UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Lucas Guerreiro Pellegrini

Detecção de *Fake News* Usando Redes Neurais Baseadas em *Transformers*

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Lucas Guerreiro Pellegrini

Detecção de *Fake News* Usando Redes Neurais Baseadas em *Transformers*

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Fernanda Maria da Cunha Santos

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

P386 Pellegrini, Lucas Guerreiro, 2002-
2025 Detecção de Fake News Usando Redes Neurais Baseadas em Transformers [recurso eletrônico] / Lucas Guerreiro Pellegrini. - 2025.

Orientadora: Fernanda Maria da Cunha Santos.
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Uberlândia, Graduação em Ciência da Computação.
Modo de acesso: Internet.
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Santos, Fernanda Maria da Cunha,1979-, (Orient.). II. Universidade Federal de Uberlândia. Graduação em Ciência da Computação. III. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

Lucas Guerreiro Pellegrini

Detecção de *Fake News* Usando Redes Neurais Baseadas em *Transformers*

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 19 de Setembro de 2025:

Fernanda Maria da Cunha Santos
Orientador

Elaine Ribeiro de Faria Paiva
Professor

Fabiano Azevedo Dorça
Professor

Uberlândia, Brasil
2025

Resumo

A rápida expansão da internet e das redes sociais tem favorecido a disseminação das chamadas "*Fake News*" (notícias falsas). A dimensão alcançada por esse fenômeno evidencia uma lacuna no combate à desinformação. Este trabalho tem como objetivo empregar modelos de classificação baseados na arquitetura *Transformer* para a detecção de notícias falsas em textos escritos em língua portuguesa. Foram desenvolvidos três modelos distintos: (1) *Encoder-Only*, (2) *Decoder-Only* e (3) *Encoder-Decoder*, todos treinados sobre um conjunto de dados obtido pela união de dois corpora. Além disso, foram implementados classificadores tradicionais, cujos resultados de validação cruzada serviram de comparação com os modelos propostos. Os três modelos *Transformers* apresentaram desempenho superior ao apresentado pelos demais modelos. Em um comparativo realizado apenas entre os modelos baseados em *Transformers*, todos apresentaram desempenho semelhante. O modelo *Encoder-Only* destacou-se por alcançar valores de acurácia e precisão superiores a 95,8%, exigindo um tempo de treinamento significativamente menor. Conclui-se que a aplicação da arquitetura *Transformer* caracteriza uma escolha eficaz de rede neural para a tarefa de classificação de notícias falsas em português e pode contribuir para o desenvolvimento de ferramentas voltadas ao enfrentamento da desinformação.

Palavras-chave: Processamento de Linguagem Natural, Redes *Transformers*, Aprendizado de Máquina, *Fake News*, Inteligência Artificial

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PLN	Processamento de Linguagem Natural
BERT	*Bidirectional Encoder Representations for Transformers*
GPT	*Generative Pretrained Transformer*
RNR	Rede Neural Recorrente
RNC	Rede Neural Convolucional
MLPT-T	Modelos de Linguagem Pré-Treinados baseados em *Transfomer*
RoBERTa	*Robustly Optimized BERT Pretraining Approach*
VM	Votação Majoritária
VM-V	Votação Majoritária com poder de Veto
LSTM	*Long Short-Term Memory*
Adam	*Adaptive Moment Estimation*
AdaGrad	*Adaptive Gradient Algorithm*
CT-BERT	COVID-Twitter-BERT
GRU	*Gated Recurrent Unit*
BiGRU	GRU bidirecional - *Bidirectional Gated Recurrent Unit*
KNN	*K-Nearest Neighbors*
RF	*Ranfom Forest*
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machine*
BOW	*Bag of Words*
ROC	*Receiver Operating Characteristic*
AUC	*Area Under (ROC) Curve*

Sumário

1 Introdução

O processamento de linguagem natural (PLN ou *Natural Language Processing* - NLP em inglês) é uma subárea da inteligência artificial que nasce com o desafio de promover a interação entre humanos e computadores por meio da própria linguagem, naturalmente utilizada por nós, seres humanos (GUPTA, 2014). As técnicas de PLN são empregadas com diversas finalidades, como tradução automática (KHAN; ABID; ABID, 2020), análise de sentimentos (HASAN; MALIHA; ARIFUZZAMAN, 2019) e outras tarefas voltadas para a geração e interpretação de linguagem natural (NWAFOR; ONYENWE, 2021).

Técnicas de PLN também são amplamente empregadas para detecção de informações falsas ou enganosas - popularmente conhecidas como *"fake news"* (OSHIKAWA; QIAN; WANG, 2018). A disseminação desse tipo de conteúdo é uma problemática que acompanha o crescimento do uso da internet no Brasil, que em 2022 atingiu quase 90% dos brasileiros, representando um aumento de mais de 20 pontos percentuais em relação ao ano de 2016 (IBGE, 2023). Uma das principais características da internet está na facilidade e velocidade da comunicação, seja através de veículos informativos ou redes sociais, o que promove maior acessibilidade à informação, mas também facilita a propagação de conteúdos falsos. Ainda, segundo Poynter (2022), mais de 4 a cada 10 brasileiros acreditam encontrar informações falsas diariamente. Diante desse cenário, é de suma importância estudar e desenvolver formas de combater a disseminação de *"fake news"* na internet.

Quando utilizadas como ferramentas de combate à propagação de notícias falsas, as técnicas de PLN necessitam de classificadores que lhes permitam predizer a veracidade das informações analisadas. Para suprir tal necessidade, é comum o emprego de redes neurais (SOUZA, 2023) e outros métodos da área do aprendizado de máquina. Proposta em 2017, a arquitetura de redes neurais denominada *Transformer* teve grande impacto na escolha desses classificadores, apresentando resultados que superavam o desempenho de demais redes em aplicações de diversas técnicas de PLN, além de possibilitar o desenvolvimento de modelos como *BERT* (*Bidirectional Encoder Representations for Transformers*) e *GPT* (*Generative Pre-trained Transformer*) (GILLIOZ et al., 2020). A arquitetura se apoia fundamentalmente em mecanismos de atenção, com o intuito de oferecer maior paralelismo e reduzir a carga de computação sequencial (VASWANI et al., 2017).

Existem, na literatura, trabalhos que fomentam o tema e fornecem resultados que reafirmam sua importância, como Dev et al. (2024), que apresenta um modelo híbrido de aprendizado de máquina para desmascarar *fake news*. Além disso, é possível encontrar trabalhos que abordam o uso de técnicas de PLN utilizando redes *Transformers* como

classificadores, dos quais é possível destacar: Gillioz et al. (2020), que compara uma série de modelos *Transformers* usados em tarefas de PLN com bases de dados estruturadas; e Praseed, Rodrigues e Thilagam (2023), que apresenta um grupo de modelos *Transformers* utilizado para detectar *fake news* na língua Hindi. Ainda, uma fonte de estudo de PLN direcionada para redes *Transformers* pode ser encontrada em Tunstall, Werra e Wolf (2022).

A linguagem de programação *Python* foi escolhida para a implementação da rede neural, junto do auxílio das bibliotecas: *Tensorflow* e *Keras* como interfaces para a modelagem e execução da rede, além de *NumPy* e *Pandas* para tratamento e manuseio dos dados. A escolha de uma arquitetura de redes neurais baseadas em *Transformers* se deu não só por causa da grande notoriedade e amplo emprego da técnica em tarefas de PLN, mas também pelo funcionamento peculiar da arquitetura que, junto dos mecanismos de atenção, permite uma maior compreensão e interpretação da relação entre as palavras do texto bem como seu contexto (GILLIOZ et al., 2020). Foi considerado o uso de uma máquina simples, sem muitas exigências de hardware (conforme especificações técnicas presentes no Capítulo 3), ainda que, na literatura, existam indicações de que Redes *Transformers* podem exigir máquinas mais robustas (GILLIOZ et al., 2020).

Quanto à base de dados, inicialmente havia a intenção de usar a Fake.br Corpus, visto que é uma base completa que já foi utilizada com intenções similares (MONTEIRO et al., 2018). Entretanto, ao decorrer do desenvolvimento desse trabalho, surgiu a necessidade de incrementar o volume de dados utilizados para treinar as redes, o que culminou no uso de uma base de dados resultante da união entre a base previamente mencionada e a FakeRecogna (GARCIA; AFONSO; PAPA, 2022).

1.1 Objetivo

O objetivo deste trabalho é desenvolver e treinar três redes neurais baseadas na arquitetura *Transformer*, capazes de classificar notícias como falsas ou verdadeiras a partir de textos em linguagem natural na língua portuguesa. Em linhas gerais, é possível enumerar os objetivos desta monografia da seguinte forma:

1. Escolher, pré-processar e tratar um conjunto de dados que atenda o domínio do problema abordado;

2. Definir, elaborar e treinar as seguintes redes neurais: (1) *Transformer* (*Encoder-Decoder*), (2) *Encoder-Only*, (3) *Decoder-Only*;

3. Testar e documentar o desempenho das redes, com o intuito de compará-las não só entre si, mas também com outros classificadores distintos;

2 Revisão Bibliográfica

2.1 *Notícias Falsas*

O termo em inglês *"fake news"* (ou notícias falsas) surgiu no século XX com o objetivo de referenciar notícias capciosas produzidas pelos grandes veículos de comunicação em massa (TEIXEIRA, 2019). Entretanto, a partir da popularização massiva do uso da internet, esses veículos comunicativos deixaram de ser os únicos e principais difusores desse tipo de notícia, destacando-se o papel de propagação das redes sociais (DELMAZO; VALENTE, 2018). Segundo Teixeira (2019), estas notícias permeiam todos os campos da internet que dizem respeito à estrutura de comunicação *online*, configurando um fenômeno de desinformação que passou a afetar significativamente o mundo a partir da segunda metade da década de 2010.

Este fenômeno de desinformação é prejudicial à sociedade como um todo, uma vez que notícias falsas confrontam as verdadeiras em uma disputa pelo que é aceito como legítimo aos olhos de quem busca se informar (TEIXEIRA, 2019). Em 2020, no início da pandemia de Covid-19 no Brasil, a quantidade de desinformação que circulava na internet - seja sobre eventuais tratamentos, medicamentos e métodos preventivos ou mesmo a própria doença - alcançou dimensões preocupantes, a ponto de ser considerada uma ameaça à saúde pública (FALCÃO; SOUZA, 2021).

2.2 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é uma área de pesquisa que explora a compreensão e manipulação da linguagem natural por meio de computadores, com a finalidade de desenvolver sistemas capazes de assimilar e manipular a linguagem, tal qual seres humanos (CHOWDHARY, 2020). Surgiu nos anos 1950 como um encontro de interesses entre linguística e inteligência artificial (NADKARNI; OHNO-MACHADO; CHAPMAN, 2011), tendo como uma de suas motivações a tradução automática de russo para inglês durante a Segunda Guerra Mundial (KHYANI; S, 2021).

Com o passar do tempo, suas motivações e finalidades se expandiram, sendo aplicado em outras tarefas além da tradução automática, como análise de sentimentos (HASAN; MALIHA; ARIFUZZAMAN, 2019) e classificação textual (TUNSTALL; WERRA; WOLF, 2022). Assim, tarefas de PLN ficaram categorizadas - de forma genérica - como geradoras ou assimiladoras de linguagem natural (KHYANI; S, 2021). Independente da tarefa em que é empregado, o uso de PLN (principalmente direcionado à classificação

textual) envolve o emprego de técnicas de processamento da entrada, possibilitando sua manipulação e visando fazê-la de forma eficiente.

Técnicas como derivação (do inglês *"stemming"*) e lematização (do inglês *"lemmatization"*) são responsáveis por diminuir o espaço de busca e facilitar a normalização. Enquanto a derivação é responsável por inferir variações da escrita de uma mesma palavra base/raiz, a lematização é o processo de associar palavras similares a um lema (uma outra palavra), com a diferença de que inclui noção de significado à ideia de similaridade entre palavras (KHYANI; S, 2021). Além dessas, são empregadas técnicas responsáveis por transformar palavras em estruturas numéricas. Essa transformação se dá primeiro pela "tokenização"do texto, onde suas palavras são convertidas em estruturas simples (como vetores *one-hot*), denominados *token encodings*, que posteriormente são convertidos em *word embeddings* (TUNSTALL; WERRA; WOLF, 2022), técnicas de representação do texto em estruturas vetoriais que carregam informações relacionadas às palavras e seus significados (GILLIOZ et al., 2020).

É possível mencionar diferentes técnicas de *embedding*, como *Glove*, *Byte Pair Encoding* e *Word2Vec*. Em particular, a *Word2Vec* marca um grande avanço no PLN, representando o estado da arte da representação de palavras em estruturas vetoriais em termos de desempenho, conforme Mikolov et al. (2013). Foi apresentada como uma alternativa à representação de palavras como unidades atômicas, que, mesmo sendo muito limitada, era muito utilizada em PLN (MIKOLOV et al., 2013). Ainda segundo os autores, alguns dos seus maiores diferenciais são a capacidade de interpretar noções de similaridade entre palavras e grupos de palavras, além de ser capaz de lidar com entradas de tamanho elevado com custo computacional inferior ao de técnicas até então utilizadas.

2.3 Modelos de Aprendizado de Máquina

Segundo Pandey, Niwaria e Chourasia (2019), Aprendizado de máquina pode ser descrito como o processo de fazer um sistema "inteligente" a ponto de torná-lo capaz de tomar decisões. Esse processo só é possível por conta do treinamento - parte do processo de aprendizagem - de um modelo, método ou algoritmo matemático que define o funcionamento da tomada de decisões. Em tarefas de classificação, esse processo é denominado aprendizagem supervisionada, em que os dados utilizados na etapa de treinamento são categorizados de acordo com as classes pré-estabelecidas (PANDEY; NIWARIA; CHOURASIA, 2019).

2.3.1 *K-Nearest Neighbor*

O K Vizinhos Mais Próximos (KNN, do inglês *K-Nearest Neighbor*) é um algoritmo vastamente conhecido pela sua simplicidade. Seu funcionamento consiste em calcular a

distância entre um elemento e os demais, denominados vizinhos, que pode ser computada por meio de diferentes métricas, como Distância Euclidiana (Equação 2.1) e Distância de Minkowski (Equação 2.2). A classificação é dada pela classe da maioria dentro da vizinhança computada (TAUNK et al., 2019).

$$\text{Distancia Euclidiana}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

$$\text{Distancia de Minkowski}(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \quad \text{para } p \geq 1 \tag{2.2}$$

Um exemplo de classificação utilizando o algoritmo KNN pode ser observado na Figura 1.

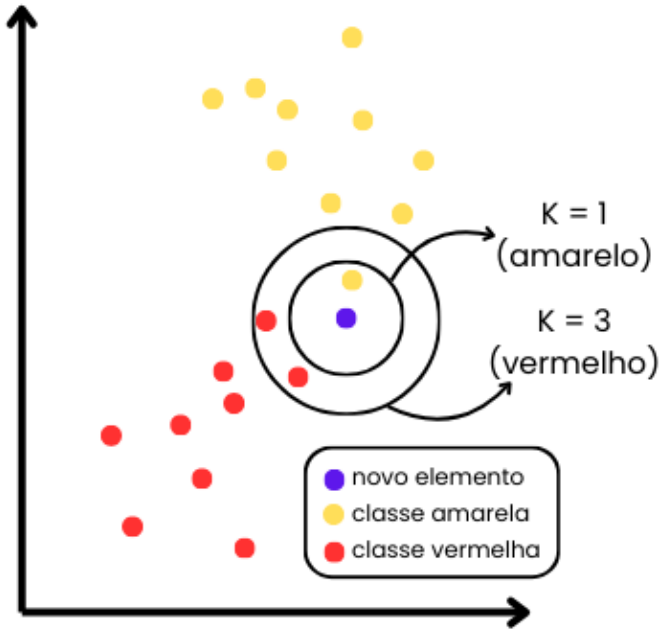

Figura 1 – Exemplo de classificação com KNN. Fonte: Elaboração própria.

2.3.2 *Ranfom Forest*

O Floresta Randômica (RF, do inglês *Random Forest*) é um algoritmo que combina diversas árvores de decisão - um algoritmo de regressão ou classificação conhecido por formar estruturas topológicas em formato de árvores, dos quais nós representam características específicas, enquanto folhas representam classes (PANDEY; NIWARIA; CHOURASIA, 2019). A classificação no RF se dá através de uma combinação dos resultados das suas árvores de decisão, podendo ser computado por meio da moda, para problemas de

classificação, por exemplo (SHAIK; SRINIVASAN, 2019). Um exemplo de classificação utilizando o algoritmo RF pode ser observado na Figura 2.

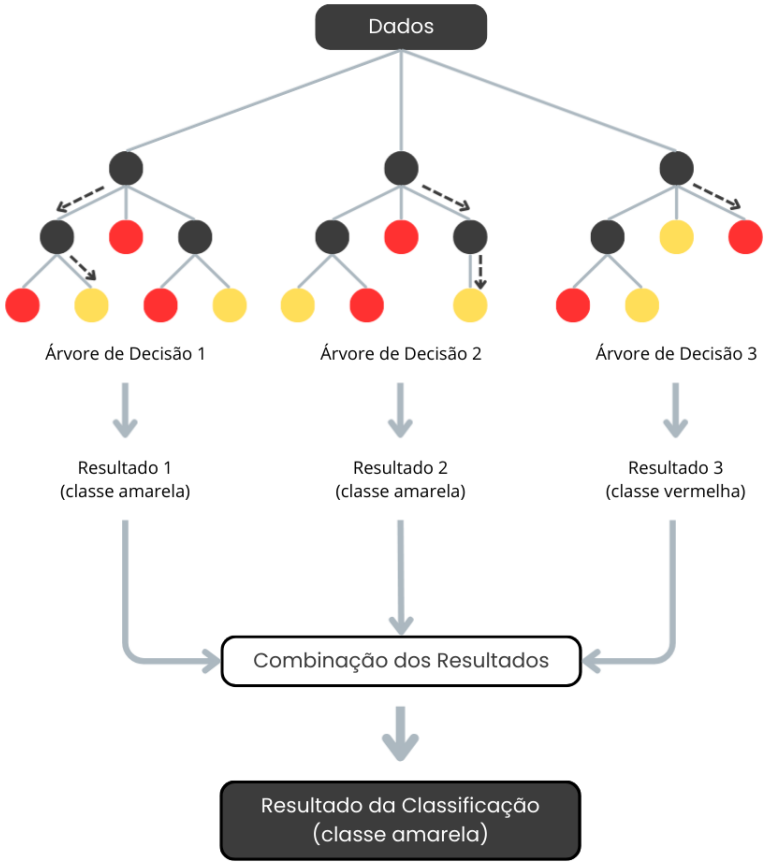

Figura 2 – Exemplo de classificação com RF. Fonte: Elaboração própria.

2.3.3 *Support Vector Machine*

A Máquina de Vetores de Suporte (SVM, do inglês *Support Vector Machine*) é um algoritmo matemático responsável por maximizar uma função objetivo. É empregado na tarefa de classificação para estabelecer um hiperplano que busque maximizar a margem entre a secção e as classes, aumentando, assim, a capacidade do modelo de classificar corretamente novos elementos (NOBLE, 2006). Um exemplo de classificação utilizando o algoritmo RF pode ser observado na Figura 3.

2.3.4 *Stochastic Gradient Descent*

O método de Descida do Gradiente Estocástico (SGD, do inglês *Stochastic Gradient Descent*) é um algoritmo matemático iterativo de minimização, cujo caráter estocástico decorre da escolha aleatória de pequenos subconjuntos do conjunto de dados em cada etapa do processo (KETKAR, 2017). No aprendizado de máquina, é aplicado a um

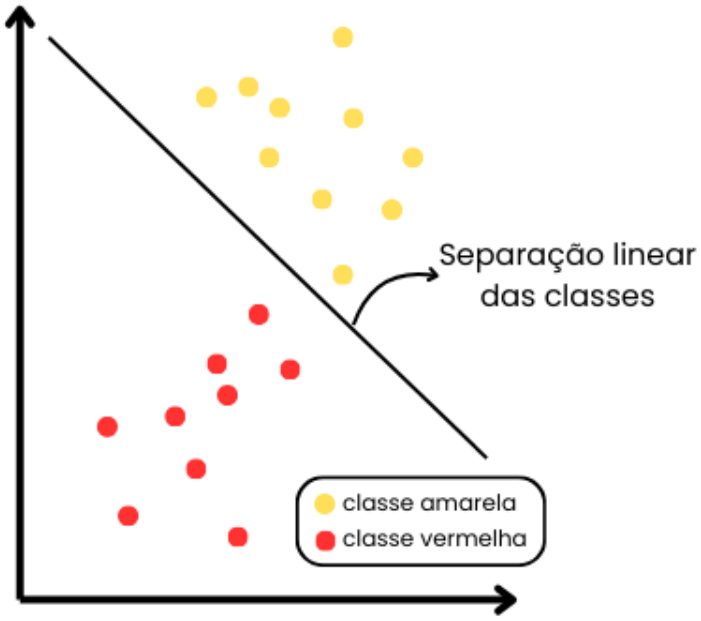

Figura 3 – Exemplo de classificação com SVM. Fonte: Elaboração própria.

classificador linear para ajustar os parâmetros do modelo de forma iterativa, por meio da minimização de alguma função de erro (SCIKIT-LEARN, 2025).

2.4 Modelo *Transformer*

O *Transformer* é o primeiro modelo arquitetural de transdução de sequências[1] baseado integralmente em mecanismos de auto-atenção para a computação das representações das entradas e saídas, em contrapartida ao uso de Redes Neurais Recorrentes (RNRs) ou Convolucionais (RNCs) (VASWANI et al., 2017). Uma das motivações do modelo, que também justifica a tamanha confiança nos mecanismos de atenção, era superar algumas restrições dos modelos concorrentes, em particular a dificuldade de implementar paralelismo (GILLIOZ et al., 2020).

Assim como na maior parte dos modelos desta natureza, a arquitetura do *Transformer* segue um modelo *encoder-decoder*. Tanto o *encoder* quanto o *decoder* são compostos por pilhas de seis camadas idênticas, cada uma possuindo duas subcamadas: uma camada de atenção *multi-head* - que consiste em múltiplas camadas de atenção funcionando em paralelo (GILLIOZ et al., 2020) - e uma rede neural *feed-forward* totalmente conectada.

[1] Um modelo cuja entrada e saída são sequências, potencialmente de tamanhos distintos

Ainda, no *decoder*, há uma terceira subcamada que executa os mecanismos de atenção paralelos sobre a saída da sua respectiva pilha (VASWANI et al., 2017).

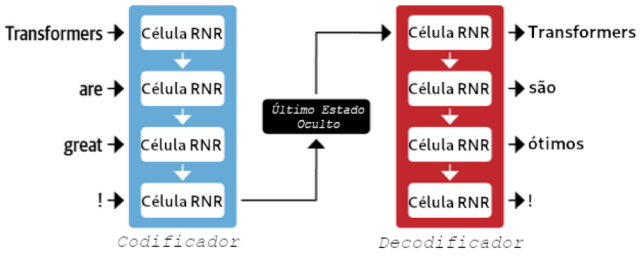

Figura 4 – Exemplo de arquitetura *encoder-decoder* convencional com pares de RNRs. Adaptado de Tunstall, Werra e Wolf (2022)

Em arquiteturas *encoder-decoder* convencionais, o *encoder* é responsável por transformar a sequência de entrada em uma representação numérica, denominada "último estado oculto", que é repassada para o *decoder* que, por sua vez, gera a sequência de saída (TUNSTALL; WERRA; WOLF, 2022), conforme ilustrado na Figura 4. Nesta imagem, é apresentada uma arquitetura *encoder-decoder* convencional simples, cuja tarefa é realizar a tradução automática da entrada em inglês para a saída em português. De acordo com Tunstall, Werra e Wolf (2022), o problema desta abordagem é que o último estado oculto do *encoder* fica responsável por repassar toda a informação significativa da entrada ao *decoder*.

Para contornar essa limitação, é possível fazer com que o *encoder* gere um estado oculto para cada item processado, o que gera uma entrada potencialmente grande para o *decoder*, fazendo com que seja necessária alguma forma de seleção ou filtragem da entrada (TUNSTALL; WERRA; WOLF, 2022). O conceito de atenção surge justamente nesse âmbito: o *decoder* é capaz de atribuir valores de peso (ou atenção) para cada estado oculto do *encoder*, sendo capaz de priorizar estados ocultos específicos (TUNSTALL; WERRA; WOLF, 2022).

Em termos práticos, um mecanismo de atenção é uma função que mapeia uma consulta (*query*) e um conjunto de pares do tipo (chave, valor) para uma saída, que é computada como uma soma ponderada dos valores, de modo que o peso associado a cada valor é calculado por uma função de compatibilidade entre a consulta e a respectiva chave (VASWANI et al., 2017). Especificamente no modelo *Transformer* proposto, a função *softmax* é aplicada a: um conjunto de consultas representadas pela matriz Q, a matriz K referente ao conjunto de chaves e a matriz V referente ao conjunto de valores, onde d_k é a dimensão das chaves e consultas, como observado na Equação 2.3.

$$Atencao(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.3}$$

Figura 5 – Arquitetura *encoder-decoder* do modelo *Transformer* originalmente proposto. Adaptado de Tunstall, Werra e Wolf (2022)

Segundo Tunstall, Werra e Wolf (2022), a ideia original do *Transformer* também envolvia atribuir tanto ao *encoder* quanto ao *decoder* seu próprio mecanismo de auto-atenção, que é um caso particular dos mecanismos de atenção onde cada elemento da sequência de entrada faz o cálculo do peso em relação a todos os outros, conforme ilustra a Figura 5. Um dos motivos para a adoção de tal mecanismo foi que uma camada com auto-atenção é capaz de operar com custo computacional inferior a uma camada recorrente (VASWANI et al., 2017).

Esta redução no custo computacional não foi a única característica responsável pelo expressivo desempenho de redes *Transformers*. Com o sucesso da ideia proposta, surgiram modelos que combinavam a arquitetura *Transformer* com aprendizado não supervisionado, no intuito de remover a necessidade de treinar arquiteturas específicas para determinadas tarefas, denominados Modelos de Linguagem Pré-Treinados baseados em *Transfomer* (MLPT-T) (TUNSTALL; WERRA; WOLF, 2022). De acordo com Tunstall, Werra e Wolf (2022) MLPT-Ts como GPT e BERT representaram um grande avanço para os modelos *Transformers*, principalmente quando usados em tarefas de PLN, sendo dois dos modelos mais proeminentes na área.

2.5 Trabalhos Relacionados

O trabalho de Praseed, Rodrigues e Thilagam (2023) busca desenvolver um grupo de Modelos de Linguagem Pré-Treinados baseados em Transformer (MLPT-Ts) capaz de detectar notícias falsas em textos em Hindi, uma linguagem cuja disponibilidade de recursos é limitada, o que dificulta a execução de tarefas de PLN. Tais dificuldades refletem em trabalhos anteriores realizados, que apresentaram resultados ineficientes ou insatisfatórios ao buscar alternativas como a tradução da entrada para a língua inglesa, ou mesmo o uso de MLPT-Ts de forma individual. A partir disso, os autores propuseram uma arquitetura que combinava *Bidirecional Encoder Representations from Transformers* (BERT), *Robustly Optimized BERT Pretraining Approach* (RoBERTa) e ELECTRA (um MLPT-T cujo funcionamento é bem diferente dos outros dois modelos adotados).

Para que o grupo de modelos funcione como um classificador, foi preciso adotar um mecanismo que recebe as predições de cada MLPT-T e produz uma classificação única.

Com isto em mente, os autores testaram algumas técnicas de seleção, como Perceptron Multi-Camadas e Árvore de decisão, além de técnicas de votação, como Votação Majoritária (VM) e Votação Majoritária com poder de Veto (VM-V). Por fim, foi escolhida a técnica VM-V, onde não só é realizada uma votação pela classificação de maior ocorrência, mas também a predição de um MLPT-T cujo grau de confiança ultrapasse um limiar estipulado θ é capaz de vetar as predições dos outros dois modelos. A Figura 6 ilustra como o VM-V é adotado dentro da arquitetura.

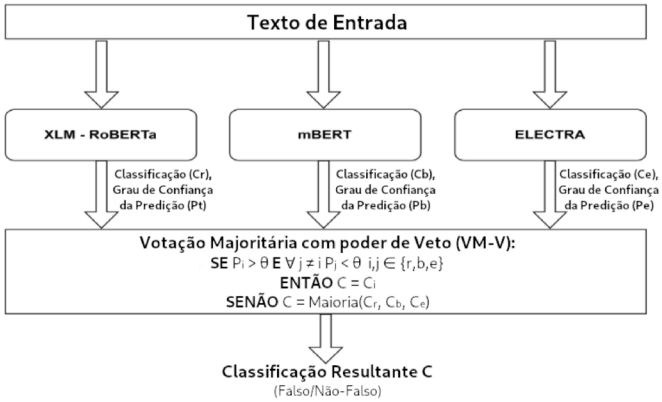

Figura 6 – Arquitetura proposta para o classificador de notícias falsas em Hindi. Adaptado de Praseed, Rodrigues e Thilagam (2023)

Ainda em Praseed, Rodrigues e Thilagam (2023), foi utilizado como entrada do modelo o *dataset* denominado CONSTRAINT2021, desenvolvido em Bhardwaj et al. (2020) para a detecção de conteúdo hostil. Contendo uma coletânea de 8191 publicações *online*, não é especificamente direcionado à problemática tratada, mas já foi utilizado em outros trabalhos envolvendo a detecção de notícias falsas. Antes de alimentar a arquitetura, a entrada é submetida a um pré-processamento que consiste da fragmentação da entrada em sentenças (que posteriormente serão transformados em listas de *tokens*), remoção de palavras pouco-significativas (palavras que não acrescem a sentença de significado ou contexto, também denominadas *stop-words*) e derivação (*stemming*). Em seguida, o *dataset* é dividido em duas partes, para realizar o treinamento e a testagem do modelo. O treinamento, tem como finalidade de ajustar o modelo, uma vez que seus pesos já foram calculados no pré-treinamento. Já a etapa de testagem é realizada com o intuito de avaliar a performance do modelo, utilizando métricas como precisão, Sensibilidade, *F1 score* e Coeficiente de Correlação de Matthews.

Segundo Praseed, Rodrigues e Thilagam (2023), os resultados apresentados pelo modelo proposto superaram, para a maioria das métricas observadas, não só o desempenho dos MLPT-Ts quando utilizados individualmente, mas também outros trabalhos realizados em cima do mesmo *dataset*. Isto levou os autores a conclusão de que o conjunto de MLPT-Ts é capaz de aperfeiçoar tarefas de PLN também em linguagens cujos recursos são encontrados com pouca disponibilidade.

Já em Dev et al. (2024), um modelo híbrido de aprendizado de máquina supervisionado é proposto, unindo uma RNR do tipo *Long Short-Term Memory* (LSTM) bidirecional e uma RNC. Motivados pela crescente difusão de notícias fraudulentas na internet, que acarretou no desenvolvimento de técnicas e ferramentas de identificação de notícias falsas, e pela ascensão de alternativas da área do aprendizado de máquina, que têm se mostrado promissoras para tarefas de PLN, os autores desenvolveram tal modelo com a finalidade de detectar notícias falsas com enfoque em publicações e artigos jornalísticos disponíveis em plataformas *online* como redes sociais.

A entrada utilizada no modelo foi um *dataset* feito pela comunidade de ciência de dados da plataforma *Kaggle*, formado por de cerca de 7800 artigos jornalísticos classificados como reais ou falsos. Para que possa ser devidamente utilizado, é submetido a um pré-processamento, que consiste da remoção de sinais de pontuação e palavras pouco-significativas, além da separação das sentenças do texto em *tokens*, que são utilizados pelo algoritmo pré-treinado *Global Vectors for Word Representation* para formar a representação vetorial efetivamente utilizada pelo modelo. Em seguida, o *dataset* é dividido em treinamento, validação e testagem. Após o treinamento, o modelo é submetido a um refinamento iterativo, que utiliza a partição de validação do *dataset* e emprega técnicas de otimização como *Adaptive Moment Estimation* (Adam) e *Adaptive Gradient Algorithm* (AdaGrad). Já na etapa de testagem, são observadas métricas como precisão, acurácia, sensibilidade e *F1 score* para avaliar o desempenho do modelo.

A decisão da arquitetura proposta em Dev et al. (2024), que culminou em um modelo híbrido proveniente da união de uma LSTM bidirecional e uma RNC, foi resultado de uma série de experimentos realizados pelos autores, onde também foram utilizadas outras técnicas além das citadas, sendo elas uma RNR simples e uma LSTM unidirecional. Foi observado que a combinação proposta superava as outras em todas as métricas observadas. Além disso, os autores realizaram comparações de seu modelo com outras técnicas e trabalhos semelhantes, como RNC, RNC Híbrido, LSTM, AdaBoost e regressão logística. O modelo híbrido proposto apresentou um desempenho superior a todas as comparações realizadas, ainda sob as mesmas métricas.

Motivados pela gravidade e dimensão que a crescente disseminação de notícias falsas atingiu durante a pandemia COVID-19, os autores em Alghamdi, Lin e Luo (2023) exploraram algumas técnicas de aprendizado de máquina baseadas em *Transformer*, com o objetivo de avaliar seus respectivos desempenhos para detecção de notícias falsas relacionadas à COVID-19, tarefa que se tornou um tópico recorrente no âmbito da PLN também durante a pandemia.

Os principais modelos explorados foram BERT e COVID-Twitter-BERT (CT-BERT), um MLPT-T baseado em BERT e fortemente direcionado ao tema tratado no artigo. Neste trabalho, os autores exploraram o desempenho dos MLPT-Ts submetidos

à algumas alterações e aprimoramentos, como o acréscimo de arquiteturas como RNC, LSTM e *Gated Recurrent Unit* (GRU), um tipo de RNR similar à LSTM.

A entrada utilizada nos experimentos foi um *dataset* formado por uma coleção de publicações, comentários e notícias sobre relacionados à temática da doença, classificados como reais ou falsos. Antes de alimentar qualquer um dos modelos, entrada foi submetida a um pré-processamento, que envolve a conversão de seus elementos em *tokens*, além da conversão de elementos como *emoticons* em texto, para que seu valor simbólico também seja considerado no processo de classificação. Por fim, os dados são divididos em três partes, para que sejam utilizados nas etapas de treinamento, validação e testagem.

Em Alghamdi, Lin e Luo (2023), foram realizados diversos experimentos, primeiro usando modelos clássicos de aprendizado de máquina, e em seguida utilizando modelos *Transformers*, onde também foi explorada a presença ou ausência de aprimoramentos e ajuste dos parâmetros durante as etapas de treinamento e validação. Nos primeiros experimentos, os modelos recorrentes foram os que apresentaram melhor desempenho, com destaque para o modelo GRU bidirecional (BiGRU). Já para os experimentos com os MLPT-Ts, a hipótese dos autores de que modelos baseados em CT-BERT resultariam nos melhores resultados foi confirmada, sendo importante ressaltar o conjunto CT-BERT+BiGRU, que superou todos os outros experimentos realizados.

Por fim, os autores concluíram que o auxílio de um modelo de aprendizagem de máquina avançado, como GRU, é capaz de melhorar o desempenho de MLPT-Ts para tarefas de PLN como a realizada no artigo. Ademais, destacaram a importância dos aperfeiçoamentos e ajustes de parâmetros durante a etapa de treinamento, uma vez que foi capaz de melhorar o desempenho de todos os MLPT-Ts experimentados. Em conclusão, o modelo CT-BERT+BiGRU atingiu um desempenho de estado da arte, obtendo um *F1 score* de 98,5%.

3 Desenvolvimento

Neste capítulo, serão apresentadas as etapas de construção das arquiteturas e desenvolvimento dos modelos propostos para detecção de *fake news* com base em Redes *Transformers*, bem como a formação e preparação da base de dados utilizada para as fases subsequentes de treinamento dos modelos. Estas etapas podem ser enumeradas da seguinte forma:

1. Coleta dos dados
2. Pré-processamento e tratamento dos dados
3. Definição dos modelos arquiteturais

Ainda, é importante ressaltar que todo o processo de desenvolvimento foi realizado em um notebook com um processador Intel Core i7-11370H sem o auxílio de GPU.

3.1 Coleta dos dados

Com o intuito de formular um conjunto de dados mais extenso e conciso, foi utilizado duas bases: a Fake.br (MONTEIRO et al., 2018) (disponível em Fake.br Corpus) e a FakeRecogna (GARCIA; AFONSO; PAPA, 2022) (disponível em FakeRecogna).

A primeira base possui 7200 textos formados pelo título e corpo da notícia, já com *stop-words* e acentos removidos. Exatamente 3600 destas notícias são etiquetadas como falsas e as outras 3600 como verdadeiras. Elas estão distribuídas de forma decrescente entre os seguintes temas: (1) Política, (2) TV e celebridades, (3) sociedade e notícias do dia-a-dia, (4) ciência e tecnologia, (5) Economia e (6) Religião. Além disso, as notícias foram extraídas de diferentes fontes, como G1, Folha de São Paulo, Estadão, Diário do Brasil, A Folha do Brasil, The Jornal Brasil e Top Five TV (MONTEIRO et al., 2018).

Já a base FakeRecogna é composta por 5951 notícias etiquetadas como falsas e outras 5951 etiquetadas como verdadeiras, totalizando 11902 textos. O conteúdo disponibilizado também foi submetido a um pré-processamento, onde destaca-se a remoção de *stop-words* e lematização. As notícias estão distribuídas de maneira decrescente entre os temas: (1) Saúde, (2) Política, (3) Entretenimento, (4) Brasil, (5) Ciência e (6) Mundo. Ademais, também foram extraídas de diferentes fontes, das quais é possível mencionar: G1, UOL, Extra, Ministério da Saúde, Boatos.org, Fato ou Fake, E-farsas, UOL Confere e outros (GARCIA; AFONSO; PAPA, 2022).

Para a construção do conjunto de dados utilizado, foi necessário concatenar o título e o corpo das notícias na segunda base de dados. Depois, os conteúdos foram unidos em um único documento csv e as etiquetas da classe foram padronizadas em 0 para notícias categorizadas como falsas e 1 para notícias categorizadas como verdadeiras.

3.2 Pré-processamento e tratamento dos dados

Com o intuito de obter o melhor desempenho dos modelos propostos e reduzir ao máximo qualquer discrepância entre os dados provenientes de duas fontes distintas, o conjunto de dados final foi submetido às técnicas de pré-processamento.

Primeiramente, foi realizada a remoção de acentos, *stop-words*, caracteres especiais e pontuação. Em seguida, foram aplicadas respectivamente as técnicas de lematização e derivação, utilizando as bibliotecas *spaCy* e NLTK *(Natural Language Toolkit)*.

Uma vez pré-processados, os dados são submetidos às estratégias de *word-embedding* e, posteriormente, *padding*. A primeira estratégia é utilizada para fazer com que a sequência textual de palavras seja transformada em uma estrutura vetorial e numérica de *tokens* (também denominada *embedding*) que pode ser interpretada matematicamente pelos modelos de classificação. Em adição, a segunda estratégia é adotada para fazer com que todas as notícias possuam exatamente a mesma quantidade de *tokens*, o que não só auxilia na formatação e padronização da entrada, como também para diminuir o tamanho das notícias e reduzir a carga de processamento nos modelos, por consequência.

Para o *word-embedding*, foi adotado um algoritmo de tokenização *text-to-sequence* simples, fornecido pela biblioteca *Keras*. O funcionamento do algoritmo pode ser dividido em 3 etapas: primeiro a tokenização, onde o texto é transformado em vetores de palavras (*tokens*); em seguida, ocorre a criação do vocabulário, em que são mapeadas todas as palavras distintas da base de dados, atribuindo a cada uma um identificador único; por fim, é realizada a conversão dos vetores de palavras em vetores de identificadores únicos.

A abordagem adotada para a aplicação do *padding* funciona de acordo com as seguintes etapas: primeiro, é feita a escolha de um limiar, isto é, um valor eleito para representar o tamanho de todas as entradas. Em seguida, todos os vetores de *embedding* são reestruturados de acordo com a lógica: vetores cuja quantidade de *tokens* é inferior ao limiar são acrescidos de *tokens* simbólicos no final da sua cadeia, até que possuam o tamanho desejado; já aqueles que extrapolam o limiar são truncados, perdendo os últimos n *tokens* excedentes.

A lógica completa de *embedding* acrescido de *padding* é exemplificada na Figura 7, onde a partir de um pequeno conjunto de textos, é aplicado o algoritmo de tokenização eleito e, em seguida, o *padding*, usando um limiar de 3 *tokens*.

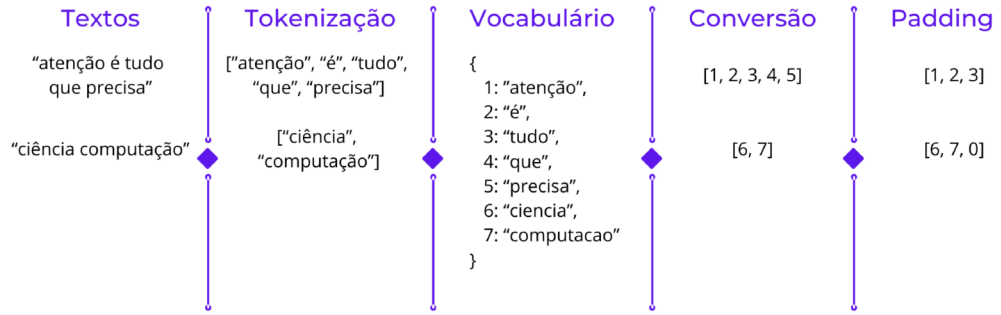

Figura 7 – Imagem ilustrativa do processo de *embedding* e *padding*. Fonte: Elaboração própria.

O limiar definido foi de 500 palavras por notícia. Essa escolha foi fortemente influenciada pelas limitações de hardware, visto que, à medida que o valor do limiar aumenta, crescia também a exigência pelas arquiteturas *Transformers* de maiores dimensões.

Somado a isto, nota-se que na base de dados há uma discrepância considerável entre as dimensões das notícias verdadeiras e falsas, conforme ilustrado na Figura 8. Com o intuito de reduzir essa discrepância, foi avaliada a remoção de notícias excessivamente longas[1]. No entanto, essa estratégia atingiu notícias majoritariamente verdadeiras, o que causou um desbalanceamento entre as classes da base de dados. Para compensar esse desequilíbrio, realizou-se a remoção das maiores notícias falsas, o que resultou em uma base de dados significativamente menor. Diante desse resultado, optou-se por não adotar essas abordagens.

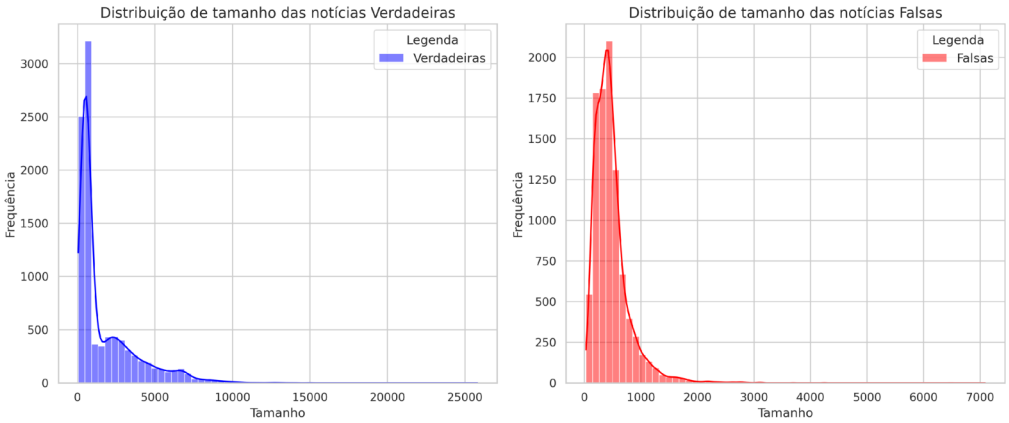

Figura 8 – Comparativo de dimensões entre notícias verdadeiras e falsas. Fonte: Elaboração própria.

[1] destaca-se que, nessa estratégia, foram experimentados alguns limites superiores distintos para o tamanho máximo das notícias

3.3 Definição dos modelos arquiteturais

Nesta sessão, são apresentados os três modelos arquiteturais desenvolvidos com base no modelo *Transformer* original. Os modelos foram projetados com a finalidade de observar a arquitetura não só como um todo, mas também suas componentes isoladas, permitindo uma compreensão mais aprofundada de sua estrutura e funcionamento. As arquiteturas modeladas neste trabalho incluem: uma arquitetura *Transformer* completa (*encoder-decoder*), uma versão *encoder-only* e uma versão *decoder-only*.

Os três modelos compartilham dos mesmos valores para dimensão: as dimensões de *embedding* e da rede neural *feedforward* são de 64, enquanto que a quantidade de cabeças codificadoras e/ou decodificadoras é de duas cabeças. Além desses valores, também são comuns aos modelos diversas classes da biblioteca *tensorflow.keras*, como *MultiHeadAttention*, *Sequential* e *Dense*[2], *Embedding*, *LayerNormalization* e *Dropout*.

Ainda, é importante ressaltar que estes valores foram reduzidos quando comparados ao modelo original em detrimento das limitações de *hardware* apresentadas, ou seja, as respectivas dimensões do modelo original são 512 (*embedding*), 2048 (rede neural) e 8 (cabeças (de)codificadoras), cujo treinamento foi realizado em 8 GPUs do modelo P100.

A primeira arquitetura desenvolvida é a *encoder-only*, constituída apenas da parte codificadora da arquitetura original. É comumente empregada em tarefas onde o foco é a compreensão da entrada, sem a necessidade da geração de uma sequência de saída, como a classificação (que é o foco deste trabalho), conforme pode ser observado em Sun et al. (2019) e Garrido-Merchan, Gozalo-Brizuela e Gonzalez-Carvajal (2023).

A estrutura e sequência de funcionamento do modelo *encoder-only* estão descritas na Figura 9. Primeiro, a sequência de entrada é submetida a um processo de *Token And Positional Embedding*, responsável por transformá-la em uma estrutura matricial denominada *token embeddings* e somá-la com informações relativas à posição dos *tokens* geradas dinamicamente. Em seguida, as cabeças codificadoras fazem o processamento da entrada, passando-a primeiro pela camada de atenção *multi-head* e, posteriormente, pela rede neural *feedforward*. Ambas possuem camadas de *dropout* com probabilidade de $0,5$ e normalização subsequentes. Por fim, o resultado das cabeças codificadoras é submetido a um *Pooling* unidimensional, seguido por uma camada de *dropout* também com probabilidade de $0,5$, para que, finalmente, passe por uma função sigmoide, resultando na saída gerada.

A arquitetura do segundo modelo é a *decoder-only*, constituída apenas da parte decodificadora da arquitetura original. Uma das aplicações da arquitetura *Transformer* de maior notoriedade da atualidade é o GPT da *OpenAI*, que foi construído em cima de uma arquitetura baseada apenas na parte decodificadora da arquitetura original. Em Radford

[2] As classes *Sequential* e *Dense* são utilizadas na construção da Rede Neural *feedforward*

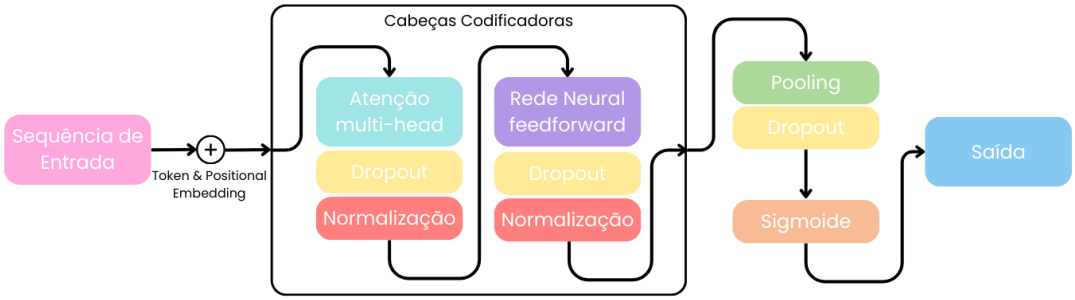

Figura 9 – Arquitetura do modelo *encoder-only*. Fonte: Elaboração própria.

et al. (2018), são realizados vários experimentos com tarefas de PLN, como Reconhecimento de Enlace Textual, Resposta Automática a Perguntas, Similaridade Semântica e, até mesmo, Classificação. O modelo desenvolvido neste trabalho se aproxima muito ao modelo *encoder-only*, previamente apresentado, com a adição de uma nova etapa no *pipeline* das cabeças decodificadoras: há, no início do processamento, uma camada de atenção *multi-head* com máscara (do inglês *mask*). O uso da máscara tem como principal objetivo fazer com que o mecanismo de atenção considere apenas *tokens* em processamento ou já processados. A estrutura do modelo pode também ser observada na Figura 10.

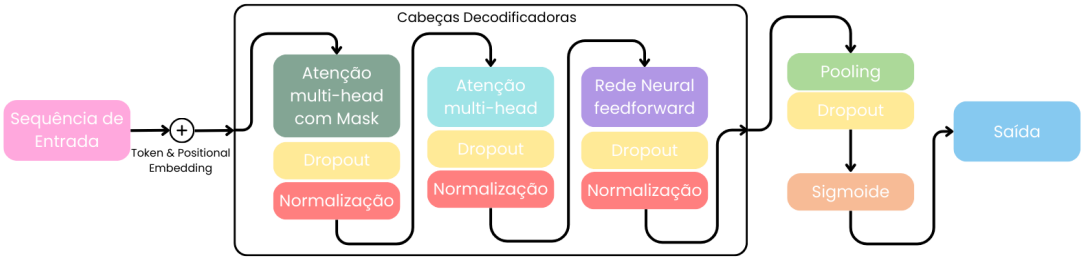

Figura 10 – Arquitetura do modelo *decoder-only*. Fonte: Elaboração própria.

Por fim, a terceira e última arquitetura desenvolvida é um modelo completo da rede *Transformer*. Conforme ilustrado na Figura 11, essa arquitetura combina os módulos *encoder* (codificador) e *decoder* (decodificador) em uma estrutura única, onde a camada codificadora é responsável por processar a sequência de entrada e gerar uma representação latente que é então utilizada pelos mecanismos de atenção da camada decodificadora para produzir a saída desejada. É importante salientar que, embora a saída seja gerada pelo decodificador, a sequência de entrada é fornecida a ambas as camadas. Essa abordagem preserva a essência da arquitetura original proposta por Vaswani et al. (2017), enquanto se adapta ao contexto do problema de classificação abordado neste trabalho.

Inicialmente, os resultados obtidos da validação cruzada pelos três modelos desenvolvidos foram comparados com os de outros classificadores. Em seguida, realizou-se uma comparação exclusiva entre esses três modelos, abrangendo as etapas de treinamento,

validação e teste. Os detalhes de cada experimento encontram-se descritos no capítulo seguinte (Capítulo 4), onde também são apresentados e discutidos os resultados obtidos.

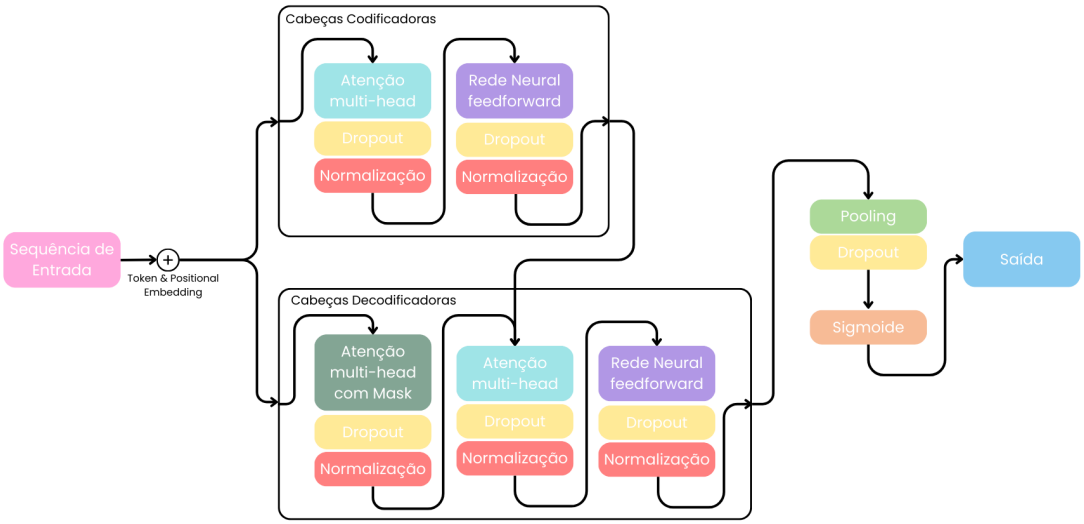

Figura 11 – Arquitetura *Transformer*. Fonte: Elaboração própria.

4 Resultados

Neste capítulo, serão apresentados os resultados da execução e a avaliação dos três modelos de classificação usando a arquitetura da rede neural *Transformer* (*encoder-decoder*, *encoder-only* e *decoder-only*), além de uma comparação com os resultados obtidos por outros classificadores caracterizados por algoritmos de Aprendizado de Máquinas.

4.1 Comparação com outros classificadores

Com o objetivo de avaliar o desempenho e a eficiência dos resultados gerados pelos modelos propostos neste trabalho, foi realizada uma comparação com demais classificadores relatados na literatura. São eles: *K-Nearest Neighbors* (KNN), *Random Forest* (RF), *Stochastic Gradient Descent* (SGD), *Support Vector Machine* (SVM). Para todos os classificadores, fez-se um experimento na base de dados aplicando validação cruzada com 10 dobras (*k-folds*), e calculadas as seguintes métricas validação: Acurácia (Equação 4.1), Precisão (Equação 4.2), Sensibilidade (Equação 4.3) e *F1-Score* (Equação 4.4), além do tempo gasto durante a execução do algoritmo, medido com auxílio da biblioteca *time*.

É importante destacar que, para os demais classificadores, foram utilizados parâmetros majoritariamente padrão definidos pela biblioteca *Scikit-learn*. Isto é, KNN com $K = 1$ e distância de Minkowski com $p = 2$ (equivalente à distância euclidiana); RF com $n\_estimators = 100$, critério *gini* e sem limitação de max_depth; SGD com função de perda *hinge*; SVM com kernel *RBF* e $C = 1$.

$$\text{Acurácia} = \frac{VP + VN}{VP + FP + VN + FN} \tag{4.1}$$

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{4.2}$$

$$\text{Sensibilidade} = \frac{VP}{VP + FN} \tag{4.3}$$

$$\textit{F1-Score} = \frac{2 \times \text{Precisão} \times \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}} \tag{4.4}$$

$$\text{Especificidade} = \frac{VN}{VN + FP} \tag{4.5}$$

Sejam:

VP: Verdadeiros Positivos

FP: Falsos Positivos

VN: Verdadeiros Negativos

FN: Falsos Negativos

Conforme ilustrado pela Tabela 1, fica claro que as três arquiteturas baseadas em *Transformers* apresentaram um desempenho significativamente superior em todas as métricas de validação, exigindo, por outro lado, um tempo de execução consideravelmente maior.

Apesar dos modelos das arquiteturas *Transformers* terem atingido valores próximos, é válido ressaltar a ótima performance da arquitetura *Encoder-Only*, principalmente destacando o seu tempo de processamento alcançado. Ao comparar a arquitetura *Encoder-Only* com a *Decoder-only*, apresentou métricas, em média, 0,48% superiores, em um tempo de processamento cerca de 1,76 vezes inferior. Embora as métricas do modelo *Encoder-Only* tenham sido, em média, 3,73% inferiores à da arquitetura *Encoder-Decoder*, o tempo gasto foi 2,66 vezes inferior.

	Tempo	**Acurácia**	**Precisão**	**Sensibilidade**	**F1-Score**
KNN	4,491s	61,34%	61,37%	61,34%	61,31 %
RF	91,718s	73,00%	73,51%	73,00%	72,85%
SGD	24,698s	62,33%	62,35%	62,33%	62,32%
SVM	590,170s	68,72%	71,04%	68,72%	67,84%
Encoder-only	657,169s	90,44%	90,57%	90,44%	90,43%
Decoder-only	1159.297s	89,97%	90,04%	89,97%	89,97%
Transformer	1750,322s	94,20%	94,20%	94,20%	94,20%

Tabela 1 – Tabela de resultados da validação cruzada

4.2 Comparativo entre as arquiteturas da rede *Transformers*

No intuito de estabelecer um comparativo específico entre as arquiteturas *Transformer* (*encoder-decoder*), *Encoder-only* e *Decoder-only*, foi realizado um experimento de treinamento e testagem. Primeiro, os dados foram divididos em segmentos aleatórios de treinamento, validação e teste, com respectivamente 80%, 10% e 10% de toda a base de dados. Em seguida, foi realizado um processo iterativo de treinamento e validação com máximo de dez épocas com aplicação da técnica de parada precoce (do inglês *early-stopping*), que interrompe o processo no caso em que o erro quadrático médio da validação, entre

uma época e outra, é inferior a 0,0005. E, por fim, os dados de teste são utilizados para obter as métricas de validação utilizadas para a comparação das arquiteturas.

Além da métricas previamente mencionadas (Acurácia, Precisão, Sensibilidade e *F1-Score*, bem como o tempo gasto no treinamento, também medido com auxílio da biblioteca *time*), foram observadas a curva ROC (do inglês *Receiver Operating Characteristic*) e a área sob a mesma curva.

A partir da análise da curva ROC, uma representação cartesiana da relação entre Sensibilidade (Equação 4.3) e Especificidade (Equação 4.5) de classificadores binários, é possível inferir um bom limiar de decisão, bem como avaliar a capacidade do modelo de distinguir numericamente entre positivos e negativos (FLACH, 2010). Esta distinção é feita a partir da área calculada sob a curva ROC[1], um valor dentro do intervalo $[0, 1]$, que, quando igual a 1 indica que o classificador prediz todos os positivos com valores superiores do que qualquer um dos negativos; quando igual a 0 indica que o classificador prediz todos os negativos com valores superiores aos positivos (FLACH, 2010). A partir das Figuras 12, 14 e 16 foi definido um limiar de decisão comum de $0,85$, fazendo com que as predições do modelo sejam reescritas de acordo com a função degrau definida na Equação 4.6

$$y'_{pred} = \begin{cases} 1, & \text{se } y_{pred} \geq 0,85 \\ 0, & \text{se } y_{pred} < 0,85 \end{cases} \tag{4.6}$$

É possível inferir, a partir da Tabela 2, que os modelos performaram de maneira muito semelhante, uma vez que a taxa de variação média[2] das métricas disponíveis na tabela é de 1,425% entre as arquiteturas *Encoder-only* e *Decoder-only*, 0,7925% entre as arquiteturas *Encoder-only* e *Transformer* e 0,6325% entre as arquiteturas *Transformer* e *Decoder-only*. Em adição a estes dados, esta semelhança também é ilustrada pelos mapas de calor apresentados nas Figuras 13, 15 e 17, bem como pelos valores de área sob a curva ROC, encontrados nas Figuras 12, 14 e 16, respectivamente de 0,99, 0,989 e 0,987 para as arquiteturas *Encoder-only*, *Decoder-only* e *Transformer*.

Encoder-only		**Decoder-only**		**Transformer**	
Tempo	212,969s	**Tempo**	401,380s	**Tempo**	603,177s
Acurácia	95,81%	**Acurácia**	94,34%	**Acurácia**	95,02%
Precisão	95,82%	**Precisão**	94,54%	**Precisão**	95,02%
Sensibilidade	95,81%	**Sensibilidade**	94,34%	**Sensibilidade**	95,02%
F1-Score	95,81%	**F1-Score**	94,33%	**F1-Score**	95,02%

Tabela 2 – Tabela de resultados do treinamento

[1] Area Under (ROC) Curve - AUC
[2] A média da diferença entre as quatro métricas de desempenho observadas

A diferença mais significativa se dá, novamente, no tempo de treinamento de cada arquitetura, que coloca a *Encoder-only* em uma posição de destaque. Não só acompanhou, mas superou as outras arquiteturas em todas as métricas observadas, como também o fez em um tempo significativamente inferior. Foi cerca de 1,88 vezes mais rápida que a arquitetura *Decoder-only* e 2,85 vezes mais rápida que a *Transformer*.

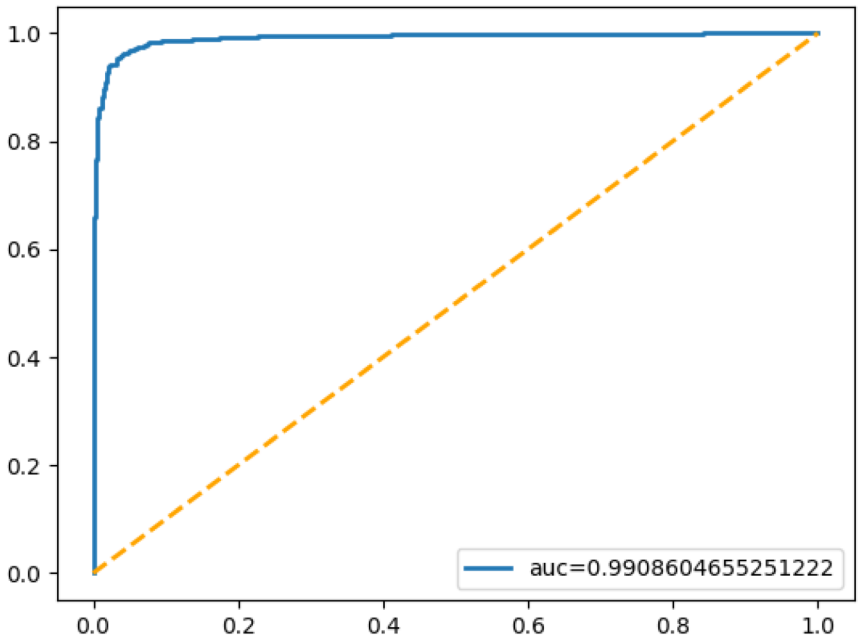

Figura 12 – Curva ROC do modelo *Encoder-only*. Fonte: Elaboração própria

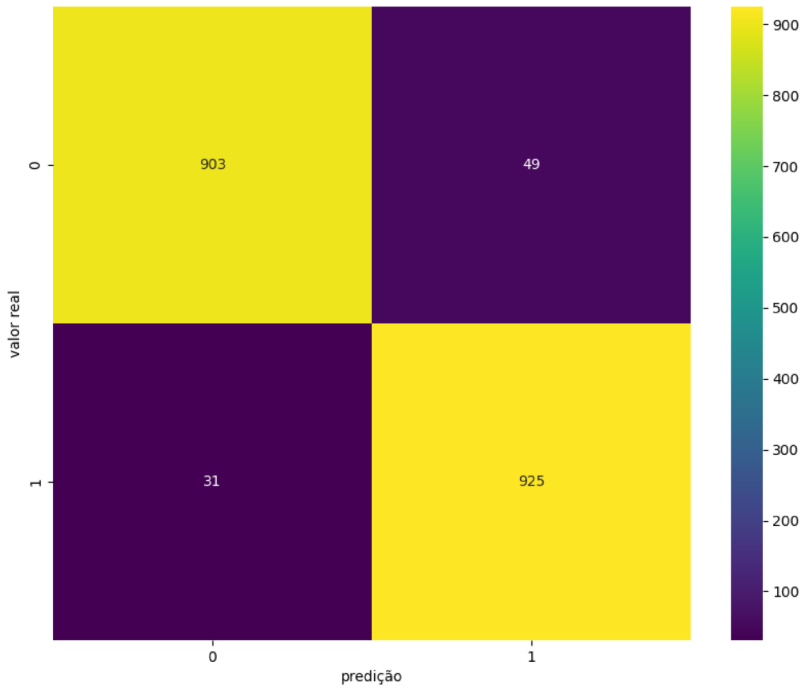

Figura 13 – Mapa de calor do modelo *Encoder-only*. Fonte: Elaboração própria

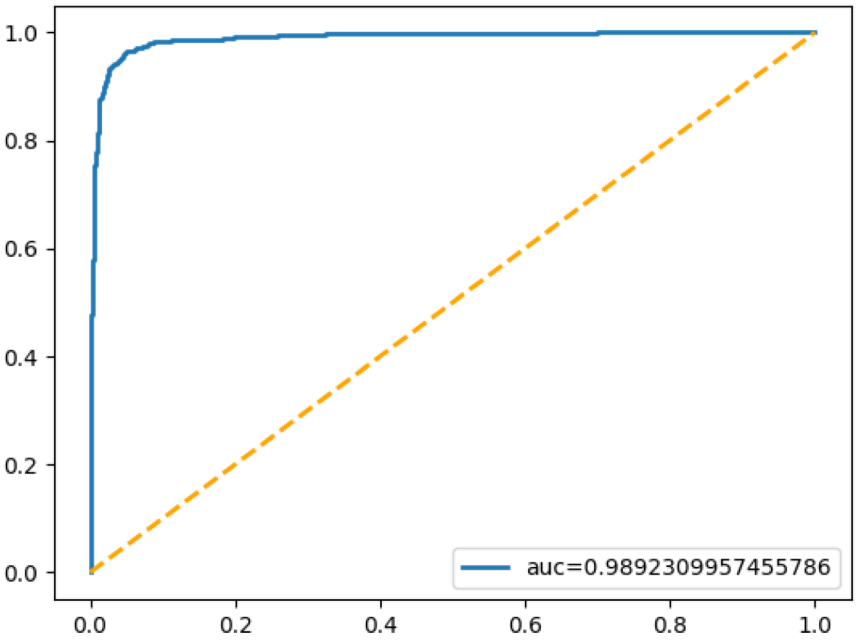

Figura 14 – Curva ROC do modelo *decoder-only*. Fonte: Elaboração própria

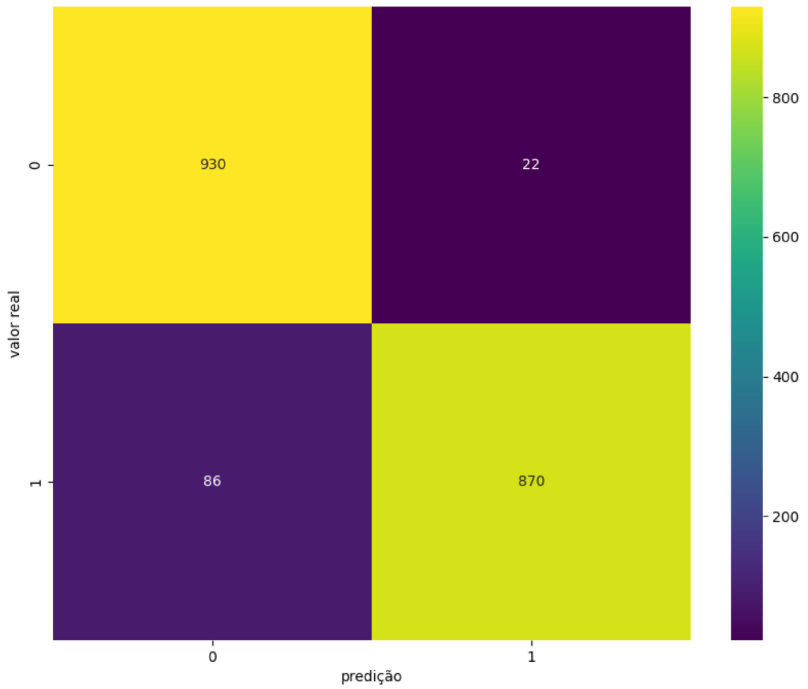

Figura 15 – Mapa de calor do modelo *Decoder-only*. Fonte: Elaboração própria

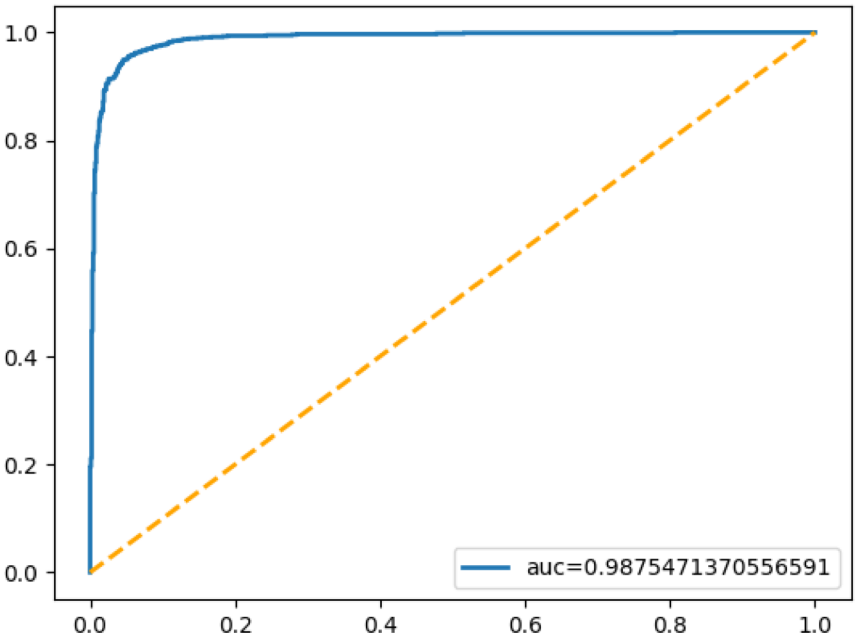

Figura 16 – Curva ROC do modelo *Transformer*. Fonte: Elaboração própria

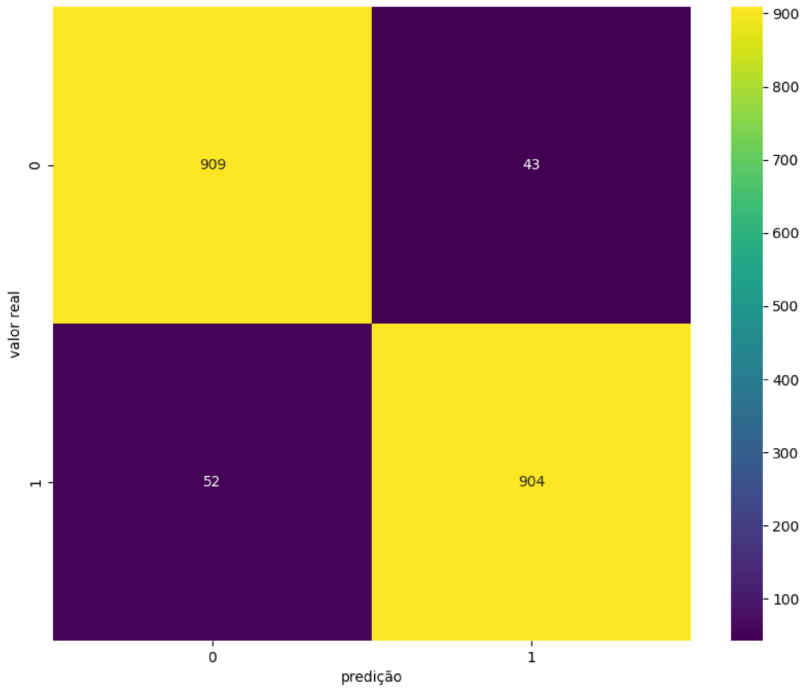

Figura 17 – Mapa de calor do modelo *Transformer*. Fonte: Elaboração própria

5 Conclusão

Neste trabalho foi enfrentado o desafio de adaptar a arquitetura *Transformer*, originalmente proposta com foco em tarefas de tradução automática, ao problema de classificação de notícias falsas na Língua Portuguesa. Para isso, foram desenvolvidos três modelos distintos de classificadores: um baseado exclusivamente no Codificador (*encoder*), outro no Decodificador (*decoder*) e um terceiro utilizando a arquitetura completa (*encoder-decoder*), preservando os princípios do artigo *"Attention is All You Need"* Vaswani et al. (2017).

A solução atendeu aos três objetivos principais listados no Capítulo 1: (1) A escolha, pré-processamento e tratamento de um conjunto de dados, que resultou na união de duas corpora (Fake.br e FakeRecogna), as quais foram submetidas a processos de remoção de *stop-words*, lematização, derivação e, posteriormente, tokenização e padding; (2) a elaboração e treinamento de três redes neurais a partir da arquitetura *Transformer*, que resultou nos classificadores *encoder-only*, *decoder-only* e *Transformer* (*encoder-decoder*) e (3) a avaliação sistemática do desempenho de classificadores, que demonstrou uma clara superioridade das três arquiteturas baseadas em *Transformers* quando comparadas a outros algoritmos experimentados neste trabalho.

A superioridade das arquiteturas propostas pôde ser observada em todas as métricas avaliadas (acurácia, precisão, sensibilidade e *F1-Score*) frente aos demais classificadores (KNN, RF, SGD e SVM). Tomando a comparação da validação cruzada com 10 dobras (Tabela 1), enquanto o desempenho desses demais modelos apresentava valores para as métricas entre 60,7% e 73,51% (com destaque para o *Random Forest*, que apresentou a melhor performance) as variações das redes *Transformer* alcançaram resultados significativamente superiores, permeando 89,97% e 94,20%. Já no treinamento, realizado apenas com os três modelos *Transformers* (Tabela 2), os resultados foram bem mais equilibrados, com métricas concentradas entre 94,33% e 95,82%. Esta pequena variação é ainda menos significativa uma vez considerada a escolha de um mesmo limiar da função degrau, adotada para simplificar e padronizar o experimento, em oposição à busca por valores ótimos de limiar para cada um dos três modelos.

Por outro lado, a diferença mais relevante entre os modelos *Transformers* residiu no tempo de processamento: a arquitetura *Encoder-only* demandou quase um terço do tempo de treinamento gasto pela arquitetura *Transformer* completa (212,969s contra 603,177s) sem nenhum comprometimento notável do desempenho (*F1-Score* de 95,81% contra 95,02%). Esse resultado sugere que, para tarefas de classificação binária em PLN, arquiteturas baseadas apenas na camada Codificadora da rede *Transformer* podem surgir

como um ponto de equilíbrio entre eficiência computacional e desempenho, uma vez que os mesmos mecanismos de atenção (atenção *multi-head*) são oferecidos em uma rede com topologia significativamente reduzida.

Por fim, os resultados apresentados neste trabalho foram satisfatórios, ainda que limitados por restrições práticas de poder computacional e capacidade de hardware, fatores esses que coagiram a adoção de redes de dimensões reduzidas, bem como estruturas de dados mais simples. Considerando tanto os resultados obtidos, quanto as limitações e dificuldades enfrentadas, são propostas as seguintes direções para avançar nessa pesquisa: (1) A expansão do escopo para tarefas de Classificação com várias classes; (2) O uso de estruturas de dados e *embeddings* mais complexos como *GloVe* e *Word2Vec*, para aperfeiçoar a extração de significado dos textos; (3) A comparação do desempenho das arquiteturas *Encoder-Only*, *Decoder-Only* e *Transformer* em tarefas de PLN que diferem da Classificação principalmente em termos de estrutura, como a Tradução Automática, que pressupõe a geração de uma sequência de saída. Esses trabalhos futuros permitiriam elucidar como cada componente da arquitetura se comporta em diferentes cenários e problemas, tal qual encontrar soluções arquiteturais mais simples, rápidas e até mais performáticas para tarefas que, empiricamente, podem ser realizadas por Redes *Transformers*.

Referências

ALGHAMDI, J.; LIN, Y.; LUO, S. Towards covid-19 fake news detection using transformer-based models. **Knowledge-Based Systems**, v. 274, p. 110642, 2023. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705123003921>. Citado 2 vezes nas páginas 18 e 19.

BHARDWAJ, M.; AKHTAR, M. S.; EKBAL, A.; DAS, A.; CHAKRABORTY, T. Hostility detection dataset in hindi. **CoRR**, abs/2011.03588, 2020. Disponível em: <https://arxiv.org/abs/2011.03588>. Citado na página 17.

CHOWDHARY, K. R. Natural language processing. In: _____. **Fundamentals of Artificial Intelligence**. New Delhi: Springer India, 2020. p. 603–649. ISBN 978-81-322-3972-7. Disponível em: <https://doi.org/10.1007/978-81-322-3972-7_19>. Citado na página 10.

DELMAZO, C.; VALENTE, J. C. Fake news nas redes sociais online: propagação e reações à desinformação em busca de cliques. **Media Jornalismo**, scielopt, v. 18, p. 155 – 169, 04 2018. ISSN 2183-5462. Disponível em: <http://scielo.pt/scielo.php?script=sci_arttext&pid=S2183-54622018000100012&nrm=iso>. Citado na página 10.

DEV, D. G.; BHATNAGAR, V.; BHATI, B. S.; GUPTA, M.; NANTHAAMORNPHONG, A. Lstmcnn: A hybrid machine learning model to unmask fake news. **Heliyon**, v. 10, n. 3, p. e25244, 2024. ISSN 2405-8440. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405844024012751>. Citado 2 vezes nas páginas 8 e 18.

FALCÃO, P.; SOUZA, A. B. d. **Pandemia de desinformação: As fake news no contexto da covid-19 no Brasil**. Fundação Oswaldo Cruz. Instituto de Comunicação e Informação Científica e Tecnológica em Saúde., 2021. Disponível em: <https://www.arca.fiocruz.br/handle/icict/47085>. Citado na página 10.

FLACH, P. A. Roc analysis. In: _____. **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 869–872. ISBN 978-0-387-30164-8. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_733>. Citado na página 28.

GARCIA, G. L.; AFONSO, L. C. S.; PAPA, J. P. Fakerecogna: A new brazilian corpus for fake news detection. In: PINHEIRO, V.; GAMALLO, P.; AMARO, R.; SCARTON, C.; BATISTA, F.; SILVA, D.; MAGRO, C.; PINTO, H. (Ed.). **Computational Processing of the Portuguese Language**. Cham: Springer International Publishing, 2022. p. 57–67. ISBN 978-3-030-98305-5. Citado 2 vezes nas páginas 9 e 20.

GARRIDO-MERCHAN, E. C.; GOZALO-BRIZUELA, R.; GONZALEZ-CARVAJAL, S. Comparing bert against traditional machine learning models in text classification. **Journal of Computational and Cognitive Engineering**, BON VIEW PUBLISHING PTE, v. 2, n. 4, p. 352–356, abr. 2023. ISSN 2810-9570. Disponível em: <http://dx.doi.org/10.47852/bonviewJCCE3202838>. Citado na página 23.

GILLIOZ, A.; CASAS, J.; MUGELLINI, E.; KHALED, O. A. Overview of the transformer-based models for nlp tasks. In: **2020 15th Conference on Computer**

Science and Information Systems (FedCSIS). [s.n.], 2020. p. 179–183. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9222960>. Citado 4 vezes nas páginas 8, 9, 11 e 14.

GUPTA, V. A survey of natural language processing techniques. **International Journal of Computer Science & Engineering Technology**, Citeseer, v. 5, n. 1, p. 14–16, 2014. Disponível em: <https://arxiv.org/abs/2212.05773>. Citado na página 8.

HASAN, M. R.; MALIHA, M.; ARIFUZZAMAN, M. Sentiment analysis with nlp on twitter data. In: IEEE. **2019 international conference on computer, communication, chemical, materials and electronic engineering (IC4ME2)**. [S.l.], 2019. p. 1–4. Citado 2 vezes nas páginas 8 e 10.

IBGE. **Pesquisa Nacional por Amostra de Domicílios Contínua**. IBGE, 2023. Disponível em: <https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=2102040>. Citado na página 8.

KETKAR, N. Stochastic gradient descent. In: _____. **Deep Learning with Python: A Hands-on Introduction**. Berkeley, CA: Apress, 2017. p. 113–132. ISBN 978-1-4842-2766-4. Disponível em: <https://doi.org/10.1007/978-1-4842-2766-4_8>. Citado na página 13.

KHAN, N. S.; ABID, A.; ABID, K. A novel natural language processing (nlp)–based machine translation model for english to pakistan sign language translation. **Cognitive Computation**, Springer, v. 12, p. 748–765, 2020. Citado na página 8.

KHYANI, D.; S, S. B. An interpretation of lemmatization and stemming in natural language processing. **Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology**, v. 22, p. 350–357, 01 2021. Disponível em: <https://www.researchgate.net/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing>. Citado 2 vezes nas páginas 10 e 11.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. **Efficient Estimation of Word Representations in Vector Space**. 2013. Disponível em: <https://arxiv.org/abs/1301.3781>. Citado na página 11.

MONTEIRO, R. A.; SANTOS, R. L. S.; PARDO, T. A. S.; ALMEIDA, T. A. de; RUIZ, E. E. S.; VALE, O. A. 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. Citado 2 vezes nas páginas 9 e 20.

NADKARNI, P. M.; OHNO-MACHADO, L.; CHAPMAN, W. W. Natural language processing: an introduction. **Journal of the American Medical Informatics Association**, v. 18, n. 5, p. 544–551, 09 2011. ISSN 1067-5027. Disponível em: <https://doi.org/10.1136/amiajnl-2011-000464>. Citado na página 10.

NOBLE, W. S. What is a support vector machine? **Nature biotechnology**, Nature Publishing Group UK London, v. 24, n. 12, p. 1565–1567, 2006. Citado na página 13.

NWAFOR, C. A.; ONYENWE, I. E. An automated multiple-choice question generation using natural language processing techniques. **arXiv preprint arXiv:2103.14757**, 2021. Citado na página 8.

OSHIKAWA, R.; QIAN, J.; WANG, W. Y. A survey on natural language processing for fake news detection. **arXiv preprint arXiv:1811.00770**, 2018. Citado na página 8.

PANDEY, D.; NIWARIA, K.; CHOURASIA, B. Machine learning algorithms: a review. **Mach. Learn**, v. 6, n. 2, 2019. Citado 2 vezes nas páginas 11 e 12.

POYNTER. **A Global Study on Information Literacy**. Poynter Institute, MediaWise, YouGov Inc., Google, 2022. Disponível em: <https://www.poynter.org/wp-content/uploads/2022/08/A-Global-Study-on-Information-Literacy-1.pdf>. Citado na página 8.

PRASEED, A.; RODRIGUES, J.; THILAGAM, P. S. Hindi fake news detection using transformer ensembles. **Engineering Applications of Artificial Intelligence**, v. 119, p. 105731, 2023. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197622007217>. Citado 4 vezes nas páginas 4, 9, 16 e 17.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. Improving language understanding by generative pre-training. 2018. Disponível em: <https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf>. Citado na página 24.

SCIKIT-LEARN. **SGDClassifier — Scikit-Learn**. 2025. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html>. Acessado em 3 de setembro de 2025. Citado na página 14.

SHAIK, A. B.; SRINIVASAN, S. A brief survey on random forest ensembles in classification model. In: BHATTACHARYYA, S.; HASSANIEN, A. E.; GUPTA, D.; KHANNA, A.; PAN, I. (Ed.). **International Conference on Innovative Computing and Communications**. Singapore: Springer Singapore, 2019. p. 253–260. ISBN 978-981-13-2354-6. Citado na página 13.

SOUZA, C. E. d. Detecção de fake news em redes sociais com o uso de redes neurais recorrentes, redes neurais gráficas e transformers. **REPOSITORIO PUCSP: Página inicial**, Pontifícia Universidade Católica de São Paulo, 2023. Disponível em: <https://repositorio.pucsp.br/jspui/handle/handle/39999>. Citado na página 8.

SUN, C.; QIU, X.; XU, Y.; HUANG, X. How to fine-tune bert for text classification? In: SUN, M.; HUANG, X.; JI, H.; LIU, Z.; LIU, Y. (Ed.). **Chinese Computational Linguistics**. Cham: Springer International Publishing, 2019. p. 194–206. ISBN 978-3-030-32381-3. Citado na página 23.

TAUNK, K.; DE, S.; VERMA, S.; SWETAPADMA, A. A brief review of nearest neighbor algorithm for learning and classification. In: **2019 International Conference on Intelligent Computing and Control Systems (ICCS)**. [S.l.: s.n.], 2019. p. 1255–1260. Citado na página 12.

TEIXEIRA, A. **Fake news contra a Vida: Desinformação Ameaça Vacinação de Combate à Febre Amarela**. Pontifícia Universidade Católica de São Paulo, 2019.

Disponível em: <https://repositorio.pucsp.br/jspui/handle/handle/21972>. Citado na página 10.

TUNSTALL, L.; WERRA, L. V.; WOLF, T. **Natural language processing with transformers**. [S.l.]: "O'Reilly Media, Inc.", 2022. Citado 6 vezes nas páginas 4, 9, 10, 11, 15 e 16.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017. Disponível em: <https://arxiv.org/abs/1706.03762>. Citado 6 vezes nas páginas 8, 14, 15, 16, 24 e 33.

Apêndices

Anexos