Insper

Trabalho de Conclusão de Curso

Relatório do Projeto Final - Capstone

Fernanda de Oliveira Pereira
Gabriel Mendonça de Mello Hermida
Rodrigo Paoliello de Medeiros

Guardrails em IA Generativa

São Paulo
2025

Fernanda de Oliveira Pereira
Gabriel Mendonça de Mello Hermida
Rodrigo Paoliello de Medeiros

Guardrails em IA Generativa

Relatório apresentado ao curso de Graduação em Engenharia de Computação, como requisito parcial para a obtenção do título de Bacharel no respectivo curso.

Orientador: Prof. Dr. Fabricio Jailson Barth

Coordenador Capstone: Prof. Dr. Luciano P. Soares

São Paulo
2025

Fernanda de Oliveira Pereira
Gabriel Mendonça de Mello Hermida
Rodrigo Paoliello de Medeiros

Guardrails em IA Generativa

Relatório apresentado ao curso de Graduação em Engenharia de Computação, como requisito parcial para a obtenção do título de Bacharel no respectivo curso.

Orientador: Prof. Dr. Fabricio Jailson Barth

Banca Examinadora

Prof. Dr. Fabricio Jailson Barth
Insper

Prof. Dr. Maciel Calebe Vidal
Insper

Prof. MSc. Raul Ikeda Gomes da Silva
Insper

Resumo

A inteligência artificial (IA) generativa, quando aplicada no desenvolvimento de assistentes virtuais, pode, em muitos casos, retornar respostas inadequadas ou incorretas, confundindo usuários e gerando desinformação. Atualmente, são desenvolvidas *guardrails* que analisam tanto as entradas fornecidas pelos usuários quanto as saídas geradas pelos modelos, garantindo que as respostas devolvidas sejam adequadas. Nesse contexto, a solução deste projeto consistiu na implementação de diferentes tipos de *guardrails* (escopo, competidores, toxicidade, alucinação, dados sensíveis e técnicas de *jailbreak*), na definição de métricas específicas para avaliar seu funcionamento e na construção de uma arquitetura suficientemente modular para ser adaptável a diversos contextos. As *guardrails* implementadas alcançaram resultados satisfatórios conforme as métricas estabelecidas, indicando viabilidade técnica para utilização inicial pela empresa parceira em consultorias de assistência virtual com IA. Apesar disso, o projeto identificou oportunidades de melhoria, especialmente relacionadas à ampliação das bases de treinamento e avaliação, à adoção de mecanismos adaptativos e ao aprimoramento da mitigação de alucinações, visando ampliar sua eficácia em aplicações práticas futuras.

Palavras-chave: Inteligência Artificial, Assistentes Virtuais, *Large Language Models* (LLM), *Guardrails*.

Sumário

1 Introdução **4**
1.1 Proposta do projeto 4
1.2 Origem do Desafio 4
1.3 Escopo do Projeto 5
1.4 Questões Éticas e Profissionais 6
1.5 Normas Técnicas 6
1.6 Licenciamento de Ferramentas e Bases de Dados 6
1.7 Riscos Envolvidos 7
1.8 Recursos Necessários 8
1.9 Cronograma 9
1.9.1 Semanas 1 a 3 9
1.9.2 Semanas 4 a 7 9
1.9.3 Semanas 8 a 12 9
1.9.4 Semanas 13 a 16 9
1.9.5 Semanas 17 a 18 9
1.10 Gestão do Projeto 10
1.11 Mapeamento dos Stakeholders 10
1.12 Pesquisa de Trabalhos Relacionados 11

2 Metodologia **13**
2.1 Descrição das Guardrails Utilizadas 13
2.1.1 Toxicidade 13
2.1.2 Jailbreak 13
2.1.3 Dados Sensíveis 14
2.1.4 Escopo 14
2.1.5 Alucinação 14
2.1.6 Competidores 14
2.2 Utilização e Construção dos Conjuntos de Dados 15
2.2.1 Critério de Utilização de Bases Reais e Sintéticas 15

2.2.2 Construção das Bases Sintéticas 15
2.2.3 Uso dos Dados nas Etapas do Desenvolvimento 16
2.3 Avaliação 16

3 Desenvolvimento **17**
3.1 Arquitetura e Implementação das *Guardrails* 17
3.1.1 Toxicidade 18
3.1.2 Jailbreak 19
3.1.3 Dados Sensíveis 22
3.1.4 Escopo e Alucinação 22
3.1.5 Competidores 24
3.2 Organização do Repositório 25
3.2.1 Visão Geral 25
3.2.2 Estrutura do Diretório 25
3.2.3 Arquitetura de Execução via *Pipeline* 25
3.2.4 Exemplo de Uso 26

4 Resultados **27**
4.1 Demonstração Interativa 27
4.2 Avaliação por Tipo de Risco Moderado 27
4.2.1 Toxicidade 27
4.2.2 Jailbreak 29
4.2.3 Dados Sensíveis 30
4.2.4 Escopo e Alucinação 30
4.2.5 Competidores 34

5 Conclusão **35**

Referências **37**

1 Introdução

1.1 Proposta do projeto

O avanço das inteligências artificiais generativas no desenvolvimento de assistentes virtuais trouxe benefícios significativos para diversas áreas, mas também levantou desafios relacionados à segurança e ao controle das respostas geradas. A ausência de mecanismos eficazes de filtragem pode resultar na produção de informações imprecisas, inadequadas ou mesmo prejudiciais, impactando negativamente na experiência do usuário e na credibilidade das organizações.

Este projeto teve como objetivo desenvolver e implementar *guardrails* capazes de mitigar significativamente esses riscos, promovendo maior segurança e confiabilidade no uso de modelos de IA generativa. As soluções desenvolvidas viabilizaram a validação de entradas e o monitoramento das saídas geradas pelos assistentes virtuais, contribuindo para que as respostas permanecessem alinhadas às expectativas e às políticas internas das empresas. A abordagem adotada foi modular, permitindo adaptações a diferentes *Large Language Models* (LLMs)[1] e personalizações específicas conforme as necessidades de cada organização.

1.2 Origem do Desafio

Este projeto surgiu da crescente adoção de IA generativa em soluções empresariais e dos desafios relacionados à segurança, confiabilidade e conformidade regulatória. A Dell Technologies, parceira do projeto, é uma multinacional de tecnologia que oferece soluções em infraestrutura de TI, nuvem, segurança cibernética e inteligência artificial.

À época do projeto, a Dell prestava consultoria na personalização de IAs generativas para empresas que demandavam assistentes virtuais adaptados a contextos específicos. O serviço incluía a configuração e ajuste dos modelos, garantindo operação dentro de escopos definidos, com segurança, eficiência e aderência às normas do setor.

Um dos principais desafios enfrentados era promover respostas mais precisas, seguras e alinhadas às diretrizes éticas e regulatórias. Para isso, foi fundamental a implementação de *guardrails*

[1] LLM (*Large Language Model*) é um tipo de modelo de aprendizado profundo treinado em grandes volumes de texto para realizar tarefas de linguagem natural. Ver: <https://sebastianraschka.com/blog/2023/LLM-overview.html>

— mecanismos que controlam e refinam as respostas, mitigando vieses, reduzindo o risco de disseminação de informações incorretas ou sensíveis e contribuindo para a aderência às regulamentações.

O projeto refletiu o compromisso da Dell com o desenvolvimento de tecnologias responsáveis e confiáveis. Ao fortalecer a segurança e a conformidade de suas soluções em IA generativa, a empresa buscou aprimorar a experiência dos clientes e consolidar sua posição como referência no setor.

Este trabalho foi desenvolvido no contexto do programa Capstone do Insper, que conecta estudantes a desafios reais do mercado. Dessa forma, a equipe atuou, ao longo do semestre, em parceria com a Dell no desenvolvimento e implementação das *guardrails* para IA generativa.

1.3 Escopo do Projeto

O escopo deste projeto compreendeu o desenvolvimento e a implementação de *guardrails* para assistentes virtuais baseados em IA generativa, com o objetivo de promover maior segurança, confiabilidade e personalização nas respostas geradas. As *guardrails* atuaram na validação de entradas e no monitoramento de saídas, prevenindo a geração de conteúdos inadequados e assegurando que os assistentes operassem dentro dos limites estabelecidos pelas empresas clientes. Para isso, foram adotadas estratégias de filtragem que verificavam se as consultas dos usuários estavam alinhadas ao escopo e contexto definidos pelas organizações, bloqueando respostas fora do domínio desejado. Além disso, foram implementados filtros específicos para bloquear menções diretas ou indiretas a empresas concorrentes, em conformidade com políticas comerciais e interesses estratégicos.

O projeto também integrou modelos especializados na detecção de toxicidade, com a finalidade de identificar e reduzir a geração de conteúdos ofensivos, discriminatórios ou inadequados. Essa etapa buscou preservar a reputação das empresas que utilizam assistentes virtuais com IA generativa.

Outro aspecto tratado foi a mitigação de alucinações — respostas incorretas ou inventadas pelas LLMs. Para isso, foram empregadas técnicas como validação cruzada de informações e comparação com bases documentais, visando reduzir a probabilidade de desinformação.

A proteção de dados sensíveis também foi abordada. O projeto desenvolveu mecanismos de análise e anonimização de *prompts* que continham dados pessoais ou confidenciais, removendo ou mascarando informações que pudessem identificar indivíduos ou organizações. Essas técnicas incluíram análise semântica e reconhecimento de padrões associados a ataques previamente documentados.

O projeto não envolveu o desenvolvimento de novos LLMs, nem a modificação de suas arquiteturas. O foco permaneceu na construção e avaliação das *guardrails*, sem contemplar o desenvolvimento de interfaces de usuário ou aplicações externas. As soluções foram testadas nos modelos *OpenAI*[2], *DeepSeek*[3] e *LLaMA*[4], demonstrando sua aplicabilidade em diferentes plataformas. Ajustes

[2]<https://openai.com>
[3]<https://deepseek.com>
[4]<https://ai.meta.com/llama>

adicionais podem ser necessários para extensões futuras, mas a delimitação estabelecida permitiu que o projeto fosse conduzido de forma objetiva dentro do cronograma e dos recursos disponíveis.

1.4 Questões Éticas e Profissionais

A implementação das *guardrails* em IAs generativas neste projeto envolveu desafios éticos e técnicos, especialmente no equilíbrio entre controle e liberdade de expressão. Conforme discutido por (AYYAMPERUMAL; GE, 2024), tais mecanismos devem restringir respostas inadequadas sem prejudicar a usabilidade dos modelos. A governança das decisões algorítmicas também foi considerada, com atenção à mitigação de possíveis vieses introduzidos ou amplificados durante o processo.

Aspectos como transparência e auditabilidade foram tratados como fundamentais. A ausência de critérios claros de filtragem pode gerar desconfiança e produzir efeitos colaterais indesejados na moderação de conteúdo (AYYAMPERUMAL; GE, 2024). Para enfrentar esse risco, o projeto incluiu documentação explícita das regras aplicadas por cada *guardrail*, facilitando sua compreensão e avaliação posterior.

Embora os mecanismos desenvolvidos tenham auxiliado na correção de respostas falsas, tóxicas ou fora de escopo, reconheceu-se o risco de introdução de novos vieses. Por essa razão, foram conduzidas análises sistemáticas para verificar se as restrições aplicadas permaneciam justas, proporcionais e alinhadas aos objetivos de segurança.

Com isso, o projeto buscou desenvolver *guardrails* eficazes e transparentes, promovendo maior controle e confiabilidade sem comprometer a autonomia dos modelos nem a experiência do usuário final.

1.5 Normas Técnicas

Na área de desenvolvimento de *software* e inteligência artificial, especialmente em pesquisa aplicada e construção de modelos baseados em *machine learning*, não há aplicação prática de normas técnicas formais, como ocorre em setores de engenharia tradicional. Portanto, o desenvolvimento segue boas práticas de engenharia de *software*, padrões acadêmicos e diretrizes de ferramentas e *frameworks* utilizados, mas não uma norma técnica específica.

1.6 Licenciamento de Ferramentas e Bases de Dados

Este projeto faz uso de diversos *frameworks* e bases de dados de terceiros, cada um regido por sua respectiva licença de uso. O *NeMo*, desenvolvido pela NVIDIA (NVIDIA, 2025), e o *Ten-*

sorFlow[5], são licenciados sob a Apache License 2.0[6], que permite uso, modificação e redistribuição, inclusive para fins comerciais, desde que mantidos os avisos de copyright e a licença original. As ferramentas *Microsoft Presidio* (MICROSOFT, 2024) e o modelo *BERTimbau* (SOUZA, 2020), da *NeuralMind*[7], estão licenciados sob a MIT License[8], igualmente permissiva e amplamente utilizada em ambientes acadêmicos e comerciais.

O serviço *Unitary AI*[9] apresenta termos próprios de uso, com restrições, sendo necessária a consulta à política de uso publicada em seu site oficial[10] antes de qualquer reutilização. Neste projeto, o serviço foi utilizado apenas para fins de comparação e estudo, sem integração direta na versão final da implementação.

Em relação às bases de dados, foi utilizado o *ToLD-Br (Toxic Language Dataset – BR)*[11], disponível na plataforma *Hugging Face*[12], e licenciado sob a *Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)*[13]. Essa licença permite uso e modificação, inclusive para fins comerciais, desde que haja atribuição adequada ao autor e que qualquer obra derivada seja distribuída sob a mesma licença. Também foi utilizada a base *Jailbreak Classification*[14], disponibilizada pelo repositório de Jack Hao, sob a licença Apache 2.0[15], com permissões amplas e exigência de atribuição semelhante.

Todos os elementos utilizados no projeto foram devidamente citados, com observância às permissões e restrições impostas por seus respectivos autores e licenças. Recomenda-se que qualquer redistribuição ou modificação deste projeto preserve essas atribuições e respeite as condições de compartilhamento, especialmente no caso da base *ToLD-Br*, que exige a manutenção da licença original em obras derivadas.

1.7 Riscos Envolvidos

Durante o desenvolvimento do projeto, foram identificados riscos técnicos com potencial de impacto tanto no cronograma quanto no desempenho das soluções. Um dos principais riscos foi a possibilidade de evolução do cenário regulatório em IA, que poderia demandar ajustes na arquitetura das *guardrails*. Além disso, a dependência de ferramentas externas, como o *framework* NeMo da NVIDIA, implicou riscos relacionados à compatibilidade, estabilidade e eventual descontinuidade de

[5]<https://www.tensorflow.org>
[6]<https://www.apache.org/licenses/LICENSE-2.0>
[7]<https://neuralmind.ai/>
[8]<https://opensource.org/licenses/MIT>
[9]<https://unitary.ai>
[10]<https://www.guardrailsai.com/>
[11]<https://aclanthology.org/2020.aacl-main.91/>
[12]<https://huggingface.co>
[13]<https://creativecommons.org/licenses/by-sa/4.0/>
[14]<https://github.com/jack-hjk/jailbreak_classification>
[15]<https://www.apache.org/licenses/LICENSE-2.0>

suporte.

A integração das *guardrails* com diferentes modelos de linguagem — como OpenAI[16], DeepSeek[17] e LLaMA[18] — representou outro desafio relevante. Cada LLM apresentou particularidades técnicas, como formatos de resposta, limites de requisição e variações nas APIs, o que exigiu adaptações específicas e introduziu riscos de incompatibilidade e perda de desempenho. Adicionalmente, houve incertezas quanto à disponibilidade das chaves de API, com a execução local sendo considerada apenas em casos com infraestrutura computacional adequada.

Também foi identificado um risco relacionado à definição de escopo via *prompts*, pois estruturas mal definidas poderiam induzir o modelo a gerar respostas fora do domínio esperado, comprometendo a eficácia das soluções.

Por fim, as limitações associadas ao idioma também representaram um ponto de atenção. Como grande parte dos modelos e ferramentas disponíveis foram treinados majoritariamente em inglês, observou-se desempenho inferior em português. A necessidade de traduzir bases de dados para testes introduziu ainda riscos adicionais, como erros de tradução que poderiam afetar negativamente a qualidade das validações realizadas.

1.8 Recursos Necessários

O projeto envolveu o uso de *APIs* de modelos de linguagem, como *OpenAI*[19], *DeepSeek*[20] e *LLaMA*[21], além da execução local de alguns desses modelos para a implantação e teste das *guardrails*. Embora o ideal fosse contar com uma GPU de maior capacidade, os experimentos foram realizados com sucesso em uma máquina pessoal equipada com uma GPU NVIDIA GeForce RTX 3060 com 6 GB de memória, processador Intel Core i7-11800H (8 núcleos físicos e 16 threads) e aproximadamente 60 GB de armazenamento dedicado ao projeto.

O ambiente também foi utilizado para rodar bibliotecas de alto custo computacional, como *NeMo* e modelos baseados em *BERT* (DEVLIN et al., 2019), exigindo controle no uso de memória e armazenamento. Não foi necessário recorrer a instâncias externas, embora essa possibilidade tenha sido considerada inicialmente.

Também foi necessária uma dedicação semanal mínima de 20 horas por parte do grupo para pesquisa e desenvolvimento. A colaboração contínua com a empresa parceira foi essencial para a definição de prioridades e objetivos ao longo das etapas do projeto, garantindo o alinhamento com as necessidades reais da aplicação.

[16] <https://openai.com>
[17] <https://deepseek.com>
[18] <https://ai.meta.com/llama>
[19] <https://openai.com>
[20] <https://deepseek.com>
[21] <https://ai.meta.com/llama>

1.9 Cronograma

O projeto foi desenvolvido ao longo do semestre, seguindo um cronograma dividido em cinco fases principais.

1.9.1 Semanas 1 a 3

Foram realizadas reuniões com a empresa parceira para compreender os desafios do projeto. Nessa etapa, houve uma definição clara dos objetivos e restrições, além da realização de uma pesquisa inicial sobre técnicas e ferramentas adequadas para a execução do trabalho.

1.9.2 Semanas 4 a 7

Foi desenvolvido a versão inicial da arquitetura para hospedar todas as *guardrails* que seriam construídas em seguida, além disso ocorreu o estudo e implementação de dois tipos de *guardrails*, *jailbreak* e toxicidade, com métricas de teste e resultados apenas para a primeira. Em paralelo, foram testados os usos de dois modelos de LLM, GPT-4o (*OpenAI*[22]) e *LLaMa 3.2*, o primeiro com requisições e o segundo localmente.

1.9.3 Semanas 8 a 12

Esta fase foi dedicada a apresentação intermediária à banca, término dos testes e resultados da *guadrail* de toxicidade e implementação de mais três tipos de *guardrails*: dados sensíveis e escopo geral de um determinado tema. Além disso, marcou início à organização estrutural final do projeto.

1.9.4 Semanas 13 a 16

Nesse período, foram estudadas, implementadas e testadas *guadrails* de escopo mais focado, alucinação e competidores. Paralelamente, é finalizada a organização estrutural do projeto e é implementado o protótipo inicial de visualização do funcionamento de um assistente virtual com as *guadrails* implementadas.

1.9.5 Semanas 17 a 18

Nesta fase final, as otimizações, documentação e apresentações finais para as bancas de avaliação da universidade foram concluídas. De forma que todo *feedback* recebido foi traduzido em ajustes finais ao projeto e sua documentação antes da conclusão oficial do trabalho.

[22] <https://openai.com>

1.10 Gestão do Projeto

O projeto foi conduzido com base na metodologia ágil, adotando o *framework* Scrum. As atividades foram organizadas em sprints, acompanhadas por reuniões semanais com o grupo, a empresa parceira e o orientador, com o objetivo de manter o alinhamento entre todas as partes envolvidas.

A definição de requisitos e prioridades foi orientada por reuniões com a empresa parceira, que permitiram identificar necessidades e restrições relevantes. Também foram realizadas análises comparativas de soluções existentes para embasar decisões técnicas e validar a abordagem escolhida.

O gerenciamento das tarefas foi feito por meio da plataforma Trello[23], com a categorização das atividades em colunas que indicavam seu status: "A Fazer", "Em Progresso"e "Concluído". Essa estrutura permitiu acompanhar o progresso do projeto de forma clara e garantir o cumprimento dos prazos estabelecidos.

1.11 Mapeamento dos Stakeholders

A tabela 1.1, a seguir, mapeia os principais envolvidos no projeto:

Stakeholder	**Posição**	**Papel no Projeto**	**Expectativas**
Rodrigo Medeiros Gabriel Hermida Fernanda Oliveira	Grupo do projeto	Desenvolver o projeto e implementar as soluções	Apresentar uma solução completa e funcional para o problema da empresa
Fabrício Barth	Professor orientador responsável	Orientar tecnicamente o grupo e validar as entregas	Garantir a qualidade acadêmica e técnica do projeto
Lais da Silva Pedro Henrique Pereira	Mentores da empresa	Acompanhar o projeto, oferecendo suporte semanal e auxiliando na resolução de desafios	Garantir que o grupo tenha direcionamento contínuo e que a solução esteja alinhada com as expectativas da empresa

Tabela 1.1: Tabela de Mapeamento dos Stakeholders

[23]<https://trello.com>

1.12 Pesquisa de Trabalhos Relacionados

O desenvolvimento deste projeto foi embasado por uma pesquisa abrangente sobre trabalhos acadêmicos e ferramentas relacionadas às principais *guadrails* para IA generativa, tais como alucinação, toxicidade, violações de escopo e exposição de dados sensíveis. O levantamento das referências permitiu identificar desafios atuais, compreender abordagens consagradas e definir estratégias eficazes para promover segurança e confiabilidade na aplicação dos modelos.

Huang et al. (HUANG et al., 2024) apresentam uma análise detalhada do fenômeno das alucinações em modelos de linguagem. Os autores propõem uma taxonomia que diferencia alucinações factuais (informações incorretas ou não verificáveis) e não factuais (incoerências lógicas ou linguísticas), destacando desafios como a dificuldade em detectar automaticamente essas ocorrências, a ausência de *benchmarks* padronizados e limitações dos dados disponíveis para avaliação. Este trabalho foi fundamental para o desenvolvimento das estratégias adotadas neste projeto, tais como a implementação de técnicas baseadas em geração aumentada por recuperação (*retrieval-augmented generation* ou RAG) (LEWIS et al., 2020) e a verificação factual das respostas.

Dong et al. (DONG et al., 2024) discutem mecanismos específicos para construção de *guardrails* em modelos de linguagem. O estudo sugere abordagens como filtragem baseada em regras, validação factual automatizada e mecanismos complementares de revisão por feedback humano. Essas estratégias foram diretamente incorporadas à arquitetura desenvolvida, especialmente na implementação de *guadrails* contra conteúdo sensível, proteção de dados pessoais e prevenção de violações de escopo.

Ayyamperumal e Ge (AYYAMPERUMAL; GE, 2024) ampliam a discussão sobre os riscos associados a grandes modelos de linguagem (LLMs), destacando três princípios fundamentais para sistemas seguros: transparência na geração de respostas, definição clara do escopo operacional e responsabilidade documentada nas decisões automáticas. Essas diretrizes embasaram significativamente as decisões do projeto, particularmente na definição dos limites de uso dos modelos e no registro das interações realizadas.

Li et al. (LI et al., 2024) abordam especificamente o fenômeno conhecido como *persona drift*, situação em que a identidade projetada por um modelo pode variar ao longo da interação. Este estudo demonstrou a importância de manter a coerência de personalidade e estilo ao longo das interações, indicando que uma deriva inconsistente pode prejudicar a experiência do usuário e comprometer a confiabilidade das respostas. As conclusões desse trabalho fundamentaram o desenvolvimento de mecanismos que buscam garantir a consistência temática e estilística nas respostas fornecidas pelos modelos empregados.

Outro aspecto relevante abordado por Xu et al. (XU et al., 2024) refere-se à criação de *guardrails adaptativas* orientadas à confiança. Os autores propõem que essas *guadrails* não sejam meramente restritivas, mas adaptativas e sensíveis ao contexto, capazes de ajustar dinamicamente seu rigor

conforme a situação. Essa abordagem reforçou a proposta deste projeto para a criação futura de um orquestrador inteligente, permitindo o controle dinâmico e contextual das *guadrails*.

Em suma, a combinação dessas referências estabeleceu uma base sólida para a construção da arquitetura modular adotada neste trabalho, garantindo que as *guardrails* desenvolvidas estejam alinhadas com as melhores práticas acadêmicas e técnicas atuais, fortalecendo a segurança e o comportamento das aplicações de *IA* generativa propostas.

2 Metodologia

2.1 Descrição das Guardrails Utilizadas

As *guardrails* serão classificadas em diferentes categorias, cada uma responsável por mitigar riscos específicos na IA generativa.

2.1.1 Toxicidade

A mitigação da toxicidade nas respostas geradas é essencial para preservar um ambiente seguro e respeitoso para os usuários, evitando discursos ofensivos, preconceituosos ou inadequados.

A mitigação da toxicidade é realizada por meio de modelos de classificação treinados via *fine-tuning* (HOWARD; RUDER, 2018) em conjuntos de dados anotados para detecção de linguagem tóxica. Esse processo adapta modelos de linguagem pré-treinados para a tarefa específica de identificar conteúdos ofensivos, discriminatórios ou impróprios.

2.1.2 Jailbreak

Ataques *jailbreak* são tentativas maliciosas de burlar as restrições impostas à IA por meio da manipulação sofisticada de *prompts*, buscando induzir respostas inadequadas ou proibidas. Esses ataques representam riscos significativos à segurança e integridade operacional de sistemas de IA generativa. Um exemplo de situação de *jailbreak* ocorre quando um usuário tenta induzir a IA a fornecer instruções perigosas — como a fabricação de explosivos ou a realização de atividades ilegais — ao envolver o pedido em um contexto fictício, como um "jogo de simulação"ou um "cenário acadêmico hipotético", explorando brechas na interpretação da IA para obter respostas que normalmente seriam bloqueadas pelos filtros de segurança.

Para prevenir esses ataques, são utilizadas técnicas que envolvem a detecção automática de padrões comuns em tentativas de *jailbreak*. Por meio de ferramentas como os detectores de *jailbreak* do *framework NeMo* (NVIDIA) e da biblioteca Guardrails AI é possível identificar *prompts* maliciosos, bloqueando-os em tempo real.

2.1.3 Dados Sensíveis

A proteção de dados sensíveis é crítica, especialmente em contextos que envolvem informações pessoais ou confidenciais, devendo estar alinhada às regulamentações legais como a Lei Geral de Proteção de Dados Pessoais (LGPD) e a *General Data Protection Regulation* (GDPR). Essa proteção visa evitar vazamentos inadvertidos ou exposições indevidas de dados durante as interações com assistentes virtuais.

Para reforçar a segurança dos dados, técnicas de mascaramento e anonimização são aplicadas. Além disso, filtros específicos são implementados para identificar e bloquear automaticamente informações pessoais, como nomes, documentos, endereços e outros dados confidenciais.

2.1.4 Escopo

A restrição de escopo favorece o alinhamento das respostas geradas pelos modelos de IA ao domínio específico pretendido, prevenindo respostas genéricas, irrelevantes ou inadequadas ao contexto desejado. Essa delimitação garante que o assistente virtual atenda às expectativas definidas pelas organizações, mantendo o foco e coerência nas interações.

Como suporte a essa restrição, foram implementadas estratégias de *Retrieval-Augmented Generation* (RAG) (LEWIS et al., 2020) e *fine-tuning* (HOWARD; RUDER, 2018), que combinam geração de texto com recuperação de informações relevantes a partir de bases externas.

2.1.5 Alucinação

Alucinação em IA generativa refere-se à criação de informações falsas ou infundadas pelas IAs, sendo um risco significativo para a confiabilidade das respostas geradas. É importante mitigar esse risco para que os usuários recebam apenas informações verificáveis e precisas.

A mitigação desse risco é realizada por meio de uma abordagem de RAG (LEWIS et al., 2020), que complementa o modelo com informações externas relevantes. Essa estratégia orienta as respostas dentro do escopo temático definido, reduzindo a ocorrência de alucinações. O modelo passa a se apoiar em bases de dados específicas, recuperando conteúdos factuais durante a geração, o que aumenta a precisão e a confiabilidade das respostas.

2.1.6 Competidores

O bloqueio às menções de empresas competidores tem papel fundamental em preservar a conformidade com políticas internas das organizações, evitando que a IA não forneça informações sensíveis ou inadequadas relativas a competidores diretos. Esse controle evita riscos jurídicos e comerciais associados à divulgação inadvertida de conteúdos competitivos.

Esse bloqueio é implementado via *fine-tuning* (HOWARD; RUDER, 2018), que ajusta o modelo para reconhecer e evitar menções a empresas concorrentes. O treinamento é feito com exemplos específicos, reforçando o comportamento de omitir nomes.

2.2 Utilização e Construção dos Conjuntos de Dados

2.2.1 Critério de Utilização de Bases Reais e Sintéticas

Os conjuntos de dados foram definidos com base na disponibilidade e na adequação de bases reais ao contexto de cada *guardrail*. Quando uma base pública era considerada suficientemente representativa para a tarefa em questão, ela foi utilizada de forma exclusiva. Se, por outro lado, a base real apresentava limitações em termos de diversidade ou cobertura, optou-se por complementá-la com dados sintéticos gerados localmente. Já na ausência total de bases adequadas, a solução adotada foi a construção integral de conjuntos sintéticos.

2.2.2 Construção das Bases Sintéticas

A geração dos dados sintéticos seguiu um processo estruturado e replicável. Foram utilizados modelos de linguagem para produzir mensagens que simulassem interações realistas com assistentes virtuais. Para orientar essa geração, definiram-se diferentes tipos de comandos (*prompts*) com instruções semânticas específicas (por exemplo, elogios a concorrentes), cenários simulados (como "fila do banco" ou "plantão noturno") e variações linguísticas, incluindo gírias, emojis, abreviações e erros tipográficos.

Os conjuntos sintéticos foram construídos com equilíbrio entre mensagens positivas e negativas, além de contemplarem exemplos claros e ambíguos. No caso da barreira de competidores no domínio bancário, por exemplo, mensagens com elogios explícitos a concorrentes foram classificadas como positivas claras, enquanto dúvidas ou suspeitas sutis foram consideradas positivas ambíguas.

Todo o processo foi automatizado por meio de *scripts* parametrizados, responsáveis por controlar a aleatoriedade, injetar ruídos linguísticos e aplicar, quando necessário, técnicas de parafraseamento para ampliar a diversidade dos exemplos. Por fim, foram realizadas validações adicionais, como a verificação de vazamento de estilo, utilizando-se métricas como a AUC para assegurar que os modelos não estivessem se baseando em padrões superficiais de escrita. Além dessa verificação quantitativa, também foram conduzidas avaliações qualitativas com análise manual de amostras geradas. A equipe inspecionou os padrões recorrentes nos *prompts* e nas mensagens produzidas, ajustando iterativamente os comandos e as opções de geração até que os exemplos refletissem com fidelidade o comportamento esperado para cada classe.

2.2.3 Uso dos Dados nas Etapas do Desenvolvimento

Os conjuntos de dados desenvolvidos foram utilizados em diferentes fases do processo de desenvolvimento, sempre com o devido cuidado para evitar qualquer tipo de vazamento entre etapas. A divisão entre treino, validação e teste foi feita de forma criteriosa e adaptada às necessidades de cada tarefa, podendo envolver dois ou três subconjuntos, conforme o caso.

Em tarefas supervisionadas que exigiram *fine-tuning* (HOWARD; RUDER, 2018), uma parte dos dados foi reservada exclusivamente para o treinamento dos modelos. Em outras situações, os dados foram empregados apenas para testes ou para a calibração de limiares de decisão (*thresholds*). O conjunto de teste final permaneceu sempre isolado das demais etapas e foi utilizado apenas na avaliação final das *guardrails*, assegurando a integridade e a validade das métricas obtidas. Essa abordagem flexível permitiu alinhar os dados às exigências de cada cenário, mantendo rigor metodológico e evitando qualquer forma de vazamento.

2.3 Avaliação

A avaliação das *guardrails* foi realizada, sempre que aplicável, por meio de testes automatizados com métricas quantitativas. Foram utilizadas métricas como precisão, *recall* e F_1-*score* para medir a capacidade de cada solução em identificar corretamente situações de risco, como conteúdos inadequados, vazamentos de dados sensíveis ou tentativas de *jailbreak*.

Utilizou-se também a métrica F_2-*score*, que atribui maior peso ao *recall*. Essa escolha se justifica pelo fato de que, em mecanismos de filtragem, falsos negativos tendem a ser mais problemáticos do que falsos positivos. Ainda assim, a seleção de métricas considerou o equilíbrio entre precisão e *recall*, evitando distorções na interpretação dos resultados.

É importante notar que a métrica F_β[1] é definida da seguinte forma:

$$F_\beta = \frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP + \beta^2 \cdot FN + FP} \tag{2.1}$$

onde TP, FP e FN representam, respectivamente, os verdadeiros positivos, falsos positivos e falsos negativos. No caso da F_2-*score*, utiliza-se $\beta = 2$, o que aumenta a penalização para falsos negativos; já para a F_1-score, $\beta = 1$, o que penaliza falsos negativos e falsos positivos igualmente.

[1] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html>

3 Desenvolvimento

3.1 Arquitetura e Implementação das *Guardrails*

O sistema de moderação foi desenvolvido a partir de uma arquitetura em dois estágios: validação da entrada do usuário e validação da resposta gerada. Para evidenciar a contribuição das *guardrails*, a Figura 3.1 ilustra o funcionamento básico de uma aplicação tradicional baseada em LLMs, sem mecanismos de contenção. Nesse cenário, o *prompt* é diretamente enviado ao modelo, e a resposta é retornada ao usuário sem qualquer tipo de filtragem.

Em contraste, a Figura 3.2 apresenta a estrutura completa da aplicação com as *guardrails* integradas. Nesse fluxo, os conteúdos são interceptados tanto na entrada quanto na saída, permitindo a aplicação de múltiplas *guadrails* especializadas em mitigar riscos distintos, como alucinação, vazamento de dados sensíveis, violações de escopo, toxicidade e referências indevidas a competidores.

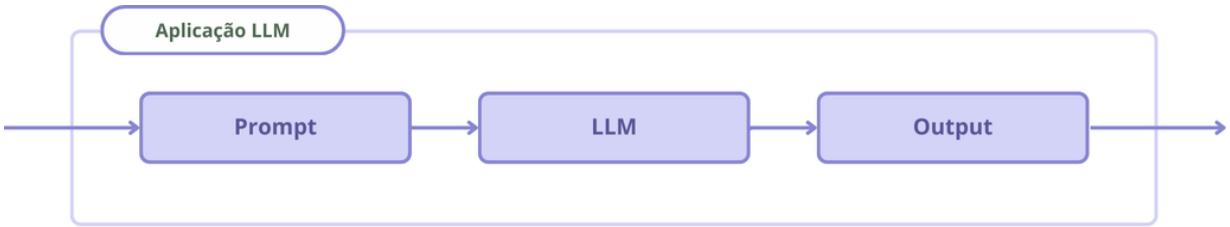

Figura 3.1: Fluxo tradicional de uma aplicação com LLM, sem *guadrails* de validação.

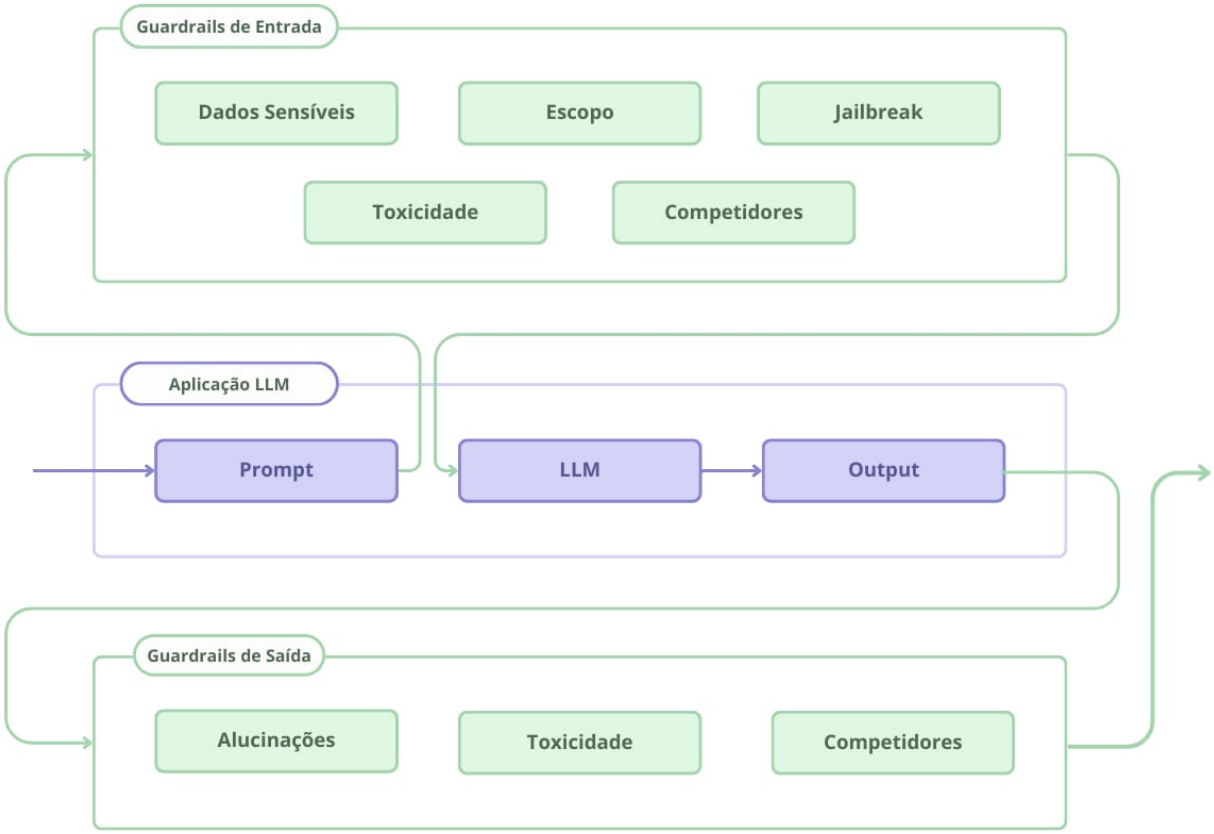

Figura 3.2: Fluxo completo com aplicação das *guardrails* nos estágios de entrada e saída da aplicação.

As subseções a seguir detalham a implementação de cada uma das *guardrails*, explicando suas funções, modelos utilizados e estratégias de integração.

3.1.1 Toxicidade

Para a tarefa de detecção de toxicidade, foram utilizados três modelos baseados na arquitetura *BERT* (DEVLIN et al., 2019): *BERTimbau* (SOUZA, 2020), *mBERT* (DEVLIN et al., 2019) e *Unbiased-Toxic-RoBERTa* (UNITARY, 2024). Cada modelo passou por um processo de *fine-tuning* supervisionado (HOWARD; RUDER, 2018) utilizando o conjunto de dados *ToLD-Br*, disponível na plataforma *Hugging Face Datasets*, que contém frases em português brasileiro rotuladas como tóxicas (ofensivas, ameaçadoras, com discurso de ódio) e não tóxicas.

O conjunto de dados foi dividido em três subconjuntos: treino (para treinamento supervisionado dos modelos), validação (para calibração dos limiares de decisão) e teste (reservado exclusivamente para avaliação final do desempenho). Antes do treinamento, todos os textos foram tokenizados com limite de 128 *tokens*, aplicando técnicas de truncamento para frases mais longas e *padding* para frases mais curtas, garantindo uniformidade das entradas.

O treinamento foi conduzido com o *framework Hugging Face Transformers* (Hugging Face, 2024), integrado ao *PyTorch Lightning* (FALCON; TEAM, 2025), permitindo uso eficiente da GPU,

checkpointing automático e organização modular do código. Durante o *fine-tuning*, foi utilizada a função de perda *Binary Cross-Entropy (BCE)* (PYTORCH, 2024), adequada para classificação binária, penalizando erros em ambas as direções (falsos positivos e falsos negativos).

Após o treinamento, realizou-se a calibração dos limiares de decisão (*thresholds*) com base no conjunto de validação. Diferentes valores no intervalo de 0 a 1 foram testados, selecionando-se o limiar que maximizou o F_2-*score*, devido à maior relevância atribuída aos falsos negativos em tarefas de moderação de conteúdo. Os *thresholds* finais definidos foram: 0.20 para o *BERTimbau*, 0.03 para o *mBERT* e 0.11 para o *Unbiased-Toxic-RoBERTa*.

Os modelos ajustados foram encapsulados em classes modulares e integradas à arquitetura principal das *guardrails*. Durante a inferência, essas classes analisam as mensagens em tempo real e bloqueiam automaticamente aquelas cuja pontuação de toxicidade ultrapasse o limiar definido, garantindo assim a moderação adequada do conteúdo gerado.

3.1.2 Jailbreak

Para desenvolver e avaliar a *guardrail* de *jailbreak*, foi utilizada uma base composta pela união do subconjunto *RedTeam_2K* da *JailbreakV-28K*[1], que contém 2.000 *prompts* adversariais, com um conjunto sintético de *prompts* benignos gerados localmente. Esses *prompts* neutros incluem perguntas informativas e instruções genéricas, construídas com o objetivo de equilibrar a distribuição das classes.

Todo o conjunto foi traduzido automaticamente para o português utilizando o Google Tradutor, buscando preservar a estrutura e o significado original dos *prompts*. A base final resultante é binária, balanceada e adaptada à língua portuguesa. Após a tradução, a base foi dividida aleatoriamente entre treino (80%) e teste (20%), garantindo que o ajuste de modelos e a avaliação fossem realizados em subconjuntos independentes.

DetectJailbreak (Guardrails AI)

O módulo *DetectJailbreak*, da biblioteca *GuardrailsAI*, implementa um sistema de detecção baseado em *ensemble*, composto por três detectores independentes que utilizam modelos do tipo *Transformer*[2]. O objetivo do sistema é identificar tentativas de *jailbreak* em *prompts* de entrada, conforme descrito em (Guardrails AI, 2024).

O primeiro detector utiliza o modelo *sentence-transformers/all-MiniLM-L6-v2*, que gera *embeddings* semânticos para mensuração de similaridade entre frases. Esse método segue a arquitetura *Sentence-BERT* (REIMERS; GUREVYCH, 2019) e emprega o modelo *MiniLM*, treinado com *distillation*[3] para reduzir o custo computacional (WANG et al., 2020).

[1] <https://huggingface.co/datasets/JailbreakV-28K/JailBreakV-28k>
[2] <https://jalammar.github.io/illustrated-transformer/>
[3] <https://neptune.ai/blog/knowledge-distillation>

O segundo detector é baseado no modelo *zhx123/ftrobertaLLM*, que utiliza a arquitetura *RoBERTa*, uma variação do *BERT* (DEVLIN et al., 2019) com alterações no processo de *pretraining* (LIU et al., 2019). Após o pré-treinamento, o modelo é ajustado por *fine-tuning* (HOWARD; RUDER, 2018) com dados rotulados para realizar a classificação binária de *prompts* (XU; WANG; CUI, 2024).

O terceiro detector, denominado *PromptSaturationDetectorV3*, é responsável pela identificação de *prompt saturation attacks*, que se caracterizam por entradas extensas, repetitivas ou com estrutura textual atípica. Esse componente utiliza um modelo binário baseado em *BERT* (DEVLIN et al., 2019), com *tokenization*[4] feita pelo *bert-base-cased*, conforme especificado em (DONG et al., 2024).

Cada detector produz uma pontuação entre 0 e 1. A pontuação final considerada corresponde ao maior valor entre os três. A classificação do *prompt* como *jailbreak* ou seguro é determinada com base em um limiar (*threshold*) previamente definido.

O uso conjunto dos três detectores permite cobrir diferentes dimensões do problema: similaridade semântica, padrões linguísticos e estrutura textual.

Jailbreak Detection Heuristics (NeMo Guardrails)

A ferramenta de detecção de *jailbreaks* da *NeMo Guardrails*, desenvolvida pela NVIDIA, implementa um conjunto de mecanismos projetados para interceptar e mitigar instruções potencialmente maliciosas enviadas a modelos de linguagem. Conforme descrito na documentação oficial (NVIDIA, 2025), o sistema opera interceptando os *prompts* antes de seu encaminhamento à LLM, aplicando regras configuráveis baseadas em heurísticas. Dependendo da avaliação, a entrada pode ser bloqueada, modificada ou sinalizada.

Essas heurísticas baseiam-se em duas estratégias principais: *Length per Perplexity* (LPP) e *Prefix and Suffix Perplexity* (PSP). Ambas utilizam a métrica de perplexidade[5], que estima o grau de imprevisibilidade de um texto.

A heurística LPP calcula a razão entre o número de tokens do *prompt* (L) e sua perplexidade (P), segundo a equação:

$$\mathrm{LPP} = \frac{L}{P} \tag{3.1}$$

Razões elevadas podem indicar tentativas estruturadas de contornar restrições, enquanto razões muito baixas sugerem incoerência intencional.

Complementarmente, a heurística PSP avalia a consistência da distribuição de perplexidade ao longo do *prompt*. Para isso, o texto é dividido em duas partes: um prefixo (início do *prompt*) e um

[4]<https://www.kaggle.com/code/satishgunjal/tokenization-in-nlp>
[5]<https://klu.ai/glossary/perplexity>

sufixo (final do *prompt*). Em seguida, calcula-se a perplexidade de cada parte separadamente — P_{pre} e P_{suf}, respectivamente.

A lógica dessa abordagem se baseia na hipótese de que instruções maliciosas muitas vezes são inseridas de forma disfarçada no final do *prompt*, precedidas por um conteúdo inócuo no início. Assim, grandes diferenças entre P_{pre} e P_{suf} podem indicar tentativas de manipulação intencional. Essa discrepância é mensurada pela fórmula:

$$\text{PSP} = |P_{\text{pre}} - P_{\text{suf}}| \tag{3.2}$$

Valores elevados de PSP sugerem que o padrão de linguagem entre as duas regiões do *prompt* é inconsistente, o que pode caracterizar um *jailbreak* disfarçado.

Para a execução dessas heurísticas, é necessária a integração com uma LLM. Foram avaliadas duas opções: o modelo GPT-4o, acessado via *Azure OpenAI*[6], e o modelo *LLaMa 3.2 Lightweight*, executado localmente por meio do motor *Ollama*[7]. A opção de acesso via *Azure* foi descartada, pois a plataforma bloqueia automaticamente *prompts* que contenham conteúdo explícito associado a *jailbreaks*, impedindo a validação das heurísticas. Por esse motivo, optou-se pelo uso local do modelo da *Meta*.

Após o processamento do *prompt*, o sistema analisa a resposta retornada pela LLM para verificar se houve recusa. Essa análise é realizada por meio de técnicas de *fuzzy matching*[8], utilizando uma lista predefinida de frases associadas a bloqueios, como "não posso ajudar com isso". Essa abordagem permite a detecção de recusas mesmo quando há variações linguísticas sutis.

Toda a configuração do sistema — incluindo o modelo selecionado, os limiares das heurísticas e demais parâmetros — é armazenada em arquivos externos. Isso permite ajustes posteriores com base em resultados empíricos observados.

Finalmente, conforme as diretrizes da própria documentação (NVIDIA, 2025), foi integrado ao projeto um servidor dedicado à execução dessas heurísticas. Esse servidor, implementado com *FastAPI*[9], está disponível no repositório oficial da NVIDIA e é ativado automaticamente pela infraestrutura do sistema. Ele expõe *endpoints* que permitem a verificação de *prompts* em tempo real, facilitando a integração direta da lógica de detecção ao fluxo de mensagens.

Fine-tuning* do *BERTimbau

Como uma das abordagens para detecção de *jailbreak*, foi realizado o *fine-tuning* do modelo *BERTimbau* (SOUZA, 2020). O modelo foi ajustado para uma tarefa de classificação binária com os rótulos *jailbreak* e benigno, utilizando exclusivamente o subconjunto de treino da base descrita

[6]<https://azure.microsoft.com/en-us/products/cognitive-services/openai-service>
[7]<https://ollama.com/>
[8]<https://cloud.google.com/discover/what-is-fuzzy-search>
[9]<https://fastapi.tiangolo.com/>

anteriormente.

O processo de *fine-tuning* permite especializar um modelo linguístico já treinado para o português para reconhecer padrões específicos associados a *prompts* adversariais. Após o treinamento, o modelo foi integrado à *pipeline* de guardrails da aplicação, operando de forma local e em tempo real. A predição é feita com base no rótulo de maior probabilidade.

Embora essa abordagem tenha se mostrado eficaz, é importante reconhecer que o uso de *prompts* sintéticos e a tradução automática dos dados originais podem introduzir vieses, influenciando positivamente os resultados e simplificando a tarefa de classificação. Esses fatores são discutidos com mais profundidade na seção de avaliação.

3.1.3 Dados Sensíveis

A implementação da *guadrail* de dados sensíveis foi desenvolvida com base na biblioteca *open-source Presidio*, mantida pela Microsoft (MICROSOFT, 2025), e integrada a um módulo dedicado escrito em *Python*. O sistema combina múltiplas estratégias de detecção, incluindo correspondência por padrões regulares (*regex*), análise de contexto e validação semântica, com o objetivo de aumentar a precisão e reduzir falsos positivos.

As entidades consideradas sensíveis incluem: CPF, CNPJ, RG, número de telefone, número de cartão de crédito, conta bancária, data de nascimento e endereço de e-mail. O processo inicia-se com a identificação de padrões estruturados por meio de expressões regulares. Em seguida, é realizada uma análise de contexto com base em palavras-chave associadas, como “documento”, “banco”, “nascimento” ou “contato”, que auxiliam na confirmação semântica do tipo da informação.

Para complementar essa análise e reforçar a confiabilidade da detecção, o sistema aplica validações estruturais específicas: verificação de dígitos verificadores de CPF e CNPJ, checagem de validade de datas, e aplicação do algoritmo de *Luhn* (HUSSEIN et al., 2013) para números de cartões de crédito.

As informações validadas são substituídas automaticamente por *tokens* neutros, como `<CPF>`, `<EMAIL>` ou `<DATA_NASC>`, utilizando o módulo de anonimização do *Presidio*. A funcionalidade de reconhecimento de entidades baseada em modelos de *NLP* (JURAFSKY; MARTIN, 2023) do *Presidio* não foi utilizada, pois não oferece suporte adequado ao idioma português, o que impactaria negativamente a precisão contextual.

3.1.4 Escopo e Alucinação

As *guardrails* de escopo e alucinação foram estruturadas em dois níveis complementares: temático e documental. O nível temático verifica se as entradas dos usuários pertencem ao escopo geral da aplicação, impedindo que perguntas irrelevantes ou inadequadas sejam encaminhadas ao modelo. Já o nível documental atua com maior granularidade, restringindo a geração das respostas apenas às

informações explicitamente presentes nos documentos indexados. Essa separação permite avaliar de forma independente tanto a capacidade geral de controle temático quanto a precisão específica na mitigação de alucinações, temas abordados detalhadamente nos resultados a seguir.

Nível Temático

O nível temático atua de forma mais ampla, validando se as mensagens recebidas estão alinhadas ao contexto geral definido pela aplicação. Sua implementação baseia-se em um modelo de classificação treinado por meio de *fine-tuning* (HOWARD; RUDER, 2018) do *BERTimbau* (SOUZA, 2020), modelo pré-treinado para o português. Esse treinamento foi realizado com um conjunto de dados balanceado contendo exemplos reais e sintéticos, incluindo casos ambíguos. Essa estratégia visou aprimorar a capacidade do classificador em diferenciar adequadamente mensagens dentro e fora do escopo, mesmo na presença de sobreposição parcial de termos.

Durante a inferência, a *guardrail* classifica cada entrada, atribuindo uma probabilidade à categoria relacionada ao domínio geral. Caso essa probabilidade não alcance um limiar pré-definido, a mensagem é bloqueada por estar fora do escopo pretendido. Esse mecanismo contribui diretamente para os resultados observados posteriormente na avaliação, especialmente na prevenção de perguntas irrelevantes ou ambíguas.

Nível Documental

O nível documental foi implementado com o objetivo de restringir a geração de respostas a informações explicitamente presentes em documentos previamente indexados, atuando tanto na mitigação de alucinações quanto no controle de escopo em nível mais específico.

A implementação segue a arquitetura ilustrada na Figura 3.3. Primeiramente, realiza-se a extração textual dos documentos por meio de ferramentas como *pdfplumber*[10], para arquivos PDF, e *BeautifulSoup*[11], para HTML. Apenas trechos em português são mantidos, segmentados em unidades de aproximadamente 200 palavras, com sobreposição de 50 palavras para preservar o contexto local.

Em seguida, esses *chunks* (trechos) são indexados em duas estruturas paralelas: uma semântica, baseada em *embeddings* gerados pelo modelo *intfloat/multilingual-e5-large* (WANG et al., 2024) e armazenados via FAISS (DOUZE et al., 2024); e outra lexical, utilizando o método BM25 (ROBERTSON; ZARAGOZA, 2009), implementado com a biblioteca *rank-bm25*[12]. Ambas são utilizadas para recuperar os K *chunks* mais relevantes em relação ao conteúdo do *prompt* do usuário.

Esses trechos são emparelhados com o *prompt* original, formando pares do tipo (prompt, chunk$_i$) que são processados por um modelo do tipo *Cross-Encoder* (*unicamp-dl/mMiniLM-L6-v2-en-pt-msmarco-v2* (BONIFACIO et al., 2021)), que calcula um score de similaridade entre o *prompt* e

[10]<https://github.com/jsvine/pdfplumber>
[11]<https://www.crummy.com/software/BeautifulSoup/>
[12]<https://github.com/dorianbrown/rank_bm25>

cada *chunk*. Caso nenhum dos scores supere um limiar pré-definido, a entrada do usuário é bloqueada por ausência de base documental suficiente. Do contrário, a entrada é permitida e os K *chunks* são repassados para a LLM como contexto para a geração da resposta.

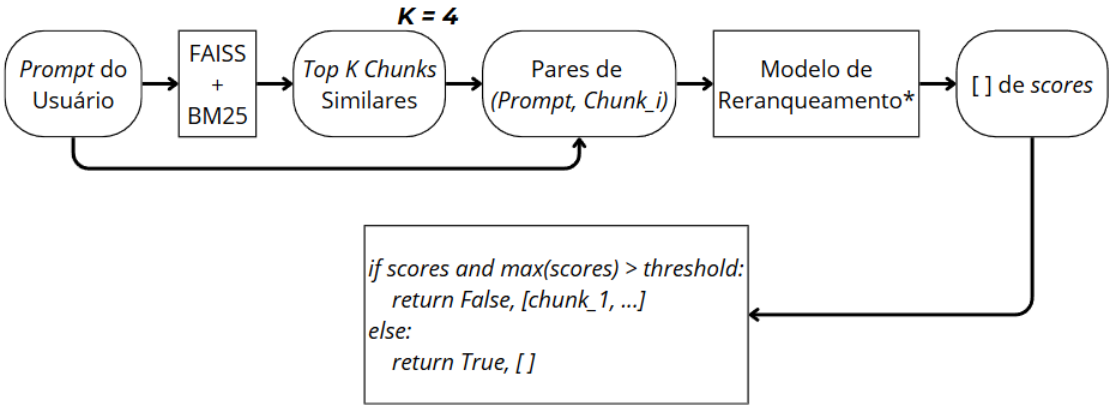

Figura 3.3: Fluxo de decisão do nível documental.

Esse processo funciona como uma *guardrail* que impede o modelo de responder perguntas para as quais não exista suporte documental explícito, reduzindo assim a incidência de alucinações e garantindo aderência ao escopo definido pelo conteúdo indexado.

3.1.5 Competidores

A *guardrail* de competidores foi desenvolvida para identificar e bloquear mensagens que façam menções indevidas a instituições concorrentes, mitigando riscos legais, comerciais e de quebra de neutralidade institucional. Seu objetivo é garantir que o assistente virtual não reproduza, promova ou compare informações que envolvam diretamente empresas concorrentes do setor.

A solução é baseada no modelo *BERTimbau* (SOUZA, 2020), *fine-tuned* (HOWARD; RUDER, 2018) para a tarefa de detecção de menções sensíveis a competidores. O treinamento foi realizado com uma base de dados sintética, altamente diversificada, composta por exemplos claros e ambíguos de menções permitidas e indevidas. Essa base foi cuidadosamente construída utilizando modelos de linguagem para gerar frases realistas, com variações linguísticas que incluem gírias, erros tipográficos, abreviações e uso contextualizado de *emojis*, simulando conversas reais de usuários.

Durante a inferência, a *guardrail* processa a entrada do usuário e classifica a mensagem como permitida ou indevida. O classificador retorna uma probabilidade associada a cada classe, e decisões são tomadas com base no rótulo mais provável. Se a mensagem for classificada como indevida, a resposta é bloqueada, assegurando que o assistente opere dentro dos limites estabelecidos pelas diretrizes de privacidade, neutralidade e *compliance*.

3.2 Organização do Repositório

3.2.1 Visão Geral

O repositório do projeto foi estruturado de forma modular, com os componentes organizados segundo o tipo de risco mitigado. Cada módulo implementa uma *guardrail* especializada, responsável por detectar e bloquear categorias específicas de risco, como *jailbreak*, toxicidade, escopo, dados sensíveis e menções a competidores. A arquitetura é fundamentada em duas classes centrais: uma classe base, que define a interface comum a todas as guardrails e assegura a padronização e a interoperabilidade entre módulos; e uma classe de pipeline, que coordena a execução sequencial das guardrails em dois estágios — entrada e saída.

Além disso, o repositório adota uma separação clara entre ambientes. A *branch* principal contém a versão de produção da biblioteca, mais enxuta e estável, preparada para integração. Já a *branch* de desenvolvimento concentra os testes automatizados, ferramentas auxiliares e experimentações realizadas durante a construção do sistema, que não fazem parte da biblioteca.

3.2.2 Estrutura do Diretório

A seguir, a organização dos principais diretórios do projeto:

```
src/
|-- tests/ # Testes automatizados
`-- llm_guardrails/
   |-- base.py # Interface padro das guardrails
   |-- pipeline.py # Lgica de execuo sequencial
   |-- jailbreak/ # Guardrail de jailbreak
   |-- off_topic/ # Guardrails de escopo
   |-- sensitive_content/ # Guardrail de dados sensveis
   |-- toxicity/ # Guardrail de toxicidade
   `-- competitors/ # Guardrail de concorrentes
```

3.2.3 Arquitetura de Execução via *Pipeline*

A execução do sistema segue uma arquitetura sequencial composta por dois estágios principais: entrada e saída. No estágio de entrada, as *guardrails* são aplicadas antes do envio do *prompt* ao modelo de linguagem. Elas podem tanto bloquear o conteúdo quanto transformá-lo — por exemplo, por meio da anonimização de dados sensíveis. Caso alguma dessas *guardrails* detecte uma violação, a execução é interrompida e uma mensagem padrão de bloqueio, definida pelo próprio módulo, é imediatamente retornada ao usuário.

Se o *prompt* for aprovado por todas as *guardrails* de entrada, ele é encaminhado à LLM, que gera uma resposta. Em seguida, essa resposta passa pelo segundo estágio da *pipeline*, onde são

aplicadas as *guardrails* de saída. Essas *guardrails* validam o conteúdo retornado pela IA e podem bloqueá-lo caso identifiquem problemas. Se nenhuma violação for detectada, a resposta é entregue normalmente ao usuário final.

A Figura 3.4 ilustra esse fluxo completo de execução, evidenciando a separação entre os dois estágios de validação e o papel de cada componente na garantia da segurança e adequação das interações.

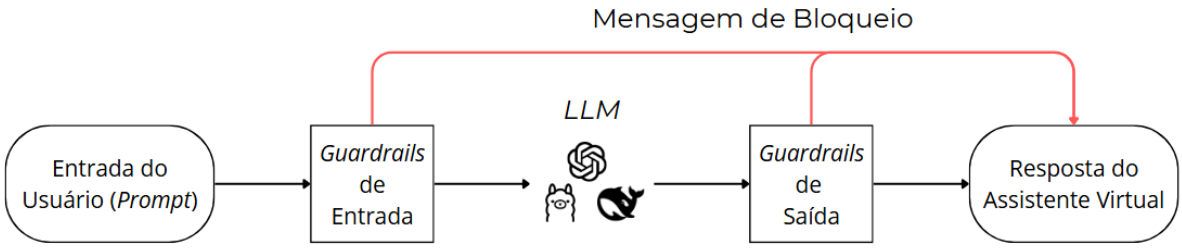

Figura 3.4: Arquitetura de execução da pipeline com guardrails de entrada e saída

3.2.4 Exemplo de Uso

O trecho a seguir apresenta um exemplo completo de uso da *pipeline*, incluindo a seleção do modelo de linguagem, a adição e remoção dinâmica de *guardrails*, e a execução do fluxo com um *prompt* de entrada.

```
pipeline = Pipeline()

pipeline.set_llm("deepseek")

pipeline.add(PresidioSensitiveDataGuardrail(), "input")
pipeline.add(DocGuardrail(threshold=-4.2234), "input")
pipeline.add(BertimbauToxicityGuardrail(threshold=0.28), "output")

pipeline.remove("BertimbauToxicityGuardrail", "output")

resposta = pipeline.run("Esse  um prompt de exemplo.")
```

Esse exemplo demonstra a flexibilidade do sistema: a qualquer momento, guardrails podem ser adicionadas, substituídas ou removidas de forma seletiva, mantendo a integridade do fluxo de validação.

4 Resultados

4.1 Demonstração Interativa

A funcionalidade das *guardrails* foi demonstrada por meio de uma aplicação web desenvolvida especificamente para simular sua atuação em um assistente virtual baseado em IA. A interface permite que o usuário envie mensagens ao sistema e acompanhe, em tempo real, os bloqueios aplicados por cada módulo. Para fins de controle e experimentação, a aplicação oferece uma barra lateral que possibilita a ativação ou desativação individual de cada *guardrail*, permitindo testar diferentes configurações de proteção ao longo da interação.

Figura 4.1: Interface da aplicação web utilizada para demonstrar o funcionamento das *guardrails*.

4.2 Avaliação por Tipo de Risco Moderado

4.2.1 Toxicidade

A lógica de decisão para o bloqueio de mensagens foi construída com base na análise das saídas de probabilidade dos modelos de classificação. Como ilustrado na Figura 4.2, cada *prompt* enviado pelo usuário é processado por um classificador binário, que retorna dois *scores*: *toxicScore*

(probabilidade de a mensagem ser tóxica) e *normalScore* (probabilidade de a mensagem ser não tóxica).

Uma mensagem é considerada tóxica e, portanto, bloqueada, quando duas condições são atendidas simultaneamente: o *toxicScore* deve ser maior que o *normalScore*, indicando maior probabilidade de toxicidade, e o *toxicScore* também precisa exceder um limiar calibrado para cada modelo durante a validação.

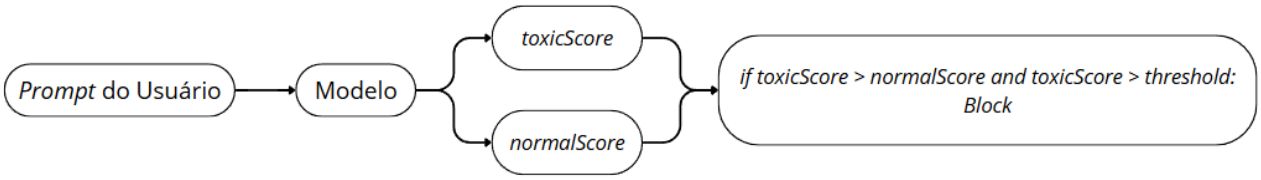

Figura 4.2: Fluxo de decisão baseado nos *scores* de classificação e limiar calibrado.

Essa abordagem permite aumentar a confiabilidade do sistema, atuando apenas sobre mensagens com alta probabilidade de toxicidade. Mensagens ambíguas, com *scores* similares ou com baixa confiança de classificação, não são bloqueadas automaticamente. Isso reduz a incidência de falsos positivos, contribuindo para uma moderação mais precisa.

Com essa lógica aplicada, os modelos foram avaliados quantitativamente sobre o conjunto de teste da base *ToLD-Br*, composto por 2100 exemplos rotulados. As métricas de avaliação incluem precisão, *recall*, F_1-*score* e F_2-*score*, apresentadas na Tabela 4.1.

Modelo	**Threshold**	**Precisão**	**Recall**	**F1-score**	**F2-score**
BERTimbau	0.20	70%	87%	77.6%	82.9%
mBERT	0.03	62%	94%	74.7%	85.0%
Unbiased-Toxic-RoBERTa	0.11	59%	94%	72.7%	84.0%

Tabela 4.1: Métricas obtidas na detecção de toxicidade sobre o conjunto de teste da *ToLD-Br*.

Os resultados refletem limitações dos modelos utilizados e a complexidade da tarefa. A detecção de toxicidade envolve interpretar nuances linguísticas como ambiguidade, ironia, sarcasmo e contexto, o que é desafiador para modelos baseados em *BERT* (DEVLIN et al., 2019), que não capturam plenamente essas dependências. Frequentemente, a toxicidade está associada a construções discursivas sutis, e não apenas ao uso de termos explícitos.

Modelos de linguagem mais avançados, como LLMs com maior capacidade de compreensão contextual, podem contribuir significativamente nesse cenário. Por conseguirem analisar sequências mais longas com atenção a relações pragmáticas entre sentenças, essas arquiteturas tendem a iden-

tificar toxicidade implícita com mais precisão. Isso ajuda a reduzir falsos negativos sem aumentar indevidamente os falsos positivos, resultando em uma moderação mais equilibrada.

4.2.2 Jailbreak

A avaliação da *guardrail* de *jailbreak* foi conduzida utilizando uma base composta pela união do subconjunto *RedTeam_2K* da *JailbreakV-28K*[1], traduzido para o português, com um conjunto sintético de *prompts* benignos gerado localmente. Esse conjunto combinado foi então dividido aleatoriamente em treino (80%) e teste (20%). Dessa forma, o modelo *BERTimbau* (SOUZA, 2020) passou por um processo de *fine-tuning* (HOWARD; RUDER, 2018) utilizando apenas o conjunto de treino, enquanto todas as avaliações quantitativas foram realizadas exclusivamente sobre o conjunto de teste.

Dentre as abordagens consideradas, o *BERTimbau* apresentou desempenho elevado, alcançando um F_1-score de 99.0%, conforme mostrado na Tabela 4.2. A alta performance observada nas métricas, no entanto, pode ser parcialmente atribuída ao uso extensivo de dados sintéticos e à tradução automática dos dados reais. A presença predominante de *prompts* sintéticos pode ter simplificado a tarefa, resultando em uma detecção artificialmente facilitada. Além disso, a tradução dos dados originais do inglês para o português pode ter introduzido padrões estilísticos que simplificaram a identificação dos exemplos adversariais pelo modelo, inflacionando os resultados.

Já o módulo *DetectJailbreak*, descrito na Seção 3.3.1, apresentou desempenho insatisfatório. Durante a calibração do limiar, o valor ótimo obtido para maximizar o F_2-score foi 0.0, resultando na classificação de todos os *prompts* como ataques. Por outro lado, com o limiar padrão de 0.81, nenhum ataque foi identificado corretamente. Isso sugere que o módulo está mal adaptado à língua portuguesa e ao contexto específico da base utilizada, reduzindo sua aplicabilidade prática.

O *framework NeMo* da NVIDIA também foi considerado, mas acabou descartado devido à baixa eficiência computacional. Sua necessidade de consulta frequente a uma LLM externa, combinada com a execução local e heurísticas complexas de perplexidade por trecho, tornou inviável sua integração no projeto, já que seu processo de classificação era muito lento.

Em resumo, apesar das limitações mencionadas, o modelo baseado no *fine-tuning* do *BERTimbau* demonstrou-se o mais eficaz e viável para ser integrado à *pipeline* da aplicação. No entanto, é importante reconhecer que os resultados obtidos podem estar inflacionados pelos fatores discutidos, e que uma validação adicional com conjuntos de dados mais realistas seria necessária para confirmar o desempenho em um cenário operacional.

Modelo	**Precisão**	**Recall**	**F1-score**
BERTimbau	99.0%	98.0%	99.0%

Tabela 4.2: Métricas obtidas na detecção de *jailbreak*.

[1]<https://huggingface.co/datasets/JailbreakV-28K/JailBreakV-28k>

4.2.3 Dados Sensíveis

Para avaliar a eficácia da *guardrail* de dados sensíveis, foi construída uma base de dados sintética composta por 320 *prompts*, dos quais 160 continham dados que deveriam ser anonimizados e 160 não continham. A base foi projetada para abranger diferentes cenários, incluindo dados válidos e inválidos, formatados e não formatados, com e sem contexto textual, além de exemplos que não envolvem qualquer dado sensível. Essa diversidade foi essencial para avaliar o comportamento da solução frente às variações linguísticas, estruturais e contextuais presentes na comunicação natural.

Os testes indicaram que a *guardrail* apresenta desempenho satisfatório, considerando o escopo atual de entidades sensíveis implementadas. As métricas obtidas foram 76% de precisão, 78% de *recall* e 77% de F_1-score, conforme apresentado na Tabela 4.3. Esses valores refletem uma performance consistente, embora ainda haja espaço para melhorias, especialmente na redução de falsos positivos em casos ambíguos e falsos negativos em situações com baixo contexto semântico.

É importante destacar que os testes foram conduzidos apenas sobre tipos de dados que a *guardrail* foi projetada para capturar, como CPF, CNPJ, RG, números de telefone, cartões de crédito, contas bancárias, datas de nascimento e endereços de e-mail. Informações sensíveis mais complexas, como nomes próprios, endereços residenciais, dados contratuais, informações financeiras específicas ou chaves de API, não foram incluídas nos testes, pois ainda não estão cobertas pela atual implementação da *guadrail*.

O desempenho pode ser aprimorado com a expansão do escopo da *guardrail*, incorporando novas entidades sensíveis e aumentando a sofisticação dos mecanismos de detecção contextual. Além disso, a criação de bases de dados maiores e mais representativas — com variações linguísticas reais, múltiplos formatos e maior diversidade de contexto — será fundamental para melhorar a capacidade de generalização da solução. Estratégias como o uso de modelos treinados especificamente para *Named Entity Recognition (NER)* em português (SOUZA; NOGUEIRA; LOTUFO, 2019), combinados com regras baseadas em padrões estruturais, também podem contribuir para aumentar a cobertura e a precisão do sistema.

Framework	**Precisão**	**Recall**	**F1-score**
Presidio	76%	78%	77%

Tabela 4.3: Métricas obtidas na detecção de dados sensíveis com base sintética.

4.2.4 Escopo e Alucinação

Nesta seção são apresentados os resultados obtidos na avaliação das *guardrails* relacionadas à contenção de escopo e à mitigação de alucinações. Inicialmente são abordados os resultados referentes ao nível temático, avaliando a capacidade dos modelos em classificar corretamente se as perguntas

dos usuários estão dentro ou fora do escopo geral estabelecido para a aplicação. Em seguida, são descritas as avaliações do nível documental, destacando primeiro a eficácia na restrição das respostas aos conteúdos específicos indexados na base documental e, posteriormente, na mitigação direta de alucinações.

Nível Temático - Escopo Geral

A avaliação da *guardrail* de escopo temático foi conduzida em três domínios distintos: setor bancário, telecomunicações e varejo eletrônico. Os modelos alcançaram resultados consistentes em todos os cenários, com acurácias variando entre 96% e 99%, conforme apresentado na Tabela 4.4.

Esse desempenho está diretamente relacionado à construção das bases de dados utilizadas no processo de *fine-tuning* (HOWARD; RUDER, 2018). As bases foram geradas de forma controlada, com separação explícita entre exemplos dentro e fora de escopo, e com número reduzido de casos ambíguos. A distinção clara entre os exemplos contribuiu para a alta performance dos classificadores, facilitando a detecção de mensagens dentro e fora do escopo pretendido.

No entanto, é importante observar que esses resultados não necessariamente se generalizam para ambientes de produção, onde os dados tendem a apresentar ruído, ambiguidade e variações linguísticas e contextuais mais complexas. A qualidade das métricas obtidas está condicionada à natureza simplificada dos dados utilizados na avaliação.

Para melhorar o comportamento da *guadrail* em cenários reais, será necessário ampliar a diversidade e complexidade das bases de dados, incorporando exemplos mais ambíguos, interações reais e casos de fronteira. Adicionalmente, a inclusão de mecanismos para detecção de ambiguidade e calibragem dinâmica dos *thresholds* pode contribuir para manter o equilíbrio entre *precisão* e *recall* em contextos menos controlados.

Domínio	**Precisão**	**Recall**	**F1-score**
Setor Bancário	96%	96%	96%
Telecomunicações	99%	99%	99%
Varejo Eletrônico	97%	97%	97%

Tabela 4.4: Métricas obtidas na detecção de escopo temático por domínio.

Nível Documental — Escopo Específico

A avaliação da *guardrail* documental voltada à restrição de escopo teve como objetivo verificar a capacidade do sistema em bloquear perguntas cuja resposta não pudesse ser fundamentada nos documentos indexados. Para isso, foi utilizado um conjunto de perguntas sintéticas balanceadas, composto por exemplos relacionados ao conteúdo dos documentos e por exemplos propositalmente

formulados para estarem fora desse escopo. Após a rotulagem, os dados foram divididos de forma estratificada entre os conjuntos de treino e teste.

O conjunto de treino foi utilizado para o ajuste do limiar de decisão, por meio da maximização do F_1-*score* com base na curva de precisão e revocação. Como resultado desse processo, foi identificado o valor de threshold ótimo igual a -4.2235. Já o conjunto de teste foi reservado exclusivamente para a avaliação final da *guadrail*, assegurando que as métricas observadas não estivessem influenciadas por dados previamente utilizados no ajuste.

Durante a inferência, o sistema integrou mecanismos de recuperação semântica com FAISS (DOUZE et al., 2024), busca lexical com *BM25* (ROBERTSON; ZARAGOZA, 2009) e reranqueamento por similaridade textual com um modelo *Cross-Encoder*. A decisão de bloqueio foi tomada com base na maior pontuação de similaridade obtida, que era então comparada ao limiar calibrado.

Os resultados obtidos no conjunto de teste demonstraram que a *guadrail* foi eficaz na identificação de perguntas fora de escopo, conforme detalhado na Tabela 4.5. Os poucos erros observados ocorreram em perguntas que, apesar de apresentarem forte similaridade lexical com os documentos, não possuíam correspondência semântica real com o conteúdo. Ainda que o reranqueamento tenha sido aplicado como etapa final de validação, casos como esses indicam limitações do modelo utilizado ou ambiguidade nos exemplos. Esse comportamento sugere que, para contextos menos controlados, pode ser necessário empregar modelos de reranqueamento mais sofisticados ou reforçar o processo de curadoria dos dados documentais.

Classe	**Precisão**	**Recall**	**F1-score**
Fora do escopo específico	99%	87%	93%

Tabela 4.5: Métricas obtidas na detecção de perguntas fora de escopo no nível documental.

Nível Documental — Mitigação de Alucinação

A avaliação da *guardrail* de nível documental voltada à mitigação de alucinações, foi conduzida a partir de perguntas cuja resposta era conhecida e presente nos documentos indexados. Para isso, foi utilizado um conjunto previamente validado de perguntas geradas com base nos próprios documentos, garantindo que cada pergunta estivesse claramente associada a um trecho específico do conteúdo.

Durante a avaliação, cada pergunta foi processada pela *guardrail* para recuperação dos trechos mais relevantes (*chunks*). Em seguida, cada pergunta foi enviada duas vezes à LLM: uma vez com os trechos recuperados (contexto) e outra sem nenhum contexto adicional. Isso permitiu gerar duas respostas distintas para a mesma entrada — uma resposta com contexto documental e outra sem. A Figura 4.3 ilustra esse processo.

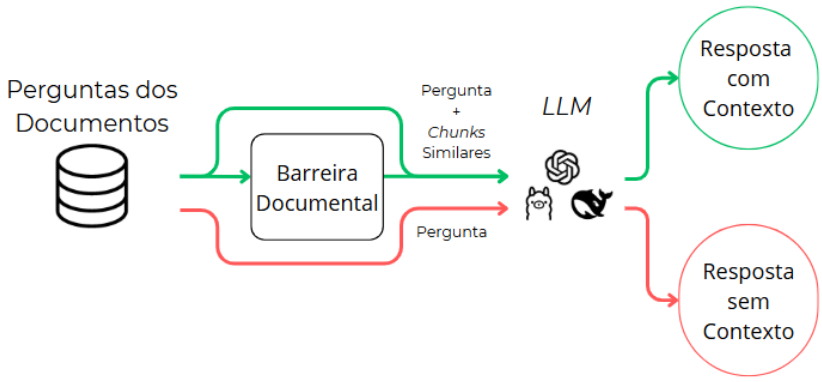

Figura 4.3: Fluxo de avaliação da guardrail de escopo documental

Com base nas respostas geradas, foi realizada a comparação entre as duas versões com o *chunk* original que gerou a pergunta dessas respostas, utilizando a métrica de similaridade de cossenos entre embeddings, gerados por meio do modelo *intfloat/multilingual-e5-large* (WANG et al., 2024). A hipótese era de que, se a *guardrail* estivesse funcionando corretamente, a resposta com contexto seria mais próxima da resposta ideal (isto é, do conteúdo original do documento), enquanto a resposta sem contexto tenderia à alucinação.

Contudo, observou-se que as diferenças semânticas entre as respostas geradas com e sem contexto nem sempre se refletiam nos valores de similaridade vetorial. Mesmo quando havia divergência clara no conteúdo, as pontuações de similaridade permaneciam próximas, tornando a métrica insuficiente. Diante disso, a análise assumiu caráter predominantemente qualitativo, fundamentada na comparação direta entre as respostas geradas e os trechos de referência.

Um exemplo concreto reforça essa limitação: durante os testes, utilizou-se o modelo GPT-4o (*OpenAI*[2]) para responder à pergunta "Quem é o presidente atual do Banco Central?". Quando fornecido o contexto afirmando que Gabriel Galípolo ocupa o cargo desde janeiro de 2025, o modelo respondeu corretamente. No entanto, ao ser questionado sem qualquer contexto adicional, a resposta gerada foi "Roberto Campos Neto", refletindo dados desatualizados do período de pré-treinamento. Apesar de semanticamente próximas, as respostas diferiam substancialmente em precisão factual, e a métrica de similaridade vetorial não foi capaz de distinguir essa diferença crítica.

Esse comportamento evidencia o papel central do contexto documental na mitigação de alucinações. Ao disponibilizar informações atualizadas e relevantes diretamente à LLM, evita-se a recorrência a conteúdos genéricos ou desatualizados aprendidos durante o pré-treinamento. Assim, mesmo em tarefas sensíveis à temporalidade e à precisão factual — como no exemplo citado — o fornecimento de contexto atua como um mecanismo de ancoragem, promovendo maior aderência ao domínio e confiabilidade nas respostas geradas.

[2]<https://openai.com>

4.2.5 Competidores

A *guardrail* de competidores foi avaliada utilizando um conjunto de 397 exemplos balanceados entre mensagens permitidas e mensagens com menções indevidas a instituições concorrentes. O modelo alcançou um F_1-*score* de 97% na detecção de menções indevidas, conforme apresentado na Tabela 4.6.

Os resultados refletem a natureza simplificada da base de dados, construída de forma sintética, com separação clara entre exemplos positivos e negativos. As menções indevidas são majoritariamente diretas e explícitas, enquanto os exemplos permitidos são neutros e livres de ambiguidade. Esse fator reduz a complexidade da tarefa, favorece as métricas e não captura integralmente a complexidade linguística e pragmática de interações reais.

Apesar do desempenho satisfatório, é importante ressaltar que esses resultados se limitam ao contexto controlado dos dados de teste. O comportamento do modelo em ambientes reais dependerá da expansão da base de dados com exemplos mais diversos, contendo ambiguidade, ruído e construções linguísticas mais naturais.

Classe	**Precisão**	**Recall**	**F1-score**
Menções Indevidas	99%	94%	97%

Tabela 4.6: Métricas obtidas na detecção de menções indevidas a concorrentes.

5 Conclusão

Este projeto demonstrou a viabilidade técnica de construir um sistema funcional de *guardrails* para IA generativa, capaz de mitigar riscos associados a *jailbreak*, toxicidade, dados sensíveis, escopo, alucinação e menções indevidas a concorrentes. A abordagem adotada, fundamentada em uma arquitetura modular, permitiu combinar diferentes estratégias, como classificação supervisionada, RAG (LEWIS et al., 2020) e detecção baseada em padrões estruturais.

Os resultados obtidos indicam que a solução funciona de maneira consistente em cenários controlados. No entanto, ainda apresenta limitações relevantes quando aplicada a ambientes produtivos. O desempenho geral, embora satisfatório, foi dependente de conjuntos de dados relativamente pequenos e, em alguns casos, sintéticos. Isso afeta diretamente a capacidade de generalização dos modelos, principalmente nas *guardrails* de escopo, concorrência e *jailbreak*, onde há maior diversidade linguística, semântica e estrutural nos dados reais.

A *guardrail* de dados sensíveis, embora funcional, possui escopo limitado, focado em poucas entidades pré-definidas. A expansão para tipos de dados mais complexos e contextuais — como informações financeiras, operacionais, chaves de acesso e outras entidades críticas — é necessária para alcançar cobertura compatível com ambientes corporativos.

Outro ponto crítico identificado refere-se à ausência de controle adaptativo na aplicação das *guardrails*. Atualmente, todas as *guadrails* são ativadas de forma fixa, sem levar em conta o contexto, o domínio da interação ou o perfil de risco envolvido. Esse comportamento pode gerar restrições excessivas em situações de baixo risco e, inversamente, falhas de contenção em contextos mais sensíveis. A introdução de um orquestrador dinâmico, capaz de ativar ou desativar as *guadrails* com base nesses fatores, constitui um passo fundamental na evolução da arquitetura.

A literatura recente reforça esse direcionamento. Xu et al. (XU; WANG; CUI, 2024) propõem o conceito de *Trust-Oriented Adaptive Guardrails*, que define essas *guadrails* não apenas como mecanismos de restrição, mas como sistemas adaptativos, sensíveis ao contexto e orientados à construção de confiança. Essa abordagem busca manter um equilíbrio mais sofisticado entre segurança, controle e flexibilidade, ajustando o rigor das restrições com base nos riscos percebidos e no cenário de operação. Tal proposta está alinhada aos desafios observados neste trabalho e aponta um caminho claro para sua continuidade.

Além da adoção de um orquestrador, destaca-se a necessidade de ampliação das bases de

dados para treinamento, calibração e avaliação dos modelos. A inclusão de dados mais diversos, com variações linguísticas, contextuais e semânticas representativas de interações reais, é essencial para reduzir falsos positivos e negativos e melhorar o comportamento das *guadrails* implementadas.

Finalmente, o aprimoramento das técnicas de mitigação de alucinação é um eixo central para o avanço do sistema. Isso inclui tanto o fortalecimento dos mecanismos baseados em RAG (LEWIS et al., 2020) quanto a integração com verificadores factuais externos, bases de conhecimento atualizadas e validação cruzada das respostas antes de sua entrega ao usuário.

Com esses avanços, o sistema poderá evoluir de um protótipo acadêmico funcional para uma solução robusta, escalável e confiável, apta a atender às exigências operacionais, regulatórias e de segurança de ambientes corporativos que integram IA generativa em aplicações críticas.

Referências

AYYAMPERUMAL, S. G.; GE, L. *Current State of LLM Risks and AI Guardrails*. 2024. ArXiv preprint. Disponível em: <https://arxiv.org/abs/2406.12934>.

BONIFACIO, L. H. et al. *mMARCO: A Multilingual Version of MS MARCO Passage Ranking Dataset*. 2021.

DEVLIN, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. Disponível em: <https://arxiv.org/abs/1810.04805>.

DONG, Y. et al. *Building Guardrails for Large Language Models*. 2024. ArXiv preprint. Disponível em: <https://arxiv.org/abs/2402.01822>.

DOUZE, M. et al. The faiss library. *arXiv eprint arXiv:2401.08281*, 2024.

FALCON, W.; TEAM, T. P. L. *PyTorch Lightning*. 2025. <https://www.pytorchlightning.ai>. Accessed: 2025-05-23.

Guardrails AI. *DetectJailbreak – Documentação oficial*. 2024. GitHub. Disponível em: <https://github.com/guardrails-ai/detect_jailbreak>.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [s.n.], 2018. p. 328–339. Disponível em: <https://aclanthology.org/P18-1031/>.

HUANG, L. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems (TOIS)*, 2024. Disponível em: <https://arxiv.org/abs/2311.05232>.

Hugging Face. *Transformers Documentation*. 2024. Disponível em: <https://huggingface.co/docs/transformers/index>.

HUSSEIN, K. W. et al. Enhance luhn algorithm for validation of credit cards numbers. *International Journal of Computer Science and Mobile Computing*, v. 2, n. 7, p. 262–272, 2013. Disponível em: <https://ijcsmc.com/docs/papers/July2013/V2I7201373.pdf>.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing*. 3rd. ed. [s.n.], 2023. Draft, available online. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020. Disponível em: <https://arxiv.org/abs/2005.11401>.

LI, K. et al. Measuring and controlling persona drift in language model dialogs. In: *Proceedings of the International Conference on Machine Learning (ICML)*. Harvard University, Northeastern University: [s.n.], 2024. Disponível em: <https://arxiv.org/pdf/2402.10962v1>.

LIU, Y. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. ArXiv preprint. Disponível em: <https://arxiv.org/abs/1907.11692>.

MICROSOFT. *Presidio: Microsoft's Open Source Project for PII and Sensitive Data Detection and Anonymization*. 2024. GitHub. Disponível em: <https://github.com/microsoft/presidio>.

MICROSOFT. *Presidio: An open-source framework for detecting and anonymizing PII*. 2025. Disponível em: <https://github.com/microsoft/presidio>.

NVIDIA. *NeMo Guardrails*. 2025. NVIDIA Documentation. Disponível em: <https://docs.nvidia.com/nemo/guardrails/index.html>.

PYTORCH. *Binary Cross Entropy with Logits Loss*. 2024. PyTorch Documentation. Disponível em: <https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html>.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *Proceedings of EMNLP*, 2019. Disponível em: <https://arxiv.org/abs/1908.10084>.

ROBERTSON, S.; ZARAGOZA, H. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, Now Publishers Inc., v. 3, n. 4, p. 333–389, 2009. Disponível em: <https://www.nowpublishers.com/article/Details/INR-019>.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Portuguese named entity recognition using bert-crf. In: *Proceedings of the 7th Brazilian Conference on Intelligent Systems (BRACIS)*. [s.n.], 2019. p. 801–806. Disponível em: <https://arxiv.org/abs/1909.10649>.

SOUZA, F. C. d. *BERTimbau: pretrained BERT models for Brazilian Portuguese = BERTimbau: modelos BERT pré-treinados para português brasileiro*. Dissertação (Dissertação (Mestrado)) — Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas, SP, 2020. 1 recurso online (62 p.). Disponível em: <https://hdl.handle.net/20.500.12733/1640809>.

UNITARY. *Unbiased Toxic Roberta*. 2024. Hugging Face. Disponível em: <https://huggingface.co/unitary/unbiased-toxic-roberta>.

WANG, L. et al. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

WANG, W. et al. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [s.n.], 2020. Disponível em: <https://arxiv.org/abs/2002.10957>.

XU, W. et al. *Trust-Oriented Adaptive Guardrails for Large Language Models*. 2024. ArXiv preprint. Disponível em: <https://arxiv.org/pdf/2408.08959>.

XU, Z.; WANG, Z.; CUI, P. *A Comprehensive Study of Jailbreak Attack versus Defense for Large Language Models*. 2024. ArXiv preprint. Disponível em: <https://arxiv.org/abs/2402.13457>.