Felipe Batista Arrais
Igor Souza Lima e Silva Caixeta
Vinicius de Castro Lopes

Uso de Geração Aumentada de Recuperação para Mitigar Alucinações em Grandes Modelos de Linguagem

São Paulo, SP

2024

Felipe Batista Arrais
Igor Souza Lima e Silva Caixeta
Vinicius de Castro Lopes

Uso de Geração Aumentada de Recuperação para Mitigar Alucinações em Grandes Modelos de Linguagem

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP
Escola Politécnica
Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Prof[a]. Dr[a]. Anarosa Alves Franco Brandão
Coorientador: Dr. João Paulo Aragão Pereira

São Paulo, SP
2024

Felipe Batista Arrais
Igor Souza Lima e Silva Caixeta
Vinicius de Castro Lopes

Uso de Geração Aumentada de Recuperação para Mitigar Alucinações em Grandes Modelos de Linguagem

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Trabalho aprovado em:

Profª. Drª. Anarosa Alves Franco Brandão
Orientador

Dr. Felipe Valencia de Almeida
Convidado 1

Prof. Dr. Kechi Hirama
Convidado 2

São Paulo, SP
2024

Agradecimentos

Agradeço primeiramente aos meus pais, Humberto e Silvana, e também à toda minha família, por serem os alicerces que sustentaram com muito amor a trilha que percorri e a pessoa que me tornei. Agradeço também aos meus amigos, sem os quais eu não saberia ser eu mesmo: André Guedes, Lucas Rocha, Pedro Barão, Antônio Gimenez, Felipe Aparecido, Hector Fugihara, Paulo Mirabile, Maia Miller e Gabriel Stephano. Ademais, agradeço à nossa Orientadora, Anarosa Brandão, ao nosso Co-Orientador, João Paulo Aragão, e aos professores do curso de Engenharia de Computação, por suas orientações cujo valor não pode ser mensurado, mas representam muito para mim.

Felipe Batista Arrais

Gostaria de agradecer à minha família por todo apoio que recebi ao longo da graduação, aos meus pais, João Jacinto Caixeta Neto e Elaine de Souza Lima e Silva. Não apenas eles, mas minha irmã, Ingrid Souza Lima e Silva Caixeta, que também conviveu comigo em São Paulo e esteve presente comigo em momentos de dificuldade e felicidade. Para finalizar, agradecer à Professora Anarosa Brandão e ao Dr. João Aragão por todo o auxílio técnico fornecido para a produção desse trabalho e aos meus colegas de Trabalho de Conclusão de Curso, Felipe Batista Arrais e Vinícius de Castro Lopes.

Igor Souza Lima e Silva Caixeta

Deixo aqui meus agradecimentos, primeiramente e sobretudo, a Deus e a Nossa Senhora de Aparecida pelos dons concedidos a mim, o suporte durante os momentos difíceis e pela capacidade para concluir a graduação no curso de Engenharia de Computação da Escola Politécnica da Universidade de São Paulo. Em segundo lugar, aos meus pais, Maria Cristina de Castro e Helenicio Lopes da Costa, por me apoiarem durante minha trajetória e fornecerem a base e segurança necessárias não somente durante a graduação, mas também no decorrer de toda minha vida. E também de maneira muito importante, à Profa. Dra. Anarosa Alves Franco Brandão e ao Dr. João Paulo Aragão Pereira pela orientação neste projeto, aos professores que contribuíram para a minha formação acadêmica e aos colegas de curso que me acompanharam até o momento.

Vinicius de Castro Lopes

"A criação bem-sucedida de inteligência artificial
seria o maior evento na história da humanidade.
Infelizmente, pode também ser o último,
a menos que aprendamos a evitar os riscos"

(Stephen Hawking)

Resumo

Esse trabalho visa a estudar maneiras de mitigar o fenômeno das alucinações presentes em grandes modelos de linguagem (Large Language Models) através da implementação de uma arquitetura de Geração Aumentada de Recuperação (Retrieval Augmented Generation) e técnicas correlacionadas. Para o estudo de caso, utilizou-se conversas geradas e classificadas sinteticamente de centrais de atendimento bancárias, nas quais o contatante pode, ou não, estar tentando aplicar um golpe. A avaliação dos resultados foi feita de maneira qualitativa através de uma aplicação na qual o modelo de linguagem forneceu as respostas contextualizada e não-contextualizada, e buscou-se analisar tanto a resposta categórica (intenção fraudulenta, intenção não-fraudulenta) quanto o desvio da probabilidade atribuído à conversa.

Palavras-chave: Engenharia de Computação, Linguagem Natural, Análise de Texto, Discurso e Tratamento de Diálogos, Inteligência Artificial.

Abstract

This work aims to study ways to mitigate the phenomenon of hallucinations present in large language models (LLMs) through the implementation of an architecture based on Retrieval Augmented Generation and correlated techniques. For the case study, synthetically generated and classified conversations from bank call centers were used, in which the caller may or may not be trying to carry out a scam. The evaluation of the results was carried out qualitatively through an application in which the language model provided contextualized and non-contextualized responses, and we sought to analyze both the categorical response (fraudulent intention, non-fraudulent intention) and the deviation from the probability assigned to the conversation.

Keywords: Computer Engineering, Natural Language Processing, Text Analysis, Speech and Dialogue Treatments, Artificial Intelligence.

Lista de ilustrações

Lista de abreviaturas e siglas

IA	Inteligência Artificial
LLM	Large Language Model
RAG	Retrieval Augmented Generation
RL	Reinforcement Learning
NLG	Geração de Linguagem Natural
NLP	Processamento de Linguagem Natural

Sumário

1 Introdução

Com a grande aprimoração de tecnologias associadas a Inteligência Artificial (IA), ferramentas de IA generativa têm ganhado enorme destaque em anos recentes, tanto em termos das capacidades delas, quanto em popularidade. Em decorrência desse fenômeno, o uso de LLMs tem sido notavelmente crescente – como exemplo disso, o *website* de conversa com o ChatGPT, modelo de linguagem da empresa OpenAI, levou apenas cinco dias desde sua criação para atingir a marca de 1 milhão de usuários.

O desenvolvimento dessas tecnologias acarretou numa maior fama de produtos como ChatGPT e Bing Chat, amplamente notados por serem capazes de produzir textos fluentes e coerentes sobre diversos temas. Entretanto, conforme LLMs são mais utilizadas, mais se torna aparente o risco de que a informação dada como resposta pelo modelo usado esteja incorreta ou imprecisa, fenômeno conhecido como "alucinação".

Tendo em vista as crescentes capacidades de tais modelos de linguagem e sua consequente aplicabilidade em cenários como chatbots empresariais e suporte a programadores, a confiança nas respostas dadas por LLMs se torna um requisito essencial nessas aplicações, uma vez que invalidade nas informações retornadas ao usuário danifica completamente a utilidade desses serviços. Por conseguinte, a ocorrência de alucinações demonstra-se como um obstáculo de enorme impacto, ao ser responsável pelas falhas dos modelos de linguagem que não foram detectadas por eles mesmos.

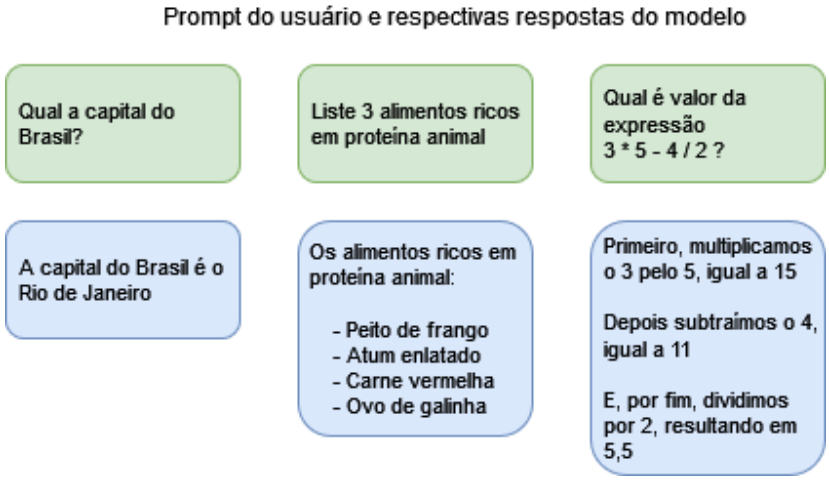

Figura 1 – Exemplo de alucinações podem ocorrer em Large Language Models. Em verde estão os prompts do usuário ao modelo; em azul, as suas respectivas respostas (ilustrativas)

1.1 Motivação

Nos últimos anos, o mundo foi surpreendido pelo rápido avanço das inteligências artificais e das LLMs. Perguntas que antes exigiam pesquisas em navegadores de internet e leitura de páginas foram diminuídas à perguntas diretas com respostas mais direcionadas.

Assim como os gregos da antiguidade clássica buscavam sanar suas dúvidas e questionamentos em oráculos, como o famoso oráculo de Delfos, a sociedade atual anseia por respostas mais rápidas para perguntas mais específicas. Seguindo este paralelo, assim como os antigos oráculos poderiam proferir ideias e tomadas de decisões equivocadas, as LLMs também podem se emaranhar e expressar ideias ou frases confusas e sem sentido, ou até mesmo erradas.

1.2 Objetivos

O objetivo deste trabalho de pesquisa é estudar a aplicação de técnicas centradas na Geração Aumentada de Recuperação (Retrieval Augmented Generation) na mitigação do problema das alucinações que ocorrem nos grandes modelos de linguagem (Large Language Models), de maneira que, ao melhorar a confiabilidade nas respostas geradas, as aplicações que utilizem tais modelos possam entregar maior valor aos seus usuários.

Para essa finalidade, será desenvolvida uma aplicação de chatbot pela qual será possível a interação entre o modelo utilizado e o usuário. Este será, assim, capaz de comparar respostas dadas por um modelo preparado com RAG adicional e um sem essa tecnologia, para que seja consolidada uma análise qualitativa do resultado da RAG. Visando a esse fim, decidiu-se utilizar o modelo LlaMa 2, um dos modelos de código aberto mais consolidados atualmente.

Com o fim de delimitar o escopo da pesquisa, o contexto a ser trabalhado durante o projeto será o mundo financeiro, isto é, aplicações bancárias que utilizem os LLMs para gerar informações aos seu usuários. Mais especificamente, será abordada a questão das fraudes em transações financeiras que utilizem cartão de crédito. Assim, tal como é feito em modelos de aplicações específicas, será feito um refino controlado do LlaMa de acordo com informações próprias do domínio bancário.

1.3 Justificativa

Dada a expressiva habilidade de compreensão e geração de conteúdo de LLMs, combinada à rápida popularidade adquirida por essa tecnologia, é possível identificar uma ampla gama de possíveis aplicações para os grandes modelos de linguagem. Dentre elas, pode-se listar não apenas chatbots de propósito geral como o ChatGPT previamente

mencionado, mas também usos como sumarização e tradução de textos, assistentes virtuais e chatbots empresariais, análise textual de sentimentos, e auxílio no desenvolvimento de código.

Portanto, a possibilidade de LLMs produzirem conteúdo sem sentido ou com informações imprecisas ou inválidas é uma realidade que prejudica o avanço de uma tecnologia verdadeiramente impactante. Diante dessa conjuntura, a redução das alucinações pode ajudar a viabilizar produtos capazes de automatizar diversos processos de negócio, e evitar que os mesmos causem algum prejuízo advindo de desinformação ou dados inadequados.

1.4 Organização do Trabalho

Após o introdutório capítulo 1, o capítulo 2 apresenta de forma sucinta os principais conhecimentos necessários para o trabalho, além das tecnologias utilizadas como base do projeto. No capítulo 3, é descrito qual foi o método de trabalho utilizado, apresentando a revisão da literatura, a obtenção dos dados sintéticos, decisões de projeto e planejamento de implementação.

No capítulo 4, é detalhada a geração do dataset, mostrando o método de obtenção dos dados para realizar o RAG, além de detalhar seu armazenamento e a forma que ele é acessado pelo sistema. O capítulo 5 exemplifica a arquitetura do sistema e como o modelo pode ser instanciado. Já no 6, é detalhado o sistema do trabalho, exibindo o levantamento de requisitos, a arquitetura web e os testes da implementação. Por fim, o capítulo 7 são as conclusões retiradas a partir deste estudo.

2 Aspectos Conceituais e Tecnologias Utilizadas

2.1 Grandes Modelos de Linguagem

O termo "modelos grandes de linguagem", também chamado de "modelos largos de linguagem" (MLL), tem como origem a expressão em inglês *Large Language Models* (LLM), e esta é usada para definir um tipo de modelo de linguagem mais recente. Modelos de linguagem são modelos de inteligência artificial feitos para processar linguagem natural e gerar uma resposta, comumente na forma de linguagem natural também. Para isso, dispõem de técnicas como redes neurais, aprendizado por reforço e aprendizado profundo, e passam por um treinamento com uma base de dados que os permitam aprender sobre o domínio da aplicação à qual eles servirão.

No caso dos grandes modelos de linguagem, a base usada para a etapa de treinamento é notavelmente grande, geralmente mensurada na ordem de dezenas ou centenas de bilhões de parâmetros. Como exemplo, o modelo GPT-3, da empresa OpenAI, foi treinado com 175 bilhões de parâmetros (BROWN et al., 2020). Devido a essas dimensões expressivas, LLMs têm se mostrado capazes de compreender conceitos e realizar tarefas de diversos campos de conhecimento, acarretando na rápida popularização de aplicações como ChatGPT e o Bing Chat. Um uso frequente de tais ferramentas é responder perguntas técnicas, que exemplifica a capacidade delas.

Por fim, vale ressaltar que o avanço dessa tecnologia não altera o fato de que funcionam de forma puramente reativa – ou seja, a saída de uma execução de um LLM é sempre a partir de um comando dado, estando tal saída correta ou não. Por isso, caso haja algum erro no comportamento do modelo, não basta analisar a qualidade dos dados do treinamento ou a forma como o modelo foi construído, mas também averiguar a validade da instrução dada à aplicação do LLM.

2.2 Alucinações

Dentro do contexto da Geração de Linguagem Natural (NLG) – no qual se incluem os LLMs – as alucinações são definidas como um fenômeno que ocorre quando o conteúdo gerado pelo modelo é considerado absurdo ou infiel com relação à entrada fornecida.(FILIPPOVA, 2020). No caso em que o conteúdo é considerado absurdo, refere-se a relação do conteúdo com os fatos reais.

Este fenômeno não tem origem única, e entre suas causas pode-se destacar (HUANG

et al., 2023):

- Os dados utilizados durante o pré-treinamento: frequentemente esses dados contêm informações incorretas, viéses sociais e limite de conhecimento em domínios específicos, características essas presentes em um imenso volume de dados;
- As fases de treinamento e ajustes aos quais o modelo é submetido também podem induzir a ocorrência das alucinações. Problemas como representação unidirecional, falhas nas arquiteturas e desalinhamento de capacidade e crença podem surgir durante esses estágios;
- A qualidade da entrada fornecida pelo usuário também pode ser um fator gerador de alucinações, pois o contexto da instrução pode estar incompleto ou impreciso.

Em um vasto estudo a respeito do fenômeno das alucinações (HUANG et al., 2023), elas foram divididas em dois tipos principais: alucinações de fatos e alucinações de fidelidade.

2.2.1 Alucinações de fatos

Nas alucinações de fatos, as respostas geradas pelo modelo LLM são, de alguma maneira, inconsistentes com os fatos do mundo real ou potencialmente enganosas, comprometendo a confiabilidade da inteligência artificial. Esse tipo de alucinação ocorre devido ao extenso volume de informações com as quais os LLMs são treinados e ao seu amplo conhecimento paramétrico, e é subdividido em dois tipos:

- **Inconsistência de fatos:** ocorre quando a saída gerada pelo modelo contradiz fatos conhecidos da realidade, e pode-se verificar a ocorrência da alucinação através de fontes confiáveis;
- **Fabricação de fatos:** nesse subtipo a saída gerada pelo modelo contém afirmações que não podem ser provadas através do conhecimento real. Por exemplo, se usuário requisitasse ao modelo "Explique-me a origem dos dragões", este poderia alucinar fabricando afirmações que levassem a crer na real existência destes seres.

2.2.2 Alucinações de fidelidade

LLMs têm sido cada vez mais utilizados para aplicações centradas no usuário, por isso é imprescindível que a saída gerada corresponda às instruções fornecidas e a informação contextual que elas possuem. Além do mais, a fidelidade da saída gerada também é medida através da sua lógica interna.

Dito isto, as alucinações de fidelidade são subdividas em três subtipos:

- **Inconsistência com instrução:** ocorre quando a resposta gerada pelo modelo não segue a instrução do usuário – considerando que a instrução não contradiga diretrizes de segurança e possua contexto suficientemente definido.
- **Inconsistência contextual:** ocorre quando a resposta gerada pelo modelo contradiz o contexto da entrada fornecida pelo usuário.
- **Inconsistência lógica:** esse tipo de alucinação é observada principalmente em tarefas que envolvem raciocínio. Pode se manisfetar tanto entre os passos intermediários do raciocínio, quanto entre os passos e a conclusão final.

2.3 Fraudes em transações financeiras

As fraudes em transações financeiras é um fenômeno complexo que afeta tanto instituições bancárias quanto indivíduos consumidores. Caracterizada pela manipulação fraudulenta de informações, dados ou recursos financeiros com o intuito de obter vantagens indevidas, a fraude representa uma ameaça significativa para a integridade das empresas e a confiança dos consumidores. Elas também podem envolver esquemas mais complexos, como lavagem de dinheiro e manipulação de mercado.

Enquanto técnicas como mecanismos de análises de dados e aprendizado de máquina são comumente empregados para identificar e então prevenir tais fraudes, elas não abordam os meios pelos quais as fraudes ocorrem. Entre eles, é dado um destaque para ligações telefônicas via central de atendimento, nas quais o cliente ou o agente do banco podem, na verdade, ser um agente malicioso personificando outra pessoa para conseguir dados bancários alheios.

2.3.1 Dados Sintéticos

Após a popularização do ChatGPT, aplicações que se valem de modelos de inteligência artificial – em especial, os Grandes Modelos de Linguagem – se popularizaram na sociedade em diversos domínios de aplicação (LIU et al., 2024). Esses domínios abarcam aplicações médicas, sistemas de recomendação de conteúdo (ou produtos), questões envolvendo a segurança de sistemas, entre outros. Contudo, diversos casos de uso enfrentam a dificuldade recorrente da escassez de dados necessários para a adaptação – ou refinamento – do modelo.

Entre os fatores que se podem citar como razão dessa escassez de dados, tanto as características próprias do domínio em particular, quanto questões de sigilo e privacidade de dados (ABAY et al., 2019) considerados sensíveis são aqueles que mais se destacam.

Nesse sentido, a geração de dados sintéticos vem como uma solução para o problema da carência de dados (NIKOLENKO, 2019), fornecendo dados que buscam imitar as

características de dados reais para adaptar os modelos de inteligência artificial aos muitos casos de uso que enfrentam esse fenômeno. Esses dados podem ser gerados por diferentes técnicas, seja através da variação a partir de dados pré-existentes, seja através de modelos generativos altamente robustos. No escopo desse trabalho – onde os dados serão utilizados como contexto adicional para cada prompt feito – escolheu-se gerar os dados a partir do ChatGPT, uma aplicação de um LLM robusto e com vasto conhecimento adquirido a partir de diversas fontes (BROWN et al., 2020).

Para diversas tarefas de Processamento de Linguagem Natural, entre elas classificação de texto, gerar dados em modelos mais robustos para alimentar modelos menos poderosos pode ser uma alternativa para fornecer o volume de dados necessário para o caso de uso visado.

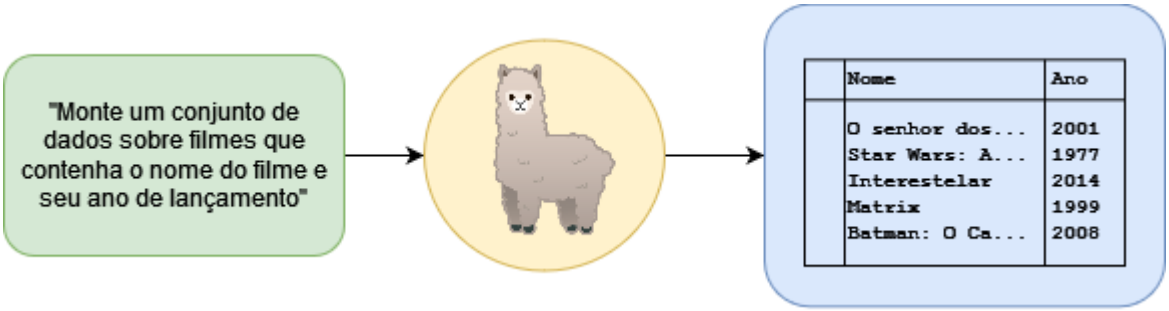

Figura 2 – Exemplo de geração de dados sintéticos através de um LLM

2.4 Geração Aumentada de Recuperação

Geração Aumentada de Recuperação, ou RAG, é uma técnica de refinamento usado em LLMs que busca aumentar o desempenho delas em tarefas que exigem maiores quantidades de conhecimento em uma área específica (LEWIS et al., 2021). Ela consiste no acréscimo de um componente recuperador ("Retriever") no início do processamento feito pelo modelo. Esse componente recebe a entrada feita pelo usuário e a utiliza para consultar um conjunto vetorizado de dados, o Vector Store. Em seguida, ele repassa a entrada do usuário junto dos dados obtidos para o LLM, para que ele possa então gerar uma resposta.

Por meio dessa técnica, é possível representar uma consulta a uma base de conhecimento de acordo com o que o usuário pede, de forma análoga a um humano consultando informações consolidadas conforme sua necessidade ao realizar uma tarefa que precise delas.

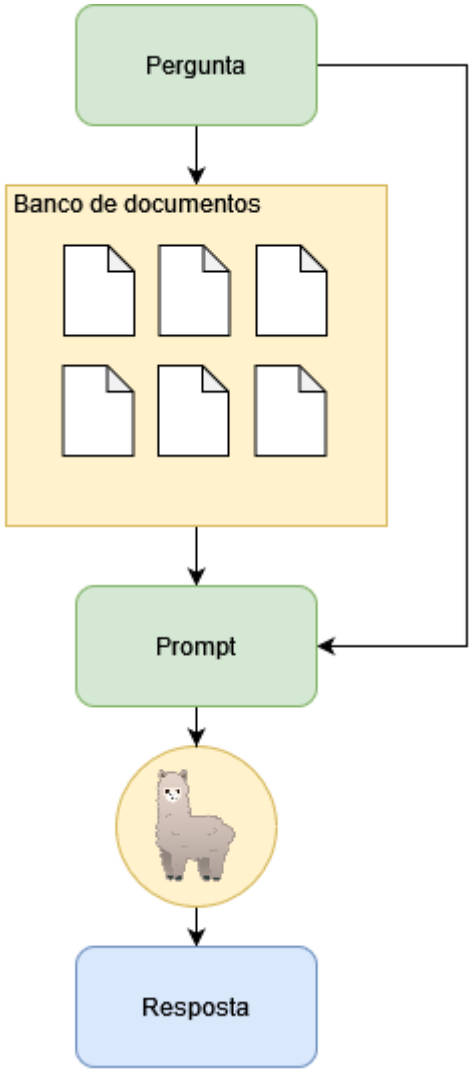

Figura 3 – Fluxo de informação na Geração Aumentada de Recuperação

2.5 Embeddings

Embeddings são representações matemáticas de informações expressas por palavras ou frases. Cada embedding, cujo significante correspondente depende do tipo de embedding armazenado, é expresso como um vetor em um espaço multidimensional, de tal forma que conceitos de significados próximos também apresentam vetores (embeddings) próximos no espaço armazenado (GARG et al., 2018) (WU et al., 2023). Assim, embeddings guardam informações de não apenas as palavras em si, mas a semântica por trás de cada uma. Por isso, o uso deles é fundamental no contexto de LLMs, uma vez que eles conseguem representar o significado por trás de significantes da linguagem natural de forma que pode ser interpretada pelos modelos que os utilizam.

2.6 Busca por similaridade

A implementação de aplicações que utilizam a técnica da Geração Aumentada de Recuperação exige que armazenemos os documentos a serem utilizados como fontes de informações para as consultas feitas ao modelo de linguagem. Para isso, é necessário implementar algoritmos e técnicas que permitam selecionar os documentos que mais se aproximem de cada consulta.

Nisto entra a busca por similaridade, uma técnica que busca por pontos de similaridade entre os diferentes pontos de dados. É necessário, porém, executar esse técnica de maneira eficiente através de algoritmos como quantização de produto ou grafos de vizinho mais próximo , uma vez que buscas exautisvas podem tornar o processo demasiadamente custoso computacionalmente (WU et al., 2023), por fim diminuindo o desempenho das aplicações que implementem esta técnica.

Ao tratarmos de dados representados na forma de vetores multidimensionais – embeddings, a distância vetorial pode ser usada como fator aproximador e facilitar a busca por informações que sejam relevantes a determinados contextos.(HJALTASON; SAMET, 2003). Esse tipo de técnica é importante principalmente nas técnicas de aprendizado de máquina modernas.

2.7 Classificação de Texto

Nos estudos sobre Inteligência Artificial, classificação de texto é uma das principais finalidades da área de NLP, pois exemplifica uma tarefa que apesar de ser muito comum para humanos, é desafiadora de ser feita por softwares de computação convencional (ou seja, alheios às técnicas de inteligência artificial). Com o advento de IA, esse tipo de categorização se tornou gradativamente mais viável, e comumente aplicada para reconhecimento de *spam* e buscas textuais semânticas, por exemplo.

2.8 Hugging Face

O Hugging Face é uma plataforma na qual estudantes, profissionais e entusiastas em Inteligência Artificial e Ciência de Dados podem compartilhar seus modelos e *datasets* de maneira aberta, além de hospedar seus modelos para que outros usuários possam experimentar suas capacidades. A plataforma possui bibliotecas próprias (como a "transformers"), implementadas sobre a linguagem de programação Python, com as quais se pode acessar os modelos e datasets disponíveis. Através dessas bibliotecas, será feito o fine-tuning do LlaMa 2 para que se adeque ao contexto de transações fraudulentas com cartões de crédito.

Além disso, a hospedagem no Hugging Face é de fácil integração com o framework LangChain, que será utilizado no desenvolvimento dos modelos envolvidos no projeto.

2.9 LangChain

LangChain é um framework Python para o uso de modelos de linguagem, como os modelos LlaMa, com foco na integração deles com outras fontes de informações, como bases de dados, ferramentas externas e outros modelos. Essa ferramenta permite a construção de fluxos de processamento de instrução mais complexos do que aquilo que é viável de fazer apenas com a interação direta com um modelo, por meio de uma estrutura modular que forma "correntes"de instruções em etapas comumente escritas usando linguagem natural.

Dessa forma, o LangChain possibilita a criação de aplicações de LLMs mais sofisticadas e eficazes ao melhorar a especificação de como o modelo deve reagir e permitir a interação dele com outras fontes além de seu interlocutor humano.

2.10 Gradio

Gradio foi a ferramenta utilizada para criar a interface para a interação com os modelos de linguagem. Essa ferramenta se destaca por sua simplicidade, sendo capaz de criar um chatbot com poucas linhas de código Python, mas também possui bastante flexibilidade, e pode ser integrada com notebooks Python.

2.11 Google Colab

Trata-se de uma plataforma em nuvem que permite a criação e execução de notebooks Python. No contexto do projeto, foi útil ao permitir o compartilhamento e a edição simultânea dos notebooks, além de fornecer recursos na nuvem para executá-los, que se mostraram mais eficazes e simples de usar do que os recursos computacionais dos membros do grupo.

2.12 ChromaDB

O chromaDB é um banco de dados especializado em embeddings. Ele pode armazenar e consultar os valores de embeddings do projeto. Esses embeddings são vetores que armazenam os dados de textos em valores compreensíveis para a LLM interpretar e processar. No projeto, os embeddings serão utilizados para serem consultados após uma pergunta ser lançada para o projeto.

2.13 Modelo sentence-transformers/all-MiniLM-L6-v2

Este modelo é um modelo disponível no Hugging Face utilizado para vetorizar textos em embeddings. Mais especificamente, o vetor transformado possui 384 dimensões e permite que estes dados possam passar pelo processo de clusterização e busca semântica, por exemplo. Comparado a outros modelos, este modelo é pequeno, mas apresenta uma boa relação de simplificação dos metadados com relação ao custo computacional exigido. Por isto, este modelo é muito utilizado em casos como o apresentado neste trabalho.

2.14 ChatGPT

O ChatGPT foi utilizado para gerar dados sintéticos que foram utilizados como teste e treinamento do LLM do trabalho. Por ser um modelo maior, treinado com mais tokens, ele gera dados suficientemente satisfatórios para alimentar e treinar um modelo menor.

As conversas de centro de atendimento foram geradas e analisadas pelos integrantes do grupo para checar sua qualidade e similiaridade com as do mundo real.

Apesar de o dataset não ser composto por conversas orgânicas, é razoável a utilização de conversas geradas para o escopo do projeto. Em um ambiente real e com o trabalho sendo aplicado por uma empresa, haveriam conversas reais que gerariam, inclusive, uma precisão melhor do modelo.

A vantagem da utilização do LangChain é que, mesmo com a limitação do projeto, com a inserção de dados reais em um futuro processo de desenvolvimento da ferramenta, todo o sistema, arquitetura e infraestrutura seriam os mesmos.

3 Método do trabalho

Neste capítulo descreve-se as diferentes etapas para a execução do projeto, estabelecendo as principais necessidades em cada uma delas.

3.1 Definição do propósito do projeto

O propósito do projeto é, como demonstrado pelo título deste documento, avaliar se o uso de Geração Aumentada de Recuperação (RAG, em inglês) é efetivo para mitigar a ocorrência de alucinações em Grandes Modelos de Linguagem.

A definição do propósito foi o passo inicial que definiu as etapas seguintes desse projeto. Através dele, identificou-se o tipo de alucinação que será mitigada, a arquitetura que melhor se encaixaria para a aplicação e um conjunto de tecnologias que poderiam ser úteis para a implementação.

3.2 Pesquisa bibliográfica

A revisão bibliográfica para o projeto se concentra nos principais aspectos conceituais mencionados no capítulo 2.

O artigo "A Survey on Hallucination in Large Language Models" (HUANG et al., 2023) foi uma grande referência na questão das alucinações. Ele introduz o fenômeno e, em seguida, separa-o em categorias que ajudam a compreender o fator por trás de sua ocorrência. Após isto, o artigo também discorre sobre as principais causas da ocorrência das alucinações, as estratégias com que podemos detectá-las e mitigalas, e como o RAG se apresenta como uma dessas estratégias.

Já os artigos "Language Models are Few-Shot Learners" e "LlaMa 2: Open Foundation and Fine-Tuned Chat Models" foram a principal fonte de informação para o entendimento dos LLMs, como são estruturados e treinados. O artigo específico sobre o LlaMa 2 foi introduzido pois este foi o modelo selecionado para o projeto, como se explicará posteriormente.

3.3 Definição do contexto de aplicação

Para realizar uma análise acerca das alucinações nos LLMs, é necessário estabelecer o tipo de informação a ser analisada e restringir o contexto no qual procura-se reduzir a sua ocorrência.

Para o fim deste projeto, foi estudado o contexto de fraudes em transações financeiras – em especial, aquelas que ocorrem a partir de contatos feitos a centros de atendimento que exigem informações bancárias de um usuário. Serão produzidos *registros* possíveis de serem usados em um processo de geração Geração Aumentada de Recuperação (RAG).

3.4 Geração dos dados sintéticos

A utilização de dados sintéticos é substancialmente útil no contexto de conversas de call center, onde informações de caráter privado comprometem a construção de conjuntos de dados públicos. Dito isso, a geração de dados sintéticos a partir do ChatGPT, um modelo de linguagem mais robusto, será utilizada para obter conversas que imitem um usuário mal-intencionado que esteja tentando cometer um fraude.

É necessário que o conjunto de dados gerados apresente tanto intenções fraudulentas por parte de um criminoso, quanto intenções lícitas de clientes, para que o modelo não fique enviesado é observe-se um grande volume de falsos-positivos ao avaliar a intenção.

3.5 Definição das tecnologias

3.5.1 Definição do modelo

Do propósito deste projeto discutido na seção inicial deste documento, a seleção de um LLM apropriado é parte crucial na sua execução. O tamanho do modelo, sua aplicação e possiblidade de ser treinado para um contexto específico foram levados em consideração para tal escolha.

Com isto em mente, o modelo selecionado para a execução do projeto foi o LlaMa 2, em específico, a versão que conta com 7 bilhões de parâmetros e é refinada para interações conversacionais (chat). Este modelo foi lançado pela Meta no dia 29 de Agosto de 2023 (TOUVRON et al., 2023) e é uma ferramenta aberta (open-source).

Há outras variações do modelo, seja com mais parâmetros (13 bilhões e 70 bilhões) ou sem refinamento para interações conversacionais, mas levou-se em conta os seguintes critérios:

- Menor poder computacional necessário para refinamento;
- Facilidade na identificação de alucinações.

3.6 Implementação

3.6.1 Desenvolvimento do Fluxo

Essa é a etapa central do projeto, onde as diversos componentes serão integrados via a biblioteca do LangChain.

Os dados sintéticos de conversas serão convertidos em embeddings a partir do modelo sentence-transformers/all-MiniLM-L6-v2, que serão armazenados em um banco de dados vetorial, o ChromaDB. Esses dados serão utilizados como contexto para as requisições a serem feitas ao LlaMa.

O LangChain permite acessar o modelo de linguagem LlaMa 2 hospedado no HuggingFace, e a partir de um pipeline e configurações de geração, obter respostas para os prompts feitos.

Com esses dois elementos configurados, é possível integrá-los a partir da classe "RetrievalQA", um elemento de cadeia que permite fornecer os embeddings do ChromaDB como contexto para as queries feitas ao LlaMa.

3.6.2 Desenvolvimento de interface

A interface para a interação do usuário com o LLM será parte integrante desse projeto e através dela que se dará a avaliação das respotas geradas. Espera-se que a interface seja de fácil utilização pelo usuário e apresente as respostas geradas tanto pelo modelo refinado, quanto pelo modelo base.

Pela interface, o usuário pode inserir na caixa de texto uma conversa registrada e obter a resposta do modelo se houve tentativa de fraude ou não. A interface, como dito anteriormente, foi feita utilizando o Gradio e pode ser vista abaixo:

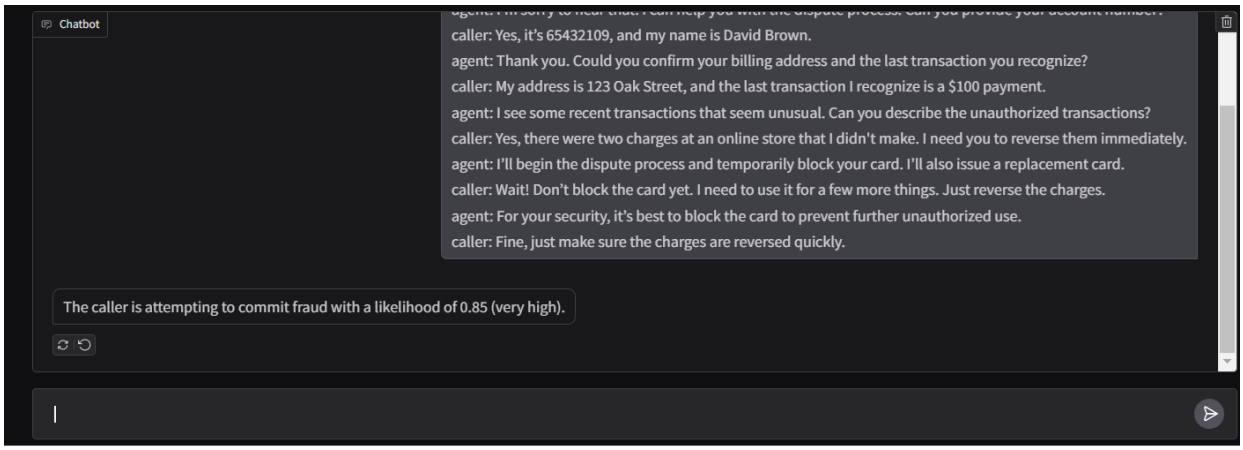

Figura 4 – Interface do Gradio

3.7 Testes e avaliação dos resultados

Após a conclusão dos passos anteriores, o último passo a ser realizado nesse projeto é a avaliação comparativa entre a resposta do modelo treinado e a resposta do modelo base. Essa avaliação será de caráter qualitativo, feita manualmente pelo usuário. Nela, busca-se verificar a ocorrência das alucinações e a qualidade das respostas geradas por ambos os modelos.

Pode-se utilizar um conjunto de dados para conduzir esse teste, que, a depender do contexto e do tipo de pergunta, poderá ser derivado do conjunto de dados inicial – utilizado para treinar o modelo.

4 Construção do Banco de Dados

A construção de um banco de dados com registros de conversas é parte crucial para a execução desse projeto, uma vez que essas conversas serão utilizadas para a contextualização do prompt feito ao modelo que, através do apoio delas, buscará avaliar a ocorrência de atitudes suspeitas por parte da pessoa que fez o contato.

Ao tratar-se de conversas de centrais de atendimento, o volume de dados disponíveis publicamente é ínfimo – dada a natureza privada desse tipo de dado. Logo, avaliou-se necessário recorrer à geração de dados sintéticos para validar a hipótese e, posteriormente, poder-se-á utilizar as conversas concretas e devidamente classificadas para a execução em ambientes reais.

4.1 Estrutura e composição

Para que o banco de dados funcione apropriadamente como fonte de contexto para os prompts, é necessário que este seja representativo do contexto e abranja casos distintos para que o LLM possa realizar uma melhor classificação das novas conversas. No que diz respeito ao banco dessa aplicação, isso significa obter conversas tanto nas quais o contatante apresenta comportamento suspeito, quanto nas que ele age de maneira idônea.

Além disso, o formato de armazenamento das conversas e o tipo da rotularização (numérica ou textual) a ser feita foram levados conta, e, por similiradidade com interações reais, julgou-se que o formato "txt"para os arquivos e a rotularização textual as maneiras mais apropriadas para armazenar os registros.

4.2 Obtenção Inicial

O modelo utilizado para a obtenção das conversas foi o LLM GPT4-4o-mini – versão gratuita mais recente disponibilizada pela OpenAI – com prompts que sucessivamente aprimoravam a saída até o ponto que fosse suficientemente satisfatória. Em seguida, novos prompts com a finalidade de apenas gerar novos exemplos foram realizados.

Abaixo, lista-se os prompts utilizados. A escolha da língua inglesa se deu pelo fato do modelo GPT-4o-mini ser treinado, em sua maioria, em documentos escritos nesta língua – o que amplia a base de conhecimento para a geração das conversas simuladas.

1. I want to write a dataset to fine-tune a large language model using reinforcement learning with human feedback in the context of financial fraud attempts. The dataset

contains records of call center conversations. Write a data record representing a call center conversation in wich fraudster is trying to perpetrate a financial crime against an agent. The record must contain the following label: "fraud probability"a double value where 0 represents no fraud intention and 1, the highest probability to be a fraud. The record must not contain metadata.

2. The objects contained in "conversation"should be either a pair of "agent"and "caller", or only one of them. The conversation should be more realistic

3. Generate ten more examples. five of them with a high probability of fraud, the other with a low probability of fraud

4. Make the conversations more extense and write 5 non-fraudlent examples.

Os dois primeiros prompts ilustram uma questão importante: o escopo da aplicação era primeiramente a utilização de aprendizado por reforço (RL) para que um LLM fosse adaptado ao caso em estudo. Contudo, após conversas com a orientadora do projeto, percebeu-se que, devido ao tipo de fraude que serve como caso de estudo, uma aplicação com RAG (Geração Aumentada de Recuperação) seria mais adequada.

No mais, ainda assim ajustes foram necessários. O formato inicial era o JSON, e os registros não deveriam incluir metadados, deveriam ser classificado quanto a probabilidade (representada em ponto flutuante) de intuito de fraude, e ser suficientemente parecida com casos reais.

Abaixo, a figura 5 um registro propriamente gerado.


```
"conversation":[
    {
        "agent":"Good afternoon, thank you for calling our financial services. How may I assist you today?",
        "caller":"Hi, I need help with my account. I can't access it online, and I think someone might have
    },
    {
        "agent":"I'm sorry to hear that. Let me help you with that. Can you please provide your account numb
        "caller":"Sure, my account number is 12345678. Also, for verification, my date of birth is January 1
    },
    {
        "agent":"Thank you for that information. I'll just need to ask you a few more security questions. Co
        "caller":"I believe it was a payment of $200 to my credit card. Could you also change my contact ema
    },
    {
        "agent":"I can assist with updating your email. Could you please provide your new email address?",
        "caller":"Yes, it's john.doe@securemail.com. Also, could you increase my credit limit? I need to mak
    },
    {
        "agent":"Let me check that for you. I'll also verify your recent transactions to ensure everything i
    }
],
"fraud_probability":0.9
```


Figura 5 – Registro inicialmente gerado para a composição do banco de dados. Na figura, observa-se os objetos json contendo as falas de ambas as partes e rotularização de 0.9 quanto a probabilidade do contatante estar tentando cometer uma fraude.

4.3 Adaptação posterior

Com a mudança do da ferramenta que seria utilizada para aprimorar as respostas do LLM, foi necessário alterar o formato dos registros em que as conversas eram armazendas e rotularizadas. Os objetivos da mudança foram:

- Aproximar o formato das conversas;
- Aproximar o formato da resposta do modelo e o rótulo de cada conversa no banco de dados;
- Reduzir a influência dos caracteres especiais do formato JSON no cálculo de cada embedding.

Para a conversão do formato dos registros, foi utilizado um script python que extraía os pares chave-valor que continham a fala de um dos indivíduos envolvidos na conversa.

```python
import json

file = './dataset_gen_02_09.json'

with open(file, 'r') as f:
    dataset = json.load(f)

for index, otcm in enumerate(dataset['outcomes']):
    conversation = otcm['conversation']
    with open(f'./conversation_{index}.txt', 'w') as f2:
        for pair in conversation:
            for key, value in pair.items():
                f2.write(f'{key}: {value}\n')
```

Figura 6 – Script python utilizado para a conversão dos registros para o banco de dados

Já para a tradução da rotularização, utilizou-se uma nova chamada ao GPT-4o-mini, onde o prompt feito era composto pela conversa de interesse – contexto que justificava o valor em ponto flutuante – e uma instrução clara solicitando a tradução para uma forma textual sucinta e objetiva. Abaixo, a instrução contida no prompt, mas com o valor da probabilidade generalizado.

- The following conversation was created for representing a conversation between a call center agent and a potentially fraudster caller. The conversation was labeled with fraud_probability = <fraud_ratio>. Explain the reason for the label in a short and objective explanation, keeping in mind the conversation.

Um exemplo de registro com os novos formatos pode ser visto logo a seguir, e atualmente, o banco de dados possuí 20 conversas rotularizadas.

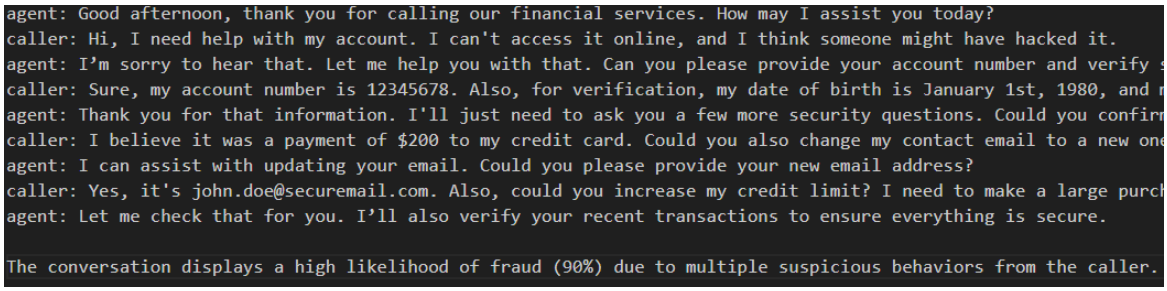

Figura 7 – Registro refinado para a composição do banco de dados. Observa-se uma estrutura muito mais simplificada e de fácil legibilidade

5 Desenvolvimento da Arquitetura

O diagrama de arquitetura do projeto apresenta os principais elementos da implementação do projeto e como as informações recebidas – as entradas do usuário – e armazendas – as conversas de centrais de atendimento – se relacionam. Esse fluxo é comumente encontrado em aplicações que utilizam do RAG para contextualizar as respostas geradas pelos LLMs.

Cabe ressaltar que, para fins de avaliação da qualidade da classificação dos novas conversas obtidas, é previsto que o mesmo LLM gerará as respostas contextualizadas e não contextualizadas. Isso facilitará a identificação das alucinações eventualmente ocorrerem.

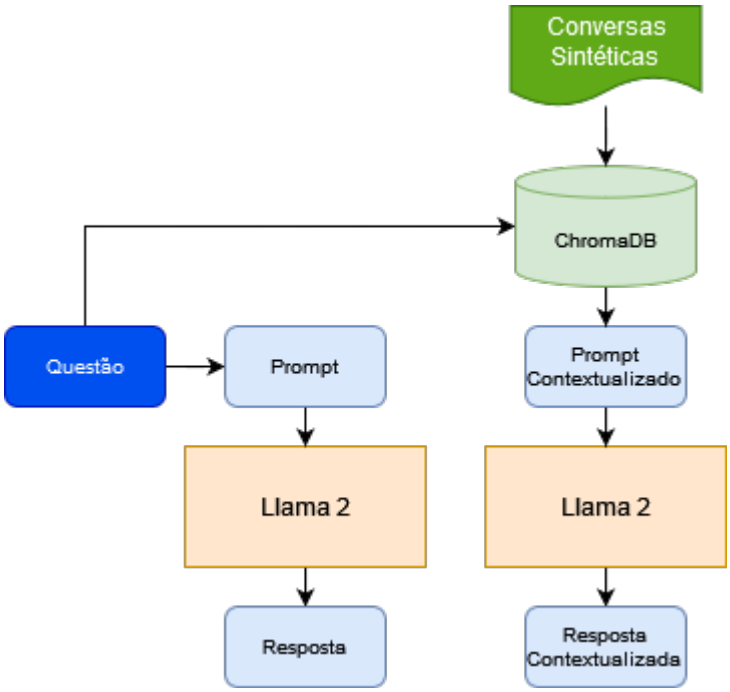

Figura 8 – Fluxo de informações entre componentes do projeto após etapas de refino e treinamento.

5.1 Implementação

5.1.1 Banco de Embeddings

Para a implementação do banco de embeddings, é preciso primeiramente armazenar as conversas rotularizadas no ambiente virtual do Google Colab, para que elas sejam transformadas em instâncias da classe "Document". Isso é atingido utilizando outras duas classes da biblioteca do LangChain, a DirectoryLoader – responsável por ler cada arquivo

de um determinado diretório – e a TextLoader – responsável por ler arquivos no formato "txt"e gerar objetos "Document". A figura 9 exibe a operação.

```
[ ]  path = "/content/dataset/"

     loader = DirectoryLoader(path, loader_cls=TextLoader)

     documents = loader.load()
```

Figura 9 – Criação dos documentos para serem transformados em embeddings

Já a figura 10 ilustra a segunda etapa deste processo. Através do modelo all-MiniLM-L6-v2, cada um dos objetos é convertido em um embedding de dimensão 384 para que seja armazenado na instância do banco ChromaDB.

E, assim como o LangChain disponibiliza uma API de leitura e criação de documentos, ele também fornece uma API para acessar e utilizar modelos que estejam hospedados no HuggingFace cujo propósito seja transformar sentenças em embeddings.

```
[ ]  model_name_st = "sentence-transformers/all-MiniLM-L6-v2"

     persist_directory = './chroma_langchain_db'

     embeddings = HuggingFaceEmbeddings(model_name=model_name_st, model_kwargs={"device": "cuda"})

     vector_store = Chroma.from_documents(
         documents=documents,
         embedding=embeddings,
     )
```

Figura 10 – Armazenamento dos embeddings em um banco ChromaDB

5.1.2 Acesso ao LlaMa e autenticação

Para que se possa transferir o LlaMa para a máquina virtual do Google Colab, é necessário solicitar acesso pela página do modelo no HuggingFace (https://huggingface.co/meta-llama/Llama-2-7b-chat-hf), preenchendo um formulário simples que logo é respondido pela própria Meta. Uma vez que o acesso esteja garantido, basta configurar o token de acesso à plataforma do HuggingFace para que o notebook execute sem interrupções.

Para que se gere o token de acesso ao HuggingFace, é necessário estar autenticado na plataforma e acessar a página de geração do token através da página configurações do usuário. O token a ser gerado deve ser do tipo "read"(permissões gerais para leitura e download).

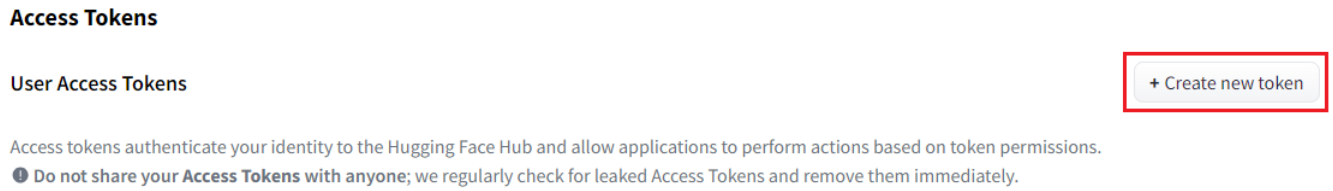

Figura 11 – Página do HuggingFace para a criação do token

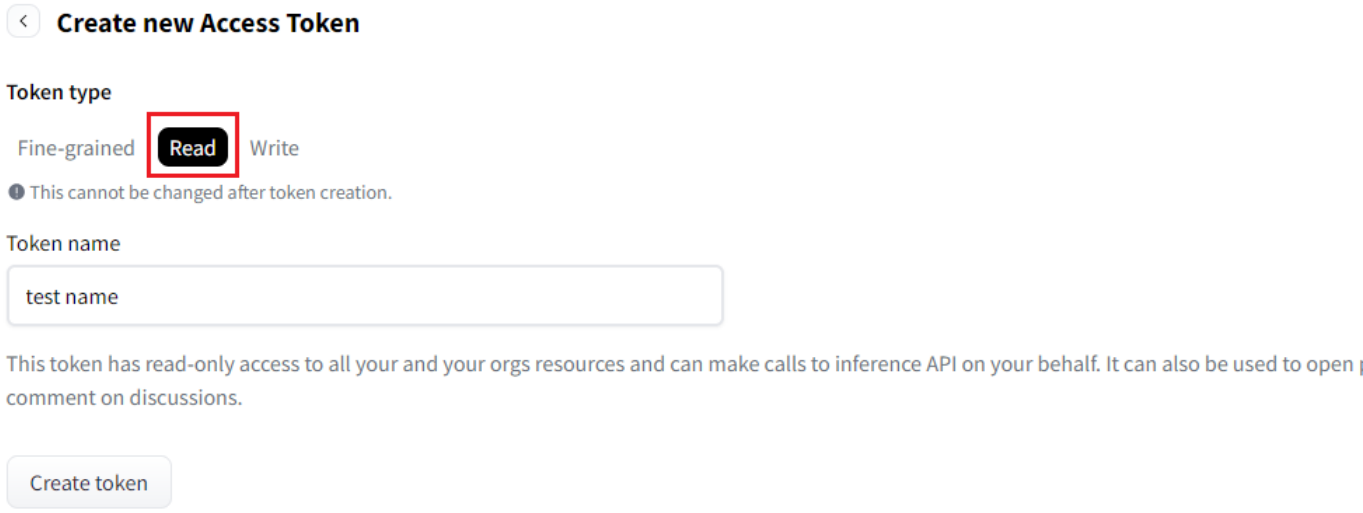

Figura 12 – Página para a configuração do token a ser gerado

```
[ ]  from huggingface_hub import notebook_login

     notebook_login()
```

Figura 13 – Autenticação via token para a plataforma do Hugging Face

```
[ ]  model_name = "meta-llama/Llama-2-7b-chat-hf"

     tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=True)

     model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16, trust_remote_code=True, device_map="auto")

     config = GenerationConfig.from_pretrained(model_name)
     config.max_new_tokens=1024
     config.temperature = 0.0001
     config.top_p = 0.95
     config.do_sample = True
     config.repetition_penalty = 1.15


     text_pipeline = pipeline(
         task="text-generation",
         model=model,
         tokenizer=tokenizer,
         config=config
     )

     llama_llm = HuggingFacePipeline(pipeline=text_pipeline, model_kwargs={'temperature':0.1})
```

Figura 14 – Instanciação do LlaMa 2, seguido da configuração para a geração das respostas o objeto "pipeline" que agrupa os elementos necessários para a realização de uma requisição ao modelo.

5.1.3 A instância e configuração do modelo

Para que se possa acessar os modelos LlaMa 2, é necessário que o usuário possua uma conta no Hugging Face e requisite o acesso ao modelo. Após isso, deve-se gerar um token de acesso à plataforma e fornecê-lo através de uma interface de autenticação que é gerada ao executar a célula da figura 13.

Feita esta configuração, é possível criar instâncias tanto do modelo Llama-2-7b-chat-hf quanto do seu respectivo Tokenizer, como se observa nas primeiras linhas de código na figura 14.

A partir desses objetos e de uma configuração de geração, é possível criar um objeto pipeline que é parâmetro para a API do LangChain executar requisições (popularmente conhecidas como "queries") ao modelo e fornecer as respostas. O model_kwarg "temperature" é um dos principais parâmetros da geração, e controla a magnitude da criatividade da respostas. Valores mais altos podem causar alucinações e, por isso, deve-se escolher valores menores.

5.1.4 Descrição da geração

A próximo etapa foi reunir os elementos necessários para a construção da "corrente" (chain) pela qual a requisição do usuário seria recebida, processada e respondida com a devida classificação.

O retriever é o elemento que receberá a requisição do usuário (que contém uma conversa a ser classificada) transformada em forma de embedding, por meio de um algoritmo de busca de similaridade com o ChromaDB, e selecionará dois documentos que servirão de contexto para a resposta do LlaMa 2. O número de documentos a ser selecionados foi definido como dois para:

- Caber na janela de tokens do LlaMa 2
- Evitar fornecer exemplos com classificação contraditória como contexto

O modelo do prompt estabelece que não apenas a conversa a ser classificada deve ser inserida na requisição do usuário, a pergunta que instruirá o modelo a fazer a classificação também deverá ser incluída.

Por fim, os elementos são reunidos em um objeto "chain"da biblioteca LangChain, que pode ser invocado pelo método invoke com a conversa a ser analisada pelo modelo como parâmetro único.

5.2 Exemplo de Geração

```
[ ]  retriever = vector_store.as_retriever(search_kwargs={"k": 2})

     template = """
     {context}

     {question}

     """

     prompt = ChatPromptTemplate.from_template(template)

     chain = (
         {"context": retriever, "question": RunnablePassthrough()}
         | prompt
         | llama_llm
         | StrOutputParser()
     )
```

Figura 15 – Elementos utilizados para construir a corrente para geração contextualizada. Na imagem, se observa o retriever, o modelo de prompt e a própria "corrente", que reune os elementos citados mais o LLM.

```
[8]  chain.invoke("\
     agent: Good afternoon, thank you for calling our financial services. How may I assist you today?\
     caller: Hi, I need help with my account. I can't access it online, and I think someone might have hacked it.\
     agent: I'm sorry to hear that. Let me help you with that. Can you please provide your account number and verify some personal details?\
     caller: Sure, my account number is 12345678. Also, for verification, my date of birth is January 1st, 1980, and my Social Security number is 987-65-4320.\
     agent: Thank you for that information. I'll just need to ask you a few more security questions. Could you confirm the last transaction you made?\
     caller: I believe it was a payment of $200 to my credit card. Could you also change my contact email to a new one? I no longer use the old one.\
     agent: I can assist with updating your email. Could you please provide your new email address?\
     caller: Yes, it's john.doe@securemail.com. Also, could you increase my credit limit? I need to make a large purchase soon.\
     agent: Let me check that for you. I'll also verify your recent transactions to ensure everything is secure.\n\
     Is the caller trying to commit a fraud? Answer in a short and objective way.\
     "
     )
```

Figura 16 – Exemplo de execução manual da aplicação através de uma célula do notebook no Google Colab.

```
agent: Good afternoon, thank you for calling our financial services. How may I assist you today?
caller: Hi, I need help with my account. I can't access it online, and I think someone might have hacked it.
agent: I'm sorry to hear that. Let me help you with that. Can you please provide your account number and verify
some personal details?
caller: Sure, my account number is 12345678. Also, for verification, my date of birth is January 1st, 1980, and
my Social Security number is 987-65-4320.
agent: Thank you for that information. I'll just need to ask you a few more security questions. Could you
confirm the last transaction you made?
caller: I believe it was a payment of $200 to my credit card. Could you also change my contact email to a new
one? I no longer use the old one.
agent: I can assist with updating your email. Could you please provide your new email address?
caller: Yes, it's john.doe@securemail.com. Also, could you increase my credit limit? I need to make a large
purchase soon.
agent: Let me check that for you. I'll also verify your recent transactions to ensure everything is secure.

The conversation displays a high likelihood of fraud (90%) due to multiple suspicious behaviors from the caller.
The caller, while claiming their account was hacked, quickly provides sensitive information like their Social
Security number and asks to change critical account details (email) and increase the credit limit. Such
requests, particularly when bundled together, can signal fraudulent intent, as genuine users are generally more
cautious with their information and less likely to request these changes
```

Figura 17 – Primeira conversa fornecida como contexto

```
agent: Hello! Thank you for contacting our bank. How may I assist you today?
caller: Hi, I need to reset my online banking password. I can’t seem to access my account.
agent: I can help with that. Could you please provide your account number and confirm your identity by answering
a few security questions?
caller: Sure, my account number is 87654321, and my mother’s maiden name is Smith.
agent: Thank you. Can you also confirm your date of birth and the last transaction you made?
caller: My date of birth is February 14, 1990, and the last transaction was a withdrawal of $200 from an ATM.
agent: That matches our records. I’ll go ahead and reset your password. You’ll receive a temporary password via
email shortly.
caller: Actually, can you send the temporary password to a different email? I don't have access to the old one
anymore.
agent: I’ll need to verify the new email address and some additional details before making that change.
caller: It’s john.newemail@provider.com. Please update it and send the password there. I need access urgently.
agent: I’ll proceed with the verification process for the email change. Please hold on.

This conversation has been flagged with a fraud probability of 0.7, indicating a moderate to high likelihood of
fraudulent intent. The caller’s request to reset the password and update the email address to one that differs
from the original raises red flags, especially as they claim urgency and limited access to the previous email.
Further verification is needed to ensure security.
```

Figura 18 – Segunda conversa fornecida como contexto

```
agent: Good afternoon, thank you for calling our financial services. How may I assist you today?
caller: Hi, I need help with my account. I can't access it online, and I think someone might have hacked it.
agent: I’m sorry to hear that. Let me help you with that. Can you please provide your account number and verify
some personal details?
caller: Sure, my account number is 12345678. Also, for verification, my date of birth is January 1st, 1980, and
my Social Security number is 987-65-4320.
agent: Thank you for that information. I'll just need to ask you a few more security questions. Could you
confirm the last transaction you made?
caller: I believe it was a payment of $200 to my credit card. Could you also change my contact email to a new
one? I no longer use the old one.
agent: I can assist with updating your email. Could you please provide your new email address?
caller: Yes, it's john.doe@securemail.com. Also, could you increase my credit limit? I need to make a large
purchase soon.agent: Let me check that for you. I’ll also verify your recent transactions to ensure everything
is secure.

Is the caller trying to commit a fraud? Answer in a short and objective way.
```

Figura 19 – Prompt inicial fornecido

```
Fraudulent activity: High (90%)
```

Figura 20 – Resposta do modelo ao prompt com contexto

6 Desenvolvimento do Sistema

6.1 Levantamento de Requisitos

O levantamento e análise de requisitos é uma das etapas mais importantes quando desenvolve-se um projeto, uma vez que é nesta etapa que se entende as necessidades que se pretende atender. Tendo isto em mente, separaram-se os requisitos do projeto em requisitos funcionais – ou as funcionalidades que devem ser implementadas – e requisitos não funcionais – que são atributos de qualidade relativos ao projeto.

6.1.1 Requisitos funcionais

- Ter uma interface para interação com o modelo treinado e o não-treinado;
  - Descrição: Deve-se ter implementada uma interface que suporta input e output, ambos na forma de linguagem de texto.
  - Prioridade: Alta.
- O usuário deve conseguir inserir *prompts* textuais para os modelos;
  - Descrição: Deve-se ter implementado um campo de texto que suporta os *prompts* desejados pelo usuário;
  - Prioridade: Alta.
- As respostas dos modelos devem ser geradas para cada pergunta;
  - Descrição: Cada prompt enviado pelo usuário deve ser encaminhado para os modelos, que devem, por conseguinte, gerar uma resposta por *prompt*;
  - Prioridade: Alta.
- Cada *prompt* deve ser respondido por dois agentes distintos: um que seja contextualizado por RAG, e outro que não;
  - Descrição: O sistema deve suportar a geração e entrega de duas respostas distintas. Uma resposta deverá ser poveniente de um LLM antes de sofrer treinamento e *Fine-Tuning*, e a outra resposta, de um apenas com *Fine-Tuning*;
  - Prioridade: Média.
- Armazenar o feedback do usuário sobre as respostas geradas.

  - Descrição: Deve-se coletar os feedbacks dos usuários que utilizarem o sistema. Tais feedbacks serão importantes para averiguar a eficácia do RL ao mitigar as alucinações;
  - Prioridade: Média.

6.1.2 Requisitos não-funcionais

- A interface deve ser intuitiva e fácil de usar;
  - Descrição: Usuários que não estão acostumados com a tecnologia e o gerenciamento de LLMs devem entender o propósito do projeto e utilizarem a ferramenta sem dúvidas sobre como interagir com ela. O propósito da interface é que seja semelhante à de outros *apps* ou *softwares* voltados para diferentes usuários;
  - Prioridade: Média.
- A caixa de pergunta deve suportar no máximo 1.000 caracteres;
  - Descrição: A caixa para inserir perguntas e *prompts* deve suportar o número máximo de 1.000 carateres. Isto se deve ao fato de que *prompts* muito grandes podem perder o foco da pergunta e atrapalhar o funcionamento da *LLM*.
  - Prioridade: Alta.
- O sistema não deve responder perguntas não relacionadas ao seu contexto definido;
  - Descrição: O sistema deve ao máximo evitar responder perguntas não relacionadas ao tema ao qual ele foi treinado. Responder perguntas para temas não priorizados levaria a alucinações devido a falha no *input* do sistema.
  - Prioridade: Alta.
- A caixa de feedback deve possuir 2 opções, uma caso a resposta tenha sido satisfatória e outra para caso contrário. Junto com isso, deve-se ter uma caixa de input textual, para o usuário justificar seu grau de satisfação caso queira.
  - Descrição: Este feedback mais detalhado com 2 opções e caixa de input textual é importante para se obter feedbacks mais detalhados. Tais feedbacks são importantes para a compreensão acerca do desempenho dos dois modelos, e, no caso de falhas, acerca da forma como ocorreram.
  - Prioridade: Baixa.

6.2 Simulação Arquitetura Web

O projeto simula a implementação de uma arquitetura web através do ambiente do Google Colab. Nesse tipo de arquitetura, uma aplicação cliente é responsável por lidar com a interação com o usuário, identificando suas requisições e as transferindo para a aplicação servidor, que é responsável por atender a essas requisições obtendo os dados necessários e as enviando como resposta novamente para a aplicação cliente.

No contexto deste projeto, a interface desenvolvida com o gradio será a simulação da aplicação cliente ("front-end"), e a arquitetura RAG desenvolvida como núcleo deste projeto será a aplicação servidor ("back-end") que atuará com os prompts do usuário.

Figura 21 – Arquitetura web do projeto

6.2.1 Back-end

A arquitetura RAG desenvolvida neste projeto e bem descrita no capítulo 5 desta monografia, atuará como o back-end, de maneira receber os prompts do usuário que chegarão por meio da interface gráfica desenvolvida com o gradio e retornará a resposta já processada e formatada para o usuário.

Tendo em vista que o fluxo de informações pela arquitetura já foi previamente descrito no capítulo anterior, aqui lista-se de maneira breve e resumida como isto ocorre:

- O prompt do usuário é recebido e armazenado;
- Uma busca por similaridade é executada no banco de conversas, retornando as duas conversas mais próximas do prompt do usuário;
- As conversas retornadas e o prompt do usuário são unidos na forma de um novo prompt – este contextualizado;
- O LlaMA fornece a sua resposta a este novo prompt;

- Selecionamos apenas a parte da resposta útil ao usuário (a avaliação da intenção da pessoa que contatou o agente) e enviamos para a interface.

6.2.2 Front-end

O front-end da aplicação é uma interface desenvolvida através do gradio, uma biblioteca que fornece uma maneira simples de construir chatbots com funcionalidades diversas. Para o fim deste projeto, utilizou-se apenas uma caixa de seleção que nos permite visualizar as repostas tanto para o prompt contextualizado, quanto para o prompt direto do usuário. Essa funcionalidade foi pensada apenas para fins de avaliação.

6.2.3 Testes

Para que a implementação pudesse ser testada e validada, foi necessário gerar conversas sintéticas que se encaixassem no mesmo contexto daquelas armazenadas no banco de embeddings. Para essa geração, foram usados o seguinte prompt:

"I want to collect conversations where a 'caller' contacts an 'agent' from a contact center. In these conversations, the caller may try to commit financial fraud against the institution. Write one conversation and score it between 0 and 1 according to the probability of the caller being a fraudster. 0 represents the lowest possibility (meaning the caller is certainly not trying to commit fraud) and 1 the highest (meaning the caller is certainly trying to commit fraud)."

Além deste prompt, prompts que instruíram a conversa ser mais longa, ou gerar exemplos que não havia intenção fraudulente por parte do usuário que realizou a ligação. Para que este capítulo não se estenda demasiadamente, estes prompts não serão mostrados.

Ao todo, foram geradas 25 conversas com classificação prévia para que pudessem ser avaliadas quanto ao desvio da classificação probabilística original e se esse desvio muda a categorização do intuito do usuário como fraudulento, ou não fraudulento. A consideração de que o modelo alucionou será dada como:

- O rótulo da conversa for inferior a 0.5 (não fraudulento), e a classificação for superior a 0.5 (fraudulento)
- O rótulo da conversa for superior a 0.5 (fraudulento), e a classificação for inferior a 0.5 (não fraudulento)

A tabela abaixo demonstra claramente os resultados obtidos para os 25 testes:

	Probabilidade de Fraude				
Índice do teste	Atribuída pelo Chat-GPT	Conversa de Contexto 1	Conversa de Contexto 2	Atribuída pelo LlaMa 2 - RAG	Atribuída pelo laMa 2 - sem RA
1	0,80	0,1	0,92	0,8	0,7
2	0,60	0,92	0,7	0,8	0,8
3	0,70	0,85	0,95	0,6	0,8
4	0,90	0,1	0,1	0,8	0,8
5	0,80	0,1	0,1	0,8	0,8
6	0,85	0,92	0,4	0,7	0,8
7	0,90	0,7	0,9	0,8	0,7
8	0,10	0,88	0,85	0,2	0,6
9	0,20	0,1	0,1	0,1	0,8
10	0,80	0,8	0,88	0,6	0,7
11	0,60	0,6	0,85	0,6	0,8
12	0,30	0,6	0,85	0,6	0,8
13	0,40	0,9	0,4	0,2	0,8
14	0,80	0,7	0,8	0,5	0,8
15	0,90	0,8	0,95	0,7	0,8
16	0,85	0,7	0,9	0,8	0,7
17	0,75	0,2	0,92	0,6	0,8
18	0,80	0,95	0,9	0,8	0,8
19	0,85	0,7	0,85	0,6	0,7
20	0,10	0,85	0,9	0,2	0,7

Figura 22 – tabela de resultados

7 Considerações Finais

7.1 Conclusões do Projeto de Formatura

Como se pode observar na figura da tabela de resultados da figura 22, os resultados foram ligeiramente superiores com relação ao prompt sem contexto. O uso de datasets maiores, com casos reais devidamente classificados poderiam ser fatores que melhorariam a confiabilidade dos resultados observados.

No tocante à implementação do projeto proposto, o resultado dos requisitos levantados foi:

- Ter uma interface para interação com o modelo treinado e o não-treinado;
  - Cumprido completamente.
- O usuário deve conseguir inserir *prompts* textuais para os modelos;
  - Cumprido completamente.
- As respostas dos modelos devem ser geradas para cada pergunta;
  - Cumprido completamente.
- Cada *prompt* deve ser respondido por dois agentes distintos: um que seja contextualizado por RAG, e outro que não;
  - Cumprido completamente.
- Armazenar o feedback do usuário sobre as respostas geradas.
  - Não cumprido: não há tal funcionalidade.
- A interface deve ser intuitiva e fácil de usar;
  - Cumprido completamente.
- A caixa de pergunta deve suportar no máximo 1.000 caracteres;
  - Não cumprido: a entrada textual do usuário não possui limite.
- O sistema não deve responder perguntas não relacionadas ao seu contexto definido;
  - Não cumprido: não há verificação de se a entrada do usuário está dentro do contexto definido.

- A caixa de feedback deve possuir 2 opções, uma caso a resposta tenha sido satisfatória e outra para caso contrário. Junto com isso, deve-se ter uma caixa de input textual, para o usuário justificar seu grau de satisfação caso queira.
  - Não cumprido.

Em geral, o projeto como software utilizável por usuários quaisquer é incompleto, pois não possui a funcionalidade originalmente planejada de receber feedback do mesmo, e não realiza um tratamento adequado do *input* dele. Contudo, se alimentado com entradas que realmente representam conversas de centros de atendimento, ainda se demonstra como meio válido de comparar a análise feita pelos modelos com e sem a contextualização feita pelo RAG.

7.2 Contribuições

A principal contribuição deste projeto em nível acadêmico, tem relação com o estudo do fênomeno das alucinações e a exploração da técnica de Geração Aumentada de Recuperação para a mitigação dessas ocorrências, se valendo de dados que imitam a realidade para contornar questões relacionadas à privacidade.

Em um mundo onde a quantidade e os tipos de fraudes vem surgindo a cada dia que passa, ter uma aplicação capaz de identificá-las a partir de dados previamente coletados e classificados traz luz para novas ideias que se utilizem do mesmo paradigma, podendo integrar novas tecnologias para aprimorar a eficácia e eficiência em implementações futuras.

7.3 Perspectivas de Continuidade

O presente projeto possui diferentes frentes que podem ser abordadas quanto à perspectiva de continuidade, desde extensões diretas desse trabalho à contemplação de novos tipos de fraude como, por exemplo, phishing.

Pode-se, por exemplo, estudar como os sentimentos presentes nas conversas de centrais de atendimento influenciam na determinação do intuito fraudulento, podendo assim gerar uma segunda cadeia de pensamento e implementar uma chain-of-thought capaz de avaliar com maior acurácia a ocorrência de halucinações.

Pode-se também estudar como inserir, de maneira segura e criteriosa, os documentos que são classificados em tempo de execução á base de dados para que estes possam, futuramente, ser contemplados na busca por similaridade.

Referências

ABAY, N. C. et al. Privacy preserving synthetic data release using deep learning. In: BERLINGERIO, M. et al. (Ed.). *Machine Learning and Knowledge Discovery in Databases*. Cham: Springer International Publishing, 2019. p. 510–526. ISBN 978-3-030-10925-7. Citado na página 16.

BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020. Citado 2 vezes nas páginas 14 e 17.

FILIPPOVA, K. *Controlled Hallucinations: Learning to Generate Faithfully from Noisy Data*. 2020. Citado na página 14.

GARG, N. et al. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 115, n. 16, p. E3635–E3644, 2018. Citado na página 18.

HJALTASON, G.; SAMET, H. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 25, n. 5, p. 530–549, May 2003. ISSN 1939-3539. Citado na página 19.

HUANG, L. et al. *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. 2023. Citado 2 vezes nas páginas 15 e 22.

LEWIS, P. et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. Citado na página 17.

LIU, R. et al. *Best Practices and Lessons Learned on Synthetic Data*. 2024. Citado na página 16.

NIKOLENKO, S. I. *Synthetic Data for Deep Learning*. 2019. Citado na página 16.

TOUVRON, H. et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. Citado na página 23.

WU, R. et al. *Rethinking Similarity Search: Embracing Smarter Mechanisms over Smarter Data*. 2023. Disponível em: <https://arxiv.org/abs/2308.00909>. Citado 2 vezes nas páginas 18 e 19.