilizando informações em várias línguas e integrando-se com ferramentas de acessibilidade, o que melhora a inclusão e a acessibilidade para todos os alunos (BOTPRESS, 2024). Além disso, ao automatizar tarefas administrativas, como o fornecimento de documentos acadêmicos, os chatbots permitem que os colaboradores se concentrem em demandas estratégicas, otimizando a eficiência institucional (OMNICHAT, 2024). A utilização de chatbots no ambiente acadêmico também tem demonstrado impactos positivos significativos. Estudos indicam que, após a incorporação de um chatbot em uma instituição de ensino superior privada, houve um aumento na precisão do atendimento ao corpo discente e aos candidatos, contribuindo para melhores índices de retenção estudantil (CASTOR et al., 2021). Portanto, a implementação de um chatbot para acesso a documentos acadêmicos não apenas aprimora a experiência dos alunos, mas também fortalece a eficiência operacional da universidade.

No contexto da UFRN e seus regulamentos acadêmicos, um trabalho recente (CUNHA, 2020) indicou que estudantes de graduação frequentemente relatam desafios

relacionados à falta de acompanhamento e informações claras desde o início do curso, especialmente durante o processo de adaptação ao Regime de Observação de Desempenho Acadêmico (RODA). Muitos alunos só percebem as implicações do regime tardiamente e, em muitos casos, não têm conhecimento sobre os procedimentos necessários para a recuperação acadêmica e os suportes institucionais disponíveis. Esses problemas de comunicação geram frustrações, e a carência de informações precisas e acessíveis se torna um obstáculo significativo para o sucesso acadêmico. Nesse cenário, o uso de tecnologias, como ambientes virtuais de comunicação no SIGAA, e-mails e aplicativos, foi identificado como uma solução promissora para melhorar a eficiência do processo de orientação acadêmica. Além disso, a utilização de recursos tecnológicos, como o SIGAA, que integra diversas funções acadêmicas, pode facilitar o acesso à informação, tornando o acompanhamento dos alunos mais ágil e transparente. A implementação de um chatbot poderia atender a essas necessidades, proporcionando respostas rápidas e contextualizadas sobre os regulamentos acadêmicos e procedimentos de recuperação, tornando a experiência dos alunos mais fluida e acessível

Também na UFRN, outro trabalho (SANTOS, 2022) mostrou que a falta de visibilidade e acesso aos recursos de apoio disponíveis para os discentes sugere que a centralização da informação em um local de fácil acesso e com maior visibilidade é fundamental. A integração de um chatbot que centralize essas informações pode ser uma solução eficiente para resolver esse problema, proporcionando um ponto de acesso único e de fácil navegação. Revela-se também o desconhecimento sobre o apoio emocional e mental oferecido pela Secretaria de Inclusão e Acessibilidade (SIA), que pode ser parcialmente resolvido com um chatbot que forneça informações e orientações claras sobre esses serviços, especialmente para alunos que enfrentam dificuldades emocionais que afetam seu desempenho acadêmico. A personalização do chatbot para lidar com consultas sensíveis pode ser uma maneira de aumentar a conscientização e o uso desses recursos de apoio. Além disso, a pesquisa revela que muitos estudantes não utilizaram a orientação acadêmica oferecida pelo RODA, mesmo aqueles em risco de cancelamento por insuficiência de desempenho acadêmico. Esse ponto destaca a necessidade de melhorar a comunicação e a interatividade dos canais de orientação, algo que pode ser aprimorado com o uso de chatbots para fornecer informações personalizadas de acompanhamento acadêmico. Por fim, a pesquisa sugere que o cancelamento de vínculos acadêmicos pode ser evitado se os estudantes forem apoiados adequadamente durante sua trajetória, o que reforça a importância de sistemas como autônomos para auxiliar na retenção de alunos, fornecendo orientação acadêmica proativa, informações sobre o desempenho e conectando os estudantes com recursos de apoio quando necessário. Essa abordagem pode melhorar o desempenho e evitar a evasão, alinhando-se aos objetivos das instituições de ensino de manter os alunos no curso e apoiar seu sucesso acadêmico.

1.2 Trabalhos Relacionados

Um dos trabalhos mais influentes no campo de chatbots baseados em LLM para contextos específicos é o Recuperação Aumentada por Geração (RAG), publicado no artigo de Lewis (LEWIS et al., 2021), que propõe o uso de RAG para tarefas de Processamento de Linguagem Natural que demandam acesso a informações externas. A abordagem combina dois módulos principais: um modelo de recuperação, responsável por identificar documentos relevantes em uma base de conhecimento, como a Wikipedia, e um modelo de geração, que usa os documentos recuperados para produzir respostas contextualizadas. Essa estratégia baseia-se em um pipeline que aproveita o Dense Passage Retrieval (DPR) como mecanismo de recuperação e o BART como gerador de respostas. O método se mostrou particularmente eficaz em benchmarks de perguntas e respostas, superando modelos puramente gerativos por integrar de maneira eficiente informações factuais que não fazem parte do treinamento. O trabalho também destaca a interpretabilidade do modelo, já que as respostas podem ser vinculadas diretamente aos documentos recuperados, e a sua adaptabilidade para cenários em que é necessário lidar com informações dinâmicas e atualizadas, tornando-o um marco na aplicação de LLMs em tarefas baseadas em conhecimento. Logo, toda a estrutura desse trabalho seguirá as etapas descritas por Lewis, tomando as adpatações necessárias para adequar ao contexto do trabalho.

É possível categorizar arquiteturas de RAG em três níveis de complexidade: modelos naïve, avançados e modulares. (GAO et al., 2024). Cada uma dessas categorias apresenta características, vantagens e limitações específicas, oferecendo uma visão abrangente sobre as diferentes abordagens para sistemas RAG. O levantamento de Gao et al. oferece uma visão abrangente das possibilidades de implementação de RAG e destaca a importância de escolher a abordagem mais adequada ao contexto de aplicação. No caso deste projeto, a abordagem modular pode ser especialmente relevante, pois permite o uso de módulos independentes para lidar com a recuperação, organização e interpretação de normas acadêmicas, garantindo maior flexibilidade para futuras expansões e melhorias.

O trabalho Chat-EUR-Lex (CHERUBINI et al., 2024) apresenta uma solução baseada em RAG para melhorar o acesso a textos legais complexos no contexto europeu, utilizando documentos do repositório EUR-Lex em inglês e italiano. A abordagem combina um sistema de recuperação de informações semânticas com um modelo de geração, permitindo interações via chatbot e proporcionando respostas contextualizadas com base nos regulamentos relevantes. Entre os pontos fortes, destacam-se a utilização de embeddings especializados para busca semântica, o tratamento de granularidade com segmentação de textos, *chunking*, e a inclusão de metadados na recuperação, o que melhora a precisão e relevância das respostas. Contudo, o projeto enfrenta desafios significativos que podem ser explorados para melhorias, como a dificuldade em lidar com consultas muito curtas, a limitação de respostas que envolvam múltiplos documentos e o impacto do limite de

comprimento de texto nos LLMs. No contexto de regulamentos acadêmicos, é possível adaptar e superar essas limitações ao refinar a segmentação dos dados, personalizar embeddings para o domínio educacional e explorar estratégias de reformulação automática de consultas curtas para garantir respostas mais completas e precisas.

O artigo sobre Korean Legal QA (RYU et al., 2023) apresenta o método Eval-RAG, que utiliza o componente de recuperação de documentos do sistema RAG para aprimorar a avaliação de respostas geradas por LLMs. Nesse método, um conjunto de documentos relevantes é recuperado com base na pergunta e, em seguida, um avaliador utiliza esses documentos para verificar a correção e validade da resposta. Este processo mostrou-se superior a métodos tradicionais de avaliação, como BLEU e BERTScore, pois evita os problemas de avaliação baseada apenas em similaridade textual. Uma vantagem é a capacidade de identificar erros factuais em respostas geradas, o que é essencial no domínio jurídico. No entanto, uma limitação identificada é o foco em apenas um documento recuperado, restringindo a capacidade de lidar com respostas que envolvem múltiplas fontes. Já o LegalBench-RAG (PIPITONE; ALAMI, 2024) introduz um benchmark projetado especificamente para avaliar a etapa de recuperação em sistemas RAG no domínio jurídico. O benchmark utiliza pares de perguntas e trechos de texto relevantes, anotados por especialistas jurídicos, para medir a precisão e recall das estratégias de recuperação. O método inclui avaliações rigorosas de diferentes estratégias, como o uso de divisões recursivas (*Recursive Text Character Splitter*) e *rerankers*. Uma contribuição importante é o foco na recuperação de pequenos trechos precisos, em vez de documentos inteiros, para evitar problemas de contexto excessivo e alucinações. Esses estudos são cruciais para embasar a avaliação deste trabalho, pois propõem metodologias que integram recuperação de informações e validação baseada em evidências, utilizando pares de pergunta-resposta e documentos relevantes para medir precisão e relevância de forma objetiva.

1.3 Objetivos

O objetivo geral deste projeto é desenvolver um chatbot inteligente baseado em modelos de linguagem de grande escala para facilitar o acesso às normas e regulamentos acadêmicos de uma universidade. Essa iniciativa visa simplificar o processo de consulta e interpretação de regulamentos por parte de alunos, professores e demais membros da comunidade acadêmica, promovendo maior acessibilidade, eficiência e precisão na obtenção de informações.

Dentre os objetivos específicos, pode-se citar:

- Identificar as principais maneiras de interpretar e consultar as normas e artigos do Regulamento dos Cursos Regulares de Graduação da UFRN.

- Implementar um modelo de linguagem treinado para compreender e responder a perguntas sobre normas acadêmicas.
- Avaliar a precisão e relevância das respostas geradas pelo sistema.
- Propor melhorias no desempenho do modelo baseado nos resultados obtidos durante os testes.

1.4 Estrutura do Trabalho

Este trabalho está estruturado em cinco capítulos principais, além das referências. O Capítulo 1 apresenta uma introdução geral ao tema, abordando os fatores que motivam a implantação do chatbot baseado em RAG para o Regulamento dos Cursos de Graduação da UFRN. Neste capítulo, são discutidas a motivação, os trabalhos relacionados, os objetivos do trabalho e a estrutura geral do documento. Em sequência, o Capítulo 2 trata da fundamentação teórica necessária para o entendimento do tema, abordando tópicos como regulamentos e normas acadêmicas, geração aumentada por recuperação, técnicas avançadas aplicadas e as principais métricas relevantes para avaliação.

O Capítulo 3 detalha a metodologia e as etapas de implementação do sistema proposto. Este capítulo inclui a criação da base de referência, com sua adaptação para uma base customizada; as técnicas de indexação, como segmentação de texto recursiva por caracteres e segmentação semântica; a recuperação de documentos, incluindo métodos como recuperação em múltiplas consultas, a abordagem HyDE, reordenação de documentos e rotas semânticas; e a geração de respostas. Também é descrita a criação do protótipo funcional do chatbot, que integra as diversas etapas anteriores.

O Capítulo 4 apresenta a análise dos resultados obtidos durante a implementação e os testes realizados, discutindo o desempenho das soluções propostas em termos de precisão, relevância e eficiência. Por fim, o Capítulo 5 traz as principais conclusões do trabalho, destacando as contribuições da pesquisa, bem como as limitações e sugestões para trabalhos futuros.

Esta estrutura foi projetada para guiar o leitor de maneira clara e objetiva, desde os conceitos introdutórios até as contribuições práticas e teóricas do trabalho, assegurando uma visão completa do desenvolvimento e resultados do sistema proposto.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica apresentada neste capítulo abrange os conceitos e abordagens fundamentais que embasam este trabalho. Primeiramente, são discutidos os regulamentos e normas acadêmicas, ressaltando sua relevância e desafios no contexto universitário. Em seguida, é apresentada a Geração Aumentada por Recuperação (RAG), destacando sua aplicação na integração de recuperação de informações com modelos de linguagem. Técnicas avançadas de recuperação, como múltiplas consultas e o método HyDE, são exploradas, evidenciando seu papel no aprimoramento da precisão e contextualização das respostas. Por fim, são descritas as métricas utilizadas para avaliar a eficácia dos sistemas, assegurando resultados consistentes e confiáveis.

2.1 Regulamentos e Normas Acadêmicas

As universidades desempenham um papel essencial na organização do ensino superior, guiadas por normas e regulamentos que garantem a autonomia acadêmica, administrativa e financeira. Essa autonomia é assegurada pelo Artigo 207 da Constituição Federal de 1988, que determina: "As universidades gozam de autonomia didático-científica, administrativa e de gestão financeira e patrimonial, e obedecerão ao princípio de indissociabilidade entre ensino, pesquisa e extensão "(CONSTITUIçãO. . . , 1988). A partir dessa prerrogativa, as universidades elaboram seus regulamentos internos, que abrangem desde as diretrizes gerais para cursos e matrículas até normas específicas para avaliação, pesquisa e extensão. No entanto, essa autonomia também implica na necessidade de gestão eficiente e no desenvolvimento de regras claras e acessíveis para todos os membros da comunidade acadêmica. Esses regulamentos, embora essenciais, frequentemente apresentam grande volume e complexidade, dificultando o acesso a informações específicas de forma rápida e precisa. Isso torna evidente a importância de soluções tecnológicas que facilitem a consulta e a compreensão dessas normas, promovendo maior acessibilidade e eficiência no dia a dia universitário.

Na Universidade Federal do Rio Grande do Norte entre os principais instrumentos normativos estão o estatuto, os regimentos, os regulamentos e as resoluções, cada um com funções específicas que visam assegurar o funcionamento harmonioso da instituição. Esses documentos estabelecem as diretrizes gerais, detalham procedimentos internos e regulamentam aspectos específicos da vida acadêmica, promovendo a eficiência e a conformidade às exigências legais e administrativas.

O Estatuto é o documento fundamental que organiza e define as bases de funci-

onamento da universidade, estabelecendo sua natureza jurídica, princípios, objetivos e estrutura administrativa. Ele regula aspectos essenciais, como a autonomia acadêmica, administrativa e financeira da instituição, além de detalhar a organização dos conselhos superiores, reitoria, departamentos e unidades acadêmicas especializadas. O Estatuto serve como referência normativa para a criação de regimentos e regulamentos, garantindo a conformidade com os princípios institucionais e legais, enquanto promove a eficiência na gestão e nas atividades acadêmicas (UFRN, 2011). Dessa forma, o Estatuto outorga três tipos de regimentos. O Regimento Geral define o conjunto de normas que organiza e regula as atividades comuns aos diversos órgãos e serviços da Universidade Federal do Rio Grande do Norte, nos âmbitos administrativo, didático-científico e disciplinar, com o objetivo de complementar e operacionalizar o Estatuto da instituição (UFRN, 2019). O Regimento Interno da Reitoria regula a estrutura e o funcionamento dos órgãos que compõem a Reitoria, detalhando suas competências e responsabilidades, e complementando o Estatuto e o Regimento Geral da Universidade. Ele define conceitos importantes como Reitoria, que é o órgão superior executivo responsável pela direção, administração e coordenação das atividades da universidade, além de estabelecer a organização das Secretarias e Superintendências, essenciais para a gestão eficiente da instituição (UFRN, 2021). Por fim temos regimentos internos de cada Centro Acadêmico, que atuam a nível de curso e tem como objetivos principais o desenvolvimento acadêmico dos cursos de graduação e programas de pós-graduação, integrando ensino, pesquisa e estágio. Além disso, busca fomentar a reflexão crítica e a investigação científica, promovendo a construção de conhecimento, e fortalecer a extensão universitária, aplicando e transferindo o conhecimento gerado para a sociedade, em conformidade com a legislação vigente.

Finalmente, no âmbito dos cursos de graduação, temos o Regulamento dos Cursos Regulares de Graduação que tem como objetivo consolidar, em um único documento, as normas acadêmicas que regem os cursos de graduação oferecidos pela instituição. Esse regulamento estabelece as diretrizes para a execução, o registro e o controle das atividades acadêmicas, que são responsabilidade dos docentes, coordenações de cursos, departamentos acadêmicos, centros acadêmicos, unidades especializadas e da Pró-Reitoria de Graduação (PROGRAD). A PROGRAD tem a função de coordenar essas atividades de forma geral, garantindo que sejam cumpridos os prazos estabelecidos pelo Calendário Universitário. Além disso, o regulamento enfatiza a importância da utilização do sistema oficial de registro e controle acadêmico da UFRN, desenvolvido e mantido pela Superintendência de Informática, para processar todas as rotinas administrativas e acadêmicas de maneira padronizada e eficiente.

Considerando toda a estrutura de documentos apresentados, este trabalho se limita a explorar os artigos e incisos do Regulamento dos Cursos Regulares de Graduação da UFRN e as resoluções internas do curso de Engenharia de Computação. A análise será voltada

para compreender como essas normas estruturam e regulam as atividades acadêmicas, promovem a integração entre ensino, pesquisa e extensão, e garantem a conformidade com os princípios institucionais e legais daqueles documentos que à outorgam. Além disso, busca-se identificar como o acesso à normas pode ser otimizado por meio de soluções tecnológicas, promovendo maior eficiência e acessibilidade no ambiente universitário.

2.2 Geração Aumentada por Recuperação

Nos primeiros estágios do PLN, modelos de n-gramas (MANNING; SCHüTZE, 1999) foram amplamente utilizados para prever palavras ou sequências de palavras com base na frequência de ocorrência em grandes corpora. Esses modelos foram simples, mas eficazes em tarefas como análise de texto e tradução automática. No entanto, com o avanço das técnicas de aprendizado de máquina, especialmente a introdução de modelos neurais, as limitações dos n-gramas, como a falta de capacidade de capturar dependências de longo alcance, se tornaram evidentes. A transição para modelos neurais (BENGIO et al., 2003) representou um avanço crucial, pois os Modelos de Linguagem Neurais (NLMs) começaram a aprender representações contínuas das palavras, superando as limitações dos modelos de n-gramas (ZHAO et al., 2024). Isso abriu caminho para o uso de Redes Neurais Recorrentes (RNNs) e mais tarde para os transformadores, uma arquitetura que revolucionou a forma como as máquinas lidam com sequências de texto. Mais recentemente, modelos como o GPT-3 e o BERT exemplificam a escala e a capacidade de adaptação dos LLMs modernos, que são pré-treinados em grandes quantidades de dados e ajustados para realizar tarefas complexas de PLN (ZHAO et al., 2024).

Esses modelos não apenas avançaram as capacidades de processamento de linguagem, mas também permitiram a implementação de tarefas gerais de linguagem com resultados excepcionais, como a geração de texto e compreensão de contexto, capacitando-os a resolver problemas que anteriormente exigiam soluções mais específicas e complexas. A utilização de LLMs, especialmente os pré-treinados, demonstrou ser altamente eficaz em diversas aplicações, desde a tradução automática até sistemas de recomendação e chatbots.

Contudo, um dos desafios mais notáveis enfrentados pelos LLMs é o fenômeno conhecido como alucinação, no qual os modelos geram informações factualmente incorretas ou completamente inventadas com alto grau de confiança. Esse problema ocorre devido à natureza probabilística dos LLMs, que são projetados para prever sequências de palavras mais prováveis com base nos dados de treinamento, sem necessariamente garantir a veracidade das informações fornecidas. Esse fenômeno surges devido à própria arquitetura do modelo, como também está relacionado à falta de conhecimento ou ao uso de informações desatualizadas (HUANG et al., 2024). Portanto há um grande risco por seu uso indiscriminado em áreas como diagnósticos médicos, serviços financeiros e assistência

jurídica, onde a precisão das respostas é essencial. Esse problema ressalta a necessidade de técnicas complementares, como a abordagem Retrieval-Augmented Generation (RAG), que permite a incorporação de fontes de dados externas em tempo real para mitigar o risco de informações incorretas.

O Retrieval-Augmented Generation (RAG) (LEWIS et al., 2021) é uma técnica que combina abordagens de recuperação de informações com geração de texto baseada em modelos de linguagem, criando um sistema híbrido capaz de oferecer respostas mais precisas e contextualizadas. Enquanto os modelos de linguagem tradicionais, como o GPT, baseiam-se exclusivamente no conhecimento armazenado durante o treinamento, o RAG incorpora informações de uma base de dados externa em tempo real, ampliando significativamente a capacidade de lidar com questões específicas e atualizadas. O uso do RAG é motivado pelas limitações intrínsecas aos modelos de linguagem tradicionais:

- Memória Limitada: Modelos como GPT possuem conhecimento fixo até a data de corte do treinamento, o que os torna ineficientes para lidar com informações recentes ou específicas de domínios especializados.
- Atualização de Dados: Com o RAG, é possível consultar bases de dados dinâmicas, permitindo a atualização contínua das informações usadas para responder às consultas.
- Precisão Contextual: A recuperação de informações direciona o modelo gerador a fornecer respostas mais alinhadas à pergunta do usuário, reduzindo alucinações típicas de modelos puramente generativos.

A arquitetura do RAG combina dois componentes principais que trabalham de forma integrada: o recuperador, *retriever* e o gerador, *generator*. Esses elementos criam um fluxo que utiliza informações externas para enriquecer a geração de respostas, oferecendo precisão e relevância mesmo em contextos especializados. O recuperador é responsável por buscar documentos relevantes em uma base de dados. Ele utiliza técnicas de recuperação de informações, com embeddings vetoriais, que analisam a semântica das consultas em vez de se limitarem à correspondência exata de palavras-chave. Essa abordagem é particularmente útil para encontrar informações com significado próximo ao da consulta, mas expressas de forma diferente. Ferramentas como FAISS (JOHNSON; DOUZE; JéGOU, 2017), amplamente reconhecida por sua eficiência na busca vetorial, e ElasticSearch, usada para buscas textuais e semânticas, são comuns na implementação dessa etapa (KARPUKHIN et al., 2020).

Após a recuperação, os documentos encontrados são enviados ao gerador, que utiliza um modelo de linguagem pré-treinado, como GPT ou T5, para formular uma resposta baseada nas evidências recuperadas. Este modelo combina o conhecimento armazenado em

seus parâmetros durante o treinamento com os documentos recuperados, criando respostas contextualizadas e específicas (LEWIS et al., 2021). O processo de geração é essencial para transformar os documentos brutos em informações compreensíveis e úteis para o usuário.

Como mostrado na Figura 2, o fluxo do RAG pode ser descrito em etapas três ou quatro principais. Primeiro, o usuário faz uma pergunta ou consulta em linguagem natural. Essa entrada é processada pelo recuperador, que busca os documentos mais relevantes em uma base de dados. A relevância é determinada por métodos como similaridade coseno ou algoritmos de busca semântica, que analisam a proximidade entre o vetor da consulta e os documentos indexados. Em seguida, o gerador utiliza as informações recuperadas como contexto adicional para criar uma resposta coerente e fundamentada. Opcionalmente, um processo de pós-tratamento pode ser aplicado para corrigir possíveis inconsistências, garantir a formatação adequada ou eliminar alucinações geradas pelo modelo.

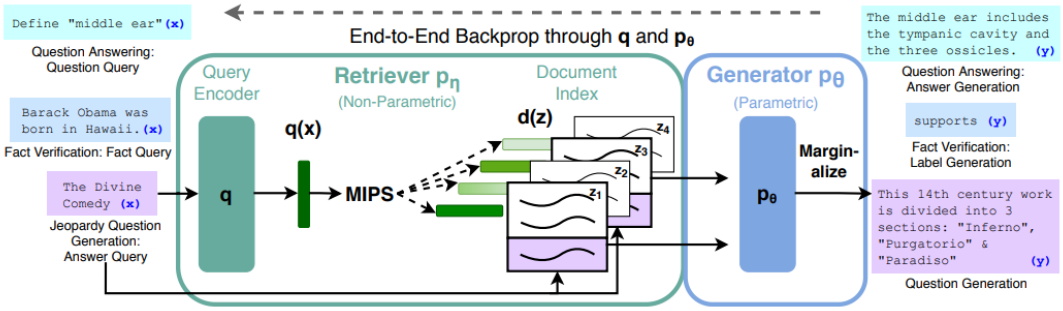

Figura 2 – Etapas do Retrieval-Augmented Generation.

Fonte: Lewis et al. (2021, p. 2).

A integração desses componentes torna o RAG uma solução poderosa para responder perguntas em domínios especializados ou com informações em constante atualização. A ferramenta Langchain ofereceu o suporte modular para implementar esse pipeline de maneira eficiente, integrando recuperação de informações e geração de respostas em uma única estrutura.

Com o avanço dessa tecnologia, surgiram diferentes paradigmas de implementação, cada um com características e níveis crescentes de sofisticação: o *Naive RAG*, que estabelece os fundamentos da integração entre recuperação e geração; o *Advanced RAG*, que aprimora os processos com técnicas mais robustas de recuperação e organização de dados; e o *Modular RAG*, que oferece flexibilidade e adaptabilidade por meio de uma arquitetura modular. Esses métodos refletem a progressão tecnológica no uso do RAG, adaptando-se a diferentes cenários e demandas de processamento de informações (GAO et al., 2024).

O Naive RAG é o paradigma mais básico e segue uma abordagem linear. Ele consiste nas principais etapas: indexação, recuperação e geração. Durante a indexação, os documentos são convertidos em chunks e armazenados em um banco de dados vetorial. Na recuperação, os chunks mais relevantes para a consulta do usuário são selecionados

com base em similaridade semântica. Por fim, na geração, os chunks recuperados são combinados com a consulta inicial para gerar a resposta. No entanto, essa abordagem apresenta limitações significativas, como dificuldade em selecionar informações precisas e relevantes, tendência à redundância e suscetibilidade à geração de respostas não suportadas pelos dados recuperados. O Advanced RAG introduz melhorias para superar as deficiências do Naive RAG. Ele otimiza os processos de pré e pós-recuperação, refinando técnicas de indexação, como a utilização de janelas deslizantes, metadados e segmentação mais granular. Também emprega métodos de reclassificação e compressão de contexto após a recuperação, buscando organizar e priorizar as informações mais relevantes. Essas estratégias tornam o Advanced RAG mais robusto, reduzindo a dependência de consultas originais imprecisas e minimizando ruídos nos resultados. Por fim, o Modular RAG representa um avanço significativo, oferecendo uma arquitetura flexível e adaptável. Ele introduz módulos específicos, como mecanismos de memória, roteamento semântico e adaptação de tarefas, permitindo maior personalização e integração com outras tecnologias, como aprendizado por reforço (GAO et al., 2024). Diferentemente das abordagens anteriores, o Modular RAG não se limita a uma sequência fixa de etapas, podendo adotar fluxos iterativos e adaptativos para recuperação e geração. Essa modularidade facilita a substituição ou reconfiguração de componentes, tornando-o mais versátil para uma ampla variedade de tarefas.

Neste trabalho, será enfatizado o uso do Advanced RAG, com foco em técnicas de pré-recuperação para otimizar a indexação e a formulação de consultas. Estratégias como segmentação refinada, uso de metadados e reescrita de consultas serão aplicadas para garantir maior relevância e precisão na recuperação de informações, aprimorando o desempenho do sistema de geração.

2.3 Técnicas avançadas

A etapa de recuperação em sistemas de RAG desempenha um papel crucial ao localizar informações relevantes em grandes bases de dados não estruturados. Sendo assim, a escolha da técnica para medir distâncias é essencial para determinar a similaridade entre esses *embeddings* descritos no DPR em sistemas de recuperação de informações. Entre as mais utilizadas estão a distância euclidiana, que avalia a separação linear entre pontos em um espaço vetorial, e a similaridade de cosseno, amplamente empregada em cenários onde a direção dos vetores é mais relevante que sua magnitude. Pesquisas voltadas para a otimização de buscas, como as relacionadas ao FAISS (JOHNSON; DOUZE; JéGOU, 2017), destacam a eficiência no uso de índices vetoriais para calcular a distância euclidiana. Por outro lado, estudos como o SBERT (REIMERS; GUREVYCH, 2019) enfatizam a relevância da similaridade de cosseno em tarefas de recuperação semântica, mostrando

sua robustez em contextos de geração de respostas e recuperação aumentada. Estratégias híbridas que combinam diferentes métricas têm ganhado destaque por proporcionar ganhos adicionais de precisão e eficiência, especialmente em sistemas de RAG. A medida que buscamos melhorar a precisão e a eficiência do processo de recuperação, técnicas avançadas de recuperação emergem como soluções essenciais. Essas abordagens não apenas ampliam a capacidade do modelo de acessar informações mais relevantes, mas também otimizam o tempo de resposta e aprimoram a qualidade do conteúdo recuperado, tornando-as fundamentais para o sucesso de sistemas RAG mais sofisticados.

Dessa forma, a técnica de múltiplas consultas, que gera variantes de uma mesma pergunta, é uma abordagem utilizada para melhorar a recuperação de informações em sistemas de RAG (LI et al., 2024). Essa estratégia aumenta a diversidade semântica das consultas, permitindo que documentos relevantes sejam identificados de forma mais abrangente, mesmo quando a consulta inicial é limitada em termos de expressividade. São introduzidas quatro estratégias distintas para reescrita de consultas, ajustando os níveis de informação presentes em cada variante. Entre essas estratégias estão a reescrita geral, que remove ruídos da consulta original, e a expansão de informações, que utiliza pseudo-respostas para enriquecer semanticamente as consultas. Essas abordagens demonstraram ser eficazes tanto na recuperação de documentos quanto na geração de respostas, especialmente em experimentos conduzidos em cenários acadêmicos e industriais. Além disso, o trabalho propõe um método adaptativo de seleção de estratégias de reescrita, reduzindo o número de reescritas necessárias enquanto otimiza a diversidade dos documentos recuperados. Essa adaptação é essencial para minimizar o ruído e aumentar a relevância das respostas geradas, um avanço significativo em relação às abordagens tradicionais de consulta única ou reescritas pouco diversificadas. A validação experimental do DMQR-RAG, conduzida com grandes modelos de linguagem como GPT-4, mostrou melhorias substanciais em métricas de recuperação, como a taxa de acerto no Top-5 e precisão, em comparação com métodos existentes, como o RAG-Fusion.

Já o Hypothetical Document Embeddings (HyDE) é uma abordagem para recuperação densa em cenários em que um modelo é aplicado a tarefas ou domínios novos sem ter sido previamente treinado ou ajustado com dados específicos desses contextos. A implementação combina a geração de documentos hipotéticos por modelos de linguagem que seguem instruções, com a codificação de similaridade baseada em aprendizado contrastivo não supervisionado. Essa técnica elimina a necessidade de rótulos de relevância, pivotando a tarefa de modelagem da relevância para um modelo generativo que cria documentos sintéticos baseados na consulta inicial. Esses documentos, mesmo podendo conter informações factualmente incorretas, capturam padrões de relevância. Posteriormente, são codificados em vetores densos por um modelo de *embedding*, que filtra os detalhes irrelevantes, permitindo a recuperação de documentos reais semanticamente similares. Os

experimentos demonstram que o HyDE supera sistemas não supervisionados e alcança desempenho comparável a modelos supervisionados em diversas tarefas, como busca na web, perguntas e respostas, e verificação de fatos, cobrindo diferentes idiomas (GAO et al., 2022). Além disso, o modelo é prático em sistemas de busca emergentes, onde o HyDE pode ser utilizado inicialmente e, posteriormente, a fim de ampliar o contexto da resposta.

Outra importante técnica é o *reranking* que se mostra como uma abordagem amplamente utilizada para melhorar a relevância e a precisão na recuperação de informações em sistemas que combinam modelos generativos e de recuperação. O modelo Re2G (GLASS et al., 2022) apresenta uma metodologia que integra recuperação inicial, reranking e geração baseada em sequências, aplicando um modelo baseado em BART. A técnica de reranking desempenha um papel crucial ao reordenar os resultados de recuperação inicial para identificar os itens mais relevantes com base em critérios específicos. Isso é especialmente valioso em sistemas que utilizam métodos híbridos, como a combinação de BM25 e recuperação neural, permitindo a fusão de resultados provenientes de fontes com métricas de pontuação incomparáveis. O reranking utiliza modelos avançados, como o BERT, para analisar pares de consulta e passagem de forma interativa, proporcionando ganhos significativos em tarefas como preenchimento de lacunas, verificação de fatos, resposta a perguntas e diálogos, conforme avaliado no benchmark KILT. Além disso, o impacto do reranking vai além da recuperação de informações, afetando diretamente a qualidade da geração de texto em sistemas que dependem de conhecimento externo. O Re2G exemplifica como o reranking, em conjunto com técnicas como distilação de conhecimento e treinamento, pode melhorar a precisão na seleção de passagens e, consequentemente, elevar o desempenho geral do modelo em diversos contextos.

Por fim, o roteamento semântico é uma técnica inovadora que busca aprimorar a interação entre modelos de linguagem de larga escala (LLMs) e sistemas automatizados por meio da definição de rotas explícitas com base no significado semântico de entradas textuais. O roteamento semântico fornece uma camada de decisão determinística, permitindo que entradas sejam processadas e redirecionadas para ações específicas com alta precisão. Essa abordagem se diferencia de arquiteturas baseadas em *prompting* ao mitigar problemas comuns de alucinações de LLMs, tornando-a mais confiável e eficiente em aplicações práticas (MANIAS; CHOUMAN; SHAMI, 2024). No contexto do roteamento semântico, os enunciados desempenham um papel crucial ao gerar representações vetoriais das entradas textuais, permitindo que a semelhança entre textos seja medida por métricas de distância, como a similaridade de cosseno. Ferramentas como o `text-embedding-ada-002` da OpenAI e o `all-MiniLM-L6-v2` da Hugging Face são amplamente utilizadas para essa finalidade. A escolha do enunciado pode impactar significativamente o desempenho do roteador, dependendo do tamanho das entradas textuais e da granularidade semântica requerida (MANIAS; CHOUMAN; SHAMI, 2024). Outro conceito relevante é a utilização

de bancos vetoriais para armazenar os enunciados. Esses bancos permitem que exemplos predefinidos sejam utilizados para generalizar decisões em novos contextos, ampliando a aplicabilidade do roteador. Adicionalmente, o uso de camadas dinâmicas no roteamento facilita a escalabilidade, permitindo que novos cenários sejam incorporados ao sistema sem reestruturar o modelo base. Para melhorar ainda mais a eficiência do sistema, técnicas de quantização de modelos têm sido empregadas, reduzindo os requisitos de recursos computacionais sem comprometer significativamente o desempenho. A integração de roteadores semânticos em sistemas de Recuperação Aumentada por Geração apresenta um potencial significativo para aprimorar a precisão de chatbots em domínios especializados. Ao empregar essas técnicas, é possível criar chatbots que interpretam perguntas de maneira mais contextual e fornecem respostas alinhadas às necessidades do usuário.

2.4 Métricas relevantes

A avaliação de sistemas RAG envolve analisar os componentes de recuperação, geração e o sistema como um todo, cada um apresentando desafios específicos. Na recuperação, destaca-se a dificuldade de avaliar a precisão e recuperação em bases de dados dinâmicas e diversas, considerando também aspectos temporais, imprecisão na separação do texto de referência e fontes de baixa qualidade (ES et al., 2023). Já na geração, o desafio está em medir a fidelidade e relevância das respostas geradas por modelos de linguagem, especialmente em tarefas interpretação de regras e normas (RYU et al., 2023). A avaliação do sistema como um todo requer analisar a sinergia entre as técnicas recuperação e geração, incluindo métricas que considerem a qualidade das respostas, latência e capacidade de lidar com consultas complexas. A falta de critérios abrangentes e *datasets* representativos reais dificulta a criação de *benchmarks* completos, evidenciando a necessidade de métricas específicas para tarefas e preferências humanas (YU et al., 2024). Dividir as métricas entre recuperação e geração é essencial para avaliar cada componente de forma independente e precisa. Essa separação permite identificar pontos específicos de melhoria em cada etapa, garantindo que os documentos relevantes sejam encontrados e que as respostas geradas sejam coerentes e úteis. Essa abordagem estruturada permite evidenciar possíveis pontos de melhoria em cada parte do sistema bem como focar naquelas que são de maior interesse do desenvolvedor.

Para avaliar a etapa recuperação de informações, poderam ser considerados nesse trabalho duas métricas, que são elas:

- *Context Recall*: Mede quantos dos documentos ou informações relevantes foram recuperados com sucesso. Ele foca em evitar a omissão de resultados importantes, ou seja, quanto maior o *recall*, menor é a chance de deixar informações relevantes de fora. Para calcular o *Context Recall*, é necessário ter os contextos recuperados e os

contextos de referência, usando métodos baseados em similaridade para avaliar a relevância entre eles.

$$\text{Context Recall} = \frac{|\text{Contextos Relevantes Recuperados} \cap \text{Contextos de Referência}|}{|\text{Contextos Relevantes Recuperados}|}$$

- *Context Precision*: Avalia a precisão do sistema em identificar e classificar documentos relevantes em comparação com outros candidatos menos relevantes ou irrelevantes. A precisão é calculada como a razão entre o número de chunks relevantes até a posição k e o número total de chunks avaliados até k. Essa métrica é útil para avaliar a qualidade dos contextos recuperados, verificando se os resultados são realmente úteis e relevantes.

$$\text{Context Precision@K} = \frac{\sum_{k=1}^{K}(\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top } K \text{ results}}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{\text{true positives@k} + \text{false positives@k}}$$

Após a etapa de recuperação, e ao considerar que somente documentos relevantes foram recuperados, pode-se avaliar a etapa de geração das respostas ao usuário de forma isolada, sem que erros da primeira etapa possam interferir no resultado. Poderão ser evidenciadas as métricas de fidelidade, corretude e também de relevância, como mostrado abaixo:

- Faithfulness: O Faithfulness, ou fidelidade, avalia a consistência factual de uma resposta gerada em relação ao contexto fornecido. Essa métrica é usada para medir o quanto as informações na resposta são consistentes com o que pode ser inferido a partir do contexto recuperado. Uma resposta é considerada fiel se todas as afirmações feitas nela puderem ser inferidas diretamente do contexto fornecido. O cálculo da métrica de fidelidade envolve, primeiramente, identificar as afirmações presentes na resposta gerada. Em seguida, cada uma dessas afirmações é analisada para verificar se pode ser inferida a partir do contexto fornecido. Por fim, a pontuação de fidelidade é determinada como a proporção de afirmações que podem ser inferidas em relação ao total de afirmações presentes na resposta.

$$\text{Fidelidade} = \frac{\text{Número de afirmações inferíveis}}{\text{Total de afirmações na resposta}}$$

- *Response Relevancy*: Avalia o quão pertinente é a resposta gerada em relação à pergunta original. Respostas incompletas ou que contêm informações redundantes recebem pontuações mais baixas, enquanto respostas mais completas e relevantes obtêm pontuações mais altas. Essa métrica utiliza o texto da pergunta original, os contextos recuperados e a resposta gerada para o cálculo. O processo baseia-se na ideia de que uma resposta relevante deve permitir a geração de perguntas artificiais que se alinhem semanticamente à pergunta original. Isso é alcançado ao solicitar a um modelo de linguagem que gere múltiplas variantes da pergunta com base na resposta, medindo em seguida a similaridade entre essas perguntas geradas e a pergunta original. A relevância da resposta é calculada como a média das similaridades cossenos entre os embeddings das perguntas geradas e o embedding da pergunta original. Embora na prática os valores dessa métrica variem entre 0 e 1, isso não é garantido matematicamente, uma vez que a similaridade cosseno pode assumir valores entre -1 e 1. A equação pode ser descrita como:

$$\text{answer relevancy} = \frac{1}{N} \sum_{i=1}^{N} cos(q_i, q_o)$$

  Onde *qi* é o embedding da i-ésima pergunta gerada, *q0* é o embedding da pergunta original, e N é o número de perguntas geradas, cujo valor padrão no *framework* é 3.

3 DESENVOLVIMENTO

O desenvolvimento do sistema foi estruturado em etapas cuidadosamente planejadas para garantir uma solução eficiente e adaptada às necessidades do domínio dos documentos de regulamentação. A primeira etapa consistiu na criação da base de referência, onde o regulamento acadêmico foi coletado, organizados e preparado para facilitar sua utilização posterior no sistema. Os textos foram pré-processados para padronização e limpeza, removendo elementos desnecessários, como formatações excessivas, e convertendo-os para um formato estruturado que pudesse ser indexado eficientemente. Além disso, foi realizada uma adaptação para lidar com a natureza personalizada da base, incorporando metadados que identificam o contexto, como capítulos, seções e artigos, permitindo um mapeamento semântico mais preciso durante as etapas de recuperação. A indexação posterior foi baseada nesse corpus estruturado e ajustado, que serve como o núcleo de informações do sistema. Essa preparação inicial foi essencial para garantir que os dados fossem acessíveis, confiáveis e devidamente formatados, formando a base sólida sobre a qual as demais etapas do sistema foram construídas.

Na segunda etapa, voltada para a indexação dos documentos, o foco foi estruturar o corpus previamente preparado de forma que o sistema pudesse localizar rapidamente os trechos mais relevantes durante o processo de recuperação. Foram empregadas duas estratégias de segmentação: a segmentação recursiva por caracteres e a segmentação semântica. A segmentação recursiva por caracteres foi utilizada para garantir que os documentos fossem divididos em blocos de tamanho fixo, respeitando limites predefinidos de caracteres, como exigências impostas por algumas ferramentas de recuperação. Essa abordagem assegura que os trechos mantivessem uma granularidade uniforme, facilitando a recuperação rápida e eficiente de informações específicas. Já a segmentação semântica focou em dividir os textos com base em seus significados e contextos. Para isso, um algoritmo foi implementado, identificando quebras naturais no texto, como seções, parágrafos ou tópicos, de forma a preservar a coerência de cada segmento e evitar divisões que pudessem fragmentar o sentido das informações.

A terceira etapa do desenvolvimento focou na recuperação das informações, empregando uma combinação de técnicas para garantir precisão e contextualização das respostas. A utilização de múltiplas consultas foi projetada para abordar a complexidade dos regulamentos acadêmicos, que frequentemente apresentam informações fragmentadas ou organizadas em diferentes seções. Ao empregar a técnica de multiplas consultas, baseadas em reformulações da pergunta original do usuário, tentou-se cobrir diferentes interpretações ou nuances da consulta, buscando apliar a cobertura e aumentar a probabilidade

de recuperar os trechos mais relevantes. A geração de respostas hipotéticas (HyDE) foi empregada para fortalecer o sistema em casos onde os documentos originais apresentavam lacunas. Nesse caso, o sistema gera hipóteses informadas, que são utilizadas como uma conexão semântica para melhorar a correspondência entre a consulta e os documentos, especialmente em cenários de linguagem ambígua ou subespecificada. Além disso, foi implementada a técnica de reranking, que reorganiza os documentos inicialmente recuperados com base em critérios como relevância semântica, proximidade contextual e similaridade com a consulta. Isso assegura que os documentos mais relevantes sejam priorizados na entrega da resposta. Por fim, o roteamento semântico foi introduzido para filtrar consultas simples, como saudações e despedidas, redirecionando essas interações diretamente, sem acionar o modelo de linguagem para processamento completo, e garantindo que os recursos fossem priorizados para consultas mais complexas e contextualmente relevantes. O uso dessas técnicas em conjunto foi motivado pela necessidade de lidar com um domínio rico e complexo, como regulamentos acadêmicos, assegurando que o chatbot oferecesse respostas claras, consistentes e contextualmente apropriadas aos usuários.

A quarta etapa do desenvolvimento foi dedicada à geração das respostas, um componente essencial para garantir que as informações recuperadas fossem apresentadas de forma clara, precisa e adaptada ao contexto da consulta do usuário. Durante essa fase, foram implementados testes com diferentes modelos de geração para avaliar a capacidade de cada abordagem em lidar com os desafios específicos do domínio. A principal razão para essa etapa foi a necessidade de comparar modelos com diferentes tamanhos e arquiteturas, como modelos menores, que poderiam ser mais rápidos e eficientes para respostas simples, e modelos maiores, que poderiam apresentar maior capacidade de contextualização e detalhamento para consultas mais complexas.

Por fim, a etapa final do desenvolvimento envolveu a criação de uma arquitetura em nuvem para garantir escalabilidade, disponibilidade e eficiência no funcionamento do chatbot. Foi utilizada a AWS como provedor de infraestrutura, aproveitando serviços como o Bedrock, para execução e gerenciamento do modelo de linguagem, e o Qdrant, para armazenamento vetorial e recuperação eficiente de embeddings. O gerenciamento das requisições foi realizado utilizando o API Gateway em conjunto com o AWS Lambda, que processava as consultas de forma serverless, garantindo baixa latência e escalabilidade. O DynamoDB foi empregado para o armazenamento de metadados estruturados e histórico de interações, enquanto o CloudWatch foi configurado para monitoramento em tempo real, permitindo o rastreamento de métricas de desempenho, detecção de falhas e análise de logs. Para a interface de comunicação com os usuários, foi escolhido o Telegram, devido à sua popularidade e suporte a bots altamente personalizáveis. O bot no Telegram foi configurado para se comunicar diretamente com a API hospedada na AWS, enviando as consultas e recebendo as respostas em tempo real. Essa combinação de serviços e tecnologias resultou

em uma solução robusta e eficiente, capaz de escalar para atender a múltiplos usuários simultaneamente, garantindo alta disponibilidade e confiabilidade, ao mesmo tempo em que oferecia uma interface amigável e de fácil acesso.

3.1 Criação da Base de Referência

O primeiro passo na criação da base de referência foi a coleta e organização do Regulamento dos Cursos de Graduação. Esse documento foi escolhido de forma a não conter o histórico de artigos revogados anteriormente, garantindo que todos os textos estivessem em um formato unificado, sem duplicação aparente. Para facilitar o processamento subsequente, os documentos foram convertidos em texto bruto utilizando o PyPDF2, assegurando que nenhum conteúdo relevante fosse perdido. Em seguida, o conteúdo foi segmentado em seções menores, contendo artigos para criar um mapeamento estruturado que vinculasse cada segmento a seus metadados, incluindo título, número de artigo e seção. Essa estrutura organizada foi essencial para suportar a indexação e recuperação nos passos posteriores. Então a continuação dessa etapa se deu ao utilizar o *framework* RAGAS (ES et al., 2023). Dado que a ferramenta não suporta diretamente o idioma português, foi necessário traduzir os documentos para o inglês. Essa etapa foi realizada utilizando a API GPT4o-mini, que oferece capacidades avançadas de tradução preservando o significado semântico do texto. Para garantir a qualidade da tradução, foram implementados passos intermediários de validação. Cada seção traduzida foi revisada automaticamente, comparando com a similaridade semântica entre os textos em português e inglês utilizando uma ferramenta de *embedding* da Amazon, o Titan Text. O objetivo foi identificar inconsistências ou perdas de contexto, garantindo que o conteúdo traduzido fosse fiel ao original e adequado para a geração de perguntas no idioma suportado pelo framework. Um limiar de 0.95 foi considerado ao analisar a similaridade semântica dos pares de texto.

Com os documentos traduzidos, iniciou-se a etapa de geração dos pares de perguntas e respostas usando o *Testset Generator* do RAGAS. O framework foi configurado para gerar perguntas com diferentes níveis de dificuldade, abrangendo questões simples, complexas e conversacionais. Como mostrado na Figura 3, esse processo utilizou a abordagem evolutiva, na qual as perguntas simples eram transformadas em versões mais elaboradas por meio de três estratégias:

- Raciocínio: Reescrita das perguntas para exigir respostas analíticas, como justificativas baseadas em múltiplos trechos do texto.
- Condicionamento: Inclusão de cenários hipotéticos ou premissas que tornassem a pergunta dependente de contextos específicos.

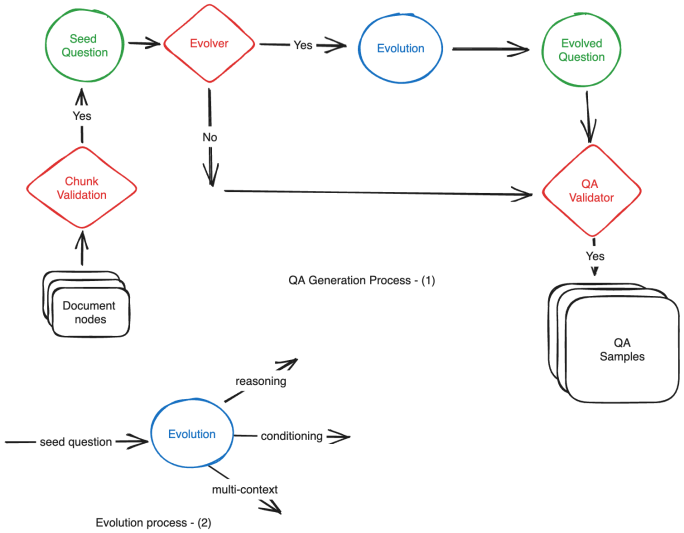

Figura 3 – Processo de Criação da Base de Dados no RAGAS.

Fonte: Es et al. (2024, p. 2).

- Múltiplos Contextos: Formulação de perguntas que exigissem informações de diferentes seções do documento, promovendo uma recuperação mais robusta.

Além disso, p *framework* também gera perguntas simulando interações conversacionais, com diálogos encadeados que permitiam avaliar a habilidade do sistema em gerenciar múltiplos turnos de consulta. Essa etapa resultou em 297 pares iniciais de perguntas e respostas vinculados a trechos específicos dos documentos e seus respectivos metadados.

Após a geração dos pares em inglês, foi realizada a tradução reversa para o português, utilizando novamente o GPT4o-mini. Essa etapa teve como objetivo garantir que o dataset estivesse adaptado ao idioma de interesse, preservando o significado semântico das perguntas e respostas originais. Para evitar problemas de inconsistência, a tradução foi validada com técnicas automatizadas de similaridade semântica e verificações manuais amostrais. O resultado foi um dataset em português para ser utilizado em avaliações e implementações do sistema. Uma das dificuldades vencidas veio a partir API GPT4o que enfrentava restrições de tokens por minuto, o que impactava a fluidez do processo de geração. Para contornar essa limitação, foi desenvolvido um limitador de requisições. O limitador gerenciava o envio de requisições à API, estabelecendo um intervalo de cinco segundos entre as chamadas e evitando bloqueios. Além disso, o código foi ajustado para registrar logs das requisições em tempo real, permitindo monitorar o progresso da geração e identificar gargalos ou falhas na comunicação com a API.

O dataset gerado foi consolidado em um formato estruturado, com cada pergunta associada aos seguintes elementos:

- Texto da Pergunta: A questão em português formulada no processo de geração.

- Contexto Utilizado: O(s) trecho(s) do regulamento utilizados para gerar a pergunta e resposta.
- Resposta de Referência: A resposta correspondente, que servirá como base para avaliações do sistema.
- Metadados: Informações como a seção, o artigo e a localização do conteúdo original.

O formato final foi definido em JSON, garantindo compatibilidade com ferramentas de indexação e recuperação posteriores. Essa estruturação foi essencial para suportar o pipeline do sistema RAG, permitindo a avaliação e validação do modelo em cenários controlados e replicáveis.

3.1.1 Adaptação para base customizada

Considerando a base de dados gerada pelo RAGAS, houve análise dos 297 pares de perguntas e respostas gerados automaticamente a partir do documento de referêencia. Este processo envolveu uma triagem manual e automatizada para avaliar a qualidade e a adequação dos pares ao contexto acadêmico. Durante essa triagem, foram aplicados critérios de relevância, verificando se as perguntas refletiam dúvidas plausíveis de estudantes de graduação e se as respostas fornecidas estavam consistentes com os trechos normativos associados. Após essa análise, apenas 112 pares foram considerados adequados para compor a base final de referência. Um problema recorrente identificado foi a presença de erros de tradução, como o caso do termo "Regime de Acompanhamento Acadêmico", traduzido erroneamente como "Regime de Monitoramento Acadêmico". Para resolver isso, foi necessário recuperar os textos originais dos regulamentos e revisar detalhadamente os pares gerados. O processo começou com a identificação automática de trechos em que palavras-chave indicavam possíveis erros de escrita. Após essa identificação, os textos foram comparados com suas traduções para verificar perdas de significado ou alterações de contexto. Utilizou-se uma abordagem mista de revisão manual e ferramentas de validação semântica, como embeddings, para garantir a fidelidade ao conteúdo original. Então os pares corrigidos foram novamente vinculados aos trechos correspondentes nos regulamentos.

Com a identificação de inconsistências e a necessidade de respostas mais alinhadas ao contexto linguístico e cultural dos regulamentos acadêmicos, foi realizada uma nova iteração no processo de geração de de respostas utilizando a API GPT4o. Desta vez, foi utilizado um prompt exclusivamente em português, ajustado para priorizar o uso de termos e expressões características do ambiente acadêmico. A construção do *prompt* envolveu testes iterativos para otimizar as instruções fornecidas ao modelo, garantindo que as perguntas geradas fossem específicas e as respostas, contextualizadas e completas. Essa etapa mitigou problemas semânticos e aprimorou a aderência às necessidades do

público-alvo. Após a geração e revisão das das respostas, foi implementada uma etapa de estruturação do dataset em formato JSON e CSV. Cada par foi novamente vinculado aos seguintes elementos: a pergunta gerada, o contexto textual utilizado como base, a resposta de referência e metadados associados, como a seção, o artigo e a localização do conteúdo no regulamento. Essa estrutura permitiu que os pares fossem facilmente indexados e recuperados nas etapas posteriores do sistema.

3.2 Indexação de Documentos

Uma das etapas cruciais no desenvolvimento de sistemas baseados em RAG é a fragmentação, ou segmentação, de textos extensos em partes menores e gerenciáveis, tarefa que é realizada por divisores de texto, ou *text splitters*. Esses divisores permitem que documentos complexos, como regulamentos e normas institucionais, sejam segmentados em blocos estruturados, preservando a coerência semântica necessária para consultas precisas sem exceder os limites das janelas de contextos de LLMs. No contexto deste trabalho, a aplicação de divisores de texto foi essencial para otimizar o desempenho do sistema de perguntas e respostas (QA), permitindo a indexação eficiente e reduzindo o custo computacional durante a busca por informações. Para isso, foram implementados dois tipos de divisores: a segmentação de texto recursiva por caracteres e segmentação semântica.

3.2.1 Segmentação de Texto Recursiva por Caracteres

A implementação da Segmentação de Texto Recursiva por Caracteres foi realizada utilizando exclusivamente o LangChain e suas classes específicas de divisão de texto, como *CharacterTextSplitter*. O objetivo principal dessa abordagem foi dividir os documentos extensos de maneira eficiente e dinâmica, ajustando os fragmentos ao limite de entrada dos modelos de linguagem, sem comprometer a coesão semântica dos conteúdos. O processo foi estruturado em etapas progressivas, considerando os diferentes níveis de granularidade fornecidos pelos delimitadores.

A primeira etapa consistiu na configuração inicial do CharacterTextSplitter, definindo parâmetros como o tamanho máximo dos blocos (*chunk_size*) e o tamanho da sobreposição entre blocos consecutivos (*chunk_overlap*). Os valores utilizados de tamanho máximo dos blocos foram ajustados para 200, 500, 1000 e 2000 caracteres, dependendo do nível de granularidade desejado. Essa flexibilidade foi importante para balancear precisão e eficiência, já que chunks menores permitem maior detalhamento na recuperação de informações, enquanto chunks maiores reduzem o número total de fragmentos gerados. Com a configuração estabelecida, o processo de segmentação foi iniciado diretamente nos textos do regulamento. A divisão foi realizada iterativamente, com o *CharacterTextSplitter* aplicando delimitadores baseados em caracteres específicos, como quebras de linha ou

pontuação, que indicam a presença de sentenças ou parágrafos. A hierarquia de divisões foi definida de forma a começar pelos delimitadores mais amplos, como parágrafos, e refinar progressivamente para sentenças e palavras, caso os blocos excedam o limite de caracteres definido. Essa hierarquia foi controlada pelos métodos nativos do LangChain, que permitem ajustar o comportamento do *splitter* conforme os critérios de tamanho e delimitadores.

Uma característica essencial da implementação foi o uso da sobreposição de 50 caracteres entre os chunks consecutivos. Isso garantiu que o contexto das extremidades de um bloco fosse preservado no próximo, reduzindo a chance de perda de informações relevantes durante a segmentação. A sobreposição foi configurada diretamente na classe CharacterTextSplitter por meio do parâmetro *chunk_overlap*, permitindo que o LangChain gerenciasse automaticamente a inserção de partes sobrepostas nos blocos adjacentes. Durante o processo, foi realizada uma validação automatizada para garantir que os chunks gerados fossem semanticamente coesos e respeitassem os limites de entrada dos modelos de linguagem. Isso incluiu a análise de trechos problemáticos, como frases incompletas ou divisões em locais inadequados. Quando necessário, os parâmetros do *CharacterTextSplitter* foram ajustados para melhorar a segmentação em casos específicos, como documentos com formatações mais complexas.

Ao final do processo, os fragmentos segmentados foram armazenados em um formato estruturado JSON com metadados associados, incluindo informações sobre o número do *chunk*, o contexto do texto original e os limites de caracteres utilizados. Essa estruturação garantiu compatibilidade com os demais componentes do pipeline de recuperação de informações e permitiu a integração direta dos chunks segmentados no sistema de busca. A abordagem com o *LangChain* proporcionou uma implementação eficiente e adaptável, capaz de lidar com diferentes tamanhos de texto e cenários de segmentação.

O gráfico apresentado na Figura 4 evidencia essa relação inversa entre o tamanho do *chunk* e a quantidade total de *chunks* gerados a partir do regulamento de graduação da UFRN. À medida que o tamanho do chunk aumenta (de 200 para 2000 caracteres), a quantidade de chunks diminui, de 1190 para apenas 140. Isso ocorre porque *chunks* maiores conseguem abarcar mais conteúdo em cada divisão, reduzindo o número total de fragmentos necessários para representar o texto completo. *Chunks* maiores melhoram a eficiência da busca, pois há menos fragmentos a processar, mas podem reduzir a precisão da recuperação de informações, uma vez que as respostas podem conter conteúdo irrelevante junto ao contexto relevante.

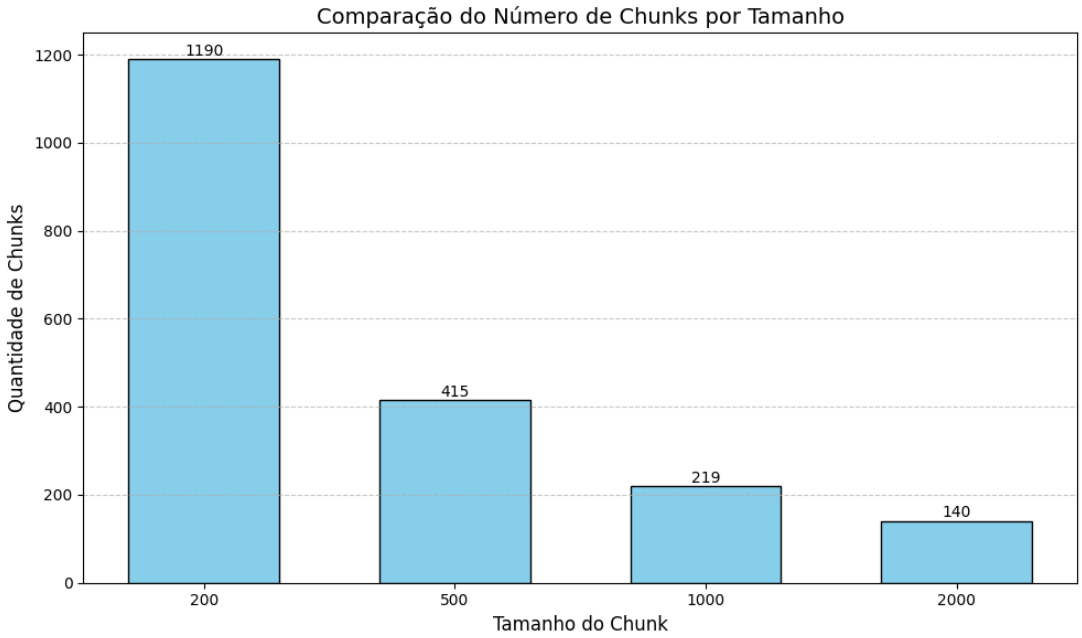

Figura 4 – Quantidade de chunks por tamanho

Fonte: Autoria Própria

3.2.2 Segmentação Semântica

A segmentação dos textos regulamentares foi complementada utilizando *parsers* personalizados que exploraram padrões textuais típicos do documento normativo. Esses padrões incluíam não apenas os artigos identificados por "Art. X"ou parágrafos destacados por "§ Y", mas também os elementos estruturais superiores, como "TÍTULO", "Seção"e "Subseção". Para automatizar a identificação dessas partes estruturais, foram utilizadas expressões regulares (Regex) cuidadosamente projetadas para localizar e capturar esses elementos em diferentes formatos e estilos de formatação presentes nos documentos.

O processo teve início com a análise do *layout* dos regulamentos, determinando os formatos consistentes e eventuais inconsistências na apresentação de títulos, seções e subseções. Expressões regulares específicas foram criadas para cada nível hierárquico: uma Regex foi desenvolvida para identificar títulos destacados em letras maiúsculas precedidas ou seguidas de espaços ou linhas em branco; outra para localizar seções iniciadas por "Seçao X", com "X"representando números romanos ou cardinais; e, finalmente, uma para subseções, que geralmente seguem a formatação "Subseção X". Após configurar essas Regex, o *parser* foi estruturado para varrer o documento de forma sequencial, registrando as informações hierárquicas associadas a cada bloco de texto encontrado. Durante a execução, o *parser* não apenas segmentava os artigos identificados por "Art. X", mas também atribuía os metadados extraídos da hierarquia do documento. Cada artigo segmentado foi vinculado ao título, seção e subseção mais próximos identificados anteriormente no texto. Para alcançar essa vinculação, o *parser* mantinha um registro contínuo do contexto

hierárquico, atualizando-o à medida que novos títulos, seções ou subseções eram detectados. Esse registro garantiu que cada artigo recebesse os metadados corretos, mesmo em casos de formatação desordenada ou seções extensas. Um desafio importante enfrentado foi garantir a consistência nos casos em que a hierarquia não estava explicitamente clara ou era interrompida por formatação inadequada. Para mitigar esse problema, foram aplicadas verificações adicionais no *script*, como a comparação de padrões conhecidos com as informações detectadas, assegurando que as hierarquias associadas aos artigos fossem completas e consistentes.

O resultado desse processo foi a segmentação do regulamento de graduação da UFRN em 339 documentos, cada um correspondente a um artigo distinto, enriquecido com metadados que incluíam o título, a seção e a subseção relacionadas. Esses metadados foram incorporados aos artigos em um formato estruturado, como JSON, permitindo uma recuperação eficiente e precisa das informações. Essa organização hierárquica não apenas facilitou a indexação dos trechos segmentados no sistema de busca, mas também adicionou uma camada contextual robusta para consultas e respostas fundamentadas. Dessa forma, o sistema foi capaz de preservar e explorar a estrutura normativa dos documentos, assegurando um desempenho eficaz em cenários de recuperação de informações.

3.3 Recuperação

A implementação das estratégias de recuperação de informações utilizou o Qdrant como solução de armazenamento vetorial, aliado ao modelo Titan Text Embedding da AWS para a geração dos *embeddings* dos documentos. Cada trecho segmentado da base de dados foi processado pelo modelo, resultando em uma representação vetorial numérica para cada segmento de texto. Esses vetores foram então armazenados no Qdrant juntamente com metadados, como o título, a seção, a subseção e o número do artigo correspondente. Essa estrutura organizada permitiu uma recuperação eficiente, baseada na similaridade vetorial, e garantiu a acessibilidade e rastreabilidade dos conteúdos durante o processo de consulta.

Nos testes, o sistema utilizou como entrada os *embeddings* gerados a partir das perguntas existentes na base de dados, comparando-os aos *embeddings* armazenados no banco vetorial. A similaridade entre os vetores foi medida por duas métricas principais: similaridade de cosseno e distância euclidiana. Os experimentos realizados indicaram que ambas as métricas apresentaram desempenho equivalente na identificação de documentos relevantes, sem impacto perceptível na qualidade da recuperação. No entanto, devido à maior eficiência computacional, a similaridade de cosseno foi selecionada como métrica padrão para as próximas etapas. O processo de recuperação baseou-se na ordenação dos documentos, com base na similaridade calculada entre o *embedding* da consulta e os vetores

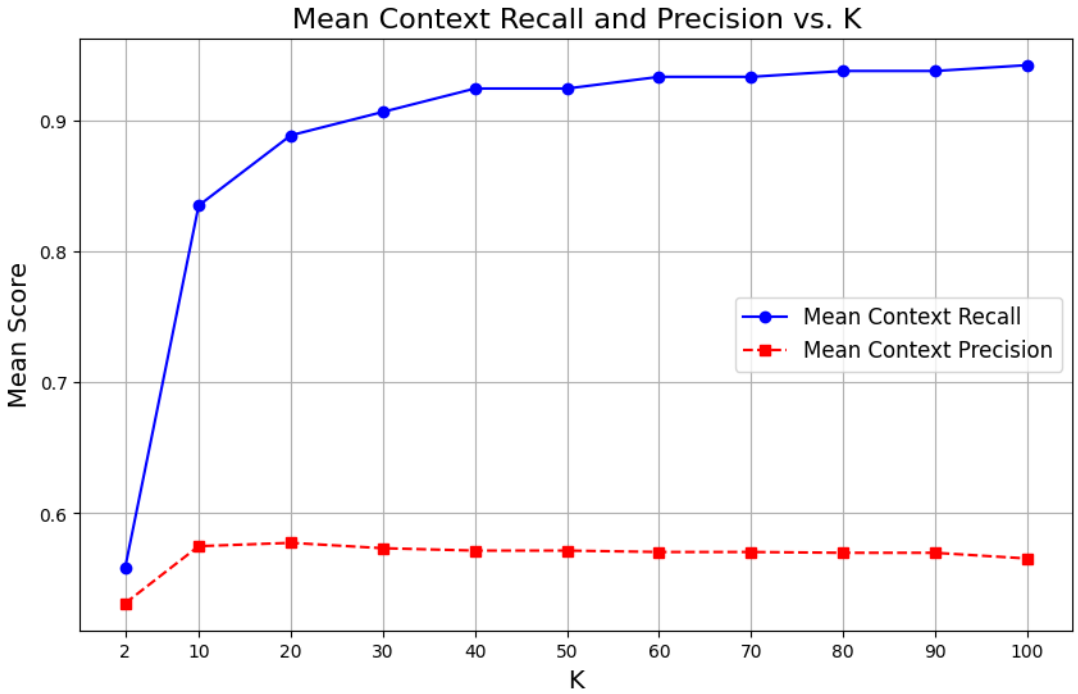

Figura 5 – Média da Precisão e Recuperação de contextos por documentos recuperados

Fonte: Autoria Própria

armazenados, resultando em uma lista classificada por relevância.

Um aspecto crítico identificado foi o impacto da quantidade de documentos retornados nos resultados finais da recuperação. Como ilustrado na Figura 5, uma quantidade reduzida de documentos recuperados pode comprometer a completude das respostas ao excluir informações potencialmente relevantes. Para mitigar esse risco, foi implementado um parâmetro ajustável, representado pelo número de documentos retornados, ou k. Essa abordagem possibilitou calibrar a recuperação conforme diferentes cenários, equilibrando precisão e *recall*. Os testes conduzidos exploraram o impacto de diferentes valores de k, avaliando os efeitos no desempenho do sistema. Valores baixos de k não mostraram grandes mudanças precisão nos documentos retornados, mas limitaram a cobertura, deixando informações importantes de fora. Em contrapartida, valores altos de k aumentaram a cobertura ao incluir mais documentos potencialmente relevantes, porém introduziram o risco de respostas contendo informações irrelevantes. Esse comportamento foi analisado para determinar o equilíbrio ideal entre precisão e *recall*.

3.3.1 Recuperação em Multiplas Consultas

A técnica de múltiplas consultas, amplamente utilizada em sistemas de RAG, foi implementada com o objetivo de diversificar semanticamente as consultas realizadas ao banco de dados vetorial, conforme descrito em (LI et al., 2024). Essa abordagem baseia-se na geração de variantes de uma mesma pergunta inicial, permitindo que diferentes inter-

pretações semânticas sejam exploradas durante o processo de recuperação de informações. Para este trabalho, cada pergunta foi expandida em cinco variantes geradas por modelo de linguagem testado, o que aumentou a abrangência da recuperação ao buscar identificar documentos relevantes que poderiam não ser recuperados com a consulta original.

A implementação da técnica iniciou-se com a seleção de cinco modelos generativos de linguagem: `Nova Micro`, `Nova Lite`, `Llama 3 8B`, `Mistral 7B` e `Mixtral 8x7B`, os quais foram utilizados para reescrever as perguntas iniciais em variantes semânticas diversificadas. Esses modelos foram configurados para garantir que as variantes geradas mantivessem consistência semântica com a pergunta original, enquanto apresentavam diferentes estruturas ou reformulações linguísticas. Após a geração das variantes, cada uma foi submetida ao sistema de recuperação de documentos, que utilizou o banco de dados vetorial baseado em *embeddings*, previamente construído a partir do modelo *Titan Text Embedding*. A recuperação de documentos foi configurada para retornar entre 1 e 30 documentos relevantes para cada variante, utilizando a métrica de similaridade de cosseno para medir a proximidade entre os *embeddings* das perguntas e os dos documentos armazenados.

Para lidar com a sobreposição de documentos recuperados, foi implementado um processo de deduplicação. Esse processo verificava documentos retornados por diferentes variantes da mesma pergunta e removia duplicatas, garantindo que apenas documentos únicos fossem mantidos para análise nas etapas subsequentes. A deduplicação foi realizada utilizando identificadores únicos associados aos documentos no banco vetorial, eliminando redundâncias sem comprometer a integridade dos resultados. Essa abordagem visou maximizar a cobertura informacional e garantir a eficiência do pipeline de recuperação.

A Tabela 1 apresenta os resultados da implementação da técnica de múltiplas consultas, indicando os valores de Recuperação Média e Precisão Média para diferentes configurações do parâmetro k, que define a quantidade de documentos únicos retornados por consulta. O resultado considera a média entre os modelos testados, visto que não apresentaram diferença considerável quanto as métricas escolhidas. Embora o processo de deduplicação tenha sido essencial para a análise, ele influenciou diretamente as métricas de precisão, uma vez que os documentos eliminados podiam alterar as posições originais na lista de resultados, dificultando a avaliação direta da ordenação inicial.

A estratégia de múltiplas consultas foi integrada ao pipeline de recuperação como um componente modular, permitindo ajustes, ou remoção, futuros para aumentar a eficiência em cenários mais complexos. A diversidade semântica introduzida pelas variantes geradas é uma característica relevante para aplicações onde a expressividade da consulta inicial pode ser limitada, especialmente em sistemas que exigem alta cobertura informacional a partir de consultas com baixo valor semântico. Embora a técnica não tenha superado os métodos tradicionais de recuperação em termos gerais de desempenho, sua aplicação combinada com

Tabela 1 – Resultado da técnica de Multiplas Consultas

K	Mean Recall	Mean Precision
5	0.668000	0.302267
10	0.761905	0.236177
15	0.741935	0.137041
25	0.853448	0.091630
30	0.906250	0.096657

outras técnicas exploradas neste trabalho poderá potencializar os resultados em cenários específicos que serão abordados no Capítulo 4. A flexibilidade do sistema desenvolvido permite a avaliação de novos parâmetros e combinações de metodologias em etapas futuras.

3.3.2 HyDE

O *Hypothetical Document Embeddings* (HyDE) é uma técnica de recuperação densa projetada para cenários onde um modelo precisa lidar com tarefas ou domínios novos sem treinamento ou ajuste prévio com dados específicos do contexto. A implementação desta abordagem combina dois componentes principais: a geração de documentos hipotéticos por modelos generativos e a codificação de similaridade utilizando aprendizado contrastivo não supervisionado, *embeddings*. O objetivo é criar documentos sintéticos baseados na consulta inicial, que, embora possam conter informações factualmente incorretas, capturam padrões de relevância sem a necessidade de rótulos explícitos de relevância.

A implementação iniciou-se com a integração de modelos generativos capazes de seguir instruções detalhadas para gerar os documentos hipotéticos. Esses documentos foram criados a partir de um *prompt* cuidadosamente elaborado, contendo instruções para o modelo produzir textos detalhados e potencialmente relacionados à consulta inicial. O *prompt* final utilizado foi ajustado iterativamente para garantir que as saídas fossem suficientemente diversificadas e cobrissem diferentes interpretações da consulta, aumentando assim a chance de capturar padrões de relevância em cenários de recuperação densa. Além da instrução principal para gerar multiplas consultas, entre as instruções enviadas ao modelo, destacam-se:

- Analisar cuidadosamente a pergunta e as informações necessárias para respondê-la.
- Formular um ou mais artigos acadêmicos abrangentes que respondam diretamente à pergunta.
- Não adicionar nenhum conhecimento externo.
- Usar um tom formal, objetivo, curto e conciso.

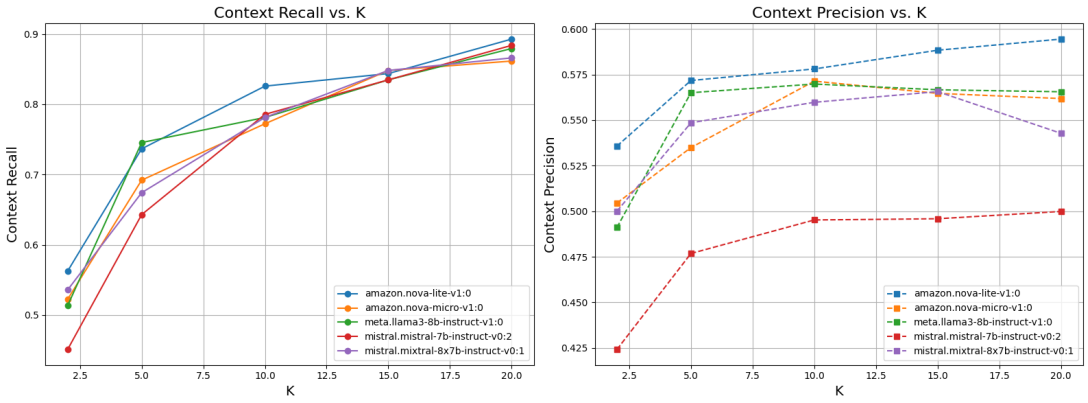

Figura 6 – Média da Precisão e Recuperação de contextos por documentos recuperados usando a técnica de documentos hipotéticos para diferentes modelos.

Fonte: Autoria Própria

- Garantir que os artigos contenham uma frase e respondam somente à pergunta.
- Escrever somente as frases dos artigos, sem textos de agradecimento, apresentação de resultados ou explicações fora do contexto.
- Evitar indicar um número hipotético dos artigos, escrevendo somente as frases.
- Garantir que cada artigo contenha um contexto único e não seja igual aos outros.

Após a geração, os documentos sintéticos foram processados por um modelo de *embedding*, o *Titan Text Embedding*, para transformá-los em vetores densos. Esses vetores foram comparados aos vetores de documentos reais previamente indexados em um banco de dados vetorial, utilizando a métrica de similaridade de cosseno. A etapa de indexação foi realizada com o auxílio do Qdrant onde cada documento real foi armazenado juntamente com seus metadados, como título, seção e contexto textual. A comparação entre os *embeddings* dos documentos hipotéticos e reais permitiu identificar os documentos reais mais semanticamente similares às consultas iniciais, mesmo em um cenário não supervisionado.

Para refinar o processo, diferentes configurações de *prompting* foram testadas, incluindo instruções de variados níveis de especificidade. Além disso, múltiplos modelos generativos foram avaliados para explorar sua capacidade de criar documentos sintéticos úteis para a recuperação densa. Essas iterações buscaram otimizar a qualidade das representações geradas, garantindo que os documentos hipotéticos capturassem informações amplas e relevantes para o domínio dos artigos acadêmicos.

Apesar dos esforços para ajustar a técnica ao contexto deste trabalho, como ilustrado na Figura 6, os resultados não foram satisfatórios. Mesmo com ajustes nos *prompts* e o uso de diferentes modelos, o método não conseguiu recuperar documentos relevantes de maneira consistente em comparação com métodos tradicionais de recuperação. Essa limitação pode

ser atribuída à complexidade do conjunto de dados utilizado ou à dificuldade do HyDE em capturar nuances específicas do domínio acadêmico. A implementação do HyDE destacou a importância de ajustar a técnica aos desafios específicos do domínio, sugerindo a necessidade de investigações adicionais no Capítulo 4 para validar sua aplicabilidade em cenários complementares.

3.3.3 Reordenação de Documentos

Após a recuperação inicial de um conjunto de documentos candidatos, a etapa de *reranking*, ou reordenação, foi implementada para reorganizar os documentos com base em sua relevância em relação à consulta do usuário. Essa etapa é fundamental para priorizar os documentos mais úteis na subsequente geração de respostas, otimizando a qualidade geral do sistema. Para este trabalho, foram utilizados dois modelos de *rerank* especializados: o Rerank 1.0 da Amazon e o Rerank 3.5 da Cohere a partir do *framework LangChain*. Esses modelos foram configurados para calcular uma pontuação de relevância para cada par consulta-documento, considerando tanto a representação da consulta quanto o conteúdo do documento, incorporando uma análise contextual mais detalhada do que a abordagem inicial baseada em similaridade vetorial.

O pipeline de *reranking* foi estruturado a partir da integração direta dos modelos mencionados com a lista de documentos candidatos gerada na recuperação inicial. A entrada para o *rerank* consistia em uma lista de documentos ordenados por similaridade vetorial, com um número variável de itens, que ia de 2 a 100. Cada documento da lista foi submetido ao modelo de *rerank*, que atribuía uma nova pontuação com base em sua relevância para a consulta. A lista foi, então, reordenada de acordo com essas pontuações, resultando em uma hierarquia de documentos mais refinada.

Os experimentos com diferentes tamanhos de conjuntos recuperados permitiram avaliar o impacto do *rerank* na precisão final do sistema. Essa análise específicas mostrou que, em cenários onde a recuperação inicial incluía documentos parcialmente relevantes ou irrelevantes, o *rerank* foi eficaz em promover os documentos mais úteis para o topo da lista. Por exemplo, textos considerados relevantes estavam, em média, nos primeiros 20% da lista após a reordenação, sugerindo que os 80% subsequentes poderiam ser descartados por não contribuírem significativamente para o contexto da consulta. Esse filtro reduzido mostrou-se vantajoso em sistemas RAG, minimizando custos computacionais e otimizando a precisão das respostas ao reduzir ruídos no conjunto de contextos fornecidos à etapa de geração.

Os modelos utilizados para o *rerank* foram configurados para processar documentos e consultas em conjunto, o que permite uma análise mais profunda das relações semânticas e contextuais entre as entradas. Isso difere significativamente da recuperação inicial, que se

baseia exclusivamente em similaridade vetorial. Essa abordagem mais detalhada mostrou-se especialmente eficaz quando os documentos candidatos continham informações relevantes diluídas entre trechos menos úteis. No entanto, à medida que a quantidade de documentos recuperados aumentava, a eficiência do *rerank* apresentou uma leve redução, devido à dificuldade de priorizar textos relevantes em meio a um volume maior de dados irrelevantes.

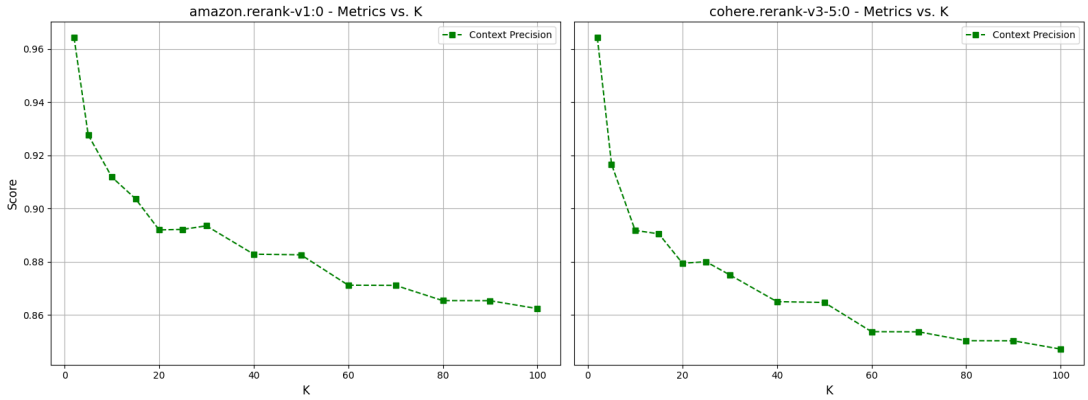

Figura 7 – Média da Precisão de contextos por documentos recuperados usando a técnica de reordenação

Fonte: Autoria Própria

Para garantir a eficiência do *rerank*, a configuração do tamanho do conjunto inicial de documentos recuperados foi tratada como um parâmetro ajustável. Essa configuração revelou-se crítica para equilibrar precisão, eficiência e custo computacional. Embora conjuntos maiores aumentem a chance de incluir documentos relevantes, eles também sobrecarregam o modelo de *rerank* com dados irrelevantes, diminuindo sua eficácia incremental. Como ilustrado na Figura 7, ambos os modelos utilizados obtiveram desempenho similar e consistentemente superior à abordagem de recuperação inicial, ilustrado na Figura 5, indicando que o *rerank* contribui de maneira significativa para a melhoria da precisão do sistema.

A implementação da etapa de *reranking* demonstra como a combinação de modelos especializados e ajustes no pipeline pode otimizar a qualidade da recuperação de informações, ao mesmo tempo em que reduz custos computacionais e melhora a eficiência do sistema. Essa abordagem reforça a importância de integrar técnicas avançadas de reordenação em sistemas baseados em RAG para garantir respostas mais precisas e relevantes.

3.3.4 Rotas semânticas

A implementação do sistema para roteamento semântico de consultas foi estruturada com base na integração de ferramentas modernas para recuperação e processamento de informações, garantindo eficiência e flexibilidade. Embora a biblioteca `LangChain` tenha

sido amplamente utilizada para gerenciar a interação com modelos de linguagem e o banco vetorial `Qdrant`, ela não fornece um recurso específico para o roteamento semântico. Dessa forma, foi necessário desenvolver uma abstração personalizada, a classe `SemanticRouter`, que permitiu implementar o processo de roteamento com base em similaridade semântica, integrando a funcionalidade de armazenamento vetorial do `Qdrant` com os embeddings gerados pela API `Bedrock`. Esse desenvolvimento possibilitou a automatização do fluxo de consultas e recuperação, associando enunciados armazenados às rotas mais relevantes com base nas pontuações de similaridade.

A estrutura principal do sistema foi organizada em torno de uma classe personalizada, denominada `SemanticRouter`, projetada para integrar o banco vetorial `Qdrant` com os embeddings gerados pela API `Bedrock`. O `Qdrant` foi configurado como um repositório vetorial, armazenando tanto os embeddings dos enunciados quanto seus metadados associados, como rotas e textos. Os embeddings foram gerados utilizando a biblioteca `BedrockEmbeddings`, que processou cada entrada textual em representações vetoriais para comparação semântica. A classe `SemanticRouter` incluiu funcionalidades como a adição de enunciados e rotas ao banco vetorial e a realização de buscas com base na similaridade. A função `add_utterance` permitiu armazenar enunciados no `Qdrant`, associando-os a rotas específicas. A função `get_route_details` foi responsável por retornar uma lista detalhada de resultados, incluindo rotas, enunciados e pontuações de similaridade, enquanto a função `get_route` foi desenvolvida para identificar a rota mais relevante com base em um limite de pontuação configurável.

Para avaliar o desempenho do sistema, foram criadas duas rotas semânticas principais. Uma para saudações e outra para despedidas, cada uma com 15 enunciados de exemplo. Também foi utilizado um conjunto de dados da base de referência, contendo pares de entrada e saída. Os enunciados foram processados para cada pergunta iterativamente por meio da classe `SemanticRouter`, e as pontuações de similaridade foram registradas para análise. O `LangChain` automatizou a interação com o `Qdrant`, permitindo a comparação entre consultas e enunciados previamente armazenados. Durante os testes, cada consulta recuperava o enunciado mais próximo com base na métrica de similaridade de cosseno, retornando o enunciado correspondente e sua rota associada. As pontuações resultantes foram analisadas utilizando a biblioteca `Pandas` e visualizadas com `Matplotlib`. Um gráfico de caixa foi gerado para representar a distribuição das pontuações de similaridade entre os enunciados armazenados e as consultas realizadas para que fosse possível decidir o melhor limiar de não acionamento das rotas de saudações e despedidas. Esse processo permitiu identificar a consistência do sistema em atribuir rotas relevantes às consultas, destacando o potencial do `SemanticRouter` para integrar-se a aplicações mais amplas de roteamento semântico.

Essa abordagem modular e integrada com ferramentas como `LangChain`, `Qdrant`

e `Bedrock` permitiu a criação de um pipeline eficiente para recuperação semântica e roteamento baseado em similaridade, garantindo precisão e escalabilidade para cenários complexos.

3.4 Geração

A etapa de geração do chatbot baseado em RAG para o Regulamento dos Cursos de Graduação da UFRN foi implementada com uma arquitetura modular que integrou ferramentas avançadas para recuperação e geração de informações. O *LangChain* foi utilizado como *wrapper* para gerenciar modelos de linguagem e automatizar o fluxo de dados entre as etapas do pipeline, enquanto a *Amazon Bedrock* forneceu acesso aos modelos Nova Lite e Nova Micro, que desempenharam um papel central na geração de respostas. A integração dessas ferramentas foi projetada para garantir que as respostas fossem precisas, relevantes e alinhadas às regras definidas no regulamento.

Na fase de recuperação, os documentos relevantes foram obtidos utilizando o banco vetorial *Qdrant*, onde os documentos segmentados e indexados previamente pelo modelo *Titan Text Embedding* da AWS foram armazenados. A recuperação inicial foi configurada para retornar até 10 documentos por consulta, priorizando aqueles com maior similaridade semântica à pergunta do usuário, com base na métrica de similaridade de cosseno. Esse limite de 10 documentos foi escolhido para balancear a relevância dos documentos recuperados e a eficiência computacional do pipeline.

Os documentos recuperados foram então processados e enviados aos modelos de linguagem gerenciados pelo *LangChain*. A *Amazon Bedrock* forneceu suporte aos modelos Nova Lite, Nova Micro, LLama 3 8B e Mistral, que receberam os documentos recuperados e a consulta original, gerando respostas contextualizadas e alinhadas ao regulamento. Durante a implementação, o Nova Lite mostrou-se eficiente em equilibrar relevância e fidelidade com tempos de resposta rápidos, enquanto o Nova Micro destacou-se pela maior precisão e economia no número de tokens gerados, reduzindo os custos operacionais associados.

Além disso, o modelo GPT-4o foi introduzido como julgador para avaliar a qualidade das respostas geradas pelos modelos. O GPT-4o foi configurado para analisar as respostas geradas e compará-las diretamente com o conteúdo do regulamento e os documentos recuperados, validando a consistência e a fidelidade das informações apresentadas. Para isso, o GPT-4o recebeu tanto a consulta inicial quanto os documentos relevantes, avaliando se a resposta gerada era precisa, coerente e sem omissões importantes. Essa etapa de validação foi crucial para assegurar que o sistema operasse de forma confiável, especialmente em um contexto acadêmico onde a precisão das respostas é essencial. O RAGAS desempenhou um papel central na integração do GPT-4o como julgador, automatizando a comparação

entre as respostas geradas e os documentos originais. O fluxo foi configurado para que cada resposta gerada pelos modelos passasse por uma validação automática pelo GPT-4o, que atribuía uma pontuação de qualidade com base em critérios definidos previamente, como relevância e fidelidade.

Essa abordagem, combinando o *LangChain*, *Amazon Bedrock*, *Qdrant* e o GPT-4o como julgador, foi projetada para garantir que as respostas geradas pelo chatbot fossem não apenas eficientes em termos de custo computacional, mas também altamente precisas e confiáveis. A recuperação inicial limitada a 10 documentos, a geração gerenciada pelo *LangChain* e a validação pelo GPT-4o formaram um pipeline robusto e adaptável, adequado às exigências do sistema baseado em RAG.

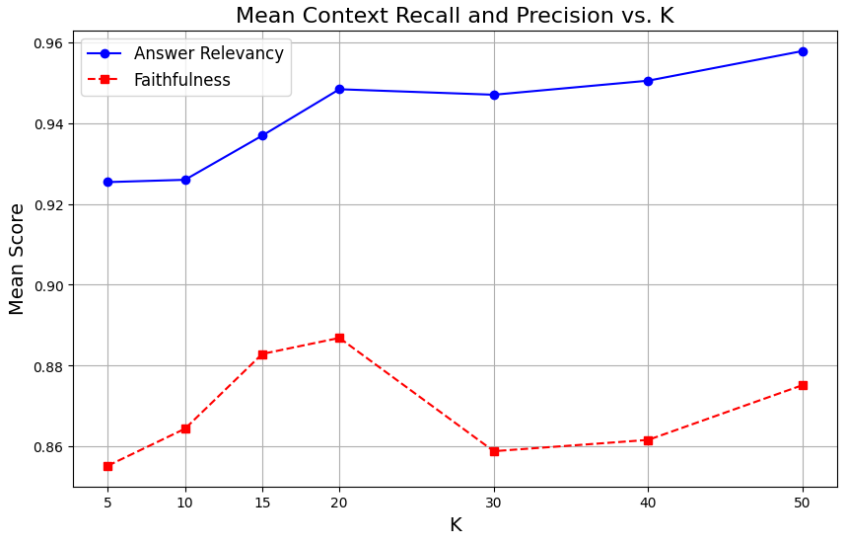

Figura 8 – Relevância e Fidelidade da resposta gerada pelo modelo em diferentes quantidades de contextos recuperados

Fonte: Autoria Própria

Para avaliar o impacto da quantidade de documentos recuperados na etapa de geração, foram realizados testes variando o número de documentos retornados pelo banco vetorial *Qdrant*. Esses testes foram configurados para recuperar de 5 a 100 documentos por consulta, utilizando a métrica de similaridade de cosseno para ordenar os documentos por relevância. Após a recuperação, os documentos foram processados pelo modelo Nova Micro, integrado via *LangChain*. A análise focou em métricas como relevância e fidelidade das respostas geradas, além do tempo de processamento, considerando o número de tokens gerados como indicador de custo computacional. A Figura 8 ilustra os resultados desses testes, destacando que, embora um maior número de documentos inicialmente recuperados tenha contribuído para um aumento na relevância e fidelidade, esses ganhos tornaram-se marginais a partir de 20 documentos. Adicionalmente, foi observado que o tempo de resposta aumentava linearmente com o número de documentos recuperados, devido ao crescimento do número de tokens processados. Esses resultados reforçam que recuperar 20

documentos é a configuração ideal, equilibrando eficiência, custo computacional e qualidade das respostas geradas.

3.5 Criação do Protótipo

A implementação do sistema proposto foi realizada utilizando uma arquitetura modular e escalável, estruturada em Python e versionada por meio do *GitHub*, garantindo rastreabilidade e controle sobre as alterações do código. O desenvolvimento seguiu o padrão de camadas, sendo dividido em três componentes principais: repositórios, serviços e modelos, cada um responsável por uma funcionalidade específica no pipeline do sistema. A camada de repositórios gerenciava o acesso aos dados persistentes, encapsulando a lógica de acesso e abstraindo os detalhes técnicos. A camada de serviços implementava a lógica de negócios, processando dados recuperados da camada de repositórios para gerar respostas, coordenar interações com modelos de LLM, e gerenciar a comunicação com o *bot*. Já a camada de modelos padronizava as estruturas de dados, definindo formatos consistentes para mensagens, feedbacks e respostas, garantindo integração e minimizando erros. Essa organização modular facilitou a manutenção, escalabilidade e desenvolvimento contínuo do sistema.

O código foi desenvolvido utilizando bibliotecas relevantes como `requests`, para interações HTTP; `langchain`, `langchain-aws` e `langchain-qdrant`, para integração com modelos de linguagem, embeddings e banco vetorial; e `loguru`, para gerência de logs, garantindo um monitoramento eficiente do sistema. A estrutura modular foi projetada para que os repositórios fossem responsáveis por interagir diretamente com as bases de dados, os serviços encapsulassem a lógica de negócios e os modelos gerassem representações dos dados manipulados.

Para o armazenamento dos dados, a arquitetura utilizou *Qdrant* como banco vetorial, responsável por armazenar as representações *embeddings* dos textos, e *DynamoDB*, configurado para gerenciar três tabelas distintas: uma para registrar mensagens recebidas, outra para armazenar feedbacks sobre a utilidade das respostas geradas, e uma terceira para metadados do sistema. Essas soluções foram integradas ao sistema por meio de serviços da *Amazon Web Services* (AWS), incluindo *API Gateway* para gerenciar requisições, *Lambda Functions* para processamento de mensagens e *Bedrock* para o gerenciamento de modelos de linguagem e embeddings. Dessa forma, o pipeline de CI/CD foi configurado no *GitHub Actions*, automatizando o processo de *deploy* sempre que um *merge* fosse realizado na *branch* principal. Esse pipeline incluía etapas para a execução de testes automatizados, *linting* de código e *build*, seguido da implantação automática no ambiente da AWS. O *API Gateway* foi configurado para receber requisições HTTP do sistema e acionar funções *Lambda*, que processavam os dados e interagiam com os repositórios.

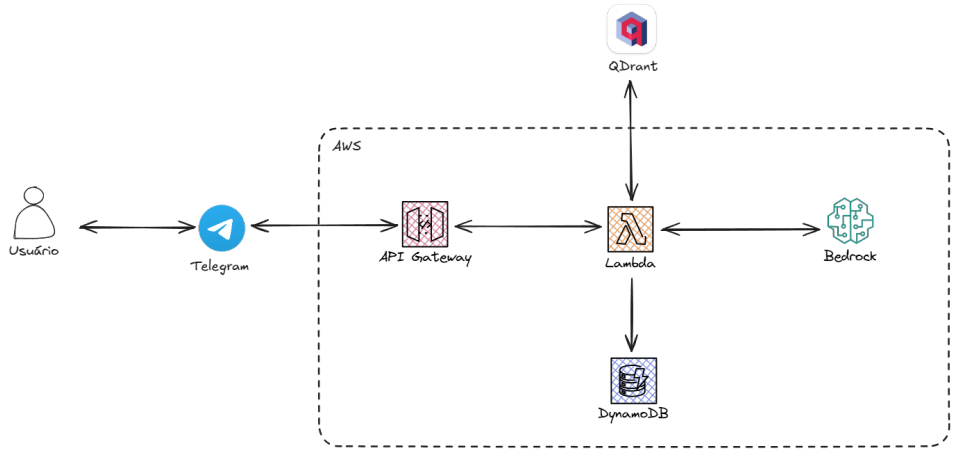

Figura 9 – Arquitetura do chatbot

Fonte: Autoria Própria

A interface com os usuários foi implementada por meio de um *bot* no Telegram, configurado para receber mensagens dos usuários e enviá-las ao *API Gateway* via *webhook*. O *bot* foi programado para capturar as mensagens em tempo real, acionando o sistema para processar as informações, recuperar contextos relevantes no *Qdrant*, gerar respostas utilizando os modelos disponíveis no *Bedrock* e registrar os dados nas tabelas do *DynamoDB*. Como mostrado na Figura 9, o fluxo de execução do sistema começava com o recebimento de uma mensagem no Telegram, que acionava o *webhook* configurado no *API Gateway*. A mensagem era enviada para uma *Lambda Function*, onde era processada para gerar embeddings utilizando a integração com o `langchain`. Esses embeddings eram comparados aos dados armazenados no *Qdrant*, utilizando a métrica de similaridade de cosseno, para identificar os contextos mais relevantes. A resposta era então gerada por meio dos modelos de linguagem no *Bedrock* e retornada ao usuário via Telegram. Adicionalmente, as mensagens recebidas e seus feedbacks eram registrados no *DynamoDB*, permitindo um monitoramento contínuo do sistema e a avaliação de sua eficácia.

Essa abordagem integrada, utilizando *GitHub Actions* para automação, AWS para infraestrutura escalável e ferramentas modernas para processamento e armazenamento de dados, garantiu a robustez e a eficiência do sistema desenvolvido. A modularidade da arquitetura facilita a adição de novas funcionalidades, enquanto a estrutura de logs e feedbacks permite um ciclo contínuo de melhoria do sistema.

4 ANÁLISE DOS RESULTADOS

Esse capítulo tem como objetivo principal avaliar o desempenho do sistema proposto de forma integrada, consolidando os resultados obtidos em cada uma das etapas descritas no desenvolvimento. Embora cada seção tenha apresentado resultados específicos para técnicas e implementações individuais, como indexação, recuperação e geração, este capítulo analisa essas técnicas em conjunto, verificando como elas interagem e impactam a eficiência, precisão e relevância do sistema como um todo. Essa abordagem permite identificar sinergias entre as etapas, validar as escolhas metodológicas feitas e discutir possíveis limitações ou melhorias, assegurando uma avaliação abrangente e consistente do sistema desenvolvido.

Considerando o resultados da indexação apresentados no Capítulo 3, a segmentação semântica reforça sua adequação ao contexto de regulamentos acadêmicos, destacando a eficácia dessa abordagem em domínios com textos formalmente estruturados e semanticamente independentes. A opção por uma divisão baseada em delimitadores específicos, como "Art. X"ou "TÍTULO - Y", demonstrou alinhamento com a natureza dos textos regulamentares, garantindo que cada segmento representasse uma unidade de sentido completa e autossuficiente. Essa estratégia facilitou a organização hierárquica dos artigos, permitindo a inclusão de metadados estruturais como título, capítulo, seção e subseção, o que agregou valor ao processo de recuperação de informações. Por preservar a integridade semântica de cada artigo, a segmentação evitou fragmentações arbitrárias e possibilitou ao sistema oferecer respostas mais precisas e contextualizadas às consultas realizadas. Essa abordagem destacou-se como uma solução eficiente para lidar com regulamentos acadêmicos, divididos em 339 documentos, um para cada artigo, onde a formalidade e a independência semântica dos textos são características essenciais para a recuperação e a geração de respostas relevantes.

Já durante a fase de recuperação, foram exploradas estratégias avançadas de recuperação, como múltiplas consultas e geração de documentos hipotéticos, HyDE, além da combinação de ambas, visando melhorar a identificação de documentos relevantes. A técnica de múltiplas consultas ampliou a cobertura de documentos ao gerar variantes de uma mesma pergunta, mas enfrentou limitações devido à redundância nos retornos e à diminuição da precisão conforme o volume de documentos únicos aumentava. Por sua vez, o método HyDE, que utiliza modelos de linguagem para criar documentos sintéticos a partir das consultas, não obteve resultados satisfatórios no contexto dos regulamentos acadêmicos, demonstrando dificuldades em capturar nuances específicas do domínio, mesmo após ajustes nos prompts e testes com diferentes modelos. Até mesmo a combinação dessas

abordagens também não se mostrou eficaz, uma vez que as limitações individuais de cada técnica foram ampliadas, resultando em baixos índices de recuperação e precisão. Esses resultados reforçam a eficiência do método tradicional de recuperação no cenário avaliado, embora as estratégias exploradas possam apresentar potencial em contextos mais complexos ou como parte de pipelines híbridos que associem metodologias complementares. Entretanto, a escolha do modelo de *rerank* desempenhou um papel crucial na performance do sistema. O Rerank 1.0 da Amazon demonstrou boa capacidade de filtrar documentos irrelevantes quando comparado ao modelo da Cohere. Este comportamento evidenciou a importância da combinação de modelos robustos que não apenas avaliassem a similaridade superficial, mas também a relevância semântica em profundidade. A análise dos resultados, portanto, indicou que uma escolha estratégica entre os diferentes modelos de *rerank* e a definição cuidadosa do número de documentos recuperados foram determinantes para a performance final do sistema, permitindo que ele opere de forma mais eficiente em contextos com diferentes características de dados e consultas. É valido também considerar a limitação do Bedrock da Amazon quanto ao *rerank* está na cobrança por consultas fragmentadas quando o número de documentos ultrapassa cem blocos ou o tamanho de tokens excede 512, o que pode gerar custos adicionais e reduzir a precisão semântica devido à fragmentação dos documentos (Amazon Web Services, 2025).

Tabela 2 – Relevância e Fidelidade para diferentes quantidades de documentos recuperados usando a abordagem *end-to-end* de recuperação tradicional, *reranking* e geração com Nova Miro.

k	Relevancy	Faithfulness	Response Time (s)	Output Tokens
20	0.945566	0.880369	1.887901	183
30	0.945139	0.879558	1.918054	183
50	0.939732	0.871012	1.983150	190
100	0.951880	0.876525	1.838934	183

Fonte: Autoria Própria

Por fim, foi avaliado o impacto da variação na quantidade inicial de documentos recuperados sobre as métricas de desempenho do sistema. Os experimentos finais foram considerados utilizando um *pipeline* baseado na abordagem de recuperação tradicional combinada com a etapa de *reranking*, descartando o uso das técnicas de múltiplas consultas e geração de documentos hipotéticos. Na Tabela 2 é mostrado que quantidade de documentos recuperados variou entre 20, 30, 50 e 100, enquanto o contexto passado para o modelo gerador de respostas foi sempre limitado aos 20 documentos com maior *score* após o *reranking*. Essa abordagem assegurou consistência na entrada do modelo gerador, permitindo avaliar os efeitos da recuperação inicial no desempenho geral do sistema. Os resultados demonstraram que a relevância alcançou seu maior valor ao recuperar inicialmente 100 documentos, sugerindo que a ampliação do conjunto inicial permitiu ao modelo

de *reranking* identificar, com maior precisão, os 20 documentos mais relevantes. Por outro lado, a fidelidade apresentou seu valor máximo quando a recuperação foi limitada a 20 documentos, com uma redução gradual à medida que a quantidade inicial aumentava, atingindo o menor valor com 50 documentos. Essa queda pode ser atribuída à introdução de ruído no *reranking* devido à presença de documentos menos relevantes no conjunto inicial. O tempo de resposta oscilou entre 1,83 e 1,98 segundos, com um leve aumento conforme o número de documentos recuperados crescia, refletindo o maior esforço computacional necessário para processar conjuntos maiores no *reranking*. Já o número médio de tokens gerados nas respostas permaneceu praticamente constante, variando entre 183 e 190, evidenciando que o tamanho do contexto efetivamente passado ao modelo gerador foi o principal fator determinante dessa métrica.

Esses resultados indicaram que, independentemente da quantidade inicial de documentos recuperados, não se observaram mudanças significativas nas métricas de desempenho. Embora a relevância tenha apresentado uma leve variação, com o maior valor registrado ao recuperar 100 documentos, e a fidelidade tenha sido ligeiramente superior ao limitar a recuperação a 20 documentos, as diferenças entre os cenários foram pequenas e não comprometeram a qualidade geral das respostas. Assim, a escolha da quantidade inicial de documentos pode ser ajustada com base em restrições de desempenho computacional ou prioridades específicas, sem impacto significativo na efetividade do sistema. Esses achados reforçam a robustez do modelo ao lidar com diferentes configurações de recuperação inicial, garantindo consistência nos resultados apresentados.

Já a análise do *bot* implementado no provedor *cloud* revelou métricas operacionais que destacam sua eficiência e desempenho em ambiente de produção. Considerando um conjunto de 20 perguntas realizadas, o consumo médio de memória durante as operações foi de 241 MB, indicando uma utilização estável e compatível com os recursos provisionados. O tempo médio para gerar uma resposta completa foi de 4586 milissegundos, abrangendo desde o recebimento da consulta pelo `Lambda` até o envio da resposta ao usuário via *bot* no Telegram. Esses resultados evidenciam a robustez e a escalabilidade da implementação, enquanto pontos de otimização futuros podem ser considerados para reduzir a latência em cenários com alto volume de consultas simultâneas.

5 CONCLUSÃO

Este trabalho apresentou uma abordagem para a criação de um chatbot especializado, utilizando as práticas e ferramentas mais avançadas em processamento de linguagem natural e sistemas de recuperação de informações. O desenvolvimento foi fundamentado em quatro etapas principais: criação e customização de um dataset utilizando o framework RAGAS, aplicação do método RAG para construção do chatbot, indexação e recuperação de dados, e testes de geração com diferentes modelos. A criação do dataset representou um passo inicial essencial, em que o framework RAGAS permitiu estruturar os dados de maneira eficiente. Após a geração inicial, foi realizada a customização do dataset, alinhando-o às necessidades específicas do projeto e garantindo uma base sólida para os modelos subsequentes. Na etapa de indexação, foram exploradas diferentes abordagens para estruturar o conteúdo de forma a otimizar sua acessibilidade e precisão. Testes comparativos entre divisores de textos recursivos por caracteres e divisão semântica indicaram que esta última era mais adequada devido à natureza altamente contextual do regulamento utilizado. Essa escolha assegurou maior coesão e relevância na recuperação das informações. Durante a etapa de recuperação, três técnicas foram testadas: HyDE, múltiplas questões e reordenação. As duas primeiras não apresentaram resultados significativos, enquanto a reordenação demonstrou uma contribuição expressiva para a qualidade final do chatbot. Isso evidencia a importância de adaptar os métodos de recuperação às especificidades do problema e às características do conjunto de dados. Finalmente, a etapa de geração envolveu a avaliação de cinco modelos diferentes, os quais produziram resultados bastante similares em qualidade. Essa consistência indica a maturidade atual dos modelos de linguagem, permitindo uma escolha baseada em critérios como custo computacional e velocidade, sem comprometer significativamente a performance. O processo desenvolvido e os experimentos realizados evidenciaram a importância de integrar métodos bem estabelecidos com customizações específicas para maximizar o desempenho de chatbots baseados em RAG. A abordagem utilizada neste trabalho oferece um caminho claro para a criação de sistemas robustos e adaptáveis a diversos cenários, com potencial para aplicações em áreas que demandem precisão e contextualização elevada.

O trabalho desenvolvido apresenta contribuições relevantes, mas também algumas limitações que merecem destaque. Entre as limitações identificadas, destaca-se a dependência da qualidade do dataset inicial gerado com o framework RAGAS. Embora o framework tenha facilitado a criação do dataset, a clareza e organização dos documentos de origem influenciaram diretamente a qualidade do material final, exigindo intervenções manuais em casos de dados confusos ou incompletos. Além disso, a natureza específica do regulamento utilizado pode dificultar a replicação imediata da metodologia em outras

áreas, dada sua complexidade e particularidades. Outro aspecto limitante foi o alto custo computacional, especialmente na aplicação de técnicas como reordenação e nos testes realizados com múltiplos modelos de geração. Por fim, embora os modelos avaliados tenham apresentado resultados próximos, diferenças sutis podem se tornar mais significativas em contextos de maior complexidade ou sensibilidade das respostas. Apesar dessas limitações, o trabalho trouxe contribuições importantes para o desenvolvimento de chatbots baseados no método RAG. Além disso, o estudo comparativo sobre métodos de indexação evidenciou a superioridade da abordagem semântica para casos em que o contexto do regulamento é crítico, proporcionando uma base mais sólida para a recuperação de informações. No âmbito da recuperação, a reordenação destacou-se como a estratégia mais eficiente entre as testadas, contribuindo significativamente para o aumento da precisão do sistema. Por fim, a aplicação de uma metodologia iterativa e avaliativa, envolvendo desde a geração e customização do dataset até os testes de recuperação e geração, oferece uma estrutura clara que pode ser replicada em projetos futuros.

Com base nos resultados obtidos, algumas direções podem ser exploradas em trabalhos futuros. Em primeiro lugar, o aprimoramento da qualidade do *dataset* inicial pode ser investigado com o uso de técnicas automatizadas, como modelos pré-treinados para identificar inconsistências nos dados. Outra possibilidade é a adaptação da metodologia desenvolvida para integração de outros documentos da universidade. Além disso, estratégias para *fine-tuning* podem ser exploradas para adequar todos os modelos ao contexto acadêmico. No campo da recuperação, métodos avançados, incluindo modelos mais recentes de reordenação ou embeddings dinâmicos, representam oportunidades para melhoria contínua. Testar modelos de linguagem mais avançados, à medida que novas versões são lançadas, também pode revelar ganhos significativos em precisão e qualidade das respostas geradas. Por fim, a expansão do chatbot para suportar interação multimodal, incorporando texto, imagem e áudio, pode ampliar sua acessibilidade e aplicação em diferentes contextos. Assim, este trabalho não apenas contribui para a evolução dos métodos de desenvolvimento de chatbots baseados no método RAG, mas também aponta direções claras para avanços futuros no campo, com potencial de impacto em aplicações em todo contexto no que se refere a vida acadêmica.

REFERÊNCIAS

Amazon Web Services. *Amazon Bedrock Pricing*. 2025. Acesso em: 10 jan. 2025. Disponível em: <https://aws.amazon.com/pt/bedrock/pricing-/>https://aws.amazon.com/pt/bedrock/pricing/.

BENGIO, Y. et al. A neural probabilistic language model. *Journal of Machine Learning Research*, v. 3, p. 1137–1155, 2003. Submitted 4/02; Published 2/03. Disponível em: <http://www.jmlr.org/papers/volume3/bengio03a/bengio03a-.pdf>http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.

BOTPRESS. *Chatbot for Education*. 2024. Disponível em: <https:-//botpress.com/pt/blog/chatbot-for-education?utm_source=chatgpt-.com>https://botpress.com/pt/blog/chatbot-for-education?utm_source=chatgpt.com.

CASTOR, E. C. S. et al. Chatbot: impactos no ambiente acadêmico de uma universidade do rio de janeiro. *P2P & Inovação*, Universidade Santa Úrsula (USU), v. 8, n. 1, p. 71–92, set. 2021/fev. 2022 2021. Esta obra está licenciada sob uma licença Creative Commons Attribution 4.0 International (CC BY-NC-SA 4.0). Disponível em: <https://doi.org/10-.21721/p2p.2021v8n1.p71-92>https://doi.org/10.21721/p2p.2021v8n1.p71-92.

CHERUBINI, M. et al. Improving the accessibility of eu laws: The chat-eur-lex project. In: *ITAL-IA 2024: IV CINI National Conference on Artificial Intelligence*. Naples, Italy: CINI, 2024.

CONSTITUIçãO da República Federativa do Brasil de 1988. Brasília, Brasil: Presidência da República, 1988. Artigo 207. Autonomia universitária. Acesso em: 06 nov. 2024. Disponível em: <http://www.planalto.gov.br/ccivil_03/constituicao/constituicao-.htm>http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm.

CUNHA, F. P. da. *Desempenho acadêmico: estudo sobre uma estratégia institucional de orientação acadêmica*. 147 p. Dissertação (Dissertação de Mestrado) — Universidade Federal do Rio Grande do Norte, Natal, 2020. Dissertação (mestrado) - Centro de Ciências Humanas, Letras e Artes, Programa de Pós-Graduação em Gestão de Processos Institucionais.

DATACAMP. 8 principais llms de código aberto para 2024 e seus usos. 2023. Disponível em: <https://www.datacamp.com/pt/blog/top-open-source-llms>https://www.datacamp.com/pt/blog/top-open-source-llms.

ES, S. et al. *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. 2023. Disponível em: <https://arxiv.org/abs/2309.15217>https://arxiv.org/abs/2309.15217.

ES, S. et al. *Testset Generation in RAGAS*. 2024. Accessed: 2025-01-11. Disponível em: <https://docs.ragas.io/en/v0.1.21/concepts/testset_generation-.html>https://docs.ragas.io/en/v0.1.21/concepts/testset_generation.html.

GAO, L. et al. *Precise Zero-Shot Dense Retrieval without Relevance Labels*. 2022. Disponível em: <https://arxiv.org/abs/2212.10496>https://arxiv.org/abs/2212.10496.

GAO, Y. et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. Disponível em: <https://arxiv.org/abs/2312.10997>https://arxiv.org/abs/2312.10997.

GLASS, M. et al. *Re2G: Retrieve, Rerank, Generate*. 2022. Disponível em: <https://arxiv.org/abs/2207.06300>https://arxiv.org/abs/2207.06300.

HUANG, L. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, Association for Computing Machinery (ACM), nov. 2024. ISSN 1558-2868. Disponível em: <http://dx.doi.org/10.1145/3703155>http://dx.doi.org/10.1145/3703155.

JOHNSON, J.; DOUZE, M.; JéGOU, H. *Billion-scale similarity search with GPUs*. 2017. Disponível em: <https://arxiv.org/abs/1702.08734>https://arxiv.org/abs/1702.08734.

KAPLAN, J. et al. *Scaling Laws for Neural Language Models*. 2020. Disponível em: <https://arxiv.org/abs/2001.08361>https://arxiv.org/abs/2001.08361.

KARPUKHIN, V. et al. *Dense Passage Retrieval for Open-Domain Question Answering*. 2020. Disponível em: <https://arxiv.org/abs/2004-.04906>https://arxiv.org/abs/2004.04906.

LEWIS, P. et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. Disponível em: <https://arxiv.org/abs/2005-.11401>https://arxiv.org/abs/2005.11401.

LI, Z. et al. *DMQR-RAG: Diverse Multi-Query Rewriting for RAG*. 2024. Disponível em: <https://arxiv.org/abs/2411.13154>https://arxiv.org/abs/2411.13154.

MANIAS, D. M.; CHOUMAN, A.; SHAMI, A. Semantic routing for enhanced performance of llm-assisted intent-based 5g core network management and orchestration. *arXiv preprint arXiv:2404.15869*, 2024.

MANNING, C. D.; SCHüTZE, H. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999. ISBN 0-262-13360-1.

OMNICHAT. *Chatbot na Educação*. 2024. Disponível em: <https://omni.chat-/chatbot-na-educacao/?utm_source=chatgpt.com>https://omni.chat/chatbot-na-educacao/?utm_source=chatgpt.com.

PAIVA, F. *Mapa do Ecossistema Brasileiro de Bots 2022*. [S.l.], 2022. Relatório que analisa o mercado de robôs de conversação no Brasil, com dados sobre uso, setores e crescimento. Disponível em: <https://www.mobiletime.com.br/pesquisas-/>https://www.mobiletime.com.br/pesquisas/.

PEREIRA, J. et al. Inacia: Integrating large language models in brazilian audit courts: Opportunities and challenges. *arXiv preprint arXiv:2401.05273*, 2024. Disponível em: <https://arxiv.org/abs/2401.05273>https://arxiv.org/abs/2401.05273.

PIPITONE, N.; ALAMI, G. H. Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain. *arXiv preprint arXiv:2408.10343*, 2024. Disponível em: <https://arxiv.org/abs/2408.10343>https://arxiv.org/abs/2408.10343.

REIMERS, N.; GUREVYCH, I. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. Disponível em: <https://arxiv.org/abs/1908-.10084>https://arxiv.org/abs/1908.10084.

RYU, C. et al. Retrieval-based evaluation for llms: A case study in korean legal qa. In: *Proceedings of the Natural Legal Language Processing Workshop 2023*. Online: Association for Computational Linguistics, 2023.

SANTOS, M. das G. *Insucesso acadêmico de estudantes: análise e estratégias de enfrentamento em uma universidade pública federal*. 145 p. Dissertação (Dissertação de Mestrado) — Universidade Federal do Rio Grande do Norte, Natal, 2022. Dissertação (mestrado) - Centro de Ciências Humanas, Letras e Artes, Programa de Pós-Graduação em Gestão de Processos Institucionais.

SERPRO. Governo federal começa a utilizar ia para "pesquisa inteligente". 2024. Disponível em: <https://www.serpro.gov.br/menu/noticias/noticias-2024/IA-governo-federal>https://www.serpro.gov.br/menu/noticias/noticias-2024/IA-governo-federal.

SKIMAI. Os 4 principais casos de uso de llm empresarial com o melhor roi. 2023. Disponível em: <https://skimai.com/pt/4-casos-de-utilizacao-de-gll-empresarial-com-o-melhor-roi-/>https://skimai.com/pt/4-casos-de-utilizacao-de-gll-empresarial-com-o-melhor-roi/.

UFRN. *Estatuto da Universidade Federal do Rio Grande do Norte*. Natal, Brasil, 2011. Atualizado pela Resolução nº 002/2011 - CONSUNI. Disponível em: <https://ufrn.br>https://ufrn.br.

UFRN. *Regimento Geral da Universidade Federal do Rio Grande do Norte*. Natal, Brasil, 2019. Art. 1º. Complementa e operacionaliza o Estatuto da Universidade. Disponível em: <https://ufrn.br>https://ufrn.br.

UFRN. *Regimento Interno da Reitoria da Universidade Federal do Rio Grande do Norte*. Natal, Brasil, 2021. Art. 1º. Complementa o Estatuto e o Regimento Geral da Universidade, definindo a estrutura e funcionamento da Reitoria. Disponível em: <https://ufrn.br>https://ufrn.br.

WEI, J. et al. *Emergent Abilities of Large Language Models*. 2022. Disponível em: <https://arxiv.org/abs/2206.07682>https://arxiv.org/abs/2206.07682.

YU, H. et al. *Evaluation of Retrieval-Augmented Generation: A Survey*. 2024. Disponível em: <https://arxiv.org/abs/2405.07437>https://arxiv.org/abs/2405.07437.

ZHAO, W. X. et al. *A Survey of Large Language Models*. 2024. Disponível em: <https://arxiv.org/abs/2303.18223>https://arxiv.org/abs/2303.18223.