PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DANTE FLESCH PEREIRA

INTEGRAÇÃO DE LLM E RAG PARA ANÁLISE DE DOCUMENTOS JURÍDICOS

Porto Alegre - RS

2024

DANTE FLESCH PEREIRA

INTEGRAÇÃO DE LLM E RAG PARA ANÁLISE DE DOCUMENTOS JURÍDICOS

Trabalho de conclusão de curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador(a): Profa. Dra. Denise Bandeira

Porto Alegre – RS

2024

INTEGRAÇÃO DE LLM E RAG PARA ANÁLISE DE DOCUMENTOS JURÍDICOS

RESUMO

O uso de Large Language Model (LLM) e de técnicas de Retrieval Augmented Generation (RAG) possibilita automatizar a análise de documentos jurídicos, uma tarefa complexa que tradicionalmente exige uma extensa revisão manual. Esse trabalho apresenta uma prova de conceito que visa demonstrar a viabilidade dessa integração no contexto de trabalho do Ministério Público do Estado do Rio Grande do Sul (MPRS), focando na análise de documentos do Tribunal de Contas do Estado do Rio Grande do Sul (TCE/RS) para identificar casos de irregularidade administrativa. A proposta inclui o desenvolvimento de um sistema capaz de realizar consultas em linguagem natural, recuperando e gerando respostas com base em informações relevantes dos documentos processados. A arquitetura combina aprendizado de máquina, processamento de linguagem natural e busca semântica, com o objetivo de otimizar o trabalho jurídico, garantindo precisão e eficiência. Resultados preliminares destacam a importância dessa abordagem para lidar com grandes volumes de texto assim como a necessidade de validação contínua das respostas geradas.

Palavras-chave: Large Language Model, Retrieval Augmented Generation, Análise Jurídica, Processamento de Linguagem Natural.

INTEGRATION OF LLM AND RAG FOR LEGAL DOCUMENT ANALYSIS

ABSTRACT

The use of Large Language Models (LLM) and Retrieval Augmented Generation (RAG) techniques offers the potential to automate the analysis of legal documents, a complex task that traditionally requires extensive manual review. This work presents a proof of concept aimed at demonstrating the feasibility of this integration in the context of Ministério Público do Estado do Rio Grande do Sul (MPRS), focusing on the analysis of documents from the Tribunal de Contas do Estado do Rio Grande do Sul (TCE/RS) to identify cases of administrative irregularities. The proposal includes the development of a system capable of performing natural language queries, retrieving and generating answers based on relevant information from processed documents. Architecture combines machine learning techniques, natural language processing and semantic search, with the goal of optimizing legal work, ensuring accuracy and efficiency. Preliminary results highlight the importance of this approach in handling large volumes of text as well as the need for continuous validation of generated responses.

Keywords: Large Language Model, Retrieval Augmented Generation, Legal Analysis, Natural Language Processing.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

SUMÁRIO

1. INTRODUÇÃO

No exercício de suas funções o Ministério Público do Estado do Rio Grande do Sul (MPRS) encontra na análise de documentos legais um aspecto importante para a fiscalização da conformidade jurídica. Essa atividade, conhecida por exigir extensas horas de análise manual detalhada, é fundamental para garantir a aderência às legislações e a defesa dos interesses da sociedade. O MPRS dispõe de uma variedade de ferramentas para combater as irregularidades no setor público, dentre elas estão as auditorias e inspeções conduzidas pelo Tribunal de Contas do Estado do Rio Grande do Sul (TCE/RS). A colaboração entre essas instituições é vital para o combate à corrupção e à improbidade administrativa.

Observa-se que a revolução em diversas áreas do conhecimento está sendo impulsionada pela inteligência artificial. A implementação de tecnologias como Large Language Model (LLM) e Retrieval Augmented Generation (RAG) abrem caminho para a automatização da análise e da classificação de documentos[2]. O setor jurídico, particularmente, está sendo transformado pela aplicação de tecnologias capazes de aprimorar processos tradicionais, como a pesquisa de jurisprudência e a revisão de documentos legais[1].

Avanços em tarefas complexas de Processamento de Linguagem Natural, incluindo compreensão de texto e resolução de questões, estão progredindo por meio do desenvolvimento de novas arquiteturas e algoritmos[2]. Diante disso, tais tecnologias surgem como ferramentas capazes de modernizar a maneira como o MPRS interage com os dados disponibilizados por diversas entidades públicas e privadas.

Nesse cenário, a técnica RAG apresenta uma série de vantagens para a implementação de soluções, pois permite a atualização contínua de informações e garante que os modelos de linguagem forneçam dados atualizados. Além disso, a

capacidade de citar fontes de informação nos resultados gera uma maior credibilidade contribuindo para o desenvolvimento de soluções mais confiáveis[3].

A metodologia adotada nesse trabalho de conclusão de curso é de pesquisa aplicada já que foca em resolver problemas específicos por meio de uma aplicação prática[12]. Por meio desse trabalho, propõe-se o desenvolvimento de uma prova de conceito para demonstrar a viabilidade da aplicação das tecnologias mencionadas na análise automatizada de documentos jurídicos relacionados à gestão pública. A ferramenta permitirá ao usuário fazer perguntas em linguagem natural em documentos relacionados à gestão da administração pública com foco na busca por irregularidades administrativas. Dessa forma, busca-se automatizar a revisão de documentos jurídicos e facilitar a identificação de informações relevantes disponibilizadas nos documentos do TCE/RS.

O trabalho está organizado com a apresentação da fundamentação teórica do tema no Capítulo 2, no qual é discutido Aprendizado de Máquina (AM), Processamento de Linguagem Natural (PLN), Large Language Model (LLM) e Retrieval Augmented Generation (RAG). No Capítulo 3 são tratados trabalhos relacionados considerando a relevância. O Capítulo 4 apresenta a proposta do estudo, seus objetivos, atividades planejadas e os recursos necessários para a realização do estudo.

No Capítulo 5 são especificados os recursos tecnológicos e os dados abertos que serão empregados, o que inclui tecnologias como LLM e RAG, e acessos aos dados do Tribunal de Contas do Estado do Rio Grande do Sul (TCE/RS). O Capítulo 6, intitulado "Arquitetura", apresenta o design do sistema que integra as tecnologias mencionadas e descreve como essa estrutura suporta a classificação de texto e a geração de respostas a partir de informações recuperadas.

No Capítulo 7, intitulado "Requisitos do Sistema", são definidos os requisitos funcionais e não funcionais do sistema, incluindo as especificações necessárias para o correto comportamento das funcionalidades descritas. O Capítulo

8 apresenta os "Casos de Uso", detalhando as interações esperadas entre o sistema e o usuário, descreve as funcionalidades-chave e os fluxos que serão implementados.

O Capítulo 9, "Diagramas de Sequência", ilustrada as dinâmicas entre diferentes componentes do sistema e mostra como as interações ocorrem em diferentes eventos e solicitações do usuário. Já o Capítulo 10 apresenta a interface de usuário proposta, incluindo o layout da tela com elementos que facilitam a usabilidade e acesso às funcionalidades disponíveis no sistema.

No Capítulo 11 são detalhadas as ferramentas e bibliotecas utilizadas no desenvolvimento. No Capítulo 12 são discutidos os modelos para embeddings e LLM que foram adotados para processamento. O Capítulo 13 aborda a implementação e principais aspectos do código Python desenvolvido. Por fim, o Capítulo 14 relata o processo de validação dos resultados com o objetivo de garantir a usabilidade do sistema.

2. FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo são brevemente discutidos conceitos essenciais ao estudo, apresentando a relevância do Aprendizado de Máquina (AM) e Processamento de Linguagem Natural (PLN) para a análise automatizada de textos. Adicionalmente, é abordada a integração de Large Language Models (LLM) e Retrieval Augmented Generation (RAG) que representa um avanço no campo do PLN ao oferecer fundamentos para a análise textual.

2.1. Aprendizado de Máquina

O contexto de aprendizado de máquina pode ser representado a partir do treinamento de uma máquina para prever um resultado com base em um grande conjunto de dados e, em seguida, utilizar o algoritmo treinado para explorar maneiras de otimizar a eficiência no domínio ao qual está sendo aplicado. A implementação das

técnicas de AM facilita a identificação de padrões e correlações que seriam difíceis de detectar manualmente[7].

A partir de sua capacidade de aprender ao analisar dados, os algoritmos de AM solucionam tarefas que seriam muito complexas para serem resolvidas por métodos convencionais. Essas tarefas geralmente são definidas em termos de como o sistema deve processar um exemplo com uma coleção de características.

Constata-se que por meio da análise dessas características os algoritmos aprendem padrões nos dados e tornam possível, por exemplo, fazer previsões com base em novas entradas. Por esse avanço, o aprendizado de máquina possui aplicações em tarefas como reconhecimento de imagens e personalização de tratamentos na área da saúde[8].

Percebe-se que essa tecnologia é útil em contextos nos quais há grandes volumes de dados e com relações claras entre entradas e saídas. Sistemas baseados em AM são eficazes para automatizar decisões, melhorar previsões e análises baseadas em padrões encontrados em conjuntos de dados. Por outro lado, é menos adequado para tarefas que exigem raciocínio complexo ou conhecimento contextual amplo[7].

2.2. Processamento de Linguagem Natural

O Processamento de Linguagem Natural é um campo de estudo focado em explorar e desenvolver técnicas para que sistemas tenham capacidade de processar a linguagem humana. Dessa forma, busca-se encontrar maneiras de resolver problemas computacionais relacionados ao entendimento e geração de linguagem humana por meio de computadores[4]. Os avanços do PLN têm sido impulsionados pelo desenvolvimento de modelos baseados em LLM. Contudo, apesar dos avanços, o PLN enfrenta desafios significativos tais como entender completamente o contexto com os detalhes da linguagem humana.

Observa-se que PLN é um campo de grande potencial na Inteligência Artificial (IA) marcado por desafios que não podem ser resolvidos apenas com os

avanços em hardware devido à complexidade da linguagem. Por outro lado, a demanda por sistemas avançados de PLN estimula a criação de novos métodos de desenvolvimento e aplicações práticas[4].

Nesse contexto de Processamento de Linguagem Natural, percebe-se que a classificação de texto e a recuperação de informação têm recebido crescente interesse por seus avanços recentes. Essa ênfase é motivada pela eficácia dos algoritmos em compreender padrões complexos e relações nos dados[5]. A partir dessas evoluções, o PLN avança como campo de estudo e amplia seu impacto na sociedade ao transformar a maneira como interagimos com computadores.

2.3. Large Language Model

Entende-se por LLM ou Large Language Model, um modelo de inteligência artificial projetado para interpretar e gerar texto humano. Esses modelos são treinados com vastas quantidades de dados textuais permitindo-lhes capturar variações linguísticas, contextos diferentes e até mesmo geração de textos relevantes em resposta a prompts específicos[4]. Estudos recentes sugerem que Large Language Models possuem um grande potencial para avaliação da relevância de documentos jurídicos[7].

Os modelos de inteligência artificial treinados com vastas quantidades de texto podem ser capazes de determinar o quão relevantes são certos documentos ou casos jurídicos uns em relação aos outros. A ideia é que os LLMs possam facilitar o processo de identificação de conexões entre diferentes textos jurídicos, o que normalmente exige um esforço humano significativo de leitura e análise[6].

Observa-se que os LLMs não são vinculados a nenhuma fonte de informação e devem ser treinados para incorporar informações atualizadas que não foram vistas durante o treinamento. Essa limitação destaca a importância de continuamente atualizar e refinar esses modelos para manter sua relevância na geração de textos[2].

É inegável que os avanços em Large Language Models melhoraram a qualidade dos textos gerados por máquinas em muitos casos de uso. No entanto, essa tarefa apresenta limitações no acesso de conhecimento externo já que a inclusão de informações novas no modelo pode ser um desafio.

2.4. Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) é uma abordagem no campo da inteligência artificial generativa que combina a capacidade de geração de textos com técnicas de recuperação de informações. Para melhorar a qualidade do conteúdo gerado essa abordagem proporciona uma busca por informações em fontes externas como banco de dados ou documentos[9].

Dentro dessa estrutura, um componente de recuperação realiza a extração de informações relevantes de um repositório de dados e um componente gerativo aplica essas informações como contexto para criar uma resposta[10]. Essa capacidade faz a RAG ser útil em tarefas que exigem conhecimento mais detalhado e atualizado.

A partir do uso de LLMs em conjunto com um componente de recuperação, observa-se uma melhora nas respostas que assim são desenvolvidas por meio de um ciclo de interação entre recuperação e geração. Dessa forma, amplia-se a capacidade dos modelos de entenderem e responderem às consultas com maior precisão e também permite a adaptação a domínios específicos[9]. Essa colaboração transforma os modelos em ferramentas capazes de responder com acuidade em diversas áreas de conhecimento.

Como qualquer sistema de inteligência artificial essa enfrenta desafios que precisam ser superados. Por sua dependência da disponibilidade de dados de alta qualidade e de dados relevantes, problemas com fontes desatualizadas ou imprecisas podem resultar na recuperação de informações irrelevantes[10].

3. TRABALHOS RELACIONADOS

Nesse capítulo de trabalhos relacionados são abordadas as aplicações de LLM e RAG em contextos educacionais e jurídicos, destacando tanto os avanços tecnológicos quanto os desafios enfrentados. Discute-se como esses modelos estão sendo adaptados para fornecer soluções personalizadas.

Em[11] é apresentada uma aplicação web projetada para fornecer tutoria personalizada em qualquer disciplina acadêmica. O estudo propõe o desenvolvimento de um tutor de inteligência artificial (AI Tutor) capaz de se adaptar a qualquer curso e fornecer respostas precisas e contextualizadas utilizando técnicas de LLM e RAG.

A aplicação web desenvolvida permite a ingestão de materiais de um curso para construir uma base de conhecimento específica. Quando um estudante faz uma pergunta, o AI Tutor recupera as informações mais relevantes dessa base de conhecimento e gera respostas detalhadas em formato de conversação sendo capaz de indicar a fonte de onde as informações foram extraídas.

Nesse contexto, os materiais de cursos são os documentos de uma disciplina que servem como base de conhecimento para o tutor de IA. Os materiais são carregados pelo usuário e posteriormente são armazenados no banco de dados vetorial. A partir de seu carregamento, os materiais (dados) são convertidos em representações vetoriais usando um modelo de embedding pré-treinado.

Observa-se que para os usuários, a interação com o programa ocorre fazendo perguntas em linguagem natural. A partir de uma busca baseada em similaridade, o programa compara o vetor da pergunta do usuário com os vetores armazenados dos materiais do curso. Dessa forma, é possível identificar os blocos de texto mais relevantes que podem fornecer a resposta à pergunta do usuário.

Após a recuperação dos blocos de texto relevantes o sistema usa um modelo de LLM para gerar uma resposta em linguagem natural. A técnica RAG permite que o modelo de linguagem não dependa apenas do conhecimento inicialmente

armazenado mas também se baseie nas informações recuperadas dos materiais do curso.

Já em[7] temos uma análise do uso de LLM em casos jurídicos que destaca a complexidade em avaliar a relevância entre casos. Percebe-se que a necessidade de compreender textos extensos e aplicar conhecimentos jurídicos específicos torna a tarefa mais complexa. O trabalho propõe um método que permite a modelos LLM avaliar a relevância de casos jurídicos sem necessitar da construção de modelos específicos para o domínio jurídico, o que é custoso e complexo.

Observa-se que a arquitetura da solução propõe uma Análise Factual Preliminar realizada por profissionais do Direito a partir dos fatos do caso. Essa análise detalha os "Fatos Materiais" (MF) e "Fatos Legais" (LF) que determinam a relevância de um caso. Essa fase ajuda a orientar o LLM na compreensão do contexto e na definição da relevância nos julgamentos.

No componente Adaptive Demo-Matching (ADM) ocorre a seleção de exemplos relevantes que correspondem aos casos de entrada. Esses exemplos são usados como referências para orientar o LLM em cada etapa do processo de anotação. Após a seleção dos exemplos, o LLM realiza a extração dos "Fatos Materiais" (MF) e "Fatos Legais" (LF) dos casos analisados. Essa extração é feita sequencialmente onde primeiro são identificados os Fatos Materiais, seguidos pelos Fatos Legais.

Por fim, a partir da extração dos fatos o LLM passa a avaliar a relevância entre pares de casos. A anotação final fornece uma pontuação de relevância para cada par de casos, baseada na similaridade dos fatos materiais e legais. Os dados anotados pelo LLM são usados para construir um conjunto de dados utilizado para treinar modelos de recuperação de casos jurídicos.

4. PROPOSTA

Esse trabalho de conclusão de curso busca desenvolver uma prova de conceito de um sistema que possui uma arquitetura baseada em LLM e RAG para automatizar a análise de documentos jurídicos. A partir de um protótipo, que aplica as tecnologias mencionadas, será possível identificar casos de irregularidades administrativas registrados nos documentos de auditorias e nas decisões do TCE/RS. A eficácia do sistema será avaliada por meio de demonstrações de interação Q&A com uma análise que se concentra na precisão das respostas geradas.

4.1. Objetivo Geral

O objetivo geral desse trabalho é validar a viabilidade de automatização da análise de documentos jurídicos ao utilizar LLM e RAG. Justifica-se estudar essas tecnologias pois o seu desenvolvimento pode transformar processos de trabalho que são tradicionalmente manuais e demorados no âmbito jurídico estadual.

4.2. Objetivos Específicos

1. Realizar a identificação dos tipos e a extração de documentos a serem analisados.

2. Explorar a implementação de uma arquitetura baseada em LLM e RAG que possa ser aplicada nesse contexto de análise de documentos.

3. Desenvolver um protótipo funcional capaz de automatizar essa análise buscando aprimorar a eficiência na identificação dos casos de irregularidade administrativa.

Os tipos de documentos relevantes para essa análise são Comunicado de Auditoria, Decisão e Relatório de Auditoria. Esses documentos identificam os possíveis casos de irregularidades administrativas os quais geralmente são documentados por meio de expressões específicas. As expressões incluem termos como “irregularidades”, “enriquecimento ilícito”, "dano ao erário", "improbidade", "superfaturamento", "sobrepreço", "crime", "fraude", "renúncia de receita", "renúncia

fiscal", entre outras. Além disso, várias formas de prejuízo são mencionadas, como "prejuízo ao erário", "prejuízo aos cofres", "prejuízo financeiro", "prejuízo à administração", "prejuízo de R$", "prejuízo apurado". A análise desses termos nos documentos ajuda na detecção e na investigação de possíveis irregularidades administrativas.

5. RECURSOS A SEREM UTILIZADOS

Para o desenvolvimento do trabalho proposto foi necessário realizar um estudo sobre LLM e RAG assim como considerar suas aplicações na análise automatizada de documentos. A compreensão dessas tecnologias é importante para avaliar como podem ser aplicadas eficazmente na classificação dos documentos que são objeto do estudo. Ademais, considerou-se essencial explorar as melhores práticas de desenvolvimento para garantir a precisão nos resultados obtidos.

Além das tecnologias mencionadas, o acesso aos dados abertos ao público que são disponibilizados pelo Tribunal de Contas do Estado do Rio Grande do Sul (TCE/RS) se apresentou como um recurso fundamental para o desenvolvimento desse projeto. A disponibilidade e a qualidade dos dados permitem uma análise esclarecedora desses documentos jurídicos. Na Figura 5.1 abaixo temos um modelo de Relatório de Auditoria disponível para download no portal do TCE/RS.

ESTADO DO RIO GRANDE DO SUL
TRIBUNAL DE CONTAS
DIRECÃO DE CONTROLE E FISCALIZACÃO
Serviço Regional de Auditoria de Santo Ângelo

Tribunal de Contas
Fl. 0 Rubrica

Página 122
Processo 04080-0200/23-2
Página da peça 1
Peça 5589064
DOCUMENTO PÚBLICO

RELATÓRIO DE AUDITORIA DE ADMISSÕES
CONCURSOS E PROCESSOS SELETIVOS PÚBLICOS

PROCESSO Nº	ORDEM DE AUDITORIA Nº
004080-0200/23-2	3088/2023-1

UNIDADE AUDITADA: Executivo Municipal
MUNICÍPIO: Santo Ângelo
ADMINISTRADORES RESPONSÁVEIS: [redacted] (2004)
[redacted] (2012)
PERÍODO DE EXAME: 1º/12 a 05/12/2023
PERÍODO DE VERIFICAÇÃO: 30/11/2023
EQUIPE DE AUDITORIA: [redacted]

A presente análise fundamenta-se no disposto no inciso III do artigo 71 da Constituição Federal, artigo 71 da Constituição Estadual e Resolução 1.051/2015, deste Tribunal, conforme suas vigências, objetivando o exame da legalidade de atos de admissão para fins de registro.

Foram constatadas **05 (cinco)** admissões relativas a período anterior ao ora examinado, sendo autoridades responsáveis os Srs. José Lima Gonçalves (2004) e Eduardo Debacco Loureiro (2012).

Na presente auditoria, foram constatadas **05 (cinco)** admissões por **Concurso Público**.

Figura 5.1 – Relatório de Auditoria do TCE/RS (Fonte: [26]).

6. ARQUITETURA

Considera-se para a tarefa modelada no sistema uma combinação de classificação de texto e geração de resposta baseada em informações recuperadas. Dessa forma, busca-se desenvolver um protótipo funcional que possa, a partir de perguntas em linguagem natural, identificar automaticamente e reportar casos de irregularidades administrativas documentadas nos relatórios do TCE/RS.

Na arquitetura mostrada na Figura 6.1 abaixo, propõe-se um sistema que trabalha com pesquisa semântica utilizando modelos de linguagem e técnicas de processamento de linguagem natural (PLN) para responder a perguntas com base em documento fornecido[3].

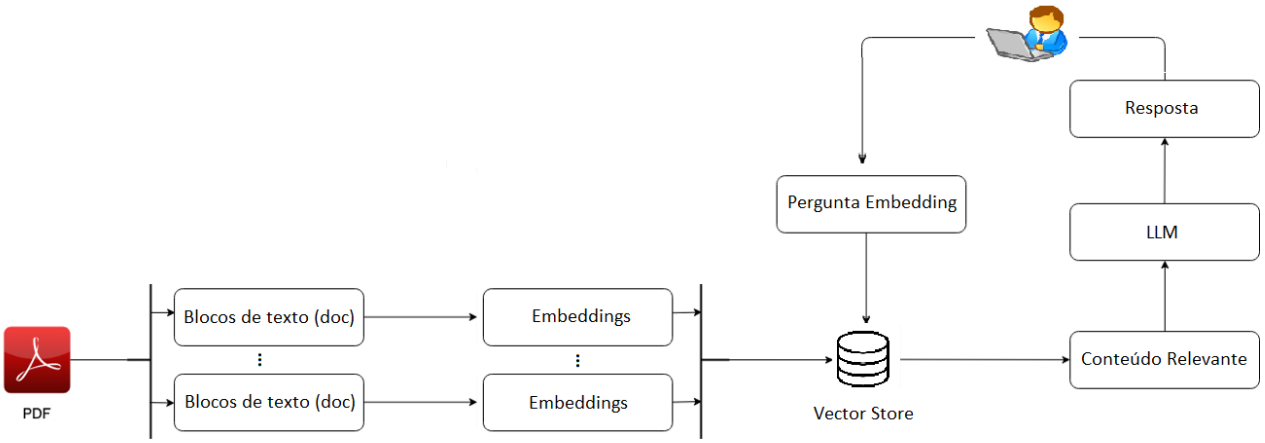

Figura 6.1 – Arquitetura do sistema (Adaptação fonte:[3]).

Verifica-se, inicialmente, que um documento em formato PDF tem o seu texto extraído para ser uma fonte de dados. Logo após, ocorre a divisão desse texto em "chunks" ou blocos de texto. Isso facilita o processamento subsequente já que trabalhar com pedaços menores de texto pode melhorar a precisão e a eficiência da análise[24].

Posteriormente cada bloco de texto é convertido em embeddings que são representações vetoriais de texto que capturam o significado semântico das palavras e frases. Eles são essenciais para permitir comparações semânticas entre diferentes blocos de texto[13]. Os embeddings gerados são armazenados em um vetor de armazenamento criando um repositório que permite consultas rápidas.

Quando uma pergunta é feita, ela também é convertida em um embedding permitindo que seja comparada semanticamente com os embeddings existentes no armazenamento. Utilizando o embedding da pergunta, realiza-se uma busca semântica nos embeddings dos documentos para encontrar os mais relevantes. Essa busca identifica quais blocos de texto têm maior semelhança semântica com a pergunta[13]. Um ranking é feito com base na similaridade dos embeddings dos resultados da busca semântica e os resultados mais relevantes são selecionados para formular a resposta final.

A partir de um modelo de linguagem de grande escala gera-se uma resposta coerente e contextualizada com base nos resultados. Esse modelo pode sintetizar informações de vários blocos de texto e gerar uma resposta que seja informativa e bem formulada[4]. Contudo, o modelo de linguagem não sabe exatamente o que o documento PDF contém. Após o sistema encontrar os blocos de texto que são relevantes para a pergunta do usuário e apresentá-los em ordem de importância, eles são enviados como contexto para o modelo de linguagem.

Por fim, a resposta gerada pelo LLM é entregue ao usuário completando o ciclo de consulta. Essa arquitetura, que combina PLN e modelos de linguagem para fornecer respostas extraídas de grandes volumes de texto não estruturado, é bastante útil para aplicações como assistência jurídica e pesquisa acadêmica já que informações detalhadas precisam ser extraídas rapidamente de vastos repositórios de textos[13].

7. REQUISITOS DO SISTEMA

A Engenharia de Requisitos tem a responsabilidade de coletar e definir corretamente as necessidades do usuário. Trata-se de uma etapa importante para garantir que o produto final esteja em conformidade com os requisitos estabelecidos[14]. Essa etapa da construção do sistema define as funções e

comportamentos que ele deverá ter conforme as necessidades levantadas com o usuário.

Nesse contexto, temos nos requisitos funcionais a descrição de como o sistema deve se comportar em resposta a entradas. Eles detalham as funcionalidades que precisam ser desenvolvidas para atender às necessidades do usuário. Por outro lado, nos requisitos não funcionais encontramos as restrições de sistemas que geralmente se aplicam ao produto como um todo, como desempenho e segurança[14].

7.1. Requisitos Funcionais (RF)

RF01 - Extração do texto de documentos

- O sistema deve ser capaz de processar documentos em formato PDF e extrair o conteúdo textual para análise.

- O texto extraído deve ser segmentado em blocos (chunks) de tamanho adequado para processamento eficiente.

RF02 - Geração de embeddings

- O sistema deve gerar representações vetoriais (embeddings) para cada bloco de texto extraído.

- Esses embeddings devem capturar o significado semântico dos textos de modo a permitir buscas por similaridade.

RF03 - Armazenamento de embeddings

- Os embeddings gerados devem ser armazenados em um repositório de vetores (vector store) para permitir consultas semânticas.

RF04 - Busca semântica

- O sistema deve permitir a entrada de perguntas em linguagem natural por parte do usuário.

- A pergunta do usuário deve ser convertida em embeddings para ser comparada semanticamente com os embeddings armazenados no banco de vetores.

- O sistema deve retornar os blocos de texto mais relevantes de acordo com a similaridade semântica entre a pergunta e o documento.

RF05 - Geração de respostas

- Com base nos blocos de texto mais relevantes o sistema deve usar um LLM para gerar uma resposta em linguagem natural para a pergunta do usuário.

- A resposta deve ser contextualizada e utilizar as informações recuperadas do documento.

RF06 - Interface de usuário (UI)

- O sistema deve fornecer uma interface de usuário (UI) que permita a inserção de perguntas e a visualização das respostas.

- Deve haver uma área para upload de documentos e outra para exibir as respostas geradas pelo modelo.

RF07 - Logs e auditoria

- O sistema deve registrar logs das interações incluindo as perguntas feitas e os documentos consultados, para fins de auditoria.

7.2. Requisitos Não Funcionais (RNF)

RNF01 - Desempenho

- O sistema deve ser capaz de processar um documento em tempo razoável mantendo a eficiência da busca e geração de respostas.

- A busca semântica deve ser realizada com tempo de resposta abaixo de 10 segundos para consultas simples.

RNF02 - Manutenibilidade

- O código deve seguir boas práticas de programação, facilitando futuras manutenções e atualizações.

8. CASOS DE USO

Nessa seção é apresentada uma proposta para a descrição de casos de uso do sistema. Entende-se por casos de uso uma técnica de descoberta de requisitos que identificam os atores envolvidos em uma interação. Dessa forma, eles nomeiam o tipo de interação e às complementam por meio de descrições textuais ou modelos gráficos, como diagramas de sequência[15]. Isso ajuda a capturar o fluxo completo de atividades ao destacar etapas e resultados esperados.

A UML não define uma estruturação específica a ser utilizada na descrição de um caso de uso. Por causa disso, há diferentes propostas de descrição. A estrutura de um caso de uso pode incluir etapas como Nome, Atores, Precondições, Fluxo principal, Fluxos alternativos, Fluxos de exceção e Pós-condições[16]. Dessa forma, essa estrutura oferece uma visão geral dos componentes necessários para uma documentação detalhada.

Caso de Uso 1: Upload de Documento

Descrição: O usuário faz o upload de um documento (PDF) no sistema para análise.

Atores: Usuário.
Precondições: O sistema deve estar disponível e operacional.

Fluxo Principal:

1. O usuário acessa a interface de upload do sistema.

2. O sistema solicita ao usuário o documento a ser enviado.

3. O usuário seleciona o arquivo e confirma o envio.

4. O sistema valida o formato do documento (PDF).

5. O sistema confirma o recebimento e inicia o processamento do documento (extração de texto).

Fluxo Alternativo:

- Se o documento estiver em um formato não suportado, o sistema exibe uma mensagem de erro.

Exceções:

- Documento corrompido: o sistema exibe uma mensagem de erro.

Pós-condições: O documento é armazenado e pronto para extração de texto e processamento.

Caso de Uso 2: Extração de Texto e Geração de Embeddings

Descrição: Após o upload de um documento, o sistema extrai o conteúdo textual e gera embeddings para processamento semântico.

Atores: Sistema.
Precondições: Documento deve estar armazenado no sistema após o upload.

Fluxo Principal:

1. O sistema realiza a leitura do documento e extrai o texto (usando PyPDF2).

2. O sistema divide o texto em blocos (chunks) para facilitar o processamento.

3. O sistema envia cada bloco para a API da OpenAI para geração de embeddings (assíncrono).

4. Os embeddings gerados são armazenados em um repositório de vetores (FAISS).

Exceções:

- Ocorre erro na extração de texto: o sistema notifica o usuário e interrompe o processamento.

Pós-condições: O texto do documento é extraído com sucesso, dividido em blocos, e os embeddings são armazenados para consultas futuras.

Caso de Uso 3: Consulta em Linguagem Natural

Descrição: O usuário faz uma pergunta em linguagem natural relacionada ao documento. O sistema utiliza embeddings para encontrar os trechos mais relevantes no documento carregado e responde com base nesses trechos.

Atores: Usuário, Sistema.

Precondições:

- O documento deve estar processado e seus embeddings armazenados no sistema.

- A cadeia de conversação (ConversationalRetrievalChain) deve estar configurada e armazenada em st.session_state.conversation.

Fluxo Principal:

1. O usuário acessa a interface de consulta.

2. O sistema exibe um campo para o usuário inserir sua pergunta em linguagem natural.

3. O usuário insere a pergunta.

4. O sistema, por meio de st.session_state.conversation, envia a pergunta ao ConversationalRetrievalChain para geração de uma resposta.

5. A ConversationalRetrievalChain realiza a busca no repositório de vetores (FAISS) usando a API da OpenAI para comparar a pergunta com os embeddings dos blocos de texto.

6. A ConversationalRetrievalChain retorna os blocos de texto mais relevantes e utiliza um LLM para gerar uma resposta contextualizada.

7. O sistema exibe a resposta ao usuário juntamente com a história da conversa.

Fluxo Alternativo:

- Se não houver blocos relevantes, o sistema notifica o usuário que nenhuma informação correspondente foi encontrada.

Exceções:

- Ocorre erro na geração dos embeddings da pergunta ou na busca semântica: o sistema exibe uma mensagem de erro.

Pós-condições:

- O usuário obtém uma resposta gerada automaticamente.
- O histórico de conversa é atualizado e exibido na interface.

Caso de Uso 4: Visualização de Resultados

Descrição: O usuário visualiza os resultados da consulta por meio de uma resposta gerada pelo sistema.
Atores: Usuário.
Precondições: O sistema deve ter retornado blocos de texto relevantes e gerado uma resposta.

Fluxo Principal:

1. O sistema apresenta a resposta gerada pelo LLM com base nos blocos de texto mais relevantes.
2. O usuário visualiza a resposta.

Fluxo Alternativo:

- O usuário pode refinar a consulta modificando a pergunta e o sistema refaz a busca semântica.

Exceções:

- Caso ocorra erro na visualização o sistema exibe uma mensagem de erro.

Pós-condições: O usuário revisa as informações extraídas.

Caso de Uso 5: Registro de Logs e Auditoria

Descrição: O sistema registra as interações do usuário, incluindo uploads de documentos e consultas feitas para fins de auditoria.
Atores: Sistema, Administrador.
Precondições: O sistema deve estar configurado para registrar logs.

Fluxo Principal:

1. O sistema registra automaticamente cada documento enviado para upload.

2. Cada consulta realizada pelo usuário é registrada com a pergunta.

3. O sistema gera logs detalhados de cada resposta fornecida ao usuário, incluindo referências aos documentos.

Exceções:

- Se ocorrer falha no sistema de registro de logs, o programa deve apresentar um erro.

Pós-condições: Todas as interações do usuário com o sistema são devidamente registradas para fins de auditoria.

9. DIAGRAMAS DE SEQUÊNCIA

Os diagramas de sequência são utilizados na UML para modelar a interação entre os atores e objetos de um sistema demonstrando a ordem dos eventos que ocorrem durante um caso de uso[15]. Eles representam como mensagens são trocadas durante a execução de um caso de uso, ajudando a compreender o fluxo.

Na fase de desenvolvimento inicial, diagramas de sequência são usados para apoiar a engenharia de requisitos e o design de alto nível. Nesse momento não é necessário incluir todas as interações que podem depender de decisões de implementação futuras[15]. Com isso, eles auxiliam no desenvolvimento mais estruturado permitindo melhorias conforme o projeto avança.

O diagrama de sequência na Figura 9.1 mostra o fluxo de upload de um documento entre o usuário e o sistema. O usuário acessa a interface, seleciona o documento e o sistema valida o formato do arquivo. Em seguida, o sistema confirma o recebimento ou exibe uma mensagem de erro caso o upload falhe.

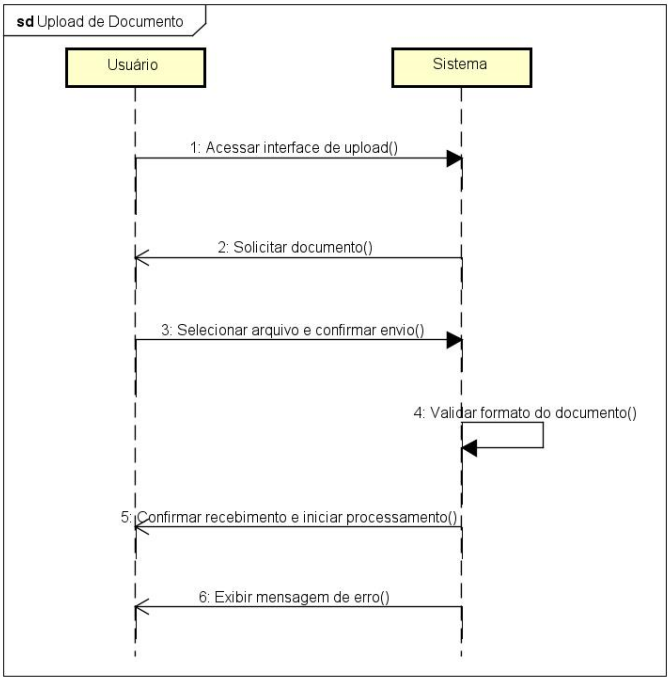

Figura 9.1 - Upload de documento (Fonte: Autor).

No diagrama de sequência na Figura 9.2 observa-se o processo de extração de texto e geração de embeddings entre o sistema e a API da OpenAI. O sistema extrai o texto, divide-o em blocos e envia para a API gerar embeddings. Em seguida, os embeddings são armazenados no repositório. O sistema notifica o usuário em caso de erro.

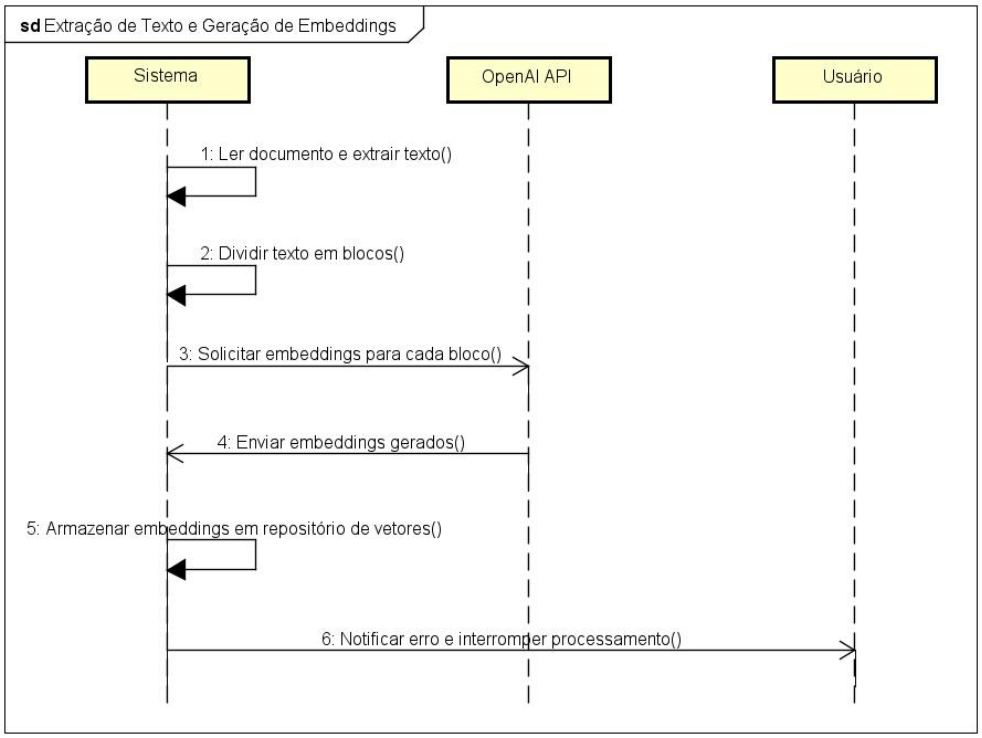

Figura 9.2 – Extração de Texto e Geração de Embeddings (Fonte: Autor).

Todo o processo de consulta em linguagem natural entre o usuário, o sistema, a ConversationalRetrievalChain, o FAISS, e a API da OpenAI está representado no diagrama de sequência na Figura 9.3. O usuário insere uma pergunta que é processada pelo sistema, a ConversationalRetrievalChain busca blocos relevantes nos embeddings e gera uma resposta por meio da API da OpenAI. Em seguida, a resposta é exibida ao usuário ou uma mensagem de erro é retornada em caso de falha.

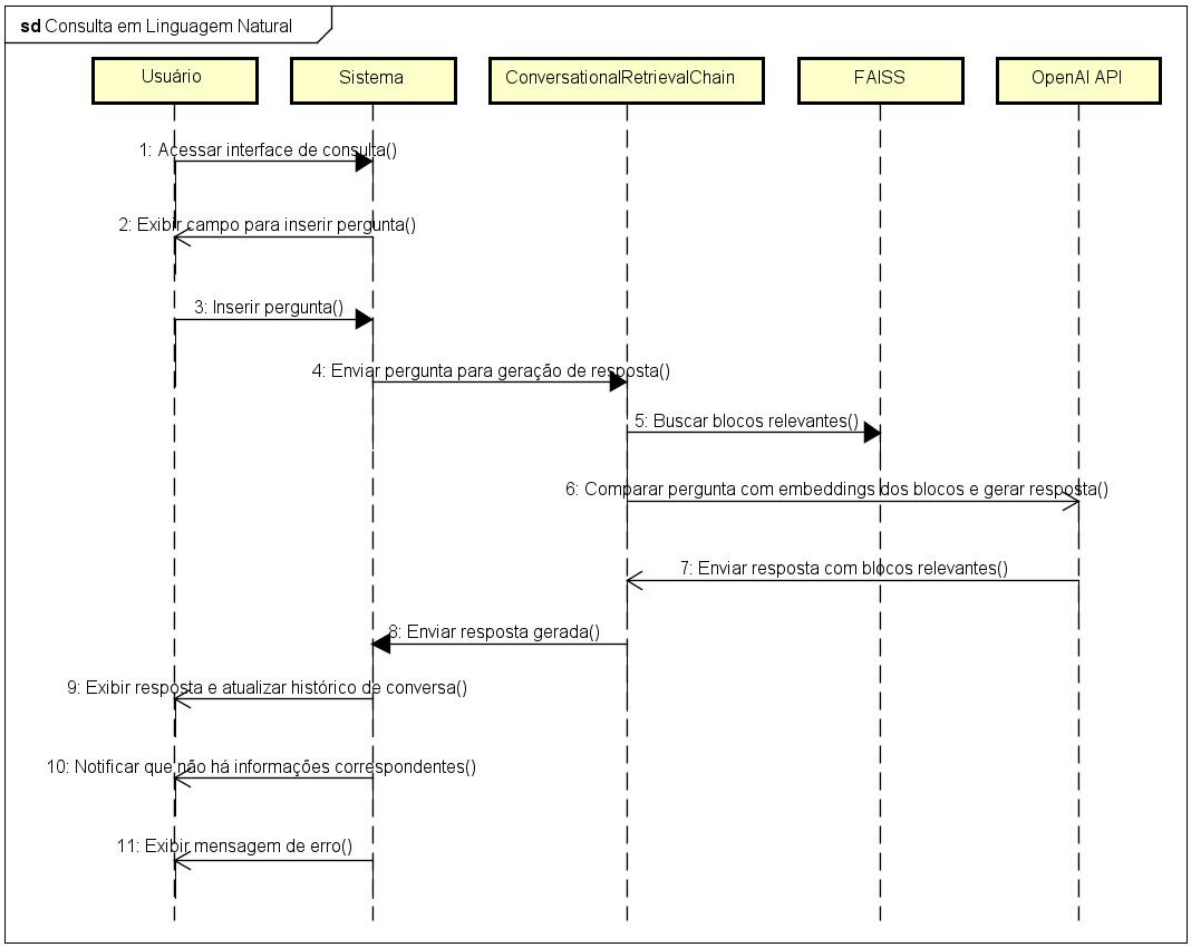

Figura 9.3 – Consulta em Linguagem Natural (Fonte: Autor).

O processo de visualização de resultados entre o sistema e o usuário está representado no diagrama de sequência na Figura 9.4. O sistema exibe uma resposta gerada com base nos blocos relevantes e o usuário visualiza essa resposta. Se necessário, o usuário pode modificar a pergunta e o sistema refaz a busca semântica exibindo uma nova resposta ou uma mensagem de erro no caso de algum problema.

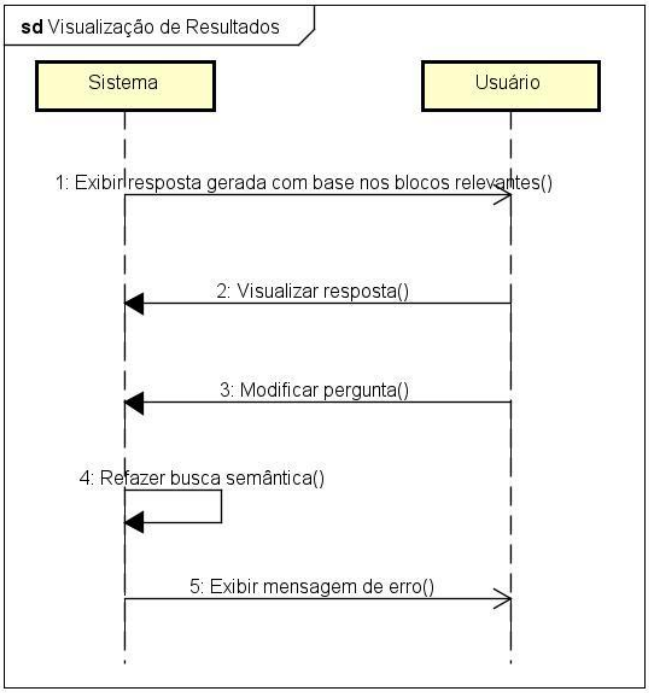

Figura 9.4 – Visualização de Resultados (Fonte: Autor).

Já o diagrama de sequência na Figura 9.5 ilustra o processo de registro de logs entre o Sistema e o Administrador. O sistema registra eventos importantes, como o envio de documentos, consultas realizadas e respostas fornecidas ao usuário. Em seguida, ele permite o acesso aos logs para auditoria ou notifica o administrador em caso de erros.

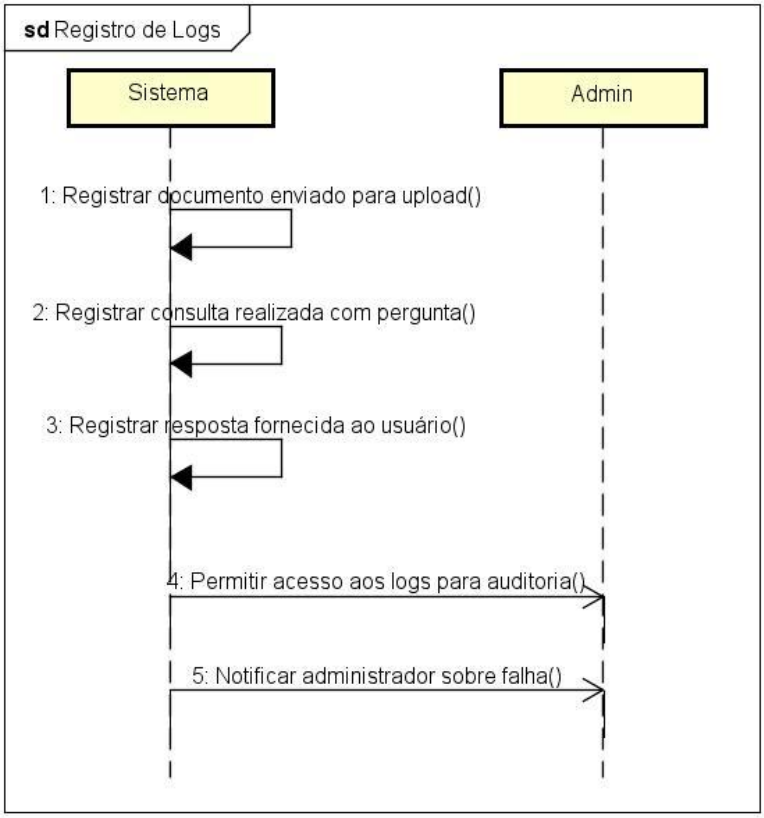

Figura 9.5 – Registro de Logs (Fonte: Autor).

10. INTERFACE DE USUÁRIO

A Figura 10.1 representa a interface de usuário da aplicação web. Ela representa a tela do sistema de processamento de documentos com a funcionalidade de pesquisa. A ideia geral é disponibilizar uma interface para o usuário carregar um documento, processá-lo e, em seguida, fazer perguntas e receber respostas sobre o conteúdo do documento.

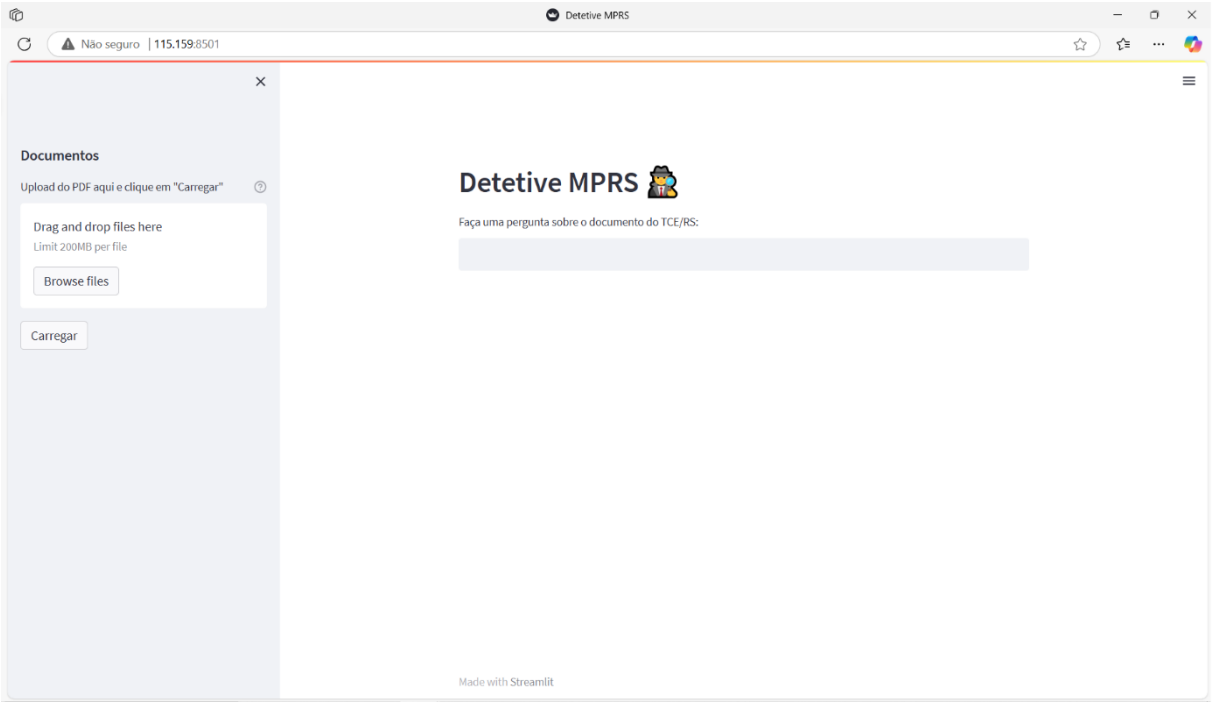

Figura 10.1 – Tela da aplicação web (Fonte: Autor).

1. **Seção de documentos** (lado esquerdo):

- Título: "Documentos".
- Instrução: "Selecionar e clicar em carregar".
- Botão: "Selecionar", permitindo que o usuário selecione um documento.
- Botão: "Carregar" para iniciar o processamento do documento selecionado.

2. **Seção de pesquisa** (lado direito):

- Título: "Detetive MPRS".

- Campo de texto: "Faça uma pergunta sobre o documento do TCE/RS", onde o usuário pode inserir uma pergunta relacionada ao conteúdo do documento carregado.

11. TECNOLOGIAS E FERRAMENTAS

Nesse capítulo estão descritas as principais tecnologias e ferramentas utilizadas para o desenvolvimento do sistema. Python é a linguagem de programação escolhida para a implementação já que ela facilita o desenvolvimento e a integração de diferentes componentes necessários para o funcionamento do sistema.

Para a criação da interface web foi utilizado o Streamlit pois essa ferramenta permite de forma simples a construção de uma interface na qual o usuário pode realizar o upload de arquivos, interagir com o sistema e visualizar os resultados gerados pelo modelo de linguagem[17].

A biblioteca dotenv foi utilizada para o carregamento de variáveis de ambiente a partir de um arquivo env. Essa biblioteca armazena e acessa dados sensíveis (chave API da OpenAI) sem expor informações no código fonte. Para a leitura e extração de texto dos documentos enviados, a biblioteca PyPDF2 foi utilizada pois permite processar arquivos PDF para extrair o conteúdo textual necessário para posterior análise e recuperação de informações[18].

Já o framework empregado, LangChain, foi escolhido por facilitar a integração de modelos de linguagem e ferramentas de recuperação de informações[13]. Dentro do LangChain, os componentes utilizados são:

- *CharacterTextSplitter*: usado para dividir o texto extraído dos PDFs em blocos menores (chunks), facilitando o processamento.

- *OpenAIEmbeddings*: para a geração de embeddings a partir dos textos. Esses embeddings são representações vetoriais que viabilizam a busca semântica[13].

- *FAISS*: é a biblioteca de pesquisa vetorial que armazena os embeddings e possibilita buscas por similaridade para o sistema responder às perguntas dos usuários de forma relevante[19].

- *ChatOpenAI*: trata-se de um modelo de linguagem desenvolvido pela OpenAI que interage com o usuário respondendo às perguntas realizadas.

- *ConversationalRetrievalChain*: ferramenta que cria uma cadeia de recuperação de informações com base em conversas anteriores permitindo ao sistema lembrar o histórico de interações e melhorar a experiência entre perguntas e respostas[20].

Para o monitoramento do sistema e o registro de atividades a biblioteca logging foi empregada. Ela possibilita a geração de logs ao registrar o processamento dos documentos e as interações do usuário[21]. Dessa forma, colabora com auditorias, análise de desempenho e identificação de problemas. Por fim, templates em HTML fazem a estilização das perguntas e respostas exibidas na interface do usuário. Dessa forma, a apresentação do conteúdo ocorre de maneira organizada e proporciona uma melhor experiência visual para o usuário.

12. MODELOS

Para o desenvolvimento do sistema são empregados os modelos text-embedding-ada-002 e GPT-4o da OpenAI já que suas características permitem atender diferentes tarefas de processamento de linguagem natural (PLN).

12.1. text-embedding-ada-002

Modelo projetado para gerar embeddings desenvolvido pela OpenAI que oferece uma boa relação custo-benefício. Apresenta bom desempenho em blocos contendo 1000 caracteres sendo projetado para diversas tarefas de PLN como busca semântica e similaridade de texto[25]. O text-embedding-ada-002 é elogiado por sua eficiência em termos de custo já que é mais barato que modelos maiores como os da família davinci[22].

12.2. GPT-4o

O modelo de linguagem natural GPT-4o da OpenAI se destaca pela capacidade de compreensão e geração de texto natural em diversos domínios. O modelo é treinado em uma ampla variedade de textos para poder prever a próxima palavra em uma sequência. Assim, ele é útil para tarefas de análise textual[23].

13. CÓDIGO PYTHON

Esse capítulo aborda a implementação e principais aspectos do código Python utilizado no presente trabalho. São detalhados os passos seguidos para a concepção do script e explicações sobre a lógica implementada com o objetivo de facilitar a compreensão do funcionamento e sua aplicação no contexto do problema estudado.

Código 13.1 – Trecho do código para logar documento.

```python
def log_documento(pdf_reader):
    """Função para extrair e logar os metadados de um documento."""
    metadata = pdf_reader.metadata
    if metadata:
        logging.info("Metadados do documento:")
        for key, value in metadata.items():
            logging.info(f"{key}: {value}")
    else:
        logging.info("Nenhum metadado encontrado no documento.")
```

Código 13.2 – Trecho do código para extrair o texto do documento.

```
def get_texto(pdf_docs):
    """Função para extrair o texto do documento."""
    start_time = time.time()
    text = ""
    for pdf in pdf_docs:
        if not pdf.name.lower().endswith('.pdf'):
            logging.warning(f"Arquivo ignorado, não é um PDF: {pdf.name}")
            continue

        try:
            pdf_reader = PdfReader(pdf)
            log_documento(pdf_reader)

            for page in pdf_reader.pages:
                text += page.extract_text()

            logging.info(f"Nome do documento: {pdf.name}")

        except Exception as e:
            logging.error(f"Erro ao processar {pdf.name}: {e}")

    end_time = time.time()
    logging.info(
        f"Tempo para extrair texto: {end_time - start_time:.2f} segundos")

    pdf_names = [
        pdf.name for pdf in pdf_docs if pdf.name.lower().endswith('.pdf')]
    logging.info(f"Documentos processados: {', '.join(pdf_names)}")
    return text
```

Código 13.3 – Trecho do código para dividir o texto em blocos.

```
def get_chunks(texto):
    """Função para dividir o texto em blocos (chunks)."""
    start_time = time.time()
    splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = splitter.split_text(texto)
    end_time = time.time()
    logging.info(f"Número de chunks gerados: {len(chunks)}")
    logging.info(
        f"Tempo para dividir o texto em chunks: {end_time - start_time:.2f} segundos")
    return chunks
```

Código 13.4 – Trecho do código para gerar/armazenar os vetores.

```
def get_vectorstore(texto_chunks):
    """Função para gerar/armazenar os vetores."""
    start_time = time.time()
    embeddings = OpenAIEmbeddings(
        model="text-embedding-ada-002", chunk_size=1000, openai_api_key=api_key)
    vectorstore = FAISS.from_texts(texts=texto_chunks, embedding=embeddings)
    end_time = time.time()
    logging.info(f"Embeddings criados e armazenados no vectorstore.")
    logging.info(
        f"Tempo para gerar e armazenar embeddings: {end_time - start_time:.2f} segundos")
    return vectorstore
```

Código 13.5 – Trecho do código para cadeia de conversação.

```
def get_chat(vectorstore):
    """Função para criar uma cadeia de conversação."""
    llm = ChatOpenAI(model="gpt-4o", temperature=0.3, openai_api_key=api_key)
    memory = ConversationBufferMemory(
        memory_key='chat_history', return_messages=True)
    conversation = ConversationalRetrievalChain.from_llm(
        llm=llm,
        retriever=vectorstore.as_retriever(),
        memory=memory
    )
    logging.info("Conversational retrieval chain criada.")
    return conversation
```

Código 13.6 – Trecho do código para gerenciar entrada do usuário.

```python
def entrada_usuario(pergunta):
    """Função para gerenciar a entrada do usuário e medir o tempo de resposta da consulta."""
    if not st.session_state.conversation:
        st.warning("Carregue um documento antes de fazer perguntas.")
        logging.warning(
            "Usuário tentou fazer uma pergunta sem carregar um documento.")
        return

    logging.info(f"Pergunta do usuário: {pergunta}")
    start_time = time.time()

    try:
        response = st.session_state.conversation({'question': pergunta})
        end_time = time.time()
        logging.info(
            f"Tempo para gerar resposta à pergunta: {end_time - start_time:.2f} segundos"
        )

        st.session_state.chat_history = response['chat_history']
        for i, message in enumerate(st.session_state.chat_history):
            if i % 2 == 0:
                st.write("", user_template.replace(
                    "{{MSG}}", message.content), unsafe_allow_html=True)
            else:
                st.write("", bot_template.replace(
                    "{{MSG}}", message.content), unsafe_allow_html=True)

        response_texts = [
            msg.content for msg in st.session_state.chat_history if msg.content
        ]
        logging.info(f"Detetive MPRS: {response_texts}")
    except Exception as e:
        logging.error(f"Erro ao processar a pergunta do usuário: {e}")
        st.error(
            "Ocorreu um erro ao processar sua pergunta. Por favor, tente novamente.")
```

14. VALIDAÇÃO DOS RESULTADOS

Nesse capítulo apresenta-se o processo de validação dos resultados com o objetivo de garantir a usabilidade do sistema desenvolvido. A validação foca em avaliar a precisão das respostas, desempenho e usabilidade do sistema.

14.1. Objetivo da Validação

O principal objetivo da validação é assegurar que o sistema atenda aos requisitos de funcionalidade e qualidade, permitindo a identificação de irregularidades administrativas relatadas em documentos do TCE/RS. Os objetivos específicos incluem:

1. Verificar a Precisão das Respostas: Avaliar se o sistema responde corretamente às perguntas com uma resposta objetiva e precisa sobre os documentos processados.

2. Avaliar o Desempenho: Verificar a capacidade do sistema de processar documentos e responder em tempo hábil, especialmente em consultas repetidas e sob diferentes cargas.

3. Confirmar a Usabilidade da Interface: Assegurar que a interface do sistema é intuitiva e que o usuário consiga realizar upload de documentos, fazer consultas e entender as respostas sem dificuldades.

14.2. Métodos de Validação

A validação foi conduzida utilizando uma combinação de abordagens: avaliação automatizada e teste com usuários.

14.2.1. Avaliação Automatizada

Na avaliação automatizada, foram definidas métricas para verificar a precisão e desempenho do sistema.

1. Precisão: O sistema carrega documentos, divide-os em chunks e gera embeddings com OpenAIEmbeddings. A precisão das respostas foi avaliada comparando as respostas do sistema com respostas esperadas em um conjunto de testes. Esses testes simulam consultas do usuário para avaliar se as respostas estão de acordo com o contexto do documento.

2. Tempo de Resposta: Cada interação foi cronometrada, especialmente as etapas de carregamento de documentos, divisão em chunks e consultas. O tempo de resposta deve ser menor que 10 segundos para perguntas simples e documentos de tamanho moderado.

3. Métricas de Logs: O código registra logs detalhados para cada interação do usuário com a função entrada_usuario(). Isso permite rastrear a entrada da pergunta e a resposta gerada, além de acompanhar o histórico de interação. Esses dados são úteis para análise de desempenho e ajustes.

14.2.2. Testes com Usuários

O sistema foi disponibilizado para usuários realizarem consultas relacionadas ao conteúdo dos documentos do TCE/RS.

- Satisfação e Feedback: Os usuários avaliaram a relevância das respostas e a facilidade de uso da interface. Também foi pedido que classifiquem o entendimento das respostas e a eficiência geral do sistema.

15. CONCLUSÃO

Na Figura 15.1 observa-se a análise do teste de precisão realizado. O sistema apresentou um desempenho com uma taxa de acerto de 100%. A validação consistiu em comparar as respostas geradas pelo programa com as respostas esperadas, fornecendo um diagnóstico de precisão em tarefas de classificação binária (respostas "SIM" ou "NÃO") para a identificação de irregularidades administrativas.

Durante o teste de precisão, 13 documentos distintos foram processados. Em cada caso, a resposta gerada pelo sistema coincidia com a resposta esperada, o que demonstra a eficácia do modelo em entender e classificar corretamente a presença ou ausência de irregularidades administrativas nos documentos.

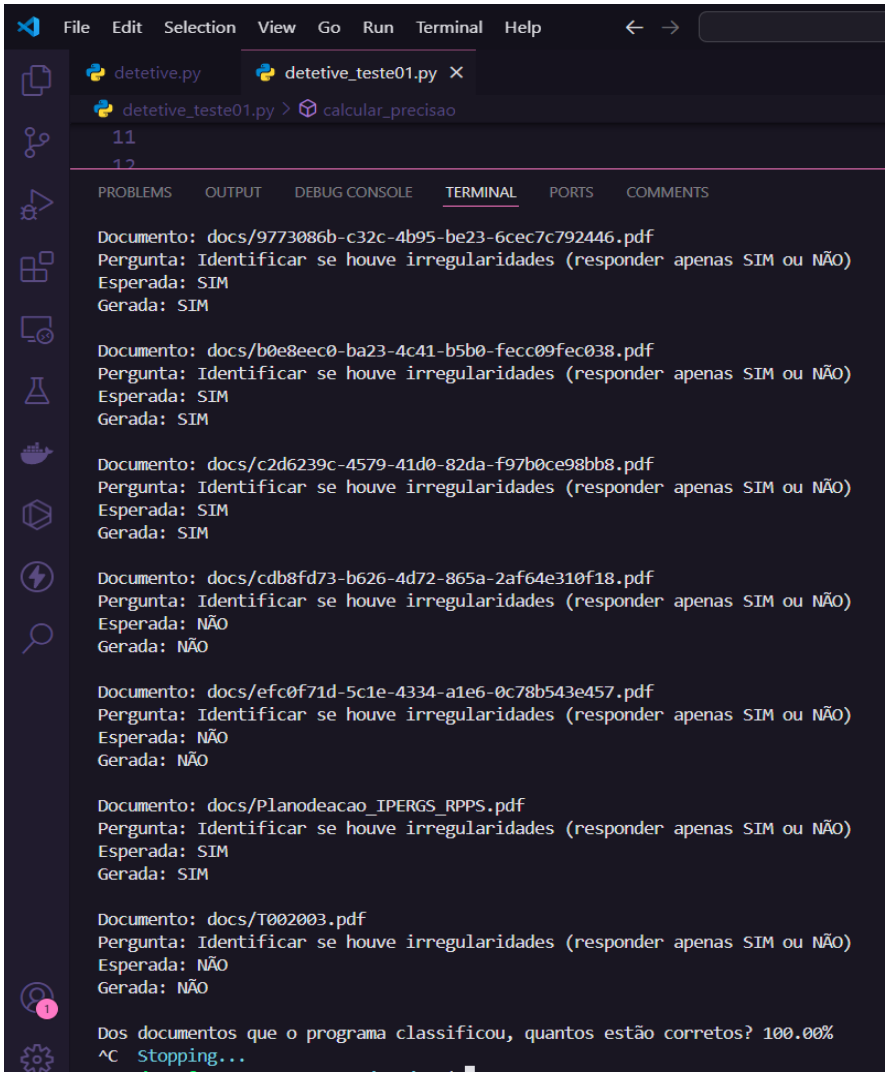

Figura 15.1 – Teste de precisão (Fonte: Autor).

Na Figura 15.2 a análise dos registros de log do sistema revela um desempenho consistente nas diferentes etapas do processamento do documento. As métricas capturadas indicam que o sistema é capaz de realizar tarefas como extração de texto, criação de embeddings e geração de respostas dentro de tempos razoáveis, assegurando que os usuários recebam a resposta de maneira oportuna.

No entanto, mesmo com um desempenho satisfatório, há sempre margem para melhorias. A análise de tempos de execução fornece insights sobre onde o sistema pode ser otimizado, como na criação e armazenamento de embeddings ou na geração de respostas complexas. Esses dados indicam onde direcionar esforços

de otimização para assegurar que o sistema continue a atender de forma eficiente as demandas dos usuários mesmo em cenários de alta carga ou documentos complexos.

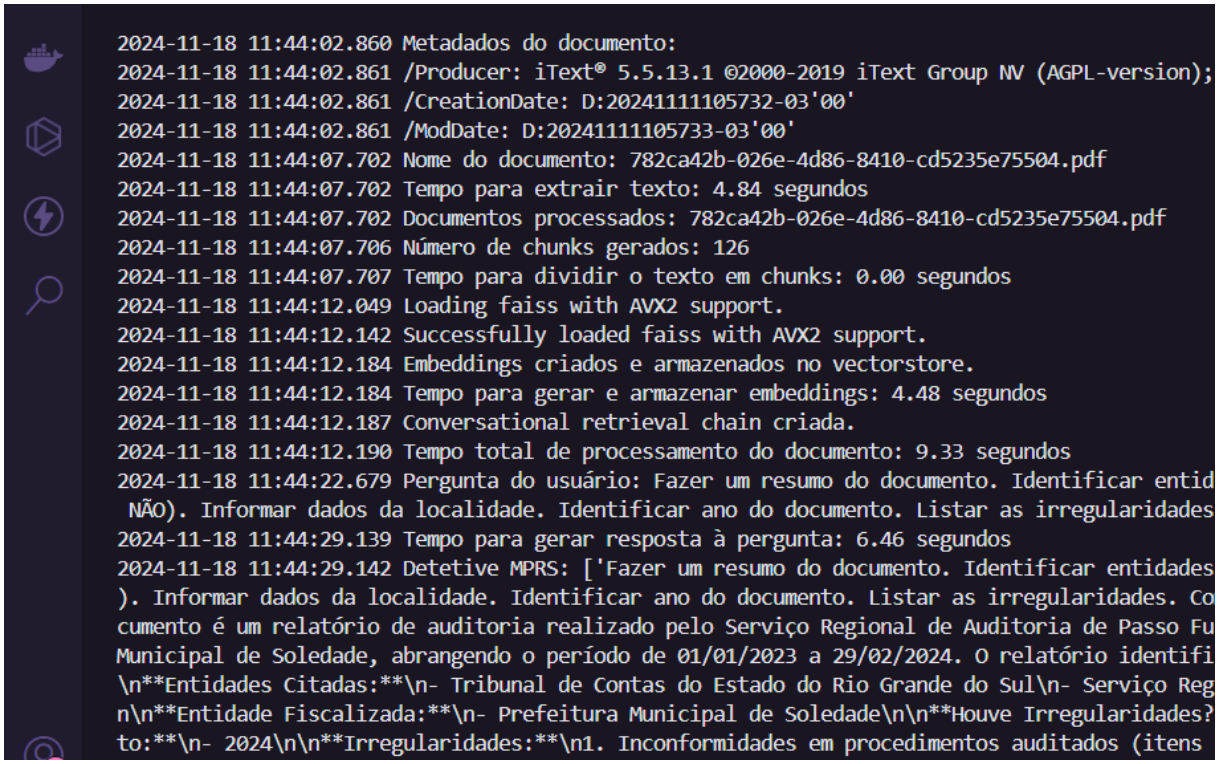

Figura 15.2 – Logs e tempo de resposta (Fonte: Autor).

A Figura 5.3 apresenta a interface para pesquisa no sistema desenvolvido. Após o carregamento do documento, o usuário pode fazer perguntas no campo de texto, como no exemplo mostrado: "Identificar se houve irregularidades". O sistema processa o conteúdo do documento e responde de forma textual, com base em análise automatizada, informando se existem irregularidades. Para isso são utilizadas técnicas de Processamento de Linguagem Natural (PLN) para interpretar a pergunta e consultar informações no documento. Nesse caso, o sistema detectou a existência de irregularidades nas admissões decorrentes de concursos públicos.

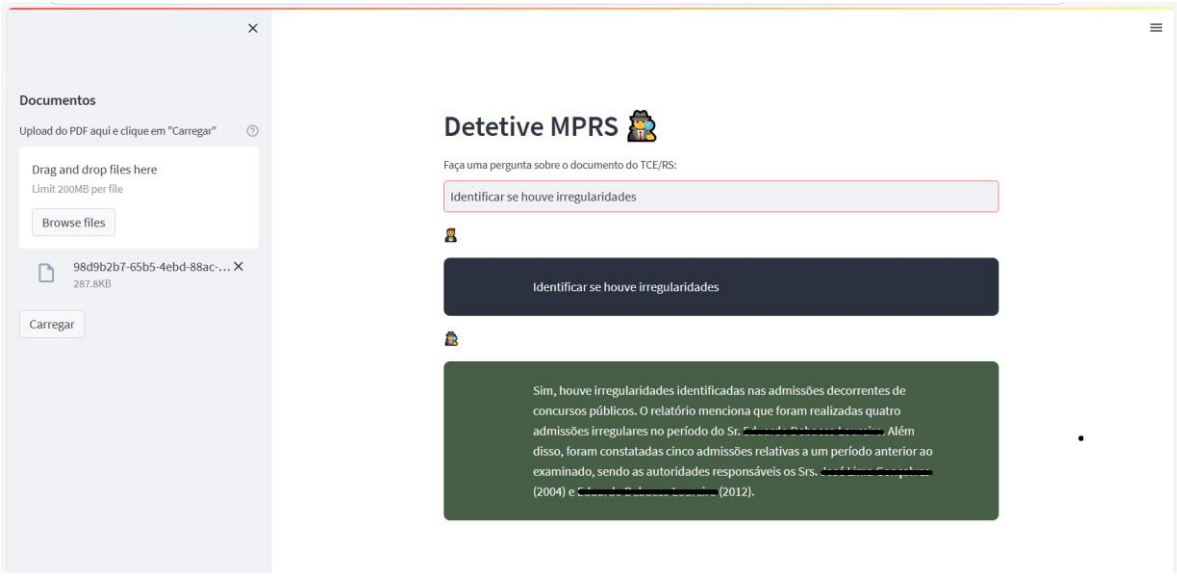

Figura 15.3 – Interface de pesquisa (Fonte: Autor).

Nos testes com usuários, os resultados do questionário de satisfação aplicado com cinco usuários indicam que o sistema apresenta pontos fortes, como a facilidade de uso e a qualidade das respostas, mas também áreas que necessitam de melhorias.

Os usuários avaliaram positivamente a intuitividade e a organização da interface, embora uma parcela tenha apontado dificuldades, destacando a necessidade de melhora no design e de mais recursos de suporte. A rapidez de resposta do sistema também foi bem recebida. A relevância das respostas geradas foi considerada satisfatória mas a inconsistência percebida em alguns momentos indica que o sistema pode ser aprimorado para oferecer melhores resultados.

A satisfação geral com o sistema é positiva e o feedback sugere que melhorias específicas podem aumentar a aceitação da aplicação. Por fim, destaca-se a importância de continuar investindo em aprimoramentos centrados no usuário para garantir que o sistema se torne mais robusto e com uma experiência de uso que atenda a diferentes expectativas.

Observa-se que a principal contribuição dessa ferramenta está na demonstração de sua viabilidade em cenários reais, agilizando a análise de

documentos por meio de inteligência artificial. No entanto, existem limitações, como a dependência de dados de alta qualidade e específicos, além da necessidade de hardware robusto para garantir a escalabilidade da aplicação. O risco de inconsistências também se apresenta em cenários onde os dados são muito variados

Como proposição de um trabalho futuro, na próxima versão do sistema que está em desenvolvimento, busca-se implementar um consumo dos dados a partir de uma API do TCE/RS para posterior indexação no Apache Solr, garantindo consultas mais rápidas e eficientes. Além disso, será implementado um front-end interativo com o mapa do Estado do Rio Grande do Sul, proporcionando uma interface intuitiva para visualização e análise das informações processadas.

REFERÊNCIAS

[1] JUSBRASIL. *Artigos. A revolução da Inteligência Artificial na área jurídica.* Disponível em: https://www.jusbrasil.com.br/artigos/a-revolucao-da-inteligencia-artificial-na-area-juridica/1793209367. Acesso em: 06 de março de 2024.

[2] TOM B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language models are few-shot learners.* Publicado em Advances in Neural Information Processing Systems. 2020. Disponível em: https://arxiv.org/abs/2005.14165. Acesso em: 10 de março de 2024.

[3] AMAZON WEB SERVICES. *Hub de conceitos de computação em nuvem*. *O que é RAG?* Disponível em: https://aws.amazon.com/pt/what-is/retrieval-augmented-generation/#:~:text=aumentada%20de%20recupera%C3%A7%C3%A3o%3F-,O%20que%20%C3%A9%20a%20gera%C3%A7%C3%A3o%20aumentada%20de%20recupera%C3%A7%C3%A3o%3F,antes%20de%20gerar%20uma%20resposta. Acesso em: 10 de março de 2024.

[4] CASELIi, H.M.; Nunes, M.G.V. (org.) *Processamento de Linguagem Natural: Conceitos, Técnicas e aplicações em português*. 2 ed. BPLN, 2024. Disponível em: https://brasileiraspln.com/livro-pln/2a-edicao. Acesso em: 20 março de 2024.

[5] KOWSARI, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E., & Brown, D. E, 2019. *Text Classification Algorithms: A Survey*. Disponível em:https://doi.org/10.3390/info10040150. Acesso em: 17 de março de 2024.

[6] SIMEONE, O. *A Brief Introduction to Machine Learning for Engineers*. Department of Informatics King's College London, 2018. Disponível em: http://arxiv.org/abs/1709.02840v3. Acesso em 10 de março de 2024.

[7] MA, S.; CHEN, C.; CHU, Q.; MAO, J. *Leveraging Large Language Models for Relevance Judgments in Legal Case Retrieval*. New York, NY, USA: ACM, 2018. Disponível em: https://arxiv.org/abs/2403.18405. Acesso em: 15 de março de 2024.

[8] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Machine Learning Basics. In: Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. Disponível em: https://www.deeplearningbook.org/contents/ml.html. Acesso em: 15 de abril de 2024.

[9] DEHGHANI, M.; ZAMANI, H.; SEVERYN, A.; METZLER, D.; YATES, A. *The Retrieval Augmented Generation Ebook.* Disponível em: https://mallahyari.github.io/rag-ebook/. Acesso em: 20 de março de 2024.

[10] BARNETT, S.; KURNIAWAN, S.; THUDUMU, S.; BRANNELLY, Z.; ABDELRAZEK, M. *Seven Failure Points When Engineering a Retrieval Augmented Generation System*. Publicado em: Proceedings of the 3rd International Conference on AI Engineering — Software Engineering for AI (CAIN 2024). New York, NY, USA: ACM, 2024. Disponível em: https://arxiv.org/abs/2401.05856. Acesso em: 20 de março de 2024.

[11] CHENXI, D. *How to Build an AI Tutor that Can Adapt to Any Course and Provide Accurate Answers Using Large Language Model and Retrieval-Augmented Generation*. Hong Kong: Department of Mathematics and Information Technology, The Education University of Hong Kong. Disponível em: https://arxiv.org/abs/2311.17696. Acesso em: 24 de março de 2024.

[12] KAUARK, Fabiana; Manhães, Fernanda Castro; Medeiros, Carlos Henrique. *Metodologia da pesquisa: um guia prático.* Itabuna:Via Litterarum, 2010. Disponível: www.pgcl.uenf.br/arquivos/livrodemetodologiadapesquisa2010_011120181549.pdf. Acesso em: 03 de abril de 2024.

[13] LANGCHAIN. *RAG Tutorial.* Disponível em: https://python.langchain.com/v0.2/docs/tutorials/rag/. Acesso em: 02 julho 2024.

[14] DIAS, Nauane Karoline Brazolino; ARAÚJO, Marco Antônio Pereira. *Boas Práticas na Engenharia de Requisitos*. Engenharia de Software Magazine, Juiz de Fora, Edição 27, p. 14-21.

[15] SOMMERVILLE, Ian. *Software Engineering*. 9. ed. Upper Saddle River: Addison-Wesley, 2011. p. 126.

[16] BEZERRA, Eduardo. *Princípios de Análise e Projeto de Sistemas com UML*. 3. ed. Rio de Janeiro: Elsevier, 2015. p. 75.

[17] STREAMLIT. *API Reference*. Disponível em: https://docs.streamlit.io/develop/api-reference. Acesso em: 18 set. 2024.

[18] PYPDF2. *Welcome to PyPDF2*. Disponível em: https://pypdf2.readthedocs.io/en/3.x/. Acesso em: 18 set. 2024.

[19] FAISS – META AI: *FAISS*. Disponível em: https://ai.meta.com/tools/faiss/. Acesso em: 19 set. 2024.

[20] LANGCHAIN. *ConversationalRetrievalChain.* Disponível em: https://api.python.langchain.com/en/latest/chains/langchain.chains.conversational_retrieval.base.ConversationalRetrievalChain.html#langchain.chains.conversational_retrieval.base.ConversationalRetrievalChain.from_llm. Acesso em: 19 set 2024.

[21] PYTHON SOFTWARE FOUNDATION. *Logging.* Disponível em: https://docs.python.org/pt-br/3/howto/logging.html. Acesso em: 20 set. 2024.

[22] OPENAI. *New and improved embedding model*. Disponível em: https://openai.com/index/new-and-improved-embedding-model/. Acesso em: 20 set. 2024.

[23] OPENAI. *Hello GPT-4o*. Disponível em: https://openai.com/index/hello-gpt-4o/. Acesso em: 20 set. 2024.

[24] MONGODB. *How to Choose the Right Chunking Strategy for Your LLM Application.* Disponível em: https://www.mongodb.com/developer/products/atlas/choosing-chunking-strategy-rag/#chunking-strategies. Acesso em: 20 set. 2024.

[25] MICROSOFT. *How to chunk documents for vector search*. Disponível em: https://learn.microsoft.com/en-us/azure/search/vector-search-how-to-chunk-documents. Acesso em: 20 set. 2024.

[26] TRIBUNAL DE CONTAS DO ESTADO DO RIO GRANDE DO SUL. Portal do Cidadão. Disponível em: https://tcers.tc.br/cidadao/. Acesso em: 10 ago. 2024.

APÊNDICE A - Questionário de Satisfação do Usuário

Instruções: Avalie os seguintes itens sobre sua experiência com o sistema. Use a escala de 1 a 5, onde:

- 1 significa "Muito insatisfeito" ou "Discordo totalmente".
- 5 significa "Muito satisfeito" ou "Concordo totalmente".

Seção 1: Usabilidade do Sistema

1. Facilidade de uso: O sistema é intuitivo e fácil de usar.
   - 1 - Muito insatisfeito: 0 respostas
   - 2 – Insatisfeito: 0 respostas
   - 3 – Neutro: 1 resposta
   - 4 – Satisfeito: 2 respostas
   - 5 - Muito satisfeito: 2 respostas

2. Interface visual: A interface é agradável e organizada.
   - 1 - Discordo totalmente: 0 respostas
   - 2 – Discordo: 0 respostas
   - 3 – Neutro: 1 resposta
   - 4 – Concordo: 2 respostas
   - 5 - Concordo totalmente: 2 respostas

3. Rapidez de Respostas: O sistema responde às perguntas em tempo razoável.
   - 1 - Muito insatisfeito: 0 respostas
   - 2 – Insatisfeito: 0 respostas
   - 3 – Neutro: 0 respostas
   - 4 – Satisfeito: 2
   - 5 - Muito satisfeito: 3

Seção 2: Qualidade das Respostas

4. Relevância das Respostas: As respostas são úteis e pertinentes às perguntas feitas.

- 1 - Muito insatisfeito: 0 respostas
- 2 – Insatisfeito: 0 respostas
- 3 – Neutro: 0 respostas
- 4 – Satisfeito: 2 respostas
- 5 - Muito satisfeito: 3 respostas

Seção 3: Experiência Geral

5. Satisfação Geral: Como você avalia sua satisfação geral com o sistema?

- 1 - Muito insatisfeito: 0 respostas
- 2 – Insatisfeito: 0 respostas
- 3 – Neutro: 0 respostas
- 4 – Satisfeito: 2 respostas
- 5 - Muito satisfeito: 3 respostas

Seção 4: Feedback Adicional

5. Sugestões de Melhoria: O que poderia ser melhorado?

- Opção de clicar nas perguntas.
- Limpar local de pesquisa automático.
- Melhor ordenamento das respostas.
- Opção para download do histórico da conversa.