DESEMPENHO DE MODELOS DE LINGUAGEM PARA CLASSIFICAÇÃO DE *FAKE NEWS*

Douglas Castro da Silva

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Junho de 2025

DESEMPENHO DE MODELOS DE LINGUAGEM PARA CLASSIFICAÇÃO DE *FAKE NEWS*

Douglas Castro da Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Geraldo Bonorino Xexéo

Aprovada por: Prof. Geraldo Bonorino Xexéo
Prof. Jano Moreira de Souza
Prof. Leandro Guimarães Marques Alvim

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2025

Castro da Silva, Douglas

DESEMPENHO DE MODELOS DE LINGUAGEM PARA CLASSIFICAÇÃO DE *FAKE NEWS*/Douglas Castro da Silva. – Rio de Janeiro: UFRJ/COPPE, 2025.

XIV, 120 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 99 – 107.

1. fake news. 2. LLM. I. Bonorino Xexéo, Geraldo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A alguém cujo valor é digno
desta dedicatória.

Agradecimentos

Gostaria de expressar minha profunda gratidão ao meu orientador, Geraldo Bonorino Xexéo, pela dedicação, paciência e valioso apoio durante todo o curso. Sua orientação e incentivo foram fundamentais para o desenvolvimento deste trabalho e para minha formação acadêmica e pessoal.

Agradeço também à minha família, que esteve sempre ao meu lado, oferecendo suporte e motivação. Em especial, agradeço à minha esposa, Thalyta de Abreu Botino, e ao meu filho, Gael Botino Castro, cuja presença e incentivo diário foram minha maior fonte de força e inspiração para superar os desafios desta jornada.

Agradecimentos as agências de pesquisa Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, Amazon Web Services - AWS, CAPES e FAPERJ pelo suporte e a Chamada CNPq/AWS Nº 64/2022 – Acesso às Plataformas de Computação em Nuvem da AWS (Cloud Credits for Research) que viabilizaram essa pesquisa.

A todos que, direta ou indiretamente, contribuíram para a realização deste trabalho, o meu sincero obrigado.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DESEMPENHO DE MODELOS DE LINGUAGEM PARA CLASSIFICAÇÃO DE *FAKE NEWS*

Douglas Castro da Silva

Junho/2025

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Esta dissertação investiga a relação de custo-benefício de grandes modelos de linguagem (LLMs) na tarefa de detecção de fake news, no contexto do projeto interdisciplinar "Inteligência Artificial para Detecção Precoce de Fake News". Inicialmente concebido para analisar dados em tempo real da plataforma Twitter (atualmente X), o projeto foi redirecionado para o uso de bases de dados pré-existentes devido a mudanças na política de acesso da plataforma. O estudo avalia diversos LLMs disponíveis na AWS, aplicando-os em múltiplas bases de notícias falsas e explorando diferentes estratégias de prompt engineering e técnicas de ajuste fino.

Três questões principais norteiam esta pesquisa: (i) LLMs são capazes de identificar fake news de forma confiável? (ii) Como seu desempenho se compara aos métodos tradicionais do estado da arte? (iii) Qual é a relação entre o tamanho do modelo, a acurácia e o custo operacional? Para apoiar essa análise, propõe-se uma nova métrica chamada PoC-score (Performance over Cost), que quantifica a eficiência de cada modelo relacionando seu F1-score ao custo por hora de operação.

Os resultados mostram que os LLMs apresentam bom desempenho em conjuntos de dados com conteúdo mais rico e estruturado, como notícias sobre COVID-19, mas enfrentam limitações em bases mais ambíguas ou ruidosas, como LIAR e PolitiFact. Embora modelos maiores tendam a obter acurácia ligeiramente superior, seu custo cresce de forma desproporcional, tornando modelos menores com ajuste fino — como o Gemma-7B — mais atrativos em cenários reais. Em suma, os experimentos demonstram que os LLMs são ferramentas promissoras para a detecção de fake news, mas sua adoção deve considerar as características do domínio, as restrições computacionais e os objetivos da aplicação.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

PERFORMANCE OF LANGUAGE MODELS FOR FAKE NEWS CLASSIFICATION

Douglas Castro da Silva

June/2025

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

This dissertation investigates the cost-effectiveness of large language models (LLMs) in the task of fake news detection, within the scope of the interdisciplinary project "Artificial Intelligence for Early Detection of Fake News." Initially designed to analyze real-time data from Twitter (now X), the project was redirected to use pre-existing datasets due to changes in the platform's access policy. The study evaluates various LLMs available on AWS across multiple fake news datasets, exploring their performance under different prompting strategies and fine-tuning techniques.

Three key research questions guide this work: (i) Are LLMs capable of reliably identifying fake news? (ii) How do their performances compare to traditional state-of-the-art methods? (iii) What is the trade-off between model size, accuracy, and operational cost? To support this evaluation, a novel metric called PoC-score (Performance over Cost) is proposed, quantifying the efficiency of each model by relating its F1-score to its cost per hour.

Results show that LLMs perform well on datasets with richer and more structured content, such as COVID-related news, but face limitations on more ambiguous or noisy datasets like LIAR or PolitiFact. While larger models tend to achieve slightly higher accuracy, their cost escalates disproportionately, making smaller models with fine-tuning strategies—such as Gemma-7B—more attractive in real-world deployments. Ultimately, the findings suggest that LLMs are promising tools for fake news detection, but their adoption should consider domain characteristics, computational constraints, and application goals.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Vivemos em uma era marcada pela comunicação digital instantânea. Plataformas como Twitter, Facebook, YouTube e outras redes sociais tornaram-se canais poderosos para a disseminação de informações. Contudo, essa velocidade na propagação de conteúdo também facilita a circulação de *fake news* e outras formas de desinformação em escala global. Esse fenômeno gera impactos sociais significativos, especialmente em contextos sensíveis como política, saúde pública e economia. No Brasil, a gravidade do problema levou o Congresso Nacional a instaurar uma Comissão Parlamentar Mista de Inquérito (CPMI) para investigar a disseminação de notícias falsas e seus efeitos na sociedade (BARBIÉRI, 2019).

Diante desse cenário, torna-se essencial o desenvolvimento de sistemas capazes de detectar e mitigar a circulação de informações falsas. Este trabalho tem como objetivo avaliar o uso de modelos de linguagem de grande porte (LLMs — *Large Language Models*) aplicados à tarefa de detecção de *fake news*, utilizando técnicas de engenharia de prompt como principal abordagem. Para isso, foram utilizados diferentes modelos de LLMs, variando desde arquiteturas menores, como o *Gemma 2B*, até modelos de porte muito maior, como o *Llama 3.1 70B*. Além disso, este estudo abrange múltiplos conjuntos de dados relacionados à detecção de desinformação, permitindo uma análise comparativa sobre a eficácia desses modelos em diferentes cenários.

O desenvolvimento deste trabalho busca não apenas compreender as limitações e potencialidades dos LLMs para esta tarefa específica, mas também contribuir com reflexões sobre o uso responsável dessas tecnologias no combate à desinformação.

1.1 Motivação

A disseminação de *fake news* tem se tornado um problema crescente na sociedade contemporânea, com impactos significativos nas esferas política, econômica, social, educacional, de saúde e científica. As *fake news* são conteúdos criados in-

tencionalmente para enganar, desinformar ou manipular, o que torna sua detecção particularmente desafiadora, principalmente quando baseada apenas no conteúdo textual (KARIMI *et al.*, 2018; SHU *et al.*, 2017).

Eventos recentes ilustram como a desinformação pode gerar consequências graves. Na esfera política, as *fake news* tiveram papel central nas eleições presidenciais dos Estados Unidos em 2016 (ALLCOTT e GENTZKOW, 2017), impactando diretamente a opinião pública. Na saúde, durante a pandemia de COVID-19, a propagação de informações falsas sobre vacinas e tratamentos causou hesitação vacinal e prejuízos à saúde pública. No contexto econômico, empresas podem ser prejudicadas por boatos falsos que afetam suas ações ou reputação. Na ciência e na educação, a disseminação de desinformação compromete a confiança no método científico e na produção de conhecimento.

Além dos impactos sociais, há preocupações legais associadas à produção e disseminação de *fake news*. Como discutido por KLEIN e WUELLER (2017), produtores de desinformação enfrentam riscos civis, como processos por difamação, danos emocionais, violação de propriedade intelectual, fraude e práticas comerciais enganosas. Paralelamente, plataformas digitais têm adotado políticas mais rigorosas para restringir conteúdos falsos, incluindo suspensão de contas e bloqueio de receitas publicitárias.

O problema é agravado por características intrínsecas às *fake news*. Muitas vezes, essas notícias combinam informações falsas com elementos verdadeiros, dificultando sua identificação (SHU *et al.*, 2017). Além disso, elas podem apresentar diferentes graus de falsidade — como meias verdades, exageros ou distorções — enquanto a maioria dos modelos de detecção ainda trata o problema como uma classificação binária (verdadeiro ou falso), o que simplifica excessivamente a complexidade do fenômeno.

O ambiente das mídias sociais amplia ainda mais o desafio. Nessas plataformas, as *fake news* são frequentemente curtas, sensacionalistas, adaptadas a contextos locais e compartilhadas de forma viral. Por outro lado, esse ambiente também oferece oportunidades: a disponibilidade de dados contextuais — como informações sobre os autores, histórico de postagens, redes de disseminação e reações dos usuários — pode ser explorada para melhorar a eficácia dos sistemas de detecção.

Portanto, desenvolver métodos eficientes e robustos para detectar *fake news* é uma necessidade urgente, tanto para proteger indivíduos e instituições quanto para preservar a integridade da informação no ambiente digital. Este trabalho busca contribuir para esse objetivo, alinhando-se às discussões recentes na literatura e explorando abordagens que consideram múltiplas fontes de informação e múltiplos graus de falsidade.

1.2 Histórico do Projeto

O projeto Inteligência Artificial para Detecção Precoce de Fake News (processo 421793/2022-8), foi aceito na Chamada CNPq/AWS Nº 64/2022 - Faixa A, reunindo um grupo multidisciplinar de pesquisadores da UFRJ pertencentes ao LINE – Laboratório de Tratamento da Informação Não Estruturada, do Prorama de de Engenharia de Sistemas e Computação da da COPPE, ao Instituto de Pesquisa e Planejamento Urbano e Regional e ao Departamento de Administração da Faculdade de Administração e Ciências Contábeis.

Na vertente computacional, a qual pertence esse trabalho, o foco é o desenvolvimento de um conjunto de ferramentas, guias e práticas baseadas no estado-da-arte de Inteligência Artificial, capazes de analisar as redes sociais em busca de narrativas falsas; e outra vertente sociotécnica na qual o foco é a criação e validação de teorias e modelos explicativos para a análise das narrativas que vêm dividindo a sociedade em grupos, ou bolhas, principalmente das falsas narrativas, seus agentes principais e que papéis assumem.

O projeto era fortemente baseado na premissa de acesso a dados da então plataforma de *microblogging* Twitter, hoje renomeada "X", de forma autorizada a entidades acadêmicas e de pesquisa, sem custo, pela administração da empresa na época. Logo após aprovado o projeto, essa forma de acesso foi proibida, passando a existir apenas uma versão paga, a preços não exequíveis para o projeto.

Dessa maneira, o grupo de pesquisa foi forçado a buscar outros caminhos, sendo que este trabalho, anteriormente focado em mensagens adquiridas em tempo real, se reorientou para o uso de bases de dados pré-existentes, buscando projetar relações de custo-benefício entre a qualidade da detecção de *fake news* e o custo de executar essa detecção com LLMs de diversos tamanhos.

1.3 Objetivo da Dissertação

O principal objetivo desta dissertação é avaliar, de forma detalhada e crítica, a relação de custo-benefício entre diferentes modelos de grandes modelos de linguagem (*LLMs*), utilizando variadas técnicas de aplicação, com o intuito de fundamentar a criação futura de um sistema de alarme que possa notificar os usuários sobre a confiabilidade do conteúdo consumido. Tal sistema visa contribuir para a formação de um público mais criterioso e agnóstico em relação às informações recebidas, reduzindo a propagação indevida de notícias falsas e outras formas de desinformação.

Para orientar essa investigação, a dissertação busca responder às seguintes perguntas de pesquisa:

1. **LLMs são capazes de identificar notícias falsas?** Avaliar a capacidade

dos modelos de linguagem em realizar a tarefa de detecção de *fake news*, considerando diferentes domínios, formatos e contextos.

2. **Como os LLMs se comparam em desempenho com o estado da arte?** Realizar uma comparação direta entre os resultados obtidos pelos LLMs e os melhores resultados reportados na literatura para cada uma das coleções utilizadas, analisando se esses modelos conseguem atingir, igualar ou superar os métodos tradicionais ou especializados.

3. **Qual é a relação de custo-benefício entre LLMs de diferentes modelos e tamanhos?** Investigar como o desempenho dos LLMs se relaciona com seu custo operacional, considerando fatores como consumo de recursos, tempo de inferência e escalabilidade, de modo a entender quais modelos oferecem a melhor relação entre acurácia e viabilidade econômica.

Para apoiar essa análise, esta dissertação também propõe uma nova métrica de avaliação chamada *PoC-score* (*Performance over Cost score*), que tem como objetivo quantificar a eficiência econômica dos modelos em relação ao seu desempenho. A métrica é definida como a razão entre o F1-score e o custo operacional do modelo (medido em USD/h). Essa abordagem permite avaliar não apenas a performance bruta dos modelos, mas também sua viabilidade prática em termos de custo-benefício, incentivando escolhas mais alinhadas com restrições operacionais do mundo real.

Para alcançar esses objetivos, a dissertação realizou uma avaliação rigorosa do desempenho de diversos *LLMs* disponíveis na plataforma AWS, submetendo-os a testes em múltiplas bases de dados contendo notícias falsas, com características e contextos variados. Foram exploradas diferentes estratégias de *prompt engineering* e técnicas de ajuste fino, buscando entender o comportamento dos modelos em cenários diversificados e identificar as abordagens que promovem maior precisão e menor ocorrência de alucinações.

Além disso, a pesquisa investigou a eficiência computacional dos modelos, enfatizando a importância de avaliar o custo de inferência em paralelo ao desempenho, dado que fatores como tempo de resposta e uso de recursos computacionais são determinantes para a aplicabilidade prática em ambientes de produção. Observou-se também que o desempenho dos modelos varia significativamente conforme o domínio dos dados, sendo superior em bases relacionadas à área da saúde, como a COVID-19, possivelmente devido ao conhecimento prévio incorporado durante o treinamento dos modelos, e inferior em bases com temas mais complexos ou diversificados, como notícias políticas e de entretenimento.

Por fim, a dissertação destacou a necessidade de explorar estratégias avançadas, como a combinação de múltiplos modelos por meio de técnicas de *ensemble*, além

da adoção de métodos para lidar com desafios impostos por textos muito longos ou muito curtos, incluindo pré-processamento, segmentação e incorporação de informações estruturadas. Essas direções são essenciais para aprimorar a robustez, a precisão e a escalabilidade dos sistemas de detecção automática de *fake news*, contribuindo para sua adoção efetiva em contextos reais.

1.4 Descrição da Dissertação

Esta introdução compõe o primeiro capítulo desta dissertação. A seguir, o Capítulo 2 apresenta uma revisão da literatura relacionada à detecção de *fake news*, abordando os conceitos fundamentais, os impactos da desinformação, as principais características de notícias falsas, além das estratégias utilizadas para sua identificação e uma análise dos trabalhos mais relevantes sobre o tema.

O Capítulo 3 trata dos *Large Language Models* (LLMs) com arquitetura *Transformer*, explicando o papel dos *embeddings*, os fundamentos da arquitetura Transformer e as principais abordagens de engenharia de *prompt*, como *zero-shot*, *few-shot*, *prompt priming* e *output constraint*.

O Capítulo 4 descreve a metodologia adotada no desenvolvimento deste trabalho, incluindo a definição do cenário de aplicação, a formulação da tarefa, a escolha das bases de dados, a elaboração dos *prompts*, a seleção dos modelos utilizados e os critérios de avaliação empregados, bem como a realização de *fine-tuning* nos modelos.

O Capítulo 5 apresenta os experimentos realizados, detalhando a execução do processo metodológico, as bases utilizadas, os resultados obtidos com diferentes modelos e configurações.

No Capítulo 6, realizamos uma análise comparativa abrangente entre os modelos avaliados, considerando tanto métricas tradicionais de desempenho, como *accuracy*, *precision*, *recall* e *F1-score*, quanto métricas econômicas, como tempo de inferência e custo operacional por hora. Essa análise permitiu compreender não apenas quais modelos apresentam maior capacidade de detecção de *fake news*, mas também quais oferecem melhor relação custo-benefício para diferentes cenários e restrições operacionais. Além disso, o capítulo introduz e aplica a métrica *PoC-score*, proposta nesta dissertação, como uma ferramenta complementar para avaliar a eficiência dos modelos, destacando casos em que modelos ligeiramente inferiores em acurácia podem ser preferíveis devido a custos significativamente menores. Também são discutidas as variações de desempenho conforme o domínio dos dados, o tamanho dos modelos e as estratégias de *prompt engineering* adotadas.

Por fim, o Capítulo 7 apresenta as conclusões gerais do trabalho, sintetizando os principais achados da pesquisa em termos de desempenho, custo e aplicabilidade dos modelos de linguagem para a tarefa de detecção de notícias falsas. São discutidas

as implicações práticas dos resultados, tanto para o desenvolvimento de sistemas de monitoramento de desinformação quanto para o uso responsável de LLMs em ambientes produtivos. Além disso, são apresentadas sugestões para trabalhos futuros.

Capítulo 2

Revisão da Literatura sobre Detecção de *Fake News*

A crescente preocupação com a propagação de *fake news* tem impulsionado pesquisas voltadas ao desenvolvimento de métodos eficazes para sua detecção e contenção. Diversas abordagens têm sido propostas, combinando técnicas de processamento de linguagem natural, aprendizado de máquina e análise de redes sociais para identificar padrões característicos da desinformação. Além disso, estratégias como a verificação de fatos e a análise da credibilidade das fontes desempenham um papel essencial nesse processo. Essas iniciativas demonstram a importância de uma abordagem multidisciplinar e contínua para enfrentar a natureza dinâmica e adaptativa das *fake news*, ressaltando a necessidade de soluções robustas e escaláveis, aliadas à promoção da educação midiática entre os usuários PYARELAL *et al.* (2023).

A detecção automática de *fake news* tem ganhado destaque como área central de pesquisa, impulsionada pela rápida disseminação de desinformação nas redes sociais e seu alto potencial de viralização. Inicialmente, essa tarefa era abordada com algoritmos tradicionais de aprendizado de máquina, como *Naive Bayes*, *Support Vector Machines* (SVM) e *Random Forests* (BREIMAN, 2001; CORTES e VAPNIK, 1995), aplicados a representações simples do texto, como *bag-of-words* e *TF-IDF* (JONES, 1972; SALTON *et al.*, 1975). Embora eficazes em cenários controlados, essas abordagens mostraram-se limitadas na generalização e na detecção de sutilezas linguísticas frequentemente presentes em conteúdos manipulativos.

Com a evolução das técnicas de Processamento de Linguagem Natural (*NLP*), abordagens mais avançadas passaram a ser adotadas, especialmente aquelas baseadas em *deep learning*. Modelos como Redes Neurais Convolucionais (CNNs) e Redes Neurais Recorrentes (RNNs), incluindo LSTM, possibilitaram a captura de dependências contextuais e padrões sintáticos e semânticos mais profundos nos textos, aprimorando significativamente os resultados na tarefa de identificação de desinformação. Estratégias complementares, como a análise linguística e a detecção de

subjetividade, também passaram a ser exploradas com sucesso, como ilustrado no estudo de WU e WANG (2021).

Além disso, tarefas auxiliares de *NLP*, como a Análise de Sentimentos e a Detecção de Posicionamento (*Stance Detection*), têm sido integradas aos modelos de classificação de *fake news* com o objetivo de fornecer informações adicionais e enriquecer a compreensão do conteúdo textual. Essas técnicas ajudam a capturar as intenções e emoções subjacentes às mensagens, aspectos frequentemente explorados na construção de narrativas falsas KÜÇÜK e CAN (2021).

Neste capítulo, exploramos os principais conceitos relacionados à desinformação, os métodos de representação textual em *NLP*, as abordagens de aprendizado de máquina e de aprendizado profundo aplicadas à detecção de *fake news*, bem como os conjuntos de dados e métricas utilizados para avaliação. Ao final, apresentamos uma análise crítica da literatura revisada, destacando limitações e desafios ainda presentes na área.

2.1 Conceitos Fundamentais

O termo *fake news* refere-se a notícias fabricadas deliberadamente para enganar os leitores, geralmente com o objetivo de obter algum benefício político ou financeiro. Segundo ALLCOTT e GENTZKOW (2017), *fake news* são "artigos de notícias que são intencionalmente e verificavelmente falsos e que podem enganar os leitores". Essa definição distingue as *fake news* de informações incorretas não intencionais (como erros jornalísticos) ou sátiras evidentes, concentrando-se em conteúdos projetados para enganar.

Além da definição conceitual, o estudo de ALLCOTT e GENTZKOW (2017), que analisou a disseminação de notícias falsas nas redes sociais durante as eleições presidenciais dos Estados Unidos em 2016, identificou duas motivações principais para a produção e disseminação desse tipo de conteúdo.

A primeira motivação é de ordem econômica: conteúdos virais atraem um grande número de acessos, o que pode resultar em receitas publicitárias significativas. Muitos produtores de *fake news*, cujas identidades foram posteriormente reveladas, estavam mais interessados nos lucros obtidos por meio da monetização de tráfego do que em influenciar diretamente o cenário político.

A segunda motivação é ideológica. Certos grupos ou indivíduos disseminam *fake news* com a intenção de favorecer candidatos ou causas políticas alinhadas com suas crenças ou interesses. Essa forma de manipulação informacional pode ter efeitos significativos sobre a opinião pública, especialmente em períodos eleitorais, tornando ainda mais urgente o desenvolvimento de mecanismos de detecção e contenção da desinformação.

Essa definição adotada por ALLCOTT e GENTZKOW (2017) é operacional e bastante difundida na literatura, mas não é isenta de críticas. Por exemplo, KLEIN e WUELLER (2017) argumentam que o termo *fake news* é impreciso e frequentemente utilizado de forma política para deslegitimar fontes de informação. Em vez disso, os autores preferem discutir a desinformação como parte de um ecossistema mais amplo, que inclui tanto a disseminação intencional de informações falsas (*disinformation*) quanto a propagação não intencional (*misinformation*).

Além das motivações econômicas e ideológicas apontadas por ALLCOTT e GENTZKOW (2017), KLEIN e WUELLER (2017) destacam que fatores sociotécnicos, como algoritmos de recomendação, bolhas informacionais e dinâmicas de polarização nas redes sociais, desempenham papel central na amplificação da desinformação. Essa perspectiva amplia o entendimento do problema, mostrando que o combate à desinformação não depende apenas da identificação do conteúdo falso, mas também de intervenções estruturais nas plataformas digitais e no próprio ecossistema informacional.

De acordo com (WARDLE e DERAKHSHAN, 2017), existem três principais tipos de distúrbios da informação (*information disorder*) que ajudam a categorizar as diferentes formas de desinformação nas mídias. O primeiro tipo é a *misinformation* (desinformação), que envolve a disseminação de informações falsas ou imprecisas sem a intenção de enganar. O segundo tipo é a *disinformation* (desinformação intencional), que é a criação e disseminação de informações falsas com a intenção deliberada de manipular os receptores. O terceiro tipo é a *malinformation* (malinformação), que consiste no uso de informações verdadeiras de forma prejudicial ou para causar dano a indivíduos ou grupos. Esses três tipos são fundamentais para entender a complexidade das fake news e suas implicações sociais e políticas. A Figura 2.1 ilustra a relação entre essas categorias, destacando como se posicionam em relação aos eixos de falsidade e dano.

Casos de *malinformation*, por exemplo, incluem discurso de ódio e assédio, nos quais informações verdadeiras são usadas estrategicamente para prejudicar indivíduos, frequentemente com base em aspectos como religião, origem ou afiliações pessoais.

2.1.1 Impactos das *Fake News*

A disseminação de *fake news* tem implicações significativas tanto no plano social quanto político. (HARRIS *et al.*, 2024) analisam os impactos sociais e políticos da disseminação de *fake news* sob duas principais dimensões: psicológica e econômica, destacando como ambas contribuem significativamente para a escala e persistência desse fenômeno.

Figura 2.1: Representação visual dos três tipos de distúrbios da informação. Fonte: Adaptado de (WARDLE e DERAKHSHAN, 2017).

Dimensão psicológica: No plano psicológico, a propagação de *fake news* está fortemente associada a mecanismos cognitivos que influenciam o comportamento dos usuários. Viéses como o efeito manada, a partidarização e o chamado eco de crença levam os indivíduos a aceitar e compartilhar informações falsas, muitas vezes sem verificar seu conteúdo. A superficialidade no consumo de notícias, como a leitura apenas de manchetes, também colabora para esse cenário, amplificado por títulos chamativos (*clickbait*). Além disso, o compartilhamento de informações falsas ocorre frequentemente de forma reativa e automática, motivado por desejos de autoexpressão, status social ou intenção de ajudar, mesmo sem consciência da falsidade da informação.

Dimensão econômica: No aspecto econômico, a digitalização da produção e distribuição de notícias alterou radicalmente o modelo midiático tradicional, substituído por um sistema algorítmico e multifacetado. Atualmente, cerca de dois terços dos usuários acessam notícias por meio de motores de busca, redes sociais e outras plataformas digitais (NEWMAN *et al.*, 2021), que priorizam conteúdos de alto engajamento para maximizar receita publicitária. Esse ambiente favorece a circulação de *fake news*, conteúdos sensacionalistas e informações enganosas, o que, por sua vez, tem minado a confiança do público na mídia tradicional. A facilidade e o baixo custo de produção e disseminação de notícias falsas, aliados à ausência de punições legais para seus autores, tornam o fenômeno ainda mais preocupante. Plataformas digitais, ao explorarem vieses cognitivos como a novidade e a confirmação, acabam incentivando a viralização de *fake news* como parte de um modelo de negócios

lucrativo.

Dando sequência aos impactos sociais da disseminação de *fake news*, um estudo recente de (ALI *et al.*, 2023) investigou a relação entre a propagação de notícias falsas e o comportamento de compra por pânico durante a pandemia de COVID-19 no Reino Unido. O estudo revelou que, em média, 33% das postagens sobre compras por pânico coletadas nas redes sociais foram classificadas como *fake news*. A pesquisa comparou o comportamento em áreas urbanas e rurais, observando que as postagens em áreas urbanas ocorreram predominantemente entre janeiro e outubro de 2021, com um pico notável em setembro, com palavras-chave relacionadas ao abastecimento de combustíveis e transporte, como "fuel", "shortage" e "petrol". Por outro lado, nas áreas rurais, o pico de postagens foi observado tanto em março de 2020 quanto em setembro de 2021, com palavras-chave como "toilet rolls", "food", "shop", e menções a grandes redes de supermercados como "Sainsbury's" e "Asda", associadas ao comportamento de compra por pânico.

Os resultados mencionados acima também indicaram que a maioria das postagens em áreas rurais foi marcada por uma maior presença de *fake news*, o que sugere uma maior vulnerabilidade a boatos e desinformação nessas regiões. Além disso, a análise dos dados revelou que as áreas rurais, possivelmente por estarem em um estágio mais inicial de preparação digital, mostraram-se mais suscetíveis à disseminação de informações errôneas, o que pode ser relacionado a uma maior dificuldade em questionar notícias falsas e ao medo exacerbado de escassez de produtos essenciais, especialmente em tempos de pandemia, quando o acesso a cuidados de saúde e recursos pode ser limitado(ALI *et al.*, 2023).

Esses achados apresentados nessa subseção reforçam a necessidade urgente de desenvolver estratégias para mitigar a propagação de *fake news*, especialmente em contextos de crise, e destacam a complexa interação entre a digitalização, os comportamentos sociais e as dinâmicas de disseminação de informações falsas.

2.1.2 Características de Notícias Falsas

A identificação de *fake news* é um desafio complexo, dado que muitas vezes essas notícias se apresentam de forma sutil, disfarçadas em conteúdos que podem parecer legítimos à primeira vista. As características de *fake news* não se limitam a elementos superficiais como títulos chamativos, mas envolvem uma série de fatores estruturais, emocionais e contextuais que tornam sua detecção difícil, mesmo para leitores atentos.

Uma das principais dificuldades reside no fato de que muitas *fake news* utilizam *estratégias de persuasão* sofisticadas para manipular o leitor. Ao contrário de conteúdos evidentemente falsos, que são facilmente identificáveis, as notícias falsas

tendem a se misturar com informações verdadeiras ou distorcidas, criando uma fachada de veracidade. Esse fenômeno é conhecido como *desinformação estratégica*, onde a falsidade se esconde por trás de uma apresentação aparentemente confiável, muitas vezes utilizando dados reais ou fontes legítimas, mas com a intenção de distorcer a realidade ou desinformar. Um exemplo notório é o caso do "Pizzagate", em que uma teoria conspiratória durante as eleições americanas de 2016 utilizou nomes e locais reais para divulgar uma narrativa completamente falsa sobre um esquema de tráfico infantil supostamente ligado a membros do Partido Democrata (UOL, 2024).

Além disso, muitas *fake news* são projetadas para *explorar os vieses cognitivos dos leitores*, como o viés de confirmação e a tendência a confiar em fontes que compartilham suas crenças. Um estudo realizado por pesquisadores associados ao Programa de Doutorado em Direito da Universidade de Brasília (UnB) e ao Mestrado em Comunicação Digital do Instituto Brasileiro de Ensino (METROPOLES, 2024), Desenvolvimento e Pesquisa (IDP) revelou que conteúdos desinformativos que evocam emoções intensas, como raiva, medo ou indignação, têm maior probabilidade de serem compartilhados. O estudo identificou que temas que tocam a moralidade e utilizam categorias religiosas, como "pecado"ou "salvação", geram reações emocionais fortes, ampliando a capacidade de replicação dessas mensagens nas redes sociais.

A *manipulação visual* também desempenha um papel significativo nas *fake news*. O uso de imagens ou vídeos alterados, muitas vezes retirados de contexto, é uma tática comum para aumentar a plausibilidade do conteúdo falso. Um exemplo marcante desse tipo de manipulação ocorreu durante as eleições brasileiras de 2018, quando circulou nas redes sociais uma imagem adulterada da então candidata à vice-presidência Manuela D'Ávila. Na montagem, ela aparecia vestindo uma camiseta com a frase "Jesus é Travesti", acompanhada de um arco-íris — algo que não constava na imagem original, em que a camiseta exibia outra mensagem. A adulteração visava associá-la a pautas polêmicas, influenciando negativamente a percepção pública durante o período eleitoral (G1, 2018). Esse caso ilustra como imagens manipuladas, que aparentam ser autênticas à primeira vista, podem reforçar narrativas enganosas e gerar fortes reações emocionais. Em ambientes digitais, onde o conteúdo visual é frequentemente interpretado como prova de autenticidade, tais manipulações tornam-se uma ferramenta poderosa de desinformação, dificultando ainda mais a detecção de *fake news*.

Em face dessas características complexas, a identificação de *fake news* requer não apenas a análise de conteúdo textual, mas também uma compreensão profunda dos contextos emocionais, sociais e cognitivos envolvidos na disseminação de informações falsas. As soluções para esse problema, portanto, demandam abordagens sofisticadas, que envolvem técnicas de processamento de linguagem natural, análise

de redes sociais e métodos de verificação de fontes. Além disso, é fundamental que os usuários desenvolvam uma postura crítica em relação ao consumo de informações, uma vez que, apesar de todo o avanço das tecnologias de detecção, as *fake news* continuam sendo uma ameaça persistente e em constante evolução.

2.2 Estratégias Utilizadas na Identificação de *Fake News*

A detecção de *fake news* começou com técnicas de representação de texto simples e interpretáveis. Entre as primeiras abordagens estão o *Bag of Words* (BoW) e o *Term Frequency* (TF)(JONES, 1972; SALTON *et al.*, 1975). Ambas representam textos como vetores numéricos, baseando-se na contagem de palavras. O BoW considera apenas a presença ou frequência das palavras em um documento, ignorando a ordem e o contexto. O TF aprimora isso ao normalizar pela frequência total de palavras no documento SALTON e BUCKLEY (1988).

Considere o exemplo com duas frases curtas:

- Documento 1: *"Notícia falsa se espalha rápido"*
- Documento 2: *"Notícia verdadeira circula devagar"*

O vocabulário conjunto é: *{"notícia", "falsa", "se", "espalha", "rápido", "verdadeira", "circula", "devagar"}.*

A representação em **BoW** para o Documento 1 seria:

$$[1, 1, 1, 1, 1, 0, 0, 0]$$

A representação em **TF-IDF** pondera a frequência da palavra pelo inverso da frequência nos documentos, usando a fórmula (JONES, 1972):

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \log\left(\frac{N}{\text{DF}(t)}\right)$$

onde t é o termo, d é o documento, D é o conjunto de documentos, N é o número total de documentos, e $\text{DF}(t)$ é o número de documentos contendo o termo t.

Considere o seguinte vocabulário (8 palavras): *{"notícia", "falsa", "se", "espalha", "rápido", "verdadeira", "circula", "devagar"}* e os dois documentos:

- Documento 1: *"Notícia falsa se espalha rápido"*
- Documento 2: *"Notícia verdadeira circula devagar"*

As frequências por termo ($\text{DF}(t)$) nos dois documentos são:

- Termos que aparecem em ambos os documentos: "notícia"(DF = 2)
- Termos exclusivos de um documento: todos os outros (DF = 1)

Com isso, para o Documento 1, temos:

$$\text{TF-IDF}_1 = \left[0,\ \frac{1}{5}\log\left(\frac{2}{1}\right),\ \frac{1}{5}\log\left(\frac{2}{1}\right),\ \frac{1}{5}\log\left(\frac{2}{1}\right),\ \frac{1}{5}\log\left(\frac{2}{1}\right),\ 0,\ 0,\ 0\right]$$

$$\text{TF-IDF}_1 \approx [0,\ 0{,}139,\ 0{,}139,\ 0{,}139,\ 0{,}139,\ 0,\ 0,\ 0]$$

Note que o valor associado ao termo "notícia"é zero, pois ele aparece em todos os documentos ($\log\left(\frac{2}{2}\right) = 0$), enquanto os termos exclusivos do documento têm maior peso.

Esses vetores escassos e ponderados formam a base para classificação com algoritmos como *Naive Bayes*, *Logistic Regression* (COX, 1958; LEWIS, 1998) e *SVM* (CORTES e VAPNIK, 1995), amplamente usados nas primeiras abordagens de detecção de *fake news*.

Em seguida, técnicas de *embedding* de palavras, como o *Word2Vec* (MIKOLOV *et al.*, 2013b), foram introduzidas, permitindo representar palavras em vetores densos, capturando relações semânticas entre elas. Já o *Doc2Vec* (LE e MIKOLOV, 2014), uma extensão do Word2Vec, permitiu a representação de documentos inteiros como vetores, o que se mostrou útil para tarefas de classificação. A evolução para modelos mais complexos levou ao uso de redes neurais como o *LSTM* (Long Short-Term Memory) (HOCHREITER e SCHMIDHUBER, 1997), que trouxe avanços na modelagem de sequências de texto, considerando dependências temporais em grandes volumes de dados.

Com o avanço das redes neurais profundas, os modelos *Transformer* (VASWANI *et al.*, 2017b), como o *BERT* (DEVLIN *et al.*, 2019) e suas variações, tornaram-se os mais utilizados na detecção de fake news, devido à sua capacidade de entender o contexto de uma maneira mais eficiente e precisa. Esses modelos preveem as palavras com base no contexto completo da sentença, superando abordagens anteriores em precisão e desempenho.

Em termos de algoritmos de aprendizado de máquina, os primeiros a serem amplamente utilizados foram *Naive Bayes* (LEWIS, 1998), *Logistic Regression* (LR) (COX, 1958) e *K-Nearest Neighbors* (KNN) (COVER e HART, 1967), que são simples e eficientes em tarefas de classificação binária. À medida que a complexidade dos dados aumentava, algoritmos como *Decision Trees* (BREIMAN *et al.*, 1984), *Support Vector Machines* (SVM) (CORTES e VAPNIK, 1995) e *Random Forest* (BREIMAN, 2001) ganharam popularidade devido ao seu poder de modelagem não linear. Mais recentemente, técnicas de *ensemble*, como *Stacking*, *Bagging* (BREI-

MAN, 1996; WOLPERT, 1992) e *Boosting* (FREUND e SCHAPIRE, 1997), foram utilizadas para combinar múltiplos modelos, melhorando a precisão e a robustez na detecção de notícias falsas. Esses métodos combinam os pontos fortes de diferentes algoritmos, permitindo uma classificação mais precisa e eficiente.

2.3 Trabalhos Relacionados

A revisão de literatura foi conduzida com o auxílio da ferramenta Parsifal[1], utilizando a base de dados Scopus como fonte principal. A string de busca adotada foi: *TITLE-ABS-KEY (("fake news detection"OR "fake news classification") AND ("machine learning") AND ("NLP"OR "natural language processing") AND ("social network"OR "Twitter")).*

Inicialmente, em 2022, foram considerados apenas artigos publicados nos cinco anos anteriores, o que resultou na recuperação de 43 estudos. Após a leitura dos títulos e resumos, foi realizada uma triagem com base na relevância para os objetivos deste trabalho, levando à seleção de um subconjunto desses artigos.

Para atualizar a revisão e incorporar os avanços mais recentes na área, uma nova busca foi realizada em 2025, utilizando os mesmos critérios e a string de busca original. Essa nova etapa resultou na identificação de mais 85 estudos relevantes, publicados desde a revisão anterior. Esses artigos passaram por uma triagem similar, e os mais pertinentes foram incluídos nesta revisão ampliada.

O objetivo desta revisão sistemática é responder às seguintes questões de pesquisa:

- Quais são as abordagens mais utilizadas em termos de modelagem do problema?
- Quais as técnicas de processamento de linguagem natural mais eficazes para essa tarefa?

PARIKH *et al.* (2019) definiram três hipóteses relacionadas à origem, disseminação e tom das notícias falsas. Eles observaram que: (1) notícias falsas tendem a ser publicadas em sites menos populares, (2) são mais frequentemente disseminadas por usuários não verificados, e (3) são escritas em um tom linguístico específico, embora não seja possível determinar se esse tom é positivo, negativo ou neutro. Esses achados levantam a hipótese de que a análise de sentimentos poderia ser útil na detecção de *fake news*.

Entretanto, BAARIR e DJEFFAL (2021) argumentam que a análise de sentimentos não contribui significativamente para a detecção de *fake news*, pois o fato

[1] `http://parsif.al`

de uma notícia apresentar um sentimento positivo ou negativo não está diretamente relacionado à veracidade do conteúdo. Eles demonstraram que essa distinção não é suficiente para classificar corretamente uma notícia como falsa ou verdadeira.

SILVA *et al.* (2020) destacaram a escassez de conjuntos de dados rotulados em português e propuseram o *Fake.Br Corpus*, com notícias falsas e verdadeiras. Eles realizaram uma análise comparativa com diversos algoritmos, onde *SVM*, *Logistic Regression* e *Random Forest* se destacaram. Além disso, observaram que o uso de *Bag-of-Words* superou modelos mais sofisticados como *Word2Vec* e *FastText*. Eles também apontaram a importância do tamanho do texto, observando que textos truncados podem representar melhor situações reais, já que modelos treinados com textos completos podem ser enganados por notícias falsas mais longas propositalmente.

DE MAGISTRIS *et al.* (2022) propuseram um modelo baseado em *stance detection* para auxiliar na detecção de notícias falsas. O processo inclui categorização da consulta, seleção de documentos relevantes por macro e subcategorias, aplicação de *NER* para filtragem e, finalmente, comparação semântica via *embedding* e similaridade de cosseno para selecionar os documentos mais semelhantes, que então são usados para classificação de *stance*. Apesar da complexidade da abordagem, os autores concluíram que o uso de *stance detection* não apresentou benefícios significativos na detecção de *fake news*.

Os trabalhos de WU e WANG (2021) e MOURATIDIS *et al.* (2021) incorporaram informações derivadas das redes sociais aos modelos de detecção de notícias falsas. Eles utilizaram técnicas como *Graph Embedding*, além de atributos como data de criação da conta, número de seguidores, URLs presentes, entre outros. Esses estudos evidenciaram que tais características sociais são úteis para complementar os modelos baseados apenas em texto.

BIRUNDA e DEVI (2021) propuseram uma abordagem estruturada para a detecção de *fake news*, envolvendo extração de recursos textuais via TF-IDF, análise de URLs e avaliação da credibilidade das fontes. Eles testaram diversos algoritmos de aprendizado de máquina e o modelo com melhor desempenho foi o *Gradient Boosting*, que alcançou uma acurácia de $99,5\%$ em um dataset coletado do *Kaggle* que contém 2050 notícias rotuladas como verdadeiras ou falsas além de metadados como fonte dos artigos de notícias, data de publicação, autor, título, texto, se contém uma imagem, tipo do texto, rótulos, etc. No entanto, o nome específico do dataset utilizado não é mencionado no artigo.

MOURATIDIS *et al.* (2021) também desenvolveram um modelo de *deep learning* com entrada multimodal, combinando *features* linguísticas (como número médio de sílabas por palavra, frases curtas/longas, taxa de advérbios/adjetivos) com atributos sociais (como número de seguidores e retweets). Segundo os autores, a combinação

desses diferentes tipos de entrada foi essencial para alcançar bons resultados, superando abordagens que usam apenas uma dessas fontes de informação.

SETIAWAN *et al.* (2021) construíram um *dataset* voltado à temática de publicidade e avaliaram diferentes modelos com técnicas de *NLP* como TF-IDF, *Bag-of-Words* e n-gramas (de 1 a 4). O estudo indicou que a utilização de n-gramas contribui positivamente para a tarefa de detecção de *fake news*.

PHANG, KELVIN KEAT HUNG AND CHUA, HUI NA AND JASSER, MUHAMMED BASHEER AND WONG, RICHARD T. K. (2023) Abordaram o problema utilizando características dos usuários das redes sociais, com foco nas bases de dados Politifact e GossipCop. O estudo revelou que as notícias provenientes das redes sociais geralmente apresentam uma qualidade inferior, o que torna a tarefa de detecção mais desafiadora. Os autores propuseram uma análise aprofundada das *features* dos usuários, identificando quais delas são mais relevantes para a classificação de notícias falsas. A tabela gerada a partir do estudo destaca a importância relativa de várias características, com destaque para o número de status (11%), o número de seguidores (8,9%), e o recent_favorite, uma métrica proposta pelos autores (8,5%). Além disso, características como recent_retweet e follower_following_ratio também se mostraram significativas, com 7,2% e 7,3%, respectivamente. O estudo sublinha como essas features podem ser utilizadas para aprimorar os modelos de detecção de notícias falsas, aproveitando informações de engajamento e rede social para complementar os dados textuais.

Diversos estudos recentes têm explorado a combinação de múltiplos classificadores para aprimorar a detecção de notícias falsas por meio de técnicas de votação. REZAEI *et al.* (2023) propuseram um modelo de *ensemble* que integra algoritmos como Random Forest, SVM, Decision Tree, LightGBM e XGBoost, utilizando métodos como *Stacking*, *Bagging* e *Boosting*. O modelo foi avaliado na base Politifact e incorporou recursos semânticos e quantitativos do texto, alcançando uma acurácia de 96,24% com o *Stacking*. Seguindo uma linha semelhante, GETHSIA e JULIET (2023) desenvolveram um *Voting Classifier* com Naive Bayes, Regressão Logística, SVM e Decision Tree, empregando técnicas clássicas de pré-processamento e vetorização, como *Bag-of-Words*, *n-grams* e *TF-IDF*, para representar o conteúdo textual. De forma complementar, SANDRILLA e DEVI (2024) também utilizaram uma abordagem baseada em *ensemble*, com foco em notícias políticas, combinando algoritmos como Random Forest, Regressão Logística, J48 e NB-tree. A técnica de votação foi aplicada tanto em *Bagging* quanto em *Boosting*, enquanto o *Stacking* utilizou um meta-classificador. Seu modelo alcançou 97% de acurácia, reforçando a eficácia dos métodos de combinação de classificadores na tarefa de detecção de *fake news*.

ALGHAMDI *et al.* (2023) investigaram o uso de modelos transformer na detecção de fake news relacionadas ao COVID-19, comparando diversas arquiteturas,

incluindo BERTBASE, BERTBASE combinado com camadas de CNN, LSTM e Bi-GRU, além de RoBERTa e o modelo especializado CT-BERT. As técnicas de NLP aplicadas envolvem tokenização, embedding de palavras com BERT e suas variações, e o uso de modelos pré-treinados para transferência de aprendizado, melhorando a análise semântica por meio de camadas CNN, LSTM e BiGRU. O modelo CT-BERT, pré-treinado com um grande corpus de tweets sobre COVID-19, obteve um desempenho superior, alcançando uma precisão de 98,5%, demonstrando a eficácia do fine-tuning em modelos transformer para a tarefa de detecção de fake news.

GUPTA *et al.* (2024) explora a utilização da representação semântica baseada em Abstract Meaning Representation (AMR) para a detecção de fake news. A principal premissa do estudo é que as abordagens convencionais de machine learning e deep learning podem não capturar de maneira eficiente as relações semânticas complexas entre palavras e entidades em textos, o que é crucial para a identificação de desinformação. Para abordar essa limitação, os autores propõem a extração de características semânticas de grafos AMR, que incluem relações semânticas, papéis semânticos e estruturas de eventos, como entrada para modelos tradicionais, como Random Forest e Naive Bayes, bem como redes neurais mais avançadas, como BiLSTM. O estudo foi realizado em três conjuntos de dados: o novo dataset FauxNSA proposto pelos próprios autores, o público Covid19-FND, focado em desinformação sobre o COVID-19, e o KFN, um conjunto público para fake news em geral. Os resultados experimentais mostraram que os modelos utilizando codificação AMR superaram os modelos tradicionais, alcançando uma acurácia de até 93,96% e um F1-score de 91,96%. A pesquisa sugere a fusão de AMR com outras abordagens para fortalecer a detecção de fake news em diferentes contextos e futuros desafios relacionados.

Na tabela a seguir, serão apresentados os principais trabalhos relacionados à detecção de *fake news*, detalhando as abordagens empregadas, os recursos utilizados e os desempenhos obtidos. A Tabela 2.1 resume e compara os resultados entre os diferentes estudos analisados.

2.3.1 Considerações finais sobre a revisão

As abordagens mais comuns para a detecção de fake news envolvem técnicas de aprendizado de máquina clássicas, como SVM e Random Forest, bem como métodos de deep learning, incluindo LSTM e CNN. Embora SILVA *et al.* (2020) tenha observado que abordagens simples, como Bag-of-Words e TF-IDF, apresentaram bons resultados, estudos mais recentes indicam que modelos baseados em transformers, como BERT, têm se destacado. Esses modelos se beneficiam de sua capacidade de capturar informações contextuais de forma mais eficaz, superando métodos mais simples em tarefas complexas de análise de texto, como a detecção de fake news

(ALGHAMDI *et al.*, 2023; WU e WANG, 2021).

Além disso, as técnicas de NLP, como a tokenização e a remoção de stopwords, continuam sendo fundamentais, mas a integração de modelos pré-treinados, como BERT, tem se mostrado particularmente poderosa. Modelos como o BERT, ao serem ajustados para tarefas específicas, como a detecção de fake news, são capazes de entender o contexto e as relações semânticas entre palavras de maneira mais profunda, o que os torna mais eficientes em comparação com abordagens tradicionais (REZAEI *et al.*, 2023).

O uso de modelos de deep learning, especialmente aqueles baseados em transformers, continua sendo a abordagem mais promissora, alcançando altas taxas de precisão e F1-score. A combinação dessas técnicas avançadas com métodos de ensemble learning, como demonstrado por GUPTA *et al.* (2024), mostra que a capacidade dos transformers de lidar com grandes volumes de dados e entender nuances contextuais faz deles uma escolha superior na detecção de fake news.

Tabela 2.1: Comparação dos Trabalhos Relacionados

Trabalho	**Algoritmos**	**Técnicas de pré-processamento**	**Datasets**
PARIKH *et al.* (2019)	Sentiment Analysis	NER, Stance Detection	FakeNewsNet, PolitiFact, BuzzFeed
BAARIR e DJEFFAL (2021)	LSVM	Sentiment Analysis, Remoção de stopwords, Steaming, TF-IDF	Getting Real about Fake News, All the News
SILVA *et al.* (2020)	LR, SVM, RF, DT, NB	BoW, Word2Vec, FastText, Features Linguisticas	FakeBr. Corpus
WU e WANG (2021)	CNN, LR	TF-IDF, N-GRAM, WORD2VEC, Análise de redes sociais	FakeNewsNet
MOURATIDIS *et al.* (2021)	NB, RF, SVM, LR, Deep Learning (SMOTE)	Features Linguisticas	LIAR
BIRUNDA e DEVI (2021)	SVM, KNN, NB, LR, RF, AdaBoosting, DT, GradientBoosting	tokenization, stopwords, lowecasing, Remoção de pontuação, TF-IDF, Credibilidade do autor	Coletado de [2] (Não especificado)
SETIAWAN *et al.* (2021)	SVM	TF-IDF, BOW, N-GRAM	Dataset único coletado pelos pesquisadores
DE MAGISTRIS *et al.* (2022)	LSTM, PCA	doc2vec	COVID-19
REZAEI *et al.* (2023)	Random Forest, SVM, Decision Tree, LightGBM, XGBoost	Análise de semelhanças semânticas, pontuação semântica, número de caracteres, palavras, sentenças, palavras em maiúsculas, média de caracteres por palavra, palavras por sentença	Politifact
GETHSIA e JULIET (2023)	Naive Bayes, Regressão Logística, SVM, Decision Tree	Tokenização, remoção de pontuação, remoção de *stopwords*, Bag-of-Words, n-grams, TF-IDF	Não especificado
SANDRILLA e DEVI (2024)	Random Forest, Regressão Logística, J48, NB-tree	Bagging, Boosting, Stacking, Votação	Redes sociais, conteúdos políticos
ALGHAMDI *et al.* (2023)	BERTBASE, BERT-BASE+CNN, LSTM, BiGRU, RoBERTa, CT-BERT	Tokenização, embedding de palavras com BERT, fine-tuning	Conjunto de dados sobre COVID-19
GUPTA *et al.* (2024)	Random Forest, Naive Bayes, BiLSTM	Extração de características semânticas de AMR, redes neurais avançadas	FauxNSA, Covid19-FND, KFN

Capítulo 3

Large Language Models com Arquitetura Transformer

Modelos de linguagem são algoritmos treinados para prever a próxima palavra em uma sequência de texto, baseando-se no contexto fornecido pelas palavras anteriores. Eles constituem a base para diversas tarefas de processamento de linguagem natural (*NLP*), como tradução automática, sumarização, resposta a perguntas e geração de texto. Tradicionalmente, esses modelos possuíam um número relativamente limitado de parâmetros e eram treinados em conjuntos de dados específicos, o que restringia sua capacidade de generalização e compreensão de contextos complexos.

Com o avanço da capacidade computacional e a disponibilidade de grandes volumes de dados textuais, emergiram os *Large Language Models* (*LLMs*), como GPT-3 (BROWN *et al.*, 2020), PaLM (CHOWDHERY *et al.*, 2022), LLaMA (TOUVRON *et al.*, 2023) e outros. Esses modelos são caracterizados por possuírem bilhões ou até trilhões de parâmetros e por serem treinados em grandes corpora heterogêneos, o que os capacita a lidar com uma ampla gama de tarefas sem necessidade de ajuste fino para cada aplicação específica.

A principal diferença entre os *LLMs* e os modelos de linguagem tradicionais reside na escala e na capacidade emergente dos primeiros. Enquanto modelos menores exigem treinamento supervisionado para desempenhar tarefas específicas, os *LLMs* são capazes de realizar tarefas complexas por meio de *prompting* — ou seja, a simples formulação de uma instrução textual é suficiente para induzir o comportamento desejado.

No processo de geração de texto, os *LLMs* operam de forma autoregressiva. Isso significa que, dada uma sequência inicial de palavras (o prompt), o modelo prediz a próxima palavra com base em probabilidades condicionais aprendidas durante o treinamento. Em seguida, essa nova palavra é incorporada ao contexto, e o processo é repetido até que o modelo atinja um critério de parada, como um número máximo de tokens ou um símbolo especial de término. Esse mecanismo possibilita a produção de

textos coerentes, contextualmente relevantes e muitas vezes indistinguíveis daqueles escritos por humanos.

LLMs também apresentam capacidades emergentes, como raciocínio lógico, compreensão semântica profunda e geração criativa, que não estavam presentes em modelos menores. Tais capacidades são, em grande parte, produto da escala de parâmetros e da diversidade dos dados utilizados no treinamento, o que confere aos *LLMs* uma versatilidade inédita no campo da inteligência artificial aplicada à linguagem.

3.1 Embeddings

Embeddings são representações vetoriais densas de dados, que permitem a transformação de entradas discretas e de alta dimensionalidade (como palavras ou frases) em vetores de números reais com menor dimensionalidade. Esses vetores capturam características semânticas e sintáticas das entradas, sendo úteis como representação interna em modelos de linguagem. A proximidade entre vetores reflete similaridade semântica entre as entradas.

Para ilustrar o conceito de embeddings, considere o seguinte exemplo simples. Suponha que temos um pequeno corpus de frases:

- O gato está no tapete.
- O cachorro está no jardim.
- O gato e o cachorro são amigos.

Utilizando uma técnica de geração de embeddings como o Word2Vec (MIKOLOV *et al.*, 2013a), é possível gerar os seguintes vetores de dimensão 4 para algumas das palavras:

- `gato`: [0.12, 0.43, 0.67, 0.89]
- `cachorro`: [0.15, 0.40, 0.65, 0.88]
- `tapete`: [0.90, 0.15, 0.35, 0.42]
- `jardim`: [0.52, 0.33, 0.33, 0.97]

A similaridade semântica entre os vetores pode ser mensurada com a similaridade de cosseno. Calculando o cosseno do ângulo entre o vetor da palavra `gato` e os demais, temos os seguintes resultados:

- Similaridade entre `gato` e `cachorro`: 0,9994

- Similaridade entre `gato` e `tapete`: 0,6117
- Similaridade entre `gato` e `jardim`: 0,8982

Como mostra a Figura 3.1, após a redução de dimensionalidade usando PCA (Principal Component Analysis), é possível observar graficamente a maior proximidade entre os vetores de `gato` e `cachorro`, o que indica sua maior similaridade semântica em comparação com as demais palavras.

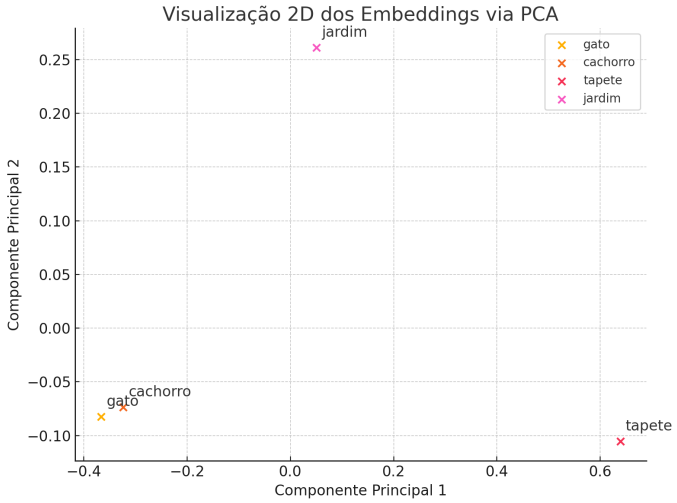


Figura 3.1: Visualização dos embeddings usando PCA. Vetores semanticamente semelhantes estão mais próximos.

3.1.1 Arquitetura Transformer

A arquitetura Transformer, proposta por VASWANI *et al.* (2017a), representa uma mudança de paradigma no campo de modelos de sequência, substituindo os tradicionais mecanismos recorrentes por camadas totalmente baseadas em atenção. Essa abordagem tem se mostrado altamente eficaz em diversas tarefas de *NLP*, como tradução automática, resumo de texto e geração de linguagem.

Camadas encoder-decoder

O Transformer é composto por duas estruturas principais: o *encoder* e o *decoder*. Ambas são formadas por empilhamentos de camadas que combinam mecanismos de atenção com redes neurais feed-forward.

O *encoder* recebe como entrada uma sequência de embeddings posicionais, que combinam informações semânticas e posicionais de cada token da entrada. Cada camada do *encoder* contém duas subcamadas principais: um mecanismo de auto-atenção multi-cabeça e uma rede feed-forward totalmente conectada. Cada uma dessas subcamadas é seguida por uma normalização de camada (*layer normalization*) e um atalho residual (*residual connection*).

O *decoder* também é composto por múltiplas camadas, e cada camada inclui três subcomponentes: uma camada de auto-atenção (com *masking* para impedir a atenção em posições futuras durante o treinamento), uma camada de atenção sobre a saída do *encoder* (também conhecida como atenção cruzada), e uma rede feed-forward. Assim como no *encoder*, cada subcomponente é seguido por conexões residuais e normalização.

Essa arquitetura permite que o modelo processe toda a sequência em paralelo, ao contrário das RNNs, e com uma capacidade superior de modelar dependências de longo alcance.

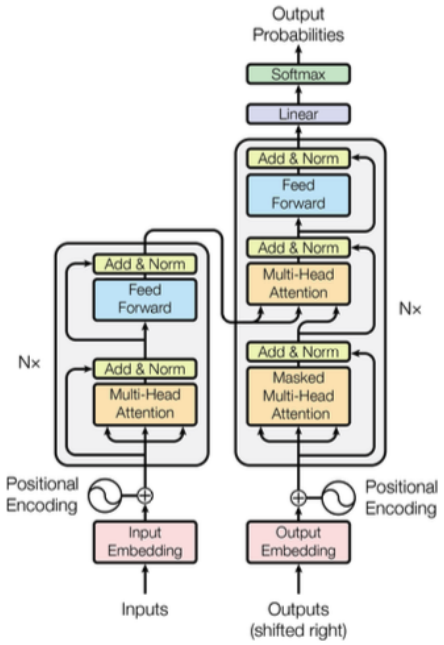


Figura 3.2: Arquitetura completa do modelo Transformer, composta por múltiplas camadas encoder e decoder (VASWANI et al., 2017).

Mecanismo de auto-atenção

O mecanismo de *autoatenção* (*self-attention*) é um componente central da arquitetura Transformer e permite que o modelo avalie a importância relativa de cada palavra em uma sequência com relação às demais. Isso é essencial para capturar relações contextuais entre palavras, independentemente da sua posição.

O funcionamento da autoatenção é baseado em três vetores associados a cada palavra da sequência: **Query** (Q), **Key** (K) e **Value** (V). Esses vetores são obtidos a partir dos embeddings de entrada por meio de multiplicações com matrizes de pesos aprendidas durante o treinamento.

Intuitivamente, podemos entender esse mecanismo como uma busca por informações relevantes. Cada palavra na frase formula uma *consulta* (Query), que é comparada às *chaves* (Keys) de todas as palavras, para avaliar a relevância dos

seus respectivos *conteúdos* (Values). A atenção final é uma média ponderada dos conteúdos com base nessas similaridades.

A equação básica da autoatenção é dada por:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{3.1}$$

Onde Q, K e V são as matrizes das queries, keys e values, e d_k é a dimensionalidade das keys, usada para normalização.

Exemplo:

Considere a frase a seguir: *"a gata branca pulou"*. Suponha que cada palavra seja representada por um vetor de dimensão 3. A matriz de embeddings $X \in \mathbb{R}^{4\times3}$ correspondente à frase seria:

$$X = \begin{bmatrix} \text{a} & \rightarrow & [0.2 & 0.1 & 0.3] \\ \text{gata} & \rightarrow & [0.4 & 0.6 & 0.5] \\ \text{branca} & \rightarrow & [0.6 & 0.5 & 0.4] \\ \text{pulou} & \rightarrow & [0.3 & 0.8 & 0.2] \end{bmatrix} \tag{3.2}$$

Cada linha da matriz representa o vetor de embedding de uma palavra da frase na ordem em que aparecem.

A partir desta matriz, são calculadas as matrizes $Q = XW^Q$, $K = XW^K$ e $V = XW^V$ utilizando pesos aprendidos $W^Q, W^K, W^V \in \mathbb{R}^{3\times3}$. As pontuações de atenção são obtidas pelo produto escalar entre Q e $K^\top$, seguido de uma normalização com a função *softmax*.

Tabela 3.1: Matriz de atenção palavra a palavra

	a	**gata**	**branca**	**pulou**
a	0.1	0.3	0.3	0.3
gata	0.2	0.2	0.5	0.1
branca	0.1	0.2	0.6	0.1
pulou	0.05	0.15	0.3	0.5

Cada linha representa a distribuição de atenção de uma palavra sobre todas as outras da frase. Por exemplo, a linha da palavra *gata* mostra que ela dá maior peso à palavra *branca* (0.5), o que é coerente com o fato de que juntas formam uma expressão semântica coesa: *gata branca*. Já a palavra *pulou* dá mais atenção a si mesma, mas também considera parcialmente a palavra anterior *branca*, indicando dependência contextual. Esse mecanismo permite que o modelo construa representações enriquecidas, considerando como cada palavra interage com o restante da

frase.

Por exemplo, suponha que a palavra “gata” tenha uma alta atenção em relação à palavra “branca”. Nesse caso, a nova representação vetorial de “gata” resultante da autoatenção será fortemente influenciada pelo vetor de “branca”, incorporando informações semânticas associadas à cor ou descrição do substantivo. De forma semelhante, a palavra “pulou” pode receber atenção de “gata”, indicando a relação sujeito-verbo. Esse processo ocorre para cada palavra na sequência, produzindo representações contextualizadas que refletem as relações internas entre os termos da sentença.

Na prática, diferentes relações linguísticas podem ser relevantes ao mesmo tempo; como relações de concordância, dependência sintática ou similaridade semântica. Para capturar essas múltiplas perspectivas de forma eficiente, o *Transformer* adota o mecanismo de *multi-headed attention*. Em vez de aplicar uma única função de atenção, o modelo utiliza diversas “cabeças” (*heads*) de atenção em paralelo, onde cada uma aprende a focar em um aspecto distinto da estrutura da frase.

No caso da frase “a gata branca pulou”, uma cabeça pode aprender a associar “gata” com “branca” (descrição), enquanto outra pode conectar “gata” com “pulou” (relação sujeito-verbo), e ainda outra pode focar no artigo “a” para marcar o início da sentença. Cada cabeça de atenção possui seus próprios parâmetros e projeta a entrada em subespaços distintos. Os resultados dessas atenções paralelas são então concatenados e projetados novamente ao espaço original por meio de uma camada linear, enriquecendo a representação contextual de cada palavra como na figura 3.3. Isso permite que o modelo aprenda e integre informações de diferentes perspectivas simultaneamente, aumentando a riqueza das representações contextuais.

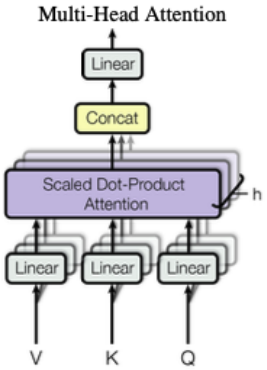


Figura 3.3: Diagrama do mecanismo de *multi-headed attention*. Cada cabeça de atenção foca em diferentes aspectos da sentença, permitindo uma representação rica e multi-dimensional das palavras.

3.2 Engenharia de Prompt

A engenharia de *prompt* é o processo de criação, refinamento e otimização de instruções fornecidas a modelos de linguagem, com o objetivo de extrair respostas mais precisas, relevantes e coerentes. Essa prática tem se tornado cada vez mais essencial à medida que os modelos evoluem e são aplicados em contextos mais diversos e exigentes. De fato, a *prompt engineering* tem emergido como um pilar central nas interações entre humanos e modelos de linguagem (Mesko, 2023; White et al., 2023, conforme citado em (LAZOVSKY *et al.*, 2025)).

Segundo Ekin (EKIN, 2023), a engenharia de *prompt* é uma habilidade crítica para explorar o potencial completo de *LLMs*. *Prompts* bem elaborados são capazes de guiar o modelo para compreender com maior clareza a intenção do usuário, resultando em interações mais eficazes e úteis.

Os *prompts* funcionam como o principal meio de comunicação entre o usuário e o modelo. A forma como uma instrução é formulada pode influenciar diretamente na resposta gerada. Por exemplo, *prompts* mais específicos tendem a gerar respostas mais focadas, como demonstrado nos exemplos abaixo:

- **Prompt genérico:** *"Escreva um poema."*
- **Resposta:**

  No silêncio da noite,
  A lua solitária levanta voo,
  Seus raios prateados acariciam a terra,
  Um toque suave de uma mão celestial.

- **Prompt específico:** *"Escreva um haicai sobre a mudança das estações."*
- **Resposta:**

  Folhas caem no outono,
  Sopro gelado do inverno,
  brotos da primavera,
  Sol do verão brilha.

Esse exemplo ilustra a importância da clareza e especificidade na construção dos prompts, permitindo ao modelo gerar saídas mais alinhadas com os objetivos do usuário.

3.2.1 Fatores que influenciam a formulação de prompts

A escolha do prompt ideal depende de múltiplos fatores:

- **Intenção do usuário:** compreender o objetivo da tarefa (ex: responder, gerar conteúdo ou resolver um problema).
- **Compreensão do modelo:** ter ciência das capacidades e limitações do ChatGPT para evitar falhas comuns.
- **Domínio da aplicação:** utilizar vocabulário e contexto específico para guiar o modelo.
- **Clareza e especificidade:** evitar ambiguidade com instruções claras e bem definidas.
- **Restrições desejadas:** como tamanho da resposta, formato ou estrutura.

Esses elementos contribuem para uma comunicação mais efetiva com o modelo e aumentam a precisão das respostas.

3.2.2 Técnicas para melhorar prompts

Diferentes abordagens podem ser usadas para aumentar a efetividade dos prompts:

Instruções claras e específicas

Prompts específicos tendem a gerar respostas mais relevantes. Por exemplo, ao invés de pedir apenas "explique o sistema solar", especificar a ordem dos planetas ou o tipo de informação desejada gera respostas mais focadas.

Uso de restrições explícitas

Definir limitações, como número de frases ou formato, ajuda o modelo a estruturar melhor a resposta, especialmente quando se deseja concisão, estrutura ou aderência a requisitos formais.

Inserção de contexto e exemplos

Adicionar contexto ou exemplos dentro do prompt fornece referências para o modelo, o que é útil especialmente em domínios técnicos ou quando se busca respostas ilustrativas.

Estímulo ao raciocínio analítico

Diferenciar perguntas simples (fato direto) de perguntas que exigem análise e julgamento pode alterar a forma como o modelo responde. Perguntas mais analíticas incentivam respostas mais reflexivas e estruturadas.

Controle da verbosidade

É possível controlar o nível de detalhamento das respostas por meio de instruções explícitas no prompt, como pedir um resumo ou uma explicação detalhada. Isso ajuda a alinhar a resposta com a necessidade do usuário.

3.3 Técnicas de Engenharia de Prompts

O uso de modelos de linguagem de grande porte (LLMs) exige a aplicação de diferentes técnicas de engenharia de *prompts* para maximizar seu desempenho em tarefas complexas. Abaixo estão algumas das principais abordagens descritas na literatura recente.

1. **Zero-shot Prompting** (RADFORD *et al.*, 2019): permite que o modelo realize tarefas inéditas sem fornecer exemplos prévios. A entrada consiste apenas em uma instrução bem elaborada, confiando no conhecimento prévio do modelo para gerar a resposta. É útil em cenários com poucos dados ou tarefas genéricas.

2. **Few-shot Prompting** (BROWN *et al.*, 2020): oferece ao modelo alguns exemplos de entrada e saída dentro do *prompt*, permitindo que ele infira o padrão desejado. Essa técnica melhora o desempenho em tarefas complexas, mas depende da escolha e formatação adequadas dos exemplos, além de consumir mais tokens.

3. **Chain-of-Thought (CoT) Prompting** (WEI *et al.*, 2022): introduz cadeias de raciocínio passo a passo dentro do *prompt*, incentivando o modelo a seguir um processo lógico antes de apresentar a resposta final. Essa técnica mostrou ganhos expressivos em tarefas que exigem raciocínio matemático ou de senso comum.

4. **Auto-CoT Prompting** (ZHANG *et al.*, 2022): automatiza a geração de cadeias de raciocínio, eliminando a necessidade de exemplos manuais. Utiliza o padrão "vamos pensar passo a passo" e gera múltiplas cadeias diversas, melhorando a robustez e a performance em tarefas simbólicas e aritméticas.

5. **Self-Consistency** (WANG *et al.*, 2022): estratégia de decodificação que complementa o CoT. Em vez de escolher apenas uma cadeia de raciocínio, amostras múltiplas são geradas e a resposta mais consistente entre elas é escolhida. Isso resulta em melhorias significativas de acurácia em diversos benchmarks.

6. **Retrieval-Augmented Generation (RAG)** (LEWIS *et al.*, 2020): técnica que incorpora recuperação de informações externas ao processo de *prompting*. A entrada do usuário é transformada em uma consulta, que busca informações relevantes em uma base de conhecimento. Esses trechos recuperados são adicionados ao *prompt*, enriquecendo o contexto e permitindo respostas mais precisas e atualizadas. RAG se mostrou eficaz em tarefas que exigem conhecimento factual, superando abordagens baseadas apenas em modelos seq2seq.

Além das técnicas mencionadas, em uma pesquisa realizada em fevereiro de 2024, mais de 29 técnicas de *prompts* distintas foram categorizadas, conforme a Figura 3.4, extraída do estudo sistemático de (SAHOO *et al.*, 2024). Essa classificação inclui variações de *prompting* voltadas à generalização, raciocínio lógico e redução de alucinações, evidenciando a diversidade e sofisticação das estratégias aplicadas em LLMs.

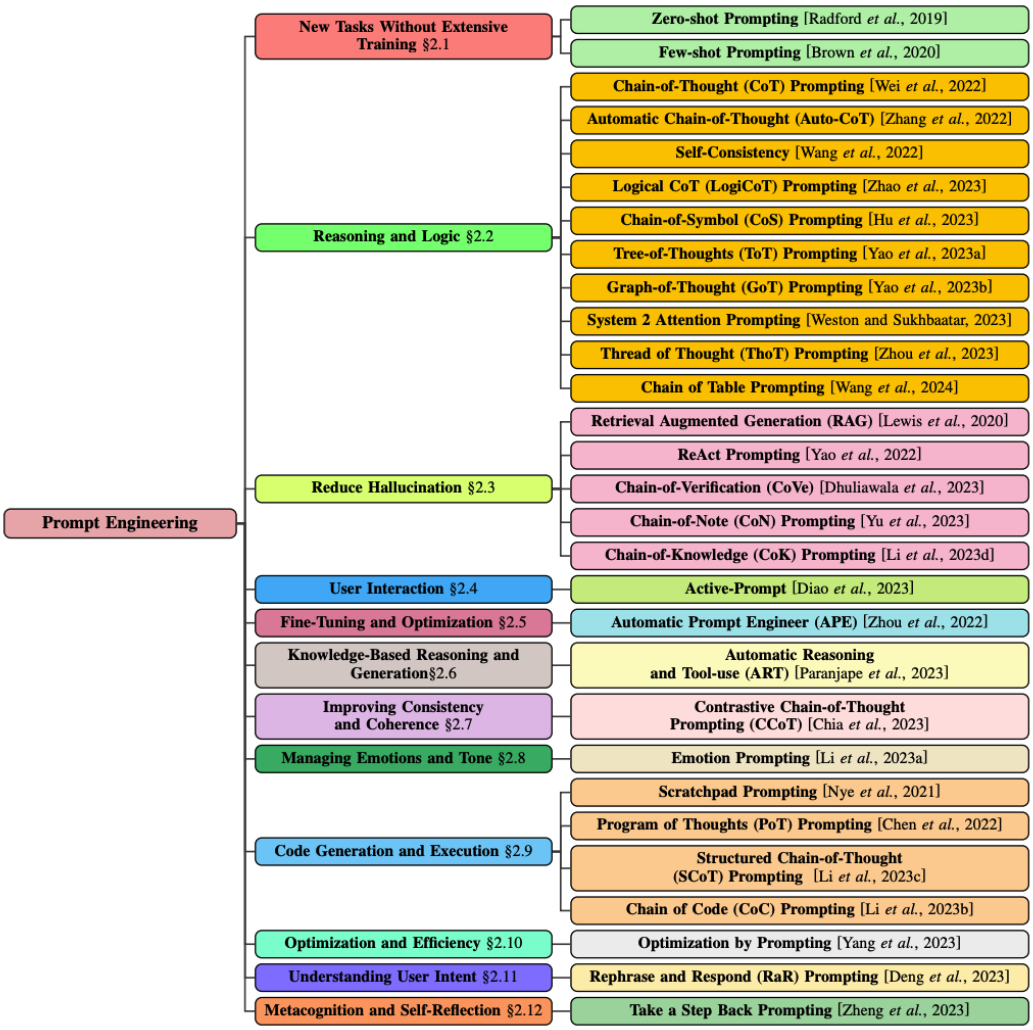


Figura 3.4: Categorias e técnicas de engenharia de *prompts* para LLMs. Adaptado de (SAHOO *et al.*, 2024).

3.3.1 Zero-Shot Prompting

A técnica de *zero-shot prompting* (RADFORD *et al.*, 2019) consiste em instruir o modelo de linguagem sem fornecer exemplos explícitos da tarefa. O prompt apresenta apenas uma descrição clara da ação desejada, permitindo que o modelo utilize seu conhecimento prévio para gerar a resposta apropriada. Segue abaixo um exemplo da aplicação da técnica *zero-shot* em um problema de sumarização.

```
Prompt:

    Resuma a seguinte frase: "O projeto foi
    desenvolvido por uma equipe interdisciplinar ao
    longo de seis meses, com foco em acessibilidade e
    sustentabilidade."
```

Resposta esperada do modelo:

```
Resumo: "Projeto de seis meses com foco em
acessibilidade e sustentabilidade, feito por
equipe interdisciplinar."
```

Esse método é útil para tarefas simples ou quando se deseja uma abordagem direta, sem necessidade de contextualização adicional por meio de exemplos.

3.3.2 Few-Shot Prompting

Na técnica de *few-shot prompting* (BROWN *et al.*, 2020), o modelo recebe alguns exemplos de entrada e saída diretamente no prompt. Esses exemplos servem como guia, ajudando o modelo a identificar padrões e replicá-los na próxima tarefa solicitada. É especialmente eficaz para tarefas que requerem interpretação de contexto ou aplicação de regras específicas. Segue abaixo um exemplo da aplicação da técnica *zero-shot* em um problema de classificação.

Prompt:

```
Classifique o sentimento das frases abaixo:
Frase: "Estou muito feliz com o
resultado."Sentimento: Positivo
Frase: "O atendimento foi lento e
ineficiente."Sentimento: Negativo
Frase: "O produto chegou no prazo, mas veio com
defeito."Sentimento:
```

Resposta esperada do modelo:

```
Negativo
```

Ao incluir exemplos explícitos, o *few-shot prompting* permite instruções mais detalhadas, promovendo maior precisão em tarefas mais complexas ou ambíguas.

3.3.3 Prompt Priming

A técnica de *prompt priming* (MURRAY, 2024) envolve a inclusão de contexto adicional no prompt, como definições ou explicações, para orientar o modelo e melhorar sua compreensão sobre a tarefa. Essa técnica pode ser usada para fornecer ao modelo informações prévias sobre o que é relevante para a tarefa, o que, por sua vez, ajuda a aumentar a precisão e a consistência das respostas geradas.

Prompt:

```
Fake news são informações falsas ou enganosas
divulgadas intencionalmente para enganar. Sua
tarefa é classificar as frases abaixo como Fake ou
Verdadeira.
Frase: "Vacina contra Covid-19 altera DNA humano
permanentemente."Classificação:
```

Resposta esperada do modelo:

```
Fake
```

Ao incluir uma definição no início do prompt, o *prompt priming* assegura que o modelo tenha uma compreensão adequada do conceito-chave, como a definição de fake news, antes de realizar a tarefa. Isso resulta em uma maior precisão nas respostas do modelo.

3.3.4 Output Constraint

A técnica de *output constraint* (LIU *et al.*, 2024) refere-se à imposição de restrições explícitas sobre o tipo e formato da saída do modelo. Isso pode ser feito por meio de instruções que limitam a resposta do modelo a um tipo específico, como uma classificação binária, evitando assim a geração de texto irrelevante ou excessivo.

Prompt:

```
Classifique a frase abaixo como Fake ou
Verdadeira.
Frase: "Vacina contra Covid-19 altera DNA humano
permanentemente."
Retorne apenas a classificação.
```

Resposta esperada do modelo:

```
Fake
```

Capítulo 4

Metodologia

Esta dissertação adota uma metodologia estruturada que será detalhada ao longo deste capítulo. Inicialmente, apresenta-se o processo de trabalho por meio de um modelo BPMN que ilustra as etapas principais da pesquisa. Em seguida, define-se o cenário do estudo e a tarefa específica a ser realizada, para então levantar os recursos necessários e apresentar as bases de dados utilizadas na avaliação. O capítulo também aborda a elaboração dos prompts para interação com os modelos de linguagem, a implementação do sistema incluindo a especificação do algoritmo e as métricas de avaliação, e, por fim, detalha a seleção dos modelos de linguagem de grande porte (LLMs) e a aplicação da técnica de fine-tuning para aprimoramento dos resultados. As seções subsequentes descrevem, assim, de forma organizada e detalhada, cada componente da metodologia empregada nesta pesquisa.

4.1 Processo de Trabalho

Na figura 4.1, podemos ver um diagrama BPMN (ROSING *et al.*, 2015) da metodologia proposta. A metodologia foi inspirada no trabalho de JÚNIOR (2024), com algumas adaptações em algumas etapas, como, por exemplo, no fine-tuning que não existia e na definição da base de dados, onde escolhemos mais de uma base para melhor atender aos requisitos da pesquisa.

O processo inicia-se com a criação de um diagrama BPMN da própria metodologia, como forma de estruturar visualmente o fluxo de trabalho. Em seguida, realiza-se a definição do cenário, que inclui a análise do contexto, a identificação das partes interessadas, a definição dos objetivos de negócio e das métricas de sucesso.

Com o cenário bem delimitado, passa-se para a determinação da tarefa, ou seja, a especificação clara do problema que será abordado. Após isso, realiza-se o levantamento dos recursos disponíveis, como equipe, infraestrutura e ferramentas.

A próxima etapa é a definição da base de dados, que inclui a busca por bases existentes, avaliação quanto à qualidade e relevância, verificação da viabilidade de

uso (considerando aspectos técnicos e legais), e, se necessário, adaptação dos dados ao contexto da tarefa. Diferente da metodologia original, optamos por utilizar mais de uma base de dados para garantir uma cobertura mais ampla e robusta para a tarefa de detecção de fake news.

Uma vez definida a base de dados, o próximo passo é elaborar os prompts, que servirão como instruções ou estímulos para os modelos de linguagem. Em seguida, passa-se à definição da implementação, onde se estabelece como o sistema será construído ou testado, incluindo arquiteturas e fluxos de execução.

Com a implementação delineada, são selecionados os LLMs candidatos, levando em consideração fatores como desempenho, custo e compatibilidade com a tarefa. Finalmente, realiza-se o fine-tuning, que inclui a avaliação do orçamento disponível, a seleção das bases de dados para o ajuste fino, a escolha dos modelos que serão ajustados e a análise da relação entre custo e ganho de desempenho para decidir a viabilidade do processo.

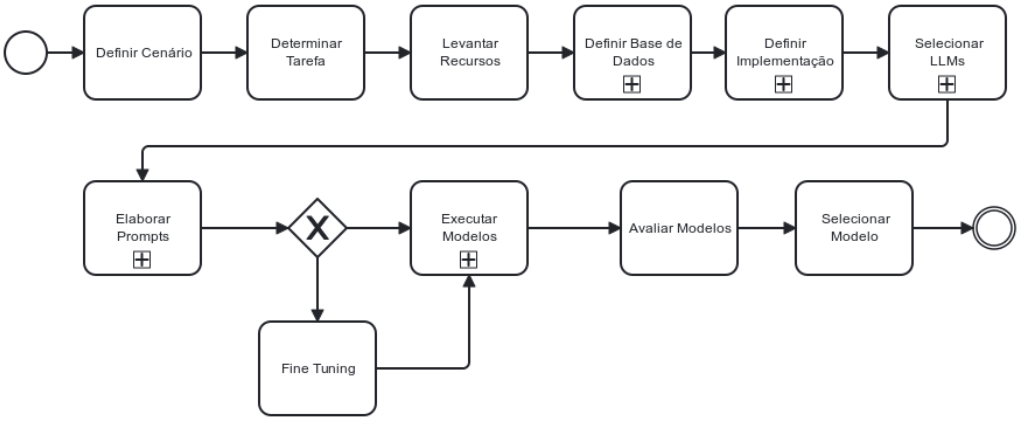


Figura 4.1: Diagrama BPMN do processo. O símbolo de + significa um subprocesso.

Nas próximas seções, detalharemos a metodologia aplicada em cada etapa do experimento, descrevendo suas respectivas entradas, saídas e decisões envolvidas. Considerando os objetivos deste trabalho, daremos ênfase especial às etapas destacadas no diagrama, por serem mais relevantes para os resultados esperados e para a avaliação da proposta.

4.2 Definição do Cenário

Nesta etapa, busca-se estabelecer o cenário do experimento, fornecendo o contexto necessário para compreender o problema abordado e a aplicabilidade das soluções propostas. A definição clara do cenário contribui para alinhar os objetivos técnicos e científicos às necessidades associadas ao combate à desinformação.

Seguindo a estrutura metodológica adaptada de (JÚNIOR, 2024), a construção do cenário envolve a identificação dos seguintes elementos principais:

- **Contexto:** descrição do ambiente no qual a solução será aplicada, considerando o ecossistema de circulação de informações online, redes sociais, portais de notícias e outros meios digitais, onde a disseminação de notícias falsas representa um problema relevante.
- **Problema:** propagação de fake news e a dificuldade em identificar automaticamente conteúdos falsos, o que pode impactar negativamente a sociedade, influenciando opiniões, decisões e até processos democráticos.
- **Objetivos:** desenvolver e avaliar um modelo capaz de detectar notícias falsas com alta efetividade, contribuindo para a mitigação da desinformação. Este objetivo é de natureza científica e social, buscando avanços na área de processamento de linguagem natural e aprendizado de máquina aplicados à detecção de fake news.
- **Métricas de Avaliação:** serão utilizados critérios amplamente aceitos na literatura de aprendizado de máquina e NLP, como acurácia, precisão, recall e F1-score, para mensurar o desempenho dos modelos propostos.

Diferentemente de cenários empresariais, este trabalho não adota abordagens como a metodologia IRACIS, uma vez que não se trata de uma aplicação orientada a negócios. O foco aqui está no desenvolvimento científico e tecnológico, visando oferecer ferramentas que possam ser aplicadas por pesquisadores, plataformas digitais ou organizações que atuam no combate à desinformação.

4.3 Definição da Tarefa

Nesta etapa, busca-se especificar claramente a tarefa a ser executada e os subobjetivos da pesquisa no contexto da detecção de fake news. A definição precisa da tarefa deve ser realizada com base na compreensão do cenário previamente estabelecido.

Deve-se identificar os subproblemas e as tarefas específicas relacionadas à detecção de fake news, com cada problema claramente definido em termos de saída e tipo de tarefa, conforme ilustrado na tabela 5.2. Isso permitirá que a solução seja composta por uma ou múltiplas subtarefas, cada uma destinada a abordar uma parte do problema de maneira eficaz e alinhada aos objetivos da pesquisa.

Tarefa	**Saída**
Classificação de Fake News	Fake, Não Fake
Análise de Sentimentos	Positivo, Negativo, Neutro

Tabela 4.1: Saídas em tarefas específicas de IA para detecção de fake news

4.4 Levantamento de Recursos

Nesta etapa, busca-se identificar e assegurar todos os recursos necessários para a pesquisa, garantindo que se disponha de todos os elementos essenciais para sua execução. Primeiramente, avaliam-se os recursos computacionais, considerando a necessidade de servidores, unidades de processamento gráfico (Graphics Processing Unit - GPU) especializadas e infraestrutura de armazenamento adequada para suportar tanto o treinamento quanto a inferência dos modelos. A escolha de provedores de nuvem, como AWS[1], Google Cloud[2] ou Azure[3], pode ser considerada para garantir escalabilidade e flexibilidade.

Além disso, identificam-se os recursos humanos necessários, como cientistas de dados, desenvolvedores e analistas de negócios, entre outros. Define-se o papel de cada membro da equipe, assegurando que todas as competências essenciais estejam cobertas. Custos como salários, hardware e licenciamento de software também devem ser considerados nessa etapa.

4.5 Definição da Base de Dados

Nesta etapa busca-se definir uma ou mais bases de teste para uso na solução. Neste trabalho utilizamos o mesmo processo sistemático e iterativo utilizado no trabalho de (JÚNIOR, 2024), ilustrado pelo diagrama 4.4 que visa assegurar a escolha de bases de dados que atendam aos requisitos do experimento.

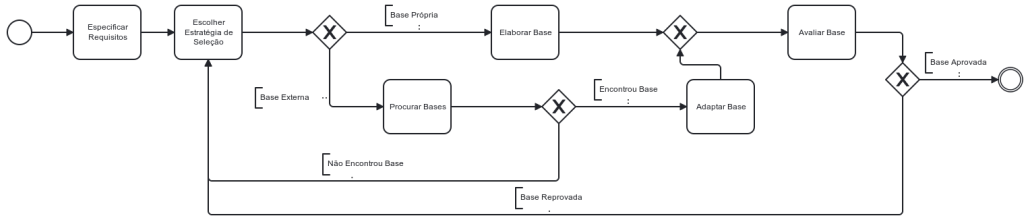


Figura 4.2: Diagrama de Processo de definição de base de dados

O primeiro passo nesta etapa é especificar os requisitos dos dados, definindo claramente os critérios e necessidades do experimento. Esses requisitos incluem características como o formato dos dados, o volume necessário, a qualidade esperada e a relevância para a tarefa que o modelo de linguagem (LLM) deverá desempenhar. Uma vez definidos os requisitos, passa-se à escolha da estratégia de obtenção dos dados. Essa estratégia pode seguir dois caminhos principais: a criação de uma base de dados própria ou a busca por bases externas.

[1] `https://aws.amazon.com/`
[2] `https://cloud.google.com/`
[3] `https://azure.microsoft.com/`

Se optar pela criação de uma base própria, os dados são coletados, organizados e rotulados de acordo com os critérios estabelecidos. Para conjuntos de dados não rotulados, o uso de *crowdsourcing* para rotulagem é uma alternativa viável (WELD *et al.*, 2021; ZHANG *et al.*, 2018). Além disso, técnicas de data *augmentation* podem ser aplicadas para aumentar a diversidade do conjunto de treinamento, melhorando o desempenho dos modelos sem necessidade de coletar mais dados (DING *et al.*, 2024). Essas técnicas permitem gerar novas amostras de forma sintética, reduzindo o custo e o esforço envolvidos na coleta.

Por outro lado, se a estratégia adotada for a busca por bases externas, plataformas como Kaggle[4], UC Irvine Machine Learning Repository[5] e Google Dataset Search Engine[6] podem ser utilizadas para localizar bases que atendam aos critérios definidos. Ao selecionar uma base externa, é necessário adaptá-la ao contexto do experimento, o que pode incluir transformações de formato, normalização dos dados e adição de anotações específicas.

A tabela 4.2 apresenta exemplos de entradas e rótulos de algumas tarefas de uso final.

Tarefa	**Entrada**	**Rótulo**
Classificação de Fake News	Vacina altera DNA humano permanentemente	Fake
Análise de Sentimentos	AGI está cada vez mais próximo	Positivo, Negativo, Neutro
Tradução de Textos	Thank you!	Obrigado!

Tabela 4.2: Exemplos de Entradas e Rótulos de algumas tarefas de uso final

4.6 Definição da Implementação

Nesta etapa, visa-se estabelecer a execução da tarefa proposta. Esta fase compreende duas subetapas: a especificação do algoritmo e a definição das métricas de avaliação. Durante esse processo, são desenvolvidos os programas e *scripts* necessários, garantindo uma coleta de dados adequada para posterior análise.

4.6.1 Especificação do Algoritmo

Nesta subetapa, determina-se a base tecnológica essencial para a implementação dos modelos, abrangendo a escolha de hardware, software e recursos computaci-

[4] `https://www.kaggle.com/datasets`
[5] `https://archive.ics.uci.edu/`
[6] `https://datasetsearch.research.google.com/`

onais. Inicialmente, especifica-se a infraestrutura, que pode incluir servidores de alto desempenho ou provedores de nuvem. Adicionalmente, configuram-se os ambientes de desenvolvimento e execução, envolvendo frameworks e bibliotecas como TensorFlow[7], PyTorch[8], Hugging Face Transformers[9] e SageMaker Python SDK[10].

Nessa fase, também são estabelecidos os parâmetros dos modelos, como a temperatura, que deve ser ajustada de acordo com a natureza da tarefa. Em problemas de classificação, por exemplo, busca-se maior determinismo nos resultados, o que geralmente implica em uma temperatura próxima de zero.

4.6.2 Definição das Métricas de Avaliação

Aqui, estabelecem-se critérios claros para mensurar o desempenho dos modelos de linguagem natural implementados. As métricas de avaliação são fundamentais para garantir que os modelos atendam aos objetivos estabelecidos. Em tarefas de classificação, por exemplo, são frequentemente utilizadas métricas como precisão, recall e F1-score(KENT *et al.*, 1955; VAN RIJSBERGEN, 1979), que proporcionam uma análise quantitativa da eficácia do modelo, identificando pontos fortes e áreas que requerem melhorias.

4.7 Seleção de LLMs

A seleção dos modelos de linguagem (LLMs) a serem testados é um processo crítico que envolve múltiplos critérios. Inicialmente, são avaliados o desempenho histórico dos modelos, a compatibilidade com a tarefa proposta, o custo de uso e a disponibilidade. No entanto, para experimentos envolvendo modelos de grande porte, é essencial considerar os recursos de hardware necessários para sua execução.

Modelos maiores, como os de bilhões de parâmetros, geralmente exigem mais memória de GPU e capacidade de processamento, sendo comuns os casos em que apenas uma GPU dedicada não é suficiente. Nesses cenários, são necessários sistemas com múltiplas GPUs, o que eleva significativamente os custos operacionais. O custo de aquisição de hardware adequado, incluindo GPUs de alta performance, além do consumo energético associado, deve ser avaliado.

Além disso, o tamanho da base de dados e a quantidade de prompts a serem testados impactam diretamente o tempo de execução dos experimentos. Grandes volumes de dados podem inviabilizar a realização de testes completos em máquinas locais, tornando necessário o uso de ambientes na nuvem, como Amazon SageMaker,

[7] https://www.tensorflow.org/
[8] https://pytorch.org/
[9] https://github.com/huggingface/transformers
[10] https://sagemaker.readthedocs.io/

Google Cloud Platform ou Microsoft Azure. Cada uma dessas soluções apresenta uma estrutura de custos que deve ser considerada, variando conforme a configuração de hardware, tempo de uso e outros fatores específicos.

Dessa forma, a escolha dos modelos e o planejamento dos experimentos envolvem não apenas aspectos técnicos, mas também uma análise cuidadosa de viabilidade financeira e logística, garantindo que os recursos disponíveis sejam adequados para o alcance dos objetivos estabelecidos.

4.8 Elaboração de Prompts

Nesta etapa busca-se elaborar os *prompts* para a tarefa definida, conforme figura 4.3. Utiliza-se a Engenharia de Prompt para elaborar *prompts* que maximizem o potencial dos modelos(SHI *et al.*, 2023).

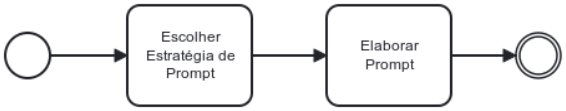


Figura 4.3: Diagrama de Processo de definição de *prompts*

Para a elaboração dos prompts, é essencial analisar as técnicas e estruturas definidas na literatura, como o Zero-Shot, o Few-Shot (BROWN *et al.*, 2020), e o Chain of Thought (CoT) (WEI *et al.*, 2022), Prompt Priming (MURRAY, 2024), Output Constraint (LIU *et al.*, 2024) dentre outros.

Com a seleção das técnicas a serem testadas, formulam-se e avaliam-se diferentes tipos de prompts, ajustando-os conforme o feedback e os resultados obtidos. O objetivo é otimizar o desempenho do modelo, garantindo que ele atenda aos requisitos da tarefa.

Devido à variedade de estratégias de prompt disponíveis, a experimentação pode ser realizada de forma interativa, utilizando trechos de texto da base e explorando versões de chat de LLMs, como o ChatGPT da OpenAI[11] ou o Claude[12]. Alternativamente, é possível usar scripts locais com serviços como o Amazon SageMaker[13] ou Amazon Bedrock[14], especialmente para modelos que não possuem interfaces interativas.

É comum que modelos de linguagem de código aberto sejam lançados em pares, apresentando uma versão base e uma versão instruída (*instruct*), como Llama-3-8b e Llama-3-8b-Instruct, ou Gemma-2b e Gemma-2b-Instruct. Recomenda-se utilizar as versões *instruct* durante a experimentação, pois esses modelos são treinados

[11] `https://chat.openai.com`
[12] `https://claude.ai/`
[13] `https://aws.amazon.com/pm/sagemaker/`
[14] `https://aws.amazon.com/bedrock/`

para seguir instruções, apresentando desempenho superior em tarefas específicas (OUYANG *et al.*, 2022). Para mais orientações, consulte o Prompting Guide[15] e a documentação oficial da OpenAI[16].

4.9 Fine-Tuning

Nessa etapa, serão selecionados alguns modelos de linguagem para a realização do *fine-tuning*, com o objetivo de avaliar se é possível obter uma performance superior na tarefa proposta. O processo de *fine-tuning* envolve uma série de decisões críticas, começando pela avaliação do orçamento disponível. Inicialmente, é necessário verificar os recursos financeiros e computacionais que podem ser alocados para essa tarefa. Isso inclui considerar os custos diretos relacionados ao uso de infraestrutura de hardware, como servidores com GPUs de alta performance, além de possíveis gastos com serviços em nuvem, onde o processamento pode ser realizado de forma mais eficiente. A definição do orçamento é fundamental para garantir que o processo seja viável e que os recursos não sejam excedidos.

Em seguida, ocorre a escolha das bases de dados que serão utilizadas para o *fine-tuning*. É essencial selecionar conjuntos de dados que sejam representativos da tarefa e que contenham informações suficientes para permitir que o modelo aprenda os padrões desejados. A qualidade dos dados impacta diretamente o desempenho final do modelo. Portanto, é necessário realizar uma curadoria cuidadosa, garantindo que os dados estejam limpos, balanceados e anotados corretamente, quando necessário. Dependendo da tarefa, podem ser utilizados dados textuais específicos, como artigos, conversas ou documentos técnicos, sempre alinhados ao domínio do problema.

Após a definição das bases de dados, é realizada a seleção dos modelos que passarão pelo processo de *fine-tuning*. Essa escolha é guiada por uma análise inicial de desempenho, levando em consideração os modelos que apresentam maior potencial para aprimorar a tarefa proposta. Modelos maiores podem oferecer desempenho superior, mas também exigem mais recursos computacionais. Além disso, é importante considerar que diferentes modelos processam os dados de maneira distinta. Portanto, durante o *fine-tuning*, é necessário garantir que os dados sejam adaptados ao formato de entrada esperado por cada modelo, evitando erros de processamento e maximizando o aprendizado.

Finalmente, é realizada uma análise de viabilidade, onde são comparados os custos do *fine-tuning* com os possíveis ganhos de desempenho que podem ser alcançados. Essa análise é essencial para garantir que o esforço e os recursos investidos sejam justificados pelos resultados obtidos. Modelos que não apresentam uma me-

[15] `https://www.promptingguide.ai/`
[16] `https://platform.openai.com/docs/guides/prompt-engineering`

lhoria significativa em relação ao seu custo de ajuste podem ser descartados, permitindo que os recursos sejam concentrados em opções mais promissoras. A análise de viabilidade também considera aspectos práticos, como o tempo necessário para o *fine-tuning*, o impacto do aumento do tamanho do modelo no tempo de inferência e os requisitos de armazenamento. Dessa forma, o processo é conduzido de maneira racional e orientada a resultados, garantindo que os modelos selecionados ofereçam o melhor equilíbrio entre desempenho e eficiência.

4.10 Executar Modelos

Esta etapa visa executar a implementação definida nos modelos selecionados conforme visto no diagrama 4.1. Como cada modelo pode possuir um formato particular para receber instruções, pode ser necessário adaptar os prompts para cada modelo a ser executado.

4.11 Avaliar Modelos

Nessa etapa, executa-se uma análise detalhada de custo e desempenho dos modelos selecionados, facilitando a tomada de decisão sobre qual modelo será implementado na solução final, conforme figura

Na avaliação de custo, estima-se todos os custos associados à implementação e execução dos modelos. Isso inclui custos computacionais, como consumo de energia, tempo de processamento e utilização de servidores e GPUs, além de despesas com armazenamento de dados ou com plataformas de computação em nuvem, ou tokens consumidos, no caso de APIs como OpenAI ou Amazon Bedrock. Paralelamente, realiza-se a avaliação de desempenho dos modelos utilizando métricas como precisão, recall, F1-score que são frequentemente utilizadas para avaliar a eficácia de modelos.

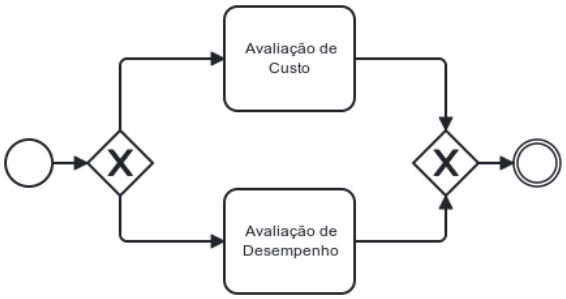


Figura 4.4: Diagrama de Processo de avaliação de modelos

4.12 Selecionar Modelo

Após a avaliação dos modelos candidatos, esta etapa consiste em selecionar aquele que apresenta o melhor equilíbrio entre desempenho e viabilidade de implementação. A decisão é fundamentada nos critérios definidos previamente, considerando tanto os aspectos quantitativos, como as métricas de desempenho (precisão, recall e F1-score), quanto os aspectos operacionais e econômicos, como os custos de utilização, escalabilidade e facilidade de integração.

A seleção prioriza o modelo que oferece os melhores resultados na tarefa de **classificação binária de fake news**, conforme estabelecido na Seção de Definição da Tarefa. Além do desempenho, são levados em conta fatores como:

- **Custo de inferência**: especialmente relevante no caso de modelos disponibilizados via API, como GPT da OpenAI ou modelos hospedados em plataformas como Amazon Bedrock.
- **Tempo de resposta**: fundamental para aplicações que exigem análise quase em tempo real.
- **Facilidade de implementação**: considerando a compatibilidade com a infraestrutura existente, necessidade (ou não) de fine-tuning e maturidade das ferramentas disponíveis.
- **Capacidade de generalização**: avaliando se o modelo mantém bom desempenho mesmo quando exposto a dados fora do conjunto de treinamento, o que é particularmente importante na detecção de fake news, devido à rápida evolução dos padrões de desinformação.

Com base nessa análise, o modelo selecionado será aquele que atende de forma mais eficiente aos requisitos definidos, garantindo não apenas alto desempenho na tarefa, mas também viabilidade técnica e econômica para uma eventual implementação em um cenário produtivo.

Capítulo 5

Experimentos

Este capítulo descreve a metodologia utilizada para a detecção de fake news em textos online, seguindo o fluxo definido no Capítulo 4. A estrutura metodológica adotada foi inspirada na dissertação de JÚNIOR (2024), adaptando o contexto do problema para o domínio de fake news, que apresenta desafios específicos relacionados à veracidade da informação e ao impacto negativo da desinformação em diversos setores da sociedade.

5.1 Definição do Cenário

A definição do cenário começa com a identificação do problema, etapa crucial para garantir que o escopo da solução seja corretamente estabelecido. Conforme abordado por JÚNIOR (2024), uma definição clara do problema é essencial para orientar o desenvolvimento da metodologia e garantir que os objetivos da solução sejam diretamente relacionados às necessidades do contexto.

No presente trabalho, o problema é definido da seguinte forma: ”Uma plataforma de notícias enfrenta desafios relacionados à disseminação de fake news, o que afeta negativamente a credibilidade da plataforma e prejudica seus usuários. Devido ao grande volume de textos gerados diariamente, a análise manual de conteúdo por moderadores humanos se torna inviável. O aumento da circulação de notícias falsas tem sido uma preocupação constante, tanto para usuários quanto para anunciantes, levando a uma queda na confiança da plataforma.”

Com base na definição do problema, são estabelecidos os componentes do cenário, que incluem o contexto, as partes interessadas, os objetivos de negócio e as métricas de sucesso. Esses componentes são apresentados na tabela 5.1

Tabela 5.1: Contexto, Partes Interessadas, Objetivos de Negócio e Métricas de Sucesso

Contexto	Desenvolvimento de um sistema de detecção de fake news para plataformas de notícias e redes sociais.
Partes Interessadas	Usuários da plataforma, jornalistas, moderadores, gestores de produto, proprietários da plataforma de notícias, anunciantes.
Objetivos de Negócio	Reduzir a disseminação de notícias falsas, melhorar a credibilidade da plataforma, proteger a experiência dos usuários, aumentar a eficiência na moderação de conteúdo e preservar a imagem da plataforma.
Métricas de Sucesso	Redução na taxa de disseminação de notícias falsas, aumento da precisão na classificação de conteúdo, redução no número de denúncias relacionadas a fake news e aumento na confiança dos usuários.

Seguindo a abordagem de JÚNIOR (2024), foi aplicado o framework IRACIS RUBLE (1997) para converter o problema identificado em frases objetivas. Isso permite que as propostas de solução sejam avaliadas com base em seu impacto. No presente contexto, o framework foi utilizado para identificar as oportunidades de melhoria e definir objetivos concretos para o sistema de detecção de fake news.

Na próxima seção, será definida a tarefa a ser executada, considerando o ambiente de implementação, o fluxo de conteúdo na plataforma e o entendimento do cenário estabelecido.

5.2 Deterninação da Tarefa

Nesta etapa, busca-se especificar claramente a tarefa central deste trabalho, que consiste na **detecção automática de fake news a partir de textos noticiosos**. Essa definição é fundamental para guiar as etapas subsequentes do desenvolvimento e avaliação dos modelos.

A partir da análise do cenário e das características da base de dados de notícias e conteúdos compartilhados, define-se que a tarefa corresponde a um problema de **classificação binária** no domínio de Processamento de Linguagem Natural (PLN). O objetivo é desenvolver e avaliar modelos capazes de receber como entrada um texto — proveniente de notícias ou postagens em redes sociais — e produzir como saída uma previsão indicando se aquele conteúdo é **Fake** ou **Não Fake**.

Para uma definição precisa do problema, também se torna relevante compreender os diferentes tipos de desinformação observados, ilustrando com exemplos reais

Tarefa	Saída
Classificação de Fake News	Fake, Não Fake

Tabela 5.2: Definição da tarefa de classificação de fake news

quando possível, de modo a delimitar com clareza o escopo da tarefa.

Ao contrário de cenários voltados a objetivos comerciais, este trabalho foca na eficácia técnica e científica da solução, buscando contribuir para o avanço da pesquisa em detecção de fake news. Assim, as decisões metodológicas são orientadas por critérios como:

- Escolha de modelos baseados em PLN, especialmente Large Language Models (LLMs);
- Técnicas de pré-processamento e representação textual, como embeddings ou prompts adaptados;

Portanto, a tarefa principal que norteia este trabalho é o desenvolvimento de um sistema capaz de realizar a **classificação binária de notícias como Fake ou Não Fake**, utilizando modelos de linguagem natural, com foco na robustez, confiabilidade e aplicabilidade da solução no enfrentamento da desinformação.

5.3 Levantamento de Recursos

Para a realização desta pesquisa, foi essencial garantir a disponibilidade de recursos necessários para sua execução, especialmente em infraestrutura, considerando seu escopo e proposta envolvendo modelos de linguagem natural (LLMs). Felizmente, a maior parte dos recursos necessários foi obtida através de créditos cedidos pela Amazon Web Services (AWS), o que possibilitou o acesso à infraestrutura necessária para o processamento de dados e a implementação dos modelos, majoritariamente através da plataforma AWS SageMaker.

Apesar de viabilizarem a pesquisa, os créditos na AWS impuseram algumas restrições, implicando no uso de LLMs disponíveis dentro da plataforma AWS SageMaker, que, apesar da ampla oferta de modelos, não possuía disponibilidade de alguns dos modelos mais recentes. O alto custo da infraestrutura para os modelos também teve impacto no escopo da pesquisa, com um custo total acumulado superior a mil dólares. Os créditos foram cedidos pelo programa Chamada CNPq/AWS Nº 64/2022 – Acesso às Plataformas de Computação em Nuvem da AWS (Cloud Credits for Research).

5.4 Definição das bases de dados

A seleção das bases de dados é um aspecto fundamental para a avaliação da adaptabilidade dos modelos de LLM na tarefa de detecção de fake news. Para garantir uma análise abrangente, buscamos bases de dados com características distintas, que se enquadrassem no escopo deste trabalho. As bases candidatas incluíram: LIAR WANG (2017), PolitiFact MISRA (2022), FakeNewsNet SHU e MAHUDESWARAN (2019), Fake-News-Ukraine[1], WELFAKE[2], COVID PATWA *et al.* (2021), FakeNewsCorpus[3], ISOT MARTINS SAMUEL DOGO e JUREK-LOUGHREY (2020), e a base de Silva SILVA *et al.* (2020), sendo esta última a única em português.

Durante o processo de seleção, algumas bases foram descartadas. A base Silva SILVA *et al.* (2020) foi excluída porque a maioria dos algoritmos testados nesta base obteve uma taxa de acerto próxima a 100%, o que indicava que a base não representava um desafio significativo para os modelos e poderia enviesar os resultados. Além disso, a base FakeNewsCorpus foi desconsiderada devido ao seu tamanho, que ultrapassa 9 GB. O alto custo computacional e financeiro para processar essa base tornaria sua utilização inviável dentro das limitações deste estudo.

Dessa forma, após a análise das bases de dados candidatas, foram selecionadas aquelas que melhor atendiam aos critérios deste estudo, considerando diversidade, relevância e viabilidade técnica. As bases escolhidas para os experimentos foram LIAR, COVID, FakeNewsNet, WELFAKE, PolitiFact e ISOT, pois apresentam um conjunto variado de notícias falsas e verdadeiras, permitindo uma avaliação abrangente do desempenho dos modelos de LLM na detecção de fake news. Além disso, essas bases possuem tamanhos adequados para processamento dentro das limitações computacionais deste trabalho, garantindo a realização de experimentos viáveis e representativos.

5.4.1 Análise das bases de dados

Para uma melhor compreensão das características dos conjuntos de dados utilizados neste estudo, apresentamos histogramas da distribuição da quantidade de tokens por notícia em cada base com remoção de outliers. A exclusão de outliers permite visualizar a tendência central da distribuição sem a influência de valores extremos, que poderiam distorcer a análise e mascarar padrões relevantes. Além disso, incluímos gráficos que ilustram a distribuição de notícias entre as classes fake e real em cada dataset, permitindo uma avaliação do balanceamento das bases e do impacto que essa distribuição pode ter no desempenho dos modelos de LLM na

[1] `https://ieee-dataport.org/documents/propaganda-and-fake-news-war-ukraine`
[2] `https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification`
[3] `https://github.com/several27/FakeNewsCorpus`

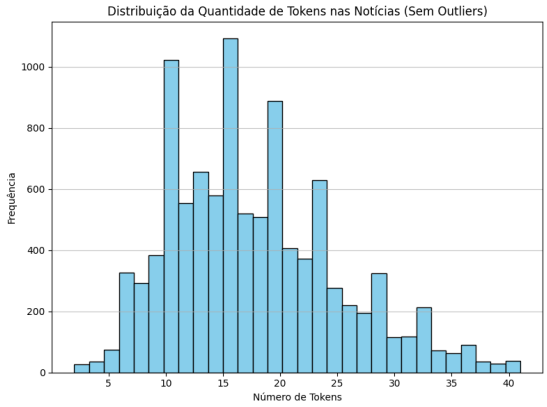
tarefa de detecção de fake news.

A análise das bases de dados utilizadas neste estudo revela que, com exceção da base PolitiFact, a maioria das bases apresenta um equilíbrio entre as classes de notícias falsas e verdadeiras, conforme ilustrado nas Figuras 5.1b e 5.4b. A base PolitiFact 5.5b, por sua vez, apresenta um desbalanceamento significativo, com cerca de três vezes mais notícias falsas do que notícias verdadeiras. Apesar dessa diferença, a média de tokens por classe é semelhante entre as bases, como pode ser observado na Figura 5.1c. Os histogramas de distribuição da quantidade de tokens, apresentados na Figura 5.1a, revelam a diversidade das bases. Na base LIAR, por exemplo, a maioria das notícias possui entre 10 e 25 tokens, enquanto bases como ISOT, WELFAKE e FakeNewsNet possuem uma maior quantidade de notícias com entre 300 e 1000 tokens, conforme mostrado nas Figuras 5.4a, 5.6a e 5.3a. Essas variações indicam diferenças significativas no nível de detalhamento das notícias em cada base.

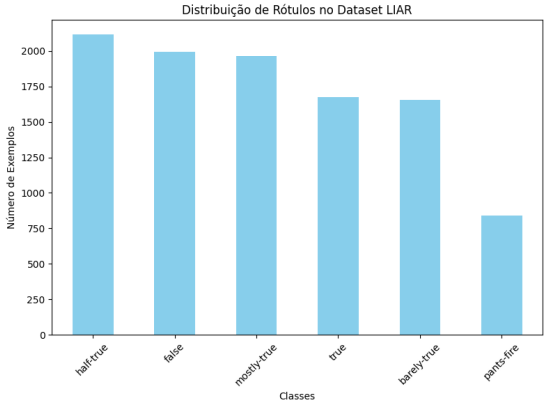
Além disso, para garantir a consistência e a robustez na avaliação dos modelos, todos os *datasets* selecionados foram divididos em conjuntos de treinamento e teste, respeitando a proporção original e a distribuição das classes em cada base. Durante os experimentos, todas as inferências realizadas pelos modelos foram feitas exclusivamente sobre o conjunto de teste, assegurando uma avaliação imparcial. A amostragem do conjunto de treinamento foi utilizada apenas para o processo de *fine-tuning* dos modelos, permitindo o ajuste de seus parâmetros sem exposição prévia ao conjunto de teste. Essa abordagem possibilita a utilização dos mesmos conjuntos de dados para analisar comparativamente a performance dos modelos tunados e não tunados, garantindo a confiabilidade e a validade dos resultados obtidos.

5.4.2 Dataset LIAR

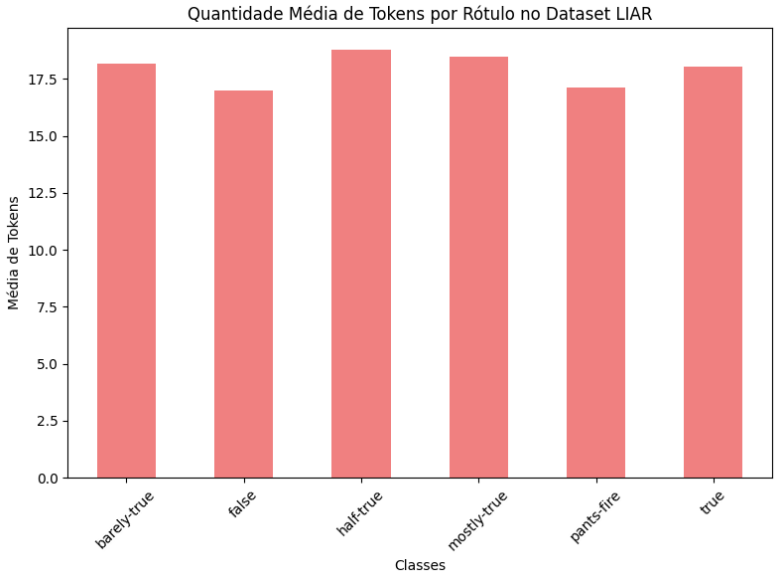
Os gráficos abaixo apresentam a distribuição estatística dos tokens e das classes na base LIAR, composta por 10.270 notícias. O histograma exibe a frequência dos tokens no conjunto de dados, enquanto a distribuição das classes mostra a proporção de cada categoria presente. Além disso, a distribuição da quantidade de tokens por classe permite visualizar variações no tamanho dos textos para cada rótulo. Embora a base de dados LIAR possua 6 classes que indicam graus de veracidade diferentes, para esse estudo as notícias com os rótulos *'true'* e *'mostly-true'* foram considerados como verdadeiras e as demais como notícias falsas.



(a) Histograma da distribuição de tokens na base LIAR.



(b) Distribuição das classes na base LIAR.



(c) Distribuição da quantidade de tokens por classe na base LIAR.

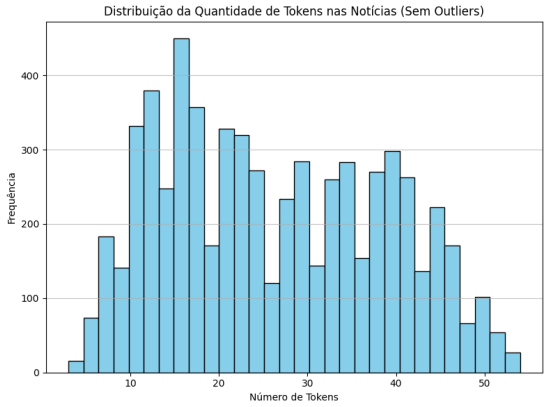
Figura 5.1: Análise da base LIAR: distribuição da quantidade de tokens, distribuição de classes e distribuição de tokens por classe.

Tabela 5.3: Dataset Liar

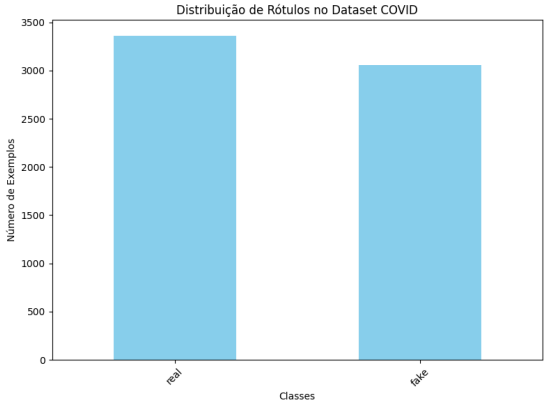
Classificação	**Texto da notícia**
`true`	Building a wall on the U.S.-Mexico border will take literally years.
`false`	Wisconsin is on pace to double the number of layoffs this year.
`false`	Says John McCain has done nothing to help the vets.
`half-true`	Suzanne Bonamici supports a plan that will cut choice for Medicare Advantage seniors.
`pants-fire`	When asked by a reporter whether he's at the center of a criminal scheme to violate campaign laws, Gov. Scott Walker nodded yes.

5.4.3 Dataset COVID

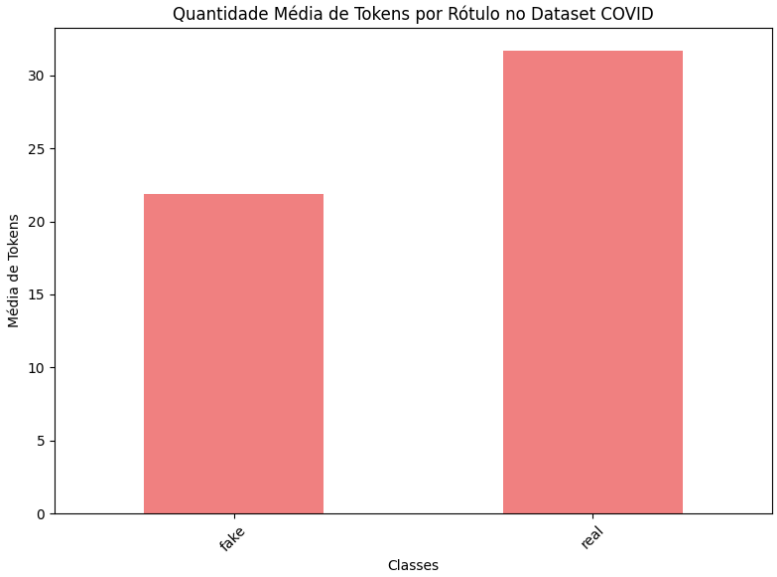
A base COVID, contendo 6.420 notícias, possui gráficos que ilustram a análise da distribuição dos tokens e das classes. O histograma apresenta a dispersão dos tokens no corpus, enquanto a distribuição das classes revela a proporção dos diferentes rótulos. A distribuição da quantidade de tokens por classe também evidencia possíveis padrões de tamanho textual associados a cada categoria.



(a) Histograma da distribuição de tokens na base COVID.



(b) Distribuição das classes na base COVID.



(c) Distribuição da quantidade de tokens por classe na base COVID.

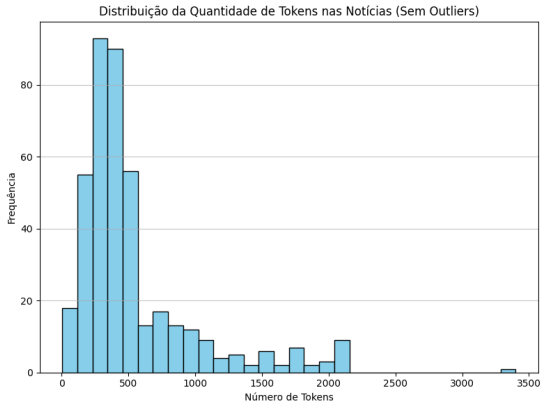
Figura 5.2: Análise da base COVID: distribuição da quantidade de tokens, distribuição de classes e distribuição de tokens por classe.

Tabela 5.4: Exemplos de tweets rotulados no dataset COVID

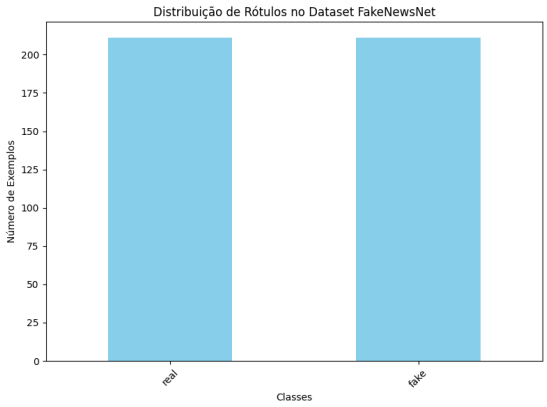
Classificação	**Texto do tweet**
`real`	The CDC currently reports 99031 deaths. In general the discrepancies in death counts between different sources are small and explicable. The death toll stands at roughly 100000 people today.
`real`	States reported 1121 deaths a small rise from last Tuesday. Southern states reported 640 of those deaths. https://t.co/YASGRTT4ux
`fake`	Politically Correct Woman (Almost) Uses Pandemic as Excuse Not to Reuse Plastic Bag https://t.co/thF8GuNFPe #coronavirus #nashville
`real`	#IndiaFightsCorona: We have 1524 #COVID testing laboratories in India and as on 25th August 2020 36827520 tests have been done : @ProfBhargava DG @ICMRDELHI #StaySafe #IndiaWillWin https://t.co/Yh3ZxknnhZ

5.4.4 Dataset FakenewsNet

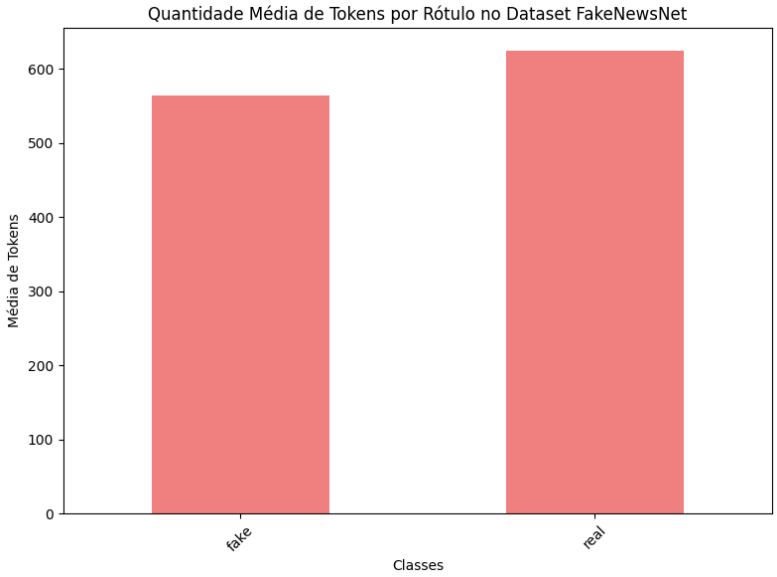
FakeNewsNet (422 notícias): Com um total de 422 notícias, a base FakeNewsNet apresenta gráficos que analisam a distribuição de tokens e classes. O histograma reflete a frequência dos tokens, a distribuição das classes exibe a proporção dos rótulos existentes, e a variação da quantidade de tokens por classe permite observar como o tamanho dos textos se comporta entre os diferentes grupos.



(a) Histograma da distribuição de tokens na base FakenewsNet.



(b) Distribuição das classes na base FakenewsNet.



(c) Distribuição da quantidade de tokens por classe na base FakenewsNet.

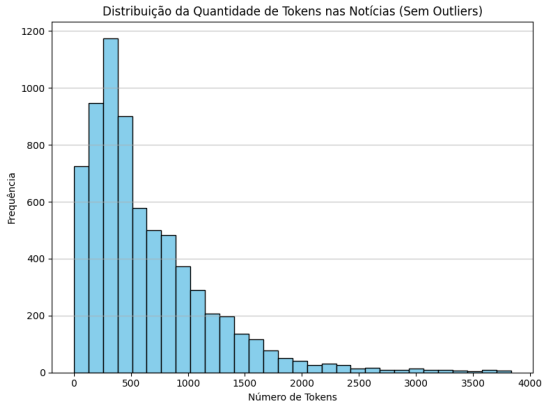
Figura 5.3: Análise da base FakenewsNet: distribuição da quantidade de tokens, distribuição de classes e distribuição de tokens por classe.

Tabela 5.5: Exemplos de notícias rotuladas no dataset FakeNewsNet

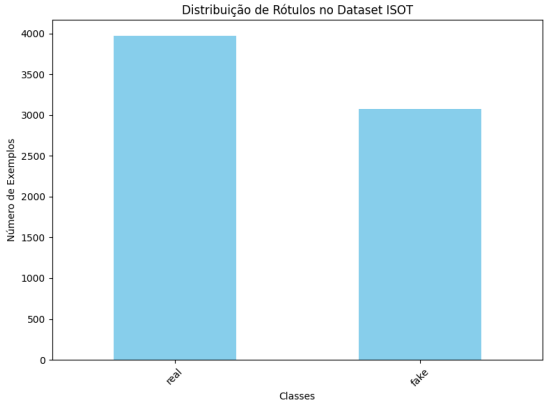
Classificação	**Texto da notícia**
`real`	Both women are former models, but Alicia Machado attacked Melania Trump for failing to do more to help those around her as the wife of a prominent political figure. [...] Trump said on Fox News' "Fox & Friends." "She was the winner, and, you know, she gained a massive amount of weight, and it was a real problem. [...]"
`fake`	When Hillary Clinton attacked Donald Trump for not releasing his tax returns and allegedly paying zero in federal income taxes, the Republican presidential nominee had just four words to say in response. "That makes me smart," Trump said. [...] Clinton suggested that Trump's alleged failure to pay federal income tax might also be part of the problem. But Trump quickly argued any taxes he would have paid would have gone to waste.

5.4.5 Dataset ISOT

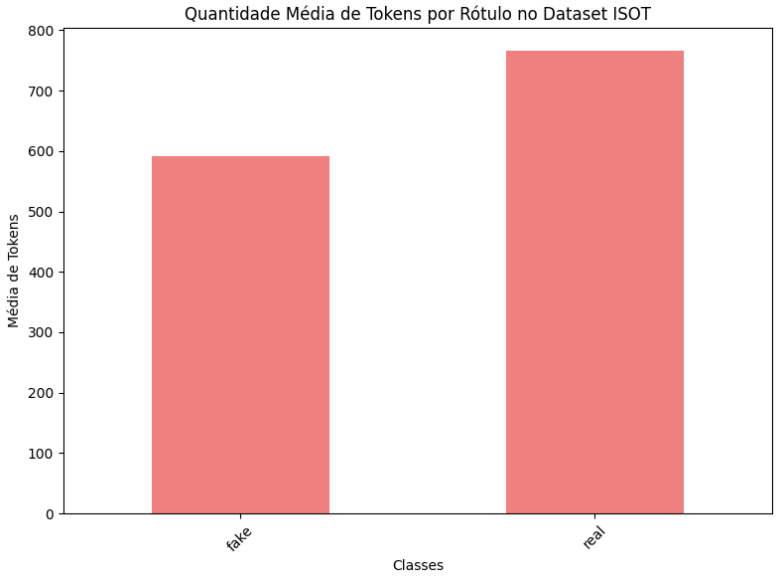
A base ISOT, composta por 7.050 notícias, possui gráficos que apresentam o histograma da distribuição de tokens, a distribuição das classes e a variação do número de tokens por categoria. Essas visualizações ajudam a compreender melhor a estrutura do conjunto de dados e suas particularidades.



(a) Histograma da distribuição de tokens na base ISOT.



(b) Distribuição das classes na base ISOT.



(c) Distribuição da quantidade de tokens por classe na base ISOT.

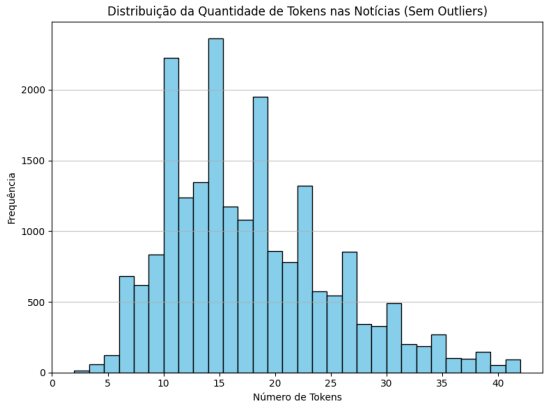
Figura 5.4: Análise da base ISOT: distribuição da quantidade de tokens, distribuição de classes e distribuição de tokens por classe.

Tabela 5.6: Exemplos de notícias rotuladas no dataset ISOT

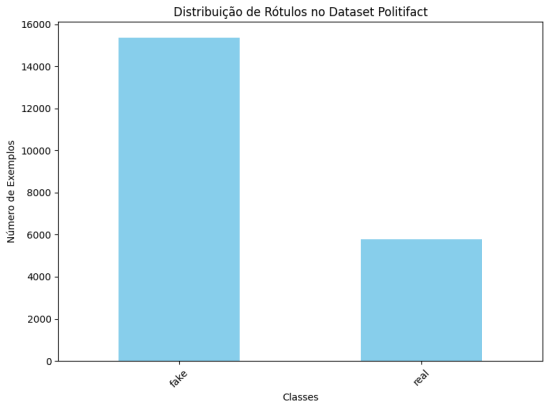
Classificação	**Texto da notícia**
`fake`	One Twitter user once asked, “Is it too much to hope that one of our parties would pick someone with no links to a pedophile sex-slave island?” Apparently, it is. Both Trump and Clinton have ties to Jeffrey Epstein and “Sex Slave Island.” [...]
`real`	(CNN) Investigators have pieced together a timeline of Ahmad Rahami’s movements between bombings and his arrest. Surveillance footage and his sister’s phone helped confirm the sequence of events. [...]
`real`	Just days after a racist Snapchat post at the University of North Dakota, another surfaced showing white students in blackface captioned “Black Lives Matter.” The university president vowed to strengthen diversity education. [...]

5.4.6 Dataset PolitFact

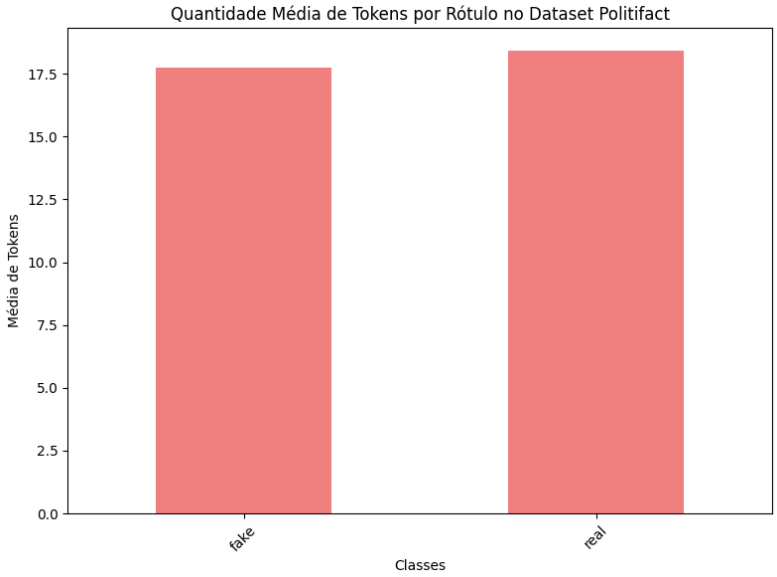
A base PolitiFact, com 21.152 notícias, é representada nos gráficos a seguir por meio da análise da distribuição de tokens, da proporção das classes e da variação da quantidade de tokens por categoria. Essas informações são úteis para entender a composição do dataset e suas características.



(a) Histograma da distribuição de tokens na base PolitFact.



(b) Distribuição das classes na base PolitFact.



(c) Distribuição da quantidade de tokens por classe na base PolitFact.

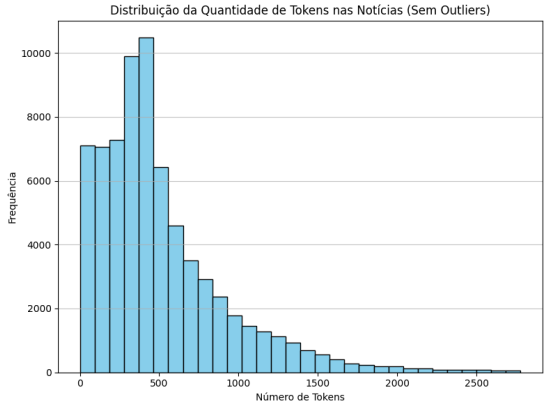
Figura 5.5: Análise da base PolitFact: distribuição da quantidade de tokens, distribuição de classes e distribuição de tokens por classe.

Tabela 5.7: Exemplos de afirmações rotuladas no dataset PolitiFact

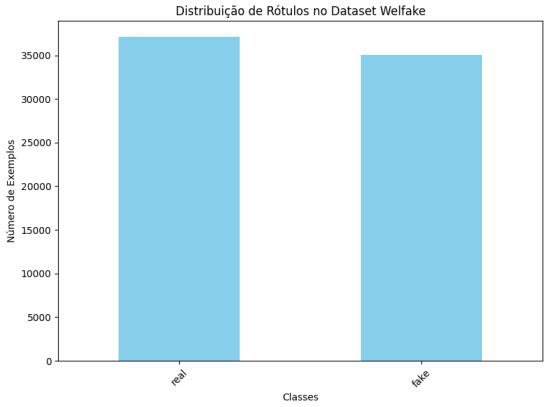
Classificação	**Texto da afirmação**
`real`	John McCain opposed bankruptcy protections for families "who were only in bankruptcy because of medical expenses they couldn't pay."
`fake`	"Bennie Thompson actively cheer-led riots in the '90s."
`real`	Says Maggie Hassan was "out of state on 30 days over the last three months."

5.4.7 Dataset WelFake

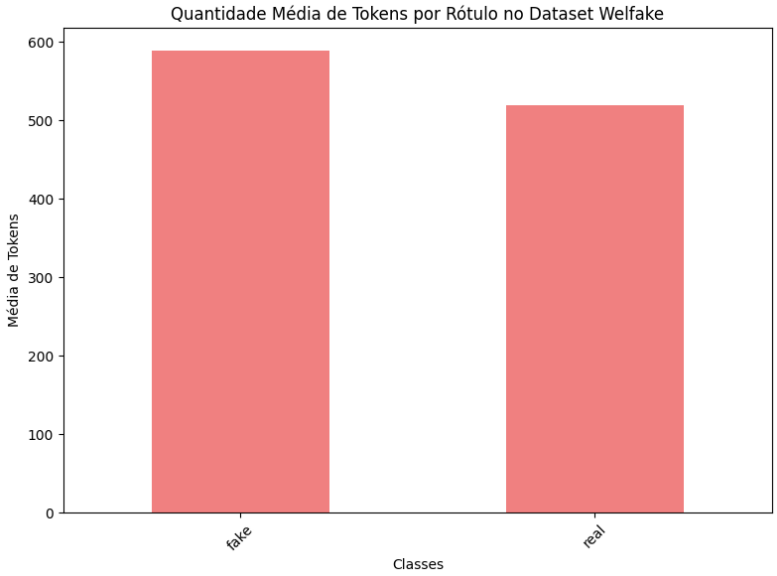
A base WeFake, originalmente composta por mais de 300.000 notícias, foi amostrada para 72.134 registros, devido a restrições de armazenamento em um dataframe do Python. Os gráficos a seguir apresentam a distribuição dos tokens, a proporção das classes e a variação da quantidade de tokens por categoria. Essas visualizações permitem identificar padrões estruturais no conjunto de dados, bem como possíveis desequilíbrios entre classes.



(a) Histograma da distribuição de tokens na base WelFake.



(b) Distribuição das classes na base WelFake.



(c) Distribuição da quantidade de tokens por classe na base WelFake.

Figura 5.6: Análise da base WelFake: distribuição da quantidade de tokens, distribuição de classes e distribuição de tokens por classe.

Tabela 5.8: Exemplos de afirmações rotuladas no dataset Welfake

Classificação	**Texto da afirmação**
`real`	”LAW ENFORCEMENT ON HIGH ALERT Following Threats Against Cops And Whites On 9-11By BlackLivesMatter And FYF911 Terrorists [VIDEO]: No comment is expected from Barack Obama Members of the FYF911 or FukYoFlag and BlackLivesMatter movements called for the lynching and hanging of white people and cops. They encouraged others on a radio show Tuesday night to turn the tide and kill white people and cops to send a message about the killing of black people in America.One of the F***YoFlag organizers is called Sunshine. She has a radio blog show hosted from Texas called, Sunshine s F***ing Opinion Radio Show. [...] ”
`fake`	”Bobby Jindal, raised Hindu, uses story of Christian conversion to woo evangelicals for potential 2016 bid: A dozen politically active pastors came here for a private dinner Friday night to hear a conversion story unique in the context of presidential politics: how Louisiana Gov. Bobby Jindal traveled from Hinduism to Protestant Christianity and, ultimately, became what he calls an “evangelical Catholic.” Over two hours, Jindal, 42, recalled talking with a girl in high school who wanted to “save my soul,” reading the Bible in a closet so his parents would not see him and feeling a stir while watching a movie during his senior year that depicted Jesus on the cross. “I was struck, and struck hard,” Jindal told the pastors. “This was the Son of God, and He had died for our sins.” [...]

5.5 Definição da Implementação

Nesta fase do projeto, foram desenvolvidos os *scripts* responsáveis pela implementação dos modelos, fazendo uso, principalmente, das bibliotecas *Transformers* e *SageMaker*. A biblioteca *Transformers*, da *Hugging Face*, foi utilizada para simplificar o uso de modelos de linguagem já pré-treinados, permitindo uma adaptação ágil desses modelos às demandas específicas da tarefa. Por sua vez, o serviço *SageMaker*, da AWS, foi empregado para gerenciar os experimentos e viabilizar sua execução em escala, oferecendo toda a infraestrutura computacional necessária.

Diante do elevado volume de dados, os *scripts* foram ajustados para registrar informações relevantes durante a execução dos modelos em cada *prompt* configurado,

como o tempo de processamento e a quantidade de *tokens* tanto na entrada quanto na saída. Esses dados são fundamentais para a avaliação do desempenho dos modelos e servirão de base para as análises posteriores de performance e custo. Os códigos desenvolvidos encontram-se no Anexo B.

5.5.1 Métricas de Avaliação

Para avaliar os modelos desenvolvidos, foram adotadas as métricas tradicionais de classificação: *F1-score*, acurácia, precisão e *recall*. Tais métricas são amplamente reconhecidas na literatura por sua eficácia em fornecer uma visão abrangente sobre o desempenho de modelos classificadores.

Esses indicadores permitiram realizar uma análise minuciosa dos resultados, garantindo que os modelos estejam alinhados com os objetivos propostos nesta pesquisa. A coleta dessas informações e sua aplicação nas avaliações foram essenciais para subsidiar a etapa seguinte de análise dos modelos, conforme detalhado na Seção 4.11.

Acurácia

A acurácia mede a proporção de previsões corretas (tanto positivas quanto negativas) em relação ao total de previsões realizadas. Ela oferece uma visão geral da taxa de acertos do modelo, sendo particularmente útil quando as classes estão balanceadas.

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Onde:
TP = Verdadeiros Positivos (True Positives)
TN = Verdadeiros Negativos (True Negatives)
FP = Falsos Positivos (False Positives)
FN = Falsos Negativos (False Negatives)

Precisão

A precisão (*Precision*) representa a proporção de instâncias classificadas como positivas que são de fato positivas. Essa métrica é especialmente relevante quando o custo de falsos positivos é alto.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{5.2}$$

Ela responde à pergunta: “Das previsões que o modelo fez como sendo ’Fake’, quantas realmente são fake?”

Recall

O *Recall*, também conhecido como sensibilidade, mede a capacidade do modelo em identificar corretamente todas as instâncias positivas. Ele é crucial quando o custo de falsos negativos é elevado, ou seja, quando é mais crítico deixar de detectar um caso positivo.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

Essa métrica responde à pergunta: "De todos os casos que realmente são 'Fake', quantos o modelo conseguiu identificar?"

F1-Score

O *F1-Score* é a média harmônica entre a precisão e o recall, fornecendo um equilíbrio entre essas duas métricas. Ele é especialmente útil quando se deseja buscar um compromisso entre evitar falsos positivos e falsos negativos.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{5.4}$$

O F1-score se mostra particularmente relevante em cenários onde há desequilíbrio entre as classes, pois penaliza modelos que tenham um bom desempenho em uma métrica, mas baixo desempenho na outra.

5.6 Selecionar LLMs

Nos experimentos, foram utilizadas três famílias de modelos distintas: a família GEMMA, onde foram utilizadas as versões GEMMA 2B Instruct e GEMMA 7B Instruct; a família LLaMA, onde utilizamos o modelo LLaMA 3.1 70B Instruct; e por fim, a família DeepSeek, da qual utilizamos o modelo DeepSeek-R1 Distill Qwen 32B.

A família GEMMA foi desenvolvida pela Google como parte da linha Gemini GEMINI (2023). O GEMMA 2B Instruct, com 2 bilhões de parâmetros, é um modelo de médio porte que se destaca pela eficiência em tarefas que exigem um bom equilíbrio entre desempenho e custo. Já o GEMMA 7B Instruct, com 7 bilhões de parâmetros, oferece maior capacidade de processamento, sendo mais adequado para tarefas complexas que exigem maior poder computacional. Ambos os modelos seguem a abordagem "Instruct", o que significa que foram treinados com um foco em interpretar e executar instruções humanas de maneira mais eficiente. Isso os diferencia de modelos não "Instruct", que geralmente não são especificamente treinados para compreender comandos ou instruções em linguagem natural, mas sim

para realizar tarefas de maneira mais geral. Os modelos Instruct são particularmente vantajosos quando se busca um desempenho melhor em interações baseadas em linguagem, pois são projetados para entender a intenção do usuário de forma mais precisa.

A família LLaMA, desenvolvida pelo Meta e apresentada em 2023, inclui o modelo LLaMA 3.1 70B Instruct, com 70 bilhões de parâmetros, que foi projetado para tarefas de entendimento e geração de linguagem natural TOUVRON *et al.* (2023). Assim como os modelos GEMMA, o LLaMA 3.1 70B Instruct segue a abordagem "Instruct", sendo ajustado para interpretar melhor as instruções e oferecer respostas mais detalhadas e precisas. Esse ajuste torna o modelo altamente eficaz em interações que exigem um entendimento profundo e execução precisa de comandos.

Além desses, também utilizamos o modelo DeepSeek-R1 Distill Qwen 32B, uma versão densa derivada do DeepSeek-R1 e baseada na arquitetura Qwen. Lançado em fevereiro de 2025, esse modelo não segue a abordagem "Instruct", mas foi desenvolvido com foco em raciocínio, tendo sido destilado a partir de padrões descobertos em modelos maiores, como parte da linha DeepSeek DEEPSEEK-AI *et al.* (2025). O DeepSeek-R1 Distill Qwen 32B demonstra desempenho competitivo com modelos como o OpenAI-o1-mini, alcançando resultados de estado da arte em diversos benchmarks, especialmente em tarefas que envolvem cadeia de pensamento e verificação. Apesar de seu potencial, apenas um número limitado de experimentos foi conduzido com esse modelo devido a restrições orçamentárias que limitaram seu uso em larga escala.

5.7 Elaboração de Prompts

Nesta seção, detalhamos a seleção das estratégias de *prompt* utilizadas nos experimentos para a detecção de *fake news*. Dado o vasto número de estratégias, foram adotadas as estratégias *one-shot, few-shot, prompt priming* e *output constraint*.

Outro elemento considerado na pesquisa foi o impacto das definições das classes da tarefa no *prompt*. Para cada uma dessas estratégias, testamos variações de prompts que incluem ou não a definição das classes da tarefa de classificação: determinar se a notícia é falsa ou verdadeira. Assim como no trabalho JÚNIOR (2024), a inclusão das definições visa fornecer uma melhor orientação ao modelo, potencialmente melhorando a qualidade da classificação. As variações de *prompt* consideradas foram:

- **Nenhuma Definição:** não fornecem qualquer definição sobre o que caracteriza uma notícia falsa ou verdadeira.

- **Definição de Falso (F):** incluem uma definição sobre o que caracteriza uma notícia falsa.
- **Definição de Verdadeiro (V):** incluem uma definição sobre o que caracteriza uma notícia verdadeira.
- **Nenhuma de Ambos:** incluem tanto uma definição sobre o que caracteriza uma notícia ser falsa quanto do que caracteriza uma notícia ser verdadeira.

Considerando as estratégias e variações selecionadas, cada modelo foi avaliado utilizando um total de oito diferentes *prompts*. No entanto, no início dos experimentos, observou-se uma alta taxa de alucinações geradas pelos modelos de linguagem (*LLMs*). Para mitigar esse problema, foi adotada a estratégia *output constraint* LIU *et al.* (2024), especificando que a resposta deveria se limitar à classificação da notícia como verdadeira ou falsa. Essa abordagem mostrou-se altamente eficaz, reduzindo significativamente a ocorrência de alucinações, que em diversos experimentos foram eliminadas ou minimizadas a níveis próximos de zero.

Ainda assim, em alguns casos, os modelos retornaram a classificação acompanhada de um texto explicando o motivo da decisão. No entanto, mesmo nesses casos, foi possível parsear a resposta facilmente, garantindo que o resultado final permanecesse adequado ao objetivo da avaliação.

5.8 Fine Tuning

Para manter a consistência e a confiabilidade dos experimentos, todos os datasets utilizados no processo de *fine-tuning* foram previamente divididos em conjuntos de treinamento e teste. Essa divisão respeita a distribuição original das classes em cada base, garantindo que os modelos possam aprender de forma representativa a partir dos dados de treino. Durante os experimentos, todas as avaliações de desempenho foram realizadas exclusivamente sobre o conjunto de teste, de modo a evitar qualquer viés causado pela exposição prévia do modelo aos dados avaliados.

É importante destacar que, para os modelos que não passaram pelo *fine-tuning*, a parte de treino foi desconsiderada, e as inferências ocorreram diretamente sobre os dados de teste, permitindo uma comparação justa entre modelos tunados e não tunados. Dessa forma, o uso do conjunto de treinamento restringe-se apenas ao ajuste fino dos parâmetros dos modelos que passaram pelo *fine-tuning*, preservando a integridade da avaliação e possibilitando uma análise clara dos ganhos proporcionados por esse procedimento.

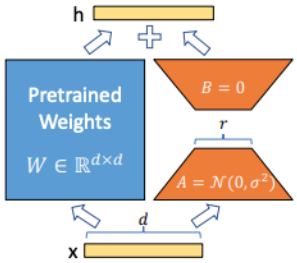


Figura 5.7: LoRA *reparametrization*

5.8.1 Lora

Para viabilizar o fine-tuning de grandes modelos de linguagem, empregamos a técnica **LoRA** (*Low-Rank Adaptation*). O LoRA foi proposto por Hu et al. HU *et al.* (2021) como uma abordagem eficiente para adaptar *Large Language Models* com um número reduzido de parâmetros, permitindo um ajuste mais eficiente e com menor custo computacional.

Uma das ideias centrais do LoRA é o uso de matrizes de *low-rank* nas camadas do modelo. Mas o que são matrizes de *low-rank*? Em termos matemáticos, o **rank** de uma matriz é o número de colunas (ou linhas) linearmente independentes nela. Em uma matriz de *low-rank*, esse número é significativamente menor do que as dimensões da matriz, o que significa que a matriz contém menos informação independente e pode ser aproximada de forma mais compacta.

Por exemplo, considere uma matriz W de tamanho $m \times n$. Se o rank dessa matriz for baixo, podemos decompor W como o produto de duas matrizes de menor ordem, A e B, de tal forma que $W \approx A \cdot B^T$, onde A tem dimensões $m \times r$ e B tem dimensões $n \times r$, e r é um número muito menor que m ou n. Isso reduz o número de parâmetros necessários para representar a matriz, sem perder muita informação relevante.

Essa decomposição de matrizes de *low-rank* é aproveitada pelo LoRA. Em vez de treinar todos os parâmetros do modelo de forma tradicional, LoRA insere matrizes de *low-rank* nas camadas de atenção e feed-forward do modelo. Essas matrizes de adaptação são treinadas enquanto os parâmetros originais do modelo são mantidos congelados, como podemos ver na figura 5.7. Assim, o LoRA permite modificar o comportamento do modelo sem a necessidade de atualizar todos os parâmetros originais, o que torna o processo de fine-tuning mais eficiente em termos de memória e tempo de treinamento.

No caso do fine-tuning de modelos como **GEMMA 7B Instruct** e **LLaMA 3.1 70B Instruct**, o LoRA foi fundamental para possibilitar o ajuste desses modelos com grandes quantidades de parâmetros, respeitando as restrições orçamentárias computacionais. Em vez de treinar milhões ou bilhões de parâmetros adicionais,

o LoRA permite que apenas um número reduzido de parâmetros (relacionados às matrizes de *low-rank*) seja ajustado durante o processo de fine-tuning.

Essa abordagem não só reduz o custo computacional, mas também diminui o risco de overfitting, pois o número de parâmetros treináveis é muito menor. Assim, o LoRA se mostrou uma técnica altamente eficiente para realizar fine-tuning em modelos de linguagem grandes, sem comprometer significativamente a performance.

Hiperparâmetros utilizados

Nesta seção, descrevemos os hiperparâmetros utilizados nos processos de fine-tuning dos modelos **GEMMA 7B Instruct** e **LLaMA 3.1 70B Instruct**. Devido a limitações de orçamento, selecionamos apenas duas bases de dados para realizar o tuning: a base **LIAR**, bastante popular e desafiadora, pois contém notícias curtas com poucas palavras; e a base **COVID**, escolhida por tratar de um tema recente e por apresentar uma quantidade de tokens moderada e dispersa, conforme evidenciado no histograma apresentado na Figura 5.2a. A escolha desses parâmetros seguiu os valores padrão oferecidos pelas ferramentas utilizadas, uma vez que limitações de orçamento também impossibilitaram a exploração de diferentes configurações de fine-tuning. Além disso, apenas dois modelos puderam ser ajustados, sendo realizados 8 experimentos para cada modelo e base de dados, variando-se apenas os tipos de prompt utilizados (zero-shot e few-shot, combinados com diferentes definições: none, false e both).

Para o **GEMMA 7B Instruct**, utilizou-se o método de fine-tuning com *LoRA*, sem treinamento baseado em instruções (*instruction-train* configurado como False) e utilizando um dataset formatado para chat. O treinamento foi realizado por 1 época, com taxa de aprendizado de 0,0001, valor de LoRA r igual a 64, LoRA alpha de 16, e sem aplicação de dropout no LoRA. A quantização foi feita com 4 bits, utilizando *double quantization* habilitada e o tipo de quantização *nf4*. O batch size de treino por dispositivo foi de 1, e de avaliação foi de 2. Outros parâmetros relevantes incluem o uso de demarcação de entrada e saída (*add input output demarcation key*), warmup ratio de 0,1, treinamento sem partir do zero (*train from scratch* configurado como False), treinamento misto em 16 bits com FP16 ativado e BF16 desativado. A estratégia de avaliação ocorreu a cada 20 steps, com acumulação de gradientes a cada 4 steps, e logs registrados a cada 8 steps. Foi aplicado weight decay de 0,2 e ativada a opção de carregar o melhor modelo ao final do treinamento. Não houve limitação para o número máximo de amostras de treino ou validação, e os parâmetros de semente para reprodutibilidade foram fixados (seed igual a 10). O comprimento máximo de entrada foi definido como 2048 tokens, e a divisão dos dados para validação usou uma proporção de 20%. Adicionalmente, usou-se Adam como otimizador com $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$, e epsilon de 1×10^{-8}, com norma máxima do

gradiente limitada a 1. Não foram utilizadas técnicas de checkpointing de gradiente nem ajuste automático de batch size. O scheduler de taxa de aprendizado utilizado foi do tipo *constant with warmup*, e não foi feita utilização de Deepspeed.

No caso do **LLaMA 3.1 70B Instruct**, o fine-tuning também foi realizado sem treinamento por instruções explícitas, mas com um dataset no formato de chat, incluindo chaves de demarcação entre entrada e saída. A configuração considerou 1 época de treino, com taxa de aprendizado de 0,0001. Foram aplicadas técnicas de LoRA com valor de r igual a 8 e alpha igual a 32, sobre os módulos de projeção de consulta e valor (*q_proj* e *v_proj*). O LoRA dropout foi de 0,05. O batch size para treino e avaliação por dispositivo foi fixado em 1. A quantização foi feita em 8 bits (*Int8 quantization*), sem a utilização do FSDP. O modelo utilizou o template de chat específico do Llama 3.1. Outros parâmetros incluem o seed de 10 para reprodução dos experimentos, divisão de 20% para validação e ausência de limitação explícita de comprimento máximo de entrada. Não houve limite máximo para o número de amostras de treino ou validação, e não foi realizada paralelização adicional via *preprocessing workers*.

O planejamento experimental foi focado em explorar variações no formato dos prompts, enquanto os parâmetros de fine-tuning permaneceram fixos por limitações orçamentárias. Assim, cada modelo foi ajustado e avaliado em 8 experimentos distintos, considerando combinações entre *zero-shot* e *few-shot* com diferentes estratégias de fornecimento de definições de contexto.

5.9 Execução dos Modelos

Durante a execução dos modelos, foram realizadas as adaptações necessárias para atender às especificidades de cada arquitetura, utilizando os prompts previamente definidos. Cada modelo demanda requisitos distintos de hardware e apresenta particularidades no formato das instruções, exigindo que os prompts estejam alinhados com a sintaxe esperada por cada família de modelos, de forma a assegurar a correta interpretação das entradas fornecidas.

O uso de modelos de linguagem de grande porte (*LLMs*) impõe demandas de hardware substancialmente superiores às de equipamentos de uso pessoal, principalmente em virtude do elevado consumo de memória de vídeo (VRAM) necessário para tarefas como inferência ou treinamento. Modelos de maior porte, como o *Llama3.1-70B*, requerem aproximadamente 100GB de VRAM, o que corresponde, em termos práticos, a cerca de seis GPUs com 24GB cada[4].

Para viabilizar a execução dos experimentos e o deployment dos modelos, foram necessárias instâncias específicas da AWS que atendessem aos requisitos de memória

[4]Estimativa baseada na documentação oficial dos modelos disponíveis.

e processamento de cada arquitetura. A escolha das instâncias considerou, principalmente, a capacidade de GPU necessária para carregar os pesos completos do modelo em memória, bem como as restrições orçamentárias do projeto. A Tabela 5.9 apresenta as instâncias utilizadas, juntamente com o custo médio por hora de utilização, com base nos preços on-demand da AWS.

Tabela 5.9: Instâncias utilizadas para deployment dos endpoints

Modelo	**Instância AWS**	**Custo Médio por Hora (USD)**
GEMMA 2B	ml.g5.xlarge	1.23
GEMMA 7B Instruct	ml.g5.12xlarge	6.91
LLaMA 3.1 70B Instruct	ml.p4d.24xlarge	40.29

As instâncias da família `g5` foram selecionadas para os modelos GEMMA 2B e GEMMA 7B Instruct, por oferecerem GPUs NVIDIA A10G, que proporcionam um bom equilíbrio entre custo e desempenho para modelos de médio porte. Já para o modelo LLaMA 3.1 70B Instruct, foi necessária a utilização da instância `p4d.24xlarge`, equipada com múltiplas GPUs NVIDIA A100, capazes de suportar as demandas elevadas de processamento e memória exigidas por modelos de larga escala.

Após a configuração dos endpoints e a definição dos prompts, considerou-se a possibilidade de aplicar técnicas de fine-tuning aos modelos. Contudo, alinhado aos objetivos deste trabalho — que priorizam uma solução economicamente viável, dentro das limitações de tempo e de infraestrutura —, foi adotada, sempre que possível, a estratégia de *Prompt Engineering* como abordagem principal, devido à sua maior flexibilidade e menor custo computacional. Ainda assim, o fine-tuning foi aplicado de forma pontual nos modelos **Gemma 7B** e **Llama 3.1 70B**, especificamente sobre os datasets **COVID** e **LIAR**. No caso do dataset COVID, o fine-tuning foi empregado para avaliar se seria possível alcançar um desempenho ainda superior, visto que os resultados já eram satisfatórios apenas com *Prompt Engineering*. Por outro lado, no dataset LIAR, a aplicação do fine-tuning se mostrou necessária, uma vez que os modelos apresentaram desempenho abaixo do esperado na configuração base, além de se tratar de um dataset amplamente utilizado e referenciado na literatura, sendo relevante garantir um bom desempenho nesse cenário.

Capítulo 6

Análise Comparativa dos Resultados

Este capítulo tem como objetivo apresentar uma análise detalhada dos resultados obtidos a partir dos experimentos realizados durante a pesquisa. Serão discutidas as instâncias utilizadas para o deployment dos modelos, bem como os custos associados a cada uma delas, fornecendo uma visão completa do processo de execução dos modelos em termos de eficiência e viabilidade econômica. Além disso, serão apresentados os resultados comparativos entre os modelos testados, com foco no desempenho e nas métricas de avaliação utilizadas.

A seguir, são apresentadas definições importantes e convenções adotadas neste trabalho para facilitar a compreensão dos resultados e evitar ambiguidades.

Definição de Alucinação: Para os propósitos deste trabalho, consideramos como *alucinação* toda situação em que o modelo de linguagem (LLM) gera uma saída textual na qual não é possível extrair uma classificação clara sobre a veracidade da notícia, ou seja, quando o modelo não apresenta uma decisão definida entre notícia verdadeira ou falsa. Essa definição é fundamental para compreender os resultados e para a análise das métricas relacionadas a erros do modelo.

Interpretação dos Rótulos: Nas matrizes de confusão apresentadas ao longo dos resultados, adotamos a seguinte convenção para os rótulos: o label `true` refere-se às notícias verdadeiras, enquanto o label `false` representa as notícias falsas. Essa distinção é importante porque, embora a tarefa principal seja a detecção de notícias falsas, algumas abordagens podem interpretar os valores booleanos de forma invertida, o que pode gerar confusão. Portanto, a definição explícita dos rótulos e suas interpretações auxilia na correta análise dos resultados e na comparação entre modelos.

6.1 Resultados na Base de dados COVID

Tabela 6.1: Comparação de Resultados para Detecção de Fake News (Base COVID, Gemma-2B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-2B-Instruct, few shot (both definition)	0,5065	0,1685	0,0955	0,7133
Gemma-2B-Instruct, few shot (false definition)	0,6603	0,6661	0,6473	0,6859
Gemma-2B-Instruct, few shot (none)	0,6481	0,6955	0,7679	0,6356
Gemma-2B-Instruct, few shot (true definition)	0,6864	0,7079	0,7259	0,6907
Gemma-2B-Instruct, zero shot (both definition)	0,4771	0,0089	0,0045	0,5556
Gemma-2B-Instruct, zero shot (false definition)	0,5893	0,3909	0,2518	0,8731
Gemma-2B-Instruct, zero shot (none)	0,6070	0,5267	0,4179	0,7123
Gemma-2B-Instruct, zero shot (true definition)	0,4813	0,0561	0,0295	0,5893

Tabela 6.2: Comparação de Resultados para Detecção de Fake News (Base COVID, Gemma-7B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct, few shot (both definition)	0,8154	0,8330	0,8795	0,7912
Gemma-7B-Instruct, few shot (false definition)	0,7626	0,8011	0,9134	0,7134
Gemma-7B-Instruct, few shot (none)	0,8196	0,8439	0,9313	0,7714
Gemma-7B-Instruct, few shot (true definition)	0,7131	0,7806	0,9750	0,6508
Gemma-7B-Instruct, zero shot (both definition)	0,8266	0,8470	0,9170	0,7870
Gemma-7B-Instruct, zero shot (false definition)	0,8145	0,8313	0,8732	0,7932
Gemma-7B-Instruct, zero shot (none)	0,7313	0,7913	0,9732	0,6667
Gemma-7B-Instruct, zero shot (true definition)	0,7481	0,8018	0,9732	0,6817

Tabela 6.3: Comparação de Resultados para Detecção de Fake News (Base COVID, Llama-3-1-70B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Llama-3-1-70B-Instruct, few shot (both definition)	0,8893	0,9015	0,9688	0,8430
Llama-3-1-70B-Instruct, few shot (false definition)	0,8785	0,8932	0,9705	0,8272
Llama-3-1-70B-Instruct, few shot (none)	0,8355	0,8617	0,9795	0,7693
Llama-3-1-70B-Instruct, few shot (true definition)	0,8790	0,8926	0,9607	0,8335
Llama-3-1-70B-Instruct, zero shot (both definition)	0,8850	0,8972	0,9589	0,8430
Llama-3-1-70B-Instruct, zero shot (false definition)	0,8701	0,8831	0,9375	0,8347
Llama-3-1-70B-Instruct, zero shot (none)	0,8472	0,8661	0,9446	0,7997
Llama-3-1-70B-Instruct, zero shot (true definition)	0,8346	0,8438	0,8536	0,8342

6.2 Resultados na Base de dados LIAR

Tabela 6.4: Comparação de Resultados para Detecção de Fake News (Base LIAR, Gemma-2B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-2B-Instruct, few shot (both definition)	0,6327	0,2289	0,1540	0,4452
Gemma-2B-Instruct, few shot (false definition)	0,5316	0,4653	0,5759	0,3903
Gemma-2B-Instruct, few shot (none)	0,4921	0,4941	0,7009	0,3815
Gemma-2B-Instruct, few shot (true definition)	0,5024	0,4759	0,6384	0,3793
Gemma-2B-Instruct, zero shot (both definition)	0,6461	0,0088	0,0045	0,5000
Gemma-2B-Instruct, zero shot (false definition)	0,6374	0,0575	0,0312	0,3590
Gemma-2B-Instruct, zero shot (none)	0,5956	0,3333	0,2857	0,4000
Gemma-2B-Instruct, zero shot (true definition)	0,6359	0,0534	0,0290	0,3333

Tabela 6.5: Comparação de Resultados para Detecção de Fake News (Base LIAR, Gemma-7B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct, few shot (both definition)	0,6477	0,2092	0,1317	0,5086
Gemma-7B-Instruct, few shot (false definition)	0,6390	0,2396	0,1607	0,4706
Gemma-7B-Instruct, few shot (none)	0,5908	0,5057	0,5915	0,4417
Gemma-7B-Instruct, few shot (true definition)	0,5656	0,4772	0,5603	0,4156
Gemma-7B-Instruct, zero shot (both definition)	0,6382	0,1763	0,1094	0,4537
Gemma-7B-Instruct, zero shot (false definition)	0,6414	0,1498	0,0893	0,4651
Gemma-7B-Instruct, zero shot (none)	0,5355	0,5157	0,6987	0,4086
Gemma-7B-Instruct, zero shot (true definition)	0,5608	0,5157	0,6607	0,4229

Tabela 6.6: Comparação de Resultados para Detecção de Fake News (Base LIAR, Llama-3-1-70B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Llama-3-1-70B-Instruct, few shot (both definition)	0,6888	0,5365	0,5089	0,5672
Llama-3-1-70B-Instruct, few shot (false definition)	0,6643	0,5614	0,6071	0,5221
Llama-3-1-70B-Instruct, few shot (none)	0,5995	0,5764	0,7701	0,4606
Llama-3-1-70B-Instruct, few shot (true definition)	0,6659	0,5626	0,6071	0,5241
Llama-3-1-70B-Instruct, zero shot (both definition)	0,6643	0,5668	0,6205	0,5216
Llama-3-1-70B-Instruct, zero shot (false definition)	0,6327	0,5706	0,6897	0,4866
Llama-3-1-70B-Instruct, zero shot (none)	0,5877	0,5763	0,7924	0,4528
Llama-3-1-70B-Instruct, zero shot (true definition)	0,6509	0,5606	0,6295	0,5054

6.3 Resultados na Base de dados Fakenewsnet

Tabela 6.7: Comparação de Resultados para Detecção de Fake News (Base FAKENEWS-NET, Gemma-7B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct, few shot (both definition)	0,5984	0,7213	0,9851	0,5690
Gemma-7B-Instruct, few shot (false definition)	0,5669	0,7059	0,9851	0,5500
Gemma-7B-Instruct, few shot (none)	0,5591	0,7053	1.0000	0,5447
Gemma-7B-Instruct, few shot (true definition)	0,5354	0,6943	1.0000	0,5317
Gemma-7B-Instruct, zero shot (both definition)	0,6457	0,7305	0,9104	0,6100
Gemma-7B-Instruct, zero shot (false definition)	0,6299	0,7251	0,9254	0,5962
Gemma-7B-Instruct, zero shot (none)	0,6063	0,7159	0,9403	0,5780
Gemma-7B-Instruct, zero shot (true definition)	0,6693	0,7500	0,9403	0,6238

Tabela 6.8: Comparação de Resultados para Detecção de Fake News (Base FAKENEWS-NET, Llama-3-1-70B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Llama-3-1-70B-Instruct, few shot (both definition)	0,6693	0,7042	0,7463	0,6667
Llama-3-1-70B-Instruct, few shot (false definition)	0,6693	0,7042	0,7463	0,6667
Llama-3-1-70B-Instruct, few shot (none)	0,6378	0,7262	0,9104	0,6040
Llama-3-1-70B-Instruct, few shot (true definition)	0,6457	0,6853	0,7313	0,6447
Llama-3-1-70B-Instruct, zero shot (both definition)	0,6614	0,7075	0,7761	0,6500
Llama-3-1-70B-Instruct, zero shot (false definition)	0,6457	0,6809	0,7164	0,6486
Llama-3-1-70B-Instruct, zero shot (none)	0,6772	0,7389	0,8657	0,6444
Llama-3-1-70B-Instruct, zero shot (true definition)	0,6063	0,6212	0,6119	0,6308

6.3.1 Resultados na Base de dados ISOT

Tabela 6.9: Comparação de Resultados para Detecção de Fake News (Base ISOT, Gemma-7B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct, few shot (both definition)	0,6676	0,7463	0,8704	0,6532
Gemma-7B-Instruct, few shot (false definition)	0,6567	0,7454	0,8948	0,6388
Gemma-7B-Instruct, few shot (none)	0,6184	0,7364	0,9487	0,6017
Gemma-7B-Instruct, few shot (true definition)	0,6392	0,7437	0,9318	0,6188
Gemma-7B-Instruct, zero shot (both definition)	0,6582	0,7530	0,9276	0,6337
Gemma-7B-Instruct, zero shot (false definition)	0,6539	0,7520	0,9343	0,6293
Gemma-7B-Instruct, zero shot (none)	0,6156	0,7382	0,9646	0,5978
Gemma-7B-Instruct, zero shot (none)	0,6567	0,7810	0,9647	0,6560
Gemma-7B-Instruct, zero shot (true definition)	0,6534	0,7519	0,9352	0,6287

Tabela 6.10: Comparação de Resultados para Detecção de Fake News (Base ISOT, Llama-3-1-70B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Llama-3-1-70B-Instruct, few shot (both definition)	0,7366	0,7990	0,9318	0,6993
Llama-3-1-70B-Instruct, few shot (false definition)	0,7329	0,7960	0,9276	0,6970
Llama-3-1-70B-Instruct, few shot (none)	0,6652	0,7659	0,9747	0,6307
Llama-3-1-70B-Instruct, few shot (true definition)	0,7329	0,7934	0,9133	0,7014
Llama-3-1-70B-Instruct, zero shot (both definition)	0,7518	0,8037	0,9049	0,7229
Llama-3-1-70B-Instruct, zero shot (false definition)	0,7461	0,7915	0,8577	0,7347
Llama-3-1-70B-Instruct, zero shot (none)	0,6648	0,7434	0,8645	0,6521
Llama-3-1-70B-Instruct, zero shot (true definition)	0,7173	0,7500	0,7551	0,7450

6.3.2 Resultados na Base de dados POLITIFACT

Tabela 6.11: Comparação de Resultados para Detecção de Fake News (Base POLITIFACT FACTCHECK DATA, Gemma-7B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct, few shot (both definition)	0,4754	0,4299	0,7217	0,3061
Gemma-7B-Instruct, few shot (false definition)	0,5474	0,4406	0,6504	0,3331
Gemma-7B-Instruct, few shot (none)	0,6231	0,4882	0,6561	0,3888
Gemma-7B-Instruct, few shot (true definition)	0,3746	0,4508	0,9367	0,2968
Gemma-7B-Instruct, zero shot (both definition)	0,6262	0,4655	0,5940	0,3827
Gemma-7B-Instruct, zero shot (false definition)	0,6371	0,4479	0,5371	0,3840
Gemma-7B-Instruct, zero shot (none)	0,5400	0,4842	0,7878	0,3495
Gemma-7B-Instruct, zero shot (none)	0,5357	0,4832	0,7899	0,3481
Gemma-7B-Instruct, zero shot (true definition)	0,5080	0,4863	0,8499	0,3406

Tabela 6.12: Comparação de Resultados para Detecção de Fake News (Base POLITIFACT FACTCHECK DATA, Llama-3-1-70B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Llama-3-1-70B-Instruct, few shot (both definition)	0,7309	0,5148	0,5210	0,5087
Llama-3-1-70B-Instruct, few shot (false definition)	0,7228	0,5335	0,5785	0,4951
Llama-3-1-70B-Instruct, few shot (none)	0,6754	0,5608	0,7562	0,4456
Llama-3-1-70B-Instruct, few shot (true definition)	0,7157	0,5563	0,6504	0,4860
Llama-3-1-70B-Instruct, zero shot (both definition)	0,7249	0,4822	0,4675	0,4979
Llama-3-1-70B-Instruct, zero shot (false definition)	0,7148	0,5134	0,5492	0,4821
Llama-3-1-70B-Instruct, zero shot (none)	0,6673	0,5394	0,7108	0,4346
Llama-3-1-70B-Instruct, zero shot (true definition)	0,7149	0,4926	0,5049	0,4808

6.3.3 Resultados na Base de dados Welfake

Tabela 6.13: Comparação de Resultados para Detecção de Fake News (Base WELFAKE, Gemma-7B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct, few shot (none)	0,5050	0,6711	0,9962	0,5060
Gemma-7B-Instruct, zero shot (both definition)	0,3513	0,5130	0,6706	0,4154
Gemma-7B-Instruct, zero shot (false definition)	0,3668	0,5322	0,7068	0,4268
Gemma-7B-Instruct, zero shot (none)	0,4717	0,6397	0,8488	0,5133
Gemma-7B-Instruct, zero shot (none)	0,4380	0,6056	0,8469	0,4714
Gemma-7B-Instruct, zero shot (true definition)	0,3755	0,5415	0,7235	0,4326

Tabela 6.14: Comparação de Resultados para Detecção de Fake News (Base WELFAKE, Llama-3-1-70B-Instruct)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Llama-3-1-70B-Instruct, few shot (both definition)	0,1974	0,3089	0,3520	0,2752
Llama-3-1-70B-Instruct, few shot (false definition)	0,2396	0,3741	0,4460	0,3222
Llama-3-1-70B-Instruct, few shot (none)	0,4180	0,5866	0,8103	0,4597
Llama-3-1-70B-Instruct, few shot (true definition)	0,1645	0,2431	0,2633	0,2258
Llama-3-1-70B-Instruct, zero shot (both definition)	0,1755	0,2729	0,3037	0,2478
Llama-3-1-70B-Instruct, zero shot (false definition)	0,1534	0,2347	0,2547	0,2175
Llama-3-1-70B-Instruct, zero shot (none)	0,3426	0,4875	0,6135	0,4044
Llama-3-1-70B-Instruct, zero shot (true definition)	0,1403	0,1830	0,1889	0,1774

6.3.4 Resultados DeepSeek

Nesta subseção, apresentamos os resultados dos experimentos realizados com o modelo **DeepSeek-R1 Distill Qwen 32B**, uma versão densa baseada na arquitetura Qwen e otimizada para tarefas de raciocínio. Os testes foram conduzidos sobre quatro bases de dados amplamente utilizadas no contexto de detecção de fake news: **LIAR**, **COVID**, **PolitiFact** e **WeLFake**.

Devido a limitações orçamentárias, não foi possível executar experimentos completos em todas as bases de dados e com todos os tipos de prompts utilizados nas demais etapas deste trabalho. Para tornar os experimentos viáveis dentro das restrições de custo, foi definida uma amostragem de **1000 notícias por base de dados** como critério fixo de avaliação. Com isso, buscamos obter uma visão representativa do desempenho do modelo DeepSeek, ainda que com cobertura parcial dos dados. A tabela 6.15 exibe os custos por hora para o modelo Deepseek utilizado na Amazon.

Tabela 6.15: Instâncias utilizadas para deployment dos endpoints Deepseek

Modelo	**Instância AWS**	**Custo Médio por Hora (USD)**
DeepSeek-R1 Distill Qwen 32B	ml.p4d.24xlarge	32,77

A seguir, apresentamos as tabelas e gráficos com os **resultados obtidos**, considerando as métricas de acurácia, precisão, recall e F1-score para os diferentes prompts e conjuntos de dados utilizados.

Tabela 6.16: Resultados do modelo DeepSeek-R1 Distill Qwen 32B na base **COVID** com diferentes prompts (zero-shot)

Prompt	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Zero-shot (false definition)	0,719	0,7064	0,6248	0,8125
Zero-shot (none)	0,767	0,7841	0,7819	0,7862
Zero-shot (true definition)	0,7809	0,8055	0,7538	0,8647

Figura 6.1: Matriz de confusão dos resultados deepsek na base covid zero shot false definition

Tabela 6.17: Resultados do modelo DeepSeek-R1 Distill Qwen 32B na base **Fake News Net** com diferentes prompts

Prompt	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Few-shot (both definition)	0,6535	0,6857	0,7164	0,6575
Few-shot (false definition)	0,6220	0,6471	0,6567	0,6377
Few-shot (none definition)	0,6378	0,6933	0,7761	0,6265
Few-shot (true definition)	0,6457	0,6763	0,7015	0,6528
Zero-shot (both definition)	0,6457	0,6617	0,6567	0,6667
Zero-shot (false definition)	0,6535	0,6508	0,6119	0,6949
Zero-shot (none definition)	0,6457	0,6980	0,7761	0,6341
Zero-shot (true definition)	0,6614	0,6767	0,6716	0,6818

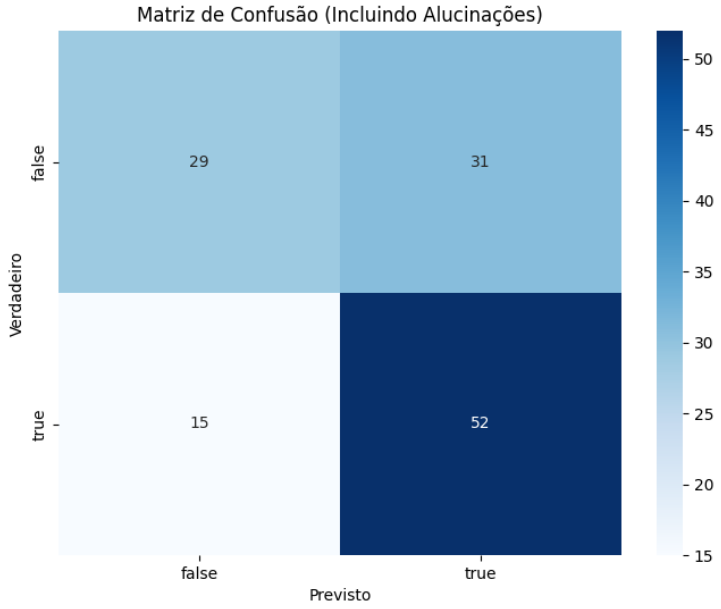


Figura 6.2: Matriz de confusão dos resultados deepsek na base fake news net few shot none

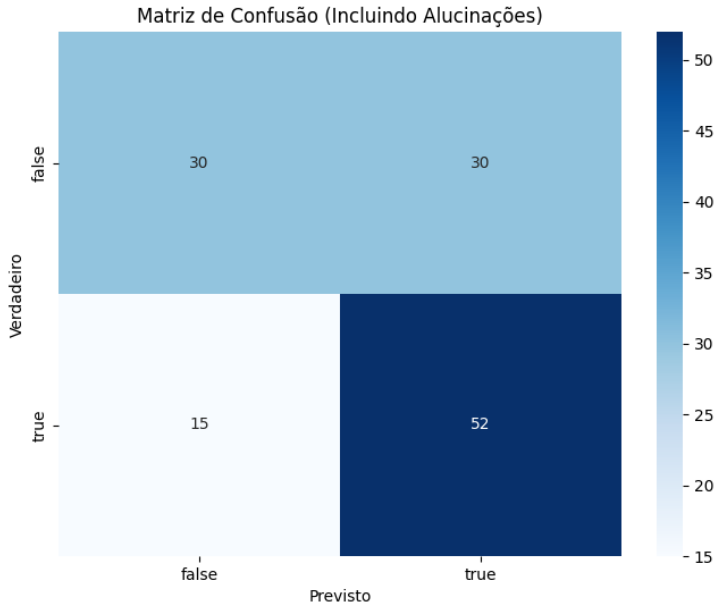


Figura 6.3: Matriz de confusão dos resultados deepsek na base fake news net zero shot none

Tabela 6.18: Resultados do modelo DeepSeek-R1 Distill Qwen 32B na base **LIAR** com diferentes prompts (zero-shot)

Prompt	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Zero-shot (false definition)	0,6158	0,2994	0,2248	0,4483
Zero-shot (none)	0,6100	0,4055	0,3684	0,4508

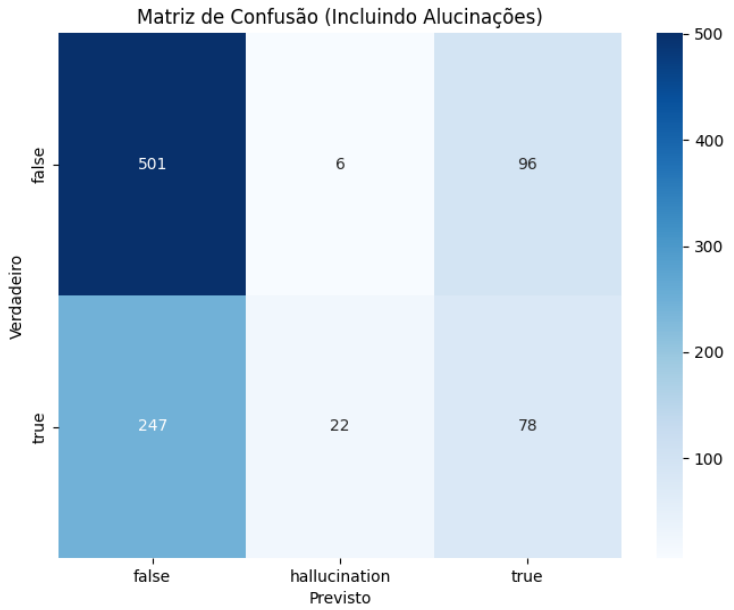


Figura 6.4: Matriz de confusão dos resultados deepsek na base liar net zero shot false definition

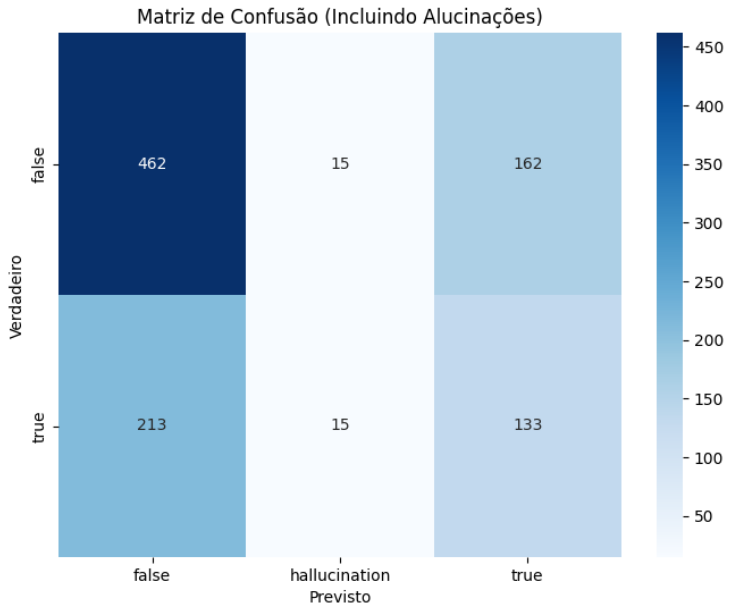


Figura 6.5: Matriz de confusão dos resultados deepsek na base liar net zero shot none

Tabela 6.19: Resultados do modelo DeepSeek-R1 Distill Qwen 32B na base **WeL-Fake** com diferentes prompts (zero-shot)

Prompt	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Zero-shot (false definition)	0,2120	0,2423	0,2400	0,2447
Zero-shot (none)	0,2790	0,3926	0,4438	0,3520
Zero-shot (true definition)	0,2017	0,2646	0,2613	0,2680

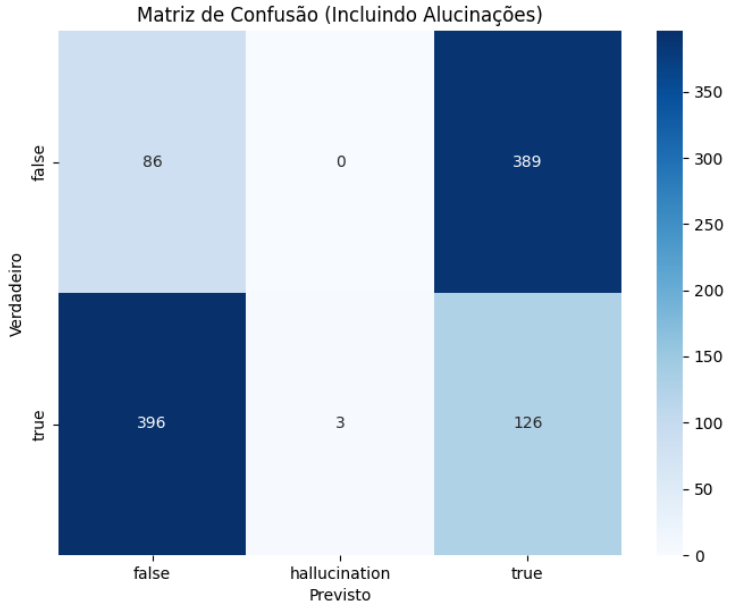


Figura 6.6: Matriz de confusão dos resultados deepsek na base welfake zero shot false definition

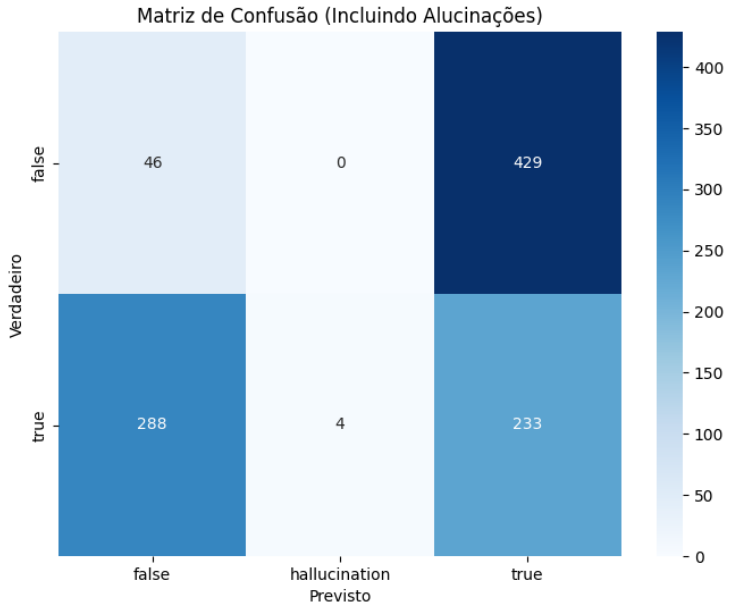


Figura 6.7: Matriz de confusão dos resultados deepsek na base welfake zero shot none

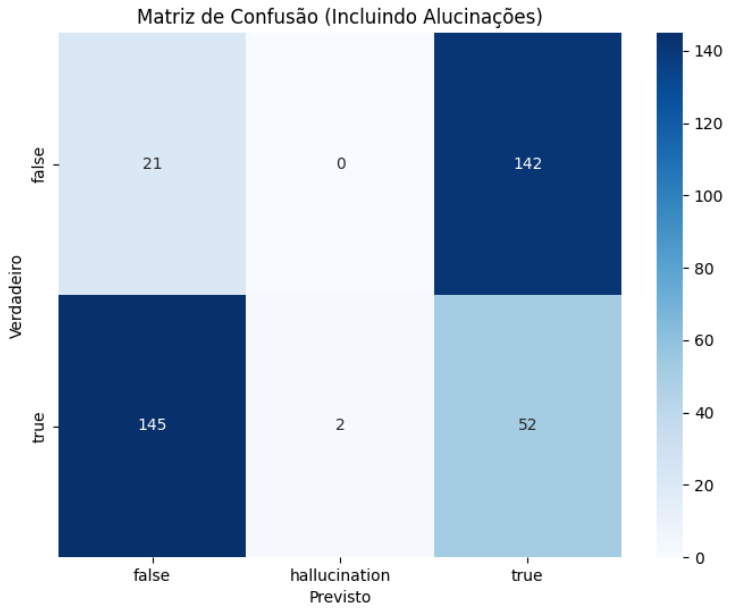


Figura 6.8: Matriz de confusão dos resultados deepsek na base welfake zero shot true definition

Apesar de não ser um modelo do tipo Instruct, o DeepSeek-R1 Distill Qwen 32B apresentou uma taxa relativamente baixa de alucinações nas respostas, conforme evidenciado na Figura 6.1. Comparando com o melhor resultado obtido com o modelo LLaMA 3.1 70B Instruct no cenário zero-shot — que alcançou um F1 Score de 0,8831 — o melhor desempenho do DeepSeek no mesmo cenário, com F1 Score de 0,8055, mostra que o modelo é competitivo, mesmo sem treinamento específico para seguir instruções.

Conclusões sobre o Modelo DeepSeek-R1 Distill Qwen 32B

Esta subseção apresenta uma análise consolidada dos resultados obtidos pelo modelo *DeepSeek-R1 Distill Qwen 32B* nas bases de dados **Fake News Net**, **LIAR** e **WeLFake**, considerando diferentes configurações de *prompts*, especialmente no modo zero-shot e few-shot.

Na base **Fake News Net**, o modelo demonstrou desempenho relativamente consistente e equilibrado, com acurácia variando entre 62,20% e 66,14%. As configurações *zero-shot (true definition)* e *zero-shot (none definition)* destacaram-se, apresentando um bom compromisso entre as métricas, especialmente em termos de F1 Score (até 0,6980) e recall elevado (acima de 0,77 nas configurações sem definição). A precisão também se manteve competitiva, indicando que o modelo é capaz de identificar a maioria das fake news presentes nesta base com uma taxa razoável de acertos.

Em contrapartida, na base **LIAR**, o desempenho foi mais modesto, com acurácia próxima de 61% em ambas as configurações zero-shot avaliadas. O F1 Score revelou dificuldades maiores no equilíbrio entre precisão e recall, com valores entre 0,2994 e 0,4055, indicando que o modelo apresenta limitações para detectar as fake news neste conjunto de dados. Apesar disso, a precisão manteve-se em níveis moderados (45%), sugerindo que, quando o modelo classifica uma notícia como falsa, há uma probabilidade razoável de acerto. Novamente, o prompt sem definição (*none*) proporcionou uma leve vantagem no desempenho geral.

Finalmente, na base **WeLFake**, os resultados indicaram um desempenho mais restrito, com acurácia entre 20,17% e 27,90%, evidenciando dificuldades do modelo na tarefa de detecção nesta base. O prompt sem definição (*none*) também foi o que obteve melhor desempenho, com F1 Score de 0,3926 e valores superiores de recall e precisão em relação às demais configurações. Os prompts contendo definições, tanto falsas quanto verdadeiras, não apresentaram melhora significativa, sugerindo que as instruções fornecidas não foram eficazes para este conjunto de dados.

Em suma, o modelo *DeepSeek-R1 Distill Qwen 32B* mostra um desempenho consistente e equilibrado na base Fake News Net, moderado na base LIAR e limitado na base WeLFake. A ausência de definições explícitas nos prompts tende a favorecer resultados ligeiramente melhores nas bases LIAR e WeLFake, enquanto na Fake News Net as definições verdadeiras podem contribuir positivamente. Esses achados indicam que o ajuste do prompt é um fator relevante para o desempenho do modelo, mas que também existem limitações intrínsecas dependendo da base de dados utilizada, reforçando a necessidade de melhorias específicas para bases mais desafiadoras como WeLFake e LIAR. Além disso, o *DeepSeek-R1 Distill Qwen 32B* posiciona-se como um modelo intermediário entre o *Gemma-7B* e o *Llama-3-70B* em termos de custo computacional e quantidade de parâmetros. No entanto, considerando sua relação custo-desempenho, o *Gemma-7B* se apresenta como uma opção mais atrativa, uma vez que o *DeepSeek-R1* requer uma instância que custa aproximadamente 32 dólares por hora, enquanto o *Gemma-7B* opera em uma instância consideravelmente mais acessível, com custo inferior a 7 dólares por hora, mantendo resultados competitivos.

6.3.5 Avaliação de tempo e custo do Fine-Tuning

Para avaliar o custo computacional e financeiro do processo de fine-tuning, medimos o tempo total de treinamento para cada combinação de modelo e base de dados, utilizando instâncias específicas da AWS adaptadas ao porte de cada modelo.

O modelo **GEMMA 7B Instruct** foi treinado na instância `ml.g5.24xlarge`, enquanto o modelo **LLaMA 3.1 70B Instruct** foi treinado na instância `ml.g5.48xlarge`. As Tabelas 6.20 e 6.21 apresentam os tempos de execução medidos

em segundos.

Tabela 6.20: Tempo de Fine-Tuning para o Modelo GEMMA

Base	**Instância AWS**	**Tempo (segundos)**
COVID	ml.g5.24xlarge	5.136
LIAR	ml.g5.24xlarge	6.731

Tabela 6.21: Tempo de Fine-Tuning para o Modelo LLAMA

Base	**Instância AWS**	**Tempo (segundos)**
COVID	ml.g5.48xlarge	10.911
LIAR	ml.g5.48xlarge	13.234

As instâncias utilizadas possuem os seguintes custos, considerando o modelo de preços on-demand na região `us-east-1` (N. Virginia):

- **ml.g5.24xlarge**: US$ 8,144 por hora.
- **ml.g5.48xlarge**: US$ 16,288 por hora.

A partir dos tempos de execução, estimamos o custo aproximado de cada experimento, conforme mostrado abaixo:

- **GEMMA - Base COVID**: aproximadamente US$ 11,65.
- **GEMMA - Base LIAR**: aproximadamente US$ 15,23.
- **LLaMA - Base COVID**: aproximadamente US$ 49,29.
- **LLaMA - Base LIAR**: aproximadamente US$ 59,99.

Esses valores demonstram que o fine-tuning de grandes modelos, mesmo utilizando técnicas de adaptação eficientes como o LoRA, envolve um custo financeiro significativo. Ressaltamos que esses valores referem-se a apenas uma execução por base, enquanto no projeto completo foram realizados múltiplos experimentos (variações de prompts), aumentando ainda mais o custo total do estudo.

6.3.6 Resultados do Fine Tuning

O processo de fine-tuning realizado nesta pesquisa buscou avaliar o impacto da especialização supervisionada dos modelos nas bases de dados COVID e LIAR. As bases selecionadas apresentam características distintas, sendo a COVID composta

por textos de tamanho moderado e a LIAR por entradas textuais bastante curtas. Essa diferença de perfil é relevante, pois influencia diretamente a capacidade dos modelos de linguagem em compreender e classificar as informações. A seguir, são apresentados os resultados obtidos após o fine-tuning.

Tabela 6.22: Comparação de Resultados para Detecção de Fake News (Base COVID, Gemma-7B-Instruct-fine-tuned)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct-fine-tuned, few shot (false definition)	0,9070	0,9176	0,9893	0,8556
Gemma-7B-Instruct-fine-tuned, few shot (none)	0,8154	0,8495	0,9955	0,7409
Gemma-7B-Instruct-fine-tuned, few shot (true definition)	0,9048	0,9187	1.0000	0,8496
Gemma-7B-Instruct-fine-tuned, zero shot (both definition)	0,9318	0,9381	0,9875	0,8934
Gemma-7B-Instruct-fine-tuned, zero shot (false definition)	0,9350	0,9404	0,9795	0,9044
Gemma-7B-Instruct-fine-tuned, zero shot (none)	0,8827	0,8983	0,9893	0,8226
Gemma-7B-Instruct-fine-tuned, zero shot (true definition)	0,9308	0,9373	0,9884	0,8913

Tabela 6.23: Comparação de Resultados para Detecção de Fake News (Base COVID, Llama-3-1-70B-Instruct-fine-tuned)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Llama-3-1-70B-Instruct-fine-tuned, few shot (both definition)	0,9393	0,9431	0,9625	0,9245
Llama-3-1-70B-Instruct-fine-tuned, few shot (false definition)	0,9327	0,9365	0,9482	0,9251
Llama-3-1-70B-Instruct-fine-tuned, few shot (none)	0,9341	0,9394	0,9759	0,9056
Llama-3-1-70B-Instruct-fine-tuned, few shot (true definition)	0,9318	0,9364	0,9598	0,9141
Llama-3-1-70B-Instruct-fine-tuned, zero shot (both definition)	0,9318	0,9372	0,9732	0,9038
Llama-3-1-70B-Instruct-fine-tuned, zero shot (false definition)	0,9262	0,9304	0,9429	0,9183
Llama-3-1-70B-Instruct-fine-tuned, zero shot (none)	0,9023	0,9142	0,9938	0,8464
Llama-3-1-70B-Instruct-fine-tuned, zero shot (true definition)	0,9257	0,9315	0,9652	0,9001

Tabela 6.24: Comparação de Resultados para Detecção de Fake News (Base LIAR, Gemma-7B-Instruct-fine-tuned)

Config	**Acurácia**	**F1 Score**	**Recall**	**Precisão**
Gemma-7B-Instruct-fine-tuned, few shot (both definition)	0,5806	0,4663	0,5179	0,4241
Gemma-7B-Instruct-fine-tuned, few shot (false definition)	0,5893	0,4790	0,5335	0,4345
Gemma-7B-Instruct-fine-tuned, few shot (none)	0,5782	0,5154	0,6339	0,4343
Gemma-7B-Instruct-fine-tuned, few shot (true definition)	0,5972	0,4950	0,5580	0,4448
Gemma-7B-Instruct-fine-tuned, zero shot (both definition)	0,6177	0,5527	0,6674	0,4716
Gemma-7B-Instruct-fine-tuned, zero shot (false definition)	0,6303	0,5585	0,6607	0,4837
Gemma-7B-Instruct-fine-tuned, zero shot (none)	0,6027	0,5630	0,7232	0,4609
Gemma-7B-Instruct-fine-tuned, zero shot (true definition)	0,6532	0,5675	0,6429	0,5079

Tabela 6.25: Comparação de Resultados para Detecção de Fake News (Base LIAR, Llama-3-1-70B-Instruct-fine-tuned)

Config	Acurácia	F1 Score	Recall	Precisão
Llama-3-1-70B-Instruct-fine-tuned, few shot (both definition)	0,6659	0,6253	0,7879	0,5184
Llama-3-1-70B-Instruct-fine-tuned, few shot (false definition)	0,6169	0,6098	0,8460	0,4767
Llama-3-1-70B-Instruct-fine-tuned, few shot (none)	0,5869	0,6041	0,8906	0,4570
Llama-3-1-70B-Instruct-fine-tuned, few shot (true definition)	0,6445	0,6237	0,8326	0,4987
Llama-3-1-70B-Instruct-fine-tuned, zero shot (both definition)	0,6817	0,2967	0,1897	0,6800
Llama-3-1-70B-Instruct-fine-tuned, zero shot (false definition)	0,7062	0,4329	0,3170	0,6827
Llama-3-1-70B-Instruct-fine-tuned, zero shot (none)	0,6975	0,4505	0,3504	0,6305
Llama-3-1-70B-Instruct-fine-tuned, zero shot (true definition)	0,7117	0,5338	0,4665	0,6239

A análise dos resultados indica que, embora tenha havido uma melhora modesta no desempenho dos modelos em ambas as bases, o desempenho na base LIAR permaneceu inferior ao esperado, não ultrapassando a marca de 70% de acurácia. Tal limitação pode ser atribuída à natureza da base, que se caracteriza por instâncias textuais extremamente curtas, como evidenciado pela análise do número médio de tokens por classe (Figura 5.1c). Essa limitação de contexto prejudica a capacidade dos modelos de extrair informações suficientes para realizar uma classificação precisa. Dessa forma, mesmo com a aplicação de técnicas de fine-tuning, os modelos apresentaram dificuldade em superar essa restrição estrutural da base de dados.

Os F1 Scores para fine-tuning e não fine-tuning são: **LIAR** – 0,5764 (com) e 0,6253 (sem); **COVID** – 0,9431 (com) e 0,9150 (sem). O fine-tuning melhora o desempenho na base COVID, mas tem impacto limitado na LIAR, possivelmente devido à falta de contexto nas entradas. Acreditamos que o baixo desempenho na base LIAR ocorre porque a classificação é feita apenas com o texto da notícia, sem informações adicionais, e a média de tokens nessa base é muito pequena, como mostrado na 5.1c. Isso reduz o contexto disponível para o LLM, dificultando a tarefa e limitando os ganhos obtidos mesmo com o fine-tuning.

6.4 Análise de Desempenho e Tempo de Execução dos Modelos de LLMs

Para avaliar o desempenho dos *LLMs*, é essencial medir o tempo de execução em função da quantidade de tokens processados pelo modelo. Neste estudo, utilizamos uma métrica de tempo de processamento calculada pela diferença de tempo entre as execuções consecutivas de predições, em segundos. A análise desse tempo de processamento é crucial para entender como a carga de trabalho, medida pelo número de tokens de entrada, influencia o desempenho do modelo.

- **Cálculo do Tempo de Processamento:** O tempo de processamento foi

calculado para cada amostra de entrada com base no número de tokens e na diferença temporal entre execuções consecutivas. Para cada intervalo de *input tokens*, calculamos a média, desvio padrão, valor mínimo e máximo do tempo de processamento.

- **Uso de Histogramas para Agrupamento:** Para agrupar os dados e calcular as métricas de tempo de processamento, optamos por utilizar histogramas. Agrupamos os valores de *input tokens* em intervalos (bins) para capturar o comportamento do modelo em diferentes faixas de quantidade de tokens. Esse método é adequado quando a distribuição dos dados não segue uma estrutura uniforme, como ocorre com os tokens de entrada dos modelos de LLMs, onde valores extremos podem ocorrer esparsamente.

- **Intervalos Não Fixos:** Em vez de usar intervalos fixos de tamanho constante (por exemplo, intervalos de 10 tokens), utilizamos intervalos dinâmicos e adaptativos. Isso é necessário para evitar distorções causadas pela dispersão dos dados. Frequentemente, a maioria dos dados se concentra em faixas de tokens mais baixos, enquanto alguns exemplos possuem um número muito alto de tokens. O uso de intervalos fixos resultaria em faixas vazias ou com poucos dados, o que poderia levar à perda de informações e à introdução de vieses na análise.

- **Intervalos Adaptativos:** Para garantir uma análise representativa, criamos intervalos adaptativos, ajustados de acordo com a distribuição dos dados de *input tokens*. Essa abordagem permite capturar de maneira mais precisa as variações no tempo de processamento entre diferentes faixas de tokens, considerando tanto os valores pequenos quanto os grandes de forma equilibrada.

Na análise do desempenho dos modelos, observamos que o tempo de processamento permanece relativamente constante, mesmo com variações significativas no número de tokens de entrada. Isso é evidenciado pelos gráficos apresentados nas Figuras 6.9 e 6.10, que mostram a média do tempo de processamento em função do número de tokens de entrada.

No gráfico referente ao dataset *Welfake* (Figura 6.10), é possível observar que, em média, o tempo de processamento por notícia é de aproximadamente 2 segundos. Isso ocorre, pois o número de tokens por notícia nesse dataset é muito maior, chegando a até 8.000 tokens, enquanto no *PolitiFact*, o máximo era de 80 tokens. Embora tenha ocorrido um aumento de 10 vezes na quantidade de tokens de entrada, o tempo de processamento dobrou, o que indica que o modelo já estava operando próximo ao seu limite de capacidade (cerca de 8.100 tokens por entrada). Esse limite impede

um aumento considerável no tempo de processamento, mesmo com o crescimento substancial no número de tokens.

Vale ressaltar que, para notícias com mais tokens do que a capacidade de entrada do modelo, os tokens excedentes são truncados. Isso significa que, quando uma notícia ultrapassa o limite máximo de tokens, apenas a parte inicial da notícia é processada, o que pode influenciar o conteúdo analisado, mas garante que o modelo consiga lidar com entradas dentro dos limites estabelecidos.

Portanto, apesar do aumento na quantidade de tokens de entrada, o modelo não apresentou uma piora significativa no tempo de processamento, demonstrando sua capacidade de lidar com entradas maiores até o limite de tokens. Esse comportamento sugere que o modelo já está otimizado para trabalhar eficientemente com as entradas de maior tamanho, sem um impacto drástico no tempo de execução.

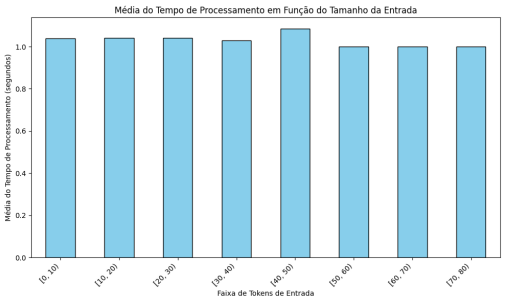


Figura 6.9: Gráfico de barras mostrando a média do tempo de processamento em função do número de tokens de entrada para o dataset Politifact.

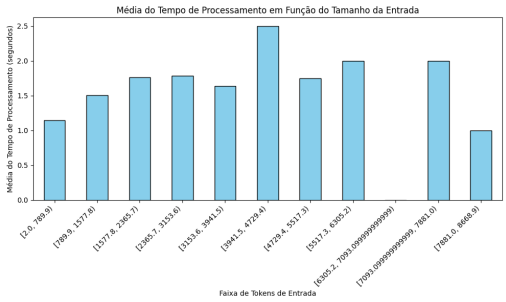


Figura 6.10: Gráfico de barras mostrando a média do tempo de processamento em função do número de tokens de entrada para o dataset Welfake.

6.4.1 Métrica PoC-score

Para complementar a análise de desempenho tradicional dos modelos, propomos a métrica *PoC-score*, que quantifica a relação entre desempenho e eficiência econômica dos modelos. O *PoC-score* é calculado como a razão entre o F1-score obtido pelo modelo e seu custo operacional por hora (USD/h), conforme expressa a fórmula:

$$PoC_{\text{score}} = \frac{\text{F1-score}}{\text{Custo por hora (USD/h)}}$$

Essa métrica incentiva a adoção de modelos custo-eficientes, penalizando soluções que, embora apresentem alto desempenho, possuam custos operacionais elevados. Além disso, ela contextualiza o desempenho ao permitir a comparação entre modelos com F1-score semelhante, mas com custos significativamente diferentes,

evidenciando cenários em que modelos ligeiramente menos precisos podem ser preferíveis devido ao melhor custo-benefício.

O *PoC-score* também reflete uma preocupação prática alinhada aos objetivos organizacionais, considerando que, em aplicações reais, restrições como orçamento, tempo de inferência ou impacto econômico são tão relevantes quanto a acurácia. Por fim, trata-se de uma métrica adaptável a múltiplos domínios, podendo ser ajustada para incorporar diferentes tipos de custos, como consumo de GPU, tempo de processamento, ou até o custo indireto de análises manuais decorrentes de falsos positivos.

No entanto, a utilização do *PoC-score* também apresenta limitações importantes. O primeiro desafio reside na própria definição do custo, que pode assumir múltiplas dimensões: monetária, computacional, energética ou até humana. Essa ambiguidade impacta diretamente a comparabilidade dos resultados, uma vez que diferentes trabalhos podem adotar critérios distintos para calcular o custo, comprometendo a reprodutibilidade e a padronização.

Além disso, a métrica pode mascarar deficiências no desempenho absoluto dos modelos. Um modelo com F1 baixo pode, aparentemente, apresentar um bom *PoC-score* se seu custo for suficientemente reduzido, o que pode levar a interpretações equivocadas ou otimizações excessivamente focadas na métrica, em detrimento da qualidade do modelo. Isso é evidente, por exemplo, no desempenho do modelo Gemma-2B, que, apesar de apresentar resultados de F1 significativamente inferiores ao Gemma-7B em praticamente todas as bases, acaba figurando com um *PoC-score* bastante competitivo devido ao seu custo extremamente baixo. Esse tipo de situação ilustra claramente como a métrica pode favorecer modelos pouco precisos quando o custo se torna o fator dominante.

Outro ponto crítico é que, diferentemente de métricas tradicionais como o F1-score, que possuem uma escala bem definida e interpretável (de 0 a 1), o *PoC-score* não é simétrico nem padronizado, dependendo diretamente da escala do custo adotado. Isso dificulta sua interpretação direta e torna sua aplicação mais sensível ao contexto específico.

Os gráficos da figura 6.11 apresentam os melhores resultados do *PoC-score* para os diferentes modelos avaliados em cada base de dados, proporcionando uma comparação clara entre desempenho e eficiência econômica na tarefa de detecção de *fake news*.

Comparação entre o PoC-score e o C-Score

Visando ampliar a análise de custo-benefício no problema de detecção de *fake news*, além do *PoC-score*, este trabalho também adota o *C-Score*, proposto por MARWAH *et al.* (2024). Essa métrica foi projetada para problemas de classificação

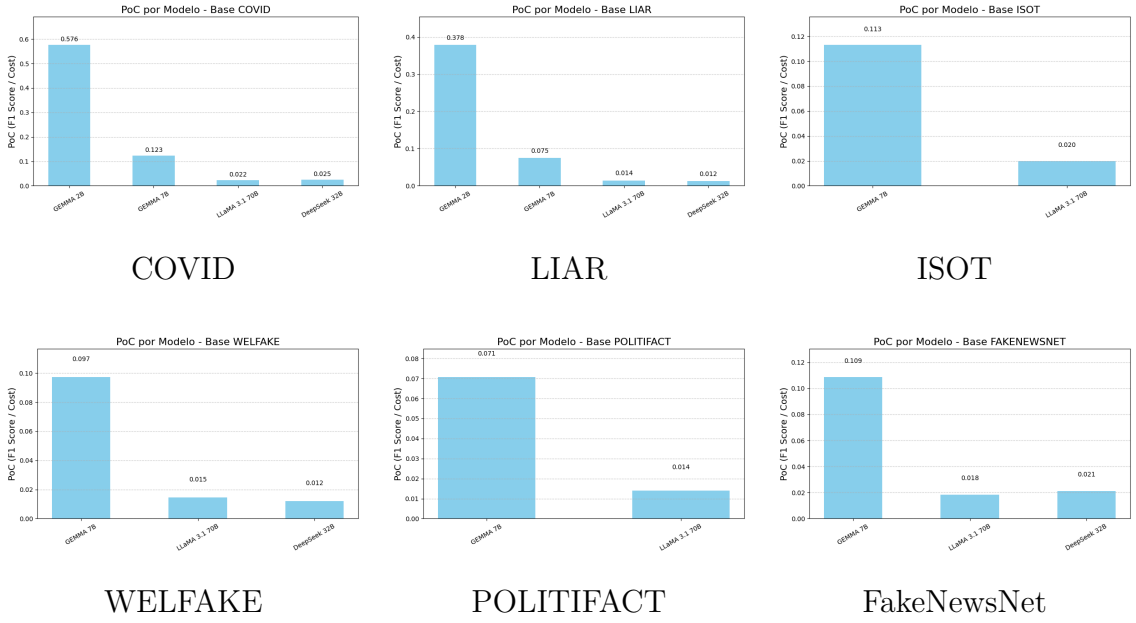


Figura 6.11: Comparação do PoC-score dos modelos em diferentes bases de dados de detecção de *fake news*.

com custos assimétricos, nos quais os impactos de *falsos positivos* (FP) e *falsos negativos* (FN) podem ter pesos distintos.

O *C-Score* é calculado por meio da seguinte fórmula:

$$CScore = \left(\frac{1}{Precision} - 1 - rc \right) \cdot Recall + rc \tag{6.1}$$

onde $Precision$ e $Recall$ são as métricas tradicionais, e rc (*cost ratio*) representa a razão entre o custo de um FN e o custo de um FP:

$$rc = \frac{Custo_{FN}}{Custo_{FP}} \tag{6.2}$$

Em contextos práticos, definir rc é um desafio, pois depende da priorização dos impactos que cada tipo de erro pode gerar. Por exemplo, classificar uma notícia falsa como verdadeira (FN) pode ter consequências diferentes de classificar uma verdadeira como falsa (FP), dependendo do domínio, aplicação ou contexto social e econômico.

Neste trabalho, adota-se inicialmente a hipótese simplificadora de que ambos os erros possuem o mesmo peso, isto é:

$$rc = 1 \tag{6.3}$$

Essa escolha permite uma comparação direta entre o *PoC-score* e o *C-Score* sem a introdução de vieses decorrentes da atribuição de pesos diferenciados entre FP e FN. Apesar disso, reconhece-se que essa suposição pode não refletir todos os cenários

do mundo real, e futuros trabalhos podem explorar diferentes valores de rc conforme o domínio.

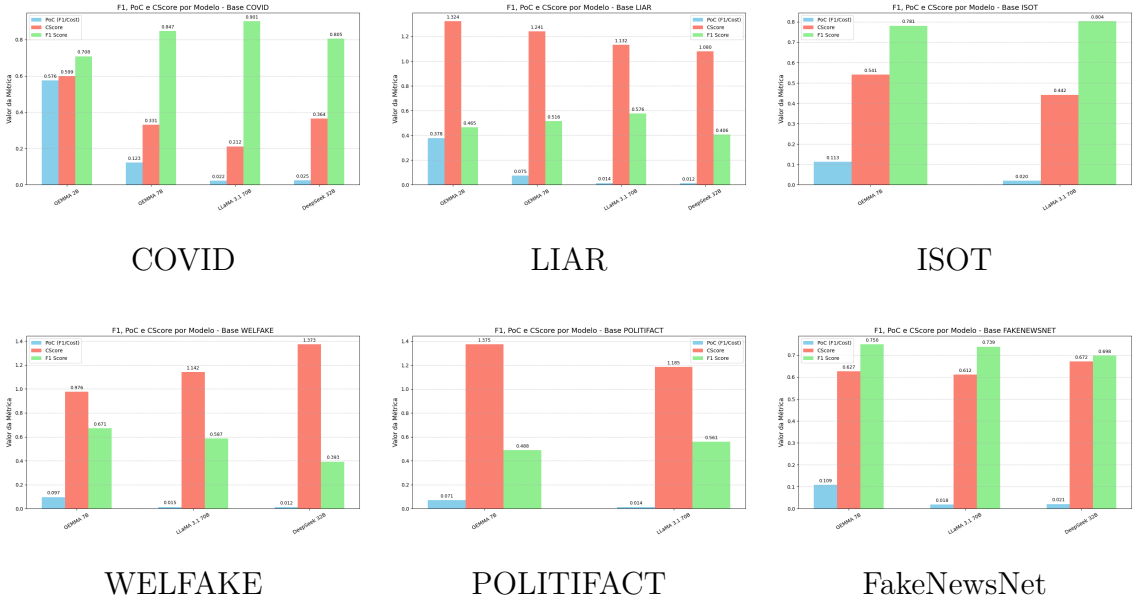
Vale destacar que, diferentemente do *PoC-score*, o *C-Score* não é uma métrica normalizada entre 0 e 1, e sua interpretação é menos intuitiva. Além disso, o *C-Score* pode assumir valores negativos, especialmente quando a precisão do modelo é inferior a $(1 + rc)$. Isso reflete uma característica intrínseca da métrica, que prioriza fortemente o equilíbrio entre precisão e recall, ponderado pelo custo relativo.

Por fim, ambas as métricas compartilham limitações comuns na quantificação do custo, como:

- **Dificuldade na definição do custo:** Determinar se o custo deve ser monetário, computacional, energético ou social é uma decisão que impacta diretamente a métrica.
- **Comparabilidade limitada:** Modelos avaliados sob diferentes definições de custo não são comparáveis diretamente, o que compromete a padronização e reprodutibilidade.
- **Risco de otimizações oportunistas:** Existe a possibilidade de manipulação dos parâmetros de custo para inflar a métrica, sem refletir melhorias reais no modelo.
- **Perda de simetria e normalização:** Diferente de métricas como F1, que variam entre 0 e 1, tanto o *PoC-score* quanto o *C-Score* dependem da escala escolhida para o custo, dificultando interpretações diretas.
- **Mascaramento de deficiências no desempenho:** Um modelo com F1 baixo pode apresentar um *PoC-score* ou *C-Score* aparentemente atrativo se o custo for suficientemente baixo. Por exemplo, observa-se que o modelo Gemma-2B, embora apresente um desempenho significativamente inferior ao Gemma-7B em termos absolutos de F1, atinge um *PoC-score* bastante competitivo devido ao seu custo operacional extremamente reduzido.

Apesar dessas limitações, a adoção de métricas custo-sensíveis, como o *PoC-score* e o *C-Score*, representa um avanço na avaliação prática de modelos, especialmente em cenários onde os recursos são finitos e o custo operacional é um fator determinante na escolha da solução.

A Figura 6.12 apresenta uma comparação entre o *C-score* e o *PoC-score* dos melhores resultados em cada base, incluindo também o *F1-score* bruto, de forma a considerar não apenas a relação custo-benefício dos modelos, mas também seu desempenho absoluto.



COVID LIAR ISOT

WELFAKE POLITIFACT FakeNewsNet

Figura 6.12: Comparação do PoC-score dos modelos em diferentes bases de dados de detecção de *fake news*.

6.4.2 Desempenho dos Resultados do Fine-Tuning

Os resultados obtidos indicam que o processo de fine-tuning dos modelos GEMMA 7B Instruct e LLaMA 3.1 70B Instruct proporciona ganhos de desempenho mensuráveis. Observa-se que, para a base COVID, o fine-tuning elevou o F1-score de 0,9150 para 0,9431 no modelo avaliado, enquanto para a base LIAR houve um incremento do F1-score de 0,5764 para 0,6253.

Esses incrementos, apesar de parecerem modestos, representam avanços importantes em cenários reais, onde pequenas melhorias na detecção de *fake news* podem impactar significativamente a qualidade das análises e a confiabilidade dos sistemas.

Contudo, tais melhorias vêm acompanhadas de um custo computacional e financeiro considerável, conforme evidenciado pelos tempos de treinamento e custos estimados para cada modelo e base. O trade-off entre custo e benefício deve ser cuidadosamente avaliado em ambientes de produção, especialmente para modelos de grande porte como o LLaMA 3.1 70B. A figura 6.13 ilustram o PoC obtido para os experimentos de *fine-tuning* nas bases COVID e LIAR, permitindo uma comparação visual clara da eficiência dos investimentos realizados.

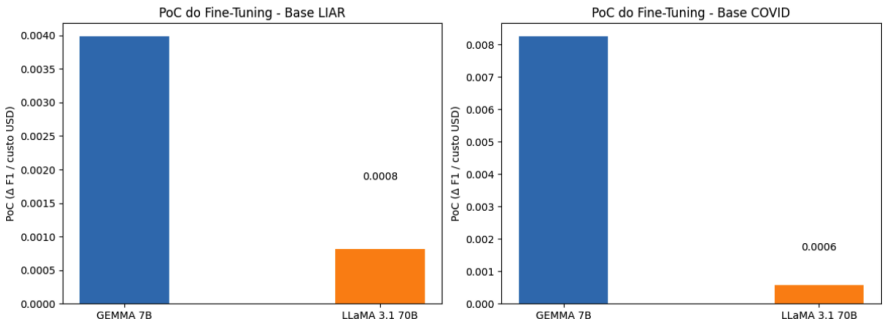


Figura 6.13: Resultados do PoC-score obtidos no fine-tuning dos modelos GEMMA 7B e LLaMA 3.1 70B para as bases LIAR e COVID.

6.5 Análise dos Resultados do *Fine-Tuning*

Os resultados obtidos a partir do fine-tuning dos modelos nas bases COVID e LIAR revelam comportamentos distintos, diretamente relacionados às características estruturais de cada conjunto de dados.

Na base COVID, observa-se que o *fine-tuning* proporcionou melhorias significativas em todas as métricas avaliadas. Para o modelo *Llama-3-1-70B-Instruct-fine-tuned*, a configuração *few-shot* com ambas as definições alcançou um F1 score de 0,9431 e acurácia de 0,9393, valores superiores aos obtidos sem *fine-tuning* (F1 score de 0,9150). Isso demonstra que a especialização supervisionada foi efetiva, possivelmente porque os textos na base COVID apresentam um número médio de *tokens* suficiente para que os modelos extraiam informações contextuais relevantes. Adicionalmente, a variação entre as diferentes estratégias de *prompting (few-shot e zero-shot)* é relativamente pequena, indicando que, após o *fine-tuning*, os modelos se tornaram menos dependentes de instruções explícitas no prompt.

Por outro lado, na base LIAR, o impacto do *fine-tuning* foi bastante limitado. Embora haja uma ligeira melhora na acurácia em algumas configurações — por exemplo, o modelo *Llama-3-1-70B-Instruct-fine-tuned* na configuração *zero-shot (true definition)* atingiu uma acurácia de 71,17% —, os F1 scores permaneceram baixos, não ultrapassando 0,6253. Além disso, há uma discrepância relevante entre precisão e *recall*, especialmente em configurações zero-shot, evidenciando instabilidade na capacidade do modelo de balancear falsos positivos e falsos negativos. Essa dificuldade está diretamente relacionada à natureza da base LIAR, composta por textos extremamente curtos, com média de tokens por instância significativamente inferior à base COVID (conforme discutido na Figura 5.1c). A escassez de contexto textual limita a capacidade dos modelos de linguagem em capturar padrões semânticos robustos, tornando a tarefa de classificação mais desafiadora, mesmo após o ajuste supervisionado.

De forma geral, o *fine-tuning* demonstrou ser uma estratégia eficaz para cenários em que há maior disponibilidade de contexto textual, como é o caso da base COVID. Entretanto, em bases como a LIAR, a estratégia apresentou ganhos marginais, reforçando a hipótese de que, para tarefas de classificação envolvendo textos muito curtos, o simples ajuste dos pesos dos modelos não é suficiente. Nesses casos, abordagens complementares, como o uso de informações estruturadas adicionais, expansão do contexto via recuperação de documentos ou enriquecimento semântico, podem ser necessárias para superar as limitações impostas pela própria natureza dos dados.

Além disso, ao comparar os F1 scores médios, observa-se que o *fine-tuning* elevou o desempenho na base COVID (F1 de 0,9431 com *fine-tuning* contra 0,9150 sem). Entretanto, na base LIAR, ocorreu o oposto, com uma leve redução no F1 score após o fine-tuning (0,5764 com fine-tuning contra 0,6253 sem). Isso corrobora a conclusão de que, embora o fine-tuning seja uma estratégia robusta, sua efetividade está fortemente condicionada às características linguísticas e estruturais do conjunto de dados utilizado.

Por fim, vale ressaltar que o desempenho na base COVID do modelo *Gemma 2B*, com um F1 de 0,9404, se equipara ao desempenho do modelo *Llama 3.1-70B* que tem um custo por hora quase dez vezes maior. Tornando assim o Gemma 7B a opção mais interessante nesse cenário.

6.5.1 Modelo Selecionado para a Solução Final

Com base na análise detalhada dos resultados apresentados nas seções anteriores, a escolha do modelo para a solução final recaiu sobre o **Gemma-7B-Instruct**. Essa decisão está fundamentada em uma análise multidimensional que considera tanto o desempenho técnico quanto fatores operacionais relacionados a custo, escalabilidade e viabilidade prática.

Em termos de desempenho, o *Gemma-7B-Instruct* apresentou resultados consistentes e competitivos na maioria das bases avaliadas, destacando-se especialmente em conjuntos como COVID, FakeNewsNet e ISOT. Embora não tenha atingido os mesmos níveis de robustez e generalização observados no *Llama-3.1-70B*, o *Gemma-7B* oferece um equilíbrio satisfatório entre *recall* e *precisão*, além de demonstrar menor sensibilidade às variações de *prompt* quando comparado ao modelo de menor porte (*Gemma-2B*).

Adicionalmente, a análise de custo computacional foi um fator decisivo. Enquanto o *Llama-3.1-70B* impõe elevados custos operacionais — inviabilizando seu uso contínuo em ambientes com restrições de orçamento ou infraestrutura —, o *Gemma-7B* opera em instâncias significativamente mais acessíveis, com custo-hora

inferior a 7 dólares, mantendo desempenho competitivo. Esse fator torna o modelo particularmente adequado para aplicações em produção, onde há necessidade de otimizar a relação custo-benefício sem sacrificar excessivamente a qualidade dos resultados.

Por fim, os resultados obtidos com o modelo *DeepSeek-R1 Distill Qwen 32B* reforçam essa escolha. Apesar de possuir mais parâmetros que o *Gemma-7B*, o *DeepSeek-R1* não apresentou ganhos proporcionais em desempenho que justificassem seu custo operacional quatro vezes maior. Isso consolida o *Gemma-7B-Instruct* como uma solução de excelente custo-efetividade para a tarefa de detecção de fake news no contexto avaliado.

Portanto, a combinação de desempenho técnico adequado, estabilidade frente às variações de configuração, e viabilidade econômica posiciona o *Gemma-7B-Instruct* como o modelo mais indicado para compor a arquitetura da solução proposta neste trabalho.

Capítulo 7

Conclusões e Trabalhos Futuros

7.1 Comparação dos Resultados em Diferentes Bases para Detecção de Fake News

Nesta seção, comparamos o desempenho dos modelos Gemma-2B-Instruct, Gemma-7B-Instruct e Llama-3.1-70B-Instruct ao serem avaliados em múltiplas bases de dados relevantes para a detecção de fake news: COVID, LIAR, FakeNewsNet, ISOT, PolitiFact FactCheck Data e WELFAKE. Essa análise destaca padrões gerais, desafios específicos de cada base, e o impacto das estratégias zero-shot e few-shot com diferentes tipos de definições.

Impacto do Tamanho do Modelo e Robustez

De forma consistente, observamos que o aumento da capacidade do modelo — da Gemma-2B para a Gemma-7B, e principalmente para o Llama-3.1-70B — traz ganhos relevantes em desempenho, sobretudo em termos de *F1-Score* e *recall*. O Llama-3.1-70B se destaca por sua robustez e estabilidade entre as configurações *zero-shot* e *few-shot*, enquanto os modelos *Gemma* menores apresentam desempenho mais errático e maior sensibilidade à formulação do *prompt* e das definições.

Comparação entre Cenários *Zero-Shot* e *Few-Shot*

Surpreendentemente, em várias bases — incluindo FakeNewsNet, ISOT e PolitiFact — o cenário zero-shot apresentou resultados superiores ou pelo menos equivalentes aos do few-shot, especialmente para o modelo Gemma-7B. Isso sugere que fornecer exemplos no few-shot nem sempre contribui para a melhoria da classificação, possivelmente devido a exemplos pouco representativos, ruidosos ou a uma sobrecarga cognitiva para o modelo, que pode causar confusão semântica.

No Llama-3.1-70B, a diferença entre *zero-shot* e *few-shot* é menos pronunciada,

reforçando sua maior capacidade de generalização e adaptabilidade a diferentes formulações.

Influência das Definições no *Prompt*

O uso da técnica *output constraint* provou-se extremamente eficiente, dado a taxa de alucinação quase nula em todo o experimento. Este resultado contrasta com estudos anteriores, como o de JÚNIOR (2024), que reportaram taxas de alucinação de até 16% em cenários sem restrição de saída, especialmente em tarefas sensíveis à geração livre de texto. Isso reforça a efetividade de estratégias que limitam o espaço de resposta dos modelos, mitigando significativamente erros factuais.

O uso de definições explícitas *("true", "false", "both")* mostrou efeitos variáveis. Para modelos menores (Gemma-2B e 7B), a inclusão da definição *"both"* frequentemente prejudicou o desempenho, indicando que o excesso de informação ou uma formulação ambígua pode levar à confusão do modelo. Já o Llama-3.1-70B apresentou maior tolerância a diferentes definições, embora não tenha havido um padrão uniforme de benefício, e em algumas bases a ausência de definições *("none")* levou a resultados melhores.

Análise por Base de Dados

Base COVID: A base mais simples do conjunto, onde o Llama-3.1-70B alcançou F1 acima de 0,84 em quase todos os cenários, com bom equilíbrio entre precisão e *recall*. Gemma-7B já apresentou melhorias significativas em relação ao 2B, mas ainda com tendências de alta sensibilidade (alto *recall* e baixa precisão).

Base LIAR: Confirmada como uma base desafiadora devido à sua alta diversidade e à menor redundância semântica presente nos dados. Grande parte das notícias contém textos muito curtos, com cerca de 15 palavras em média, o que limita significativamente o contexto disponível para os modelos. Essa escassez de informação contextual dificulta a identificação precisa da veracidade das afirmações, impactando negativamente o desempenho, especialmente dos modelos menores e em cenários *zero-shot*. Nenhum modelo conseguiu ultrapassar F1 próximo a 0,57, evidenciando a complexidade do desafio proposto por esta base.

FakeNewsNet: Ambos os modelos enfrentaram dificuldades, com acurácias abaixo de 70%. Surpreendentemente, nesta base, o melhor resultado em termos de F1 foi para o modelo Gemma 7b.

ISOT: Mostrou-se mais fácil que FakeNewsNet e PolitiFact. Llama-3.1-70B atingiu os melhores resultados, com acurácia superior a 75% e F1 acima de 0,80 na melhor configuração zero-shot. Gemma-7B teve desempenho inferior, com alta sensibilidade, porém com muitos falsos positivos.

PolitiFact FactCheck Data: Trata-se de uma base desafiadora, caracterizada por textos curtos, muitas vezes ambíguos, e por um forte desbalanceamento, com aproximadamente três vezes mais notícias falsas do que verdadeiras. Esse fator contribui para enviesar os modelos na predição da classe majoritária, o que se reflete na predominância de alta sensibilidade em detrimento da precisão, indicando um alto custo em falsos positivos. O desempenho geral ficou abaixo do esperado para ambos os modelos, com baixa precisão e F1 abaixo de 0,49 no Gemma-7B, e apenas modestos ganhos no Llama-3.1-70B, que alcançou F1 de até 0,56.

WELFAKE: O cenário mais crítico para ambos os modelos. A base é composta por textos longos, com média de aproximadamente 500 *tokens* por notícia, podendo ultrapassar 1.000 *tokens* em muitos casos. Esse fator impõe uma elevada sobrecarga cognitiva aos modelos, especialmente no processamento de relações de longo alcance e na retenção de informações relevantes. O Gemma-7B alcançou acurácia moderada (50%) e F1 até 0,67 em *few-shot (none)*, porém com alto *recall* e baixa precisão. Já o Llama-3.1-70B apresentou desempenho quase aleatório, com várias configurações abaixo de 24% em acurácia e F1 muito baixos, o que reforça a elevada dificuldade imposta por essa base.

Padrões Gerais e Implicações

O Llama-3.1-70B se mostra o modelo mais apto para a tarefa, oferecendo maior estabilidade, menor sensibilidade à formulação do *prompt*, e melhor equilíbrio entre métricas em bases variadas.

O desempenho inferior em bases como LIAR, PolitiFact e WELFAKE indica limitações dos modelos de linguagem generalistas para tarefas complexas, sobretudo em domínios com textos curtos, ambíguos, ou altamente diversificados.

A preferência pelo zero-shot em vez do *few-shot* em várias situações sugere que modelos instruídos podem beneficiar-se mais de *prompts* claros e diretos do que de exemplos ilustrativos, especialmente se estes forem mal formulados ou pouco representativos.

A alta sensibilidade com baixa precisão é uma tendência recorrente, refletindo a prioridade dos modelos em minimizar falsos negativos às custas de muitos falsos positivos — uma característica importante a ser considerada em aplicações práticas, pois falsos positivos excessivos podem prejudicar a confiança no sistema.

7.2 Conclusão

A presente pesquisa buscou responder às questões delineadas na Seção 1.3, avaliando o uso de modelos de linguagem de grande porte (LLMs) na tarefa de detecção

de *fake news*. As respostas às perguntas propostas são apresentadas a seguir.

1. **LLMs são capazes de identificar notícias falsas?**
   De maneira geral, sim. Os resultados obtidos demonstram que os LLMs possuem capacidade concreta para realizar a detecção de *fake news* em múltiplos contextos. Entretanto, seu desempenho é altamente dependente das características dos dados. Nas bases com textos mais longos, maior riqueza contextual e menor nível de ruído, como COVID e ISOT, os modelos apresentaram resultados excelentes, superando *F1-scores* de 0,90 em alguns casos. Por outro lado, em bases mais desafiadoras, como LIAR e PolitiFact, compostas por textos curtos e muitas vezes ambíguos, o desempenho dos LLMs se mostrou inferior, indicando limitações claras nesse tipo de cenário.

2. **Como os LLMs se comparam em desempenho com o estado da arte?**
   De maneira geral, os LLMs apresentaram desempenho competitivo em algumas bases, mas ainda ficam atrás do estado da arte em outras. Na base COVID, os modelos se mostraram particularmente fortes, com destaque para o LLaMA 3.1 70B, que obteve um F1 de 0,9015 e recall de 0,9688, valor bastante competitivo frente aos melhores resultados da literatura (accuracy de 98,5%). Na FakeNewsNet, embora os F1 estejam na faixa de 0,69 a 0,75, os modelos podem ser considerados competitivos, especialmente devido à complexidade da base e ausência de benchmarks amplamente consolidados.

   Por outro lado, nas bases ISOT, PolitiFact e WELFake, os LLMs ficaram significativamente abaixo dos resultados reportados na literatura, que superam 95% de acurácia com modelos tradicionais relativamente simples, como Regressão Logística e Stacking Ensemble. Na base LIAR, os LLMs também não superaram o estado da arte, embora os resultados estejam na faixa tipicamente observada em estudos anteriores, considerando a dificuldade intrínseca da base.

   Esses achados indicam que, embora os LLMs sejam ferramentas poderosas e apresentem desempenho competitivo em alguns cenários, eles ainda não substituem completamente modelos supervisionados tradicionais, especialmente em bases mais simples ou bem estruturadas, onde representações vetoriais e classificadores convencionais como SVM, Random Forest e ensembles continuam superiores.

3. **Qual é a relação de custo-benefício entre LLMs de diferentes modelos e tamanhos?**
   A análise revelou que, embora exista uma tendência de modelos maiores apresentarem melhor desempenho, esse ganho não cresce na mesma proporção do

aumento no tamanho do modelo. Por exemplo, ao comparar o LLaMA 3.1-70B com o Gemma-7B — cuja diferença de tamanho é de 10 vezes — observa-se que o F1-score do LLaMA é, em média, apenas de 5 a 10 pontos percentuais maior nas bases em que supera o Gemma. No entanto, o custo operacional cresce de forma muito mais acentuada, sendo aproximadamente 6 vezes maior para o LLaMA 70B. Esse descompasso entre aumento de desempenho e escalada de custos torna os modelos intermediários, como o Gemma-7B, significativamente mais atraentes do ponto de vista de custo-benefício. Em diversos casos, esses modelos menores alcançam desempenhos competitivos, próximos ou até superiores aos modelos maiores, especialmente quando se consideram bases específicas ou cenários de zero-shot. Além disso, estratégias como fine-tuning se mostraram extremamente eficazes para potencializar o desempenho dos modelos menores, ampliando ainda mais sua vantagem relativa em termos de eficiência e viabilidade econômica.

De forma geral, os experimentos realizados demonstram que os LLMs são ferramentas promissoras para a detecção de *fake news*, especialmente quando aplicados em domínios com maior riqueza textual. No entanto, seu uso deve ser ponderado de acordo com as características da aplicação, o volume de dados, os custos operacionais e a necessidade (ou não) de realizar ajustes finos nos modelos.

7.3 Trabalhos Futuros

Como trabalhos futuros, destaca-se a necessidade de uma exploração mais aprofundada de técnicas de *fine-tuning*. Investigar abordagens de ajuste fino mais sofisticadas, potencialmente combinadas com estratégias de *prompt engineering*, pode revelar melhorias significativas no desempenho e na estabilidade dos modelos.

Outra direção importante envolve a análise do custo de inferência em relação à performance dos modelos, fator crucial para aplicações práticas. Avaliar a eficiência computacional e otimizar o uso dos grandes modelos de linguagem (LLMs) em ambientes de produção permitirá um melhor balanceamento entre custo e benefício, contribuindo para a viabilidade do uso desses modelos em larga escala.

Além disso, os resultados indicam que o alto desempenho observado na base COVID pode estar associado ao amplo conhecimento prévio dos LLMs em domínios médicos e científicos, reflexo do conteúdo diverso utilizado durante o treinamento. Em contrapartida, bases com temáticas mais variáveis e complexas, como notícias políticas e de entretenimento, demonstraram resultados inferiores, sugerindo que futuras pesquisas devem explorar estratégias específicas para lidar com essa diversidade, como o enriquecimento semântico ou recuperação contextual de documentos.

Outra linha promissora é a avaliação de técnicas de *ensemble*, que combinam previsões de múltiplos modelos para aumentar a robustez e a precisão na detecção de *fake news*. Essa abordagem pode aproveitar os pontos fortes de diferentes arquiteturas e configurações, mitigando as limitações individuais observadas nos experimentos.

Por fim, sugere-se investigar métodos para lidar com os desafios impostos por bases com textos muito longos ou muito curtos, como as bases WELFAKE e LIAR, respectivamente. Para isso, o uso de técnicas de pré-processamento, segmentação de texto, e inclusão de informações estruturadas ou metadados pode ser essencial para superar as limitações atuais.

Adicionalmente, mesmo nos cenários em que os LLMs não apresentaram o melhor desempenho bruto em termos de métricas supervisionadas, eles se mostram extremamente promissores como componentes auxiliares em pipelines de detecção de *fake news*. Entre as aplicações potenciais, destacam-se: (i) geração de *embeddings* semânticos ricos que podem ser utilizados como entrada para classificadores tradicionais, (ii) enriquecimento dos dados por meio de sumarização, expansão textual e extração de entidades, (iii) pré-processamento semântico avançado, incluindo desambiguação e normalização de textos, (iv) filtragem preliminar de conteúdos irrelevantes ou não noticiosos e (v) integração em modelos híbridos, nos quais os LLMs fornecem representações contextuais ou classificações preliminares que são posteriormente refinadas por modelos mais especializados e eficientes.

Essas direções não apenas visam a melhoria do desempenho técnico dos modelos, mas também a construção de sistemas mais robustos, interpretáveis e eficientes, alinhados às demandas práticas e operacionais da detecção de *fake news* em ambientes reais.

Referências Bibliográficas

ALGHAMDI, J., LIN, Y., LUO, S., 2023, "Towards COVID-19 fake news detection using transformer-based models", *Knowledge-Based Systems*. doi: 10.1016/j.knosys.2023.110642.

ALI, M., GOMES, L. M., AZAB, N., et al., 2023, "Panic buying and fake news in urban vs. rural England: A case study of Twitter during COVID-19", *Technological Forecasting and Social Change*, v. 193, pp. 122598. doi: 10.1016/j.techfore.2023.122598. Disponível em: <https://doi.org/10.1016/j.techfore.2023.122598>.

ALLCOTT, H., GENTZKOW, M., 2017, "Social Media and Fake News in the 2016 Election", *Journal of Economic Perspectives*, v. 31, n. 2 (5), pp. 211–36. doi: 10.1257/jep.31.2.211. Disponível em: <https://www.aeaweb.org/articles?id=10.1257/jep.31.2.211>.

BAARIR, N. F., DJEFFAL, A., 2021, "Fake News detection Using Machine Learning". In: *2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH)*, pp. 125–130. doi: 10.1109/IHSH51661.2021.9378748.

BARBIÉRI, L. F., 2019, "Congresso instala comissão para investigar divulgação de informações falsas", *G1*, (9). Disponível em: <https://g1.globo.com/politica/noticia/2019/09/04/congresso-instala-comissao-para-investigar-divulgacao-de-informacoes-falsas.ghtml>. 4/9/2019.

BIRUNDA, S., DEVI, R., 2021, "A Novel Score-Based Multi-Source Fake News Detection using Gradient Boosting Algorithm", *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, pp. 406–414. doi: 10.1109/ICAIS50930.2021.9395896.

BREIMAN, L., 1996, "Bagging predictors", *Machine Learning*, v. 24, n. 2, pp. 123–140.

BREIMAN, L., 2001, "Random Forests", *Machine Learning*, v. 45, n. 1, pp. 5–32. doi: 10.1023/A:1010933404324.

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., et al., 1984, *Classification and Regression Trees*. Belmont, CA, Wadsworth International Group. ISBN: 9780412048418.

BROWN, T. B., MANN, B., RYDER, N., et al., 2020, "Language models are few-shot learners", *Advances in Neural Information Processing Systems*, v. 33, pp. 1877–1901.

CHOWDHERY, A., NARANG, S., DEVLIN, J., et al., 2022, "PaLM: Scaling Language Modeling with Pathways", *arXiv preprint arXiv:2204.02311*. 540 billion-parameter Pathways Language Model.

CORTES, C., VAPNIK, V., 1995, "Support-vector networks", *Machine Learning*, v. 20, n. 3, pp. 273–297.

COVER, T. M., HART, P. E., 1967, "Nearest neighbor pattern classification", *IEEE Transactions on Information Theory*, v. 13, n. 1, pp. 21–27. doi: 10.1109/TIT.1967.1053964.

COX, D. R., 1958, "The regression analysis of binary sequences", *Journal of the Royal Statistical Society: Series B (Methodological)*, v. 20, n. 2, pp. 215–242.

DE MAGISTRIS, G., RUSSO, S., ROMA, P., et al., 2022, "An Explainable Fake News Detector Based on Named Entity Recognition and Stance Classification Applied to COVID-19", *Information (Switzerland)*, v. 13, n. 3. doi: 10.3390/info13030137. Disponível em: <https://www.mdpi.com/2078-2489/13/3/137>.

DEEPSEEK-AI, GUO, D., YANG, D., et al., 2025. "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning". Disponível em: <https://arxiv.org/abs/2501.12948>.

DEVLIN, J., CHANG, M.-W., LEE, K., et al., 2019, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *ArXiv*, v. abs/1810.04805.

DING, B., QIN, C., ZHAO, R., et al., 2024, "Data Augmentation using LLMs: Data Perspectives, Learning Paradigms and Challenges", (March). Disponível em: <http://arxiv.org/abs/2403.02990>. Disponível em: http://arxiv.org/abs/2403.02990.

EKIN, S., 2023. "Prompt Engineering For ChatGPT: A Quick Guide To Techniques, Tips, And Best Practices". TechRxiv, abr. Accessed via TechRxiv.

FREUND, Y., SCHAPIRE, R. E., 1997, "A decision-theoretic generalization of online learning and an application to boosting", *Journal of Computer and System Sciences*, v. 55, n. 1, pp. 119–139.

G1, 2018. "É #FAKE imagem em que Manuela D'Ávila aparece com camiseta 'Jesus é travesti"'. Disponível em: <https://g1.globo.com/fato-ou-fake/noticia/2018/10/02/e-fake-imagem-em-que-manuela-davila-aparece-com-camiseta-jesus-e-travesti.ghtml>.

GEMINI, 2023. "Gemini: A Family of Highly Capable Multimodal Models". Disponível em: <https://arxiv.org/abs/2312.11805>. Acessado em: 26 abr. 2025.

GETHSIA, P., JULIET, S., 2023, "An Enhanced Approach for Fake News Detection using Ensemble Techniques". In: *Proceedings of the 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE. doi: 10.1109/ICACCS57279.2023.10112856.

GUPTA, S., YADAV, N., KUNDU, S., et al., 2024, "FakEDAMR: Fake News Detection Using Abstract Meaning Representation Network". In: *Studies in Computational Intelligence*, Springer. doi: 10.1007/978-3-031-53468-3_26.

HARRIS, S., HADI, H., AHMAD, N., et al., 2024, "Fake News Detection Revisited: An Extensive Review of Theoretical Frameworks, Dataset Assessments, Model Constraints, and Forward-Looking Research Agendas", *Technologies*, v. Technologies 2024, (11). doi: 10.3390/technologies12110222.

HOCHREITER, S., SCHMIDHUBER, J., 1997, "Long Short-Term Memory", *Neural Computation*, v. 9, n. 8, pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HU, E. J., SHEN, Y., WALLIS, P., et al., 2021. "LoRA: Low-Rank Adaptation of Large Language Models". Disponível em: <https://arxiv.org/abs/2106.09685>.

JONES, K. S., 1972, "A statistical interpretation of term specificity and its application in retrieval", *Journal of Documentation*, v. 28, n. 1, pp. 11–21.

JÚNIOR, P. R. X., 2024, *Desempenho e Economicidade de Modelos de Linguagem para Classificação de Toxicidade em Jogos*. Dissertação de mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, ago. Orientador: Geraldo Bonorino Xexéo, D.Sc.

KARIMI, H., ROY, P., SABA-SADIYA, S., et al., 2018, "Multi-Source Multi-Class Fake News Detection". In: *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1546–1557, Santa Fe, New Mexico, USA, ago. Association for Computational Linguistics. Disponível em: <https://aclanthology.org/C18-1131>.

KENT, A., BERRY, M. M., HATCH, F. U., et al., 1955, "Machine Literature Searching VIII: Operational Criteria for Designing Information Retrieval Systems", *American Documentation*, v. 6, n. 2, pp. 93–101. doi: 10.1002/asi.5090060203. Disponível em: <https://doi.org/10.1002/asi.5090060203>.

KLEIN, D., WUELLER, J., 2017, "Fake News: A Legal Perspective", *Journal of Internet Law*, (April). Disponível em: <https://ssrn.com/abstract=2958790>. Available at SSRN: https://ssrn.com/abstract=2958790.

KÜÇÜK, D., CAN, F., 2021, "Stance Detection: Concepts, Approaches, Resources, and Outstanding Issues". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, p. 2673–2676, New York, NY, USA. Association for Computing Machinery. ISBN: 9781450380379. doi: 10.1145/3404835.3462815. Disponível em: <https://doi.org/10.1145/3404835.3462815>.

LAZOVSKY, S., RAZ, T., KENETT, Y. N., 2025, "The Art of Creative Inquiry—From Question Asking to Prompt Engineering", *Journal of Creative Behavior*, v. 59, pp. e671. doi: 10.1002/jocb.671.

LE, Q., MIKOLOV, T., 2014, "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pp. 1188–1196. Disponível em: <https://proceedings.mlr.press/v32/le14.html>.

LEWIS, D. D., 1998, "Naive (Bayes) at forty: The independence assumption in information retrieval". In: *Machine Learning: ECML-98*, pp. 4–15. Springer.

LEWIS, P., PEREZ, E., PIKTUS, A., et al., 2020, "Retrieval-augmented generation for knowledge-intensive NLP tasks", *Advances in Neural Information Processing Systems*, v. 33, pp. 9459–9474.

LIU, M. X., LIU, F., FIANNACA, A. J., et al., 2024, "We Need Structured Output: Towards User-centered Constraints on Large Language Model Output",

arXiv preprint arXiv:2404.07362. Disponível em: <https://arxiv.org/abs/2404.07362>. Accessed: 2025-05-07.

MARTINS SAMUEL DOGO, D. P., JUREK-LOUGHREY, A., 2020, "Exploring Thematic Coherence in Fake News", (December). Disponível em: <https://paperswithcode.com/dataset/isot-fake-news-dataset>.

MARWAH, M., NARAYANAN, A., JOU, S., et al., 2024. "Is F1 Score Suboptimal for Cybersecurity Models? Introducing Cscore, a Cost-Aware Alternative for Model Assessment". Disponível em: <https://arxiv.org/abs/2407.14664>.

METROPOLES, 2024. "Pesquisa da UnB revela o impacto das fake news no cérebro; entenda". Disponível em: <https://www.metropoles.com/distrito-federal/pesquisa-da-unb-revela-o-impacto-das-fake-news-no-cerebro-entenda>.

MIKOLOV, T., CHEN, K., CORRADO, G., et al., 2013a, "Efficient Estimation of Word Representations in Vector Space", *arXiv preprint arXiv:1301.3781*. doi: 10.48550/arXiv.1301.3781. Disponível em: <https://arxiv.org/abs/1301.3781>.

MIKOLOV, T., CHEN, K., CORRADO, G., et al., 2013b, "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of the International Conference on Learning Representations (ICLR)*, b. Disponível em: <https://arxiv.org/abs/1301.3781>.

MISRA, R., 2022. "PolitiFact". Disponível em: <https://www.kaggle.com/datasets/rmisra/politifact-fact-check-dataset>.

MOURATIDIS, D., NIKIFOROS, M., KERMANIDIS, K., 2021, "Deep learning for fake news detection in a pairwise textual input schema", *Computation*, v. 9, n. 2, pp. 1–15. doi: 10.3390/computation9020020. Disponível em: <https://www.mdpi.com/2079-3197/9/2/20>.

MURRAY, M. D., 2024, "Prompt Engineering and Priming in Law", (July). Disponível em: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4909532>. Accessed: 2025-05-07.

NEWMAN, N., FLETCHER, R., SCHULZ, A., et al., 2021. "Reuters Institute Digital News Report 2021". Disponível em: <https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2021>. Accessed: 2025-05-01.

OUYANG, L., WU, J., JIANG, X., et al., 2022, "Training language models to follow instructions with human feedback", *Advances in Neural Information Processing Systems*, v. 35, pp. 27730–27744.

PARIKH, S. B., PATIL, V., ATREY, P. K., 2019, "On the Origin, Proliferation and Tone of Fake News". In: *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 135–140. doi: 10.1109/MIPR.2019.00031.

PATWA, P., SHARMA, S., PYKL, S., et al., 2021, "Fighting an Infodemic: COVID-19 Fake News Dataset", Disponível em: <`https://arxiv.org/pdf/2011.03327`>.

PHANG, KELVIN KEAT HUNG AND CHUA, HUI NA AND JASSER, MUHAMMED BASHEER AND WONG, RICHARD T. K., 2023, "Fake News Detection Using Social Media User Network and Engagement Features". In: *ICSET 2023 - 2023 IEEE 13th International Conference on System Engineering and Technology, Proceedings*. IEEE. doi: 10.1109/ICSET59111.2023.10295146.

PYARELAL, A., R., R., PRABAHARAN, S. R. S., 2023, "Fake News Detection Using Machine Learning and Deep Learning Techniques". In: *Proceedings of the 2023 International Conference on Advances in Computing, Communication and Networking (ICAC3N)*, pp. 1–6. IEEE. doi: 10.1109/ICAC3N60023.2023.10541312.

RADFORD, A., WU, J., CHILD, R., et al., 2019, "Language models are unsupervised multitask learners", OpenAI Technical Report.

REZAEI, S., KAHANI, M., BEHKAMAL, B., et al., 2023, "Early multiclass ensemble-based fake news detection using content features", *Social Network Analysis and Mining*.

ROSING, M. V., WHITE, S., CUMMINS, F., et al., 2015, "Business Process Model and Notation (BPMN)". In: *The Complete Business Process Handbook*, v. 1, Morgan Kaufmann, pp. 429–453, mar. ISBN: 978-0-12-799959-3. doi: 10.1016/B978-0-12-799959-3.00021-5.

RUBLE, D. A., 1997, *Practical Analysis and Design for Client/Server and GUI Systems*. USA, Prentice-Hall, Inc. ISBN: 013521758X.

SAHOO, P., SINGH, A. K., SAHA, S., et al., 2024, "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Appli-

cations", (feb). Department of Computer Science and Engineering, IIT Patna; Stanford University; Amazon AI.

SALTON, G., BUCKLEY, C., 1988, "Term-weighting approaches in automatic text retrieval", *Information Processing & Management*, v. 24, n. 5, pp. 513–523.

SALTON, G., WONG, A., YANG, C. S., 1975, "A vector space model for automatic indexing", *Communications of the ACM*, v. 18, n. 11, pp. 613–620.

SANDRILLA, R., DEVI, S., 2024, "Ensemble Classifier Based Fake News Identification in Online Social Network", *ASEAN Engineering Journal*, v. 14. doi: 10.11113/aej.V14.18150.

SETIAWAN, R., PONNAM, V., SENGAN, S., et al., 2021, "Certain Investigation of Fake News Detection from Facebook and Twitter Using Artificial Intelligence Approach", *Wireless Personal Communications*. doi: 10.1007/s11277-021-08720-9. Disponível em: <https://link.springer.com/article/10.1007/s11277-021-08720-9>.

SHI, F., QING, P., YANG, D., et al., 2023, "Prompt Space Optimizing Few-shot Reasoning Success with Large Language Models", *arXiv preprint arXiv:2306.03799*. Disponível em: <http://arxiv.org/abs/2306.03799>.

SHU, K., MAHUDESWARAN, D., 2019. "FakeNewsNet: A Data Repository with News Content, Social Context and Spatialtemporal Information for Studying Fake News on Social Media". Disponível em: <https://www.kaggle.com/datasets/mdepak/fakenewsnet>.

SHU, K., SLIVA, A., WANG, S., et al., 2017, "Fake news detection on social media: A data mining perspective", *ACM SIGKDD Explorations Newsletter*, v. 19, n. 1, pp. 22–36.

SILVA, R., SANTOS, R., ALMEIDA, T., et al., 2020, "Towards automatically filtering fake news in Portuguese", *Expert Systems with Applications*, v. 146. ISSN: 09574174. doi: 10.1016/j.eswa.2020.113199. Disponível em: <https://doi.org/10.1016/j.eswa.2020.113199>. cited By 38.

TOUVRON, H., LAVRIL, T., IZACARD, G., et al., 2023, "LLaMA: Open and Efficient Foundation Language Models", *arXiv*. Disponível em: <https://arxiv.org/abs/2302.13971>. Acessado em: 26 abr. 2025.

UOL, 2024. “PizzaGate: a fake news que quase terminou em tragédia nos EUA”. Disponível em: <https://noticias.uol.com.br/internacional/ultimas-noticias/2024/09/19/pizza-gate-fake-news-eua.htm>.

VAN RIJSBERGEN, C. J., 1979, *Information Retrieval*. 2nd edition ed. Newton, MA, USA, Butterworth-Heinemann. Disponível em: <https://dl.acm.org/doi/10.1145/3606367>.

VASWANI, A., SHAZEER, N., PARMAR, N., et al., 2017a, “Attention is All You Need”, *Advances in neural information processing systems*, v. 30. Disponível em: <https://arxiv.org/abs/1706.03762>.

VASWANI, A., SHAZEER, N. M., PARMAR, N., et al., 2017b, “Attention is All you Need”, *ArXiv*, v. abs/1706.03762.

WANG, W. Y., 2017, “Liar, Liar Pants on Fire: A New Benchmark Dataset for Fake News Detection”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 422–426, Vancouver, Canada, jul. Association for Computational Linguistics. doi: 10.18653/v1/P17-2067. Disponível em: <https://aclanthology.org/P17-2067>.

WANG, X., WEI, J., SCHUURMANS, D., et al., 2022, “Self-consistency improves chain of thought reasoning in language models”, *arXiv preprint arXiv:2203.11171*.

WARDLE, C., DERAKHSHAN, H., 2017, *Information Disorder: Toward an Interdisciplinary Framework for Research and Policymaking*. Relatório técnico, Council of Europe. Disponível em: <https://rm.coe.int/information-disorder-report-november-2017/1680764666>. Accessed: 2025-05-01.

WEI, J., WANG, X., SCHUURMANS, D., et al., 2022, “Chain-of-thought prompting elicits reasoning in large language models”, *Advances in Neural Information Processing Systems*.

WELD, H., HUANG, G., LEE, J., et al., 2021. “CONDA: A CONtextual Dual-Annotated Dataset for In-game Toxicity Understanding and Detection”. Disponível em: <https://arxiv.org/abs/2106.06213>.

WOLPERT, D. H., 1992, “Stacked generalization”, *Neural Networks*, v. 5, n. 2, pp. 241–259.

WU, X., WANG, J., 2021, "SAFS: Social-Article Features-Stacking Model for Fake News Detection", *2021 4th International Conference on Artificial Intelligence and Big Data, ICAIBD 2021*, pp. 530–535. doi: 10.1109/ICAIBD51990.2021.9459078. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85113731987&doi=10.1109%2fICAIBD51990.2021.9459078&partnerID=40&md5=35d944e4b02798bda7f76202c4e7e7ff>.

ZHANG, J., CHANG, J. P., DANESCU-NICULESCU-MIZIL, C., et al., 2018, "Conversations Gone Awry: Detecting Early Signs of Conversational Failure". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1353–1363, Brussels, Belgium. Association for Computational Linguistics. doi: 10.18653/v1/D18-1164. Disponível em: <https://aclanthology.org/D18-1164>.

ZHANG, Z., ZHANG, A., LI, M., et al., 2022, "Automatic chain of thought prompting in large language models", *arXiv preprint arXiv:2210.03493*.

Apêndice A

Estrutura dos Prompts

Os prompts foram estruturados em duas categorias principais: **base** e **definitions**. O prompt base é dividido em versões *zero-shot* e *few-shot*, enquanto as definições servem para contextualizar o conceito de notícia falsa e verdadeira.

- **Base:**
  - *Zero-shot:* O modelo recebe a mensagem para classificar, com a instrução clara de retornar apenas a classificação em uma única palavra, *true* ou *false*, sem explicações adicionais.
  - *Few-shot:* Além da mensagem a ser classificada, o modelo recebe exemplos práticos previamente classificados para auxiliar no entendimento da tarefa.
- **Definitions:** São textos explicativos opcionais que descrevem os conceitos de *Fake News* e *True News*, baseados em referências acadêmicas relevantes. Esses textos podem ser incluídos ou omitidos no prompt para verificar o impacto no desempenho do modelo.

A.1 Exemplo do JSON do Prompt

```
{
    "base": {
      "zero-shot": "Classify the following statement as
'False' or 'True':\n Message: {{message}}.\n Return
only the Classification in a single word (true / false)
without any further explanation:",
      "few-shot": "Classify the following statement as
'False' or 'True': \n Message: {{message}}.\n Example 1:
```

```
 Message: The CDC currently reports 99031 deaths. In
general the discrepancies in death counts between
different sources are small and explicable. The death
toll stands at roughly 100000 people today.
Conclusion: True \n Retraction -Hydroxychloroquine or
chloroquine with or without a macrolide for treatment
of COVID-19: a multinational registry analysis - The
Lancet       https://t.co/L5V2x6G9or. Conclusion: False
\n Now classify the following message: \n Message: {{
message}} Return only the Classification in a single
word without any further explanation (True / False):"
  },
  "definitions": {
    "none": "",
    "false-definition": "Fake News: According to Tandoc
et al. (2018), fake news refers to content deliberately
created and disseminated with the intent to deceive the
public, often for financial, political, or other gains.
These pieces mimic real news in format but lack any
commitment to factual accuracy. Unlike other forms of
misinformation like rumors or satire, fake news is
crafted to appear credible while misleading its
audience on purpose.",
    "true-definition": "True News: True news, as
outlined by Kovach and Rosenstiel (2001), follows core
journalistic principles, relying on rigorous
fact-checking, verifiable sources, and transparency.
The primary aim is to inform the public accurately and
impartially, with a strong commitment to truth,
avoiding manipulation or distortion of facts.",
    "both-definition": "Fake News: According to Tandoc
et al. (2018), fake news refers to content deliberately
created and disseminated with the intent to deceive the
public, often for financial, political, or other gains.
These pieces mimic real news in format but lack any
commitment to factual accuracy. Unlike other forms of
misinformation like rumors or satire, fake news is
crafted to appear credible while misleading its
audience on purpose. True News: True news, as outlined
```

```
    by Kovach and Rosenstiel (2001), follows core
    journalistic principles, relying on rigorous
    fact-checking, verifiable sources, and transparency.
    The primary aim is to inform the public accurately and
    impartially, with a strong commitment to truth,
    avoiding manipulation or distortion of facts."
11    }
12 }
```

Listing A.1: Estrutura JSON dos Prompts

A.2 Exemplo Prático

Considerando a notícia abaixo, a classificação deve ser feita conforme o prompt:

`Elon Musck To New Baby; Get A Job Kid! https://t.co/bc8Re0Ai3Y #christmas #covid19 #elonmusk #achristmascarol`

O modelo recebe a instrução no formato zero-shot ou few-shot para classificar a mensagem apenas como *true* ou *false*.

Exemplo de prompt preenchido (zero-shot):

> Classify the following statement as 'False' or 'True':
> Message: Elon Musck To New Baby; Get A Job Kid! https://t.co/bc8Re0Ai3Y #christmas #covid19 #elonmusk #achristmascarol.
> Return only the Classification in a single word (true / false) without any further explanation:

O modelo deve responder com apenas uma palavra: `false` (neste caso, dado que o título é falso e sensacionalista).

Apêndice B

Códigos Utilizados

Nesta seção são apresentados os principais códigos usados para realizar as predições nos experimentos desta dissertação.

```

import boto3
from sagemaker.predictor import Predictor, retrieve_default
import sagemaker
from utils import (
    generate_instruction_prompt,
    prepare_prompt_list,
    process_csv,
    get_generated_text,
)
import pandas as pd
import time
from transformers import GPT2Tokenizer
import csv
import threading


tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

datasets = {
    'liar': {
        'path': '../../datasets-fake-news/processed/liar-test.csv',
        'prompt_path': 'prompts/prompts-liar.json'
    },
    'covid': {
        'path': '../../datasets-fake-news/processed/covid-test.csv',
        'prompt_path': 'prompts/prompts-covid.json'
    },
    'fakenews-net': {
        'path':
   '../../datasets-fake-news/processed/split_data/fakenewsnet_test.csv',
```

```
        'prompt_path': 'prompts/prompts-fakenewsnet.json'
    },
    'isot': {
        'path':
'../../datasets-fake-news/processed/split_data/isot_complete_test.csv',
        'prompt_path': 'prompts/prompts-isot.json'
    },
    'politifact_factcheck_data': {
        'path':
'../../datasets-fake-news/processed/split_data/politifact_factcheck_test.csv',
        'prompt_path': 'prompts/prompts-politifact.json'
    },
    'welfake': {
        'path':
'../../datasets-fake-news/processed/split_data/welfake_test.csv',
        'prompt_path': 'prompts/prompts-welfake.json'
    }
}


models = {
     'gemma-2b': {
         'model_id': 'google/gemma-2b',
         'model_name': 'Gemma-2B-Instruct',
         'model_family': 'gemma',
         'endpoint_name':
'jumpstart-dft-hf-llm-gemma-7b-instr-20241103-193822',
         'new': True,
         'tuned_for': None
     },
     "gemma-7b": {
         "model_id": "google/gemma-7b",
         "model_name": "Gemma-7B-Instruct",
         "model_family": "gemma",
         'endpoint_name':
'jumpstart-dft-hf-llm-gemma-7b-instr-20241116-155953',
         "new": True,
         'tuned_for': None
     },
     "llama-3-1-70b": {
         "model_id": "meta-llama/Meta-Llama-3-70B-Instruct",
         "model_name": "Llama-3-1-70B-Instruct",
         "model_family": "llama3",
         "endpoint_name":
"jumpstart-dft-llama-3-1-70b-instruc-20241225-133356",
         # "inference_component_name":
"meta-textgeneration-llama-3-70b-instruct-20240606-134652",
```

```
            "new": True,
            'max_tokens': 7000
    },
    "deep-seek-32B": {
        "model_id": "deepseek-coder/deepseek-coder-32b",
        "model_name": "deep-seek-32B",
        "model_family": "deepseek",
        "endpoint_name":
    "jumpstart-dft-deepseek-llm-r1-disti-20250503-175058",
        "new": True,
        'max_tokens': 1024
    }
}


def extract_predicted_label(generated_text):
    predicted_label = 'HALLUCINATION'
    if(predicted_label == None):
        return predicted_label
    if(generated_text.startswith('true')):
        predicted_label = 'true'
    elif(generated_text.startswith('false')):
        predicted_label = 'false'
    else:
        _str = generated_text.split('user')[0]
        if 'true' in _str:
            predicted_label = 'true'
        elif 'real' in _str:
            predicted_label = 'true'
        elif 'false' in _str:
            predicted_label = 'false'
        elif 'fake' in _str:
            predicted_label = 'false'

    return predicted_label

def read_dataset(key):
    return pd.read_csv(datasets[key]['path'], nrows=1000)


def run_prediction(current_model_name, current_dataset_name):

    current_model = models[current_model_name]
    model_id = current_model.get("model_id")
    model_name = current_model.get("model_name")
    model_family = current_model.get("model_family")
    endpoint_name = current_model.get("endpoint_name")
```

```
    isNew = current_model.get("new")
    tuned_for = current_model.get("tuned_for")
    max_tokens = current_model.get("max_tokens")

    if(tuned_for != None and tuned_for != current_dataset_name):
        return

    predictor = (
        # Predictor(endpoint_name)
        retrieve_default(
            endpoint_name=endpoint_name
        )
        if current_model.get("new") == False
        else retrieve_default(
            endpoint_name=endpoint_name,

inference_component_name=current_model.get("inference_component_name"),
        )
    )

    predictor.content_type = "application/json"
    predictor.accept = "application/json"

    prompt_path = datasets[current_dataset_name]['prompt_path']
    prompt_list = prepare_prompt_list(prompt_path)

    for key, prompts in prompt_list.items():
        for _prompt in prompts:
            definition = _prompt["definition"]
            prompt = _prompt["prompt"]
            print(
                f"Running predictions for {model_name} with prompt
{key} and definitions: {definition}\n\n"
            )

            start_time = time.strftime("%Y-%m-%dT%H:%M:%S")
            filename =
f"predictions-{model_name}-{key}-{definition}-{start_time}"
            with open(

f"../results/{model_name}/{current_dataset_name}/{filename}.csv",
"w", newline=""
            ) as file:
                writer = csv.writer(file)
                writer.writerow(
                    ["true", "prediction", "input_tokens",
"output_tokens", "current_time"]
```

```
                )

                dataset = read_dataset(current_dataset_name)
                for index, row in dataset.iterrows():
                    text = row['text']
                    text_limited_by_max_tokens = text[:max_tokens]
                    input_data = generate_instruction_prompt(prompt,
    model_family, text_limited_by_max_tokens)
                    #input_data = generate_instruction_prompt(prompt,
    model_family, row['text'])
                    payload = None
                    if model_family == 'deepseek':
                        payload = input_data
                    else:
                        payload = {
                            "inputs": input_data["prompt"],
                            "parameters": input_data["parameters"],
                        }

                    response = predictor.predict(payload)
                    generated_text = get_generated_text(response,
    isNew, model_family)

                    if generated_text == None:
                        generated_text = ''

                    generated_text = generated_text.lower()
                    predicted_label =
    extract_predicted_label(generated_text)
                    input_tokens = len(text.split(' '))
    #len(tokenizer.encode(prompt))
                    output_tokens = len(generated_text.split(' '))
    #len(tokenizer.encode(generated_text))

                    writer.writerow(
                        [row['label'], predicted_label, input_tokens,
    output_tokens, time.strftime("%Y-%m-%dT%H:%M:%S")]
                    )


    print('-----------------------------------------------------')
                    print('Prompt: ', prompt)
                    print('Dataset: ', current_dataset_name)
                    print('Model: ', current_model_name)
                    print( f'{index + 1} / {dataset.size} Items
    processed')
                    print('Generated text:', generated_text, '\n')
```

```
                    print('Input:', row['text'], '\nLabel:',
    row['label'], '\nPrediction:',predicted_label, '\n')

    print('------------------------------------------------------')

            end_time = time.strftime("%Y-%m-%dT%H:%M:%S")


run_prediction('llama-3-1-70b', 'fakenews-net')
run_prediction('llama-3-1-70b', 'isot')
run_prediction('llama-3-1-70b', 'politifact_factcheck_data')
run_prediction('llama-3-1-70b', 'welfake')

run_prediction('deep-seek-32B', 'covid')
run_prediction('deep-seek-32B', 'liar')
run_prediction('deep-seek-32B', 'fakenews-net')
```

Listing B.1: Código de execução das predições usando LLMs

```
import json
import csv
import re


def generate_instruction_prompt(prompt, model_name, question):
    instruction_prompt = prompt.replace("{{message}}", question)
    default_parameters = {
        "temperature": 0.01,
        "return_full_text": False,
    }

    switch_case = {
        "gemma": {
            "prompt":
    f"<bos>\n<start_of_turn>user\n{instruction_prompt}<end_of_turn>\n<start_of_convers
            "parameters": default_parameters,
        },
        # for llama3
        "llama3": {
            "prompt":
    f"<|begin_of_text|><|start_header_id|>user<|end_header_id|>\n{instruction_prompt}<
            "parameters": {
                "temperature": 0.01,
                "details": False,
                "return_full_text": False,
                "max_new_tokens": 256,
                "stop": "<|eot_id|>",
            },
```

```
        },
        "mistral": {
            "prompt": f"<s>[INST]{instruction_prompt}[/INST]",
            "parameters": default_parameters,
        },
        "falcon": {
            "prompt": f"User: {instruction_prompt}\nAssistant:",
            "parameters": default_parameters,
        },
        "deepseek": {
            "messages": [
                {
                    "role": "user",
                    "content": f"{instruction_prompt}"
                }
            ],
            "max_tokens": 1024
        }
    }

    if model_name not in switch_case:
        errMessage = "Model " + model_name + " not found"
        raise ValueError(errMessage)
    return switch_case[model_name]


def prepare_prompt_list(prompt_path = "prompts/prompts.json"):
    prompt_list = {}
    with open(prompt_path, "r") as read_file:
        prompts = json.load(read_file)

        bases = prompts["base"]
        definitions = prompts["definitions"]

        for base_key, base in bases.items():
            for definition_key, definition in definitions.items():
                prompt = f"\n{definition}\n\n{base}"
                # create a list of prompts for each key
                if base_key not in prompt_list:
                    prompt_list[base_key] = []
                prompt_list[base_key].append(
                    {"prompt": prompt, "definition": definition_key}
                )

    return prompt_list


```

```
def process_csv(input_file, output_file, isMistral=False):
    with open(input_file, "r") as csv_in, open(output_file, "w", newline="") as csv_out:
        reader = csv.reader(csv_in)
        writer = csv.writer(csv_out)

        # Write the header to the output file
        writer.writerow(next(reader))

        for row in reader:
            # Use regex to capture 'Toxic', 'Not-Toxic' or Neutral in the prediction
            toxic_match = re.search(r"^['\"]?(Toxic)['\"]?$", row[1], re.IGNORECASE)
            not_toxic_match = re.search(
                r"^['\"]?(Not-Toxic)['\"]?$", row[1], re.IGNORECASE
            )
            neutral_match = re.search(r"^['\"]?(Neutral)['\"]?$", row[1], re.IGNORECASE)

            if toxic_match or not_toxic_match or (neutral_match and isMistral):
                if toxic_match and not_toxic_match:
                    row[1] = -1
                elif toxic_match:
                    row[1] = 1
                elif not_toxic_match:
                    row[1] = 0
                else:
                    row[1] = 0

                if isMistral and neutral_match:
                    row[1] = 0
            else:
                # If not found, replace the prediction with an empty string
                row[1] = -1
            writer.writerow(row)


def get_generated_text(res, isNew, model_family):
    generatedText = ""
    # print('Response:', res)
    # print('isNew:', isNew)
    print('model_family:', model_family)
    print('res:', res)

```

```
    if model_family == 'deepseek':
        return res["choices"][0]["message"]["content"]

    if isNew == True:
        if model_family == "llama3":
            generated_text = res["generated_text"]
        else:
            generated_text = res[0]["generated_text"]
    else:
        res = res.decode("utf-8")
        res = json.loads(res)
        generated_text = res[0]["generated_text"]

    return generated_text




def extract_from_file_name(file_name):
    file_name = file_name.replace("predictions-", "")

    # get the valuens until the first Instruct-

    file_name = file_name.split("-Instruct-")
    model_name = file_name[0] + "-Instruct"

    # remove the date
    prompt_strategy = file_name[1].split("-2024")[0]

    return model_name, prompt_strategy
```

Listing B.2: Código de execução das predições usando LLMs

O código completo se encontra no GitHub [1]

[1] https://github.com/douglascastrorj/llm-dissertation

Lista de Comentários