UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

Pablo Weslley Silva da Fonseca

Detecção de Fake News: Uma abordagem baseada em Large Language Models e Prompt Engineering

Recife

Março de 2025

Pablo Weslley Silva da Fonseca

Detecção de Fake News: Uma abordagem baseada em Large Language Models e Prompt Engineering

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Curso de Bacharelado em Sistemas de Informação

Recife

Março de 2025

Dados Internacionais de Catalogação na Publicação
Sistema Integrado de Bibliotecas da UFRPE
Bibliotecário(a): Suely Manzi – CRB-4 809

F676d Fonseca, Pablo Weslley Silva da.
Detecção de Fake News:: uma abordagem baseada em Large Language Models e Prompt Engineering / Pablo Weslley Silva da Fonseca. – Recife, 2025.
69 f.; il.

Orientador(a): Rinaldo José de Lima.

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal Rural de Pernambuco, Bacharelado em Sistemas da Informação, Recife, BR-PE, 2025.

Inclui referências e apêndice(s).

1. Notícias falsas. 2. Inteligência artificial. 3. Aprendizado do computador. 4. Redes neurais (Computação) 5. Sistemas de Recuperação da Informação. I. Lima, Rinaldo José de, orient. II. Título

CDD 004

Pablo Weslley Silva da Fonseca

Detecção de Fake News: Uma abordagem baseada em Large Language Models e Prompt Engineering

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada em: 20 de Março de 2025.

BANCA EXAMINADORA

Rinaldo José de Lima (Orientador)
Departamento de Computação
Universidade Federal Rural de Pernambuco

Douglas Veras e Silva
Departamento de Computação
Universidade Federal Rural de Pernambuco

“Estamos afogados em informação, enquanto famintos por sabedoria.”

(Edward O. Wilson, 2016)

Resumo

Este trabalho aborda o uso de Large Language Models (LLMs) para a detecção de *fake news* ou notícias falsas no idioma inglês e português. As notícias falsas têm gerado impactos negativos, como desinformação e conflitos sociais, sendo amplamente disseminadas pelas redes sociais. Embora métodos tradicionais de verificação sejam eficazes, como checagem manual e agências de verificação de fatos, a aplicação de algoritmos de machine learning e deep learning trouxe avanços importantes. No entanto, esses modelos apresentam limitações, como perda de contexto semântico e custos de treinamento. A introdução da arquitetura Transformers possibilitou avanços significativos com LLMs, como BERT, GPT e T5, devido à sua capacidade de compreender padrões linguísticos complexos. Este trabalho propõe uma abordagem de detecção de notícias falsas a partir recuperações de informações pela Web e o modelo Qwen2.5-7B-Instruct, comparando o desempenho com propostas que combina recuperação de informações com modelos tradicionais e LLMs. Os resultados destacam vantagens e desvantagens, contribuindo para futuras melhorias em sistemas automatizados de detecção de notícias falsas.

Palavras-chave: Detecção de notícias falsas, LLMs, Machine Learning, Deep Learning, RAG

Abstract

This works addresses the use of Large Language Models (LLMs) for the detection of fake news in English and Portuguese. Fake news has generated negative impacts, such as misinformation and social conflicts, and is widely disseminated through social networks. Although traditional verification methods are effective, such as manual checking and fact-checking agencies, the application of machine learning and deep learning algorithms has brought important advances. However, these models have limitations, such as loss of semantic context and training costs. The introduction of the Transformers architecture has enabled significant advances with LLMs, such as BERT, GPT and T5, due to their ability to understand complex linguistic patterns. This paper proposes an approach for detecting fake news from information retrieval on the Web and the Qwen2.5-7B-Instruct model, comparing the performance with proposals that combine information retrieval with traditional models and LLMs. The results highlight advantages and disadvantages, contributing to future improvements in automated fake news detection systems.

Keywords: Fake news detection, LLMs, Machine Learning, Deep Learning, RAG

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

LLMs	Large Language Models
PLN	Processamento de Linguagem Natural
DP	Deep Learning
ML	Machine Learning
GNN	Graph neural networks
BERT	Bidirectional Encoder Representations from Transformers
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
GloVe	Global Vectors for Word Representation
RAG	Retrieval-Augmented Generation
API	Application Programming Interface

Sumário

1 Introdução

As informações fraudulentas ou também chamadas de notícias falsas têm levantado dúvidas em relação à integridade das informações que consumimos pela internet (OLAN et al., 2024). Nos EUA, ”Fake News”se tornou um termo politizado, frequentemente usado para atacar a mídia tradicional, pois se refere a notícias fabricadas intencionalmente para enganar. Já as notícias falsas, no sentido geral, mantém um significado mais técnico e descritivo, englobando vários tipos de desinformação. Durante todo o trabalho, estaremos usando o sentido geral. A desinformação se tornou um dos maiores desafios da era digital. O consumo dessas notícias confunde nossos pensamentos, gerando um evento de decisões que cria situações de conflitos na sociedade, como foi o caso das eleições presidenciais de 2016 dos Estados Unidos (ZHANG; GHORBANI, 2020). Combater as notícias falsas é uma tarefa difícil, pois as multiplataformas de comunicação disponíveis na internet, em especial as redes sociais, espalham rapidamente essas informações por meio de compartilhamentos, postagens e seguidores (RAO; KALYANI, 2022), dificultando a resposta de jornalistas e plataformas de verificação de fatos, além de carência de pensamento crítico na maioria da população mundial. A Figura 1 mostra a evolução e presença das notícias falsas ao longo de 10 anos.

Figura 1 – notícias falsas trends (2015–2025)
(Google, 2025)

Isso reforça a importância de melhorar a estratégia de aceitação social, desenvolvendo o senso crítico dos usuários e providenciando ferramentas que auxiliem a identificar se uma notícia é fraudulenta. Dessa forma, podemos diminuir a proliferação e os impactos negativos das notícias falsas (OLAN et al., 2024). As formas humanas de verificar notícias falsas consistem em: processo manual de verificação e organizações focadas em verificar fatos (PAPAGEORGIOU et al., 2024). O processo manual

envolve especialistas que possuem acesso a informações verídicas ou sabem quais são as fontes confiáveis para obtê-las. As organizações de verificação são entidades especializadas em verificar a veracidade das informações propagadas por vários sites de notícias disponíveis na web, os mais conhecidos são:

- PolitiFact (HU et al., 2022): Sem fins lucrativos, é um site nos Estados Unidos que avalia as notícias políticas, classificando como “true”, “mostly true”, “half true”, “mostly false”, “false” e “pants on fire”.
- Snopes (SNOPES, 2025): Pioneiro em verificação de fatos de forma online. Desde 1994, o site contribui na verificação de notícias no ramo político, social e outros, focando principalmente nas notícias propagadas por redes sociais.
- Suggest (SUGGEST, 2025):Focado mais na parte de celebridades e entretenimento, o site investiga as histórias propagadas por revistas e sites, avaliando cada história em uma escala de 0 a 10, sendo 0 indicando ser um rumor completamente falso e 10 completamente verdadeira.

No ramo da inteligência artificial, estratégias têm sido testadas para auxiliar na detecção de notícias falsas, por exemplo, algoritmos de machine learning: Logistic Regression, Support Vector Machine, Multinomial Naive Bayes; e modelos de deep learning: Convolutional Neural Networks, Recurrent Neural Networks, Long Short-Term Memory. Essas técnicas trouxeram avanços na detecção automática, contudo, essas abordagens possuem desafios na modelagem de textos, pois dependem de um tamanho bem definido de entradas e saídas. Além disso, não é possível trabalhar diretamente com textos crus, há perda de semântica e contexto no processo de reduzir o tamanho do texto original e transformá-lo em vetores (ALGHAMDI; LUO; LIN, 2024). A introdução da arquitetura Transformers (VASWANI et al., 2017) possibilitou avanços na forma como modelamos os textos. Com melhores desempenhos e menos limitações comparados com os modelos de DP, escalamos para os Large Language Models (LLMs) como BERT (entendimento de texto), GPT (geração de texto), T5 (tradução e sumarização) e outros modelos generalistas treinados com uma quantidade massiva de dados, como GPT-3, LLaMA e Gemini (PAPAGEORGIOU et al., 2024). Devido ao vasto tamanho de dados utilizados no treinamento, os LLMs conseguem analisar e entender a complexidade de padrões linguísticos mais complexos e não são restritos à

base de treinamento por meio de técnicas de recuperação de informações no prompt. Por conta disso, são capazes de identificar indícios de falsidade nas notícias, independente do tempo, tornando-os valiosos na implementação de sistemas automatizados em detectar inconsistências e conteúdo enganoso (BOISSONNEAULT; HENSEN, 2024). Além disso, diversas abordagens recentes têm sido propostas para aprimorar a detecção de fake news. (LIAO et al., 2023) introduziu um framework que melhora a verificação de fatos através da recuperação de múltiplas evidências, combinando aprendizado profundo com técnicas de recuperação de informações. Essa abordagem torna o sistema mais transparente e confiável, pois fornece justificativas embasadas em fontes externas. No entanto, essa dependência pode introduzir viés e impactar o tempo de resposta do sistema. Já (HU et al., 2023) propõe um método de leitura dupla para verificação de fatos, onde o modelo revisita a evidência antes de tomar uma decisão, garantindo uma análise mais confiável e interpretável. Apesar das vantagens, essa abordagem exige maior capacidade computacional e pode não ser viável para aplicações em tempo real.

A partir disso, o trabalho propõe uma abordagem inovadora utilizando LLMs para detecção de notícias falsas na língua inglesa e brasileira. Inspirado no STEEL e aprimorado com a técnica HyDE, este trabalho propõe o STEEL_HyDE, um modelo que combina busca na web, recuperação de evidências e inferência com Large Language Models (LLMs) para avaliar a veracidade de notícias. O sistema inicia capturando evidências online através da Google Custom Search API, refinando os resultados com LLMs para reduzir ruído e aumentar a precisão. Para otimizar a recuperação, aplicamos HyDE, que cria uma paráfrase da notícia para melhorar capturar evidências com mais relevância a notícia. Entende-se por ”relevância” das evidências quando os pontos princípais que caracterizam a notícia estão sendo trazidos nas evidências, logo a evidência que possui mais informações sobre essas características da notícia, é a mais relevante. Essas evidências são então organizadas e inseridas em um prompt para análise do LLM, que classifica a notícia como verdadeira, falsa ou sem informação suficiente (NEI), garantindo transparência ao apresentar uma explicação fundamentada. O modelo Qwen2.5-7B-Instruct foi escolhido por equilibrar eficiência, custo e capacidade de contexto, permitindo inferências mais acessíveis e escaláveis. Realizamos uma comparação de desempenho entre nossa proposta com o que temos no

estado da arte de modelos machine learning e deep learning, além de comparação com outros resultados de desempenho com LLMs mais robustos. Por fim, apontamos fraqueza e vantagens de utilizar modelos com menos parâmetros, sugerindo outras análises e melhorias nos trabalhos futuros.

1.1 Objetivo Geral

Esse projeto visa criar uma estratégia que auxilia na detecção de notícias falsas. A estratégia consiste em receber um texto referente a notícia, analisada por um modelo LLM pequeno (entre 7 a 9 bilhões de parâmetros) mediante técnicas de engenharia de prompt e recuperação de informações disponíveis na internet, sendo a seleção dessas informações baseadas em um questionamento do próprio agente de “o que é relevante?” na análise e julgamento da notícia. No final da análise, o processo deve retornar uma saída fornecendo informações se a notícia é notícias falsas ou legítima, um texto explicando o que levou a classificação da notícia e o nível de confiabilidade referente a classificação.

1.1.1 Objetivos Específicos

Podemos dizer que alcançamos nosso objetivo, uma vez que conquistamos os seguintes tópicos:

- Verificar a capacidade de LLMs menores na tarefa de detecção de notícias falsas.
- Propor e implementar um método para escolha de um modelo LLM.
- Implementar um agente que fornece uma análise em relação à notícia, determinando se é falsa ou verdadeira com um nível de confiança.
- Realizar uma avaliação comparativa de desempenho da solução proposta em relação aos métodos existentes no estado da arte.

1.2 Contribuições

Modelos de machine learning e deep learning na detecção de notícias falsas dependem geralmente de grandes datasets e treinamento para classificação textual,

exigindo o uso extensivo de hardware. Essas abordagens não acompanham a evolução das notícias falsas e podem se tornar tendenciosas, pois classificam com base apenas no treinamento inicial, sem considerar informações externas. Além disso, enfrentam desafios como redução do tamanho do texto, remoção de stop words e perda de contexto, afetando a interpretação de nuances como ironia e sarcasmo. Outra limitação é a falta de transparência (black box), comprometendo confiabilidade e questões éticas (MOLNAR, 2020). Contribuimos na solução desses problemas pelas conquistas abaixo:

- Mitigação da desatualização de conhecimento: Nossa proposta soluciona essas limitações por meio de consultas via Web API para buscar evidências em tempo real, reduzindo a obsolescência dos dados e ampliando a análise contextual.
- Aumento no escopo de tamanho de texto: capacidade de tamanho do texto ao utilizar um modelo capaz de processar até 4096 tokens, permitindo maior compreensão sem cortes excessivos.
- Inclusão de julgamento de relevância sobre as evidências: introduzimos um novo processo de pensamento crítico que reduz a dependência dos algoritmos de relevância das Web Search APIs sobre o resultado.
- Redução do custo de hardware: diferente dos modelos tradicionais que exigem hardware robusto para treinamento, nossa abordagem aproveita modelos pequenos pré-treinados, tornando a solução mais acessível e escalável, além de ser mais eficiente que abordagens de prompt engineering com modelos maiores.

1.3 Organização do Trabalho

O trabalho está dividido em 5 capitulos: Capítulo 2 introduz os conceitos essenciais para entender o desenvolvimento do trabalho; Capítulo 3 aborda os trabalhos relacionados e artigos que possuem o mesmo objetivo ou ideias relacionadas para evolução na solução; Capítulo 4 descreve a metodologia utilizada para desenvolver o projeto; Capítulo 5 mostraremos nosso processo de experimentos e resultados; por fim no Capítulo 6 tratamos nossas conclusões e sugestões de trabalhos futuros.

2 Referencial Teórico

2.1 Processamento de Linguagem Natural (PLN)

Área que estuda a interação de humanos e computadores por meio da linguagem natural, possibilita que as máquinas compreendam, interpretem e gerem textos de maneira semelhante aos seres humanos. Para isso, o PLN combina técnicas de linguística computacional, estatística, machine learnign e deep learning para processar e analisar grandes volumes de dados textuais. Dessa forma, conseguimos desempenhar tarefas como: divisão de texto em unidades menores (tokenização), padronização das palavras para reduzir variações morfológicas (normalização), classificar o tom emocional de um texto (análise de sentimento), tradução e geração de texto.

2.2 Machine Learning

Um dos pilares no estudo no campo de inteligência artificial. São algoritmos com capacidade de imitar o comportamento humano inteligente, atravez do aprendizado de padrões e previsões a partir de dados referentes a um problema específico. Começa-se com um conjunto de dados (números, texto, fotos...), que são coletados e preparados para serem usados no treinamento do modelo. Durante o treinamento, o modelo fornecará respostas para cada informação, refinando suas respostas por meio de funções que ajustam seus parametros conforme o tempo. No final, o modelo poderá ser capaz de explicar o que aconteceu (função descritiva), prever o que acontecerá (função preditiva) e fazer sugestões sobre quais ações tomar (função prescritiva) (GOODFELLOW YOSHUA BENGIO, 2024). No geral, o aprendizado de máquina é dividido em:

- Aprendizado não-supervisionado: procura padrões em dados não rotulados, onde não há orientação explícita sobre as classes ou categorias dos exemplos. Seu objetivo é identificar padrões emergentes e estruturas subjacentes nos dados, podendo encontrar informações ou tendências que as pessoas não estão procurando explicitamente. Por exemplo, podemos analisar dados de vendas on-line e identificar diferentes tipos de clientes que fazem compras.

- Aprendizado supervisionado: treinados com conjuntos de dados rotulados, que permitem que os modelos aprendam e se tornem mais precisos ao longo do tempo. Envolve o ajuste das suas previsões para cada entradas por meio feedbacks com base nos rótulos reais, permitindo a generalização do modelo sobre o conhecimento adquirido. O objetivo é aprender como os dados se relacionam para fazer previsões ou classificações. Por exemplo, um algoritmo seria treinado com fotos de cães e outras coisas, todas rotuladas por humanos, e a máquina aprenderia maneiras de identificar fotos de cães por conta própria.

- Aprendizado semi-supervisionado: diferente da aprendizagem supervisionada (requer um grande volume de dados rotulados) e da não supervisionada (trabalha exclusivamente com dados não rotulados), a abordagem semi-supervisionada utiliza um pequeno conjunto de dados rotulados combinado com um grande volume de dados não rotulados para melhorar a eficiência do aprendizado. Dessa forma, mitigamos a dificuldade e o alto custo associados à rotulagem manual de grandes volumes de dados.

2.3 Embeddings para Linguagem Natural

Embeddings de palavras é considerado o centro das tarefas PLN, consiste em formas de representações criadas por métodos estatísticos ou modelos linguísticos, que leva o entendimento de linguagem para a máquina. A partir de treinamentos por grandes corpus de texto, conseguimos transformar palavras em vetores distribuidos que capturam a semantica dos textos. Entretanto, possuem limitações quanto a captura de informações semanticas e contexto, nos últimos anos essas limitações têm sido mitigadas pelos métodos: Bag-of-words, TF, TF-IDF, Word2Vec, Glove e BERT.

O problema é dado um corpus com n documentos $S = \{d_1, d_2, ..., d_n\}$, cada documento consistem em m palavras $W = \{w_1, w_2, ..., w_m\}$ com um vocabulário V. A representação de um vector de embedding $\vec{w_i}$ é definido pelo mapeamento de cada palavra w_i em um documento d_i no espaço continuo $\mathbb{R}^d$, onde d é a dimensão do espaço do vetor, demonstrado pela Equação 2.1 abaixo:

$$w_i \rightarrow \vec{w_i}, \vec{w_i} \in \mathbb{R}^d \tag{2.1}$$

2.3.1 Deep Learning

É um subcampo de machine learning que aplica um modelo matemático inspirado no funcionamento do cérebro humano (rede neurais) para resoluções de problemas, sem a necessidade de dados instruídos manualmente. Essas redes possuem múltiplas camadas para aprender padrões diretamente a partir de grandes volumes de dados. O neurônio artificial transmite informações para a próxima camada da rede, transformando a entrada em representações cada vez mais abstratas, permitindo que o modelo aprenda características complexas. Por conta dessa evolução, conseguimos avançar em aplicações como reconhecimento de imagens, processamento de linguagem natural, geração de texto, entretanto seu comportamento de ”caixas-pretas”e a necessidade de hardwares mais robustos tem se tornado um desafio na comunidade acadêmica.

2.3.2 Arquitetura Transformers

Inspirada por sistemas biológicos humanos, a intuição por trás do mecanismo de atenção assume que apenas partes de um texto são relevantes, onde dado um texto, humanos conseguem identificar de forma seletiva o que é mais vital e relevante no contexto, enquanto ignoram o que é informação irrelevante.

Introduzida por (VASWANI et al., 2017), revolucionou o campo das PLN por conta do paralelismo e eficiência no aprendizado de sequências. Composto por duas partes principais, encoder (processa a entrada e gera representações contextuais dos tokens) e decoder (usa as representações para gerar a saída, sendo particularmente útil em tarefas como tradução automática). Os blocos-chave na sua arquitetura são: Self-Attention responsável pela avaliação da importância de cada palavra em relação a todas as outras em uma sequência, capturando dependências de curto e longo alcance de maneira eficiente; Positional Encoding adiciona informações sobre a posição dos tokens na sequência; Camadas Feed-Forward aplica transformações não lineares nos embeddings das palavras para capturar padrões complexos; Mecanismo de Atenção Multi-Head que expande a capacidade do modelo ao permitir múltiplos focos de atenção simultâneos sobre diferentes partes do texto; Normalização e Dropout para melhorar a estabilidade do treinamento e reduzir o overfitting.

2.3.3 Familia BERT

Antes da aparição do BERT (Bidirectional Encoder Representations from Transformers), os modelos de processamento de linguagem natural usavam arquiteturas unidirecionais ou bidirecionais restritas, isso limitava a capacidade do modelo de capturar dependências contextuais completas entre palavras em uma sentença, podemos citar o ELMO e GPT como exemplo. O BERT supera essas limitações, pois implementa os Transformers bidirecionais em todas as camadas, possibilitando a consideração do contexto completo de uma palavra, podendo ser da esquerda para a direita ou vice-versa (DEVLIN, 2018). O BERT usa apenas a parte do codificador para modelar a linguagem bidirecionalmente, sendo o Transformer a base da sua implementação. O mBERT (Multilingual BERT) é uma extensão do modelo BERT (Bidirectional Encoder Representations from Transformers), desenvolvido pela equipe do Google AI Language, usando 104 idiomas diferentes no seu pré-treinamento. O principal foco foi em atingir uma capacidade maior para neutralidade linguística, em outras palavras, a habilidade do modelo de não se sobrepor a especificidades de um idioma durante a generalização para outros (PIRES, 2019).

(LIU, 2019) foi responsável pela RoBERTa, otimizada a partir do modelo BERT por meio de maior volume de dados, mascaramento dinâmico por época de treinamento, batch maior e mais iterações, com remoção do Next Sentence Prediction utilizado no BERT. Mantendo a arquitetura e apenas com mudanças no treinamento, foi possível desempenhar melhor nas tarefas de GLUE benchmark comparado com o BERT. (SOUZA; NOGUEIRA; LOTUFO, 2020) criou uma versão do BERT ajustada para a lingua portuguesa, o BERTimbau. Treinado a partir de fontes como Wikipedia, livros e notícias, esta versão oferece melhor desempenho em tarefas NLP comparado com o mBERT para a língua portuguesa, pois este não é otimizado para um idioma específico e pode ter desempenho inferior em determinadas tarefas. O treinamento seguiu da mesma forma que o BERT original e seus resultados demonstraram ganhos significativos de desempenho para PLN na lingua portuguesa. Os autores disponibilizaram duas versões, BERTimbau Base (110 milhões de parâmetros) e BERTimbau Large (340 milhões de parâmetros).

2.4 Large Language Models

Reivindicaram os sistemas de inteligência artificiais, por serem capazes de processar, gerar texto com coerência e relevância próximo ao desempenho humano em diversas áreas, como: tradução, sumarização e recuperações de informações (NAVEED et al., 2023). A arquitetura Transformers permitiu o processamento eficiente de sequências de dados, capturando dependências de longo alcance no texto. Com a evolução na capacidade computacional e acesso a fontes de dados gigantes, o desenvolvimento dos LLMs foi impulsionado a medida que escalamos o treinamento de modelos utilizando uma enorme quantidade de parâmetros e volumes de texto, desde livros a websites (NAVEED et al., 2023). Podemos dividir as LLMs em:

- Pre-trained LLMs: Treinados inicialmente em vastas quantidades de dados não rotulados (livros, artigos, páginas da web, etc.) para aprender os padrões, a estrutura e o significado da linguagem. O treino é focado em tarefas gerais, como previsão de próximas palavras ou preenchimento de lacunas em frases.
- Fine-Tuned LLMs: São resultados de um segundo treinamento nos Pre-trained LLMs. Utilizando um conjunto de dados menores específico para uma tarefa, o treino realiza um ajuste fine no parametros do modelo para torna-lo mais eficiente na tarefa proposta pelo dataset.

2.4.1 Prompt Engineering

É uma técnica essencial para melhorar a desempenho do modelo sem necessidade de ajustes em seus pesos ou re-treinamento. Envolve formular estrategicamente as entradas ou instruções, otimizando os prompts (comandos que enviamos para o modelo) para modelar e projetar como o modelo responderá. (SAHOO et al., 2024) destrincha derivações de práticas de prompt enginerring, dentre essas, as principais e mais comuns são:

- Zero-Shot Learning: O modelo recebe um prompt sem exemplos anteriores da tarefa e deve gerar a resposta diretamente com base no conhecimento adquirido durante o treinamento. São os mais comuns no dia a dia, levam geralmente a

respostas menos precisas quando utilizado em tarefas complexas e economiza na quantidade de tokens.

- Few-Shot Learning: O modelo recebe alguns exemplos antes de ser solicitado a realizar a tarefa, permitindo que aprenda o padrão e forneça respostas mais precisas. O desempenho do modelo retorna respostas mais precisas sem necessidade de novos treinamentos, mas também possui um aumento no custo computacional por conta do aumento no número de tokens.

- Chain of Thought (CoT): O modelo é instruído a dividir o raciocínio em etapas antes de chegar à resposta final, o que melhora o desempenho em tarefas que exigem lógica e interpretação aprofundada. Aumenta o tempo de resposta e custo computacional, mas há um ganho em coerência e precisão nas suas respostas, especialmente em problemas de raciocínio lógico.

Na Figura 2, demonstramos exemplos do funcionamento das técnicas comentadas acima.

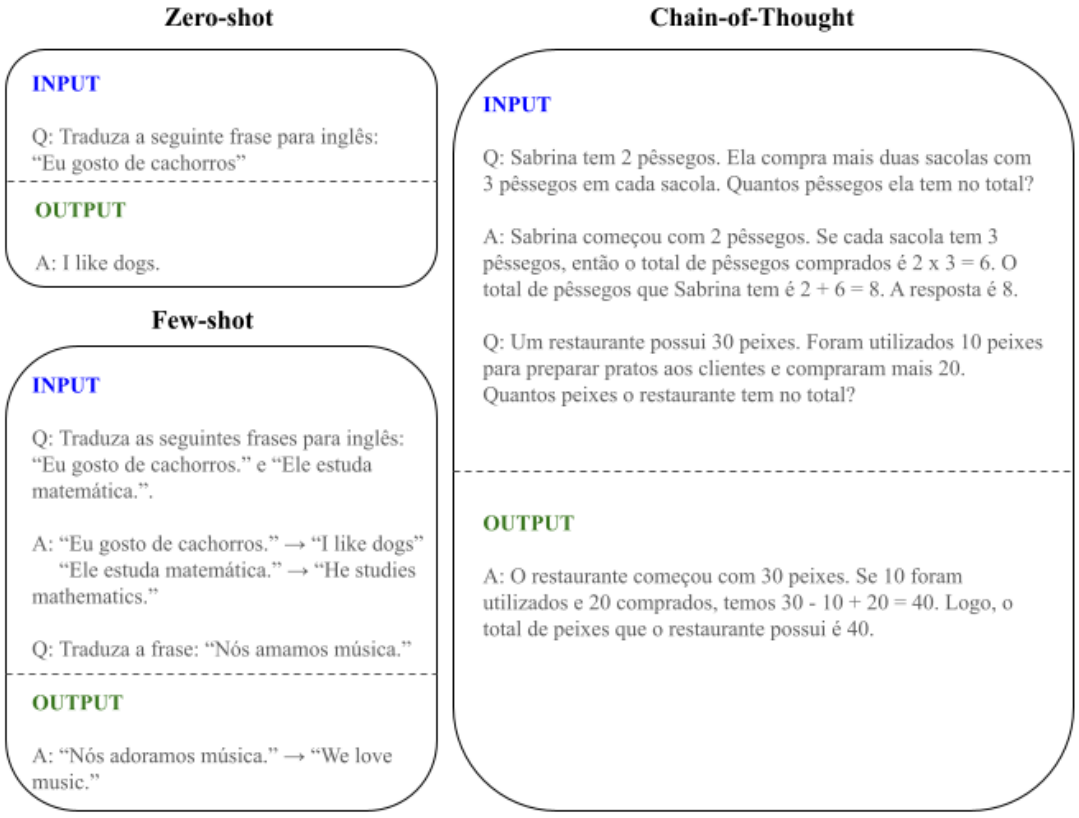

Figura 2 – Exemplos do funcionamento de prompts Zero-shot, Few-shot e Chain-of-Thought. Fonte: Autoria própria

,

2.4.2 Fine-tuning

É uma técnica no aprendizado de máquina que permite uma adaptação de um modelo pré-treinado a uma atividade específica. Consiste em reutilizar o conhecimento adquirido pelo modelo em um campo de conhecimento e afunilar esse conhecimento para torná-lo mais especialista em uma tarefa específica. É feita uma inicialização de um modelo pré-treinado com pesos e parâmetros definidos para uma tarefa geral, por exemplo, processamento de linguagem natural; em seguida, esse mesmo modelo é ajustado por meio de um treinamento, utilizando um conjunto de dados menor com natureza que remete a uma tarefa específica, por exemplo, avaliar a satisfação de usuários por meio de comentários sobre um produto. Desse modo, economizamos tempo e recurso computacional, aproveitando a capacidade de um modelo generalizado e evitando o processo de treinar um modelo do zero (PARTHASARATHY et al., 2024). No campo das LLMs, (NAVEED et al., 2023) lista diferentes tipos de fine-tuning:

- Transfer Learning: Tornar um LLMs pre-treinado especialista em tarefas específicas.
- Instruction-tuning: habilita o modelo a responder de forma efetiva as consultas feitas pelo usuário. A ideia é tornar o LLMs generalista para obedecer instruções, por meio da generalização de exemplos de um dataset com prompt input-output de instruções.
- Alignment-tuning: treina o modelo para torná-lo prestativo, honesto e inofensivo, caracterizado pela sigla ”HHH”: helpful, honest and harmless. Consiste em utilizar o feedback humano para alinhar o modelo aos valores éticos humanos, pedindo ao LLMs gerar respostas inesperadas e ajustar os parâmetros para evitar essas respostas.

2.4.3 Sistemas RAG

Introduzido por (LEWIS et al., 2020), Retrieval-Augmented Generation surgem como uma solução para melhorar a capacidade dos LLMs ao integrar um mecanismo de recuperação de informação com geração de texto. A partir disso, conseguimos introduzir informações atualizadas ao modelo, sem a necessidade de novos treinamentos

que custariam tempo e recursos, além de permitir flexibilidade e escalabilidade para as IA. Essa abordagem visa mitigar problemas como alucinações e desatualização de conhecimento nos modelos, demonstrado na Figura 3, garantindo que as respostas geradas sejam mais precisas e baseadas em fatos. (HUANG; HUANG, 2024) comenta que existem três partes fundamentais no funcionamento do RAG: Indexamento, Retrieval e Generation.

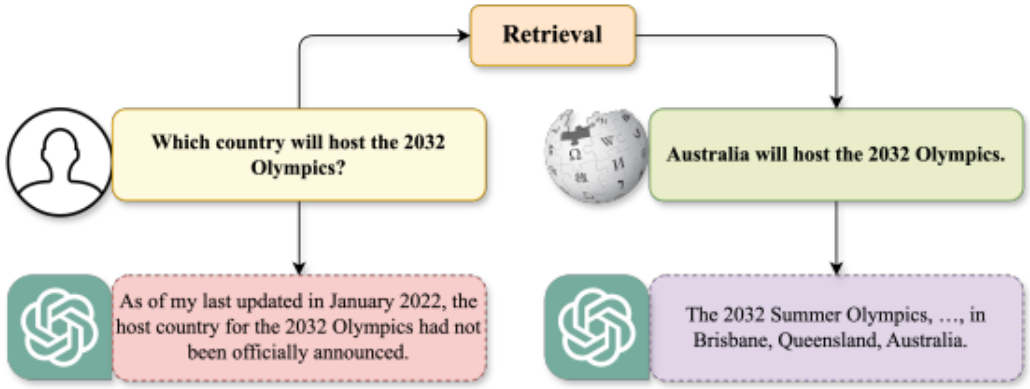

Figura 3 – Demonstração da utilização do RAG pelo ChatGPT para resolver problemas que vão além da sua época de treinamento. Fonte: (HUANG; HUANG, 2024)

,

Indexamento é responsável por preparar o texto para recuperação, como tokenização, derivação e remoção de palavras irrelevantes. Segmentos de texto são organizados em setenças para facilitar a procura por documentos compatíveis. Os modelos pretreinados como BERT, revolucionaram como é feito o indexamento por meio de representação de textos por vetores semânticos, pois uma vez armazenados possibilitam uma rápida e precisa recuperação.

Retrieval recupera documentos no local de armazenamento das sentenças ou vetores para o uso posterior na geração de texto. A forma mais tradicional é ordenar os documentos através da frequência de termos e presença, mas as estratégias mais recentes utilizam distâncias de vetores para capturar documentos por similaridade, evitando omitir perda de semântica nas consultas, desafio enfrentado pela forma tradicional.

Generation envolve produzir texto que seja relevante e reflete as informações encontradas nos documentos recuperados. O método comumente utilizado é concatenar a consulta com as informações recuperadas, posteriormente alimentadas em um

LLM para geração de texto. O texto gerado pelo RAG deve ser preciso em cobrir as informações dos documentos recuperados e estar alinhado com a intenção da consulta.

2.5 Métodos de Classificação

- Regressão Logística: Utiliza a função sigmoide (mapeia qualquer valor de entrada para um valor entre 0 e 1, sendo útil para calcular probabilidades) para calcular a probabilidade de cada caso e baseando-se em um limiar, é atribuído o caso a uma das classes da classificação binária.

- Random Forest: Consiste em ter um conjunto de árvores de decisão individuais, treinadas de forma independente, logo em seguida, suas previsões são combinadas para um resultado, dessa forma prevenimos o overfitting usando várias árvores como média. Na divisão das árvores, é diminuída a correlação entre elas e define-se um subconjunto aleatório das características para ditar cada divisão.

- K-Nearest Neighbor (KNN): Esse algoritmo baseia-se em pontos de dados semelhantes, calculando os pontos mais próximos com base em uma distância atribuída por um método matemático, por exemplo: Distância Euclidiana, Distância de Manhattan, Distância de Minkowski e Distância Cosine.

- Linear SVC: Inspirado no algoritmo de Support Vector Machine (SVM), esse algoritmo calcula hiperplano para separar diferentes classes. Dependendo do cenário, o hiperplano pode se comportar como linha (o problema é caracterizado por duas dimensões), plano (o problema é em três dimensões) ou como próprio hiperplano (dimensões superiores a três). É feita uma busca para encontrar o hiperplano que maximiza a margem entre duas classes, sendo a margem a distância entre o hiperplano e os vetores de suporte (os pontos de dados mais próximos de cada classe)

- Máquina de Vetores de Suporte: Diferente do Linear SVC, apesar dos dados não serem linearmente separáveis, este algoritmo consegue fazer a classificação mapeando os pontos de dados para um espaço dimensional superior, dessa forma eles podem ser separados pelo hiperplano. O hiperplano tem como fórmula $w.x + b = 0$, para um conjunto de dados $\{(x_i, y_i)\}$, onde x_i são os vetores

de características e y_i são as etiquetas de classe de valores $+1$ ou -1, visando encontrar um vetor de pesos w e um termo que tende a b.

- Multinomial Naive Bayes: É um algoritmo que faz parte dos classificadores Naive Bayes. Seu funcionamento envolve assumir que os dados seguem uma distribuição multinomial, o que facilita na contagem de dados, como frequência de palavras em um texto. Todo o algoritmo gira em torno do Teorema de Bayes, que relaciona as probabilidades condicionais de eventos, dado como:

$$P(C_k|x) = \frac{P(C_k) \cdot P(x|C_k)}{P(x)} \tag{2.2}$$

  - $P(Ck|x)$ é a probabilidade posterior da classe Ck dado o vetor de características x
  - $P(Ck)$ é a probabilidade a priori da classe Ck
  - $P(x|Ck)$ é a probabilidade de observar o vetor de características x dado que a classe é Ck
  - $P(x)$ é a probabilidade de observar o vetor de características x

- XGBoost: É uma implementação avançada do algoritmo de Gradient Boosting Machine (GMB), que combina vários modelos fracos para formar um modelo forte, de forma que o modelo fraco corrige os erros do seu antecessor, melhorando a cada iteração. Inicia-se com um modelo simples, como, por exemplo, uma distribuição de probabilidade uniforme e a cada interação, seu modelo sucessor é treinado para prever os seus erros, assim sendo adicionado aos conjuntos de modelos. Esse algoritmo também utiliza uma função de perda diferencial e regularização para evitar o overfitting.

- Rede Neural Convolucional (CNN): Tem uma aplicação mais focada em visão computacional, porém por ser boa em detectar padrões, é viável utilizar em dados de texto. A CNN recebe uma imagem como entrada, aplica-se uma convolução na imagem produzindo mapas de ativação, resumindo-se a filtros que detectam diferentes características e após os passos da ativação, pooling, flattening, temos o processamento dessas características, produzindo uma saída final em forma de probabilidade para cada classe. Com o word embedding, conseguimos

representar cada palavra por um vetor em um espaço vetorial; assim capturamos os relacionamentos e similaridades entre as palavras. Vale salientar que por ser texto, a convolução ocorre em uma dimensão

- Long Short-Term Memory: É um tipo de rede neural recorrente (RNN), frequentemente utilizada em dados de sequência, como processamento de texto, tradução automática, reconhecimento de voz e séries temporais. Seu algoritmo consiste em receber um vetor de entrada x_t no tempo t, passando por três componentes: Porta de Entrada (controla quais valores antigos serão repassados adiante), Porta de Esquecimento (decide quais informações anteriores são descartados, mantendo apenas informações relevantes ao longo do tempo), Porta de Saída (controla os dados que vão influenciar as previsões ou decisões). Estes três portões, com a célula de memória (responsável por armazenar informações por longos períodos), é o que compõe a célula LSTM.

- Graph Neural Networks (GNNs): São uma classe de modelos de aprendizado profundo projetados para trabalhar com dados estruturados em grafos. Diferentemente das abordagens tradicionais de aprendizado de máquina, que tratam os dados como tabelas ou sequências, os GNNs são capazes de capturar relações complexas e dependências estruturais entre entidades representadas como vértices e arestas em um grafo. Consiste em um processo iterativo de propagação de informação, conhecido como ”message passing”, no qual cada nó recebe e agrega informações de seus vizinhos ao longo de várias camadas.

2.6 Métodos de Avaliação

- Matriz de Confusão: É uma ferramenta para avaliar o desempenho de modelos focados em classificação. É feita uma representação tabular com quatro elementos: True Positive (TP), False Positive (FP), True Negative (TN) e False Negative (FN), mostrando não apenas a quantidade de acertos, mas também os erros cometidos pelo modelo. A matriz é especialmente útil para problemas de classificação binária, porém também pode ser aplicada para problemas de classificação multiclasse.

	Predito Positivo	**Predito Negativo**
Real Positivo	True Positive (TP)	False Negative (FN)
Real Negativo	False Positive (FP)	True Negative (TN)

Tabela 1 – Representação de uma matriz de confusão

- Acurácia: É uma medição da proporção de exemplos classificados corretamente pelo modelo em relação ao total de exemplos no conjunto de dados. Dependendo dos dados, essa métrica pode ser enganosa, por exemplo, se não existir um equilíbrio no conjunto dos dados, a distribuição das classes não é uniforme, impactando no cálculo da acurácia. Podemos expressar a fórmula da seguinte forma:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

  Calcular apenas a acurácia não é adequado, pois um modelo pode alcançar uma alta acurácia ao classificar predominantemente a classe majoritária, mas ainda assim falhar em identificar a classe minoritária.

- Precisão: É uma medição da proporção de exemplos classificados como positivos corretamente em relação ao total de exemplos classificados como positivos, de forma mais simples, quando estamos avaliando a precisão nos perguntamos "Entre todas as instâncias previstas como positivas, quantas são realmente positivas?". É especialmente útil quando o foco está na minimização de falsos positivos, ou seja, quando é crucial evitar a classificação incorreta de exemplos negativos como positivos. Podemos expressar a fórmula da seguinte forma:

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

  Ter uma alta precisão indica que o modelo tem uma baixa taxa de falsos positivos, ou seja, a maioria dos exemplos classificados como positivos são realmente positivos. Entretanto, a precisão não considera os falsos negativos e, portanto, a precisão sozinha não fornece uma imagem completa do desempenho do modelo.

- Recall: Também conhecido como sensibilidade ou taxa de verdadeiros positivos. É uma métrica de avaliação focada na capacidade do modelo de identificar corretamente todas as instâncias positivas. O recall é particularmente importante em

situações onde perder uma instância positiva (ou seja, ter um falso negativo) é mais prejudicial do que classificar incorretamente uma instância negativa como positiva (FP). Podemos expressar a fórmula da seguinte forma:

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

Calcular apenas o recall não fornece uma imagem completa do desempenho do modelo, pois não leva em consideração os falsos positivos.

- F1-score: É uma métrica que combina as informações de precisão e recall para fornecer uma medida geral do desempenho do modelo, equilibrando a capacidade de classificar corretamente exemplos positivos e negativos. Essencial para se utilizar em cenários onde há um desequilíbrio entre as classes, pois considera tanto os falsos positivos quanto os falsos negativos, lidando com situações em que uma alta precisão pode ser acompanhada de um baixo recall e vice-versa. Isso oferece uma visão mais abrangente da qualidade do modelo, pois os modelos são penalizados nessa métrica quando possui um resultado ótimo em uma métrica, porém péssimo na outra. Duas maneiras de calcular são demonstradas pela equação 2.6:

$$F_1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.6}$$

Essa métrica é recomendada para ser utilizada em situações quando a distribuição das classes é desigual, os domínios dos falsos positivos quanto os falsos negativos têm custos significativos e procura de ter eficácia na classificação pelo modelo.

- F1-Macro: Calcula a média aritmétrica simples do F1-score de cada classe. Dessa forma, avaliamos o desempenho geral em problemas com classes desbalanceadas, pois atribuimos o mesmo peso para todas as classes. A equação 2.7 desmonstra o processo para calcular o F1-Macro, sendo N o número de classes e F1-score$_i$ é o F1-score de cada classe.

$$F_1Ma = \frac{1}{N} \sum_{i=1}^{N} \text{F1-score}_i \tag{2.7}$$

- F1-Micro: Propõe uma avaliação considerando TP, FP e FN de todas as classes, calculando a média global. Dessa forma, avaliamos o desempenho geral do modelo tratando as classes de forma agregada, e por conta disso, o resultado é mais influenciado pelas classes majoritárias. Calculamos usando a equação 2.8

$$F_1Mi = \frac{2 \times TP_{total}}{2 \times TP_{total} + FP_{total} + FN_{total}} \tag{2.8}$$

3 Trabalhos Relacionados

3.1 Abordagens

Nos últimos anos, diversos estudos propuseram abordagens inovadoras para aprimorar a detecção de notícias falsas, desde técnicas de aprendizado de máquina, deep learning e, mais recentemente, modelos LLMs (na Tabela 2 compilamos os trabalhos que serão comentados). Nos modelos tradicionais, a tarefa de classificação de textos enganosos mostrou-se com potencial utilizando técnicas como o modelo híbrido CNN-RNN proposto por (NASIR; KHAN; VARLAMIS, 2021), uso de evidências através de uma fonte padrão externa (LIAO et al., 2023) e revisitar as evidências caso sejam insuficientes para avaliar a notícia (HU et al., 2023). Com a adição dos LLMs, conseguimos demonstrar a capacidade dos LLMs de entendimento de contexto para detecção de notícias falsas, por propostas inspiradas nas mesmas técnicas, WebGLM com recuperação de evidências (LIU et al., 2023) e ProgramFC (PAN et al., 2023).

(WU et al., 2021) basea-se em redes de atenção interativa hierárquicas e sensíveis a evidências. Consiste em combinar múltiplos níveis de atenção para capturar relações contextuais entre alegações e evidências, aprimorando a interpretabilidade da decisão ao atribuir pesos diferenciados nas diferentes partes do texto e evidências associadas. Dessa forma, é possível recuperar informações em diferentes granularidades, desde palavras até sentenças completas, além de sugerir explicações para entender jquais evidências foram mais relevantes para a verificação da alegação.

Semelhante ao anterior, (VO; LEE, 2021) se inspira em redes neurais atencionais multi-head hierárquicas. A proposta é melhorar a capacidade de diferenciação entre informações verdadeiras e falsas ao explorar múltiplas fontes de evidência e suas inter-relações, por meio da captura múltiplos padrões de relevância dentro do texto, interações entre as evidências de forma mais organizada e combinação de informações textuais e metadados.

(WU et al., 2023) combina redes neurais em grafos (GNNs) e aprendizado contrastivo adversarial, com os nós representando alegações, documentos de suporte e

fontes, enquanto as arestas capturam relações semânticas e de credibilidade entre essas entidades. Dessa forma, a falta de generalização diante de ataques adversariais e da incapacidade de lidar com a estrutura complexa das informações online pelos modelos tradicionais é mitigada, superando modelos convencionais de detecção de notícias falsas, especialmente em cenários onde há manipulação intencional das evidências.

(NASIR; KHAN; VARLAMIS, 2021) usa a CNN para extrair características representativas dos textos, enquanto a RNN lida com a dependência temporal e semântica das palavras. A combinação das arquiteturas permite capturar padrões locais e de longo alcance no texto, melhorando a precisão da detecção. Entretanto, depende de um grande volume de dados rotulados para treinamento e apresenta dificuldades na generalização para diferentes domínios de notícias falsas, o que pode reduzir sua eficácia em ambientes com novas fontes de desinformação.

(LIAO et al., 2023) introduz um framework que aprimora a detecção de notícias falsas por meio da recuperação de múltiplas evidências. Ele combina modelos de aprendizado profundo com técnicas de recuperação de informações para verificar a veracidade de uma notícia com base em fontes externas. A abordagem melhora a interpretação dos resultados ao fornecer justificativas baseadas em evidências externas, tornando o sistema mais transparente e confiável. Além disso, sua estrutura modular permite integração com diferentes modelos de verificação. A dependência de fontes externas pode introduzir viés e inconsistências, além de impactar o tempo de resposta do sistema, tornando-o mais lento do que modelos puramente baseados em texto.

(HU et al., 2023) apresenta um método baseado em leitura dupla para verificação de fatos, onde o modelo revisita a evidência antes de tomar uma decisão. Essa estratégia é inspirada no comportamento humano ao verificar informações. Demonstra melhorias na interpretabilidade dos modelos de verificação e reduz o risco de erros causados por informações parciais ou descontextualizada, garantindo que a decisão final seja mais confiável. Todavia, requer maior capacidade computacional e tempo de inferência, pois envolve múltiplas passagens sobre o mesmo conjunto de evidências, o que pode limitar sua aplicabilidade em sistemas de tempo real.

(LIU et al., 2023) combina um LLM otimizado com acesso à web para melhorar a precisão na resposta a perguntas. O modelo busca evidências em fontes online antes

de gerar uma resposta, reduzindo a alucinação de informações. A recuperação de informações em tempo real aprimora a precisão e a atualidade das respostas, tornando o modelo mais útil em responder questões do usuário. A dependência de informações externas pode tornar o modelo vulnerável as fontes não confiáveis, além de aumentar o tempo de resposta e os custos computacionais.

(PAN et al., 2023) apresenta um pipeline de verificação de fatos para LLMs, que permite corrigir saídas errôneas dos modelos de geração de texto. A abordagem utiliza um sistema de fine-tuning combinado com verificadores externos. Reduz a propagação de desinformação gerada por LLMs, tornando os modelos mais confiáveis e adequados para tarefas sensíveis. O processo de fine-tuning e validação pode ser computacionalmente caro, e a abordagem ainda depende da qualidade das bases de dados utilizadas para treinamento e verificação.

(JIN et al., 2024) combina o processamento de texto e imagem para identificar padrões de desinformação em notícias, explorando como imagens podem ser manipuladas para enganar leitores. O modelo utilizado é ajustado com técnicas de aprendizado contrastivo e raciocínio baseado em evidências, permitindo uma análise mais aprofundada das relações entre o conteúdo textual e visual. Ao incorporar texto e imagem na sua análise, generalizamos para diferentes tipos de notícias falsas, expandindo a quantidade que podem ser analisadas e superando abordagens baseadas apenas em texto (possibilita-se identificar manipulações visuais associadas à desinformação). Contudo, modelos multimodais exigem recursos significativos para treinamento e inferência, dependendo de um grande volume de dados rotulados corretamente, o que pode ser um desafio para algumas aplicações escalarem.

(LI et al., 2024) propôs um framework automatizado para detecção de notícias falsas que combina facilidade de uso e interpretabilidade. A estrutura alavanca os recursos de raciocínio e estimativa de incerteza dos LLMs, oferecendo recuperação de evidências mais robusta. Ela também contorna as limitações de depender de um corpus solitário predefinido ao obter evidências diretamente da Internet expansiva.

Nossa proposta se diferencia das seis primeiras por utilizamos LLMs e recuperamos evidências em tempo real, em vez dos modelos tradicionais e fontes estáticas. A estrutura foi inspirada no (LI et al., 2024), por conta da captura de evidências via

Web API e os parâmetros baseados nos resultados apresentados no trabalho, com a diferença de introduzir um passo de relevância das evidências com base na sumarização, semelhante sumarização de texto apresentado por (LIAO et al., 2023); e uso de um LLM menor comparado com GPT-3.5-Turbo. Diferente de (JIN et al., 2024), que analisa imagem e texto, focamos apenas na análise de notícias em texto.

Autor	**Abordagem**	**Dataset**	**Metodologia**	**Resultado**
(NASIR; KHAN; VARLAMIS, 2021)	Hybrid CNN-RNN	FA-KES/ ISOT	Extração de características do texto e aprendizado de dependências de longo prazo	Melhor desempenho, mas overfitting no ISOT e generalização ruim no FA-KES.
(WU et al., 2021)	EHIAN	PolitiFact/ Snopes	Redes de atenção interativa hierárquicas e sensíveis a evidências	Melhoria no F1-Ma de 1.8% e 1.3%.
(VO; LEE, 2021)	MAC	PolitiFact/ Snopes	Redes neurais de atenção multi-head hierárquicas	Aumento no AUC de 6.1% e 7.9%.
(WU et al., 2023)	GETRAL	PolitiFact/ Snopes	GNNs e aprendizado contrastivo adversarial	Melhoria no F1-Ma e F1-Mi de 1.5%.
(LIAO et al., 2023)	MUSER	PolitiFact/ GossipCop/ Weibo	Sumarização do texto e raciocínio multi-passos	Melhoria no F1-Macro e F1-Micro de 3%.
(HU et al., 2023)	Read it Twice	CHEF	Critérios de plausibilidade, plenitude e suficiência na captura de evidências	Aumento de 4.30% no F1-Micro e 4.32% no F1-Macro.
(LIU et al., 2023)	WEBGLM	WebGLM-QA	Sistema de QA aprimorado com Web e alinhado a preferências humanas	Melhor desempenho em fluência, veracidade e baixa redundância.
(PAN et al., 2023)	ProgramFC	HOVER/ FEVEROUS-S	Few-shot e raciocínio multi-passos	Melhoras de 14.77% para few-shot com 4 exemplos.
(JIN et al., 2024)	M-DRUM	HFFN	Extração de informações multimodais	Supera modelos SOTA e principais LVLMs.
(LI et al., 2024)	STEEL	PolitiFact/ LIAR/ CHEF	Extração de evidências via Web API e prompt engineering	Aumento de 5% no F1-Macro e F1-Micro.
Nossa Proposta	STEEL_HyDE	LIAR/ Fake.Br Corpus	Extração de evidências via LLM e Web API com raciocínio em prompt engineering	Melhoras de 14% em R-F e 2% em P-T.

Tabela 2 – Compilação de trabalhos relacionados

3.2 Sistemas Web para detecção de notícias falsas

No mundo real, temos alguns sites e sistemas disponíveis que auxiliam no combate a notícias falsas, desenvolvidos por organizações, fundações de pesquisa e desenvolvimento, tais como: FactCheck.org (mantido pela Universidade da Pensilvânia, com foco na verificação de discursos políticos e desinformação) e Aos Fatos (plataforma brasileira que verifica declarações políticas e notícias falsas compartilhadas nas redes sociais), ambas com proposta semelhante ao Scopus e Politifact.

Na Figura 4, 5, 6, mostram-se três sistemas desenvolvidos pela Google, (Núcleo Interinstitucional de Linguística Computacional (NILC), 2025) e (MARINOVA et al., 2020), respectivamente.

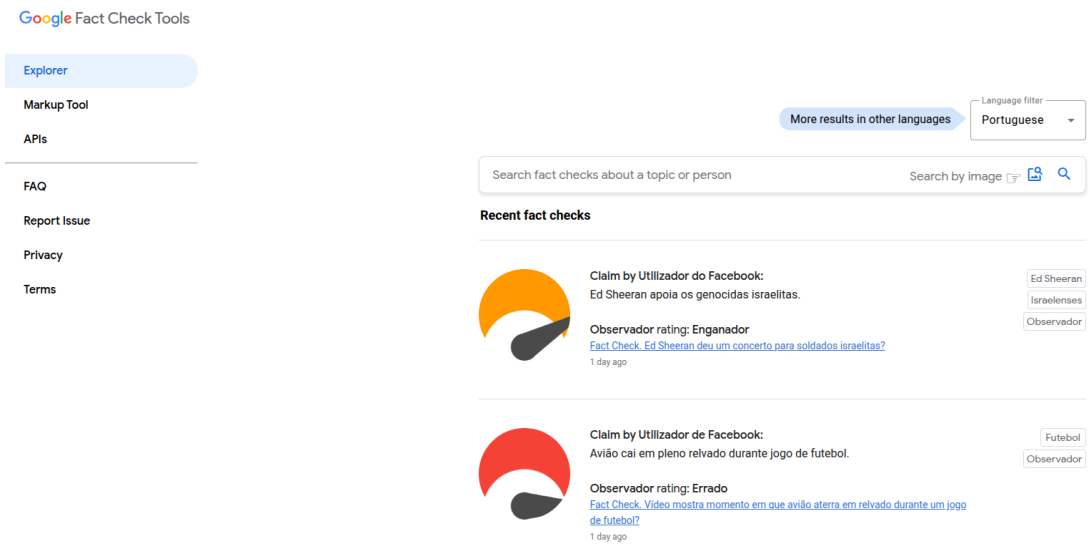

Figura 4 – Captura da tela da ferramenta do Google que agrega verificações de diversas agências e permite buscas sobre a veracidade de informações. Visa facilitar o trabalho de verificadores de factos, jornalistas e investigadores. Disponível em: Google Fact Check Tools

,

Figura 5 – Captura de tela do FakeCheck: sistema aplica métodos para extrair atributos linguísticos do texto e os utiliza em um modelo de aprendizado de máquina, que classifica a notícia como verdadeira ou falsa. O texto deve ter pelo menos 100 palavras, pois o sistema foi ”treinado”dessa forma. Os modelos disponibilizados foram treinados com o córpus Fake.Br.Estão disponíveis dois modelos de detecção: ”Palavras do Texto”e ”Classes Gramaticais”. Disponível em: Fake Check

,

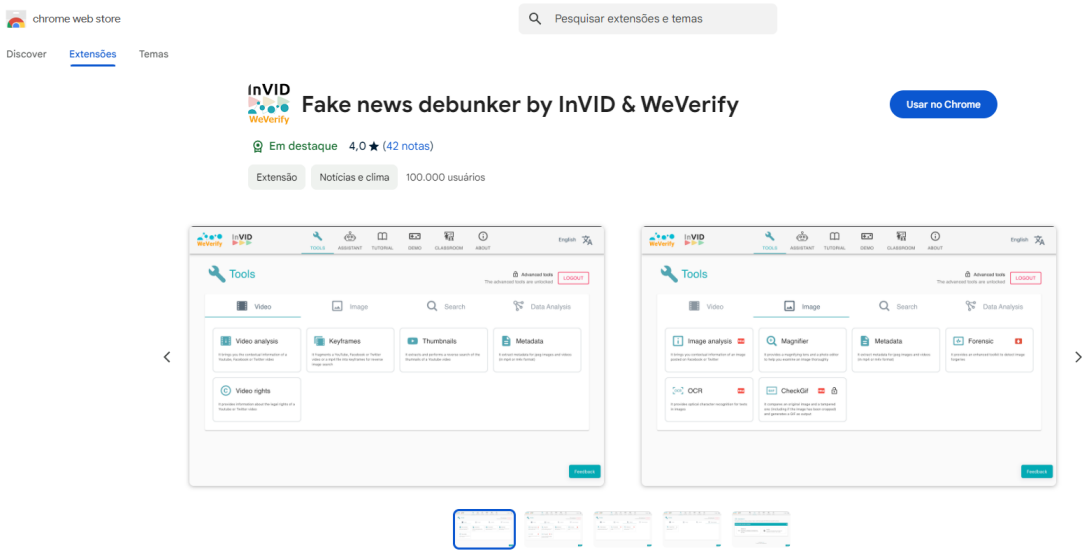

Figura 6 – Captura de tela da extensão da plataforma WeVerify. Nela é possível fazer análise de imagens, verificação de vídeos, análise de conteúdo textual, rastreamento de conteúdo, detecção de bots e perfis suspeitos. Algumas funções mais avançadas estão disponíveis apenas para jornalistas, pesquisadores e pessoas que trabalham com verificação de fatos.Disponível em: chromewebstore.google.com

,

4 Proposta

Inspirada no STEEL (LI et al., 2024), junto com a incrementação da técnica HyDE (GAO et al., 2022), nomeamos nossa proposta pela sigla STEEL_HyDE. A ideia de incluir o HyDE é imitar a pesquisa por evidências feita por um jornalista ou alguém com pensamento crítico. Essas pessoas realizam a pesquisa na web para procurar informações sobre as evidências, entretanto procuram por links onde possuem informações que refutam ou sustentam os aspectos que constroem a notícias, esses aspectos sendo legítimos ou não. Dessa forma, o usuário não se baseia apenas nos primeiros resultados retornados pelo navegador, mas aqueles que ajudam na construção de um julgamento sólido para a notícia.

Para mitigar as limitações de desatualização de conhecimento, no começo da interação, um texto da notícia C é utilizado para capturar evidências $E_v = \{E_1, E_2, E_3, ...\}$ na internet (utilizamos Google Custom Search API), que posteriormente serão avaliadas e selecionadas pelo modelo LLM, reduzindo a dependência de seleção por relevância da API. Com base nos resultados de benchmark da seção 5.4, escolhemos o modelo Qwen2.5-7B-Instruct (YANG et al., 2024), que permite aumento no escopo dos textos por disponibilizar textos maiores sem cortes para inferência; e mais leve comparado com o LLM do STEEL, reduzindo o custo do hardware e possibilitando uma escalabilidade acessível. O agente retornará uma saída contendo y sendo $y \in \{\text{true}, \text{false}\}$ e explicação E_x da análise, mitigando o problema 'black box' apresentado nas propostas com modelos tradicionais. O código está disponível no GitHub [1].

A Figura 7 ilustra os dois módulos que constitui o agente, a interação entre os dois é compactada pelo mecanismo de pesquisa interativa.

[1] <https://github.com/pbbllo/Trabalho_de_Conclusao_de_Curso>

4.1 Retrieval Module

4.1.1 Web Search

A notícia C é processada na consulta web para obter os links e resultados contendo evidências. Pela documentação oficial da API Custom Search JSON (Google Developers, 2025), o comprimento máximo de uma solicitação de pesquisa é de 2.048 caracteres, e por conta disso, no momento das consultas, estamos restringindo o número de caracteres para 100, para evitar erros de API por conta da limitação de caracteres, uso excessivo de cotas e consultas muito longas que podem afetar o desempenho e relevância dos resultados retornados. Assim como o STEEL (LI et al., 2024), estamos capturando no máximo 10 links, porém nem todos são utilizados por conta da limitação do tamanho de contexto imposto pelas LLMs. No mecanismo de busca, para conferir credibilidade às consultas, baseamos na lista de 1.368 sites de notícias em inglês (PAPADOGIANNAKIS et al., 2023) e uma lista de 73 sites de notícias brasileiras, construída pela captura dos domínios das notícias reais do Fake.Br Corpus e os principais sites (nosso julgamento) utilizados pelos jornalistas brasileiros. Dessa forma, termos certeza que estamos recuperando evidências de fontes confiáveis. Os resultados são organizados em ordem decrescente de relevância do algoritmo de pesquisa, o resultado mais relevante sempre na primeira posição.

A função 1 recebe duas entradas: claim (a alegação que precisa ser verificada),

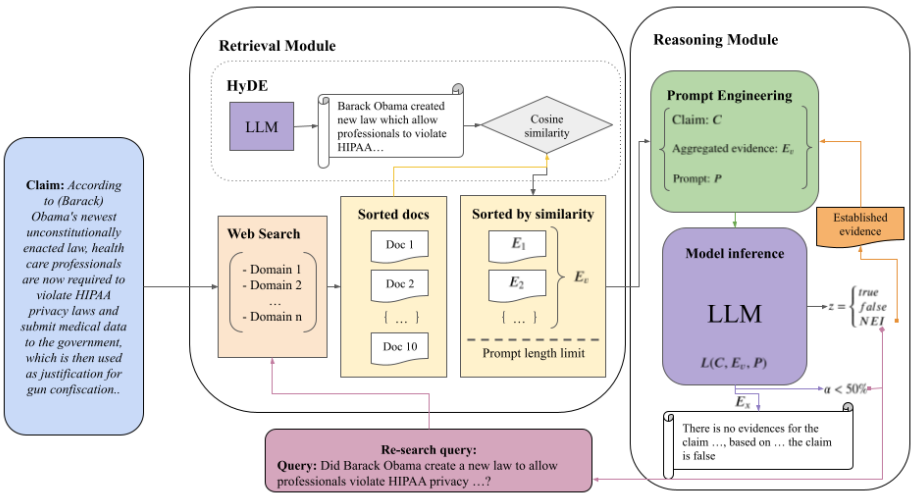

Figura 7 – Representação do funcionamento da proposta do Agente para detecção de notícias falsas. Fonte: Autoria própria

,

Algorithm 1 Web Search

Require: claim, query
Ensure: evidences
1: web_results ← web_search(query)
2: **if** web_results is not empty **then**
3: list_similarities ← HyDE(claim, web_results)
4: documents_desc ← Sort web_results by list_similarities in descending order
5: documents_top_3 ← documents_desc[1..3]
6: claim_len ← len(claim)
7: max_len ← 4096 - claim_len
8: evidences ← process_documents(documents_top_3, max_len)
9: **return** evidences
10: **end if**
11: **return** []

query (a consulta de busca usada para encontrar informações relevantes na web). Na linha 1 inicia uma busca na web com base na query e na linha 2 verificamos se houve resultados; se não houver resultados, a função retorna uma lista vazia de evidências, caso o contrário, o processo de recuperação continua. Na linha 3, chamamos a função 2 (nossa proposta de mudança comparado com o STEEL). Na linha 4 a 5 os documentos são ordenados com base na similaridade e os três mais relevantes são selecionados, quantidade esta escolhida com base nos resultados apresentados por (LI et al., 2024). Na linha 6 a 7, o número de tokens da alegação é calculado usando o tokenizer do LLM, e o tamanho máximo do contexto é determinado, considerando o limite do modelo e o espaço ocupado pela alegação. Na linha 8 a 9, os documentos recuperados são ajustados para caber dentro do limite máximo de contexto do modelo por meio de uma função chamada *process_documents*, retornando as evidências processadas, ou uma lista vazia se nenhum resultado for encontrado na web.

4.1.2 HyDE

Incluirmos a técnica HyDE, com o LLM transformando a notícia C em uma paráfrase P e ordenando os resultados do mais similar ao menos similar a P. Essa seleção de evidência busca que o processo generativo capture as informações que ajudem a julgar a característica chaves que compõe a notícia (GAO et al., 2022). Após o HyDE, filtramos as top três mais similares e percorremos essa lista seguindo o mesmo processo feito pelo STEEL (LI et al., 2024). Calculamos o tamanho da notícia C e subtraímos do tamanho pré-definido pelo tamanho de contexto do LLM, resultando no tamanho

máximo de informações $T = 4096 - len(C)$. Desse modo, incluirmos a evidência por completo caso esteja dentro do limite de tamanho de T e seguindo para as próximas evidências, até chegar no caso em que incluímos fragmentos da evidência para completar o valor de T. Com esse processo, recuperamos informações relevantes mantendo a integridade.

Algorithm 2 HyDE

Require: claim, web_results
Ensure: list_similarities
1: paraphrase ← generate_paraphrase(claim)
2: encode ← embedding_model(”paraphrase-multilingual-MiniLM-L12-v2”)
3: web_results_embedding ← encode(web_results)
4: paraphrase_embedding ← encode(paraphrase)
5: paraphrase_nm ← normalize(paraphrase_embedding)
6: list_similarities ← []
7: **for** document in web_results_embedding **do**
8: document_nm ← normalize(document)
9: con_sim ← similarity(document_nm, paraphrase_nm)
10: Append con_sim to list_similarities
11: **end for**
12: **return** list_similarities

Na linha 1 parafraseamos a notícia com o próprio LLM usando o HyDE prompt A.1. Na linha 2, um modelo de embeddings chamado *’paraphrase-multilingual-MiniLM-L12-v2’* da biblioteca SentenceTransformer é carregado para converter os textos em vetores numéricos (não foi realizado nenhum benchmark, apenas a configuração já estudada). Na linha 3 a 4, os resultados da busca na web, bem como a alegação parafraseada, são transformados em embeddings pelo modelo. Na linha 5 o embedding da alegação parafraseada é normalizado usando a norma L2 para garantir que todos os vetores tenham magnitude unitária, facilitando a comparação de similaridade. Na linha 6 a 10, para cada documento recuperado na web, seu embedding é normalizado (norma L2) e comparado com o embedding da alegação parafraseada usando cosseno de similaridade, métrica que mede a proximidade semântica entre o documento e a paráfrase da alegação. Na linha 12, retornamos a lista de similaridades de cada resultado web.

4.2 Reasoning Module

Com a lista de evidências, agregamos em um prompt para a inferência no LLM. O prompt instruirá o LLM a usar as evidências para avaliar a notícia, desse modo o agente pode fazer decisões com base em evidências ou decidir se precisa fazer mais pesquisas para avaliar a notícia. A saída é restringida a três categorias: true, false e NEI (Not Enough Information), com uma explicação baseada nas evidências passadas e uma taxa de confiança $Conf$ fornecida pelo LLMs. Quando temos a saída NEI ou $\alpha < 50$, "Informação comprimida" e "Nova consulta" também vão estar na saída para a coleta de novas evidências, sendo "Informação comprimida" para não perder o raciocínio da última interação e "Nova consulta" para procurar por informações que o LLMs julgou estarem faltando. Para lidar com problemas de alucinações e respostas inconsistentes, (XIONG et al., 2023) propõem métodos de autoconsciência e autojulgamento para que os LLMs forneçam pontuações de confiança entre 0 e 100%. Porém, ajustamos esse intervalo de confiança introduzindo um coeficiente de excesso de confiança (LI et al., 2024), para combater o problema dos LLMs modernos com excesso de confiança (XIONG et al., 2023; WANG et al., 2023). Desse modo, a confiança final do agente é dada pela equação 4.1.

$$\alpha = \beta \times Conf \tag{4.1}$$

A função do 3 recebe três parâmetros: claim (a alegação que está sendo analisada), evidences (um conjunto de evidências recuperadas que podem ajudar na verificação da alegação) e Reasoning Prompt A.1 (formato entrada para estruturar o raciocínio do LLM). Se existirem evidências, elas são concatenadas em uma única string, separadas por quebras de linha, caso contrário, a variável evidences fica vazia. Em seguida, um dicionário chamado message é criado para representar a entrada do modelo de IA, onde o prompt é formatado substituindo search_text pelo claim e context_str pelas evidences, permitindo que o modelo processe a alegação no contexto das evidências recuperadas. Na linha 2 realizamos a inferência, que retorna um output contendo a resposta gerada. Na linha 3 a 7, expressões regulares são usadas para extrair partes estruturadas da resposta do modelo, como explanation (a explicação do modelo sobre a alegação), answer (a conclusão sobre se a alegação é verdadeira ou falsa), query

(as consultas de pesquisa sugeridas pelo modelo), compressed_info (um resumo da evidência usada na decisão) e confidence (um valor numérico que representa a confiança do modelo na resposta). Na linha 8 a 9, ajustamos o valor da confiança pela equação 4.1, criamos um dicionário chamado output_map para armazenar os resultados extraídos do regex. Na linha 10 retornamos o output (a resposta completa gerada pelo modelo) e output_map (o dicionário com as informações processadas).

Algorithm 3 Reasoning

Require: claim, evidences, prompt
Ensure: output, output_map
1: input ← format_prompt(claim, evidences, prompt)
2: output ← generate_answer(input)
3: explanation ← get_explanation(output)
4: answer ← get_answer(output)
5: query ← get_query(output)
6: compressed_info ← get_compressed_info(output)
7: confidence ← get_confidence(output)
8: real_confidence ← calculate_confidence(confidence)
9: output_map ← {explanation, answer, query, compressed_info, real_confidence}
10: **return** output, output_map

4.3 Re-Search Mechanism

O mecanismo consiste em fazer novas buscas para garantir evidências mais robustas e precisas que deem base ou refutem a notícia em análise. Quando as condições para uma nova pesquisa são atendidas (saída NEI ou $\alpha < 50$), comprimimos as informações raciocinadas pelo agente em “evidência estabelecida”. Após isso, o LLM sugere uma nova consulta web para obter mais informações, que ajudem a chegar na resposta diferente de NEI e alcançar uma confiança maior que 50. Pela lista de prompts na seção A.1, existe uma diferença no prompt do STEEL que instrui OPENAI’s GPT-3.5-turbo à criação de várias consultas (LI et al., 2024), em comparação com a nossa proposta: o prompt instrui a criação de apenas uma consulta. O motivo dessa mudança é que o LLM que estamos utilizando pode gerar saídas mais inconsistentes por ser um modelo menor.

Dessa forma, quando o LLM determina que o conjunto de evidências atual é insuficiente para ter uma avaliação confiável sobre a notícia, a saída será NEI. A saída fará com que o agente avance à próxima iteração, até alcançarmos nossas condições

de parada. Escolhemos duas condições para encerrar o processo de pesquisa exaustiva. A primeira condição é quando $\alpha > 50$, o que significa que temos uma taxa de confiança aceitável para as evidências, julgamento e resposta do LLM. A segunda condição é o encerramento do mecanismo de pesquisa depois da terceira interação (escolha baseada nos resultados apresentados na Figura 8). Abaixo temos o pseudo-código 4 demonstrando o mecanismo implementado.

Algorithm 4 Re-search mecanismo

Require: claim, language
Ensure: answer, confidence, explanation, output

```
1: step ← 1
2: query ← claim                ▷ Na primeira iteração, a consulta é a própria alegação
3: compressed_info ← Empty string
4: while step ≤ 3 do
5:     evidences ← Retriever(claim, query)
6:     prompt ← REASONING_PROMPT[language]
7:     if compressed_info is not empty then
8:         evidences ← evidences + compressed_info
9:     end if
10:    output, output_map ← Reasoning(claim, evidences, prompt)
11:    confidence ← output_map["confidence"]
12:    query ← output_map["query"]
13:    compressed_info ← output_map["compressed_info"]
14:    step ← step + 1
15:    if output_map["answer"] == '2' then
16:        continue
17:    end if
18:    if confidence > 50 then
19:        break
20:    end if
21: end while
22: return output_map["answer"], confidence, output_map["explanation"], output
```

A função 4 é definido para receber *claim*, que representa a alegação a ser analisada e *language* referente ao idioma da notícia. Diferente de (LI et al., 2024), isso propociona flexibilidade para receber notícia de qualquer idioma (na nossa proposta focamos no inglês e brasileiro). Na linha 1 a 3, as variáveis são inicializadas: *step* começa em 1 para controlar o número de iterações, *query* recebe a própria *claim* inicialmente, *compressed_info* começa como uma string vazia para armazenar informações resumidas das iterações anteriores. Na linha 4, o loop principal se repete até que o *step* atinja o limite de 3. Em cada iteração, na linha 5 o retrieval module busca na web

as evidências para analisar a alegação. Na linha 6 a 8, prompt é carregado para guiar o reasoning module na verificação da alegação, e se *compressed_info* não estiver vazia, ela é adicionada às evidências recuperadas. Na linha 10, o reasoning module recebe a alegação, as evidências coletadas e o prompt para gerar uma resposta estruturada, dividida em output (resposta completa do modelo) e output_map (dicionário contendo explicação, confiança, queries e informações resumidas). Na linha 11 a 13, são extraídos os dados do *output_map*, como confiança, query e informações resumidas e o contador de iterações é incrementado. Na linha 14 a 16, se a resposta for inconclusiva e ainda houver iterações disponíveis, o loop continua para tentar melhorar a resposta. Na linha 17 a 19, se a confiança for superior a 50, o loop é interrompido, pois a resposta já tem um nível de confiança considerado aceitável. Finalmente, na linha 22 a função retorna a resposta final do modelo, o nível de confiança associado à resposta, a explicação detalhada do raciocínio do modelo e a resposta completa.

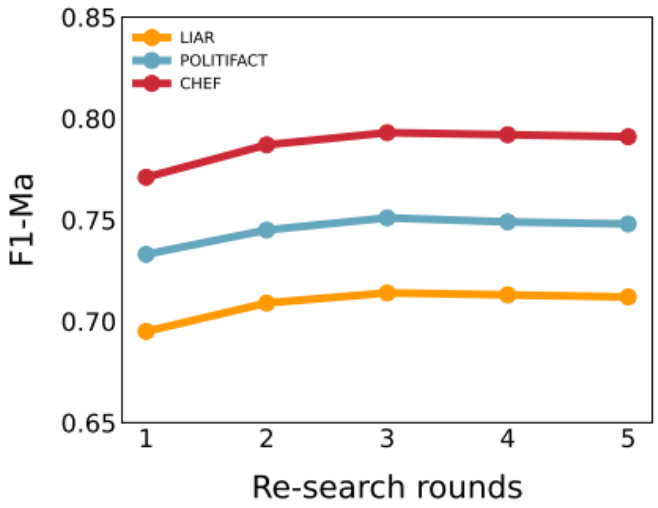

Figura 8 – Evolução do desempenho comparado com número de rodadas de pesquisa. Valor do F1-Ma por rodadas de pesquisas. Há uma crescente no F1-Ma da primeira interação até a terceira. Após a terceira interação, praticamente não possui aumento no F1-Ma. Logo, entende-se que, depois da terceira rodada, não há um aumento significativo na taxa de desempenho em classificar as notícias. Utilizamos esse resultado do STEEL para definir um número máximo de paradas, pois o Google Search Web API tinha uma limitação de uso gratuito, dificuldando rodadas próprias de teste. Fonte: (LI et al., 2024)

5 Experimentos

Nessa seção, descreveremos o processo de configuração e construção dos experimentos e resultados. Na seção 5.1 comentamos o configuração dos pacotes e hardware utilizado para os experimentos. Na seção 5.2 descrevemos a preparação dos datasets para as rodadas de experimentos. Na seção 5.3 são pontuados os detalhes parametrização, metodologia e descrição das métricas para avaliação dos desempenhos. Na seção 5.4 avaliamos os desempenho de três modelos LLMs candidatos para nossa proposta. Finalmente, na seção 5.5 comparamos o desempenho da nossa proposta aos modelos candidatos e aos trabalhos relacionados.

5.1 Especificações do sistema e versões

O Google Colab foi utilizado durante todo o processo de experimentos. Utilizamos a GPU T4 de 16GB para fazer o carregamento dos modelos. As bibliotecas do Python e suas versões são mostradas na tabela abaixo.

Biblioteca	**Versão**
Python	3.11.11
unsloth	2025.1.1
pyarrow	17.0.0
transformers	4.47.1
torch	2.5.1
sklearn	1.6.0

Tabela 3 – Versão do Python e pacotes utilizados

5.2 Datasets

Para os experimentos, foram utilizados dois datasets com notícias do mundo real que abrangem diferentes tópicos (política, religião, economia, etc.), LIAR dataset (WANG, 2017) para notícias em inglês e Fake.Br Corpus dataset (SILVA et al., 2020) para notícias em português brasileiro. Desconsideramos notícias com rótulos que poderiam causar confusão e ambiguidade: ”pants-fire”, ”barelytrue”, ”half-true”, ”mostly-true”,

considerando apenas rótulos “true” e “false”. Após isso, alteramos a coluna “label” com valores “0” para notícias falsas e “1” para notícias verdadeiras; e coluna “text” com o texto completo das notícias, sem nenhum pré-processamento feito pelos autores, apenas o texto original. Para o LIAR, apenas consideramos as instâncias de treino para as rodadas, pois o tratamento feito pelo (LI et al., 2024) não foi descrito com detalhes e o repositório não estava disponível para reproduzir o tratamentto. Desse modo, mantivemos os datasets com quantidade equilibrada para avaliação, perto de 50% para notícias falsas e verdadeiras. Na Tabela 4 temos um resumo das análises dos datasets depois desse ajuste.

	LIAR	**Fake.Br Corpus**
#Notícias Verdadeiras	1676 (45%)	3600 (50%)
#Notícias falsas	1995 (55%)	3600 (50%)
#Total	3671	7200

Tabela 4 – Estatísticas dos datasets

5.3 Detalhes da Implementação

Fizemos apenas uma rodada percorrendo todas as instâncias dos datasets. Nas inferências, definimos a temperatura para 0, top-p para 0.75 e limite de tokens no prompt para 4096 (o modelo vai aceitar o input de texto para no máximo 4096 tokens); e o valor do parâmetro β para 0.7 (LI et al., 2024). Tratamos a detecção de notícias falsas como uma classificação binária, avaliando as saídas 0 ou 1 para determinar o desempenho modelo. Em caso de saídas 2 (não foi possível encontrar evidências suficientes para ter confiança na resposta) convertemos para 0, pois interpretamos que não existem evidências para sustentar a veracidade da notícia.

Utilizamos as métricas apresentadas Tabela 5 para avaliar o desempenho nos experimentos: A para acurácia, P para precisão, R para Recall e F1 para F1-score. Importante salientar que estamos utilizando métricas com sufixo -F, F1-Mi, F1-Ma (nessa ordem) como métricas de peso, pois estamos procurando verificar o quão bem estamos detectando as notícias falsas e verificar o desempenho geral. Usamos a Tabela 6 como referência para análises com matriz de confusão.

Métrica	Descrição
A (Acurácia)	Porcentagem da correspondencia das notícias classificadas com as respostas esperadas.
F1-Ma	Média aritmética do F1-score de cada classe. Evita que classes majoritárias dominem a métrica.
F1-Mi	Precisão e Recall considerando todas as instâncias do dataset. Leva em conta o número total de exemplos corretamente classificados.
F1-T	Desempenho geral na previsão de notícias reais.
P-T	Desempenho de consistência em relação as notícias reais. De todas as notícias classificadas como reais, quantas realmente são reais?
R-T	Desempenho de cobertura em relação as notícias reais. De todas as reais, quantas o modelo classificou corretamente reais?
F1-F	Desempenho geral na previsão de notícias falsas.
P-F	Desempenho de consistência em relação as notícias falsas. De todas as notícias classificadas como falsas, quantas realmente são falsas?
R-F	Desempenho de cobertura em relação as notícias falsas. De todas as notícias falsas, quantas o modelo classificou corretamente como falsas?

Tabela 5 – Descrição das métricas nas avaliações

Classes	Abordagem
TP	Notícias falsas classificadas corretamente como notícias falsas
FN	Notícias falsas classificadas erroneamente como notícias verdadeiras
TN	Notícias verdadeiras classificadas corretamente como notícias verdadeiras
FP	Notícias verdadeiras classificadas erroneamente como notícias falsas

Tabela 6 – Descrição das classes em avaliações usando matriz de confusão. As matrizes de confusão foram criadas usando notícias falsas como classe de interesse.

5.4 Benchmark

Nessa seção testamos o conhecimento e poder de interpretação de contexto dos modelos candidatos à proposta. Fizemos rodadas usando os prompts de benchmark A.1 em três modelos de instrução: Ministral-8B-Instruct-2410 (AI, 2025) derivado do Mistral-7B (JIANG et al., 2023), Qwen2.5-7B-Instruct (YANG et al., 2024) e Llama-3.1-8B-Instruct (DUBEY et al., 2024). O motivo da escolha de modelos de instrução, remete a especialidade em obedecer orientações, tornando mais eficiente aos comandos nos prompts. Na Tabela 7 referente aos resultados no LIAR, Qwen se saiu melhor nas métricas de peso, pois de modo geral classificou melhor as notícias falsas (F1-F) e teve uma cobertura maior (R-F). Apesar do Llama ter tido desempenho melhor no F1-Ma, entendemos que essa métrica ficou discrepante por conta dos valores superiores nas métricas '-T', além do Qwen ter ficado pouco atrás no F1-Mi e P-F. Llama e Qwen tiveram resultados competitivos.

	LIAR								
	A	**F1-Ma**	**F1-Mi**	**F1-T**	**P-T**	**R-T**	**F1-F**	**P-F**	**R-F**
Llama-3.1-8B-Instruct	**0.6203**	**0.6089**	**0.6203**	**0.5420**	0.6031	**0.4922**	0.6757	**0.6305**	0.7278
Qwen2.5-7B-Instruct	0.5846	0.4823	0.5846	0.2521	**0.7080**	0.1533	**0.7124**	0.5710	**0.9469**
Ministral-8B-Instruct-2410	0.5898	0.4998	0.5898	0.2876	0.6941	0.1814	0.7119	0.5756	0.9328

Tabela 7 – Llama teve uma acurácia e P-F superior a 4% e 6% respectivamente, porém foi superado no R-F e F1-F pelos outros dois, Qwen performando melhor no R-F e F1-F em 22% e 4% comparado com Llama.

Pela na Figura 9, no 1ª quadrante (TP) e 2ª quadrante (FN), Qwen performou melhor em classificar corretamente notícias falsas como notícias falsas e confundiu menos notícias falsas como True News, comparados aos outros dois modelos.

Na Tabela 8 referente aos resultados no Fake.Br Corpus, podemos perceber desempenho superior do Qwen comparado aos demais. Apesar do Llama ter tido maior consistência em classificar notícias falsas (P-F), nas demais métricas de peso, Qwen teve desempenho superior. Na Figura 10, observamos como a Llama confundiu bem mais as notícias falsas com True News comparado com o Qwen com base no 2º quadrante.

	Fake.Br Corpus								
	A	**F1-Ma**	**F1-Mi**	**F1-T**	**P-T**	**R-T**	**F1-F**	**P-F**	**R-F**
Llama-3.1-8B-Instruct	0.7522	0.7429	0.7522	0.7918	0.6828	**0.9422**	0.6941	**0.9068**	0.5622
Qwen2.5-7B-Instruct	**0.8118**	**0.8112**	**0.8118**	**0.8216**	**0.7810**	0.8667	**0.8009**	0.8502	**0.7569**
Ministral-8B-Instruct-2410	0.7853	0.7813	0.7853	0.8109	0.7244	0.9208	0.7516	0.8914	0.6497

Tabela 8 – Com exceção da P-F, onde Llama teve desempenho superior em 6% comparado com o Qwen, Qwen ultrapassou os Llama na acurácia, R-F e F1-F, em 6%, 19% e 11%.

Acreditamos que LLaMA e Qwen se destacaram por conta da curadoria dos textos retirados da web para dataset de treinamento. O Qwen apresentou um comportamento de raciocínio crítico e o Llama demonstrou depender mais de fontes, comparando os textos em negritos nos exemplos seção A.2. Dessa forma, o Qwen pode ter mais sucesso em identificar as nuanças nas notícias falsas e o Llama possívelmente teve mais acesso a fontes na sua base de treinamento, permitindo identificar as notícias verdadeiras com mais certeza.

5.5 Resultados da principais

Dado os resultados de benchmark, escolhemos o Qwen2.5-7B-Instruct como modelo LLM a ser utilizado na proposta. Na subseção 5.5.1 queremos analisar o impacto da nossa implementação, se permitiu uma evolução ou não em desempenho. Na subseção 5.5.2 procuramos comparar nosso desempenho em relação aos trabalhos relacionados que usam a captura de evidências externas para avaliação de notícias. Finalmennte, na subseção 5.5.3 comparamos nossa proposta com estratégias que tiveram LLMs na sua implementação.

5.5.1 Proposta VS Desempenho dos Modelos Benchmarks

A Tabela 9 mostra o desempenho em relação às notícias inglesas, tivemos uma melhora na acurácia, F1-Mi, P-T, F1-F sob os melhores resultados de benchmark. Em relação ao Qwen, houve uma melhora em todas as métricas, com exceção do R-F (atrás por 5%). Pela Figura 9, podemos verificar que com o mecanismo de pesquisa, os Falsos Positivos diminuíram e True Negatives aumentaram, o que mostra que nossa estratégia melhorou o desempenho em classificar as notícias verdadeiras e falsas. Entretanto, com decréscimo de 5% nos True Positives e False Negatives (refletido no valor R-F), podemos dizer perdemos um pouco a cobertura na classificação de notícias falsas.

Nas notícias brasileiras, a Tabela 10 mostra que não houve melhora no desempenho, o Qwen permaneceu com os melhores índices. Pela Figura 10, tivemos uma piora em todos os quadrantes, em especial para notícias verdadeiras, com o aumento de 18% nos FP e decréscimo 18% no TN, pensando nas métricas referente a notícias verdadeiras.

	LIAR								
	A	**F1-Ma**	**F1-Mi**	**F1-T**	**P-T**	**R-T**	**F1-F**	**P-F**	**R-F**
Llama-3.1-8B-Instruct	0.6203	**0.6089**	0.6203	**0.5420**	0.6031	**0.4922**	0.6757	**0.6305**	0.7278
Qwen2.5-7B-Instruct	0.5846	0.4823	0.5846	0.2521	0.7080	0.1533	0.7124	0.5710	**0.9469**
Ministral-8B-Instruct-2410	0.5898	0.4998	0.5898	0.2876	0.6941	0.1814	0.7119	0.5756	0.9328
STEEL_HyDE	**0.6295**	0.5819	**0.6295**	0.4408	**0.7090**	0.3198	**0.7230**	0.6089	0.8897

Tabela 9 – Proposta VS Benchmark results no dataset inglês

Com base na Figura 11, um dos possíveis motivos para essa piora é a quantidade pequena de sites para recuperar informações, interferindo no raciocínio com base

	Fake.Br Corpus								
	A	**F1-Ma**	**F1-Mi**	**F1-T**	**P-T**	**R-T**	**F1-F**	**P-F**	**R-F**
Llama-3.1-8B-Instruct	0.7522	0.7429	0.7522	0.7918	0.6828	**0.9422**	0.6941	**0.9068**	0.5622
Qwen2.5-7B-Instruct	**0.8118**	**0.8112**	**0.8118**	**0.8216**	**0.7810**	0.8667	**0.8009**	0.8502	**0.7569**
Ministral-8B-Instruct-2410	0.7853	0.7813	0.7853	0.8109	0.7244	0.9208	0.7516	0.8914	0.6497
STEEL_HyDE	0.7000	0.7000	0.7000	0.6984	0.7021	0.6947	0.7016	0.6979	0.7053

Tabela 10 – Proposta VS Benchmark results no dataset brasileiro

em evidências, fazendo que agente não chegue a um julgamento da notícia. Isso salienta a importância de mais fontes de confiança para análises de notícias e detecção de notícias falsas.

5.5.2 Proposta VS Estratégias Retriever

Comparamos o desempenho com estratégias que utilizam machine learning e deep learning com evidências no seu raciocínio. Para o LIAR dataset, usamos as mesma comparações feitas por (LI et al., 2024). Analisando as métricas na Tabela 11, conseguimos ter desempenhos superiores no P-T, F1-F e R-F de 7%, 8% e 26%, respectivamente. Isso demonstra que o LLM escolhendo qual evidência é relevante, pode melhorar na detecção de notícias falsas.

	LIAR							
	F1-Ma	**F1-Mi**	**F1-T**	**P-T**	**R-T**	**F1-F**	**P-F**	**R-F**
EHIAN	0.5910	0.5930	0.5590	0.5430	0.5480	0.6300	0.6030	0.6170
MAC	0.6030	0.6010	0.5620	0.5580	0.5670	0.6250	0.6230	0.6210
GET	0.6140	0.6100	0.5720	0.5670	0.5790	0.6410	0.6540	0.6320
ReRead	0.6110	0.6150	0.5870	0.5810	0.5960	0.6330	0.6280	0.6260
MUSER	**0.6450**	**0.6420**	**0.6470**	0.6400	**0.6540**	0.6430	**0.6500**	0.6360
STEEL_HyDE	0.5819	0.6295	0.4408	**0.7090**	0.3198	**0.7230**	0.6089	**0.8897**

Tabela 11 – STEEL_HyDE vs Algoritmos de ML e DP no dataset inglês

Para notícias brasileiras, não tivemos sucesso em encontrar artigos com a mesma proposta de recuperação de evidências. Desse modo, fizemos as comparações baseado no trabalho de (LAKZAEI; CHEHREGHANI; BAGHERI, 2025), que utilizou aprendizagem semi-supervisionada em GNNs com 10% do dataset rotulado e o resto sem rótulo, diminuindo um pouco a diferença que o treinamento supervisionado pode fazer no desempenho, pois o proprio modelo deve aprender os padrões com poucas dicas do dataset, algo próximo do que estamos fazendo. Com base na Tabela 12, nossa proposta perdeu apenas para (LAKZAEI; CHEHREGHANI; BAGHERI, 2025),

ficando acima do segundo melhor resultado por 8% na acurácia e F1-Ma. Apesar de esperamos um desempenho ruim na comparação porquê não fizemos treinamentos, conseguimos competir na classificação de notícias de forma geral, isso demonstra o potencial dos LLMs para tarefas PLN. O artigo com os resultados dos trabalhos com GNNs, utilizou apenas a métrica de acurária e F1-Ma para comparação, por isso desconsideramos as demais métricas usadas nas outras comparações desse trabalho.

Fake.Br Corpus		
	A	**F1-Ma**
GATv2	0.6171	0.6130
LSTM-CP-GCN	0.5153	0.5110
TGNCL	0.5673	0.5581
TextGCN	0.6235	0.6112
L2Q	0.5701	0.5636
Co-GNN	0.6166	0.6166
NOL-GAT	**0.8126**	**0.8139**
STEEL_HyDE	0.7000	0.7000

Tabela 12 – STEEL_HyDE vs Algoritmos GNN

5.5.3 Proposta VS LLMs (LIAR Dataset)

Utilizando a mesma tabela de comparação de performances LLMs no LIAR (LI et al., 2024). Analisando os resultados na Tabela 13, STEEL prevaleceu demonstrando consistência em identificar tanto notícias falsas e verdadeiras comparação com os outros métodos. Nossa proposta teve um desempenho superior na precisão de notícias verdadeiras sobre os demais. Apesar de termos desempenhos superiores no R-F e P-T de 14% e 3%, a abordagem de (LI et al., 2024) prevalece sendo mais consistente, com valores das demais métricas acima das nossas.

Um motivo que pesou desempenho abaixo em algumas métricas, ênfase nas métricas com sufixo '-T', foi o modelo base utilizado. O Qwen2.5 demonstrou dificuldades em identificar notícias verdadeiras, isso pode ter interferido no processo de análise das notícias durante o mecanismo de pesquisa. É possível que (LI et al., 2024) conseguiu acessar mais evidências para as notícias verdadeiras, pois fez pesquisas na web exceto pelos sites duvidosos apontado por (PAPADOGIANNAKIS et al., 2023), nós tivemos uma abordagem mais restrita usando apenas os sites confiáveis na pesquisa, pois

	LIAR							
Methods	**F1-Ma**	**F1-Mi**	**F1-T**	**P-T**	**R-T**	**F1-F**	**P-F**	**R-F**
GPT-3.5-turbo	0.5630	0.5410	0.5590	0.5720	0.5670	0.5550	0.5640	0.5600
Vicuna-7B	0.5280	0.5350	0.5210	0.5430	0.5520	0.5190	0.5380	0.5260
WEBGLM-2B	0.6010	0.5970	0.5580	0.5630	0.5710	0.6220	0.6040	0.6180
ProgramFC	0.6310	0.6130	0.6370	0.6070	0.6390	0.6250	0.6110	0.6280
STEEL	**0.7140**	**0.6890**	**0.6850**	0.6800	**0.6910**	**0.7430**	**0.7250**	0.7520
STEEL_HyDE	0.5819	0.6295	0.4408	**0.7090**	0.3198	0.7230	0.6089	**0.8897**

Tabela 13 – Comparação de performace entre os métodos LLMs. STEEL sobresaiu desempenho melhor em todas as métricas, exceto pela P-T e R-F. Nossa proposta performou melhor no P-T e R-F, acima de 3% e 14% respectivamente. Nossa estratégia demostra ter potêncial para detectar notícias falsas, apesar de se sair mal em encontrar notícias verdadeiras.

conseguiriamos ter um ambiente controlado para notícias inglêsas e brasileiras. Além disso, aparenta ter feito um tratamento diferente no LIAR, poís houve um desequilíbrio na quantidade de notícias falsas e verdadeiras (quantidade significantemente maior para verdadeiras), entretanto por conta do repositório ter expirado, tornou-se difícil verificar se o tratamento influênciou nos resultados. Finalmente, a diferença pequena de prompt entre as duas propostas pode ter mudado de forma sucinta o comportamento em algumas análises (salientamos que a mudança no nosso prompt foi necessária para evitar alucinações, por conta do LLM ser menor).

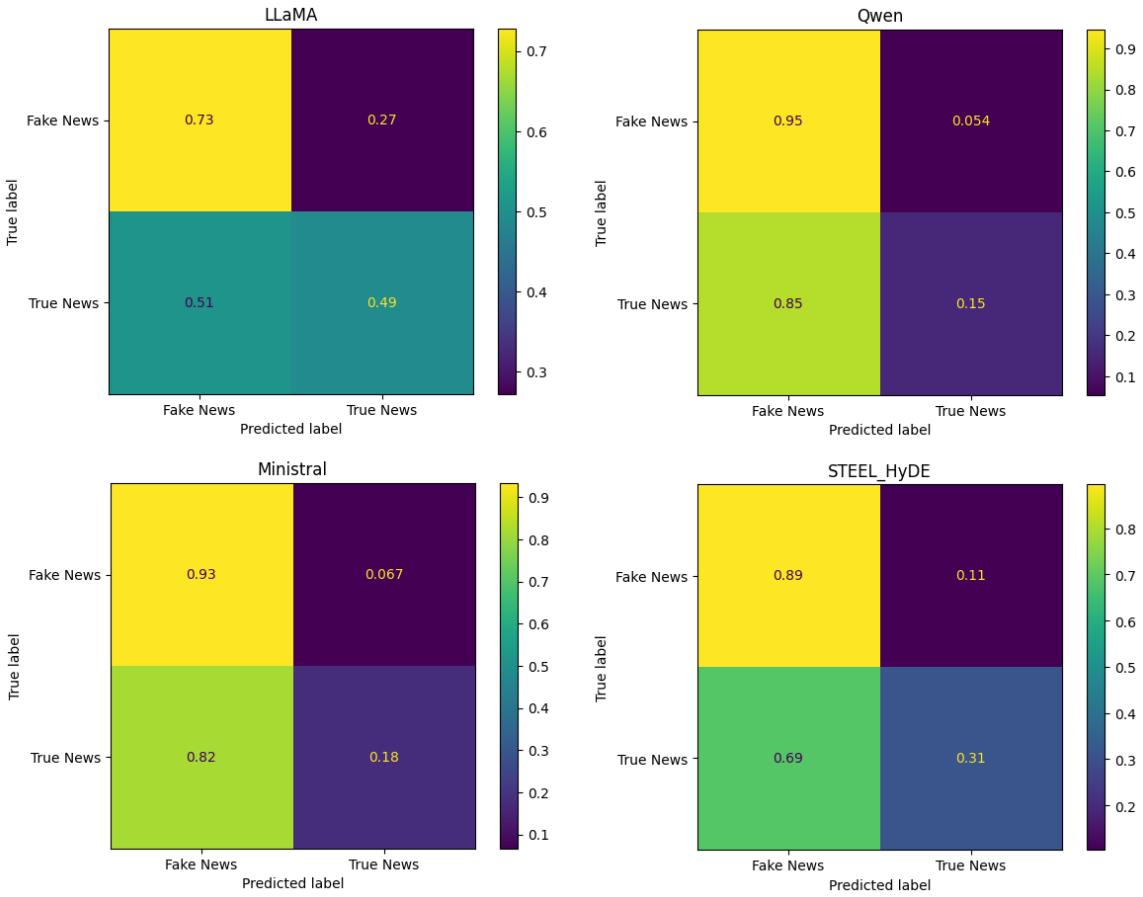

Figura 9 – Matrizes de confusão referente ao resultado no LIAR. Fonte: Autoria propria

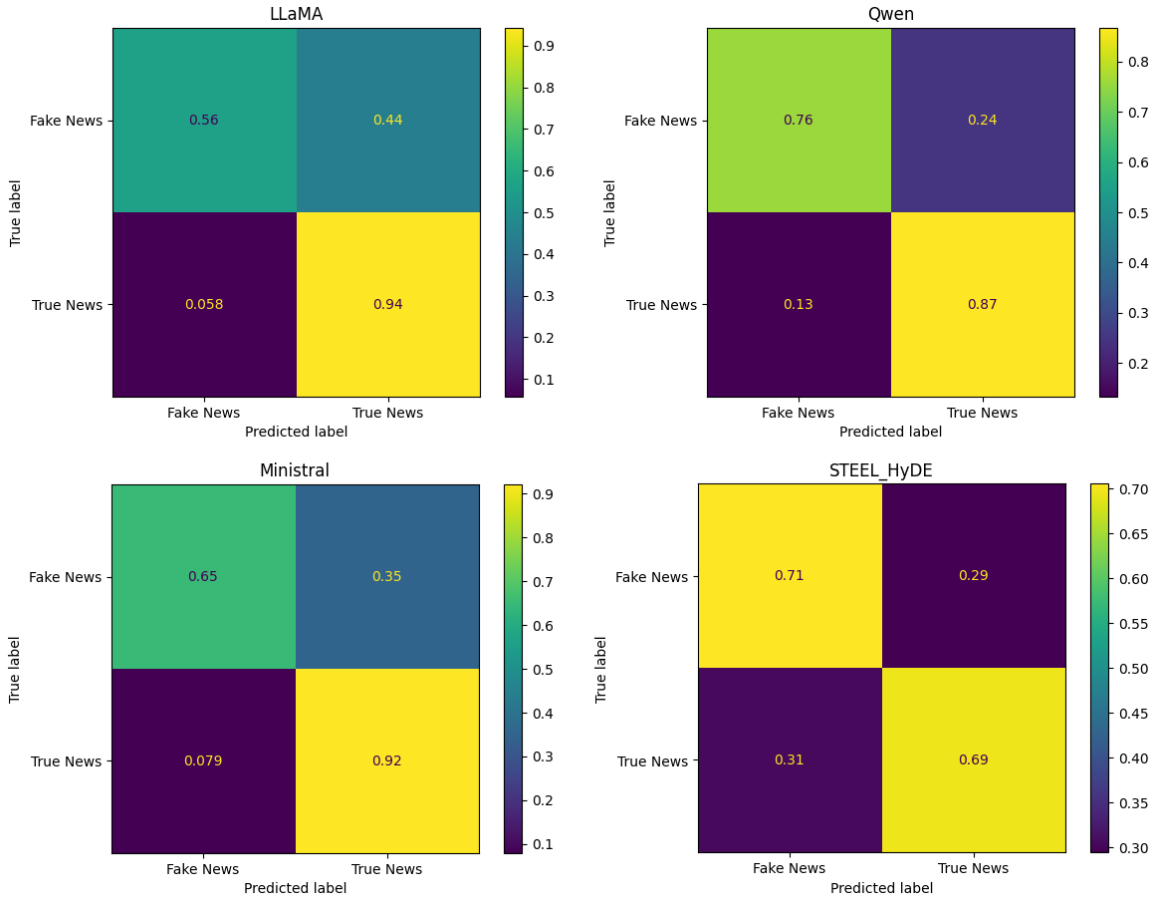

Figura 10 – Matrizes de confusão referente ao resultados no Fake.Br Corpus. Fonte: Autoria propria

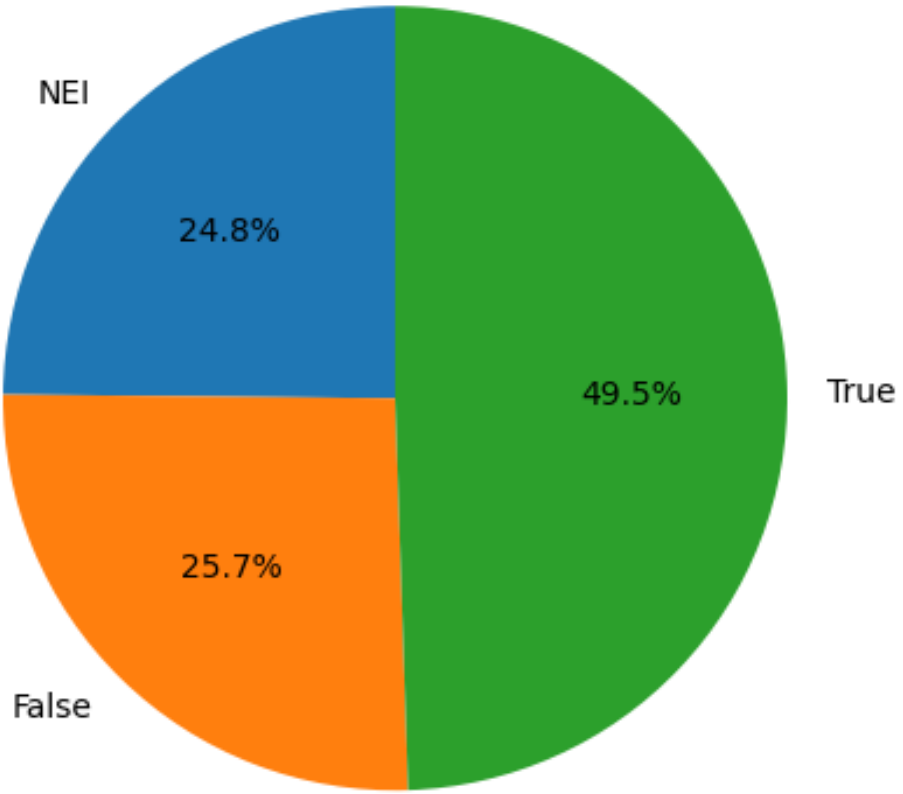

Figura 11 – Gráfico de pizza em relação a porcentagem de cada resposta do agente para notícias brasileiras. Aproximadamente em 25% das previsões o modelo julgou não ter evidências suficientes para alcançar em uma resposta. Fonte: Autoria propria

6 Conclusão

Nesta pesquisa, propusemos uma abordagem baseada em LLMs para a detecção de notícias falsas em inglês e português. Após um processo de benchmarking nos conjuntos de dados LIAR e FakeBr.Corpus com três modelos open-source (Ministral-8B-Instruct-2410, Qwen2.5-7B-Instruct e LLaMA-3.1-8B-Instruct), o modelo Qwen2.5-7B-Instruct foi selecionado devido ao seu desempenho superior em identificar notícias falsas. A proposta demonstrou ganhos notáveis de acurácia e F1-score nas notícias em inglês sobre os modelos de benchmark e desempenhou melhor na métrica de F1-score sobre algoritmos de machine learning e deep learning. Todavia, em relação às notícias brasileiras, tivemos uma piora no desempenho comparado com o benchmark e apresentamos desempenho competitivo sobre os algoritmos GNN. Isso pode ser uma indicação de que, a quantidade pequena de fontes confiáveis pode ter impactado nas rodadas de pesquisas na procura por informações das notícias, interferindo no processo de análise com base nas evidências.

Em relação aos LLMs de modo geral, o STEEL demonstrou ser consistente em classificar ambas as classes, porém nossa proposta conseguiu detectar mais notícias falsas comparando o desempenho de ambas no R-F. Isso indica que utilizar os LLMs para julgar a relevância dos resultados das pesquisas, além do rankeamento de relevância da API, pode ajudar a encontrar evidências valiosas na detecção das notícias falsas. Vale ressaltar que o modelo utilizado na proposta possui um quantidade consideravelmente menor de parâmetros comparado com o STEEL, levantando uma questão se realmente é necessário utilizar LLMs tão robustos para detecção de notícias falsas. Finalmente, essa pesquisa reforça a importância utilizar estratégias de retrieval para tarefas de detecção de notícias falsas, uma vez que os modelos desempenham melhor possuindo informações externas do mundo real.

6.1 Limitações

Assim como pontuado por (PAPAGEORGIOU et al., 2024), LLMs trazem novas dimensões no desafio de detecção de notícias falsas. Dentre suas limitações, estão:

- A habilidade de entender e explicar como chegou na decisão. Por terem um comportamento de ”black boxes”, é extremamente importante salientar sua função coadjuvante em aplicações sensiveis como detecção de notícias falsas. Isso pode trazer consequências negativas em assuntos juridicos e regulatórios em caso da falta de clareza de como o processo de decisão é feito.

- Entendimento de Contexto: Apesar da evolução sobre os algoritmos tradicionais, os LLMs possuem desafios para entender a nuança na linguagem humana, como ironia, sarcasmo e sátira, o que dificulta em diferenciar o que é genuino e enganoso.

- A qualidade dos datasets tem um prazo de tempo e podem conter conteúdo tendênciosos que prejudicam as saídas dos modelos. Certos dados podem conter tópicos, eventos ou pontos de vista que sobrepassam ou afunilam o significado real das notícias.

- Embora os LLMs sejam altamente eficazes na análise e classificação de notícias, eles estão sujeitos a um problema recorrente: as alucinações. Essas ocorrem quando o modelo gera informações imprecisas, irreais ou não fundamentadas em evidências verificáveis. As alucinações podem comprometer a confiabilidade da detecção de fake news, gerando interpretações equivocadas ou justificativas inconsistentes.

6.2 Trabalho Futuro

A proposta demonstra potêncial quanto a detecção de notícias falsas e para alcançar performances melhores, é necessário novas análises e melhorias.

- Novas proposta bases para notícias brasileiras: Para uma comparação mais coerente, seria interessante realizar rodadas com Fake.Br Corpus utilizando o código fonte da estratégias EHIAN (WU et al., 2021), MAC (VO; LEE, 2021), GETLAN (WU et al., 2023), ReRead (HU et al., 2023) e MUSER (LIAO et al., 2023).

- Fine-tuning: O trabalho (BOISSONNEAULT; HENSEN, 2024) mostrou desempenhos expressivos após a realização de *fine-tuning* do ChatGPT e Google Gemini.

A ideia é fazer o mesmo para Qwen2.5-7B-Instruct utilizando dados balanceados em inglês e português para aprimorar sua generalização em contextos multilíngues.

- Atualização de domínios: (PAPADOGIANNAKIS et al., 2023) foi o responsável por separar os sites legitimos e fradulentos para notícias em inglês. Realizar a mesma estratégia para listar sites brasileiros traria mais opções de captura de evidências para notícias brasileira, possivelmente melhorando o desempenho. Há a possibilidade de testar seguindo a mesma estratégia do STEEL, obter resultados da API de pesquisa removendo qualquer domínio compatível com os fraudulentos listados. Entretanto, estariamos voltando ao mesmo problema de não ter controle sobre os demais sites que não foram analisados pela pesquisa, podendo impactar negativamente no julgamento do agente.

- Melhorias no RAG: (WANG et al., 2024) lista outras técnicas de RAG, além do HyDE, que podem trazer novas possibilidades na forma de como recuparemos as evidências.

- Novas rodadas com o STEEL sem adaptações, mantendo o modelo GPT-3.5-Turbo e web search API, combinado com o HyDE para novas avaliações comparativas.

- Arquitetura de multiplos modelos: A implementação com dois ou mais modelos no sistema para fazer o julgamento das notícias.

Referências

AI, M. *Ministral-8B: Model Announcement*. 2025. <https://mistral.ai/en/news/ministraux>. Accessed: 17-02-2025.

ALGHAMDI, J.; LUO, S.; LIN, Y. A comprehensive survey on machine learning approaches for fake news detection. *Multimedia Tools and Applications*, Springer, v. 83, n. 17, p. 51009–51067, 2024.

BOISSONNEAULT, D.; HENSEN, E. Fake news detection with large language models on the liar dataset. 2024.

DEVLIN, J. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

DUBEY, A. et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

GAO, L. et al. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*, 2022.

GOODFELLOW YOSHUA BENGIO, A. C. I. *Machine Learning: Guia Definitivo - Parte 1*. 2024. Acessado em: 25 fev. 2025. Disponível em: <https://www.deeplearningbook.com.br/machine-learning-guia-definitivo-parte-1/>.

Google. *Google Trends: "fake news"*. 2025. Accessed: 2025-02-18. Disponível em: <https://trends.google.com/trends/explore?date=2015-02-18%202025-02-18&q="fake%20news">.

Google Developers. *Using REST to invoke the API*. 2025. Acesso em: 16 fev. 2025. Disponível em: <https://developers.google.com/custom-search/v1/using_rest?hl=pt-br>.

HU, L. et al. Deep learning for fake news detection: A comprehensive survey. *AI open*, Elsevier, v. 3, p. 133–155, 2022.

HU, X. et al. Read it twice: Towards faithfully interpretable fact verification by revisiting evidence. In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. [S.l.: s.n.], 2023. p. 2319–2323.

HUANG, Y.; HUANG, J. A survey on retrieval-augmented text generation for large language models. *arXiv preprint arXiv:2404.10981*, 2024.

JIANG, A. Q. et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

JIN, R. et al. Fake news detection and manipulation reasoning via large vision-language models. *arXiv preprint arXiv:2407.02042*, 2024.

LAKZAEI, B.; CHEHREGHANI, M. H.; BAGHERI, A. Neighborhood-order learning graph attention network for fake news detection. *arXiv preprint arXiv:2502.06927*, 2025.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, v. 33, p. 9459–9474, 2020.

LI, G. et al. Re-search for the truth: Multi-round retrieval-augmented large language models are strong fake news detectors. *arXiv preprint arXiv:2403.09747*, 2024.

LIAO, H. et al. Muser: A multi-step evidence retrieval enhancement framework for fake news detection. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2023. p. 4461–4472.

LIU, X. et al. Webglm: Towards an efficient web-enhanced question answering system with human preferences. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2023. p. 4549–4560.

LIU, Y. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, v. 364, 2019.

MARINOVA, Z. et al. Weverify: Wider and enhanced verification for you project overview and tools. In: IEEE. *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. [S.l.], 2020. p. 1–4.

MOLNAR, C. *Interpretable machine learning*. [S.l.]: Lulu. com, 2020.

NASIR, J. A.; KHAN, O. S.; VARLAMIS, I. Fake news detection: A hybrid cnn-rnn based deep learning approach. *International journal of information management data insights*, Elsevier, v. 1, n. 1, p. 100007, 2021.

NAVEED, H. et al. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2023.

Núcleo Interinstitucional de Linguística Computacional (NILC). *NILC Fake News Detection Tool*. 2025. <https://nilc-fakenews.herokuapp.com>. Acesso em: 17 fev. 2025.

OLAN, F. et al. Fake news on social media: the impact on society. *Information Systems Frontiers*, Springer, v. 26, n. 2, p. 443–458, 2024.

PAN, L. et al. Fact-checking complex claims with program-guided reasoning. *arXiv preprint arXiv:2305.12744*, 2023.

PAPADOGIANNAKIS, E. et al. Who funds misinformation? a systematic analysis of the ad-related profit routines of fake news sites. In: *Proceedings of the ACM Web Conference 2023*. [S.l.: s.n.], 2023. p. 2765–2776.

PAPAGEORGIOU, E. et al. A survey on the use of large language models (llms) in fake news. *Future Internet*, MDPI, v. 16, n. 8, p. 298, 2024.

PARTHASARATHY, V. B. et al. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, 2024.

PIRES, T. How multilingual is multilingual bert. *arXiv preprint arXiv:1906.01502*, 2019.

RAO, B. N.; KALYANI, V. A study on positive and negative effects of social media on society. *Journal of Science & Technology (JST)*, v. 7, n. 10, p. 46–54, 2022.

SAHOO, P. et al. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

SILVA, R. M. et al. Towards automatically filtering fake news in portuguese. *Expert Systems with Applications*, v. 146, p. 113199, 2020. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417420300257>.

SNOPES. *Available online:*. 2025. Accessed: 2025-02-18. Disponível em: <https://www.snopes.com>.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: pretrained bert models for brazilian portuguese. In: SPRINGER. *Brazilian conference on intelligent systems*. [S.l.], 2020. p. 403–417.

SUGGEST. *Available online:*. 2025. Accessed: 2025-02-18. Disponível em: <https://www.snopes.com>.

VASWANI, A. et al. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

VO, N.; LEE, K. Hierarchical multi-head attentive network for evidence-aware fake news detection. *arXiv preprint arXiv:2102.02680*, 2021.

WANG, W. Y. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

WANG, X. et al. Searching for best practices in retrieval-augmented generation. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. [S.l.: s.n.], 2024. p. 17716–17736.

WANG, Y. et al. Self-knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002*, 2023.

WU, J. et al. Adversarial contrastive learning for evidence-aware fake news detection with graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 36, n. 11, p. 5591–5604, 2023.

WU, L. et al. Evidence-aware hierarchical interactive attention networks for explainable claim verification. In: *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*. [S.l.: s.n.], 2021. p. 1388–1394.

XIONG, M. et al. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*, 2023.

YANG, A. et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

ZHANG, X.; GHORBANI, A. A. An overview of online fake news: Characterization, detection, and discussion. *Information Processing & Management*, Elsevier, v. 57, n. 2, p. 102025, 2020.

A Appendix

A.1 Prompts

```
Instructions: Based on the provided web search results, analyze in
   english whether the information has enough evidence to decide
   whether the claim is true or false.
Claim: {search_text}
Web search result: {context_str}
Respond "1" if this claim is true, respond "0" if this claim is false
   . If you think you need more information, respond "2".
You must provide your evaluation of the claim in the following format
   :
Explanation: [Explain why you make this judgment.]
Answer: [0/1/2]
Query: [Paraphrase the statement so that it is more suitable for web
   search engines to search for evidence.]
Compressed information: [Compressed information]
Confidence: [0~100%]
Don't forget to provide the Confidence.
```

Listing A.1 – English Reasoning Prompt

```
Instructions: Given a claim, I want to verify the truth or falsehood
   of the claim, help me paraphrase the statement so that it is more
   suitable for websearch engines to search for evidence.
Claim: {search_text}
Please provide your output in following format:
Query: [paraphrase the statement so that it is more suitable for
   websearch engines to search for evidence.]
```

Listing A.2 – English HyDE Prompt

```
Instructions: Analyze and judge if the claim is true or false.
Claim: {search_text}
```

```
Please provide your evaluation of the claim based on the following
   format:
Explanation: [Explain why you make this judgement.]
Answer: [0/1] (0 if this claim is false or 1 if this claim is true)
Remember to fill in all required fields based on your judgement. You
   must and can only choose one answer from 1 or 0.''']
```

Listing A.3 – English Benchmark Prompt

```
Instruções: Com base nos resultados da pesquisa na web fornecidos,
   analise em portuguêsse as informações possuem evidências
   suficientes para decidir se a notícia é verdadeira ou falsa.
Notícia: {search_text}
Resultado da pesquisa na web: {context_str}
Responda "1" se a notícia for verdadeira e "0" se a notícia for falsa
   .
Se você achar que precisa de mais informações, responda "2".
Você deve fornecer sua avaliação da notícia no seguinte formato:
Explicação: [Explique por que você fez esse julgamento.]
Resposta: [0/1/2]
Consulta: [Parafrasear a notícia para que ela seja mais adequada para
    os mecanismos de busca na web procurarem por evidências.]
Informação comprimida: [Informação resumida]
Confiança: [0~100%]
Não esqueça de fornecer a Confiança.
```

Listing A.4 – Portuguese Reasoning Prompt

```
Instruções: Dada uma notícia, eu quero verificar a verdade ou
   falsidade da notícia, ajude-me a parafrasear a declaração para que
    ela seja mais adequada para os mecanismos de busca na web
   procurarem por evidências.
Notícia: {search_text}
Por favor, forneça sua resposta seguinte formato:
Consulta: [Parafrasear a declaração para que ela seja mais adequada
   para os mecanismos de busca na web procurarem por evidências.]
```

Listing A.5 – Portuguese HyDE Prompt

```
Instruções: Analise e julgue se a notícia é verdadeira ou falsa.
Notícia: {search_text}
Forneça sua avaliação da notícia com base no seguinte formato:
Explicação: [Explique por que você fez esse julgamento.]
Resposta: [0/1] (0 se esta notícia for falsa ou 1 se esta notícia for
    verdadeira)
Lembre-se de preencher todos os campos obrigatórios com base no seu
   julgamento. Você deve e só pode escolher uma resposta entre 1 ou
   0.''']
```

Listing A.6 – Portuguese Benchmark Prompt

A.2 Exemplo de saídas dos modelos

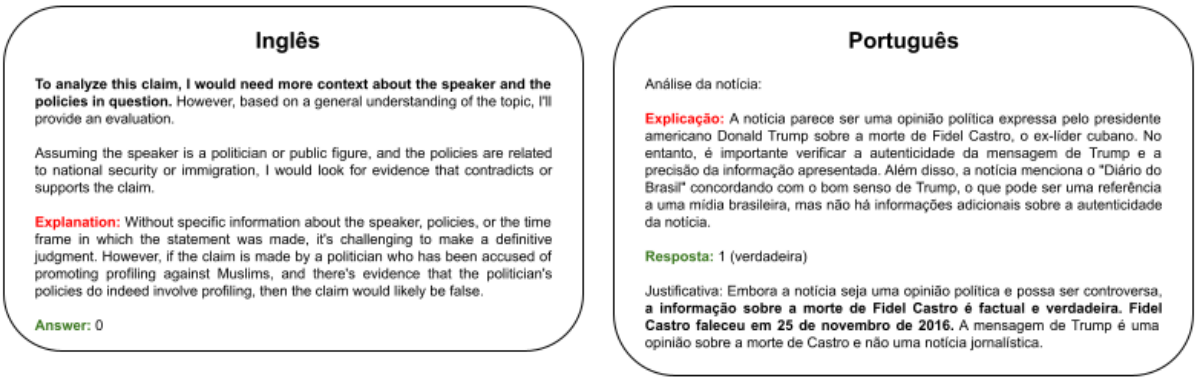

Figura 12 – Exemplo de saída do Llama3.1-8B-Instruct. Fonte: Autoria própria

,

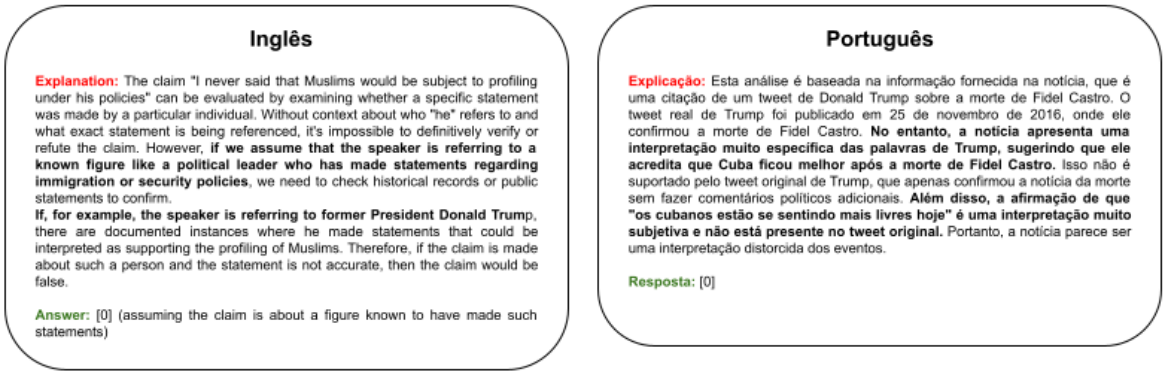

Figura 13 – Exemplo de saída do Qwen2.5-7B-Instruct. Fonte: Autoria própria

,

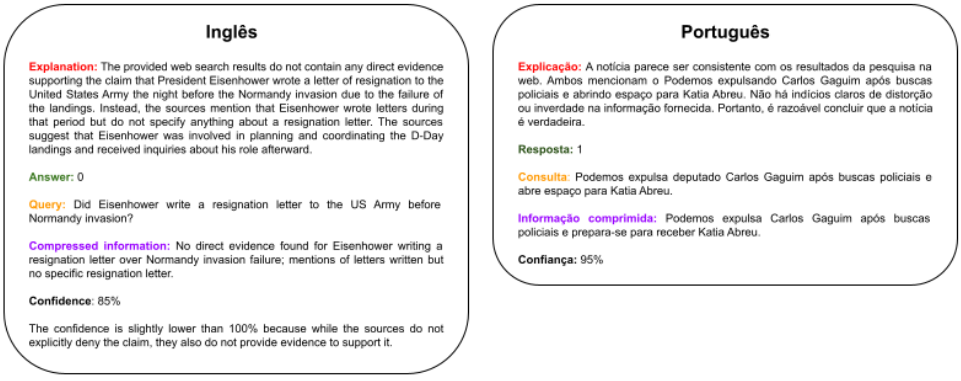

Figura 14 – Exemplo de saída do STEEL_HyDE. Fonte: Autoria própria

,