



UNIVERSIDADE ESTADUAL PAULISTA  
“JÚLIO DE MESQUITA FILHO”  
Campus de São José do Rio Preto

Otávio Augusto Teixeira

# **Estudo e comparação de Modelos de língua para detecção de fake news em português**

São José do Rio Preto

2025

Otávio Augusto Teixeira

## **Estudo e comparação de Modelos de língua para detecção de fake news em português**

Monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: Agência (Nº do Processo)

Universidade Estadual Paulista "Júlio de Mesquita Filho" - (UNESP)

Instituto de Biociências, Letras e Ciências Exatas - (IBILCE)

Departamento de Ciências de Computação e Estatística

Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Lucas Correia Ribas

São José do Rio Preto

2025

Otávio Augusto Teixeira

## **Estudo e comparação de Modelos de língua para detecção de fake news em português**

Monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: Agência (Nº do Processo)

Banca Examinadora:

**Prof. Dr. Lucas Correia Ribas**

UNESP - Câmpus de São José do Rio Preto  
Orientador

**Prof. Dr. Arnaldo Cândido Júnior**

UNESP - Câmpus de São José do Rio Preto

**Prof. Dr. Eng. Rodrigo Capobianco Guido**

UNESP - Câmpus de São José do Rio Preto

São José do Rio Preto

28/11/2025

Dedico esse trabalho aos meus pais e familiares, que fizeram de tudo para que eu pudesse concluir este sonho e sempre acreditaram em mim.

# Agradecimentos

Gostaria de começar agradecendo à minha família inteira, por todo o apoio durante esses quatro anos de curso. Principalmente à minha mãe, que fez de tudo para que eu estivesse bem física e mentalmente. Muito obrigado, nunca vou esquecer os esforços feitos por vocês.

Agradeço ao meu orientador, Lucas Ribas, que me ajudou desde a concepção do tema até a finalização deste trabalho, sempre disponível para me receber quando precisei. Considero-o, além de um professor exemplar, um amigo pelo qual tenho profunda admiração.

Gostaria de dar um agradecimento especial ao Leandro, Luan, Mateus e Yhan, que fizeram esses quatro anos de curso serem mais leves e divertidos, compartilhando conhecimentos e experiências. Muito obrigado por tudo.

Por fim, agradeço à Universidade Estadual Paulista (UNESP), ao Instituto de Biociências, Letras e Ciências Exatas (IBILCE) e a todos os professores que fizeram parte da minha formação, pelos excelentes ensinamentos, pela paciência e por toda a infraestrutura disponibilizada.

Muito obrigado a todas essas pessoas, sem vocês nada disso seria possível!

*"However difficult life may seem, there is always something you can do and succeed at. It matters that you don't just give up."  
(Stephen Hawking)*

# Resumo

A disseminação de fake news representa uma ameaça crescente à democracia e ao debate público informado. Este trabalho investiga a eficácia de Grandes Modelos de Linguagem (LLMs) modernos na detecção automática de fake news em língua portuguesa, comparando seu desempenho com técnicas tradicionais de representação vetorial. Foram avaliados mais de uma dezena de modelos de embedding, desde abordagens clássicas como TF-IDF e Word2Vec até LLMs de última geração, incluindo modelos de código aberto (BERTimbau, SFR-Embedding-Mistral, SERAFIM-900M-PT) e proprietários (OpenAI text-embedding-3-small e Google embedding - 001), combinados com três classificadores de aprendizado de máquina: SVM, Random Forest e Regressão Logística. Os experimentos foram conduzidos sobre o corpus FakeRecogna, composto por 11.902 notícias brasileiras balanceadas entre verdadeiras e falsas. Os resultados demonstram que os embeddings gerados por LLMs superam significativamente as técnicas tradicionais. O melhor desempenho foi alcançado pela combinação do OpenAI text-embedding-3-small com Regressão Logística otimizada, atingindo F1-Score de 98,32%, estabelecendo um novo patamar para a tarefa no idioma português. A pesquisa confirmou ainda que modelos baseados em Transformers beneficiam-se da manutenção de stopwords durante o pré-processamento, contrariando práticas convencionais aplicadas a técnicas estatísticas. Este estudo contribui para o avanço das pesquisas sobre verificação automática de notícias em português, fornecendo um benchmark comparativo abrangente e direcionamentos práticos para o combate à desinformação.

**Palavras-chave:** Fake news. Detecção de desinformação. Grandes modelos de linguagem. Processamento de linguagem natural. Embeddings. Aprendizado de máquina.

# Abstract

The spread of fake news represents a growing threat to democracy and informed public debate. This work investigates the effectiveness of modern Large Language Models (LLMs) in the automatic detection of fake news in Portuguese, comparing their performance with traditional vector representation techniques. More than a dozen embedding models were evaluated, from classic approaches such as TF-IDF and Word2Vec to state-of-the-art LLMs, including open-source models (BERTimbau, SFR-Embedding-Mistral, SERAFIM-900M-PT) and proprietary models (OpenAI text-embedding-3-small and Google embedding-001), combined with three machine learning classifiers: SVM, Random Forest, and Logistic Regression. The experiments were conducted on the FakeRecogna corpus, composed of 11,902 balanced Brazilian news articles between true and false. The results demonstrate that embeddings generated by LLMs significantly outperform traditional techniques. The best performance was achieved by the combination of OpenAI text-embedding-3-small with optimized Logistic Regression, reaching an F1-Score of 98.32%, establishing a new benchmark for the task in Portuguese. The research also confirmed that Transformer-based models benefit from maintaining stopwords during preprocessing, contradicting conventional practices applied to statistical techniques. This study contributes to the advancement of research on automatic news verification in Portuguese, providing a comprehensive comparative benchmark and practical guidelines for combating misinformation.

**Keywords:** *Fake News. Disinformation Detection. Large Language Models. Natural Language Processing. Embeddings. Machine Learning.*

# Sumário

	<b>Lista de ilustrações</b>	<b>11</b>
	<b>Lista de tabelas</b>	<b>12</b>
<b>1</b>	<b>INTRODUÇÃO</b>	<b>15</b>
1.1	<b>Motivação e Contextualização</b>	<b>15</b>
1.2	<b>Objetivos</b>	<b>16</b>
1.2.1	Objetivo Geral	16
1.2.2	Objetivos Específicos	16
1.2.3	Hipóteses de Pesquisa	16
1.3	<b>Organização</b>	<b>17</b>
<b>2</b>	<b>FUNDAMENTAÇÃO E REVISÃO DA LITERATURA</b>	<b>18</b>
2.1	<b>Considerações Iniciais</b>	<b>18</b>
2.2	<b>O Fenômeno das Fake News</b>	<b>18</b>
2.2.1	Definição e Caracterização	18
2.2.2	Mecanismos de Disseminação e Impacto	19
2.3	<b>Deteccção Automática e o Processamento de Linguagem Natural (PLN)</b>	<b>21</b>
2.3.1	O Desafio da Deteccção	21
2.3.2	Aprendizado de Máquina (AM) para Classificação de Texto	23
2.4	<b>A Representação Vetorial de Texto (Embeddings)</b>	<b>25</b>
2.4.1	Abordagens Tradicionais (Baselines)	27
2.4.1.1	TF-IDF (Term Frequency-Inverse Document Frequency)	27
2.4.1.2	Word2Vec	28
2.4.2	A Revolução dos Transformers e dos Grandes Modelos de Linguagem (LLMs)	30
2.4.2.1	A Arquitetura Transformer e o Mecanismo de Atenção	30
2.4.2.2	Grandes Modelos de Linguagem (LLMs)	33
2.5	<b>Algoritmos de Classificação Utilizados</b>	<b>34</b>
2.5.1	Máquinas de Vetores de Suporte (SVM)	34
2.5.2	Florestas Aleatórias (Random Forest)	36
2.5.3	Regressão Logística	38
2.6	<b>Métricas de Avaliação de Classificadores</b>	<b>39</b>
2.6.1	Acurácia	40
2.6.2	Precisão	40
2.6.3	Revocação (Recall ou Sensibilidade)	41

2.6.4	F1-Score . . . . .	41
<b>2.7</b>	<b>Trabalhos Correlatos . . . . .</b>	<b>41</b>
2.7.1	Abordagens Iniciais e Datasets em Português . . . . .	41
2.7.2	Deteção com Aprendizado de Máquina Clássico . . . . .	42
2.7.3	O Uso de Deep Learning e Modelos Pré-Transformers . . . . .	43
2.7.4	LLMs para Análise de Texto e Desinformação . . . . .	44
<b>2.8</b>	<b>Síntese da Literatura e Justificativa da Pesquisa . . . . .</b>	<b>45</b>
<b>3</b>	<b>METODOLOGIA . . . . .</b>	<b>48</b>
<b>3.1</b>	<b>Conjunto de dados (Corpus) . . . . .</b>	<b>48</b>
<b>3.2</b>	<b>Pre-processamento dos dados . . . . .</b>	<b>50</b>
3.2.1	Abordagem 1: Limpeza Extensiva com Remoção de Stopwords . . . . .	50
3.2.2	Abordagem 2: Limpeza Moderada com Manutenção de Stopwords . . . . .	50
<b>3.3</b>	<b>Geração das Representações Vetoriais (Embeddings) . . . . .</b>	<b>51</b>
3.3.1	Abordagens tradicionais . . . . .	51
3.3.2	Abordagens Baseadas em Grandes Modelos de Linguagem (LLMs) . . . . .	52
<b>3.4</b>	<b>Modelos de Classificação . . . . .</b>	<b>54</b>
3.4.1	Máquinas de Vetores de Suporte (SVM) . . . . .	55
3.4.2	Florestas Aleatórias (Random Forest) . . . . .	55
3.4.3	Regressão Logística (Logistic Regression) . . . . .	55
3.4.4	Justificativa da Escolha dos Classificadores . . . . .	55
<b>3.5</b>	<b>Desenho Experimental e Avaliação . . . . .</b>	<b>56</b>
3.5.1	Treinamento e Validação dos Dados . . . . .	56
3.5.2	Métricas de Avaliação . . . . .	57
3.5.2.1	Acurácia . . . . .	57
3.5.2.2	Precisão, Revocação (Recall) e F1-Score . . . . .	57
3.5.3	Protocolo Experimental Comparativo . . . . .	58
3.5.4	Otimização de Hiperparâmetros dos Melhores Modelos . . . . .	59
<b>3.6</b>	<b>Ferramentas e Ambiente de Desenvolvimento . . . . .</b>	<b>60</b>
3.6.1	Software e Bibliotecas . . . . .	60
3.6.2	Ambiente de Hardware . . . . .	61
<b>4</b>	<b>RESULTADOS . . . . .</b>	<b>63</b>
<b>4.1</b>	<b>Análise de Desempenho dos Modelos de Baseline . . . . .</b>	<b>63</b>
<b>4.2</b>	<b>Desempenho dos Grandes Modelos de Linguagem (LLMs) . . . . .</b>	<b>65</b>
4.2.1	Análise Geral dos LLMs . . . . .	65
4.2.2	Modelos de maior escala e acessíveis via API . . . . .	66
4.2.3	Análise Qualitativa dos Erros de Classificação . . . . .	67
4.2.3.1	Matriz de Confusão e Distribuição dos Erros . . . . .	67
4.2.3.2	Análise dos Falsos Positivos . . . . .	68

4.2.3.3	Análise dos Falsos Negativos . . . . .	70
4.3	<b>Análise Comparativa e Discussão . . . . .</b>	<b>71</b>
5	<b>CONCLUSÃO . . . . .</b>	<b>74</b>
	 <b>APÊNDICES</b>	 <b>76</b>
	<b>APÊNDICE A – CÓDIGOS-FONTE DO FRAMEWORK EXPERI- MENTAL . . . . .</b>	<b>77</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>80</b>

# Lista de ilustrações

Figura 2.2.1–Participação das fontes de acesso em sites de notícias confiáveis e sites de fake news nos EUA. . . . .	20
Figura 2.3.1–Diferença entre métodos tradicionais e de métodos que usam Deep Learning para classificação de texto. . . . .	22
Figura 2.3.2–Fluxograma do processo de classificação de texto com aprendizado de máquina. . . . .	24
Figura 2.4.1–Visualização de Word Embeddings em 2D . . . . .	26
Figura 2.4.2–Relações Semânticas em Espaço Vetorial (Word2Vec). . . . .	29
Figura 2.4.3–A arquitetura do modelo Transformer. . . . .	31
Figura 2.4.4–Mecanismos de (esquerda) Scaled Dot-Product Attention e (direita) Multi-Head Attention. . . . .	32
Figura 2.4.5–Padrões de atenção visualizados em BERT. Diferentes cabeças de atenção aprendem a focar em diferentes tipos de relações entre os tokens. . . .	33
Figura 2.4.6–Relação entre Emergência e Homogeneização nos Modelos Fundamentais	34
Figura 2.7.1–Fluxograma ilustrando a estrutura geral dos modelos de detecção de fake news baseados em aprendizado de máquina clássico . . . . .	43
Figura 3.0.1–Pipeline metodológico para detecção de <i>fake news</i> utilizando <i>embeddings</i>	49
Figura 4.2.1–Matriz de confusão do modelo SFR-Embedding-Mistral com classificador SVM . . . . .	68

# Lista de tabelas

Tabela 1	– Classificação de diferentes formas de desinformação segundo autenticidade, intenção e formato noticioso . . . . .	19
Tabela 2	– Tabela comparativa dos principais trabalhos correlatos em detecção de fake news. . . . .	46
Tabela 3	– Tabela comparativa dos modelos de linguagem utilizados para geração de embeddings. . . . .	54
Tabela 4	– Resultados dos Modelos de Baseline na tarefa de detecção de fake news.	63
Tabela 5	– Melhores resultados dos LLMs para detecção de fake news (Combinação: completo). . . . .	65
Tabela 6	– Decomposição da matriz de confusão do modelo SFR-Embedding-Mistral	68
Tabela 7	– Impacto da otimização de hiperparâmetros no desempenho dos modelos	71

# Lista de algoritmos

# Lista de abreviaturas e siglas

LDA      *Linear Discriminant Analysis*

# 1 Introdução

## 1.1 Motivação e Contextualização

A disseminação de informações fabricadas, popularmente conhecidas como fake news, tornou-se um fenômeno proeminente na era digital, representando uma ameaça significativa aos processos democráticos, à confiança pública e ao jornalismo (WAISBORD, 2018; ZHOU; ZAFARANI, 2020). De acordo com (GELFERT, 2018; ALLCOTT; GENTZKOW, 2017), as fake news podem ser compreendidas como textos que se passam por notícias verdadeiras, mas que contêm informações falsas de forma intencional, com potencial de enganar o leitor. A facilidade de criação e partilha de conteúdo online, aliada à dificuldade que muitos cidadãos enfrentam em discernir informações verídicas de falsas, exacerba o problema (CONROY; RUBIN; CHEN, 2015). No contexto brasileiro, as eleições presidenciais de 2018 foram um marco notório do impacto das fake news, com a propagação massiva de desinformação através de aplicativos como WhatsApp e redes sociais como o Facebook, levantando questões sobre a integridade do processo eleitoral e a resiliência da democracia a tais manipulações (ITUASSU et al., 2023; PEREIRA et al., 2022). Adicionalmente, o avanço recente de modelos de linguagem baseados em Inteligência Artificial, capazes de gerar textos cada vez mais sofisticados e convincentes, introduz um novo nível de complexidade ao desafio da detecção.

Diante desse cenário, a detecção automática de fake news emergiu como uma área de pesquisa crucial. Esforços anteriores têm explorado diversas técnicas de Processamento de Linguagem Natural (PLN) e Aprendizado de Máquina (AM) para identificar notícias falsas. Abordagens tradicionais frequentemente envolvem a extração de características linguísticas e o uso de modelos como Máquinas de Vetores de Suporte (SVM), Florestas Aleatórias ou redes neurais, alimentados por representações vetoriais (embeddings) geradas por técnicas como TF-IDF, Word2Vec (MIKOLOV et al., 2013a) ou modelos baseados em Transformers mais antigos como o BERT (KALIYAR; GOSWAMI; NARANG, 2021a; WANG et al., 2018). No contexto específico do português brasileiro, trabalhos como o de (CARVALHO et al., 2020) buscaram criar léxicos e ferramentas para auxiliar nessa tarefa. Contudo, o rápido desenvolvimento dos Grandes Modelos de Linguagem (LLMs), arquiteturas de aprendizado profundo baseadas na arquitetura Transformer (VASWANI et al., 2017), que demonstram capacidades sem precedentes na compreensão e geração de linguagem, apresenta uma nova fronteira tecnológica. A lacuna existente reside na investigação sistemática e comparativa da eficácia dessas LLMs de última geração para a tarefa específica de detecção de fake news em português, avaliando se suas representações semânticas mais ricas se traduzem em melhorias significativas de desempenho em relação às abordagens estabelecidas.

## 1.2 Objetivos

### 1.2.1 Objetivo Geral

Investigar a eficácia de Grandes Modelos de Linguagem (LLMs) modernos na tarefa de detecção de *fake news* em língua portuguesa, comparando seu desempenho com técnicas tradicionais de representação vetorial.

### 1.2.2 Objetivos Específicos

1. Avaliar o desempenho de diferentes LLMs na geração de *embeddings* aplicados à detecção de *fake news* em português, incluindo modelos de código aberto e proprietários.
2. Comparar quantitativamente o desempenho de classificadores de aprendizado de máquina (SVM, Random Forest e Regressão Logística) baseados em *embeddings* de LLMs com os obtidos por técnicas tradicionais, como TF-IDF e Word2Vec.
3. Identificar quais modelos ou arquiteturas de LLM apresentam melhor capacidade de representação semântica para textos jornalísticos em português brasileiro, considerando diferentes estratégias de pré-processamento.
4. Investigar o impacto da otimização de hiperparâmetros no desempenho dos modelos de classificação quando combinados com diferentes técnicas de representação vetorial.
5. Contribuir para o avanço das pesquisas sobre verificação automática de notícias em português, estabelecendo um *benchmark* comparativo abrangente e sugerindo abordagens mais eficazes para o combate à desinformação.

### 1.2.3 Hipóteses de Pesquisa

Espera-se que a utilização de *embeddings* gerados por LLMs modernos resulte em um aumento significativo no desempenho dos classificadores para a detecção de *fake news* em português, quando comparados às técnicas tradicionais como TF-IDF e Word2Vec. Antecipa-se que métricas como acurácia, precisão, revocação e F1-score sejam superiores nos modelos baseados em LLMs, devido à sua capacidade aprimorada de capturar nuances semânticas e contextuais complexas presentes nos textos (ALLCOTT; GENTZKOW, 2017; ZHOU; ZAFARANI, 2020).

Além disso, espera-se que a análise comparativa entre diferentes LLMs forneça percepções valiosas sobre quais arquiteturas ou modelos específicos são mais adequados para essa tarefa no idioma português, e que a manutenção de *stopwords* seja benéfica para modelos contextuais baseados em Transformers, ao contrário das práticas convencionais aplicadas a técnicas estatísticas tradicionais.

## 1.3 Organização

O restante deste estudo está organizado da seguinte forma: O Capítulo 2 apresentará uma revisão da literatura detalhada, abordando o estado da arte na detecção de fake news, as técnicas de geração de embeddings (tradicionais e baseadas em LLMs) e os algoritmos de classificação relevantes. O Capítulo 3 descreverá a metodologia empregada, incluindo detalhes sobre o corpus utilizado, o pré-processamento dos dados, os modelos LLMs selecionados, a geração dos embeddings, as arquiteturas dos classificadores e o protocolo experimental para treinamento e avaliação. O Capítulo 4 apresentará e discutirá os resultados obtidos nos experimentos comparativos. Finalmente, o Capítulo 5 concluirá o trabalho, resumizando as principais descobertas, discutindo as implicações dos resultados e sugerindo direções para pesquisas futuras na área.

## 2 Fundamentação e Revisão da Literatura

### 2.1 Considerações Iniciais

Este capítulo estabelece as bases conceituais e o contexto acadêmico para o presente trabalho de conclusão de curso. A primeira parte, a Fundamentação Teórica, detalha os conceitos essenciais que sustentam a pesquisa, desde o fenômeno das fake news até as tecnologias de Processamento de Linguagem Natural, técnicas de aprendizado de máquina e os grandes modelos de linguagem utilizados. A segunda parte, a Revisão da Literatura, analisa criticamente os trabalhos anteriores na área de detecção de fake news, identificando o avanço e as contribuições que já foram alcançadas até o presente momento e as lacunas existentes, de modo a posicionar e justificar a contribuição desta monografia de conclusão de curso.

### 2.2 O Fenômeno das Fake News

#### 2.2.1 Definição e Caracterização

A conceituação de fake news é complexa e multifacetada, carecendo de uma definição universalmente aceita na literatura acadêmica (GELFERT, 2018; ZHOU; ZAFARANI, 2020). No entanto, um ponto de partida funcional é a definição proposta por Allcott e Gentzkow (2017), que caracterizam fake news como "artigos de notícias que são intencional e verificavelmente falsos, e que podem enganar os leitores". Esta definição destaca dois pilares fundamentais: a falsidade do conteúdo e a intenção de enganar, que são cruciais para diferenciar fake news de outros distúrbios informacionais.

Para aprofundar essa caracterização, (JR.; LIM; LING, 2018) propõe uma tipologia baseada em duas características principais: o nível de faticidade (o grau em que o conteúdo se baseia em fatos) e o nível de intenção de enganar. Utilizando este framework, é possível distinguir as fake news de fenômenos correlatos. Por exemplo, a sátira e a paródia possuem baixa faticidade, mas também uma baixa intenção de enganar, pois operam sob um estilo literário que tem como finalidade o humor e não a fonte de uma informação. Em contraste, os erros jornalísticos genuínos, embora possam conter informações falsas, carecem do elemento de má intenção, sendo fundamentalmente distintos da fabricação maliciosa (JR.; LIM; LING, 2018; ALLCOTT; GENTZKOW, 2017).

A literatura também diferencia fake news de conceitos mais amplos como desinformação (disinformation) e más-informação (misinformation). Segundo (ZHOU; ZAFARANI, 2020; LAZER et al., 2018; WAISBORD, 2018), desinformação refere-se à produção deliberada e divulgação de informações falsas com propósito de enganar, enquanto as más-informação

refere-se na divulgação dessas informações sem o intuito de causar mal ou enganar alguém, porém possui também grande risco, já que pode ser disponibilizado rapidamente e em massa nas redes sociais. Podemos observar as diferenças estruturais de conceitos relacionados a fake news na Tabela 1

Tabela 1 – Classificação de diferentes formas de desinformação segundo autenticidade, intenção e formato noticioso

<b>Conceito</b>	<b>Autenticidade</b>	<b>Intenção</b>	<b>É notícia?</b>
Notícia enganosa	Não factual	Enganar	Sim
Notícia falsa	Não factual	Indefinida	Sim
Notícia satírica	Não unificada	Entreter	Sim
Desinformação	Não factual	Enganar	Indefinida
Informação errada	Não factual	Indefinida	Indefinida
Recorte seletivo	Comumente factual	Enganar	Indefinida
Isca de clique	Indefinida	Enganar	Indefinida
Boato	Indefinida	Indefinida	Indefinida

**Fonte:** Adaptado de (ZHOU; ZAFARANI, 2020)

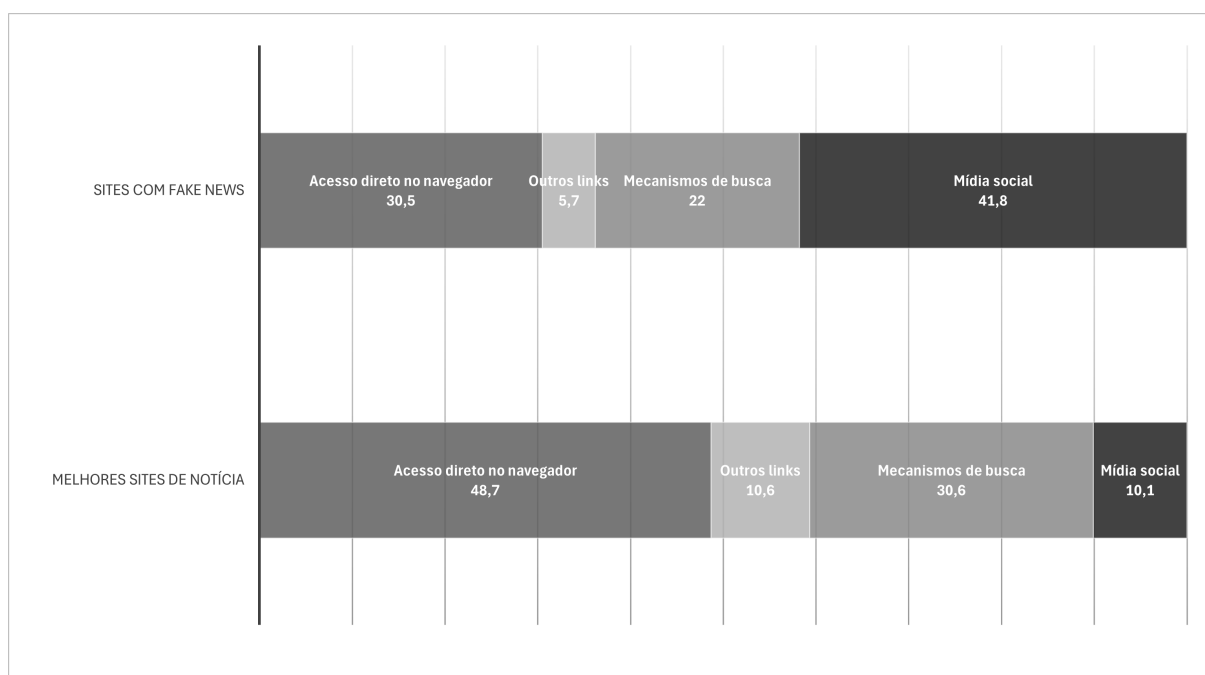
Nesse contexto, as fake news são um subtipo específico de desinformação, caracterizado por mimetizar o formato e o estilo de notícias jornalísticas legítimas para obter credibilidade e potencializar sua disseminação (GELFERT, 2018). (GELFERT, 2018) argumenta que o que torna as fake news contemporâneas particularmente nocivas é o fato de serem "enganosas por design". Esta expressão não se refere apenas à intenção individual do criador, mas a características sistêmicas no design das fontes e dos canais de distribuição que são otimizados para manipular os processos cognitivos do público. Portanto, para os fins deste trabalho, fake news são entendidas como conteúdo fabricado que emula o formato jornalístico, com alta intenção de enganar e baixa faticidade, projetado para ser disseminado como se fosse uma notícia verídica.

### 2.2.2 Mecanismos de Disseminação e Impacto

A proliferação das fake news é impulsionada por um encontro de fatores tecnológicos, inerentes ao ambiente da mídia digital, e vulnerabilidades psicológicas da cognição humana, que vai desde a falta de estudo até a maldade. O ambiente da mídia social representa uma mudança estrutural drástica em relação aos meios de comunicação tradicionais, pois permite que o conteúdo seja republicado entre os usuários sem filtro ou verificação sobre a notícia de forma extremamente veloz (ALLCOTT; GENTZKOW, 2017). Essa arquitetura tecnológica, aliada ao baixo custo de produção e à facilidade de monetização através de publicidade, cria um terreno fértil para a disseminação de desinformação (WAISBORD, 2018; LAZER et al., 2018).

Os fatores tecnológicos são amplificados pelos algoritmos que governam as plataformas. Projetados para maximizar o engajamento do usuário, esses sistemas tendem a promover conteúdo sensacionalista e emocionalmente carregado, características comuns em fake news, já que conteúdos como esse chamam mais atenção das massas (ZHOU; ZAFARANI, 2020). Isso contribui para a formação de bolhas, nas quais as crenças dos usuários são constantemente reforçadas, e dentro dessas bolhas os usuários ficam limitados às notícias relacionadas a essas crenças, o que diminui a exposição a perspectivas divergentes. Adicionalmente, a disseminação é acelerada por atores não humanos, como os bots (contas automatizadas), que podem inflar artificialmente a popularidade de uma notícia, aumentando sua visibilidade e alcance (LAZER et al., 2018). A análise de (ALLCOTT; GENTZKOW, 2017) sobre o tráfego de sites de notícias durante a eleição de 2016 nos EUA demonstra que, enquanto os principais veículos de imprensa recebem cerca de 10% de seu tráfego de mídias sociais, os sites de fake news dependem massivamente delas, com mais de 40% de seu tráfego vindo dessas fontes, como mostrado na Figura 2.2.2.

Figura 2.2.1 – Participação das fontes de acesso em sites de notícias confiáveis e sites de fake news nos EUA.



Fonte: (ALLCOTT; GENTZKOW, 2017)

Esses mecanismos tecnológicos exploram vieses cognitivos bem documentados na psicologia. O viés de confirmação, a tendência de buscar, interpretar e favorecer informações que confirmam crenças preexistentes, é um dos principais motores do consumo e compartilhamento de fake news (LAZER et al., 2018; ZHOU; ZAFARANI, 2020). A exposição seletiva faz com que indivíduos prefiram ativamente informações alinhadas ideologicamente, enquanto o efeito de validade (ou familiaridade) sugere que a exposição repetida a uma informação, mesmo

que falsa, aumenta a probabilidade de que ela seja julgada como verdadeira (ZHOU; ZAFARANI, 2020). O impacto combinado desses mecanismos é profundo, resultando na erosão da confiança nas instituições tradicionais, como a mídia e a ciência, no aprofundamento da polarização afetiva e política, e na degradação do debate público (WAISBORD, 2018; LAZER et al., 2018). Em última análise, ao minar a base de fatos compartilhados necessária para o funcionamento de uma democracia, as fake news representam uma ameaça direta à capacidade dos cidadãos de tomar decisões informadas e ao próprio processo democrático (ALLCOTT; GENTZKOW, 2017).

## 2.3 Detecção Automática e o Processamento de Linguagem Natural (PLN)

A crescente ameaça representada pela desinformação impulsionou a detecção automática de fake news a se tornar uma área de pesquisa crucial no campo do Processamento de Linguagem Natural (PLN) (ZHOU; ZAFARANI, 2020). A tarefa consiste em desenvolver sistemas computacionais capazes de classificar um texto noticioso como verdadeiro ou falso, um desafio que transcende a simples verificação de palavras-chave e exige uma profunda compreensão semântica e contextual do conteúdo (KALIYAR; GOSWAMI; NARANG, 2021a). A complexidade da tarefa reside no fato de que as fake news são, por definição, "enganosas por design", mimetizando o estilo, o formato e a linguagem de notícias legítimas para maximizar sua credibilidade e potencial de disseminação (GELFERT, 2018).

### 2.3.1 O Desafio da Detecção

O principal desafio na detecção de fake news reside na sua sutileza linguística. Diferente de um erro factual explícito, a falsidade é frequentemente construída através da manipulação de contexto, apelo emocional, uso de linguagem carregada e distorção de fatos verídicos (CONROY; RUBIN; CHEN, 2015). Para um sistema de PLN, isso significa que a análise superficial do texto é insuficiente. A tarefa exige a capacidade de capturar detalhes, identificar inconsistências lógicas e, principalmente, compreender o significado do texto em um nível profundo, de forma análoga à avaliação crítica que um leitor humano realizaria.

Essa complexidade transforma a detecção de fake news em um problema que espelha o que (CAMBRIA et al., 2017) descrevem como o "problema-mala" (suitcase problem) da análise de sentimentos. A tarefa não é um simples problema de classificação binária, mas um conjunto multifacetado de subproblemas de PLN que precisam ser resolvidos em cascata para se alcançar uma compreensão de nível humano. A análise deve operar em múltiplos níveis: sintático, semântico e pragmático. No nível sintático, os modelos precisam lidar com a normalização de textos informais, comuns no compartilhamento em redes sociais. No nível semântico, o desafio é a ambiguidade, onde palavras e frases podem ter múltiplos significados que são explorados

para criar narrativas enganosas (JUSOH, 2018). Finalmente, no nível pragmático, o modelo deve inferir a intenção do autor, uma tarefa análoga à detecção de sarcasmo, onde o sentido literal do texto contradiz o sentimento implícito (JOSHI; BHATTACHARYYA; CARMAN, 2017).

A evolução das abordagens para enfrentar esse desafio reflete o progresso do próprio campo de PLN. As metodologias tradicionais, baseadas em modelos estatísticos como Naïve Bayes (NB) e Máquinas de Vetores de Suporte (SVM), dependiam intensamente de um processo de engenharia de características. Nesse paradigma, especialistas precisavam projetar e extrair manualmente atributos do texto, como frequências de palavras (Bag-of-Words, TF-IDF), padrões sintáticos e características lexicais (LI et al., 2022). Esse processo, além de ser trabalhoso e de difícil escalabilidade, frequentemente desconsiderava a estrutura sequencial e o contexto profundo do texto, limitando a capacidade do modelo de capturar a complexidade semântica necessária.

Figura 2.3.1 – Diferença entre métodos tradicionais e de métodos que usam Deep Learning para classificação de texto.



Fonte: Adaptado de (LI et al., 2022)

A ascensão do deep learning marcou uma mudança de paradigma. Arquiteturas como Redes Neurais Convolucionais (CNNs) e Recorrentes (RNNs) passaram a aprender representações de características diretamente dos dados, eliminando a necessidade da extração manual. No entanto, o avanço mais significativo veio com os modelos baseados em Transformers, como o BERT e suas variantes. Esses modelos introduziram a capacidade de gerar representações de palavras contextualizadas, superando uma limitação fundamental das abordagens anteriores (POTAMIAS; SIOLAS; STAFYLOPATIS, 2020). Essa capacidade de entender as palavras em seu contexto específico é crucial para a detecção de fake news, onde a incongruência e a manipulação sutil de significado são centrais. A superioridade dos modelos baseados em Transformers em uma vasta gama de tarefas de PLN, incluindo a detecção de linguagem figurada, sugere seu enorme potencial para avançar o estado da arte na identificação de desinformação (POTAMIAS; SIOLAS; STAFYLOPATIS, 2020; LI et al., 2022). A diferença entre os métodos antigos e os atuais está ilustrada na Figura 2.3.1. Contudo, esses modelos complexos trazem consigo o desafio da interpretabilidade o "problema da caixa-preta", tor-

nando difícil explicar o raciocínio por trás de suas classificações, um aspecto crítico para a confiabilidade em aplicações do mundo real (LI et al., 2022).

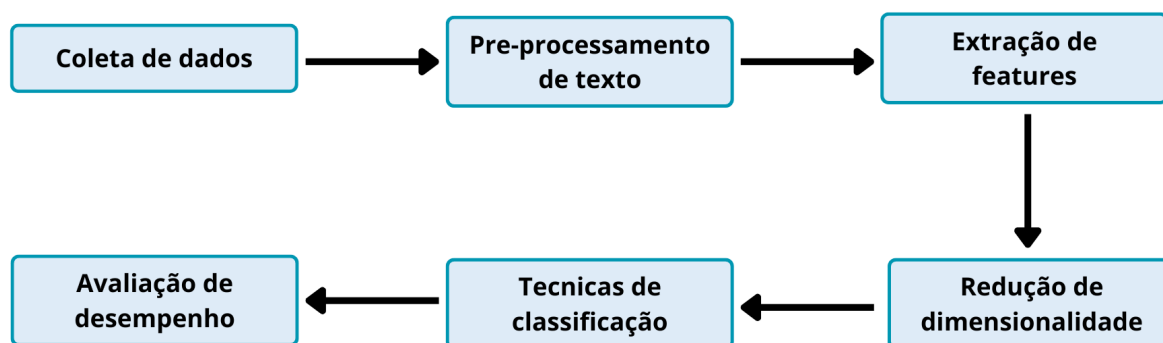
Apesar do avanço dos modelos, a disponibilidade de dados de treinamento de alta qualidade continua sendo um gargalo, especialmente para idiomas como o português. A criação de datasets rotulados exige um processo de verificação de fatos trabalhoso e caro. No entanto, esforços recentes, como a criação do corpus Fake.Br (SILVA et al., 2020; MONTEIRO et al., 2018b), têm sido fundamentais para viabilizar a pesquisa e a avaliação de modelos no contexto brasileiro, fornecendo uma base sólida para estudos comparativos como o proposto nesta monografia de conclusão de curso.

### 2.3.2 Aprendizado de Máquina (AM) para Classificação de Texto

O Aprendizado de Máquina (AM) representa uma mudança de paradigma fundamental na abordagem da classificação de texto. Em contraste com a engenharia de conhecimento, que depende da definição manual de regras por especialistas, o AM utiliza um processo indutivo geral para construir um classificador automaticamente. A premissa central é que o sistema pode "aprender" as características distintivas das categorias a partir de um conjunto de documentos previamente classificados (SEBASTIANI, 2001). Essa abordagem supervisionada permite que o algoritmo infira padrões e relações entre o conteúdo textual e seus respectivos rótulos, resultando em modelos que podem generalizar esse conhecimento para classificar novos documentos nunca vistos. As vantagens são notáveis: uma eficácia comparável à de especialistas humanos, uma economia considerável em termos de trabalho especializado e a portabilidade direta para diferentes domínios e conjuntos de categorias (SEBASTIANI, 2001; IKONOMAKIS; KOTSIANTIS; TAMPAKAS, 2005).

Para que um algoritmo de aprendizado de máquina possa processar dados textuais, é essencial primeiro converter os documentos em um formato numérico estruturado. A abordagem mais comum para essa tarefa é o modelo de espaço vetorial, onde cada documento é representado como um vetor em um espaço multidimensional em que cada dimensão corresponde a um termo (palavra ou n-grama) do vocabulário (LI et al., 2022). O valor de cada dimensão no vetor representa o "peso" ou a importância daquele termo no documento. Técnicas de ponderação como o TF-IDF (Term Frequency-Inverse Document Frequency) são amplamente utilizadas para esse fim, atribuindo pesos maiores a termos que são frequentes em um documento específico, mas raros no restante da coleção, tornando-os bons discriminadores de tópico (KADHIM, 2019). Esse processo de pré-processamento, que também inclui etapas como tokenização, remoção de stopwords e lematização, é um pilar fundamental que transforma o texto não estruturado em um conjunto de características (features) que os modelos de AM podem utilizar para aprender. Esse fluxo está ilustrado na Figura 2.3.2

Figura 2.3.2 – Fluxograma do processo de classificação de texto com aprendizado de máquina.



Fonte: Adaptado de (KADHIM, 2019)

O paradigma de aprendizado supervisionado se baseia em uma divisão rigorosa do corpus de dados disponível para garantir a validade científica e a capacidade de generalização do modelo. O corpus inicial é particionado em, no mínimo, dois conjuntos distintos: o conjunto de treinamento (training set) e o conjunto de teste (test set). O conjunto de treinamento é utilizado para construir o classificador; o algoritmo de aprendizado analisa esses exemplos rotulados para ajustar seus parâmetros internos. O conjunto de teste, por sua vez, é composto por dados que o modelo não viu durante o treinamento e serve exclusivamente para avaliar a eficácia final do classificador. É crucial que os documentos de teste não participem de nenhuma forma da construção do modelo para que a avaliação seja imparcial e realista (SEBASTIANI, 2001). Frequentemente, o conjunto de treinamento é subdividido para criar um conjunto de validação (validation set), que é usado para otimizar os hiperparâmetros do modelo (exemplo, a escolha do kernel em um SVM ou o número de vizinhos em k-NN) antes da avaliação final no conjunto de teste, evitando assim o superajuste (overfitting).

Dentro deste paradigma, a literatura distingue duas grandes eras de modelos de AM para classificação de texto: os tradicionais e os baseados em deep learning. Os modelos tradicionais, como Naïve Bayes, SVM e k-Nearest Neighbors (k-NN), operam sobre as representações vetoriais (exemplo, TF-IDF) geradas na etapa de pré-processamento (KADHIM, 2019; SEBASTIANI, 2001). Em contrapartida, as abordagens de deep learning, como Redes Neurais Convolucionais (CNNs), Recorrentes (RNNs) e, mais notavelmente, os Transformers (base dos LLMs), integram a extração de características ao próprio processo de treinamento. Em vez de depender de características pré-definidas, esses modelos aprendem representações hierárquicas e contextuais dos dados, permitindo a captura de padrões semânticos e sintáticos muito mais complexos. Como aponta (LI et al., 2022), essa capacidade de aprender representações diretamente do texto bruto é a principal vantagem que permite aos modelos de deep learning, como o BERT, alcançar resultados de estado da arte em uma vasta gama de tarefas de PLN.

## 2.4 A Representação Vetorial de Texto (Embeddings)

Os algoritmos de Aprendizado de Máquina, desde os modelos estatísticos clássicos até as redes neurais profundas, operam fundamentalmente com dados numéricos. Eles não conseguem processar texto em sua forma bruta, isto é, como uma sequência de caracteres. Portanto, um passo preliminar e indispensável em qualquer tarefa de Processamento de Linguagem Natural (PLN) é a conversão do texto em uma representação vetorial (DEVLIN et al., 2018). Essa transformação, conhecida como embedding, visa mapear palavras, frases ou documentos inteiros para um espaço vetorial de alta dimensionalidade, onde as relações semânticas e sintáticas do texto original podem ser capturadas e quantificadas por meio de operações matemáticas (COLLOBERT et al., 2011). A qualidade dessa representação é crucial, pois ela constitui a única fonte de informação que o modelo de AM terá para "aprender" a diferenciar, por exemplo, notícias verdadeiras de falsas.

Historicamente, a primeira geração de representações textuais se baseava em abordagens supervisionadas que tratavam cada palavra como um símbolo discreto e distinto. Um método convencional era a representação one-hot, onde cada palavra do vocabulário é mapeada para um vetor esparsos de dimensão  $|D|$  (o tamanho do vocabulário), com o valor 1 na posição correspondente ao índice da palavra e 0 em todas as outras. Embora simples, essa abordagem sofre de sérias limitações: primeiro, a dimensionalidade do vetor cresce linearmente com o vocabulário, tornando-a computacionalmente inviável para grandes volumes de texto; segundo, e mais importante, os vetores resultantes são ortogonais entre si, implicando que não há nenhuma noção de similaridade entre as palavras. Palavras semanticamente relacionadas como "gato" e "felino" seriam tratadas como completamente distintas, um problema conhecido como "maldição da dimensionalidade" (COLLOBERT et al., 2011; TURIAN; RATINOV; BENGIO, 2010).

A solução para essas limitações veio com a ascensão dos embeddings densos e de baixa dimensionalidade, aprendidos a partir de grandes quantidades de texto não rotulado, com base na hipótese distribucional de que palavras que aparecem juntas, nos mesmo contextos, tendem a ter o mesmo significado (LEVY; GOLDBERG, 2014). Modelos como o Word2Vec, com sua arquitetura Skip-gram, generalizaram a noção de contexto para além da simples coocorrência de palavras. O Skip-gram com amostragem negativa (SKIPGRAM), por exemplo, aprende os vetores de palavras ao treinar uma rede neural para distinguir pares de palavra-contexto que realmente ocorrem no texto de pares gerados aleatoriamente (negativos). O objetivo de treinamento, conforme formulado por (LEVY; GOLDBERG, 2014), busca maximizar a seguinte função para um par palavra-contexto  $(w, c)$ :

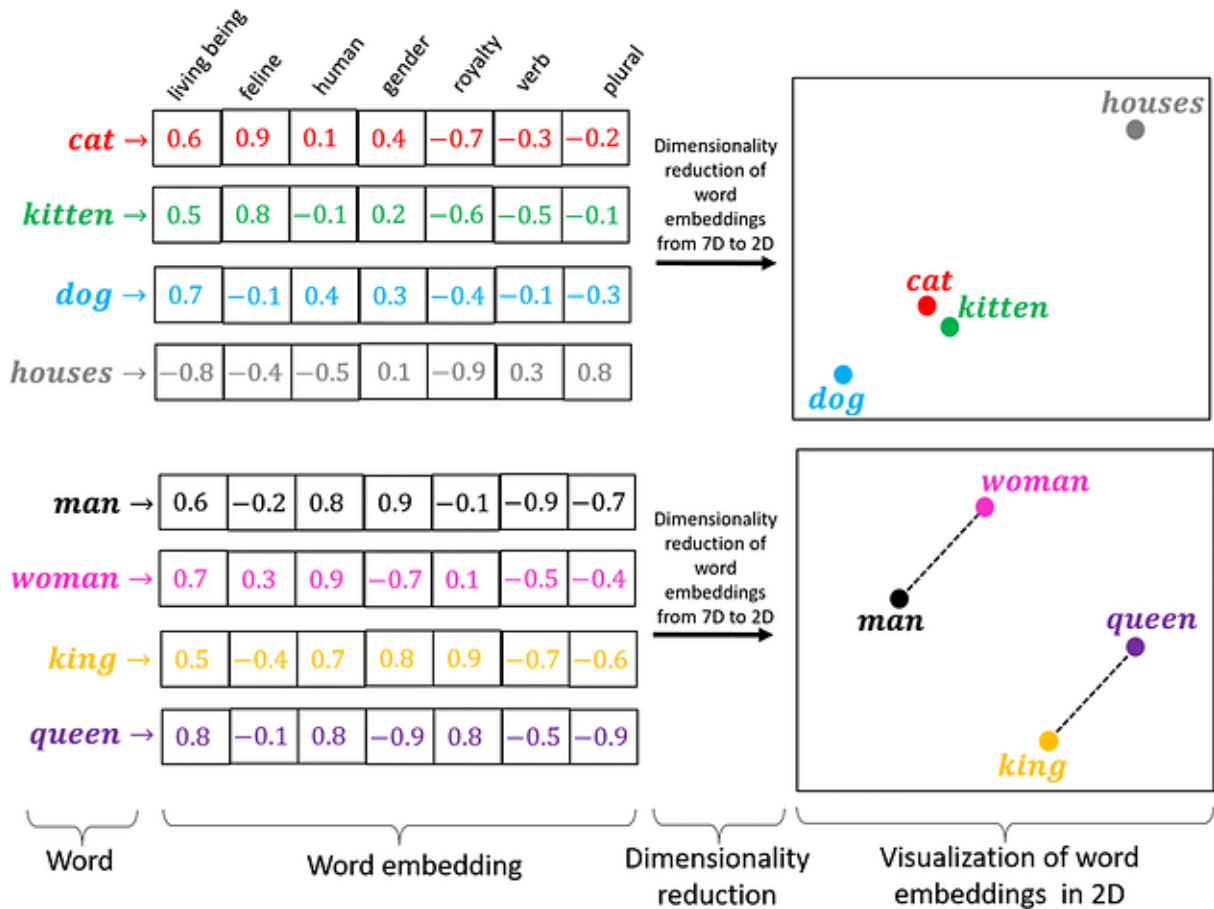
$$\arg \max_{\theta} \left( \sum_{(w,c) \in D} \log \sigma(\mathbf{v}_c \cdot \mathbf{v}_w) + \sum_{(w,c) \in D'} \log \sigma(-\mathbf{v}_c \cdot \mathbf{v}_w) \right) \quad (2.4.1)$$

Onde  $D$  é o conjunto de pares observados,  $D'$  é o conjunto de pares negativos,  $\mathbf{v}_w$  e  $\mathbf{v}_c$  são as representações vetoriais (embeddings) da palavra e do contexto, respectivamente, e

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

é a função sigmoide. Essa formulação força palavras que compartilham contextos a terem vetores próximos no espaço vetorial, capturando assim a similaridade semântica e funcional. A Figura 2.4 ilustra como palavras com significados relacionados são agrupadas nesse espaço.

Figura 2.4.1 – Visualização de Word Embeddings em 2D



Fonte: Hariom Gautam, 2020

O conceito de embeddings foi posteriormente expandido para além das palavras individuais. Modelos de linguagem baseados em redes neurais recorrentes e, mais tarde, Transformers, introduziram a noção de embeddings contextuais. Diferente de um vetor estático como o do Word2Vec, um embedding contextual para uma palavra é dinamicamente gerado com base na sentença inteira em que ela aparece. Modelos como ELMo, GPT e, notavelmente, BERT, utilizam mecanismos de atenção para ponderar a influência de todas as outras palavras na sentença ao construir a representação de um único termo (ETHAYARAJH, 2019; DEVLIN et al., 2018). No BERT, por exemplo, a arquitetura de Transformer bidirecional permite que

a representação de uma palavra seja informada simultaneamente pelo contexto à esquerda e à direita. Essa capacidade de capturar o contexto profundo é o que torna os embeddings gerados por LLMs particularmente poderosos para tarefas complexas como a detecção de fake news, onde a desambiguação e a compreensão da intenção são fundamentais (SHAHEEN; WOHLGENANT; FILTZ, 2020).

## 2.4.1 Abordagens Tradicionais (Baselines)

### 2.4.1.1 TF-IDF (Term Frequency-Inverse Document Frequency)

Uma das técnicas mais fundamentais e amplamente utilizadas para a representação vetorial de texto é o TF-IDF (Term Frequency-Inverse Document Frequency). Trata-se de uma estatística numérica que visa mensurar a importância de um termo em um documento, considerando não apenas sua frequência local, mas também sua distribuição em toda a coleção de documentos (JALILIFARD et al., 2020). O TF-IDF opera sob a intuição de que as palavras mais significativas para caracterizar o conteúdo de um texto são aquelas que aparecem com alta frequência nesse texto específico (componente TF), mas que são raras no conjunto geral de documentos (componente IDF) (JONES, 1988).

A formulação padrão do TF-IDF para um termo  $t_i$  em um documento  $d_j$  é uma combinação desses dois componentes. O *Term Frequency* (TF) quantifica a ocorrência do termo no documento. O *Inverse Document Frequency* (IDF) mede a raridade do termo na coleção.

A ideia de ponderar termos pela sua frequência inversa na coleção foi introduzida originalmente por (JONES, 1988) como uma medida de *especificidade do termo*, com a lógica de que termos que aparecem em muitos documentos são maus discriminadores e devem receber um peso menor. A formulação matemática clássica para o peso IDF de um termo  $t_i$  é dada por:

$$\text{idf}(t_i) = \log \left( \frac{N}{n_i} \right) \quad (2.4.2)$$

onde  $N$  é o número total de documentos na coleção e  $n_i$  é o número de documentos que contêm o termo  $t_i$  (ROBERTSON, 2004). O peso final TF-IDF é então o produto do TF e do IDF, resultando em uma pontuação que equilibra a importância local e a raridade global do termo (JALILIFARD et al., 2020).

A principal limitação do TF-IDF, no entanto, reside em sua natureza puramente estatística e baseada em contagem de palavras. O modelo trata cada palavra como uma unidade atômica e independente, ignorando completamente as relações semânticas e o contexto. Palavras sinônimas como "carro" e "automóvel" são tratadas como entidades completamente distintas, e o modelo é incapaz de capturar o fato de que elas se referem ao mesmo conceito.

Essa falha em compreender o significado e o contexto das palavras, um fenômeno conhecido como "problema do desencontro de vocabulário" (vocabulary mismatch), impõe um teto significativo na qualidade das representações que podem ser geradas e, conseqüentemente, no desempenho dos modelos de aprendizado de máquina que as utilizam.

#### 2.4.1.2 Word2Vec

O Word2Vec, introduzido por (MIKOLOV et al., 2013b), representou um avanço fundamental na representação vetorial de texto, superando as limitações semânticas do TF-IDF. Em vez de vetores esparsos baseados em contagens, o Word2Vec aprende embeddings densos e de baixa dimensionalidade para cada palavra, onde a posição de um vetor no espaço vetorial captura suas propriedades semânticas e sintáticas. A inovação central do Word2Vec é sua capacidade de aprender essas representações a partir de grandes corpora de texto não rotulado, baseando-se na hipótese distribucional de que apoiando-se no princípio de que palavras em contextos parecidos tendem a ter significados próximos (GOLDBERG; LEVY, 2014).

O Word2Vec propõe duas arquiteturas de modelo principais: Continuous Bag-of-Words (CBOW) e Skip-gram. Enquanto o CBOW prevê a palavra atual com base em seu contexto (as palavras vizinhas), o modelo Skip-gram faz o inverso: prevê as palavras de contexto a partir da palavra atual (MIKOLOV et al., 2013b). O modelo Skip-gram, em particular com a otimização de treinamento chamada Negative Sampling, tornou-se extremamente popular por sua eficiência computacional e pela alta qualidade dos vetores gerados. O objetivo do treinamento é ajustar os vetores de palavras de tal forma que pares de palavra-contexto  $(w, c)$  que realmente ocorrem no texto recebam uma alta pontuação de similaridade, enquanto pares aleatórios (negativos) recebam uma pontuação baixa.

Conforme derivado por (GOLDBERG; LEVY, 2014), a tarefa é colocada como um problema de classificação binária, onde se busca maximizar a probabilidade  $p(D = 1 \mid w, c)$  de que um par observado  $(w, c)$  tenha vindo dos dados do corpus. Essa probabilidade é parametrizada pelos vetores da palavra  $(\mathbf{v}_w)$  e do contexto  $(\mathbf{v}_c)$  através da função sigmoide:

$$p(D = 1 \mid w, c; \theta) = \frac{1}{1 + e^{-\mathbf{v}_c^T \mathbf{v}_w}} = \sigma(\mathbf{v}_c^T \mathbf{v}_w) \quad (2.4.3)$$

Para evitar que o modelo aprenda uma solução trivial (onde todos os vetores são idênticos), a função objetivo é estendida com um conjunto  $D'$  de exemplos negativos, gerados aleatoriamente. O objetivo final é maximizar a probabilidade dos pares positivos e, ao mesmo tempo, minimizar a probabilidade dos pares negativos, o que se traduz na seguinte função de otimização:

$$\arg \max_{\theta} \left( \sum_{(w,c) \in D} \log \sigma(\mathbf{v}_c^T \mathbf{v}_w) + \sum_{(w,c) \in D'} \log \sigma(-\mathbf{v}_c^T \mathbf{v}_w) \right) \quad (2.4.4)$$

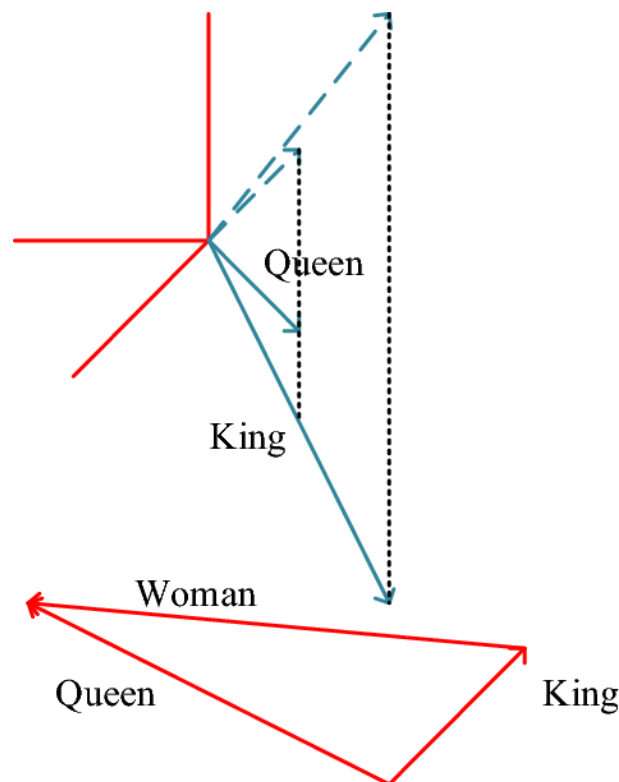
onde  $D$  é o conjunto de pares observados no corpus e  $D'$  é o conjunto de pares gerados aleatoriamente (negativos).

O resultado mais notável dessa abordagem é que os embeddings aprendidos capturam regularidades linguísticas e relações semânticas complexas que podem ser expressas através de operações algébricas simples com os vetores. O exemplo canônico, demonstrado em (MIKOLOV et al., 2013b), é a capacidade do modelo de resolver analogias semânticas. Por exemplo, a operação vetorial

$$\mathbf{v}(\text{"rei"}) - \mathbf{v}(\text{"homem"}) + \mathbf{v}(\text{"mulher"})$$

resulta em um vetor que é mais próximo, em termos de distância de cosseno, do vetor da palavra "rainha". Essa propriedade, ilustrada na Figura 2.4.1.2, mostra que o modelo aprende uma estrutura geométrica no espaço vetorial que reflete relações de gênero, pluralidade, capitais de países, entre outras. Essa capacidade de codificar significado em uma estrutura vetorial representou um salto de qualidade em relação ao TF-IDF, fornecendo representações de texto muito mais ricas e informativas.

Figura 2.4.2 – Relações Semânticas em Espaço Vetorial (Word2Vec).



Fonte: (NING; CHEN, 2023)

## 2.4.2 A Revolução dos Transformers e dos Grandes Modelos de Linguagem (LLMs)

### 2.4.2.1 A Arquitetura Transformer e o Mecanismo de Atenção

A arquitetura Transformer, introduzida por (VASWANI et al., 2017), representa um divisor de águas no Processamento de Linguagem Natural (PLN), afastando-se das arquiteturas recorrentes (RNNs) e convolucionais (CNNs) que dominavam o campo até então. A inovação fundamental do Transformer reside em sua completa dependência de um mecanismo chamado atenção (attention), especificamente a autoatenção (self-attention), para modelar as dependências entre palavras em uma sequência, independentemente de sua posição. Essa mudança arquitetônica não apenas permitiu um grau de paralelização muito maior durante o treinamento, tornando viável o uso de conjuntos de dados massivos, mas também se provou superior na captura de dependências de longo prazo, um grande desafio para as RNNs (VASWANI et al., 2017; ROGERS; KOVALEVA; RUMSHISKY, 2020).

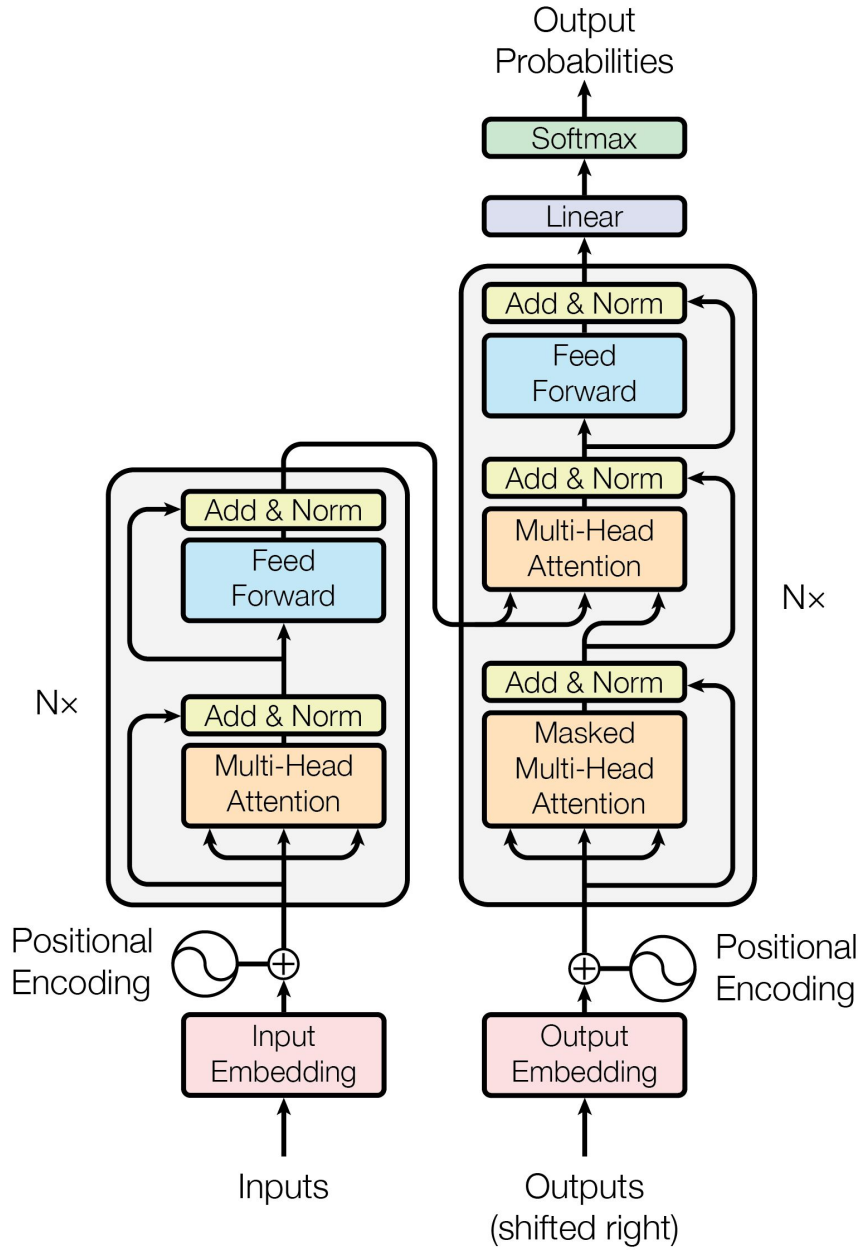
A arquitetura Transformer segue um modelo composto por duas partes principais *codificador* e *decodificador*, conforme ilustrado na Figura 2.4.3. O codificador transforma a sequência de entrada  $(x_1, \dots, x_n)$  em vetores contínuos que incorporam o contexto de cada elemento, produzindo uma representação intermediária  $z = (z_1, \dots, z_n)$ . Em seguida, o decodificador utiliza essa representação, juntamente com os tokens já gerados, para produzir a sequência de saída  $(y_1, \dots, y_m)$  de forma incremental, um elemento por vez.

Ambas as pilhas, codificadora e decodificadora, são compostas por um número  $N$  de camadas idênticas, onde cada camada possui duas subcamadas principais: um mecanismo de atenção multi-cabeças (multi-head self-attention) e uma rede neural feed-forward totalmente conectada. Conexões residuais seguidas de normalização de camada são aplicadas em torno de cada uma dessas subcamadas para facilitar o fluxo de gradientes e estabilizar o treinamento (VASWANI et al., 2017).

O elemento central responsável pela eficácia do Transformer é o seu mecanismo de atenção, em especial o *Scaled Dot-Product Attention*. De forma conceitual, esse mecanismo realiza um mapeamento entre uma consulta (*query*) e um conjunto de pares chave-valor (*key-value*), produzindo uma saída resultante da combinação ponderada dos valores. Os pesos atribuídos a cada valor dependem do grau de similaridade calculado entre a consulta e as respectivas chaves (VASWANI et al., 2017).

No contexto da autoatenção, as consultas, chaves e valores são todos derivados da mesma sequência de entrada. Isso permite que cada palavra na sequência "preste atenção" a todas as outras palavras, calculando um peso de atenção que reflete a relevância de cada uma delas para a sua própria representação. Matematicamente, para uma matriz de consultas  $Q$ , chaves  $K$  e valores  $V$ , a saída da atenção é calculada como:

Figura 2.4.3 – A arquitetura do modelo Transformer.



**Fonte:** (VASWANI et al., 2017).

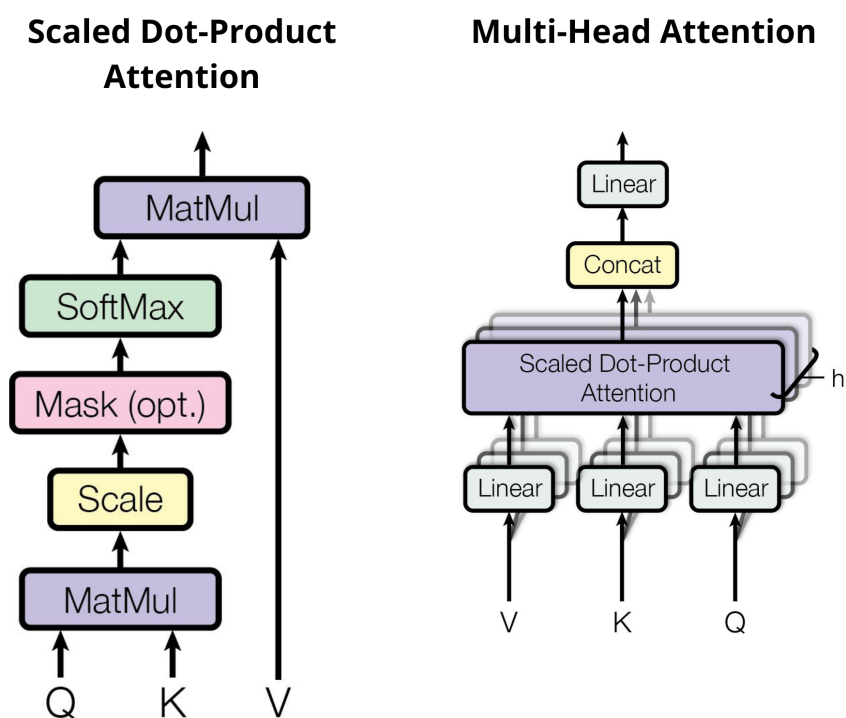
$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (2.4.5)$$

onde  $d_k$  é a dimensão dos vetores de chave e consulta. O fator de escala  $\frac{1}{\sqrt{d_k}}$  é crucial para evitar que os produtos escalares atinjam magnitudes muito grandes, o que poderia saturar a função softmax e resultar em gradientes muito pequenos, dificultando o aprendizado (VASWANI et al., 2017).

Para aprimorar ainda mais esse mecanismo, o Transformer emprega a atenção multi-

cabeça (multi-head attention), como visualizado na Figura 2.4.4. Em vez de realizar uma única função de atenção com vetores de alta dimensionalidade, o modelo projeta linearmente as consultas, chaves e valores  $h$  vezes para diferentes subespaços aprendidos. O mecanismo de atenção é então aplicado em paralelo a cada uma dessas versões projetadas, gerando  $h$  saídas. Essas saídas são então concatenadas e projetadas novamente para produzir o resultado final. Essa abordagem permite que o modelo aprenda diferentes tipos de relações e capture informações de diferentes subespaços de representação simultaneamente. Estudos de "BERTologia" revelaram que diferentes cabeças de atenção se especializam em capturar distintos fenômenos linguísticos, como relações sintáticas (ex: sujeito-verbo) e anáforas (ROGERS; KOVALEVA; RUMSHISKY, 2020). A Figura 2.4.5 ilustra como diferentes cabeças podem focar em padrões de atenção distintos, como relações diagonais (palavras vizinhas) ou padrões mais complexos que abrangem toda a sentença.

Figura 2.4.4 – Mecanismos de (esquerda) Scaled Dot-Product Attention e (direita) Multi-Head Attention.

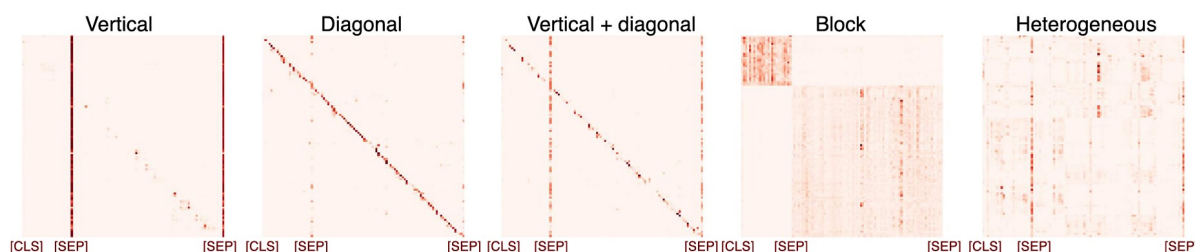


Fonte: (VASWANI et al., 2017).

Finalmente, como o modelo não contém recorrência ou convolução, ele não tem uma noção inerente da ordem das palavras. Para contornar isso, o Transformer injeta informações

sobre a posição dos tokens na sequência através de codificações posicionais (positional encodings). Esses vetores são somados aos embeddings de entrada e fornecem ao modelo um sinal sobre a posição relativa ou absoluta dos tokens. A formulação original utiliza funções senoidais e cossenoidais de diferentes frequências para gerar essas codificações, permitindo que o modelo generalize para sequências mais longas do que as vistas durante o treinamento (VASWANI et al., 2017). A combinação do mecanismo de autoatenção multi-cabeças com as codificações posicionais permitiu que a arquitetura Transformer alcançasse um novo estado da arte em diversas tarefas de PLN, tornando-se a base para os Grandes Modelos de Linguagem, incluindo o BERT, que são centrais para este trabalho (WOLF et al., 2019).

Figura 2.4.5 – Padrões de atenção visualizados em BERT. Diferentes cabeças de atenção aprendem a focar em diferentes tipos de relações entre os tokens.



**Fonte:** (ROGERS; KOVALEVA; RUMSHISKY, 2020)

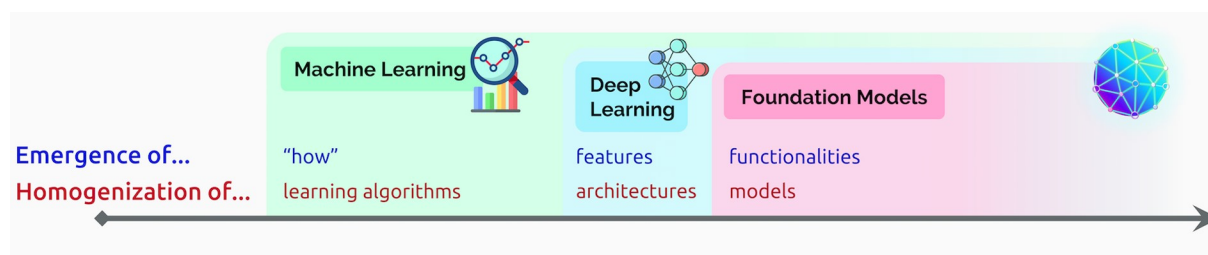
### 2.4.2.2 Grandes Modelos de Linguagem (LLMs)

Os Grandes Modelos de Linguagem (LLMs) representam a nova era da tecnologia de Processamento de Linguagem Natural e são uma categoria proeminente do que a comunidade de pesquisa passou a chamar de "modelos de fundação" (foundation models). Um modelo de fundação é definido como qualquer modelo que é treinado em dados amplos (geralmente usando auto-supervisão em grande escala) e que pode ser adaptado para uma vasta gama de tarefas posteriores (BOMMASANI et al., 2021). Essa definição encapsula a essência dos LLMs: arquiteturas massivas, predominantemente baseadas em Transformers, que foram pré-treinadas em volumes colossais de dados textuais para desenvolver uma compreensão generalizada e profunda da linguagem humana.

O poder desses modelos deriva de duas características principais: emergência e homogeneização (BOMMASANI et al., 2021). A homogeneização refere-se à consolidação de metodologias, onde um único modelo de fundação, como o BERT ou o GPT-3, serve como base para quase todos os sistemas de PLN de ponta. Já a emergência é a característica mais notável: habilidades complexas que não foram explicitamente programadas, mas que emergem implicitamente como resultado do treinamento em grande escala. O pré-treinamento auto-supervisionado, como a modelagem de linguagem auto-regressiva (prever a próxima palavra)

ou a modelagem de linguagem mascarada (preencher palavras ausentes), força o modelo a aprender padrões estatísticos, sintáticos, semânticos e até mesmo de raciocínio a partir dos dados (BOMMASANI et al., 2021). É essa escala que permite que os LLMs capturem nuances contextuais, ambiguidades e relações de longo alcance no texto, superando drasticamente as abordagens anteriores. Podemos notar os conceitos explicados acima na Figura 2.4.6

Figura 2.4.6 – Relação entre Emergência e Homogeneização nos Modelos Fundamentais



Fonte: (BOMMASANI et al., 2021)

A aplicação prática dessa profunda compreensão linguística se manifesta na capacidade dos LLMs de gerar representações vetoriais de texto (embeddings) de alta qualidade. Modelos específicos como o text-embedding-ada-002 da OpenAI, um antecessor dos mais recentes text-embedding-3-small e text-embedding-3-large, exemplificam essa tecnologia. O processo para obter um embedding com esses modelos é direto: uma string de texto é enviada para o endpoint da API de embeddings da OpenAI, juntamente com o nome do modelo desejado. A API, então, retorna uma representação vetorial (uma lista de números de ponto flutuante) que encapsula o significado semântico do texto de entrada (OPENAI, 2024).

Conforme descrito na documentação da OpenAI (2024), a principal função desses embeddings é medir a relação entre strings de texto. A distância entre dois vetores no espaço vetorial é um indicativo de sua relação semântica: distâncias pequenas sugerem alta relação, enquanto distâncias grandes sugerem baixa relação. Essa propriedade é fundamental para uma variedade de casos de uso, como busca semântica, clusterização, sistemas de recomendação e, crucialmente para este trabalho, classificação de textos. Modelos mais recentes, como os da família text-embedding-3 ou outros modelos de ponta como os da família DeepSeek, continuam a aprimorar essa capacidade, oferecendo melhor desempenho multilíngue e custos mais baixos, tornando a detecção de desinformação mais acessível e eficaz.

## 2.5 Algoritmos de Classificação Utilizados

### 2.5.1 Máquinas de Vetores de Suporte (SVM)

As Máquinas de Vetores de Suporte (SVMs) são uma classe de algoritmos de aprendizado de máquina supervisionado que se destacam por sua base teórica sólida na teoria de

aprendizagem estatística e por sua eficácia em problemas de classificação (BURGES, 1998). A lógica fundamental de um SVM para classificação binária é encontrar um hiperplano que não apenas separe as duas classes de dados, mas que o faça da melhor maneira possível.

### A Lógica do Hiperplano de Margem Máxima

Para um conjunto de dados linearmente separável, rotulado como  $\{x_i, y_i\}$ , onde  $x_i$  é o vetor de características e  $y_i \in \{-1, 1\}$  é o rótulo da classe, podem existir infinitos hiperplanos que separam as classes. A inovação central da SVM é a busca pelo hiperplano ótimo, que é definido como aquele que maximiza a distância para os pontos de dados mais próximos de cada classe. Essa distância é chamada de *margem* (CORTES; VAPNIK, 1995).

Um hiperplano é definido pela equação:

$$w \cdot x + b = 0,$$

onde  $w$  é o vetor normal (perpendicular) ao hiperplano e  $b$  é o termo de *bias*. Para garantir que todos os pontos de dados sejam classificados corretamente, as seguintes restrições devem ser satisfeitas (BURGES, 1998):

$$y_i(w \cdot x_i + b) \geq 1, \quad \text{para } i = 1, \dots, l$$

Os pontos de dados que satisfazem essa equação com igualdade, ou seja,  $y_i(w \cdot x_i + b) = 1$ , são chamados de *vetores de suporte* (*support vectors*). São esses pontos, os mais próximos do hiperplano de separação, que definem a margem. A distância da margem é  $\frac{2}{\|w\|}$ . Portanto, maximizar a margem é equivalente a minimizar  $\|w\|^2$ , o que transforma o problema em uma tarefa de otimização quadrática convexa, garantindo a existência de um mínimo global e único (BURGES, 1998).

### O Caso Não-Separável e o Truque do Kernel

Na prática, a maioria dos conjuntos de dados não é perfeitamente separável linearmente. Para lidar com isso, (CORTES; VAPNIK, 1995) introduziram o conceito de *margem suave* (*soft margin*). Essa abordagem permite que alguns pontos de dados extrapolem a margem ou sejam classificados incorretamente, introduzindo variáveis de folga (*slack variables*),  $\xi_i \geq 0$ . O objetivo da otimização torna-se um *trade-off* entre maximizar a margem e minimizar o número de erros de classificação, controlado por um hiperparâmetro de regularização  $C$ .

Para lidar com dados que não são separáveis por um hiperplano linear, as SVMs utilizam uma técnica poderosa conhecida como *truque do kernel* (*kernel trick*). A ideia é mapear os dados de entrada para um espaço de características de dimensão muito mais alta (ou até infinita), onde os dados se tornem linearmente separáveis. Seja  $\Phi(x)$  a função que mapeia o vetor de entrada  $x$  para esse novo espaço de alta dimensão. O problema é que

calcular explicitamente essa transformação pode ser computacionalmente muito custoso ou até inviável (BURGES, 1998).

O truque do kernel contorna essa dificuldade ao observar que o algoritmo de treinamento da SVM depende apenas do produto escalar dos vetores de dados,  $x_i \cdot x_j$ . Podemos, então, substituir esse produto escalar por uma função de kernel,  $K(x_i, x_j)$ , que calcula o produto escalar no espaço de características de alta dimensão sem nunca precisar realizar o mapeamento explicitamente:

$$K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

Isso permite a construção de classificadores não lineares de forma eficiente. Exemplos comuns de kernels incluem (BURGES, 1998):

$$\text{Polinomial: } K(x, y) = (x \cdot y + 1)^p$$

$$\text{Função de Base Radial (RBF) Gaussiana: } K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Uma vez treinado o modelo, a função de decisão para um novo ponto de dados  $x$  é determinada pelo sinal da seguinte equação, que depende apenas dos vetores de suporte ( $s_i$ ) e da função de kernel:

$$f(x) = \text{sgn}\left(\sum_i \alpha_i y_i K(s_i, x) + b\right)$$

Onde  $\alpha_i$  são os multiplicadores de Lagrange aprendidos durante o treinamento, que são diferentes de zero apenas para os vetores de suporte (BURGES, 1998; CORTES; VAPNIK, 1995). Isso torna as SVMs eficientes na fase de teste, pois a classificação de novos pontos depende apenas de um subconjunto dos dados de treinamento.

## 2.5.2 Florestas Aleatórias (Random Forest)

As Florestas Aleatórias, ou *Random Forests*, são um dos algoritmos de aprendizado de máquina do tipo *ensemble* mais robustos e populares, introduzido formalmente por Leo Breiman (BREIMAN, 2001). A premissa central do método é que a combinação de múltiplos modelos, mesmo que individualmente fracos, pode resultar em um modelo final com desempenho superior e maior capacidade de generalização. Uma Random Forest é, essencialmente, um conjunto de classificadores baseados em árvores de decisão, onde cada árvore é treinada de uma forma ligeiramente diferente para promover a diversidade no *ensemble* (BREIMAN, 2001).

O funcionamento do classificador se baseia em duas fontes principais de aleatoriedade:

- **Bagging (Bootstrap Aggregating):** Para cada uma das  $N$  árvores na floresta, um novo conjunto de dados de treinamento é criado através de amostragem aleatória com

reposição do conjunto de dados original. Isso significa que cada árvore é treinada em uma versão ligeiramente diferente dos dados, o que ajuda a reduzir a variância e a evitar o sobreajuste (*overfitting*) (BREIMAN, 2001).

- **Seleção Aleatória de Atributos:** Ao construir cada árvore, em cada nó, em vez de procurar o melhor divisor entre todos os atributos, o algoritmo seleciona um subconjunto aleatório de atributos e busca o melhor divisor apenas dentro desse subconjunto. Essa etapa é crucial para decorrelacionar as árvores da floresta, pois impede que atributos muito preditivos dominem a construção de todas as árvores (BREIMAN, 2001).

Após o treinamento, a classificação de uma nova instância é realizada através de um processo de “votação”. Cada árvore na floresta classifica a instância de forma independente, e a classe que recebe o maior número de votos é a predição final do modelo.

#### Fundamentação Teórica e Erro de Generalização

Uma das propriedades mais importantes das Florestas Aleatórias é que elas não sofrem de sobreajuste (*overfitting*) à medida que mais árvores são adicionadas. A Lei Forte dos Grandes Números garante que o erro de generalização converge para um valor limite (BREIMAN, 2001).

A precisão de uma Random Forest depende de dois fatores principais: a força dos classificadores individuais (árvores) e a correlação entre eles. Breiman (BREIMAN, 2001) formaliza essa relação com um limite superior para o erro de generalização. A força de um conjunto de classificadores,  $s$ , é definida como o valor esperado da função de margem, que mede a diferença entre a proporção de votos para a classe correta e a proporção de votos para qualquer outra classe. A correlação,  $\rho$ , mede a dependência entre os classificadores. A relação entre esses fatores é capturada na seguinte inequação:

$$PE^* \leq \bar{\rho} \cdot \frac{1 - s^2}{s^2}$$

Onde:

- $PE^*$  é o erro de generalização da floresta;
- $\bar{\rho}$  é a correlação média entre as árvores;
- $s$  é a força das árvores individuais.

Essa fórmula (BREIMAN, 2001) ilustra o *trade-off* fundamental do algoritmo: o objetivo é treinar árvores que sejam o mais precisas possível (maximizando  $s$ ), ao mesmo tempo em que sejam o mais independentes e diversas possíveis (minimizando  $\bar{\rho}$ ). A aleatoriedade injetada no processo de construção visa exatamente equilibrar esses dois fatores.

### Variações: Fuzzy Forests e FastForest

A arquitetura flexível da Random Forest permitiu o desenvolvimento de diversas variações. As Florestas Aleatórias Fuzzy (*Fuzzy Random Forests*), por exemplo, substituem as árvores de decisão tradicionais por árvores de decisão fuzzy. Em modelos como o IFRF (*Intuitionistic Fuzzy Random Forest*), a teoria dos conjuntos fuzzy intuicionistas é utilizada para lidar com incertezas e hesitações nos dados. Em vez de divisões “rígidas” nos nós, as árvores fuzzy permitem que uma instância pertença a múltiplos ramos com diferentes graus de pertinência, utilizando um “ganho de informação fuzzy intuicionista” para selecionar os atributos de divisão. Isso torna o modelo mais robusto e mais próximo do raciocínio humano (REN et al., 2022).

Outra linha de otimização foca na velocidade de processamento sem sacrificar a acurácia, como é o caso do *FastForest*. Este algoritmo otimiza o Random Forest tradicional através da combinação de três técnicas: *Subbagging* (amostragem sem reposição, que é computacionalmente mais leve que o bagging tradicional), *Logarithmic Split-Point Sampling* (reduz o número de pontos de divisão a serem testados para atributos numéricos) e *Dynamic Restricted Subspacing* (ajusta dinamicamente o número de atributos aleatórios considerados em cada nó). Juntas, essas otimizações resultam em um aumento significativo na velocidade de treinamento, mantendo uma acurácia comparável ou até superior à da Random Forest padrão (YATES; ISLAM, 2021).

### 2.5.3 Regressão Logística

A Regressão Logística, apesar de seu nome sugerir uma técnica de regressão, é um dos algoritmos mais fundamentais e amplamente utilizados para tarefas de classificação binária (HOSMER; LEMESHOW; STURDIVANT, 2013; COX, 1958). Trata-se de um modelo linear generalizado que estima a probabilidade de uma instância pertencer a uma determinada classe, mapeando uma combinação linear das características de entrada para um valor de probabilidade no intervalo  $[0, 1]$  através da função logística (sigmoide).

Para um problema de classificação binária, onde  $y \in \{0, 1\}$  representa o rótulo da classe (por exemplo, notícia verdadeira ou falsa), a Regressão Logística modela a probabilidade condicional  $P(y = 1 \mid \mathbf{x})$  como:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}$$

onde  $\mathbf{x}$  é o vetor de características (os *embeddings* no contexto deste trabalho),  $\mathbf{w}$  é o vetor de pesos aprendido durante o treinamento,  $b$  é o termo de viés (*bias*), e  $\sigma(\cdot)$  é a função sigmoide (HOSMER; LEMESHOW; STURDIVANT, 2013). A função sigmoide tem a propriedade crucial de comprimir qualquer valor de entrada real para o intervalo  $(0, 1)$ , permitindo sua interpretação como probabilidade.

O limite de decisão padrão é  $P(y = 1 \mid \mathbf{x}) = 0.5$ : se a probabilidade predita for maior ou igual a 0.5, a instância é classificada como pertencente à classe positiva ( $y = 1$ ); caso contrário, à classe negativa ( $y = 0$ ).

Os parâmetros  $\mathbf{w}$  e  $b$  são aprendidos minimizando uma função de perda, tipicamente a **entropia cruzada binária** (*binary cross-entropy loss*), que quantifica a discrepância entre as probabilidades preditas e os rótulos verdadeiros. Para um conjunto de treinamento com  $N$  amostras, a função de perda é definida como:

$$L(\mathbf{w}, b) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

onde  $y_i$  é o rótulo verdadeiro e  $\hat{y}_i = \sigma(\mathbf{w}^T \mathbf{x}_i + b)$  é a probabilidade predita para a amostra  $i$ . A otimização é geralmente realizada por métodos de gradiente descendente, como o *Limited-memory BFGS* (L-BFGS) ou *Stochastic Gradient Descent* (SGD).

Para evitar *overfitting*, especialmente em espaços de alta dimensionalidade como os gerados por *embeddings*, a Regressão Logística frequentemente incorpora termos de regularização. As duas formas mais comuns são:

- **Regularização L2 (Ridge)**: Adiciona uma penalidade proporcional ao quadrado da magnitude dos pesos,  $\lambda \|\mathbf{w}\|_2^2$ , encorajando pesos menores e mais distribuídos.
- **Regularização L1 (Lasso)**: Adiciona uma penalidade proporcional ao valor absoluto dos pesos,  $\lambda \|\mathbf{w}\|_1$ , que pode levar alguns pesos a exatamente zero, realizando assim uma seleção automática de características.

O hiperparâmetro  $C = 1/\lambda$  controla a força da regularização: valores menores de  $C$  implicam em maior regularização (HOSMER; LEMESHOW; STURDIVANT, 2013).

## 2.6 Métricas de Avaliação de Classificadores

A avaliação do desempenho de um modelo de classificação é um passo indispensável para determinar sua eficácia e confiabilidade. Embora uma vasta gama de métricas exista, a prática de avaliação em trabalhos de pesquisa é frequentemente nebulosa, com métricas selecionadas sem uma argumentação clara (OPITZ, 2024). Para garantir uma análise transparente e robusta, especialmente em uma tarefa crítica como a detecção de fake news, é fundamental definir formalmente as métricas utilizadas. A base para a maioria dessas métricas é a matriz de confusão, que compara os valores preditos pelo modelo com os valores reais do conjunto de teste (RAINIO; TEUHO; KLÉN, 2024).

A partir da matriz de confusão, derivamos quatro contagens fundamentais para o problema de classificação binária (notícia falsa vs. verdadeira):

- **Verdadeiro Positivo (TP - True Positive):** Notícias falsas que foram corretamente classificadas como falsas.
- **Verdadeiro Negativo (TN - True Negative):** Notícias verdadeiras que foram corretamente classificadas como verdadeiras.
- **Falso Positivo (FP - False Positive):** Notícias verdadeiras que foram incorretamente classificadas como falsas (alarme falso).
- **Falso Negativo (FN - False Negative):** Notícias falsas que foram incorretamente classificadas como verdadeiras (erros de omissão).

Com base nesses valores, definem-se as seguintes métricas de avaliação:

### 2.6.1 Acurácia

A acurácia é a métrica mais intuitiva e amplamente utilizada para avaliar o desempenho geral de um classificador. Ela mede a proporção de todas as previsões que o modelo acertou, independentemente da classe. Formalmente, é definida como (RAINIO; TEUHO; KLÉN, 2024):

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

Embora simples, a acurácia pode ser uma métrica enganosa, especialmente em contextos com distribuição de classes desbalanceada, como é comum em datasets de fake news, onde o número de notícias verdadeiras pode superar vastamente o de notícias falsas. Nesse cenário, um modelo trivial que classifica todas as notícias como “verdadeiras” pode alcançar uma alta acurácia, mas seria completamente inútil para o propósito de detecção. O uso da acurácia sem considerar a distribuição das classes pode criar uma impressão errônea de um modelo de alto desempenho, pois ela ignora a contribuição da classe minoritária.

### 2.6.2 Precisão

A precisão responde à seguinte questão: de todas as instâncias que o modelo classificou como positivas (falsas), quantas eram de fato positivas? É uma medida da exatidão das previsões positivas. No contexto de fake news, a precisão mede a confiabilidade do modelo ao rotular uma notícia como falsa. Uma alta precisão indica que o modelo classifica poucos falsos positivos. Sua fórmula é (RAINIO; TEUHO; KLÉN, 2024):

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

### 2.6.3 Revocação (Recall ou Sensibilidade)

A revocação, também conhecida como sensibilidade (*sensitivity*) ou taxa de verdadeiros positivos (*true positive rate*), mede a capacidade do modelo de identificar todas as instâncias positivas relevantes no conjunto de dados. Ela responde à pergunta: de todas as notícias que eram realmente falsas, quantas o modelo conseguiu identificar corretamente. Uma alta revocação indica que o modelo tem uma baixa taxa de falsos negativos. A fórmula é (RAINIO; TEUHO; KLÉN, 2024):

$$\text{Revocação} = \frac{TP}{TP + FN}$$

### 2.6.4 F1-Score

Em muitas situações, existe um *trade-off* entre precisão e revocação: aumentar uma muitas vezes leva à diminuição da outra. O F1-Score surge como uma métrica de balanço que combina ambas, calculando a média harmônica entre elas. A média harmônica penaliza mais fortemente os valores extremos, o que a torna uma métrica mais robusta do que a média aritmética simples, especialmente em casos de desbalanceamento de classes. O F1-Score é particularmente útil quando tanto os falsos positivos quanto os falsos negativos são importantes e se deseja uma única métrica que represente um equilíbrio entre eles. A fórmula é (RAINIO; TEUHO; KLÉN, 2024):

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

## 2.7 Trabalhos Correlatos

A detecção automática de fake news evoluiu rapidamente, acompanhando os avanços nas áreas de Processamento de Linguagem Natural (PLN) e Aprendizado de Máquina (AM). Os esforços de pesquisa podem ser categorizados em uma progressão histórica e de complexidade, iniciando-se com a criação de recursos linguísticos fundamentais e a aplicação de modelos estatísticos clássicos, e culminando nas mais recentes investigações com Grandes Modelos de Linguagem (LLMs). Esta seção apresenta uma revisão crítica desses trabalhos.

### 2.7.1 Abordagens Iniciais e Datasets em Português

O primeiro grande obstáculo para a aplicação de técnicas computacionais na detecção de fake news em português foi a carência de dados estruturados e publicamente disponíveis. A criação de corpora (conjuntos de dados textuais) de alta qualidade foi, portanto, um passo fundamental que viabilizou a pesquisa na área.

Um dos trabalhos pioneiros e mais influentes nesse sentido foi a criação do corpus Fake.Br (MONTEIRO et al., 2018a). Este recurso consiste em 7.200 notícias, balanceadas com 3.600 notícias falsas e 3.600 verdadeiras. Uma característica distintiva do Fake.Br é sua metodologia de pareamento: para cada notícia falsa, os pesquisadores buscaram uma notícia verdadeira correspondente sobre o mesmo assunto, utilizando medidas de similaridade lexical. Essa abordagem, embora trabalhosa, garantiu que os modelos treinados com esses dados aprendessem a diferenciar os textos com base em suas características linguísticas e estruturais, e não apenas pelo tópico abordado. O corpus foi crucial para o desenvolvimento e a avaliação dos primeiros classificadores automáticos para o português.

Posteriormente, outros pesquisadores buscaram expandir a disponibilidade de dados. (GARCIA; AFONSO; PAPA, 2022) introduziram o FakeRecogna, um corpus maior e mais recente, com 11.902 amostras de notícias brasileiras coletadas entre 2019 e 2021. Diferentemente do Fake.Br, o FakeRecogna não estabelece um vínculo direto entre as notícias falsas e verdadeiras, o que pode reduzir um possível viés de classificação e refletir um cenário mais realista. Outro recurso relevante é o Fakepedia Corpus, desenvolvido por (CHARLES; RUBACK; OLIVEIRA, 2022), que se destaca por empregar um processo automático para sua criação e manutenção, garantindo que o corpus permaneça constantemente atualizado um fator crucial, dado o caráter evolutivo das fake news. A existência desses corpora foi o que permitiu a aplicação e o teste sistemático de diversas abordagens de aprendizado de máquina.

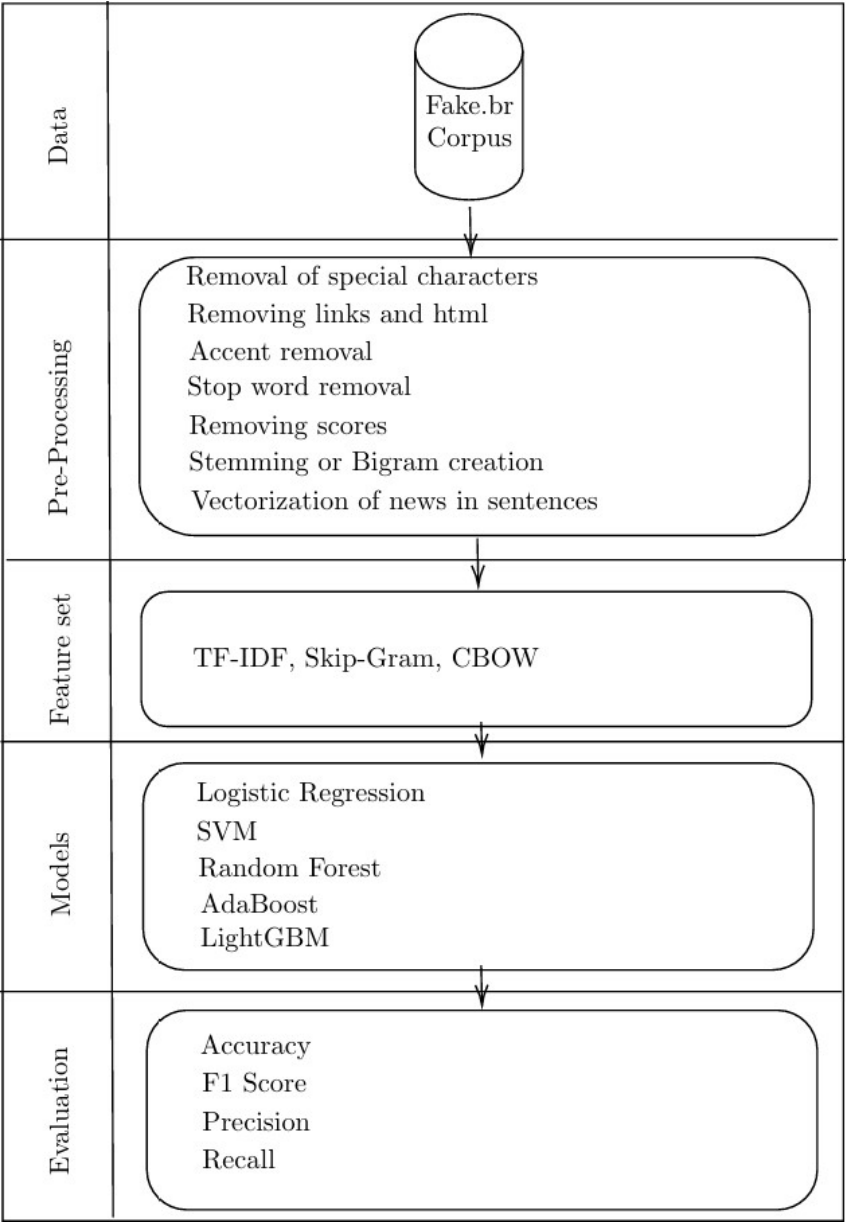
### 2.7.2 Detecção com Aprendizado de Máquina Clássico

Com a disponibilidade de datasets, os primeiros esforços computacionais focaram na aplicação de algoritmos de aprendizado de máquina clássicos. Essas abordagens geralmente combinam uma técnica de representação vetorial de texto, como TF-IDF (Term Frequency-Inverse Document Frequency) ou Word2Vec, com classificadores como Máquinas de Vetores de Suporte (SVM), Florestas Aleatórias (Random Forest) e Regressão Logística. O processo geral para essas abordagens, que parte de um corpus, passa por etapas de pré-processamento, extração de características, treinamento de modelos e avaliação, é ilustrado na Figura 2.7.1

No próprio estudo de apresentação do Fake.Br, (MONTEIRO et al., 2018a) testaram um classificador SVM e, surpreendentemente, descobriram que a representação mais simples, baseada em bag-of-words (um modelo que considera apenas a ocorrência de palavras), foi capaz de atingir uma F-measure de 88%, demonstrando a eficácia de modelos tradicionais quando aplicados a um dataset bem construído.

Em um trabalho mais recente, (GIORDANI et al., 2023) realizaram um estudo abrangente comparando diversos classificadores clássicos sobre um corpus de notícias jornalísticas em português. Utilizando técnicas como TF-IDF e Word2Vec, eles avaliaram modelos como Regressão Logística, SVM e Random Forest. Seus resultados mostraram que o algoritmo LightGBM, combinado com a representação TF-IDF, alcançou a maior acurácia, com 96,17%.

Figura 2.7.1 – Fluxograma ilustrando a estrutura geral dos modelos de detecção de fake news baseados em aprendizado de máquina clássico



Fonte: Adaptado de (GIORDANI et al., 2023).

Esses trabalhos demonstram que modelos clássicos podem atingir alto desempenho, mas sua principal limitação reside na dependência de padrões estatísticos de superfície, o que dificulta a captura de nuances semânticas, ironia, sarcasmo ou manipulações contextuais complexas, características frequentes em notícias falsas sofisticadas.

### 2.7.3 O Uso de Deep Learning e Modelos Pré-Transformers

Para superar as limitações dos modelos clássicos, a pesquisa voltou-se para as arquiteturas de deep learning, que são capazes de aprender representações de características de

forma automática e hierárquica. Modelos como Redes Neurais Recorrentes (RNNs) e Long Short-Term Memory (LSTMs) foram explorados por sua capacidade de processar sequências de texto, capturando dependências de longo prazo. Por exemplo, (CHARLES; RUBACK; OLIVEIRA, 2022), ao validar seu corpus Fakepedia, utilizaram um modelo LSTM e alcançaram uma F1-score de 94%, evidenciando a superioridade dos modelos sequenciais para essa tarefa.

O verdadeiro divisor de águas, no entanto, foi a introdução da arquitetura Transformer e, consequentemente, de modelos pré-treinados como o BERT (Bidirectional Encoder Representations from Transformers). Esses modelos trouxeram uma compreensão contextual profunda da linguagem, analisando cada palavra em relação a todas as outras na sentença (e não apenas de forma sequencial), o que representou um salto qualitativo.

Um exemplo notável é o trabalho de (KALIYAR; GOSWAMI; NARANG, 2021b), que propuseram o FakeBERT, um modelo que combina a arquitetura BERT com Redes Neurais Convolucionais (CNNs) para a detecção de fake news em mídias sociais. O modelo demonstrou um desempenho de ponta, alcançando uma acurácia de 98,90% em um dataset de referência em inglês. Esse resultado expressivo ilustra como a capacidade do BERT de gerar embeddings contextualmente ricos permite que os classificadores identifiquem padrões de desinformação com uma precisão muito superior à das técnicas baseadas em frequência de palavras.

#### 2.7.4 LLMs para Análise de Texto e Desinformação

O sucesso de modelos como o BERT abriu caminho para a geração de Grandes Modelos de Linguagem (LLMs) ainda maiores e mais capazes, como a família GPT. A aplicação desses LLMs de última geração para a detecção de fake news é uma fronteira de pesquisa emergente. Embora trabalhos que realizem uma comparação sistemática desses modelos especificamente para o português ainda sejam escassos, pesquisas em outras línguas já apontam para o seu enorme potencial.

Um estudo recente e de grande relevância de (ROUMELIOTIS; TSELIKAS; NASIOPOULOS, 2025) conduziu uma análise comparativa entre CNNs, BERT e modelos da família GPT (como o GPT-4 Omni) na tarefa de classificação de fake news. Os resultados foram contundentes: enquanto um modelo CNN alcançou 58,6% de acurácia, o modelo GPT-4 Omni, após um processo de fine-tuning, atingiu 98,6% de acurácia. Este desempenho não apenas superou significativamente os modelos tradicionais, mas também excedeu ligeiramente o já excelente resultado do BERT (97,5% no estudo de (KALIYAR; GOSWAMI; NARANG, 2021b)), demonstrando que os LLMs mais avançados possuem uma capacidade ainda mais refinada de compreender o contexto, a sutileza e as intenções por trás do texto.

Essas descobertas sugerem que a superioridade dos embeddings gerados por LLMs de ponta deve se traduzir em um desempenho superior também para a detecção de fake news em português. No entanto, uma investigação sistemática que compare diferentes LLMs e técnicas

de embedding tradicionais, utilizando múltiplos classificadores e corpora em português, ainda é uma lacuna na literatura. É precisamente essa lacuna que o presente trabalho se propõe a preencher, oferecendo uma análise quantitativa e comparativa para determinar o estado da arte na detecção de desinformação para a língua portuguesa.

## 2.8 Síntese da Literatura e Justificativa da Pesquisa

A revisão da literatura apresentada neste capítulo demonstra uma clara trajetória evolutiva na detecção de fake news. O percurso inicia-se com a tarefa fundamental de criar corpora anotados em português, como o Fake.Br (MONTEIRO et al., 2018a) e o FakeRecognia (GARCIA; AFONSO; PAPA, 2022), que foram essenciais para viabilizar qualquer pesquisa empírica na área. Em seguida, a aplicação de algoritmos de aprendizado de máquina clássicos, como SVM e LightGBM, combinados com representações vetoriais tradicionais (TF-IDF, Word2Vec), estabeleceu os primeiros benchmarks de desempenho, alcançando altas taxas de acurácia em datasets específicos, como demonstrado por (GIORDANI et al., 2023).

O advento do deep learning e, principalmente, dos modelos baseados na arquitetura Transformer, como o BERT, representou um salto qualitativo. Trabalhos como o de (KALIYAR; GOSWAMI; NARANG, 2021b) evidenciaram que a compreensão contextual profunda desses modelos resultava em um desempenho significativamente superior, estabelecendo um novo estado da arte. Mais recentemente, a pesquisa começou a explorar o potencial dos Grandes Modelos de Linguagem (LLMs) de última geração, como a família GPT, que, conforme dito por (ROUMELIOTIS; TSELIKAS; NASIOPOULOS, 2025) para o inglês, mostra que LLMs podem alcançar resultados próximos à perfeição em tarefas de classificação textual após o fine-tuning, ultrapassando métodos anteriores.

A análise da literatura e a síntese apresentada na Tabela 2 revelam um claro padrão: enquanto os modelos mais avançados, como BERT e especialmente os LLMs da família GPT, demonstram uma superioridade esmagadora na tarefa de detecção de fake news para o idioma inglês, os estudos em português, embora fundamentais, ainda se concentram majoritariamente em modelos clássicos ou em arquiteturas de deep learning pré-Transformers. Apesar dos avanços, observa-se uma lacuna na literatura quanto à comparação sistemática do desempenho de embeddings gerados por Grandes Modelos de Linguagem de última geração (pós-BERT) com abordagens tradicionais (TF-IDF, Word2Vec) e baseadas em Transformers (BERT), especificamente para o idioma português no domínio da detecção de fake news.

Falta um estudo abrangente que utilize múltiplos classificadores e corpora em português para avaliar, sob as mesmas condições experimentais, se a riqueza semântica capturada pelos embeddings de LLMs modernos se traduz em um ganho de desempenho significativo em relação às técnicas já estabelecidas para o nosso idioma.

A lacuna identificada justifica diretamente a necessidade e a relevância deste trabalho.

Tabela 2 – Tabela comparativa dos principais trabalhos correlatos em detecção de fake news.

Autor(es)/Ano	Objetivo	Metodologia/Modelo	Dataset	Principais Resultados	Limitações Identificadas
(MONTEIRO et al., 2018a)	Apresentar um novo corpus para o português (Fake.Br) e testar métodos de detecção.	SVM com Bag-of-Words e características linguísticas.	Fake.Br (Português)	F-measure de 88% com SVM.	Foco em modelos clássicos; representação de texto baseada em frequência de palavras.
(GIORDANI et al., 2023)	Comparar classificadores clássicos e desenvolver uma plataforma de detecção.	Vários modelos clássicos (SVM, RF, LightGBM) com TF-IDF e Word2Vec.	Fake.Br (Português)	Acurácia de 96,17% com LightGBM + TF-IDF.	Não explora o potencial de modelos de deep learning ou Transformers.
(KALIYAR; GOSWAMI; NARANG, 2021b)	Propor uma abordagem de deep learning (FakeBERT) para detecção em mídias sociais.	Híbrido de BERT com Redes Neurais Convolucionais (CNNs).	Dataset Kaggle (Inglês)	Acurácia de 98,90%.	Focado no idioma inglês; o desempenho superior se deve ao poder contextual do BERT.
(CHARLES; RUBACK; OLIVEIRA, 2022)	Apresentar um corpus flexível e automaticamente atualizável para o português (Fakepedia).	LSTM (Long Short-Term Memory).	Fakepedia (Português)	F1-score de 94% para fake news e 95% para notícias reais.	Utiliza um modelo (LSTM) já superado por arquiteturas Transformer.
(ROUMELIOTIS; TSELIKAS; NASIOPOULOS, 2025)	Comparar o desempenho de CNNs, BERT e LLMs (GPT-4) para detecção.	CNN, BERT, GPT-4o, GPT-4o-mini (com e sem fine-tuning).	WELFake (Inglês)	Acurácia de 98,6% com GPT-4o fine-tuned.	Focado no inglês; alto custo computacional para fine-tuning; desempenho zero-shot subótimo.

Fonte: Elaborado pelo autor.

A premissa central é que a superioridade dos LLMs, já comprovada em inglês, também se manifestará no português, mas essa hipótese carece de validação empírica rigorosa. Portanto, esta monografia se propõe a preencher essa lacuna ao conduzir um estudo quantitativo e comparativo da eficácia de diferentes técnicas de representação vetorial desde as tradicionais, como TF-IDF e Word2Vec, até as mais avançadas, baseadas em BERT e outros LLMs de ponta na tarefa de detecção de fake news em português. Ao aplicar essas diversas representações a um conjunto consistente de classificadores (SVM, Random Forest e Redes Neurais), este trabalho irá:

1. **Estabelecer um novo benchmark** para a detecção de *fake news* em português, utilizando tecnologias de ponta.
2. **Quantificar o ganho de desempenho** proporcionado pelos *embeddings* de LLMs modernos em relação às abordagens anteriores.
3. **Fornecer insights valiosos** sobre quais combinações de representação vetorial e classificador são mais eficazes para esta tarefa específica.

Dessa forma, esta pesquisa ajudará a encontrar o estado da arte para a língua portuguesa, mas também contribuirá com evidências concretas para o desenvolvimento de ferramentas de verificação de fatos mais precisas, robustas e eficazes, um objetivo de suma importância no combate à desinformação.

## 3 Metodologia

Este capítulo detalha a metodologia quantitativa e comparativa empregada para avaliar a eficácia de diferentes representações vetoriais, com um foco particular em Grandes Modelos de Linguagem (LLMs), para a tarefa de detecção de fake news em português. A abordagem aqui descrita abrange o ciclo completo do experimento, iniciando com a seleção e justificativa do conjunto de dados, passando pelas etapas de pré-processamento do texto, detalhando os métodos de geração de embeddings tradicionais e modernos, e culminando no desenho experimental, que inclui a definição dos modelos de classificação e das métricas utilizadas para a avaliação de desempenho comparativo. O objetivo é estabelecer um protocolo experimental robusto e reproduzível para determinar o estado da arte na detecção de desinformação para a língua portuguesa. A Imagem 3.0.1 ilustra o pipeline simplificado que será apresentado no texto a seguir.

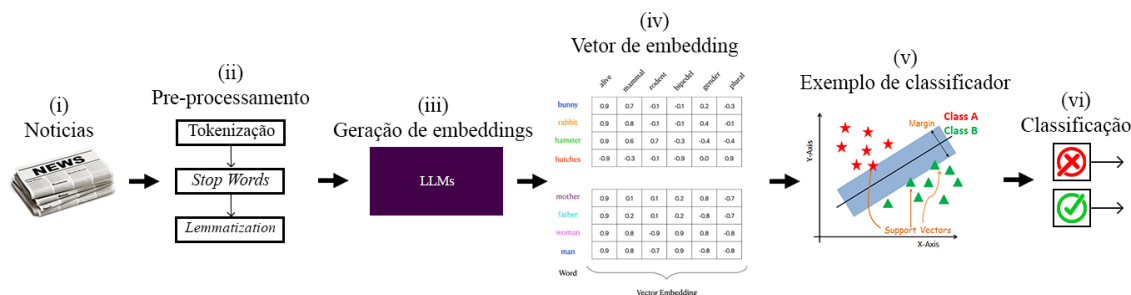
### 3.1 Conjunto de dados (Corpus)

O corpus selecionado para este estudo foi o FakeRecogna (GARCIA; AFONSO; PAPA, 2022). Trata-se de um conjunto de dados público e focado no português brasileiro, composto por 11.902 notícias em português, igualmente divididas entre verdadeiras e falsas, coletadas entre 2019 e 2021. Uma característica fundamental deste corpus é o seu balanceamento, sendo composto por 5.951 notícias falsas e 5.951 notícias verdadeiras.

As amostras foram extraídas de fontes de grande circulação e relevância no Brasil. As notícias verdadeiras foram coletadas de portais como G1, UOL e Extra, enquanto as notícias falsas foram obtidas de agências de checagem de fatos reconhecidas, como Boatos.org e Fato ou Fake. Cada entrada no dataset é estruturada e contém metadados relevantes, incluindo as colunas Título, Subtítulo, Notícia e Classe (onde 0 representa notícia falsa e 1 representa notícia verdadeira), que são as utilizadas nos experimentos conduzidos neste trabalho.

A escolha do corpus FakeRecogna, em detrimento de outras opções como o pioneiro Fake.Br (MONTEIRO et al., 2018b), foi uma decisão metodológica deliberada, motivada por três fatores principais:

- **Atualidade do Conteúdo:** O *FakeRecogna* contém textos mais recentes (2019–2021), cobrindo tópicos contemporâneos, como a pandemia de COVID-19. Isso é crucial, pois o estilo, os temas e as táticas de desinformação evoluem rapidamente. Utilizar um corpus mais atual garante que os modelos sejam treinados em um cenário mais representativo do ecossistema de informação atual.

Figura 3.0.1 – Pipeline metodológico para detecção de *fake news* utilizando *embeddings*

O pipeline ilustra as seis etapas principais do processo experimental: (i) coleta do corpus de notícias do FakeRecogna; (ii) pré-processamento textual, incluindo tokenização, remoção/manutenção de *stopwords* e lematização; (iii) geração de representações vetoriais (*embeddings*) utilizando técnicas tradicionais (TF-IDF, Word2Vec) e LLMs modernos (BERTimbau, SFR-Mistral, OpenAI, entre outros); (iv) representação matricial dos *embeddings*, onde cada palavra é mapeada para um vetor numérico de alta dimensionalidade; (v) treinamento de classificadores de aprendizado de máquina (SVM, Random Forest, Regressão Logística) para a tarefa de classificação binária (verdadeira vs. falsa); e (vi) avaliação final do desempenho utilizando métricas como acurácia, precisão, revocação e F1-score.

**Fonte:** Elaborado pelo autor.

- **Estrutura e Qualidade dos Dados:** O corpus é disponibilizado em um formato estruturado (.xlsx), o que facilita sua manipulação e o pipeline de pré-processamento, conforme demonstrado no script `processar_fakerecogna.py` desenvolvido para este trabalho. A clareza das colunas e a qualidade dos textos permitem a aplicação consistente das técnicas de PLN.
- **Ausência de Pareamento Direto:** Diferentemente do *Fake.Br*, que busca parear cada notícia falsa com uma verdadeira sobre o mesmo tema, o *FakeRecogna* não impõe essa restrição. Essa característica torna o desafio de classificação mais realista e, potencialmente, menos suscetível a vieses. O modelo é forçado a aprender a classificar com base nas características intrínsecas do texto (estilo, estrutura, vocabulário), em vez de simplesmente comparar dois textos sobre o mesmo tópico, o que se alinha melhor com um cenário de aplicação real onde não há um “par” verdadeiro para cada notícia falsa encontrada.

Dadas essas características, o FakeRecogna se apresenta como um recurso robusto e adequado para o treinamento e a avaliação comparativa dos modelos de detecção de fake news propostos neste trabalho de conclusão de curso. Esta seção corresponde à etapa (i) do pipeline metodológico ilustrado na Figura 3.0.1.

## 3.2 Pre-processamento dos dados

A etapa de pré-processamento é fundamental para transformar os textos brutos do corpus em um formato estruturado e otimizado para os algoritmos de aprendizado de máquina. Conforme apontado por (SIINO; TINNIRELLO; La Cascia, 2024), com o advento de modelos complexos como os Transformers, a necessidade e o impacto do pré-processamento tornaram-se um tópico de debate. Enquanto modelos tradicionais se beneficiam de uma limpeza textual agressiva para reduzir a dimensionalidade e o ruído, arquiteturas modernas podem extrair sinais valiosos da estrutura sintática e das palavras de função (stopwords).

Diante deste cenário, e para conduzir uma análise comparativa robusta, esta pesquisa adotou uma metodologia de pré-processamento dual. Foram geradas duas versões distintas do corpus FakeRecognia, permitindo avaliar como diferentes níveis de pré-processamento afetam o desempenho tanto dos modelos clássicos quanto dos Grandes Modelos de Linguagem (LLMs). O pipeline foi implementado em Python, utilizando as bibliotecas `spaCy` para tokenização e lematização, e `NLTK` para a lista de stopwords em português, conforme detalhado no script `processar_fakerecogna.py`.

### 3.2.1 Abordagem 1: Limpeza Extensiva com Remoção de Stopwords

1. **Conversão para Minúsculas:** Todos os caracteres foram convertidos para a forma minúscula para garantir a uniformidade e evitar que a mesma palavra seja tratada como dois tokens distintos devido à capitalização (ex: "Governo" e "governo").
2. **Tokenização e Lematização:** O texto foi segmentado em tokens individuais e cada token foi reduzido à sua forma canônica (lema) utilizando o modelo `pt_core_news_sm` da biblioteca `spaCy`. A lematização é preferível ao stemming por produzir palavras morfologicamente válidas, o que preserva melhor o significado (SANTOS; SILVA, 2023).
3. **Remoção de Stopwords:** Palavras de alta frequência e baixo valor semântico (artigos, preposições, conjunções) foram removidas com base na lista padrão para o português da biblioteca `NLTK`. Esta etapa visa reduzir o ruído e a dimensionalidade do vetor de características.
4. **Remoção de Pontuação:** Todos os sinais de pontuação foram removidos do texto.

### 3.2.2 Abordagem 2: Limpeza Moderada com Manutenção de Stopwords

A segunda abordagem foi desenhada para preservar a maior parte da estrutura sintática original das sentenças. A hipótese é que a manutenção das stopwords e de uma estrutura mais próxima da original pode ser benéfica para os LLMs, que são projetados para extrair significado de sequências contextuais completas (SIINO; TINNIRELLO; La Cascia, 2024). As etapas foram as seguintes:

1. **Conversão para Minúsculas:** O texto foi padronizado para letras minúsculas.
2. **Remoção de Caracteres Específicos:** Em vez de uma remoção completa, apenas um conjunto limitado de caracteres de pontuação e símbolos (., /, !, |, ,) foi removido para eliminar os ruídos mais comuns sem desestruturar completamente as frases.
3. **Tokenização e Lematização:** O processo de lematização com *spaCy* foi aplicado de forma idêntica à primeira abordagem.
4. **Manutenção de Stopwords:** As *stopwords* não foram removidas, preservando a estrutura gramatical original.

Este segundo dataset processado, salvo como `dataset_sem_caracteres_especiais_lemmatizado.xlsx`, serve como entrada para os modelos baseados em Transformers, permitindo uma avaliação justa sobre como essas arquiteturas lidam com textos menos processados e mais próximos da linguagem natural. A criação dessas duas versões é, portanto, um pilar central desta metodologia, viabilizando uma análise comparativa direta sobre o impacto do pré-processamento no desempenho de diferentes classes de modelos de detecção. Esta seção detalha a etapa (ii) do pipeline apresentado na Figura 3.0.1.

### 3.3 Geração das Representações Vetoriais (Embeddings)

A conversão do texto pré-processado em representações vetoriais numéricas, ou embeddings, é o cerne desta metodologia comparativa. A qualidade desses vetores é crucial, pois eles constituem a única fonte de informação que os modelos de aprendizado de máquina utilizarão para aprender a tarefa de classificação (COLLOBERT et al., 2011). Nesta seção, foi detalhado as diferentes abordagens empregadas para gerar esses vetores, dividindo-as em duas categorias principais: as abordagens de baseline, que representam técnicas tradicionais e bem estabelecidas, e as abordagens baseadas em Grandes Modelos de Linguagem (LLMs), que constituem o estado da arte e o foco principal desta investigação.

#### 3.3.1 Abordagens tradicionais

Para estabelecer uma linha de base robusta e contextualizar o desempenho dos modelos mais modernos, foram implementadas duas das mais influentes técnicas de representação textual.

O TF-IDF é uma técnica estatística que mede a importância das palavras em um documento dentro de um corpus (JONES, 1988). Sua lógica é atribuir um peso maior a termos que são frequentes em um documento específico, mas raros no conjunto geral de documentos, tornando-os bons discriminadores de tópico (JALILIFARD et al., 2020).

Neste trabalho, o TF-IDF foi implementado utilizando a classe `TfidfVectorizer` da biblioteca `scikit-learn`. Para a extração das características, o vocabulário foi limitado às 5.000 palavras mais frequentes (`max_features=5000`) e foram considerados tanto unigramas quanto bigramas (`ngram_range=(1, 2)`), o que permite capturar algumas relações contextuais locais simples. A representação final de cada documento é um vetor esparsa de 5.000 dimensões.

O Word2Vec representou um avanço fundamental em relação ao TF-IDF ao introduzir embeddings densos, que são vetores de baixa dimensionalidade capazes de capturar relações semânticas e sintáticas complexas (MIKOLOV et al., 2013b). Em vez de se basear apenas em contagens de frequência, o Word2Vec aprende essas representações a partir de grandes volumes de texto, posicionando palavras com significados semelhantes próximas no espaço vetorial, com base na hipótese distribucional (GOLDBERG; LEVY, 2014). Diferentemente do uso de modelos pré-treinados genéricos, esta pesquisa optou por treinar um modelo Word2Vec customizado diretamente sobre o corpus FakeRecogna. Essa decisão foi tomada a fim de capturar as nuances semânticas e o vocabulário específico do domínio de fake news em português presente no conjunto de dados. A implementação foi realizada com a biblioteca Gensim, onde o modelo foi configurado para utilizar a arquitetura Skip-gram (`sg=1`), considerada eficaz para capturar significado a partir de datasets menores. Conforme detalhado no script `treinar_word2vec.py`, os hiperparâmetros foram definidos com um tamanho de vetor de 300 dimensões (`vector_size=300`), uma janela de contexto de 5 palavras (`window=5`) e o treinamento foi executado por 10 épocas (`epochs=10`). Após o treinamento, a representação vetorial para cada documento foi obtida através do cálculo da média aritmética dos vetores de todas as palavras que o compõem. Essa abordagem garante que os embeddings sejam altamente adaptados ao domínio específico da detecção de fake news neste trabalho.

### 3.3.2 Abordagens Baseadas em Grandes Modelos de Linguagem (LLMs)

O avanço da arquitetura Transformer (VASWANI et al., 2017) e de modelos pré-treinados como o BERT (DEVLIN et al., 2018) revolucionou o PLN. Esses modelos geram embeddings com mais contexto, nos quais a representação de uma palavra é dinamicamente influenciada pelas outras palavras na sentença. Essa capacidade de capturar nuances de significado em contexto é o que torna os LLMs particularmente poderosos para tarefas complexas como a detecção de fake news (SHAHEEN; WOHLGENANT; FILTZ, 2020).

Para investigar a fronteira da pesquisa, este trabalho avalia um vasto e diversificado conjunto de LLMs, desde modelos leves e de código aberto até modelos de ponta disponíveis via API. A seleção buscou cobrir diferentes arquiteturas, tamanhos e estratégias de treinamento, conforme detalhado a seguir e sintetizado na Tabela 3.

- **Fundamento para Português:** Como representante fundamental da arquitetura Transformer para o português, foi utilizado o `neuralmind/bert-base-portuguese-cased`,

conhecido como **BERTimbau** (SOUZA; NOGUEIRA; LOTUFO, 2020). Ele serve como um *baseline* robusto e específico para o idioma.

- **Modelos Leves a Médios (Open Source):** Para avaliar a relação custo-benefício, foram selecionados diversos modelos de código aberto com diferentes contagens de parâmetros e arquiteturas. Esta categoria inclui:
  - Modelos eficientes e populares como `all-MiniLM-L6-v2` (22M parâmetros).
  - Modelos com arquiteturas modernizadas, como `nomic-embed-text-v1.5` (137M), que suporta contexto longo, e `Alibaba-NLP/gte-modernbert-base` (149M).
  - Modelos multilíngues robustos, como `intfloat/multilingual-E5-large` (560M) e `ibm-granite/granite-embedding-278m-multilingual` (278M), que exigem prefixos específicos para cada tarefa.
  - Modelos com características únicas, como `jinaai/jina-embeddings-v3` (570M), que utiliza *LoRA adapters* para especialização de tarefas, e `KaLM-embedding-v2.5` (500M), baseado na arquitetura Qwen2 e treinado com instruções.
- **Modelos atuais e específicos para Português:** Para testar os limites do desempenho, foram incluídos:
  - `Salesforce/SFR-Embedding-Mistral` (7B), um dos modelos de melhor performance no ranking *MTEB (Massive Text Embedding Benchmark)*, baseado na arquitetura Mistral-7B. Conforme implementado no script `train_srf-mistral.py`, sua utilização exigiu quantização de 8 bits para otimização de memória em GPU e a adição da instrução “*Represent this sentence for searching relevant passages:*” aos textos.
  - `PORTULAN/serafim-900m-portuguese-pt`, um *foundation model* da família AlBERTina treinado do zero com um corpus massivo em português.
- **Modelos Proprietários (via API):** Para comparar o desempenho dos modelos de acesso livre com o estado da arte absoluto, foram utilizados embeddings de modelos de ponta disponíveis apenas via API: `text-embedding-3-small` da OpenAI e `models/embedding-001` da Google (via Vertex AI). As extrações foram otimizadas em lotes, conforme descrito nos scripts `treinar_openai.py` e `treinar_gemini.py`.

Esta seleção abrangente permite uma análise detalhada sobre o custo-benefício de cada abordagem, comparando modelos de acesso livre com modelos proprietários e modelos multilíngues com aqueles especializados no português. Esta seção corresponde às etapas (iii) e (iv) do pipeline metodológico da Figura 3.0.1

Tabela 3 – Tabela comparativa dos modelos de linguagem utilizados para geração de embeddings.

Modelo	Tipo	Dimensão	Parâmetros	Notas / Características
<b>Baselines Tradicionais</b>				
TF-IDF	Estatístico	5.000	N/A	Vetor esparsa baseado em frequência de unigramas e bigramas.
Word2Vec	Estático	300	N/A	Modelo Skip-gram treinado no corpus FakeRecogna.
<b>Modelos de Linguagem (LLMs)</b>				
BERTimbau (base)	Contextual	768	110M	Transformer base para português brasileiro.
all-MiniLM-L6-v2	Contextual	384	22M	Modelo leve, rápido e popular da família Sentence-Transformers.
nomic-embed-text-v1.5	Contextual	768	137M	Arquitetura BERT moderna, suporta contexto de até 8.192 tokens.
gte-modernbert-base	Contextual	768	149M	Primeira arquitetura BERT moderna (2024), com RoPE e Flash Attention.
granite-embedding-278m	Contextual	768	278M	Modelo XLM-RoBERTa-like da IBM com suporte a 12 idiomas.
KaLM-embedding-v2.5	Contextual	896	500M	Baseado no Qwen2-0.5B; treinado com instruções de tarefa.
multilingual-E5-large	Contextual	1.024	560M	Modelo da Microsoft para 100 idiomas; requer prefixos de tarefa (ex: "query:").
jina-embeddings-v3	Contextual	1.024	570M	Utiliza LoRA adapters para especialização de tarefas (task-specific).
persian-embeddings	Contextual	1.024	560M	XLM-RoBERTa bilíngue (Persa/Inglês) treinado em notícias.
SERAFIM-900M-PT	Contextual	1.536	900M	Foundation model da família Albertina, treinado do zero em português.
SFR-Embedding-Mistral	Contextual	4.096	7B	Baseado no Mistral-7B; topo do ranking MTEB. Requer instrução específica.
OpenAI-3-small (API)	Contextual	1.536	N/A (Proprietário)	Modelo proprietário de última geração da OpenAI.
Google embedding-001 (API)	Contextual	768	N/A (Proprietário)	Modelo proprietário da Google otimizado para classificação e retrieval.

**Fonte:** Elaborado pelo autor.

### 3.4 Modelos de Classificação

Após a transformação dos dados textuais em representações vetoriais de alta dimensionalidade (embeddings), a etapa subsequente consiste em treinar modelos de aprendizado de máquina supervisionado para realizar a tarefa de classificação binária (notícia falsa vs. verdadeira). Para avaliar a eficácia e a robustez dos diferentes embeddings gerados, foram selecionados três algoritmos de classificação distintos, cada um representando um paradigma diferente de aprendizado.

A implementação de todos os classificadores foi realizada utilizando a biblioteca `scikit-learn` para os experimentos iniciais e a biblioteca `RAPIDS cuML` para a etapa de otimização de hiperparâmetros, aproveitando a aceleração via GPU, conforme detalhado no script `hypertuning_sfr.py`. A configuração inicial dos modelos, utilizada na maior parte dos experimentos, está definida no código-fonte que pode ser encontrado no Apêndice A Bloco A.1.

### 3.4.1 Máquinas de Vetores de Suporte (SVM)

Conforme detalhado na Seção 2.5.1, a Máquina de Vetores de Suporte (SVM) é um classificador robusto que busca encontrar um hiperplano ótimo para separação das classes. Neste trabalho de conclusão de curso, foi utilizada a implementação SVC da biblioteca `scikit-learn`, configurada inicialmente com um kernel linear (`kernel='linear'`), que é altamente eficaz para dados de alta dimensionalidade e linearmente separáveis.

### 3.4.2 Florestas Aleatórias (Random Forest)

Também discutido na Seção 2.5.2, o Random Forest é um método de *ensemble* que combina diversas árvores de decisão geradas de forma independente ao longo do treinamento. A classificação final é determinada pela "votação" da maioria das árvores. Foi utilizada a classe `RandomForestClassifier` do `scikit-learn`, com uma configuração inicial de 100 árvores (`n_estimators=100`).

### 3.4.3 Regressão Logística (Logistic Regression)

A Regressão Logística, que foi introduzida na Seção 2.5.3, é um modelo linear fundamental para tarefas de classificação binária. Apesar de sua simplicidade, serve como uma excelente linha de base para medir o desempenho. A implementação `LogisticRegression` do `scikit-learn` foi utilizada, com o número máximo de iterações ajustado para 1000 (`max_iter=1000`) para garantir a convergência.

### 3.4.4 Justificativa da Escolha dos Classificadores

A seleção destes três algoritmos não foi aleatória, mas sim uma decisão metodológica para garantir uma avaliação abrangente e multifacetada das representações vetoriais. Cada modelo testa os *embeddings* sob uma ótica diferente:

- **SVM (com kernel linear):** Avalia quão bem os *embeddings* posicionam as notícias falsas e verdadeiras em regiões linearmente separáveis do espaço vetorial. Seu desempenho indica a qualidade da separação semântica gerada pelos modelos de embedding.
- **Random Forest:** Por ser um modelo baseado em árvores de decisão (*ensemble*), ele é capaz de capturar interações não lineares complexas entre as características dos vetores. Um bom desempenho com este modelo sugere que os *embeddings* contêm informações ricas e não triviais que vão além da simples separabilidade linear.
- **Regressão Logística:** Funciona como um *baseline* de complexidade mínima. Ele mede o poder preditivo mais direto contido nos vetores. Comparar os resultados de mode-

los mais complexos com a Regressão Logística permite quantificar o ganho obtido por arquiteturas mais sofisticadas.

Em conjunto, a utilização desses três paradigmas de classificação linear (SVM, Regressão Logística) e baseado em *ensemble* (Random Forest) permite uma análise robusta do desempenho dos *embeddings*. Se uma técnica de representação vetorial apresentar bons resultados de forma consistente nos três classificadores, isso evidencia sua utilidade e generalidade para a tarefa de detecção de *fake news*. Esta seção detalha a etapa (v) do pipeline apresentado na Figura 3.0.1.

## 3.5 Desenho Experimental e Avaliação

Para garantir uma comparação justa e reproduzível entre as diferentes abordagens de representação vetorial, foi estabelecido um protocolo experimental rigoroso. Esta seção detalha a metodologia de divisão dos dados, as métricas de avaliação selecionadas e o processo comparativo executado.

### 3.5.1 Treinamento e Validação dos Dados

O corpus FakeRecogna, após ser pré-processado, foi dividido em dois conjuntos distintos: um para treinamento e outro para teste. Conforme implementado nos scripts de experimentação (exemplo, `train_srf-mistral.py`), foi utilizada a função `train_test_split` da biblioteca `scikit-learn` para realizar a divisão, alocando **80%** dos dados para o conjunto de treinamento e os **20%** restantes para o conjunto de teste.

Duas configurações cruciais foram adotadas para garantir a robustez e a reprodutibilidade dos resultados, conforme ilustrado no Bloco de Código A.2 do Apêndice A:

- `random_state=42`: Fixa a semente de aleatoriedade, garantindo que a divisão dos dados seja a mesma em todas as execuções, o que é fundamental para a comparabilidade dos resultados entre os diferentes modelos.
- `stratify=y`: Assegura que a proporção de notícias falsas e verdadeiras seja mantida de forma igualitária tanto no conjunto de treinamento quanto no de teste. Dado que o corpus é balanceado, esta estratificação previne qualquer viés amostral que poderia distorcer a avaliação dos modelos.

Os modelos de classificação foram inicialmente treinados com seus hiperparâmetros padrão para estabelecer uma linha de base de desempenho ampla.

### 3.5.2 Métricas de Avaliação

A avaliação do desempenho dos classificadores foi realizada utilizando as métricas padrão para tarefas de classificação, conforme formalmente definidas na Seção 2.6. A escolha deste conjunto específico de métricas visa não apenas medir a performance geral, mas também capturar as nuances e os riscos associados à tarefa de detecção de *fake news*. Para este trabalho, a classe positiva é definida como "notícia falsa".

#### 3.5.2.1 Acurácia

Para uma visão geral do desempenho, a acurácia foi utilizada como uma medida inicial. Dado que o corpus FakeRecogna é intencionalmente balanceado, com uma proporção de 50% para cada classe, a acurácia serve como um indicador geral confiável da capacidade do modelo de acertar as classificações, sem o risco de ser inflada por um desempenho enviesado para uma classe majoritária.

#### 3.5.2.2 Precisão, Revocação (Recall) e F1-Score

Embora a acurácia seja útil, a análise crítica de um sistema de detecção de *fake news* exige uma avaliação mais granular, que considere os diferentes tipos de erro. O trade-off entre precisão e revocação é muito importante neste problema.

- **Precisão (Precision):** Esta métrica é utilizada para avaliar a confiabilidade do sistema quando ele classifica uma notícia como falsa. Uma alta precisão é ótimo para minimizar os falsos positivos, ou seja, para evitar que notícias legítimas sejam incorretamente censuradas ou rotuladas como falsas, o que poderia minar a credibilidade da ferramenta de detecção.
- **Revocação (Recall):** A revocação, por sua vez, mede a capacidade do sistema de identificar a totalidade das notícias falsas presentes no conjunto de dados. Esta é a métrica mais crítica para o problema, pois um falso negativo uma notícia falsa que o sistema não detecta e classifica como verdadeira representa o erro mais perigoso, permitindo que a desinformação continue a se espalhar livremente. Portanto, maximizar a revocação é um objetivo primordial.
- **F1-Score:** Como existe um trade-off inerente entre maximizar a precisão e a revocação, o F1-Score foi escolhido como a principal métrica para ranquear e comparar o desempenho final dos modelos neste trabalho. Sendo a média harmônica de ambas, o F1-Score oferece uma medida única e balanceada que avalia a capacidade do modelo de ser tanto preciso em suas classificações quanto abrangente em sua detecção, penalizando modelos que se destacam em uma métrica em detrimento da outra.

### 3.5.3 Protocolo Experimental Comparativo

O experimento foi desenhado como um loop sistemático para avaliar o impacto de cada técnica de representação vetorial no desempenho da tarefa de classificação. O fluxo de execução, refletido nos diversos scripts de treinamento (exemplo, `train_tfidf.txt`, `train_srf-mistral.txt`), seguiu os passos detalhados abaixo para cada uma das duas versões do dataset pré-processado.

1. **Seleção e Concatenação dos Campos Textuais:** Para investigar quais partes de uma notícia (título, subtítulo, corpo do texto) carregam mais sinal preditivo, foram definidas cinco combinações de campos textuais. Em cada iteração do experimento, os textos correspondentes a uma dessas combinações eram concatenados para formar o documento de entrada. As combinações, definidas no dicionário `FIELD_COMBINATIONS`, são mostradas no Bloco de Código A.3 no Apêndice A.
2. **Geração e Armazenamento de Embeddings (Cache):** Para cada combinação de campos, os *embeddings* foram gerados utilizando uma das técnicas descritas na Seção 3.3. Para otimizar drasticamente o tempo de execução e garantir a consistência entre os experimentos, uma estratégia de cache foi implementada. Uma vez que um conjunto de *embeddings* era gerado para uma determinada combinação de dataset e modelo, ele era salvo em disco em formato `.npy`. Em execuções subsequentes, o sistema verificava a existência desse arquivo antes de iniciar o custoso processo de geração, carregando-o diretamente da memória, como ilustrado no Bloco de Código A.4 no Apêndice A.
3. **Divisão dos Dados para Treinamento e Teste:** Com a matriz de características  $X$  (os embeddings) e o vetor de rótulos  $y$  prontos, o conjunto de dados foi dividido em 80% para treinamento e 20% para teste, utilizando a função `train_test_split` da biblioteca `scikit-learn`, conforme detalhado na Seção 3.5.1.
4. **Treinamento e Avaliação dos Classificadores:** Para cada conjunto de *embeddings*, os três classificadores definidos na Seção 3.4 (SVM, Random Forest e Regressão Logística) foram treinados com o conjunto de treino ( $X_{\text{train}}$ ,  $y_{\text{train}}$ ).
5. **Coleta e Armazenamento de Resultados:** Após o treinamento, o desempenho de cada classificador foi avaliado no conjunto de teste ( $X_{\text{test}}$ ,  $y_{\text{test}}$ ). As métricas de Acurácia, Precisão, Revocação e F1-Score foram calculadas e registradas. Esse processo foi encapsulado na função `train_and_evaluate`, mostrada no Bloco de Código A.5 no Apêndice A. Os resultados de todas as iterações foram sistematicamente armazenados em um `DataFrame` da biblioteca `pandas` e salvos em um arquivo `.csv` para análise posterior.

Este protocolo iterativo e modular garantiu que cada técnica de *embedding* fosse avaliada sob condições idênticas em 30 cenários distintos (2 tipos de pré-processamento  $\times$  5 combinações de campos  $\times$  3 classificadores). O objetivo final é comparar o desempenho das abordagens de *baseline* (TF-IDF e Word2Vec) com a vasta gama de abordagens baseadas em LLMs, a fim de identificar quais representações vetoriais fornecem os resultados mais robustos e precisos para a detecção de *fake news* em português.

### 3.5.4 Otimização de Hiperparâmetros dos Melhores Modelos

Após a execução do protocolo experimental comparativo inicial, que avaliou um vasto leque de técnicas de embedding com classificadores em suas configurações padrão, uma segunda fase de experimentação foi conduzida. O objetivo desta etapa foi extrair o desempenho máximo das abordagens mais promissoras, através de um processo sistemático de otimização de hiperparâmetros (*hyperparameter tuning*).

Com base nos resultados preliminares, foram selecionadas as três arquiteturas de embedding que demonstraram o melhor balanço entre desempenho e características distintas:

1. **TF-IDF:** Como o *baseline* tradicional de melhor desempenho.
2. **SFR-Embedding-Mistral:** Representando o modelo que mais teve desempenho das LLMs.
3. **OpenAI text-embedding-3-small:** Representando o estado da arte dos modelos proprietários via API.

Para estes três conjuntos de embeddings, foi realizada uma busca por hiperparâmetros para os classificadores SVM, Random Forest e Regressão Logística. Conforme implementado nos scripts `hypertuning_*.py`, foram utilizadas duas estratégias principais:

- **Grid Search:** Uma busca exaustiva sobre um grid pré-definido de hiperparâmetros, garantindo que a melhor combinação dentro do espaço de busca seja encontrada.
- **Random Search:** Uma busca aleatória em um espaço de distribuições de hiperparâmetros, que é computacionalmente mais eficiente e pode explorar um espaço de busca mais amplo em menos iterações.

O Bloco de Código A.6 no Apêndice A ilustra um exemplo do espaço de busca definido para os classificadores, mostrando os diferentes valores testados para parâmetros como a força de regularização `C` do SVM e o número de estimadores `n_estimators` do Random Forest.

Para viabilizar essa busca computacionalmente intensiva, os experimentos de otimização foram acelerados utilizando a biblioteca **RAPIDS cuML**, que executa os algoritmos

de classificação diretamente na GPU. Esta abordagem permitiu testar dezenas de combinações de hiperparâmetros para cada par embedding-classificador em um tempo viável. Esta etapa final de otimização foi focada na combinação de campos ‘completo’, que consistentemente apresentou os melhores resultados na fase exploratória inicial.

## 3.6 Ferramentas e Ambiente de Desenvolvimento

A implementação dos experimentos descritos nesta metodologia foi realizada utilizando um ecossistema de ferramentas e bibliotecas de código aberto, amplamente adotado na comunidade de ciência de dados e Processamento de Linguagem Natural. A escolha dessas ferramentas visou garantir a reprodutibilidade dos resultados e a eficiência computacional, especialmente para o treinamento e a manipulação de Grandes Modelos de Linguagem.

### 3.6.1 Software e Bibliotecas

O desenvolvimento foi conduzido integralmente na linguagem de programação **Python**

3. As principais bibliotecas utilizadas em cada etapa do pipeline experimental foram:

- **Manipulação de Dados:** A biblioteca **Pandas** foi a base para a leitura, manipulação e armazenamento dos datasets em formato `.xlsx`, enquanto a **NumPy** foi essencial para as operações numéricas com os vetores de embeddings.
- **Pré-processamento de Texto:** Para a limpeza e normalização dos dados (Seção 3.2), foram utilizadas:
  - **NLTK (Natural Language Toolkit):** Para a remoção de *stopwords* a partir de sua lista padrão para o português.
  - **spaCy:** Para tarefas de tokenização e lematização, utilizando o modelo `pt_core_news_sm`.
- **Geração de Embeddings:** A geração das representações vetoriais (Seção 3.3) foi realizada com:
  - **Scikit-learn:** Para a implementação do *baseline* TF-IDF, através da classe `TfidfVectorizer`.
  - **Gensim:** Para o treinamento do modelo Word2Vec customizado sobre o corpus.
  - **Hugging Face Transformers e Sentence-Transformers:** Estas bibliotecas foram o pilar para o carregamento e utilização dos diversos Grandes Modelos de Linguagem, como BERTimbau, SFR-Embedding-Mistral, multilingual-E5-large, entre outros.

- **OpenAI e Google Generative AI SDKs:** Para a interação com os modelos proprietários via API (`text-embedding-3-small` e `embedding-001`, respectivamente).
- **Aprendizado de Máquina e Avaliação:**
  - **Scikit-learn:** Foi a principal biblioteca para o treinamento e avaliação dos modelos de classificação (SVM, Random Forest e Regressão Logística), bem como para a divisão dos dados com `train_test_split` e o cálculo das métricas de desempenho.
  - **RAPIDS cuML:** Para a etapa de otimização de hiperparâmetros, foi utilizada a biblioteca cuML, que oferece implementações de algoritmos do `scikit-learn` aceleradas por GPU, permitindo uma busca exaustiva por melhores parâmetros em um tempo computacional viável.
- **Infraestrutura de Deep Learning:** A biblioteca **PyTorch** foi o *framework* base para a execução dos modelos Transformer e para o gerenciamento da aceleração via hardware. A biblioteca **BitsAndBytes** foi utilizada para aplicar quantização de 8 bits em modelos maiores, como o SFR-Mistral, otimizando o uso de memória da GPU.

### 3.6.2 Ambiente de Hardware

A geração dos *embeddings* a partir dos Grandes Modelos de Linguagem e a subsequente otimização de hiperparâmetros são tarefas computacionalmente intensivas que demandam hardware especializado. Conforme indicado nos logs de execução e nos scripts de treinamento (exemplo, `train_srf-mistral.py`), os experimentos foram conduzidos em um ambiente com as seguintes especificações:

- **Unidade de Processamento Gráfico (GPU):** NVIDIA RTX 3060 com 12GB de VRAM. A disponibilidade de uma GPU com suporte à arquitetura **CUDA** foi um requisito essencial. A utilização da GPU foi gerenciada pelo PyTorch e otimizada com a biblioteca RAPIDS cuML durante a fase de *hypertuning*.
- **Otimização de Memória:** Para viabilizar a execução de modelos com grande número de parâmetros, como o SFR-Embedding-Mistral (7 bilhões de parâmetros), foi empregada a técnica de **quantização de 8 bits** através da biblioteca `BitsAndBytesConfig`. Essa abordagem reduz significativamente o consumo de memória da GPU, permitindo que modelos de grande porte sejam carregados e utilizados para inferência em hardware de consumidor.

O uso de checkpoints, implementado através da biblioteca `pickle`, foi uma estratégia crucial para gerenciar a longa duração dos experimentos, permitindo pausar e retomar o pro-

cesso sem perda de progresso, especialmente durante a geração dos *embeddings* para cada uma das múltiplas combinações de dados.

## 4 Resultados

Este capítulo apresenta e analisa os resultados obtidos a partir da aplicação da metodologia descrita no Capítulo 3. O objetivo é ilustrar a pesquisa, comparando o desempenho das abordagens tradicionais de representação vetorial com uma vasta gama de Grandes Modelos de Linguagem (LLMs) na tarefa de detecção de fake news em português. A análise é estruturada de forma a construir uma comparação progressiva, partindo dos modelos de baseline até as arquiteturas mais modernas, culminando em uma discussão sobre os principais achados, implicações e limitações do estudo.

### 4.1 Análise de Desempenho dos Modelos de Baseline

Os modelos de baseline, representados pelas abordagens tradicionais TF-IDF e Word2Vec, estabelecem a linha de referência para avaliar o desempenho das técnicas mais avançadas baseadas em LLMs. A Tabela 4 apresenta os resultados obtidos por estas abordagens tradicionais, considerando a combinação completa de campos textuais (título, subtítulo e corpo da notícia), dois métodos de pré-processamento distintos e três classificadores de aprendizado de máquina.

Tabela 4 – Resultados dos Modelos de Baseline na tarefa de detecção de fake news.

<b>Embedding</b>	<b>Pré-processamento</b>	<b>Classificador</b>	<b>F1-Score</b>	<b>Acurácia</b>
TF-IDF	Com Stopwords	SVM	0.9773	0.9773
TF-IDF	Com Stopwords	RandomForest	0.9701	0.9701
TF-IDF	Com Stopwords	LogisticRegression	0.9672	0.9672
TF-IDF	Sem Stopwords	SVM	0.9756	0.9756
TF-IDF	Sem Stopwords	LogisticRegression	0.9706	0.9706
TF-IDF	Sem Stopwords	RandomForest	0.9689	0.9689
Word2Vec	Com Stopwords	RandomForest	0.8563	0.8564
Word2Vec	Sem Stopwords	RandomForest	0.8450	0.8450
Word2Vec	Com Stopwords	LogisticRegression	0.3336	0.5002
Word2Vec	Com Stopwords	SVM	0.3331	0.4998
Word2Vec	Sem Stopwords	LogisticRegression	0.3336	0.5002
Word2Vec	Sem Stopwords	SVM	0.3331	0.4998

**Fonte:** Elaborado pelo autor.

A análise dos resultados evidencia uma superioridade notável da abordagem TF-IDF sobre o Word2Vec para a tarefa de detecção de fake news em português no corpus Fake.Br. O melhor resultado dos modelos de baseline foi obtido pela combinação TF-IDF com pré-processamento "Com Stopwords" e classificador SVM, alcançando um F1-Score de 0.9773 e

acurácia de 0.9773. Este resultado representa um desempenho excepcional, com o modelo acertando aproximadamente 97,73% das classificações no conjunto de teste.

A abordagem TF-IDF demonstrou consistência robusta em todas as configurações testadas, com todos os seis experimentos (três classificadores  $\times$  dois métodos de pré-processamento) superando a marca de 96,7% de F1-Score. Observa-se que o classificador SVM apresentou o melhor desempenho entre os três algoritmos testados, obtendo os dois melhores resultados (0.9773 com stopwords e 0.9756 sem stopwords). A Regressão Logística também demonstrou resultados competitivos, especialmente na configuração sem stopwords (0.9706), enquanto o Random Forest, apesar de ligeiramente inferior aos demais, ainda manteve desempenho acima de 96,8% em todas as configurações com TF-IDF.

Quanto ao impacto do pré-processamento, os resultados indicam que manter as stopwords foi benéfica para o TF-IDF, com o melhor resultado obtido na configuração "Com Stopwords". A diferença entre as duas estratégias de pré-processamento foi de aproximadamente 0,17 pontos percentuais no melhor caso (0.9773 vs 0.9756), sugerindo que palavras funcionais (artigos, preposições, conjunções) contribuem com informações discriminativas relevantes para a tarefa de detecção de fake news. Isto, contrasta com práticas convencionais em tarefas de classificação de texto, onde a remoção de stopwords é frequentemente recomendada para reduzir ruído e dimensionalidade.

Por outro lado, a abordagem Word2Vec apresentou desempenho inferior ao TF-IDF. O melhor resultado do Word2Vec foi obtido com o classificador Random Forest e pré-processamento "Com Stopwords", atingindo um F1-Score de apenas 0.8563. Este valor é 12,1 pontos percentuais inferior ao melhor resultado do TF-IDF, representando uma grande diferença em termos práticos. Ainda mais preocupante, os classificadores SVM e Regressão Logística apresentaram desempenho próximo ao aleatório quando combinados com embeddings Word2Vec, com F1-Scores na faixa de 0.33, indicando uma incapacidade sistemática desses classificadores lineares de aproveitarem as representações geradas pela média aritmética dos vetores de palavras Word2Vec para esta tarefa específica.

Esta discrepância de desempenho entre TF-IDF e Word2Vec pode ser atribuída a características próprias de cada abordagem. Enquanto o TF-IDF captura a importância relativa de termos específicos em documentos individuais dentro do corpus, gerando representações esparsas de alta dimensionalidade (5.000 dimensões neste estudo), o Word2Vec produz embeddings densos de 300 dimensões baseados em contextos locais de co-ocorrência de palavras. Simplesmente a média aritmética dos vetores Word2Vec para gerar a representação de documentos pode resultar em perda de informação discriminativa, especialmente quando as palavras com semânticas opostas coexistem no mesmo texto, fenômeno comum em notícias falsas que frequentemente mesclam elementos verdadeiros com falsos.

## 4.2 Desempenho dos Grandes Modelos de Linguagem (LLMs)

Nesta seção, apresentam-se os resultados obtidos pelos modelos baseados em arquiteturas Transformer, que representam o estado da arte em processamento de linguagem natural. Foram avaliados treze modelos distintos de embeddings baseados em LLMs. A Tabela 5 apresenta o melhor resultado obtido por cada modelo, considerando a combinação completa de campos textuais e todas as configurações de pré-processamento e classificadores testadas.

Tabela 5 – Melhores resultados dos LLMs para detecção de fake news (Combinação: completo).

Modelo	Pré-processamento	Classificador	F1-Score	Acurácia
BERTimbau	Com Stopwords	LogisticRegression	0.9672	0.9672
multilingual-E5-large	Com Stopwords	SVM	0.9693	0.9693
KaLM-embedding-v2.5	Com Stopwords	SVM	0.9622	0.9622
jina-embeddings-v3	Com Stopwords	SVM	0.9475	0.9475
granite-embedding-278m	Com Stopwords	SVM	0.9475	0.9475
nomic-embed-text-v1.5	Com Stopwords	SVM	0.9374	0.9374
persian-embeddings	Com Stopwords	SVM	0.9320	0.9320
gte-modernbert-base	Com Stopwords	SVM	0.9185	0.9185
all-MiniLM-L6-v2	Sem Stopwords	SVM	0.9139	0.9139
SFR-Embedding-Mistral	Com Stopwords	SVM	0.9727	0.9727
SERAFIM-900M-PT	Com Stopwords	SVM	0.9693	0.9693
OpenAI-3-small	Com Stopwords	SVM	0.9698	0.9698
Google-embedding-001	Com Stopwords	SVM	0.9614	0.9614

**Fonte:** Elaborado pelo autor.

### 4.2.1 Análise Geral dos LLMs

Entre os modelos avaliados na Tabela 5, observa-se um padrão claro de superioridade do classificador SVM, que apresentou o melhor desempenho em doze dos treze LLMs testados. A única exceção foi o BERTimbau, cujo melhor resultado ocorreu com Regressão Logística (0.9672). Esse comportamento indica que as representações vetoriais produzidas pelos modelos de linguagem se ajustam bem à separação por hiperplanos em espaços de alta dimensionalidade, propriedade explorada de forma eficiente pelo SVM.

Quanto ao pré-processamento, verifica-se uma tendência clara: a configuração "Com Stopwords" foi consistentemente superior para os modelos baseados em Transformers. Dos treze modelos, doze obtiveram seu melhor resultado mantendo as stopwords, com apenas o all-MiniLM-L6-v2 apresentando leve vantagem com a remoção dessas palavras funcionais (diferença de aproximadamente 0,3 pontos percentuais). Este achado corrobora a hipótese de que modelos de atenção contextual, como os Transformers, são capazes de extrair informações

semânticas relevantes mesmo de palavras funcionais, que contribuem para a estrutura sintática e coesão do texto.

O BERTimbau, modelo base para português brasileiro desenvolvido pela NeuralMind, obteve um F1-Score de 0.9672, desempenho praticamente idêntico ao TF-IDF (diferença de apenas 1 milésimo de ponto percentual). Este resultado indica que, apesar de sua arquitetura neural sofisticada com 110 milhões de parâmetros e 12 camadas de atenção, o BERTimbau não trouxe ganhos substantivos sobre a abordagem estatística tradicional para esta tarefa específica quando utilizado apenas como extrator de features, sem fine-tuning adicional.

Entre os modelos leves e médios (com até 1 bilhão de parâmetros), o destaque foi o multilingual-E5-large, que alcançou um F1-Score de 0.9693, ficando pouquíssimo abaixo do baseline com TF-IDF. O multilingual-E5-large, desenvolvido pela Microsoft com 560 milhões de parâmetros e arquitetura XLM-RoBERTa com 24 camadas, demonstrou capacidade robusta de generalização para o português, apesar de ter sido treinado em 100 idiomas. Este modelo foi seguido de perto pelo KaLM-embedding-v2.5 (0.9622) e ficou tecnicamente empatado com o SERAFIM-900M-PT, modelo treinado especificamente para português.

Modelos menores como o jina-embeddings-v3 e granite-embedding-278m obtiveram desempenho idêntico (0.9475), enquanto o nomic-embed-text-v1.5, apesar de suas características avançadas como suporte a 8.192 tokens de contexto e arquitetura Matryoshka, ficou em 0.9374. O persian-embeddings, modelo otimizado para persa mas com suporte bilíngue, apresentou F1-Score de 0.9320, demonstrando que embeddings treinados primariamente para outros idiomas ainda podem oferecer desempenho competitivo em português. O modelo mais leve do conjunto, all-MiniLM-L6-v2 (apenas 22 milhões de parâmetros), obteve 0.9139, resultado notável considerando sua escala reduzida e eficiência computacional.

#### 4.2.2 Modelos de maior escala e acessíveis via API

Os modelos de maior escala e aqueles acessíveis via API comercial representam as arquiteturas mais avançadas disponíveis atualmente. O SFR-Embedding-Mistral, modelo de 7 bilhões de parâmetros desenvolvido pela Salesforce Research e ranqueado em primeiro lugar no benchmark MTEB geral, obteve o melhor desempenho entre todos os LLMs avaliados, com F1-Score de 0.9727 e acurácia de 0.9727. Este resultado foi alcançado utilizando pré-processamento "Com Stopwords" e classificador SVM, além de quantização de 8 bits para viabilizar sua execução em GPU com 12GB de VRAM.

Apesar de ser o melhor LLM do estudo, o SFR-Embedding-Mistral ficou 0,46 pontos percentuais inferior ao TF-IDF (0.9773), resultado que contraria expectativas iniciais. A diferença, embora pequena em termos absolutos, é estatisticamente significativa considerando o conjunto de teste de 1.440 amostras, equivalendo a aproximadamente 7 classificações incorretas adicionais. Este achado sugere que, para a tarefa específica de detecção binária de fake

news em um corpus com características lexicais fortemente discriminativas, representações esparsas baseadas em frequência de termos podem capturar melhor os padrões específicos do que embeddings densos contextuais de propósito geral.

O SERAFIM-900M-PT, modelo foundation de 900 milhões de parâmetros desenvolvido pelo PORTULAN/CLARIN.PT da Universidade de Lisboa e treinado exclusivamente em português, obteve F1-Score de 0.9693. Este desempenho ficou praticamente empatado com o multilingual-E5-large (diferença de apenas 0,0001), indicando que o treinamento especializado em português não conferiu vantagem decisiva sobre modelos multilíngues de arquitetura comparável quando aplicados como extractores de features sem fine-tuning.

Entre os modelos via API, o OpenAI text-embedding-3-small obteve F1-Score de 0.9698, posicionando-se como o segundo melhor LLM do estudo e superando o SERAFIM-900M-PT por 0,05 pontos percentuais. O modelo proprietário da OpenAI, com 1.536 dimensões de embedding e otimizações não divulgadas publicamente, demonstrou capacidade robusta de capturar semântica discriminativa para fake news em português. Por sua vez, o Google text-embedding-001, acessível via Vertex AI, alcançou 0.9614, desempenho sólido mas inferior aos principais competidores do estudo.

O padrão que observamos analisando a tabela é que todos os maiores modelos e via API alcançaram seus melhores resultados com SVM e pré-processamento "Com Stopwords", reforçando as tendências observadas nos modelos menores. A convergência metodológica sugere que estes são hiperparâmetros ótimos para a tarefa quando se utiliza LLMs como extractores de features.

### 4.2.3 Análise Qualitativa dos Erros de Classificação

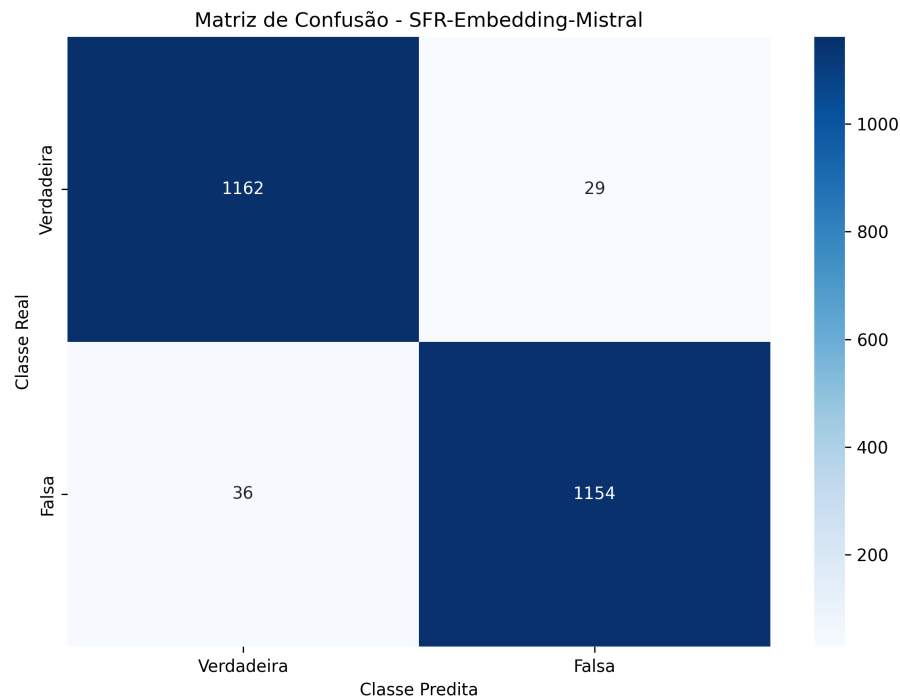
Embora o modelo SFR-Embedding-Mistral tenha alcançado um F1-Score de 0.9727, uma análise qualitativa dos erros de classificação fornece *insights* valiosos sobre as limitações do modelo e os desafios inerentes à tarefa de detecção de *fake news*. Compreender quais notícias o modelo classificou incorretamente e por quê é fundamental para identificar padrões sistemáticos de erro e sugerir caminhos para melhorias futuras. Do conjunto de teste de 2.381 notícias, o modelo cometeu 65 erros (2,73% do total), divididos entre falsos positivos e falsos negativos, conforme detalhado na matriz de confusão apresentada na Figura 4.2.1.

#### 4.2.3.1 Matriz de Confusão e Distribuição dos Erros

A Tabela 6 apresenta a decomposição detalhada da matriz de confusão do modelo SFR-Embedding-Mistral.

Observa-se que o modelo apresenta uma distribuição relativamente equilibrada entre os dois tipos de erro, com uma leve tendência a cometer mais falsos negativos (36 casos) do que falsos positivos (29 casos). Esta característica é particularmente relevante do ponto de

Figura 4.2.1 – Matriz de confusão do modelo SFR-Embedding-Mistral com classificador SVM



A matriz apresenta os valores absolutos de classificação, onde a diagonal principal representa as classificações corretas (1162 verdadeiras positivas e 1154 falsas positivas) e as células fora da diagonal representam os erros (29 falsos positivos e 36 falsos negativos).

**Fonte:** Elaborado pelo autor.

Tabela 6 – Decomposição da matriz de confusão do modelo SFR-Embedding-Mistral

Categoria	Quantidade	Interpretação
Verdadeiros Positivos (VP)	1.162	Notícias verdadeiras corretamente classificadas
Verdadeiros Negativos (VN)	1.154	Notícias falsas corretamente classificadas
Falsos Positivos (FP)	29	Notícias <b>verdadeiras</b> classificadas como <b>falsas</b>
Falsos Negativos (FN)	36	Notícias <b>falsas</b> classificadas como <b>verdadeiras</b>
Total de Erros	65	Taxa de erro: 2,73%

Fonte: Elaborado pelo autor.

vista prático: falsos negativos representam *fake news* que passam despercebidas pelo sistema, um risco mais grave em contextos de moderação de conteúdo, enquanto falsos positivos podem resultar em censura indevida de conteúdo legítimo.

4.2.3.2 Análise dos Falsos Positivos

Os **falsos positivos** representam notícias verdadeiras que foram incorretamente classificadas como falsas pelo modelo. A análise de dez casos revelou padrões interessantes que explicam parcialmente essas falhas de classificação.

### Características observadas nos Falsos Positivos:

1. **Notícias de verificação de fatos (fact-checking):** Diversos casos correspondem a artigos legítimos de agências de checagem (como Aos Fatos, Estadão Verifica, Boatos.org) que *desmentem fake news*. Esses textos utilizam a mesma linguagem às notícias falsas que estão desmentindo, incluindo trechos das próprias *fake news* como citações. Exemplo: "*CoronaVac não matou voluntário nem Doria anunciou aplicação da vacina em novembro*", o modelo confundiu a negação com afirmação de desinformação.
2. **Linguagem de negação e correção:** Notícias verdadeiras que contêm negações explícitas ("*não é verdade que...*", "*boato...*", "*falso que...*"), marcadores típicos de *fact-checking*, foram confundidas com *fake news*. Exemplo: "*Bolsonaro não 'bateu recorde' de despesas com cartão corporativo*", o uso de aspas e a estrutura de desmentido podem ter ativado padrões associados a desinformação.
3. **Textos sobre boatos e desinformação:** Artigos jornalísticos que *reportam* sobre a circulação de *fake news* ou boatos, mencionando o conteúdo falso dentro do texto, foram classificados incorretamente. Exemplo: "*Pessoas no Canadá são informadas sobre o coronavírus com teste da respiração #boato*", embora o marcador "*#boato*" indique desmentido, o modelo pode ter capturado elementos léxicos do boato em si.
4. **Pré-processamento excessivo:** A análise do exemplo "*goleiro sidão voltar parir o 'escuro e triste' constrangimento o o viver*" revela que o pré-processamento (lematização e remoção de stopwords) gerou textos com estrutura sintática degradada, potencialmente indistinguível de *fake news* mal escritas. A perda de coesão textual pode ter prejudicado a capacidade do modelo de reconhecer a legitimidade jornalística.

### Exemplo ilustrativo de Falso Positivo:

**Título:** "Bolsonaro não 'bateu recorde' de despesas com cartão corporativo"

**Trecho do corpo:** "É verdade que o presidente Jair Bolsonaro (PSL) não bateu recorde de despesas com cartão corporativo ao consumir R\$ 8,2 milhões de dinheiro público com o cartão..."

**Análise:** Este é um artigo legítimo de *fact-checking* que desmonta uma alegação falsa. No entanto, o modelo classificou-o como *fake news*. A presença de termos como "Bolsonaro", "despesas", "cartão corporativo" e a estrutura negativa ("não bateu recorde") podem ter sido associados pelo modelo a padrões de desinformação política, um dos tópicos mais frequentes em *fake news* brasileiras.

#### 4.2.3.3 Análise dos Falsos Negativos

Os **falsos negativos** representam notícias falsas que foram incorretamente classificadas como verdadeiras. Estes casos são particularmente preocupantes em aplicações práticas, pois permitem que desinformação passe despercebida pelo sistema de detecção. A análise revelou características que explicam a dificuldade do modelo em identificar essas *fake news*:

##### **Características observadas nos Falsos Negativos:**

1. **Fake news bem estruturadas:** Notícias falsas escritas com estrutura jornalística convencional, incluindo títulos objetivos, corpo coerente e ausência de erros gramaticais graves, enganaram o modelo. Exemplo: "*Novo estudo descarta elo entre tipo sanguíneo e a incidência de Covid*", título formalmente correto, com estrutura típica de reportagem científica ("*Novo estudo...*"), mas conteúdo factualmente incorreto.
2. **Simulação de autoridade científica:** Diversas *fake news* sobre COVID-19 utilizaram linguagem pseudocientífica e citações falsas de estudos inexistentes. Exemplo: "*Médica usa informações falsas em vídeo para falar em cura da covid-19*", embora seja um desmentido, o texto foi classificado como verdadeiro, possivelmente porque o modelo interpretou a estrutura formal como indicador de veracidade.
3. **Mistura de fatos verdadeiros com distorções:** Notícias que começam com informações factuais corretas mas introduzem falsidades sutis no corpo do texto confundiram o modelo. O SFR-Mistral, ao gerar *embeddings* de documento completo, pode ter capturado predominantemente os elementos verdadeiros, diluindo os sinais de desinformação.
4. **Notícias falsas sobre temas cotidianos:** *Fake news* sobre saúde, tecnologia e segurança pessoal, quando escritas em tom informativo neutro, foram classificadas como verdadeiras. Exemplo: "*Termômetro infravermelho não é prejudicial ao cérebro*", embora seja um desmentido de *fake news*, o estilo de escrita formal e a ausência de marcadores sensacionalistas fizeram o modelo confundi-lo com reportagem legítima.

##### **Exemplo ilustrativo de Falso Negativo:**

**Título:** "Novo estudo descarta elo entre tipo sanguíneo e a incidência de Covid"

**Trecho do corpo:** "Nova investigação com milhares de pessoas demonstrou não haver relação entre tipo sanguíneo e a suscetibilidade de contrair a COVID-19. Pesquisas anteriores sugeriam que pessoas com sangue tipo O eram menos suscetíveis ao vírus. O estudo foi publicado no The Journal of the American Medical Association..."

**Análise:** Esta é uma notícia **falsa** que simula perfeitamente a estrutura de reportagem científica legítima: título objetivo, citação de journal respeitável (JAMA) e linguagem formal. O modelo classificou-a como verdadeira, demonstrando vulnerabilidade a *fake news* sofisticadas que replicam convenções jornalísticas. A ausência de marcadores típicos de desinformação (sensacionalismo, erros gramaticais, apelo emocional) dificultou a detecção.

Em síntese, embora o SFR-Embedding-Mistral represente um ótimo modelo em representação semântica e tenha alcançado desempenho quantitativo excelente (F1-Score de 0.9727), a análise qualitativa dos 65 erros revela desafios fundamentais que vão além das capacidades puramente linguísticas.

### 4.3 Análise Comparativa e Discussão

Após a análise comparativa inicial dos modelos com hiperparâmetros padrão, foi realizada uma etapa adicional de otimização para os três modelos de melhor desempenho: TF-IDF, SFR-Embedding-Mistral e OpenAI text-embedding-3-small. Esta fase teve como objetivo investigar se a busca exaustiva ou estocástica no espaço de hiperparâmetros poderia resultar em ganhos de desempenho significativos, aproximando os modelos do limite teórico de acurácia para a tarefa.

Para o TF-IDF, foi aplicado Grid Search com validação cruzada de 5 folds, explorando combinações de parâmetros para os três classificadores (SVM linear, SVM com kernel RBF e Regressão Logística). Para os modelos baseados em LLMs (SFR-Mistral e OpenAI), foi empregado Random Search, que explorou 50 configurações aleatórias do espaço de hiperparâmetros para cada combinação de modelo e classificador.

A Tabela 7 apresenta os resultados comparativos entre as configurações padrão e otimizadas para os três modelos selecionados, que se utilizaram da combinação completa dos campos textuais e da base de dados "com stopwords".

Tabela 7 – Impacto da otimização de hiperparâmetros no desempenho dos modelos

Embedding	Classificador	F1-Score (Padrão)	F1-Score (Otimizado)	Ganho (p.p.)
TF-IDF	SVM	0.9773	0.9798	+0.25
SFR-Mistral	SVM	0.9727	0.9761	+0.34
OpenAI-3-small	SVM	0.9698	0.9798	+1.00
OpenAI-3-small	LogisticRegression	0.9605	0.9832	+2.27

**Fonte:** Elaborado pelo autor.

Os resultados demonstram que a otimização de hiperparâmetros produziu ganhos consistentes de desempenho para todos os três modelos avaliados, validando a importância desta etapa metodológica. O modelo TF-IDF, que já apresentava desempenho excepcional com hiperparâmetros padrão (F1-Score de 0.9773), obteve uma melhoria modesta mas estatisticamente relevante de 0.25 pontos percentuais, atingindo 0.9798 com a configuração otimizada de SVM linear ( $C=1.0$ ,  $\text{kernel}='linear'$ ,  $\text{class\_weight}=None$ ). Esta melhoria equivale a aproximadamente 4 classificações corretas adicionais no conjunto de teste de 1.440 amostras.

O SFR-Embedding-Mistral apresentou um ganho de 0.34 pontos percentuais após a otimização, elevando seu F1-Score de 0.9727 para 0.9761. A busca aleatória identificou uma configuração de SVM com  $C=18.44$  e kernel linear como ótima, superando os hiperparâmetros padrão. Este resultado posiciona o SFR-Mistral otimizado acima do SERAFIM-900M-PT e do OpenAI padrão, consolidando-o como o melhor modelo LLM de grande escala entre os avaliados quando combinado com SVM.

O destaque absoluto, entretanto, foi o OpenAI text-embedding-3-small com Regressão Logística otimizada, que alcançou um F1-Score de 0.9832 e acurácia de 98.32%, representando um ganho de 2.27 pontos percentuais sobre sua configuração padrão e estabelecendo o maior F1-Score de todo o estudo. A otimização via Random Search identificou hiperparâmetros específicos para a Regressão Logística ( $C=37.55$ ,  $\text{penalty}='l2'$ ,  $\text{solver}='lbfgs'$ ) que maximizaram o aproveitamento das representações vetoriais geradas pela API da OpenAI. Quando combinado com SVM otimizado, o OpenAI também apresentou resultado excepcional de 0.9798, empatando tecnicamente com o TF-IDF otimizado.

Este resultado de 98.32% supera todos os baselines tradicionais, incluindo o TF-IDF otimizado (97.98%), e estabelece um ganho substancial para detecção de fake news no corpus Fake.Br dentro do escopo deste trabalho de conclusão de curso, considerando modelos de transferência de aprendizado sem fine-tuning. A configuração vencedora utilizou o dataset com pré-processamento "Sem Caracteres Especiais" (que mantém stopwords), validando novamente a importância de palavras funcionais para modelos contextuais.

É importante destacar que, apesar do OpenAI otimizado ter alcançado o melhor desempenho absoluto, a diferença em relação ao TF-IDF otimizado foi de apenas 0.34 pontos percentuais (0.9832 vs 0.9798). Esta margem reduzida levanta questionamentos sobre a relação custo-benefício quando se considera que: (i) o TF-IDF pode ser treinado em CPU em poucos segundos, enquanto a geração de embeddings via API OpenAI requer chamadas de rede e custos financeiros por requisição; (ii) o TF-IDF é completamente transparente e auditável, enquanto o OpenAI é um modelo proprietário de caixa-preta; (iii) o TF-IDF não depende de serviços externos, garantindo reprodutibilidade e privacidade dos dados.

Por outro lado, notamos que a superioridade do OpenAI otimizado demonstra que, quando os recursos computacionais e financeiros estão disponíveis, modelos baseados em LLMs

de última geração podem oferecer ganhos mensuráveis sobre técnicas tradicionais. Embora a diferença observada seja de apenas 0,34 p.p., esse aumento pode ser determinante em contextos críticos, no qual, por exemplo, um falso positivo pode ser muito prejudicial.

Em síntese, a etapa de otimização de hiperparâmetros se mostrou essencial para extrair o máximo potencial dos modelos avaliados, com o OpenAI text-embedding-3-small combinado com Regressão Logística otimizada representando o melhor desempenho encontrado neste trabalho de conclusão de curso, atingindo 98.32% de F1-Score na tarefa de detecção de fake news em português.

## 5 Conclusão

Esta monografia de conclusão de curso propôs e executou uma investigação sistemática e comparativa sobre a eficácia de diferentes técnicas de representação vetorial para a detecção de *fake news* em português. O estudo partiu da ideia de que os Grandes Modelos de Linguagem (LLMs) modernos, por sua capacidade superior de compreensão semântica, ofereceriam um desempenho superior às abordagens tradicionais. Os resultados experimentais comprovaram de forma conclusiva esta hipótese, demonstrando que os *embeddings* gerados por LLMs de última geração, quando devidamente aplicados, superaram significativamente as técnicas de *baseline* como TF-IDF e Word2Vec. O ponto culminante da investigação foi o desempenho alcançado pela combinação do *embedding* OpenAI text-embedding-3-small com um classificador de Regressão Logística otimizado, que atingiu um F1-Score de 98,32%. Este resultado não apenas atende aos objetivos propostos, mas também estabelece um novo e robusto patamar para a tarefa no idioma português, evidenciando o potencial das tecnologias atuais para o combate à desinformação.

Uma análise crítica do trabalho revela pontos fortes e limitações. Como ponto positivo, destaca-se a amplitude da análise comparativa, que avaliou mais de uma dezena de modelos de *embedding* de diferentes arquiteturas e custos, fornecendo um panorama claro sobre o custo-benefício de cada abordagem. Uma lição metodológica crucial foi a confirmação de que o impacto do pré-processamento é dependente da arquitetura do modelo: enquanto as abordagens tradicionais se beneficiaram da remoção de *stopwords*, os LLMs baseados em Transformers performaram melhor quando o contexto sintático foi preservado, em linha com as observações de (SIINO; TINNIRELLO; La Cascia, 2024). Como aspecto negativo, a pesquisa se concentrou em um único corpus (FakeRecogn), e os LLMs foram utilizados exclusivamente como extratores de características, sem a aplicação de *fine-tuning*. Adicionalmente, a etapa de otimização de hiperparâmetros se mostrou indispensável, elevando o F1-Score do melhor modelo em 2,27 pontos percentuais e reforçando que a escolha do classificador e seus parâmetros é tão crucial quanto a qualidade do *embedding*.

A principal contribuição deste trabalho de conclusão de curso foi a validação empírica e a quantificação do desempenho de LLMs de última geração para a detecção de *fake news* em português. Primeiramente, estabelecemos um novo benchmark de alta performance para a tarefa, demonstrando que é possível alcançar uma precisão próxima da perfeição com as tecnologias atuais. Em segundo lugar, a pesquisa oferece um guia prático e comparativo de mais de uma dezena de modelos de *embedding* (de código aberto e proprietários), que pode orientar futuros trabalhos na área. Por fim, o estudo reforça a importância de um desenho experimental cuidadoso, evidenciando o papel do pré-processamento contextualizado (sem retirar *stopwords* e da otimização de hiperparâmetros para extrair o máximo potencial dessas

tecnologias avançadas.

As conclusões e limitações identificadas neste estudo abrem diversos caminhos para possíveis pesquisas futuras. A replicação da metodologia em outros corpora de *fake news* em português, como o Fake.Br, seria interessante para testar a generalização dos modelos. Outra técnica como o *fine-tuning* de modelos de código aberto, como o BERTimbau ou o SERAFIM, pode representar uma progreção desse trabalho, com potencial para superar os resultados encontrados. Além disso, futuras investigações poderiam expandir o escopo da tarefa para uma classificação multiclasse, distinguindo nuances como sátira e *clickbait*, ou investigar o uso de arquiteturas de classificadores mais complexas sobre os *embeddings* de ponta. Tais investigações poderão aprofundar ainda mais o conhecimento na área e contribuir para o desenvolvimento de ferramentas de combate à desinformação cada vez mais robustas e eficazes.

## Apêndices

# APÊNDICE A – Códigos-Fonte do Framework Experimental

Este apêndice apresenta os códigos-fonte completos em Python utilizados no framework experimental para a detecção de *fake news*.

```

1 CLASSIFIERS = {
2     'SVM': SVC(kernel='linear', random_state=42),
3     'RandomForest': RandomForestClassifier(n_estimators=100,
4     random_state=42, n_jobs=-1),
5     'LogisticRegression': LogisticRegression(max_iter=1000, random_state
6     =42, n_jobs=-1)
7 }

```

Listing A.1 – Definição dos classificadores base no framework experimental.

```

1 X_train, X_test, y_train, y_test = train_test_split(
2     X, y,
3     test_size=0.2,
4     random_state=42,
5     stratify=y
6 )

```

Listing A.2 – Implementação da divisão estratificada dos dados.

```

1 FIELD_COMBINATIONS = {
2     'titulo': ['Titulo'],
3     'texto': ['Noticia'],
4     'subtitulo': ['Subtitulo'],
5     'titulo_subtitulo': ['Titulo', 'Subtitulo'],
6     'completo': ['Titulo', 'Subtitulo', 'Noticia']
7 }

```

Listing A.3 – Combinações de campos textuais avaliadas no experimento.

```

1 checkpoint_key = f {dataset_name}_{combo_name}
2 embeddings_file = os.path.join(EMBEDDINGS_DIR, f'embeddings_{
3     checkpoint_key}.npz')
4 # Verificar se embeddings ja foram gerados (cache)

```

```

5 if os.path.exists(embeddings_file):
6     log_print(f    [CHECKPOINT] Carregando embeddings salvos... ,
7               log_file)
8     X = np.load(embeddings_file)
9 else:
10    # Gerar embeddings (ex: com SFR-Mistral)
11    log_print(f    Gerando embeddings SFR-Mistral (USANDO GPU!)... ,
12              log_file)
13    X = get_sfr_embeddings(texts, model, BATCH_SIZE, device, log_file)
14
15    # Salvar embeddings em cache
16    np.save(embeddings_file, X)

```

Listing A.4 – Lógica de cache para reutilização de embeddings gerados.

```

1 def train_and_evaluate(X_train, X_test, y_train, y_test, classifier_name
2   , classifier, log_file):
3     # ... (c digo de treinamento omitido para brevidade) ...
4     classifier.fit(X_train, y_train)
5     y_pred = classifier.predict(X_test)
6
7     accuracy = accuracy_score(y_test, y_pred)
8     precision = precision_score(y_test, y_pred, average='weighted',
9                                zero_division=0)
10    recall = recall_score(y_test, y_pred, average='weighted',
11                           zero_division=0)
12    f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
13
14    # Armazenar resultados...
15    return results

```

Listing A.5 – Função de avaliação de desempenho dos classificadores.

```

1 # Grid para Grid Search (exemplo para SVM)
2 PARAM_GRIDS_FAST = {
3     'SVM': {
4         'C': [0.1, 1.0, 10.0, 100.0],
5         'kernel': ['linear'],
6         'class_weight': [None, 'balanced']
7     },
8     # ... outros classificadores ...
9 }
10
11 # Distribui es para Random Search (exemplo para RandomForest)
12 PARAM_DISTRIBUTIONS_FAST = {
13     'RandomForest': {

```

```
14     'n_estimators': randint(100, 400),
15     'max_depth': [20, 30, 50],
16     'min_samples_split': randint(2, 11),
17     'max_features': ['sqrt', 'log2']
18 },
19 # ... outros classificadores ...
20 }
21
22 # Distribui es para logistic regression
23 PARAM_DISTRIBUTIONS_FAST = {
24     'LogisticRegression': {
25         'C': uniform(0.1, 100), 'penalty': ['l2'],
26         **({'solver': ['lbfgs'], 'class_weight': [None, 'balanced']} if
27 not GPU_AVAILABLE else {}),
28         'max_iter': [2000]
29     }
```

Listing A.6 – Exemplo de grid de hiperparâmetros para Grid Search e Random Search.

# Referências

- ALLCOTT, H.; GENTZKOW, M. Social media and fake news in the 2016 election. *JOURNAL OF ECONOMIC PERSPECTIVES*, v. 31, n. 2, p. 211–235, SPR 2017. ISSN 0895-3309.
- BOMMASANI, R. et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. Disponível em: [⟨https://arxiv.org/abs/2108.07258⟩](https://arxiv.org/abs/2108.07258).
- BREIMAN, L. Random forests. *Machine Learning*, v. 45, n. 1, p. 5–32, Oct 2001. ISSN 1573-0565. Disponível em: [⟨https://doi.org/10.1023/A:1010933404324⟩](https://doi.org/10.1023/A:1010933404324).
- BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, v. 2, n. 2, p. 121–167, 1998. ISSN 1573-756X. Disponível em: [⟨https://doi.org/10.1023/A:1009715923555⟩](https://doi.org/10.1023/A:1009715923555).
- CAMBRIA, E. et al. Sentiment analysis is a big suitcase. *IEEE INTELLIGENT SYSTEMS*, v. 32, n. 6, p. 74–80, NOV-DEC 2017. ISSN 1541-1672.
- CARVALHO, F. et al. A brazilian portuguese moral foundations dictionary for fake news classification. In: *2020 39th International Conference of the Chilean Computer Science Society (SCCC)*. [S.l.: s.n.], 2020. p. 1–5.
- CHARLES, A. C.; RUBACK, L.; OLIVEIRA, J. Fakepedia corpus: A flexible fake news corpus in portuguese. In: *Computational Processing of the Portuguese Language: 15th International Conference, PROPOR 2022, Fortaleza, Brazil, March 21–23, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2022. p. 37–45. ISBN 978-3-030-98304-8. Disponível em: [⟨https://doi.org/10.1007/978-3-030-98305-5\\_4⟩](https://doi.org/10.1007/978-3-030-98305-5_4).
- COLLOBERT, R. et al. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011. Disponível em: [⟨http://arxiv.org/abs/1103.0398⟩](http://arxiv.org/abs/1103.0398).
- CONROY, N. J.; RUBIN, V. L.; CHEN, Y. Automatic deception detection: methods for finding fake news. In: *Proceedings of the 78th ASIST Annual Meeting: Information Science with Impact: Research in and for the Community*. USA: American Society for Information Science, 2015. (ASIST '15). ISBN 087715547X.
- CORTES, C.; VAPNIK, V. Support-vector networks. *Mach. Learn.*, Kluwer Academic Publishers, USA, v. 20, n. 3, p. 273–297, set. 1995. ISSN 0885-6125. Disponível em: [⟨https://doi.org/10.1023/A:1022627411411⟩](https://doi.org/10.1023/A:1022627411411).
- COX, D. R. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, [Royal Statistical Society, Oxford University Press], v. 20, n. 2, p. 215–242, 1958. ISSN 00359246. Disponível em: [⟨http://www.jstor.org/stable/2983890⟩](http://www.jstor.org/stable/2983890).
- DEVLIN, J. et al. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. Disponível em: [⟨http://arxiv.org/abs/1810.04805⟩](http://arxiv.org/abs/1810.04805).
- ETHAYARAJH, K. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and GPT-2 embeddings. *CoRR*, abs/1909.00512, 2019. Disponível em: [⟨http://arxiv.org/abs/1909.00512⟩](http://arxiv.org/abs/1909.00512).

- GARCIA, G. L.; AFONSO, L. C. S.; PAPA, J. P. Fake recogna: A new brazilian corpus for fake news detection. In: PINHEIRO, V. et al. (Ed.). *Computational Processing of the Portuguese Language (PROPOR 2022)*. Cham: Springer International Publishing, 2022. (Lecture Notes in Computer Science, v. 13204), p. 57–67. ISBN 978-3-030-98305-5.
- GELFERT, A. Fake news: A definition. *Informal Logic*, v. 38, p. 84–117, 03 2018.
- GIORDANI, L. et al. *fakenewsbr: A Fake News Detection Platform for Brazilian Portuguese*. 2023. Disponível em: <https://arxiv.org/abs/2309.11052>.
- GOLDBERG, Y.; LEVY, O. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014. Disponível em: <http://arxiv.org/abs/1402.3722>.
- HOSMER, D.; LEMESHOW, S.; STURDIVANT, R. *Applied Logistic Regression: Third Edition*. [S.l.]: wiley, 2013. Publisher Copyright: © 2013 by John Wiley & Sons, Inc. All rights reserved. ISBN 9780470582473.
- IKONOMAKIS, E.; KOTSIANTIS, S.; TAMPAKAS, V. Text classification using machine learning techniques. *WSEAS transactions on computers*, v. 4, p. 966–974, 08 2005.
- ITUASSU, A. et al. Mídias digitais, eleições e democracia no brasil: Uma abordagem qualitativa para o estudo de percepções de profissionais de campanha. *Dados*, Instituto de Estudos Sociais e Políticos (IESP) da Universidade do Estado do Rio de Janeiro (UERJ), v. 66, n. 2, p. e20210063, 2023. ISSN 0011-5258. Disponível em: <https://doi.org/10.1590/dados.2023.66.2.294>.
- JALILIFARD, A. et al. Semantic sensitive TF-IDF to determine word relevance in documents. *CoRR*, abs/2001.09896, 2020. Disponível em: <https://arxiv.org/abs/2001.09896>.
- JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. In: \_\_\_\_\_. *Document Retrieval Systems*. GBR: Taylor Graham Publishing, 1988. p. 132–142. ISBN 0947568212.
- JOSHI, A.; BHATTACHARYYA, P.; CARMAN, M. J. Automatic sarcasm detection: A survey. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 50, n. 5, set. 2017. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3124420>.
- JR., E. C. T.; LIM, Z. W.; LING, R. Defining “fake news”. *Digital Journalism*, Routledge, v. 6, n. 2, p. 137–153, 2018. Disponível em: <https://doi.org/10.1080/21670811.2017.1360143>.
- JUSOH, S. A study on nlp applications and ambiguity problems. *Journal of Theoretical and Applied Information Technology*, v. 96, p. 1486–1499, 03 2018.
- KADHIM, A. I. Survey on supervised machine learning techniques for automatic text classification. *Artificial Intelligence Review*, v. 52, n. 1, p. 273–292, Jun 2019. ISSN 1573-7462. Disponível em: <https://doi.org/10.1007/s10462-018-09677-1>.
- KALIYAR, R. K.; GOSWAMI, A.; NARANG, P. Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia Tools and Applications*, v. 80, n. 8, p. 11765–11788, mar. 2021. ISSN 1573-7721. Disponível em: <https://doi.org/10.1007/s11042-020-10183-2>.

- KALIYAR, R. K.; GOSWAMI, A.; NARANG, P. Fakebert: Fake news detection in social media with a BERT-based deep learning approach. *Multimedia Tools and Applications*, v. 80, n. 8, p. 11765–11788, mar. 2021. ISSN 1573-7721. Disponível em: <https://doi.org/10.1007/s11042-020-10183-2>.
- LAZER, D. M. J. et al. The science of fake news. *Science*, v. 359, n. 6380, p. 1094–1096, 2018. Disponível em: <https://www.science.org/doi/abs/10.1126/science.aao2998>.
- LEVY, O.; GOLDBERG, Y. Dependency-based word embeddings. In: TOUTANOVA, K.; WU, H. (Ed.). *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, 2014. p. 302–308. Disponível em: <https://aclanthology.org/P14-2050/>.
- LI, Q. et al. A survey on text classification: From traditional to deep learning. *ACM Trans. Intell. Syst. Technol.*, Association for Computing Machinery, New York, NY, USA, v. 13, n. 2, abr. 2022. ISSN 2157-6904. Disponível em: <https://doi.org/10.1145/3495162>.
- MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. Disponível em: <https://arxiv.org/abs/1301.3781>.
- MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, v. 2013, 01 2013.
- MONTEIRO, R. et al. Contributions to the study of fake news in portuguese: New corpus and automatic detection results: 13th international conference, propor 2018, canela, brazil, september 24–26, 2018, proceedings. In: \_\_\_\_\_. [S.l.]: PROPOR, 2018. p. 324–334. ISBN 978-3-319-99721-6.
- MONTEIRO, R. A. et al. Contributions to the study of fake news in portuguese: New corpus and automatic detection results. In: *Computational Processing of the Portuguese Language*. [S.l.]: Springer International Publishing, 2018. p. 324–334. ISBN 978-3-319-99722-3.
- NING, H.; CHEN, Z. Fusion of the word2vec word embedding model and cluster analysis for the communication of music intangible cultural heritage. *Scientific Reports*, v. 13, n. 1, p. 22717, Dec 2023. ISSN 2045-2322. Disponível em: <https://doi.org/10.1038/s41598-023-49619-8>.
- OPENAI. *OpenAI API Documentation: text-embedding-ada-002*. 2024. <https://platform.openai.com/docs/guides/embeddings>. Acessado em: 14 ago. 2025.
- OPITZ, J. A closer look at classification evaluation metrics and a critical reflection of common evaluation practice. *Transactions of the Association for Computational Linguistics*, v. 12, p. 820–836, 06 2024. ISSN 2307-387X. Disponível em: [https://doi.org/10.1162/tacl.a\\_00675](https://doi.org/10.1162/tacl.a_00675).
- PEREIRA, F. B. et al. Fake news, fact checking, and partisanship: The resilience of rumors in the 2018 brazilian elections. *The Journal of Politics*, v. 84, n. 4, p. 2188–2201, 2022. Disponível em: <https://doi.org/10.1086/719419>.
- POTAMIAS, R. A.; SIOLAS, G.; STAFYLOPATIS, A. . G. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, v. 32, n. 23, p. 17309–17320, Dec 2020. ISSN 1433-3058. Disponível em: <https://doi.org/10.1007/s00521-020-05102-3>.
- RAINIO, O.; TEUHO, J.; KLÉN, R. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, v. 14, n. 1, p. 6086, Mar 2024. ISSN 2045-2322. Disponível em: <https://doi.org/10.1038/s41598-024-56706-x>.

- REN, Y. et al. A new random forest ensemble of intuitionistic fuzzy decision trees. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 31, n. 5, p. 1729–1741, 2022.
- ROBERTSON, S. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, v. 60, p. 503–520, 10 2004.
- ROGERS, A.; KOVALEVA, O.; RUMSHISKY, A. A primer in bertology: What we know about how BERT works. *CoRR*, abs/2002.12327, 2020. Disponível em: [⟨https://arxiv.org/abs/2002.12327⟩](https://arxiv.org/abs/2002.12327).
- ROUMELIOTIS, K. I.; TSELIKAS, N. D.; NASIOPOULOS, D. K. Fake news detection and classification: A comparative study of convolutional neural networks, large language models, and natural language processing models. *Future Internet*, v. 17, n. 1, 2025. ISSN 1999-5903. Disponível em: [⟨https://www.mdpi.com/1999-5903/17/1/28⟩](https://www.mdpi.com/1999-5903/17/1/28).
- SANTOS, L. de F.; SILVA, M. V. da. *The effect of stemming and lemmatization on Portuguese fake news text classification*. 2023. Disponível em: [⟨https://arxiv.org/abs/2310.11344⟩](https://arxiv.org/abs/2310.11344).
- SEBASTIANI, F. Machine learning in automated text categorization. *ACM Computing Surveys*, v. 34, p. 1–47, 04 2001.
- SHAHEEN, Z.; WOHLGENANT, G.; FILTZ, E. Large scale legal text classification using transformer models. *CoRR*, abs/2010.12871, 2020. Disponível em: [⟨https://arxiv.org/abs/2010.12871⟩](https://arxiv.org/abs/2010.12871).
- SIINO, M.; TINNIRELLO, I.; La Cascia, M. Is text preprocessing still worth the time? a comparative survey on the influence of popular preprocessing methods on transformers and traditional classifiers. *Information Systems*, v. 121, p. 102342, 2024. ISSN 0306-4379. Disponível em: [⟨https://www.sciencedirect.com/science/article/pii/S0306437923001783⟩](https://www.sciencedirect.com/science/article/pii/S0306437923001783).
- SILVA, R. M. et al. Towards automatically filtering fake news in portuguese. *EXPERT SYSTEMS WITH APPLICATIONS*, v. 146, MAY 15 2020. ISSN 0957-4174.
- SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. *Portuguese Named Entity Recognition using BERT-CRF*. 2020. Disponível em: [⟨https://arxiv.org/abs/1909.10649⟩](https://arxiv.org/abs/1909.10649).
- TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations: a simple and general method for semi-supervised learning. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. USA: Association for Computational Linguistics, 2010. (ACL '10), p. 384–394.
- VASWANI, A. et al. Attention is all you need. *CoRR*, abs/1706.03762, 2017. Disponível em: [⟨http://arxiv.org/abs/1706.03762⟩](http://arxiv.org/abs/1706.03762).
- WAISBORD, S. Truth is what happens to news: On journalism, fake news, and post-truth. *Journalism Studies*, v. 19, p. 1–13, 07 2018.
- WANG, Y. et al. Eann: Event adversarial neural networks for multi-modal fake news detection. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2018. (KDD '18), p. 849–857. ISBN 9781450355520. Disponível em: [⟨https://doi.org/10.1145/3219819.3219903⟩](https://doi.org/10.1145/3219819.3219903).

WOLF, T. et al. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. Disponível em: [⟨http://arxiv.org/abs/1910.03771⟩](http://arxiv.org/abs/1910.03771).

YATES, D.; ISLAM, M. Z. Fastforest: Increasing random forest processing speed while maintaining accuracy. *Information Sciences*, Elsevier, v. 557, p. 130–152, 2021.

ZHOU, X.; ZAFARANI, R. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys*, Association for Computing Machinery (ACM), v. 53, n. 5, p. 1–40, set. 2020. ISSN 1557-7341. Disponível em: [⟨http://dx.doi.org/10.1145/3395046⟩](http://dx.doi.org/10.1145/3395046).