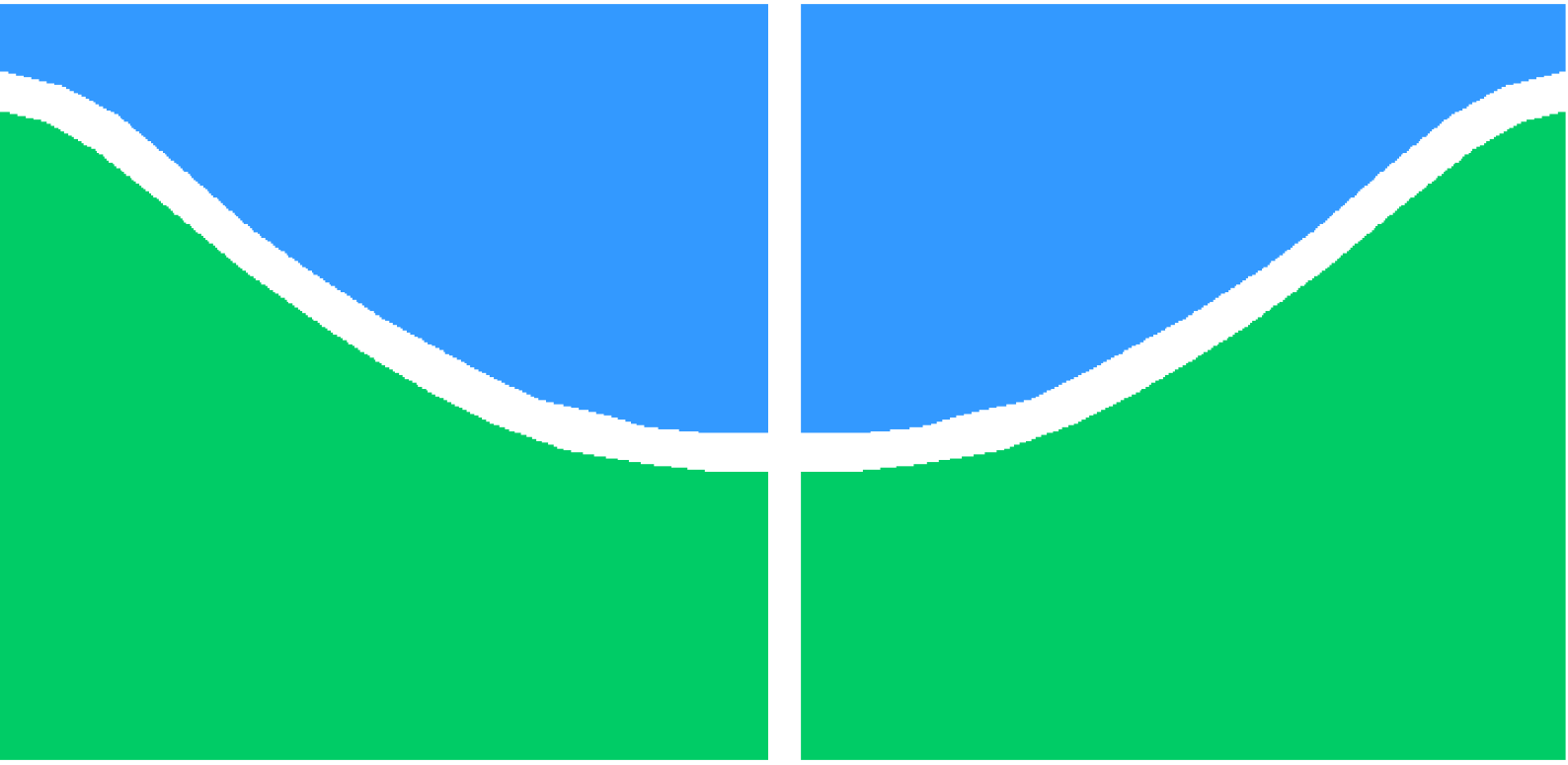


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Treinando e comparando modelos de *Deep Learning* para a tarefa de classificação de *Fake News* a partir de seus elementos textuais.

Autor: Marcos Gabriel Tavares

Orientador: Professor Dr. Fabricio Ataides Braz

Brasília, DF

2024

Marcos Gabriel Tavares

Treinando e comparando modelos de *Deep Learning* para a tarefa de classificação de *Fake News* a partir de seus elementos textuais.

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Professor Dr. Fabricio Ataides Braz

Brasília, DF

2024

Marcos Gabriel Tavares

Treinando e comparando modelos de *Deep Learning* para a tarefa de classificação de *Fake News* a partir de seus elementos textuais.

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 10 de julho de 2024:

Professor Dr. Fabricio Ataides Braz
Orientador

Professor Dr. Nilton Correia da Silva
Convidado 1

Professor Dr. Henrique Marra Taira Menegaz
Convidado 2

Brasília, DF
2024

Agradecimentos

Gostaria de agradecer ao meu pais, pois sem eles, sem o amor e o apoio que me foi dado, não seria capaz de chegar aqui.

"As máquinas me surpreendem muito frequentemente."
(Alan Mathison Turing)

Resumo

No cenário pós-Covid, as tecnologias de informação evoluem continuamente, impulsionando novas formas de consumo de informações pela internet. O acesso crescente à internet expõe os indivíduos a uma quantidade cada vez maior de informações, tornando-se um ambiente competitivo onde agentes maliciosos aproveitam o caos para disseminar Fake News com objetivos financeiros ou políticos. Esse fenômeno é especialmente evidente em aplicativos de mensagem, onde notícias falsas são compartilhadas em grupos obscuros, dificultando seu controle. Diante desse desafio, torna-se necessária uma abordagem automatizada para detectar essas notícias falsas. Este trabalho visa desenvolver um modelo que sane essa necessidade. Esse trabalho visa coletar dados de Fake News rotulados e treinar três modelos de aprendizado de maquina com três arquiteturas de *Deep Learning* distintas utilizando *embeddings* para a representação de dados, para que seja possível compara-los e selecionar o mais adequado para a tarefa.

Palavras-chave: Inteligência Artificial; Processamento de Linguagem Natural (NLP); Convoluted Neural Networks; Long Short Term Memory; Bidirectional Encoder Representations from Transoformers; Classificação; Fake News; Deep Learning; Aprendizado de máquina; Redes neurais; Datasets;

Abstract

In the post-Covid scenario, information technologies are continuously evolving, driving new ways of consuming information online. The growing internet access exposes individuals to an increasing amount of information, creating a competitive environment where malicious actors exploit the chaos to spread Fake News for financial or political purposes. This phenomenon is particularly evident in messaging apps, where fake news are shared in obscure groups, making moderation challenging. Faced with this challenge, an automated approach becomes necessary to detect such fake news. This work aims to develop a model to address this need. This study aims to colect labeled fake news data to train three machine learning models using three distinct *Deep Learning* architectures using embeddings for data representation, allowing for comparison and selection of the most suitable model for the task.

Key-words: Artificial Inteligence; Natural Language Processing (NLP); Convoluted Neural Networks; Long Short Term Memory; Bidirectional Encoder Representations from Transoformers; Classification; Fake News; Deep Learning; Machine Learning; Neural Nets; Datasets;

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UnB	Universidade de Brasília
LSTM	Long Short Term Memory
CNN	Convoluted Neural Network
RNN	Recurrent Neural Network
BERT	Bidirectional Enconding Representations from Transformers
PLN	Processamento de Linguagem Natural
TF-IDF	Term Frequency-Inverse Document Frequency
Word2Vec	Word to Vector
ReLu	Rectified Linear Unit
LLM	Large Language Model

Sumário

1 Introdução

1.1 Contexto

No mundo pós-Covid, onde as tecnologias de informação estão em constante evolução, as novas formas de se consumir informação através da internet prosperam. Devido ao crescente numero de pessoas com acesso à internet, os indivíduos hoje são expostos a cada vez mais informação, e em meio a esse ambiente competitivo, onde novas plataformas disputam constantemente pela atenção de seus potenciais clientes, agentes maliciosos fazem uso do caos para espalhar informações polêmicas com objetivo financeiro ou politico.

Em aplicativos de mensagem instantânea noticias falsas e polemicas, também conhecidas como *Fake News*, são compartilhadas privadamente em grupos obscuros, dificultando ainda mais seu controle. Como exemplo desse fenômeno, durante a pandemia do COVID-19, uma enxurrada de desinformação se alastrou pela internet com objetivos políticos e financeiros como apontado no estudo de (NETO et al., 2020). O volume dessas noticias que já era imenso cresceu mais ainda, se fazendo a necessidade de uma forma automatizada de se detectá-las.

1.2 Problema

As pessoas hoje são expostas a quantidades cada vez maiores de *Fake News* em plataformas digitais, e distinguir uma *Fake News* de uma noticia real não é uma tarefa trivial, devido a isso averiguar a veracidade de todas elas se torna uma tarefa hercúlea. Tendo em vista a situação atual do espaço virtual, se faz necessário uma forma de indicar a probabilidade de que uma noticia seja falsa para que os envolvidos façam decisões mais prudentes com base nessa informação.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é realizar o treinamento supervisionado de diversos modelos de aprendizado de máquina utilizando dados rotulados de notícias falsas (*Fake News*), a fim de identificar o mais eficaz na detecção dessas notícias por meio da análise de seus elementos textuais. Essa avaliação será conduzida mediante a comparação das métricas desses modelos em um subconjunto específico de teste.

1.3.2 Objetivos Específicos

1. Selecionar dados em português brasileiro contendo notícias falsas rotuladas.
    - Realizar analise exploratória dos dados.
    - Realizar limpeza dos dados.
    - Separar dados em conjuntos de treino, validação e teste.
    - Realizar aumento de dados no conjunto de treino.
2. Treinar modelos de classificação binaria para identificar *Fake news*.
    - Treinar modelo de classificação binaria utilizando o *transformer BERT*.
    - Treinar modelos de classificação binaria utilizando 2 arquiteturas, sendo elas *CNN* e *LSTM*, com camadas de *embedding* inicializadas com pesos pré-treinados em português utilizando *CBOW* e *Skip-Gram*.
3. Comparar resultados dos modelos utilizando o conjunto de teste.

1.4 Organização do Trabalho

Este trabalho de conclusão de curso está organizado nos seguintes capítulos:

- **Capítulo 1 - Introdução:** neste capítulo foram apresentados o contexto do trabalho, o problema de pesquisa e os objetivos deste trabalho.
- **Capítulo 2 - Referencial teórico:** descreve os conceitos que fundamentam este trabalho.
- **Capítulo 3 - Materiais e Métodos:** apresenta o plano metodológico adotado de forma mais detalhada e caracteriza o objeto de estudo.
- **Capítulo 4 - Resultados e conclusão:** Apresenta os resultados obtidos a partir da execução deste trabalho e discute o projeto como um todo.

2 Referencial Teórico

2.1 Processamento de Linguagem Natural (PLN)

O Processamento de Linguagem Natural (PLN) é um campo multidisciplinar que combina conhecimentos de linguística e aprendizado de máquina (FACE, 2021). Seu objetivo é capacitar os computadores a compreender e interpretar a linguagem humana. Uma das principais dificuldades enfrentadas no PLN é a ambiguidade presente na linguagem natural. Palavras e frases podem ter múltiplos significados, dependendo do contexto em que são utilizadas o que é fácil para humanos compreender mas um desafio para computadores (KHURANA et al., 2023). Portanto, os algoritmos de PLN precisam ser capazes de analisar e inferir o sentido correto com base no contexto.

O artigo (KHURANA et al., 2023) enumera as seguintes aplicações para PLN:

1. Tradução de maquina: Traduzir sentenças em diversas linguás mantendo o sentido e regras gramaticais
2. Categorização de Texto: Consumir uma quantidade grande de dados como documentos jurídicos, noticias e relatórios militares e atribui-los à categorias predeterminadas.
3. Extração de informação: Identificar itens de interesse como nomes, lugares, eventos, datas e valores em dados textuais.
4. Sumarização: Sumarizar dados mantendo seu sentido intacto.
5. Sistemas de diálogo (*chatbots*): Criar sistemas capazes de interagir com humanos utilizando linguagem natural.

2.1.1 Métricas

Para que possamos decidir se um modelo de classificação está capturando um padrão de forma acurada é necessário avaliar o modelo (BIRD; KLEIN; LOPER, 2009) e para isso métricas são utilizadas.

- Matriz de confusão: Uma matriz de confusão é uma tabela onde cada célula [i, j], sendo i a linha e j a coluna, indica o quão frequente a etiqueta j foi indicada quando a etiqueta correta era i. Dessa forma as células na diagonal representam etiquetas corretamente indicadas (True Positive, True Negative) e as fora da diagonal representam etiquetas erroneamente indicadas (False Positive, False Negative)

(BIRD; KLEIN; LOPER, 2009). Um exemplo de matriz de confusão pode ser visto na TABELA 1.

	Fake News	Não *Fake News*
Fake News	True Positive	False Negative
Não *Fake News*	False Positive	True Negative

TABELA 1 – Matriz de Confusão

Os termos utilizados na TABELA 1 são:

- True Positive (Verdadeiro Positivo): *Fake News* corretamente indicadas como *Fake News* pelo modelo
- False Postive (Falso Positivo): Não *Fake News* erroneamente indicadas como *Fake News* pelo modelo
- True Negative (Verdadeiro Negativo): Não *Fake News* corretamente idicadas como não *Fake News* pelo modelo
- False Negative (Falso Negativo): *Fake News* erroneamente indicadas como não *Fake News* pelo modelo.

- *Support*: Número de instâncias de uma classe.
- *Precision*: Sendo T o conjunto de rótulos reais e S o conjunto de rótulos inferidos, *Precision* é numero de elementos da intersecção entre T e S dividido pelo número de elementos de S (GODBOLE; SARAWAGI, 2004).

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- *Recall*: Sendo T o conjunto de rótulos reais e S o conjunto de rótulos inferidos, *Recall* é numero de elementos da intersecção entre T e S dividido pelo número de elementos de T (GODBOLE; SARAWAGI, 2004).

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- *F1-Score*: *F1-Score* é a media harmônica entre *Precision* e *Recall* (BIRD; KLEIN; LOPER, 2009).

$$F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

- *Accuracy*: *Acurracy* é porcentagem de inferências corretas sobre o total de inferências (BIRD; KLEIN; LOPER, 2009).

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$$

2.2 Nuvem de Palavras

Uma nuvem de palavras é uma representação visual das palavras mais frequentes em um determinado conjunto de texto. Uma nuvem de palavras pode ser gerada a partir de documentos, discursos, artigos científicos, noticias ou qualquer outra fonte de texto relevante. Nuvens de palavras têm sido usadas há anos em vários áreas, contudo encontra mais uso em sistemas de informação baseados na web, como sistemas de gerenciamento de conteúdo, plataformas de crowdsourcing, redes sociais redes e outros (KALMUKOV, 2021).

2.3 Stop Words

Stop Words são palavras extremamente comuns que têm pouco valor na seleção de documentos correspondentes às necessidades do usuário e, portanto, são excluídas completamente do vocabulário. Essas palavras, conhecidas como Stop Words, são determinadas por meio de uma estratégia de coleta que envolve classificar os termos por frequência na coleção (o número total de vezes que cada termo aparece na coleção de documentos) (MANNING; RAGHAVAN; SCHUETZE, 2009).

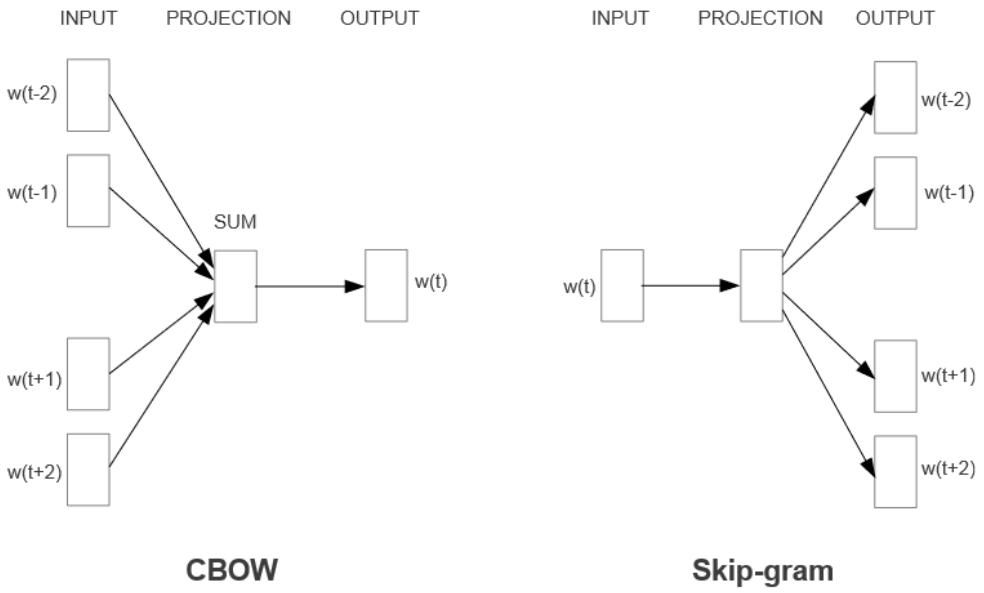
As palavras mais frequentes, muitas vezes filtradas manualmente com base em seu conteúdo semântico em relação ao domínio dos documentos indexados, formam uma lista conhecida como Stop List. Esses termos são descartados durante o processo de indexação, reduzindo significativamente o número de postagens que o sistema precisa armazenar. Embora a exclusão de Stop Words geralmente não prejudique pesquisas por palavras-chave comuns, ela pode afetar negativamente pesquisas por frases específicas, destacando a importância de considerar o contexto ao determinar as Stop Words. Algumas das palavras consideradas Stop Words são artigos, conjunções, preposições e pronomes (MANNING; RAGHAVAN; SCHUETZE, 2009).

2.4 *Word2Vec (Word to Vector)*

O software Word2Vec desenvolvido no artigo (MIKOLOV et al., 2013) propõe e faz uso de 2 arquiteturas para computar representações de palavras em vetores. Este algorítimo atribui vetores de números aleatórios para as palavras de um texto em especifico então, utilizando uma das duas arquiteturas propostas, tenta prever uma palavra em especifico ou as palavras na vizinhança de uma palavra selecionada, apos isso é calculada a similaridade por cosseno entre o vetor previsto e o vetor da palavra atual no texto. Essa diferença é então utilizada para treinar o modelo da arquitetura em questão através de retropropagação. A primeiro arquitetura denominada *CBOW (Continuous Bag-of-Words)* prediz a palavra atual baseado no contexto, já a segunda arquitetura, denominada *Skip-*

Gram, prediz as palavras na vizinhança de uma palavra, o oposto de *CBOW*. A FIGURA 1 contem a arquitetura dos dois modelos representadas de forma visual.

FIGURA 1 – CBOW - Skip-Gram



Fonte: (MIKOLOV et al., 2013)

2.5 *TF-IDF (Term Frequency–Inverse Document Frequency)*

Proposto inicialmente por (LEE; Huei Chuang; SEAMONS, 1997), TF-IDF tem como objetivo criar uma matriz de pesos para termos em um documento diminuindo o impacto de termos mais frequentes.

Segundo o modelo TF-IDF implementado no pacote *Scikit-Learn* (PEDREGOSA et al., 2011): *TF* é a frequência de um termo dentro de um documento. *IDF* é o logaritmo natural da frequência da presença de um termo em um grupo de documentos. TF-IDF é a multiplicaçao desses dois termos. O resultado é uma matriz de pesos para os termos nos documentos que pode ser utilizada para representar o documento. Essas siglas correspondem às equações:

$$TF(t,d) = n(t)/n(d) \tag{2.1}$$

Onde:

n(t) - é o numero de vezes que um termo aparece em um documento.

n(d) - é o numero total de termos em um documento.

TF(t, d) - é a frequência do termo em um documento.

$$IDF(t) = \log_n(d(todos)/d(t)) + 1 \tag{2.2}$$

Onde:

d(todos) - é o numero total de documentos.

d(t) - é o numero de documentos que contém o termo.

IDF(t) - é o logaritmo natural da frequência da presença do termo em um grupo de documentos.

$$TFIDF(t,d) = TF(t,d) * IDF(t) \tag{2.3}$$

Onde:

TFIDF(t) - é o a frequência do termo no documento vezes o logaritmo natural da frequência da presença do termo em um grupo de documentos.

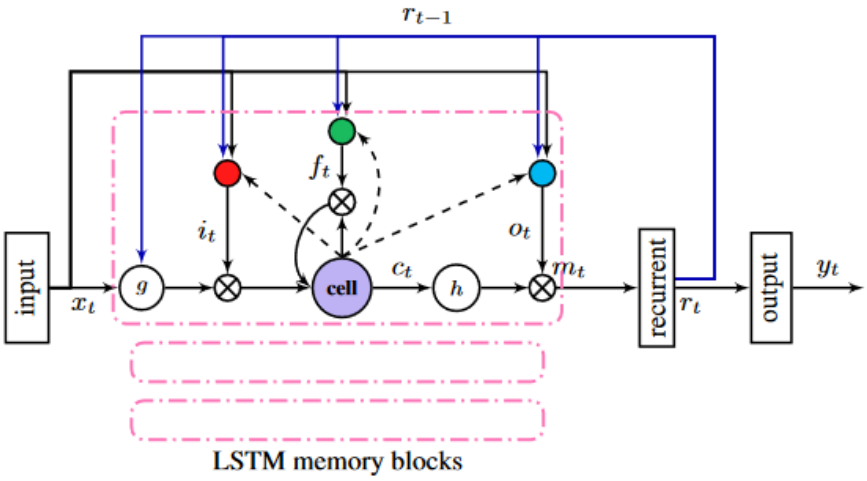
2.6 Aprendizado de máquina

2.6.1 *Deep Learning*

2.6.1.1 LSTM (Long Short Term Memory)

Long Short-Term Memory (LSTM) é um tipo especifico de *Recurrent Neural Network* que foi projetado para modelar sequencias temporais e sua dependências de longo alcance de forma mais acurada que RNNs convencionais (DAHL et al., 2012). Um LSTM contém unidade especiais chamadas blocos de memoria na camada oculta recorrente. Esses blocos de memoria contém células de memória que conectam a si mesmas armazenando o estado temporal da rede, também possuindo unidade multiplicativas especiais chamadas *gates* para controlar o fluxo de informação. Cada bloco de memória na arquitetura contém um *input gate*, um *output gate* e um *forget gate*. O *input gate* controla o numero de ativações de entrada na célula de memória. O *output gate* controla o fluxo de saída de ativações para o resto da rede. O *forget gate* escala o estado interno de cada célula antes de adicionar ele como input para a célula através da conexão auto-recorrente da célula, causando assim o esquecimento adaptativo da memória da célula (DAHL et al., 2012). A FIGURA 2 representa uma arquitetura LSTM com apenas um bloco de memória.

FIGURA 2 – Arquitetura LSTM com um bloco de memória

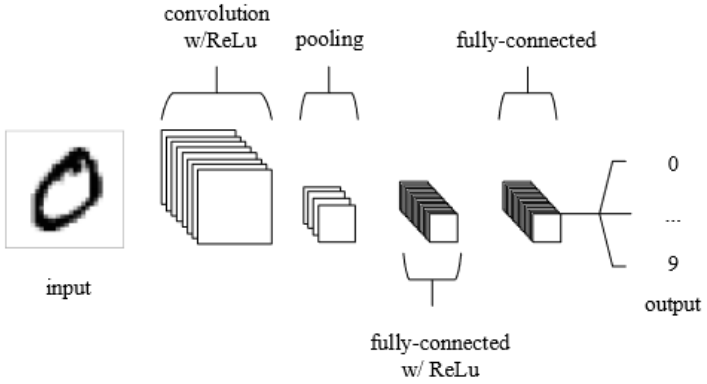


Fonte: (DAHL et al., 2012)

2.6.1.2 CNN (Convolutional Neural Networks)

A *Convolutional Neural Network* (CNN) é uma das redes neurais mais significativas no campo de *Deep Learning*. Tendo em vista que as CNNs obtiveram conquistas impressionantes em diversas áreas, incluindo, mas não se limitando à visão computacional e processamento de linguagem natural, elas atraíram muita atenção tanto da indústria quanto da academia nos últimos anos (LI et al., 2022). CNNS são compostas de 3 tipos de camadas, sendo elas as *convolutional layers*, *pooling layers* e as *fully-connected layers*. Quando essas camadas são empilhadas, uma arquitetura CNN (FIGURA 3) é formada (O'SHEA; NASH, 2015).

FIGURA 3 – Arquitetura CNN composta de 5 camadas



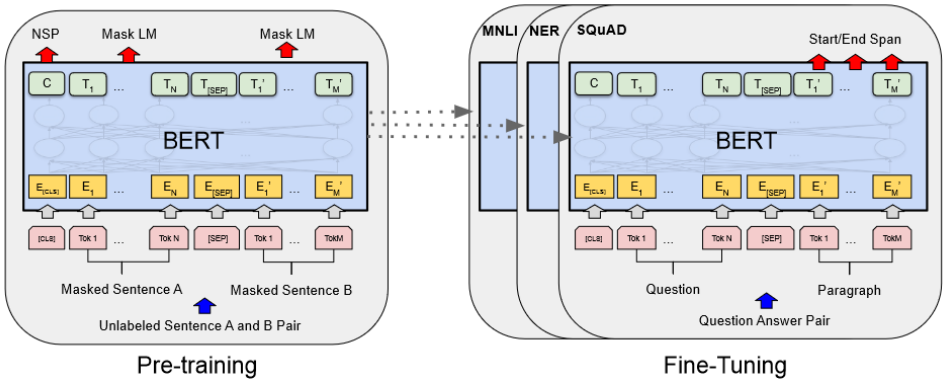
Fonte: (O'SHEA; NASH, 2015)

- *Convolutional layers*: Essa camada determina a saída dos neurônios que estão conectados às regiões locais da entrada através do calculo do produto escalar entre seus pesos e a região conectada ao volume da entrada. A *rectified linear unit* (também conhecida como ReLu) tem como objetivo aplicar uma função de ativação como, por exemplo, uma sigmoide, a cada elemento da saída produzida pela camada anterior.
- *Pooling layers*: Essa camada ira realizar um *downsampling* ao longo das dimensões espaciais de uma dada entrada, reduzindo ainda mais o numero de parâmetros naquela ativação.
- *Fully-connected layers*: Essa camada ira realizar a mesma função que uma rede neural comum, gerando *scores* para cada classe a partir das ativações das outras camadas, podendo assim ser utilizada para classificação.

2.6.1.3 BERT (Bidirectional Encoder Representations from Transformers)

A arquitetura BERT foi projetada para ser pré treinada em representações bidirecionais profundas em texto não rotulado através do condicionamento conjunto em ambos contextos, o direito e o esquerdo, em todas as camadas. Como resultado, o modelo BERT pré treinado pode ser afinado com apenas mais uma camada de saída adicional para criar modelos de aprendizado de maquina *state of the art* para um vasta gama de tarefas sem modificações significativas em sua arquitetura (DEVLIN et al., 2019). Na FIGURA 4, fora a camada de saída, a mesma arquitetura é utilizada para o pré treinamento e para o afinamento.

FIGURA 4 – Arquitetura BERT - Pré treinamento e afinamento



Fonte: (DEVLIN et al., 2019)

2.6.2 Aumento de dados

Aumento de dados (*Data augmentation*) se refere à estrategias para aumentar a diversidade dos exemplos de treino sem explicitamente coletar novos dados (FENG et al., 2021). As técnicas de *Data augmentation* podem ser divididas em 3 categorias:

- Técnicas baseadas em regras: Usam transformações predefinidas fáceis de calcular, sem componentes de modelo.
- Técnicas de interpolação de exemplos: Interpolam dois ou mais exemplos reais para criar novos itens.
- Técnicas baseadas em modelos: Fazem uso de modelos para transformar exemplos reais.

2.6.2.1 Parafraseamento

Parafraseamento é uma técnica de aumento de dados baseada em modelos que gera paráfrases de itens reais utilizando um modelo generativo.

2.6.2.2 *Backtranslation*

Backtranslation é uma técnica de aumento de dados baseada em modelos que gera novas frases com o mesmo significado que a frase original, expandindo significativamente seu conjunto de dados. Ela funciona traduzindo a frase original para um idioma estrangeiro e depois traduzindo-a de volta para o idioma original.

2.6.3 Otimização de hiperparâmetros

A otimização de hiperparâmetros consiste na utilização de algorítimos para explorar diferentes combinações de parâmetros de treinamento de um modelo com o objetivo de otimizar a sua performance em relação à(s) métrica(s) de interesse. Todo sistema de aprendizado de maquina possui hiperparâmetros, e a tarefa mais básica em aprendizado de maquina automático é definir esses hiperparâmetros de forma automática. As redes neurais profundas mais recentes dependem de uma vasta gama de escolhas de hiperparâmetros sobre a arquitetura, regularização e otimização (HUTTER; KOTTHOFF; VANSCHOREN, 2019).

Esses parâmetros frequentemente influenciam a complexidade, comportamento, velocidade e outros comportamentos do algorítimo, e esses valores devem ser selecionados com cuidado para que seja possível atingir uma performance otimizada. Determinar esses parâmetros com tentativa e erro não só é demorado como é enviesado e irreproduzível (BISCHL et al., 2021).

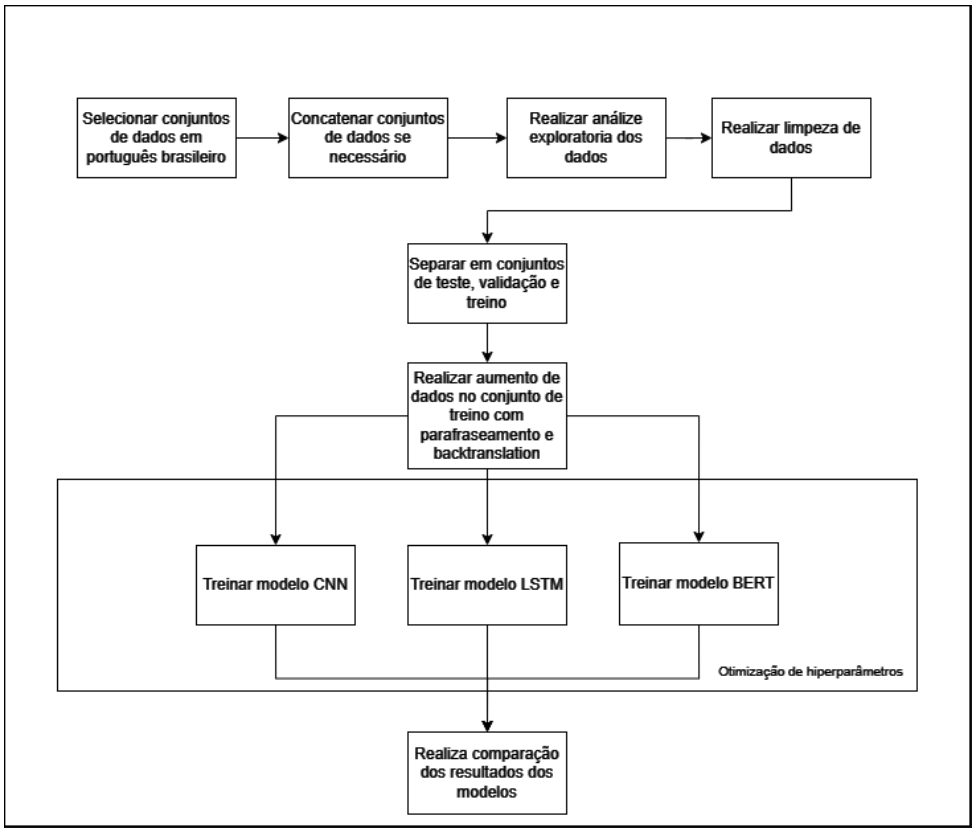
3 Materiais e Métodos

3.1 Considerações Iniciais

Neste capítulo apresenta-se os materiais e métodos utilizados para alcançar os objetivos desse trabalho. Será detalhado o plano metodológico adotado com as etapas que serão seguidas para treinar os modelos de aprendizado de máquina que irão sugerir se o texto alimentado a eles é *Fake News* ou não e a conseguinte comparação entre eles para encontrar o modelo que performa essa tarefa de forma mais adequada. O fluxograma do projeto está detalhado na FIGURA 5.

3.2 Plano Metodológico

FIGURA 5 – Fluxograma do Projeto



3.2.1 Selecionar conjuntos de dados em português brasileiro

Para iniciar o projeto foi necessário realizar a seleção de dados rotulados com *Fake News* em texto na língua portuguesa. Um conjunto de dados viável deve possuir noticias

falsas e verdadeiras rotuladas para para que o modelo de aprendizado de máquina esteja coerente com o objetivo deste trabalho. As bases serão concatenadas em um único conjunto de dados.

3.2.2 Realizar análise exploratória dos dados

A análise exploratória dos dados desempenha um papel fundamental no desenvolvimento de inteligência artificial, pois proporciona uma compreensão aprofundada das características dos dados. Durante essa etapa, é possível identificar a presença de dados inconsistentes ou irrelevantes, os quais podem ser removidos. A ausência dessa análise e das correções sugeridas pode levar ao treinamento do modelo com informações inadequadas, prejudicando seu desempenho e resultando em resultados incorretos. Assim, a análise exploratória é crucial para assegurar que os dados estejam adequados antes do início do treinamento, resultando em dados de maior qualidade e, consequentemente, em resultados mais satisfatórios. Os passos para realizar a análise exploratória são:

- Verificar estado inicial do dado.
- Verificar a existência de texto não rotulado.
- Verificar a existência de valores nulos.
- Verificar a quantidade de valores de cada classe de interesse.
- Verificar número de caracteres de cada item.

3.2.3 Realizar a limpeza dos dados

Tendo sido realizada a análise exploratória dos dados pode-se dar inicio a limpeza dos dados. A limpeza dos dados consiste em remover elementos indesejados do nosso conjunto de dados. A seguir estão os passos a serem executados para realizar a limpeza:

- Remover itens marcados como arquivo de mídia: Serão removidos os itens marcados como arquivo de mídia
- Remover texto não rotulado: Serão removidos itens no conjunto de dados que não possuem rótulo.
- Remover links: Serão removidos o link para sites online como por exemplo:

  https://www.link.com.
- Remover espaços vazios dos textos: Serão removidos espaços vazios de no inicio e fim do texto.

- Remover valores nulos e texto vazio: Serão removidos itens no conjunto de dados que possuem valor nulo ou texto vazio.
- Remover texto duplicado: Serão removidos itens no conjunto de dados que possuem valores idênticos.

3.2.4 Separar o conjunto de dados em conjuntos de teste, validação e treinamento

Para que seja possível iniciar o treinamento dos modelos, é necessário dividir o conjunto de dados em 3 conjuntos, sendo eles:

- Teste: O conjunto de teste é necessário para verificar a performance do modelo após o treinamento com dados que ainda não foram vistos por ele. Este conjunto contém 10% dos itens do conjunto inicial.
- Validação: O conjunto de validação é necessário para verificar a performance dos modelo durante o treinamento para o ajuste de hiperparâmetros. Este conjunto contém 10% dos itens do conjunto inicial
- Treinamento: O conjunto de treinamento é necessário para realizar o treinamento do modelo. Este conjunto contém 80% dos itens do conjunto inicial

Os conjuntos de teste e validação serão sujeitos a um filtro com *TF-IDF* para evitar vazamento de dados para o conjunto de treino.

3.2.5 Realizar aumento de dados no conjunto de treinamento

Serão utilizadas as técnicas de parafraseamento e *back-translation* para gerar dados artificiais de treino. Esses dados serão utilizados como hiperparâmetro. Para a técnica de parafraseamento será feito uso de um *LLM* capaz de reconhecer e gerar texto em português em conjunto com a ferramenta Ollama. Para a técnica de *backtranslation* será feito o uso do modelo capaz de traduzir texto em português para inglês e texto em inglês para português.

3.2.6 Treinar modelos de aprendizado de máquina

Para o treinamento dos modelos serão utilizadas as ferramentas Optuna (AKIBA et al., 2019) e MLflow (ZAHARIA et al., 2019) para realizar um procedimento de otimização de hiperparâmetros com o objetivo de encontrar um conjunto de hiperparâmetros que maximize a métrica acurácia. Serão utilizados quatro variações do conjunto de treinamento:

- Conjunto de treino sem aumento de dados.
- Conjunto de treino com aumento de dados com a técnica de parafraseamento.
- Conjunto de treino com aumento de dados com a técnica de *backtranslation*.
- Conjunto de treino com aumento de dados com as técnicas de parafraseamento e *backtranslation*.

3.2.6.1 Treinar o modelo BERT

Para realizar o treinamento do modelo BERT será utilizada a biblioteca Hugging Face em conjunto com TensorFlow na linguagem Python. Aqui estão alguns do modelos pré-treinados em português que podem ser selecionados como base para o treinamento:

- BERTimbau
- distilbert-base-multilingual-cased
- xlm-roberta-base
- google/mt5-base

3.2.6.2 Treinar o modelo *CNN*

O modelo *CNN* será treinado utilizando a biblioteca TensorFlow em conjunto com Keras na linguagem Python. Será selecionada uma arquitetura contendo primariamente *convolutional layers* e uma camada de *embedding* que será inicializada utilizando pesos pré-treinados com *CBOW* ou *Skip-Gram*.

3.2.6.3 Treinar o modelo *LSTM*

O modelo *LSTM* será treinado utilizando a biblioteca TensorFlow em conjunto com Keras na linguagem Python. Será selecionada uma arquitetura contendo primariamente camadas *LSTM* e uma camada de *embedding* que será inicializada utilizando pesos pré-treinados com *CBOW* ou *Skip-Gram*.

3.2.7 Realizar a comparação dos resultados dos modelos

Os 3 melhores modelos (*BERT*, *CNN* e *LSTM*) frutos da otimização de hiperparâmetros baseada na acurácia serão comparados de forma quantitativa com base nas métricas descritas na secção 2.1.1. As métricas serão obtidas da execução dos modelos no conjunto de teste.

3.3 Ferramentas

3.3.1 Google Colab

O Google Colab, cujo nome é uma abreviação de Google Colaboratory, constitui uma plataforma de código aberto disponibilizada pelo Google. Essa plataforma viabiliza a execução de código Python diretamente no navegador, eliminando a necessidade de instalação de software no computador local. Utilizando o ambiente de execução do Jupyter Notebook, o Colab proporciona recursos de processamento em nuvem como unidades de processamento gráfico disponibilizadas pela Google (BISONG, 2019).

3.3.2 Hugging Face

A Hugging Face é uma plataforma abrangente que inclui a biblioteca de código aberto Transformers e o Hugging Face Hub. A biblioteca Transformers é notável por oferecer uma variedade de modelos pré-treinados, como BERT e GPT, simplificando o treinamento e a aplicação de modelos de linguagem de última geração em tarefas de processamento de linguagem natural. O Hugging Face Hub, por sua vez, proporciona uma plataforma online onde desenvolvedores podem compartilhar, descobrir e utilizar modelos de aprendizado de máquina e outros recursos, fomentando a colaboração e tornando a tecnologia avançada mais acessível à comunidade de IA (WOLF et al., 2020).

3.3.3 GitHub

O GitHub é uma plataforma colaborativa de desenvolvimento que oferece serviços de hospedagem para controle de versão por meio do sistema Git. Lançado em 2008, o GitHub possibilita a colaboração entre desenvolvedores em projetos de software, facilitando o rastreamento de alterações no código, a gestão de problemas e a coordenação de esforços em equipes distribuídas globalmente. Os repositórios no GitHub funcionam como locais centralizados para armazenar, versionar e colaborar em código, incluindo ferramentas como rastreamento de problemas, solicitações de pull e integração contínua, contribuindo para a eficiência e qualidade no processo de desenvolvimento de software. Amplamente adotada na comunidade de desenvolvimento, essa plataforma oferece um ambiente robusto para o trabalho colaborativo e o compartilhamento de projetos de código aberto (GitHub. . . , 2023).

3.3.4 Jupyter

O Jupyter Notebook se destaca como um laboratório virtual de código aberto e baseado em navegador, revolucionando a forma como conduzimos e comunicamos a pesquisa científica. Mais do que um simples caderno, ele se transforma em um aliado poderoso

para registrar fluxos de trabalho, código, dados e visualizações, detalhando cada etapa do processo de pesquisa com clareza e precisão. Sua natureza híbrida, legível por humanos e máquinas, facilita a interoperabilidade e promove a comunicação acadêmica transparente. Os notebooks residem em repositórios online, servindo como portais para um universo de objetos de pesquisa interligados, como conjuntos de dados, código, documentação de métodos, fluxos de trabalho e publicações (Kluyver et al., 2016).

3.3.5 Ollama

O Ollama é um projeto open-source criado para facilitar a execução de *large language models* (LLM) como Llama 3 diretamente na sua máquina local. Ele funciona como uma ponte entre a complexidade desses poderosos modelos de inteligência artificial e uma experiência amigável para o usuário (OLLAMA, 2024). Esta ferramenta possui varias variáveis de configuração[1] e neste trabalho será modificada apenas a variável de temperatura, que indica o nível de criatividade do modelo. Quanto maior a temperatura mais criativo e aleatório responde o modelo.

3.4 Bibliotecas

3.4.1 Pandas

O pandas é uma biblioteca de código aberto em Python amplamente utilizada para manipulação e análise de dados. Essa ferramenta oferece estruturas de dados eficientes, como DataFrames e Series, que permitem a organização e manipulação intuitiva de conjuntos de dados tabulares e séries temporais. Com uma variedade de funções poderosas, o pandas facilita tarefas essenciais em ciência de dados, incluindo limpeza, transformação, indexação e agregação de dados. Sua flexibilidade e integração com outras bibliotecas, como NumPy e Matplotlib, fazem do pandas uma escolha popular entre cientistas de dados e desenvolvedores para análise e processamento de dados em projetos Python (TEAM, 2020).

3.4.2 Numpy

O NumPy é uma poderosa biblioteca de código aberto para a linguagem de programação Python, projetada para facilitar a manipulação eficiente de arrays e matrizes multidimensionais. Essa biblioteca fornece estruturas de dados otimizadas e uma ampla gama de funções para realizar operações numéricas, algebraicas e estatísticas de forma eficaz. Ao oferecer uma interface amigável para trabalhar com grandes conjuntos de dados numéricos, o NumPy é essencial para cientistas de dados, engenheiros e pesquisadores

[1] Parâmetros de configuração Ollama. Disponível em: <https://github.com/ollama/ollama/blob/main/docs/modelfile.md#parameter>

que necessitam de desempenho otimizado e funcionalidades avançadas em computação numérica (HARRIS et al., 2020).

3.4.3 Matplotib

O Matplotlib é uma biblioteca de visualização de dados em Python amplamente utilizada, que oferece uma variedade de ferramentas para criação de gráficos estáticos, interativos e visualizações complexas. Essa biblioteca fornece uma interface flexível para a criação de gráficos de linhas, barras, dispersão, histogramas, entre outros, permitindo que os usuários personalizem detalhes visuais e apliquem estilos específicos. O Matplotlib é particularmente valioso para cientistas de dados, pesquisadores e desenvolvedores que buscam comunicar efetivamente a partir de dados por meio de representações visuais (HUNTER, 2007).

3.4.4 Seaborn

O Seaborn é uma biblioteca de visualização de dados em Python que se baseia no Matplotlib e oferece uma interface de alto nível para criar gráficos estatísticos informativos e visualmente atraentes. Projetado especialmente para trabalhar em conjunto com estruturas de dados do pandas, o Seaborn simplifica a criação de visualizações complexas com poucas linhas de código. Ele fornece funções otimizadas para gráficos de dispersão, box plots, gráficos de barras, mapas de calor e outras visualizações estatísticas, facilitando a exploração e comunicação de padrões em dados (WASKOM, 2021).

3.4.5 WordCloud

A biblioteca WordCloud em Python é uma ferramenta eficaz para criar visualizações de nuvens de palavras a partir de textos, destacando as palavras mais frequentes de maneira gráfica e esteticamente atraente. Desenvolvida para simplificar a geração dessas nuvens de palavras, a biblioteca oferece opções de personalização, como cores, fontes e formas, permitindo que os usuários criem representações visuais distintas. Essa biblioteca é particularmente útil em análise de texto, mineração de dados e visualização de informações relevantes em grandes conjuntos de palavras (WORDCLOUD. . . , 2023).

3.4.6 TensorFlow

TensorFlow é um framework de código aberto para aprendizado de máquina e aprendizado profundo, desenvolvido pela equipe do Google Brain. Ele se tornou uma das ferramentas mais populares e amplamente adotadas para construção e treinamento de modelos de aprendizado profundo em diversas aplicações, incluindo visão computacional,

processamento de linguagem natural, reconhecimento de voz e muito mais. O TensorFlow é conhecido por sua arquitetura flexível e eficiente, permitindo a criação de modelos complexos com facilidade. Ele utiliza grafos computacionais para representar operações matemáticas, oferecendo a capacidade de otimizar e distribuir automaticamente o treinamento em hardware diversificado (ABADI et al., 2015).

3.4.7 Keras

Keras é uma interface de alto nível para construção e treinamento de redes neurais em Python, originalmente independente e agora integrada ao TensorFlow. Reconhecido por sua simplicidade e modularidade, o Keras oferece uma API intuitiva que permite aos usuários criar rapidamente modelos neurais complexos com menos código. Sua flexibilidade facilita a prototipagem e experimentação, sendo amplamente utilizado em tarefas de aprendizado profundo, como visão computacional e processamento de linguagem natural. A combinação de facilidade de uso e poder fez do Keras uma escolha popular na comunidade de aprendizado de máquina (CHOLLET, 2015).

3.4.8 Scikit-Learn

Scikit-learn é uma biblioteca de aprendizado de máquina em Python, oferecendo ferramentas eficientes e fáceis de usar para uma variedade de tarefas, como classificação, regressão e clustering. Conhecida por sua abordagem consistente e bem documentada, é amplamente utilizada em ambientes acadêmicos e industriais. A biblioteca fornece diversos algoritmos, métricas de avaliação e utilitários para pré-processamento de dados, sendo reconhecida pela eficiência, flexibilidade e integração com outras bibliotecas Python (PEDREGOSA et al., 2011).

3.4.9 Optuna

A Optuna é uma biblioteca de software de otimização automática de hiperparâmetros, projetada especificamente para aprendizado de máquina. Ela oferece uma API imperativa de estilo *define-by-run* para a interação do usuário. Graças a essa API, o código escrito com Optuna se beneficia de alta modularidade, permitindo ao usuário construir dinamicamente os espaços de busca para os hiperparâmetros (AKIBA et al., 2019). Aqui estão alguns conceitos importantes da ferramenta:

- *Study*: Otimização baseada em uma função de objetivo.
- *Trial*: Um única execução da função de objetivo.
- *Sampler*: O *sampler* possui a responsabilidade de determinar os valores dos parametros a serem avaliados em um *trial*.

- *Pruner*: O *pruner* é um funcionalidade que permite o Optuna a terminar um *trial* prematuramente caso ele não pareça promissor.

3.4.10 MlFlow

O MLflow surge como uma solução para muitos dos desafios enfrentados no dinâmico cenário do aprendizado de máquina. Ele oferece ferramentas e simplifica processos para otimizar o ciclo de vida de ML e promover a colaboração entre profissionais da área. A ferramenta fornece uma plataforma unificada para navegar pelo intrincado labirinto do desenvolvimento, implementação e gerenciamento de modelos. Ele visa permitir a inovação no desenvolvimento de soluções de aprendizado de máquina, simplificando tarefas como registro, organização e controle de linhagem, que costumam ser trabalhosas no desenvolvimento de modelos. Esse foco possibilita a garantia de que seus projetos de ML sejam robustos, transparentes e preparados para os desafios do mundo real (ZAHARIA et al., 2019).

3.5 Considerações Finais

O trabalho de conclusão de curso proposto aborda uma questão relevante na área de processamento de linguagem natural, que é a classificação de notícias falsas. As etapas planejadas foram estruturadas com o objetivo de treinar múltiplos modelos em diferentes arquiteturas para que seus resultados possam ser comparados. Para esse fim, é necessário selecionar dados adequados e prepara-los para os inicio do treinamento dos modelos. É importante lembrar que a qualidade dos dados coletados podem influenciar diretamente no desempenho dos modelos e nas análises subsequentes.

4 Resultados e conclusão

4.1 Considerações Iniciais

Neste capítulo é serão descritos e discutidos os resultados obtidos em cada etapa descrita no plano metodológico no Capitulo 3.

4.2 Selecionar conjuntos de dados em português brasileiro, contendo textos rotulados como Fake News ou não Fake News

A procura por conjuntos de dados adequados para os objetivos deste trabalho resultou na obtenção dos seguintes itens:

- **FakeWhatsApp.BR** (CUNHA, 2021).
- **Fakepedia Corpus** (CHARLES; RUBACK; OLIVEIRA, 2022).
- **FakeTrueBr** (CHAVARRO et al., 2023).
- **Multilabel** (MORAIS et al., 2019).

Estes conjuntos de dados foram concatenados em um arquivo *csv* que contém texto rotulado como *misinformation*, sendo o valor 0 atribuído para não Fake News, o valor 1 para Fake News e o valor -1 atribuído para dados não rotulados. O *notebook*[1] utilizado no preparo dos dados para o inicio da fase de treino está disponível no Github.

4.3 Realizar a análise exploratória dos dados

- Verificar estado inicial do dado. O estado inicial do dado está disposto na TABELA 2 logo abaixo:

Colunas	Contagem Total	Tipo de dado
text	299525	object
misinformation	299525	int64

TABELA 2 – Estado inicial do conjunto de dados

[1] *Notebook* de preparo dos dados. Disponível em: <https://github.com/marcosgtavares/TCC2-FGA/blob/v1/notebooks/TCC_CONCAT_CLEAN.ipynb>

- Verificar existência de texto não rotulado. O número de itens não rotulados no conjunto de dados é de 261312. Esses itens são indicados pelo número -1.

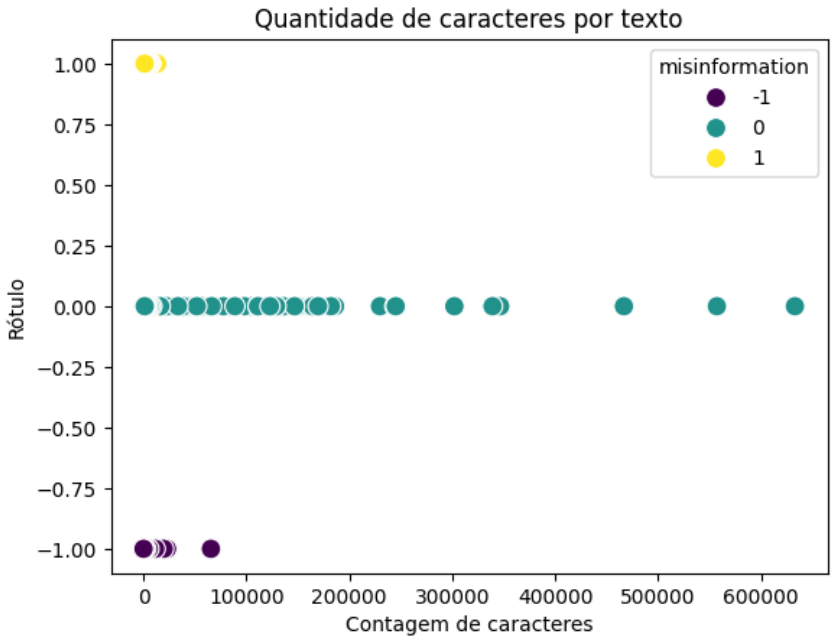
- Verificar a quantidade de valores de cada classe de interesse. O número de classes de interesse está indicado na TABELA 3 logo abaixo. O número 1 corresponde à Fake News, o número 0 corresponde a Não Fake News.

Rótulos	Contagem
1(Fake News)	19494
0(Não Fake News)	18719

TABELA 3 – Número inicial de classes de interesse no conjunto de dados

- Verificar o número e caracteres de cada item. A menor quantidade de caracteres em um item de texto é 1, a maior quantidade de caracteres em um item de texto é 633047. O conjunto de todos os dados concatenados possui uma média de 383.766 caracteres por item. A FIGURA 6 logo abaixo contém um gráfico da quantidade de caracteres por texto.

FIGURA 6 – Gráfico de dispersão do número de caracteres por texto



4.4 Realizar a limpeza dos dados

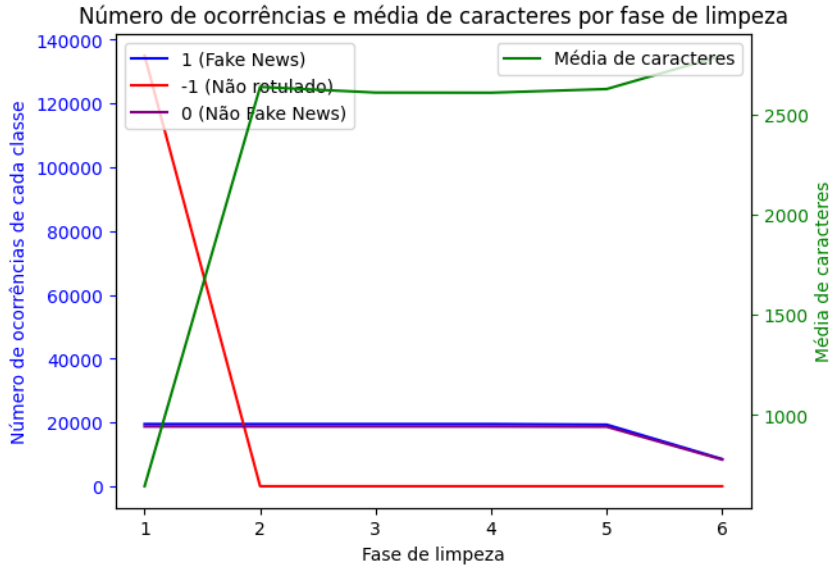
4.4.1 Visualização do efeito das fases de limpeza nos dados

A seguir estão as Legendas de cada fase na FIGURA 7:

- Remover itens marcados como arquivo de mídia. Fase 1.

- Remover texto não rotulado. Fase 2.
- Remover links. Fase 3.
- Remover espaços vazios dos textos. Fase 4.
- Remover valores nulos e texto vazio. Fase 5.
- Remover texto duplicado. Fase 6.

FIGURA 7 – Gráfico de numero de ocorrências e número de caracteres por fases de limpeza



O conjunto de dados, após a limpeza, possui o número de instâncias de cada classe indicadas na TABELA 4 a seguir. O número médio de caracteres foi 2793.675, significantemente maior que o inicial.

Rótulos	Contagem
1(Fake News)	8510
0(Não Fake News)	8338

TABELA 4 – Número de classes de interesse após a limpeza no conjunto de dados

4.4.2 Geração de *WordClouds* para visualização dinâmica das palavras em cada classe

Após o dado ter sido limpo pela primeira vez, foram criadas *WordClouds* para visualizar as palavras de cada classe de forma mais dinâmica. As *StopWords* foram removidas para a geração das *WordClouds*. A *WordCloud* para o rótulo 0 (Não Fake News)

se encontra na FIGURA 8. A *WordCloud* para o rótulo 1 (Fake News) se encontra na FIGURA 9.

FIGURA 8 – WordCloud do rótulo 0 (Não Fake News)

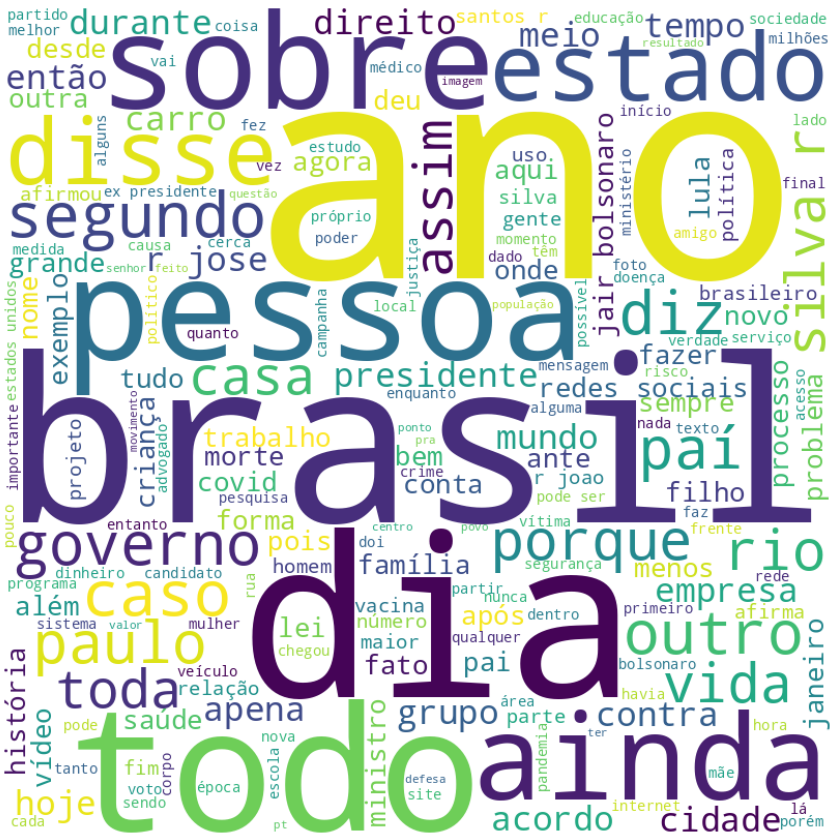


FIGURA 9 – WordCloud do rótulo 1 (Fake News)

4.5 Separar o conjunto de dados limpo resultante da limpeza em conjuntos de teste, validação e treinamento

Após o conjunto de dados ter sido limpo, ele foi separado em 3 outros conjuntos, sendo eles os conjuntos de teste, validação e treinamento. O conjunto de treinamento recebeu 80% do total dos textos, tendo recebido a maior carga. O conjunto de validação e o conjunto de teste receberam 10% do total da carga cada. Isso resultou nos valores dispostos na TABELA 5 logo abaixo:

Conjuntos	Rótulo 0 (Não Fake News)	Rótulo 1 (Fake News)	Total
Teste	842	842	1684
Validação	842	842	1684
Treino	6826	6654	13480

TABELA 5 – Quantidades de cada rótulo em cada conjunto

Ao separar os conjuntos de validação e teste, foi-se gerado um vetorizador *TF-IDF* com todos os dados limpos para que fosse possível gerar representações estatísticas *TF-IDF* dos textos. Utilizando esse resultado, a similaridade por cosseno das matrizes de pesos foi calculada, e apenas os itens com similaridade inferior a 0.26 entre todos os

textos foram incluídos nos dados de validação e de teste. O número 0.26 foi selecionado de forma que os dados que passassem por essa linha de corte possuíssem no mínimo 1684 itens de cada classe, sendo 1684 aproximadamente 10% da quantidade total de dados.

4.6 Realizar aumento de dados no conjunto de treinamento

Devido à limitações computacionais os textos que foram submetidos às técnicas de aumento de dados são aqueles cuja quantidade de conjuntos de caracteres (palavras) separados por espaço em branco é menor ou igual à 200. O número 200 foi escolhido levando em conta limitações de hardware.

4.6.1 Parafraseamento

Para realizar o aumento de dados utilizando a técnica de parafraseamento, foi-se utilizado o *LLM LLama 3* (AI@META, 2024) com 8 bilhões de parâmetros em conjunto com a ferramenta Ollama com o parâmetro temperatura igual à 0.2 (o valor padrão é 0.8, portanto a criatividade/aleatoriedade foi diminuída). O *notebook*[2] utilizado na execução desta técnica está disponível no Github. Um total de 7943 itens dos dados originais de treino foram parafraseados. O prompt utilizado foi:

Reescreva todo o texto abaixo de forma que ele mantenha sua mensagem intacta, ou seja, o parafraseie, trocando palavras por sinônimos quando possível deixando o texto de forma significantemente diferente com criatividade.

TEXTO A SER PARAFRASEADO

4.6.2 *Backtranslation*

Para realizar o aumento de dados utilizando a técnica de *Backtranslation* foi utilizado o modelo *mbart-large-50-many-to-many-mmt* (TANG et al., 2020). Os textos foram traduzidos para inglês e posteriormente traduzidos para português. O *notebook*[3] utilizado na execução desta técnica está disponível no Github. Um total de 7943 itens dos dados originais de treino foram submetidos à *backtranslation*.

4.7 Treinar modelos de aprendizado de máquina

Todos os modelos foram submetidos à otimização de hiperparâmetros. O modelo BERT foi sujeito à 24 iterações, cobrindo todas as possibilidades do hiperparâmetros

[2] *Notebook* de execução da técnica de parafraseamento. Disponível em: <https://github.com/marcosgtavares/TCC2-FGA/blob/v1/notebooks/TCC_PARAPHRASE_AUG.ipynb>

[3] *Notebook* de execução da técnica de *backtranslation*. Disponível em: <https://github.com/marcosgtavares/TCC2-FGA/blob/v1/notebooks/TCC_BACKTRANSLATE_AUG.ipynb>

configurados. Os modelos CNN e LSTM foram sujeitos à 400 iterações cada, contudo não foram exploradas todas as possibilidades possíveis devido à limitações de hardware e tempo. Outro fato que contribuiu para esse limite foi o fato dos modelos CNN e LSTM possuírem parâmetros com valores contínuos como *learning rate* e *epsilon*. Os hiperparâmetros categóricos foram iterados manualmente, totalizando 8 possibilidades para os modelo CNN e LSTM, sendo elas a combinação dos possíveis conjuntos de treino com os possíveis *embeddings* iniciais. Para o modelo BERT foram 4 possibilidades, sendo elas apenas os possíveis conjuntos de treino.

4.7.1 Treinar o modelo BERT

No treinamento do modelo BERT foi utilizado o *transformer bert-base-portuguese-cased* da *NeuralMind* também conhecido como BERTimbau base (SOUZA; NOGUEIRA; LOTUFO, 2020) em conjunto com TensorFlow, Hugging Face, Optuna e MLflow. O *notebook*[4] utilizado no treinamento deste modelo está disponível no Github. Utilizando o melhor modelo, foi obtida uma acurácia de 80.58% no conjunto de teste. As métricas obtidas no conjunto de teste estão representadas na TABELA 6. A matriz de confusão obtida no conjunto de teste está na FIGURA 10. Para obter este resultado foram necessárias 24 iterações utilizando Optuna com HyperbandPruner[5] e BruteForceSampler[6] sobre os parâmetros descritos na TABELA 7. Foram realizadas 6 iterações com cada possibilidade do conjunto de treinamento. Os 20 melhores resultados baseados no conjunto de validação estão presentes na FIGURA 13 nos apêndices. O melhor resultado foi obtido com um conjunto de treinamento com aumento de dados utilizando a técnica de parafraseamento.

Rótulos	Precision	Recall	F1-Score	Support
0 (Não Fake News)	0.89	0.70	0.78	842
1 (Fake News)	0.75	0.91	0.82	842

TABELA 6 – Métricas obtidas do melhor modelo BERT

[4] *Notebook* de treino BERT. Disponível em: <https://github.com/marcosgtavares/TCC2-FGA/blob/v1/notebooks/TCC_TRAIN_BERT.ipynb>

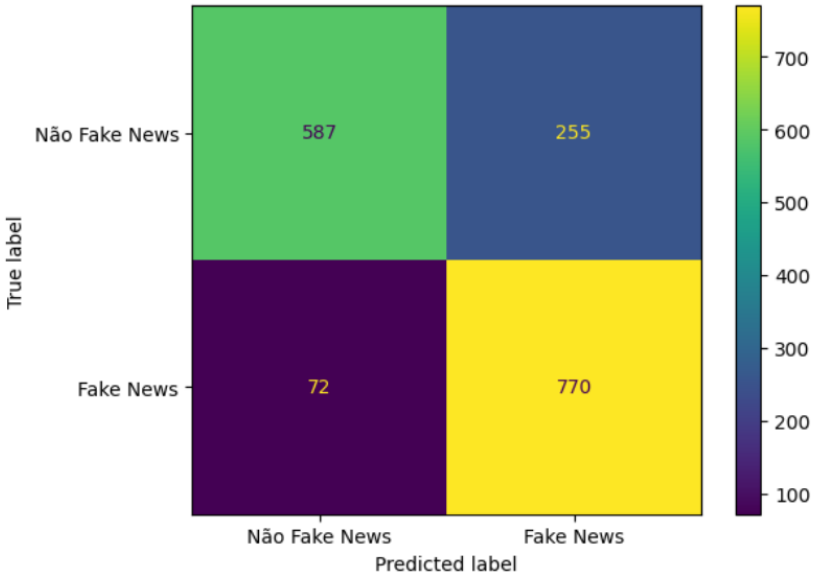
[5] Documentação do HyperbandPruner. Disponível em: <https://optuna.readthedocs.io/en/stable/reference/generated/optuna.pruners.HyperbandPruner.html>

[6] Documentação do BruteForceSampler. Disponível em: <https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.BruteForceSampler.html>

Parâmetros	Valores possíveis
Learning Rate	Float, 1×10^{-4}
Epsilon	Float, 1×10^{-8}
Tokens maximos	Int, entre 96 e 160 com passo de 32
Epochs	Int, 20 com *early stopping*
Batch size	Int, 16 e 24
Conjunto de treino	Cátegorico, com os valores possiveis sendo Dataset de treino sem aumento de dados, Dataset de treino com aumento de dados com a técnica *Backtranslation*, Dataset de treino com aumento de dados com a técnica Parafraseamento e Dataset de treino com aumento de dados com as técnicas de *Backtranslation* e Parafraseamento.

TABELA 7 – Hiperparâmetros sendo refinados com Optuna no modelo BERT

FIGURA 10 – Matriz de confusão do melhor modelo BERT



4.7.2 Treinar modelos CNN

No treinamento do modelo CNN foi-se utilizada a arquitetura TextConvoNet (SONI; CHOUHAN; RATHORE, 2022) em conjunto com TensorFlow, Optuna e MLflow. Os pesos pré-treinados dos *embeddings*[7] CBOW e Skip-Gram possuem 300 dimensões e são produtos do artigo (HARTMANN et al., 2017). O *notebook*[8] utilizado no treinamento deste modelo está disponível no Github. Utilizando o melhor modelo, foi obtida uma acurácia de 81.05% no conjunto de teste. As métricas obtidas no conjunto de teste estão

[7] Repositório de *Word Embeddings* do NILC. Disponível em: <http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc>

[8] *Notebook* de treino CNN. Disponível em: <https://github.com/marcosgtavares/TCC2-FGA/blob/v1/notebooks/TCC_TRAIN_CNN.ipynb>

representadas na TABELA 8. A matriz de confusão obtida no conjunto de teste está na FIGURA 11. Para obter este resultado foram necessárias 400 iterações utilizando Optuna com HyperbandPruner e TPESampler[9] sobre os parâmetros descritos na TABELA 9. Foram realizadas 50 iterações com cada combinação do conjunto de treinamento e *embeddings*. Os 20 melhores resultados baseados no conjunto de validação estão presentes na FIGURA 14 nos apêndices. O melhor resultado foi obtido com um conjunto de treinamento sem aumento de dados, inicializando a camada de *embeddings* do modelo com Skip-Gram.

Rótulos	Precision	Recall	F1-Score	Support
0 (Não Fake News)	0.82	0.79	0.81	842
1 (Fake News)	0.80	0.83	0.81	842

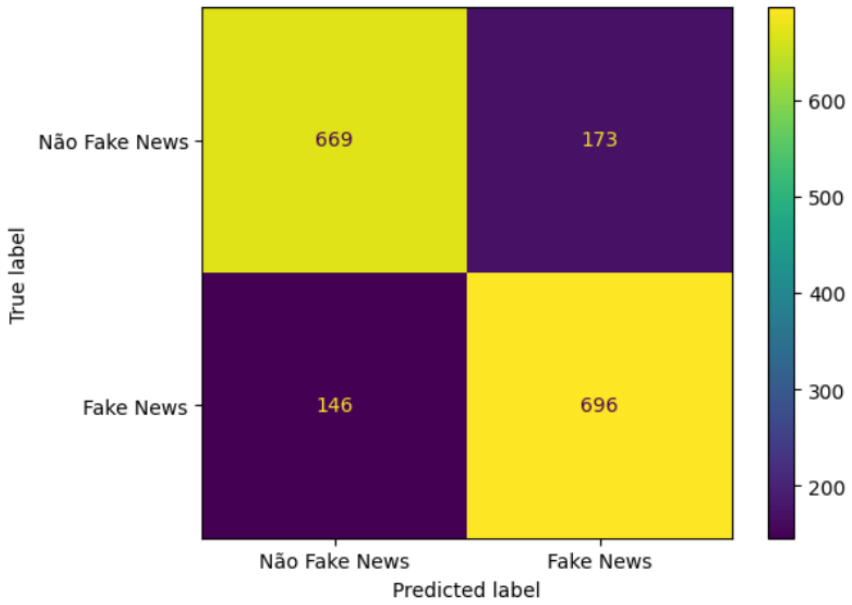
TABELA 8 – Métricas obtidas do melhor modelo CNN

Parâmetros	Valores possíveis
Learning Rate	Float, entre 1×10^{-6} e 1×10^{-3}
Epsilon	Float, entre 1×10^{-8} e 1×10^{-5}
Tokens maximos	Int, entre 600 e 3000 com passo de 600
Epochs	Int, 20 com *early stopping*
Batch size	Int, entre 8 e 64 com passo de 8
Word2Vec embeddings	Cátegorico, com os valores possíveis sendo CBOW e Skip-Gram
Conjunto de treino	Cátegorico, com os valores possiveis sendo Dataset de treino sem aumento de dados, Dataset de treino com aumento de dados com a técnica *Backtranslation*, Dataset de treino com aumento de dados com a técnica Parafraseamento e Dataset de treino com aumento de dados com as técnicas de *Backtranslation* e Parafraseamento.

TABELA 9 – Hiperparâmetros sendo refinados com Optuna no modelo CNN

[9]Documentação do TPESampler. Disponível em: <https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.TPESampler.html>

FIGURA 11 – Matriz de confusão do melhor modelo CNN



4.7.3 Treinar modelos LSTM

No treinamento do modelo LSTM foi-se utilizada a arquitetura descrita no guia do TensorFlow para classificação de texto com RNNs[10] em conjunto com TensorFlow, Optuna e MLflow. Os pesos pré-treinados dos *embeddings* CBOW e Skip-Gram possuem 300 dimensões e são produtos do artigo (HARTMANN et al., 2017). O *notebook*[11] utilizado no treinamento deste modelo está disponível no Github. Utilizando o melhor modelo, foi obtida uma acurácia de 76.54% no conjunto de teste. As métricas obtidas no conjunto de teste estão representadas na TABELA 10. A matriz de confusão obtida no conjunto de teste está na FIGURA 12. Para obter este resultado foram necessárias 400 iterações utilizando Optuna com HyperbandPruner e TPESampler sobre os parâmetros descritos na TABELA 11. Foram realizadas 50 iterações com cada combinação do conjunto de treinamento e *embeddings*. Os 20 melhores resultados baseados no conjunto de validação estão presentes na FIGURA 15 nos apêndices. O melhor resultado foi obtido com um conjunto de treinamento com aumento de dados utilizando as técnicas de *backtranslation* e parafraseamento, inicializando a camada de *embeddings* do modelo com CBOW.

Rótulos	Precision	Recall	F1-Score	Support
0 (Não Fake News)	0.75	0.79	0.77	842
1 (Fake News)	0.78	0.74	0.76	842

TABELA 10 – Matriz de confusão do melhor modelo LSTM

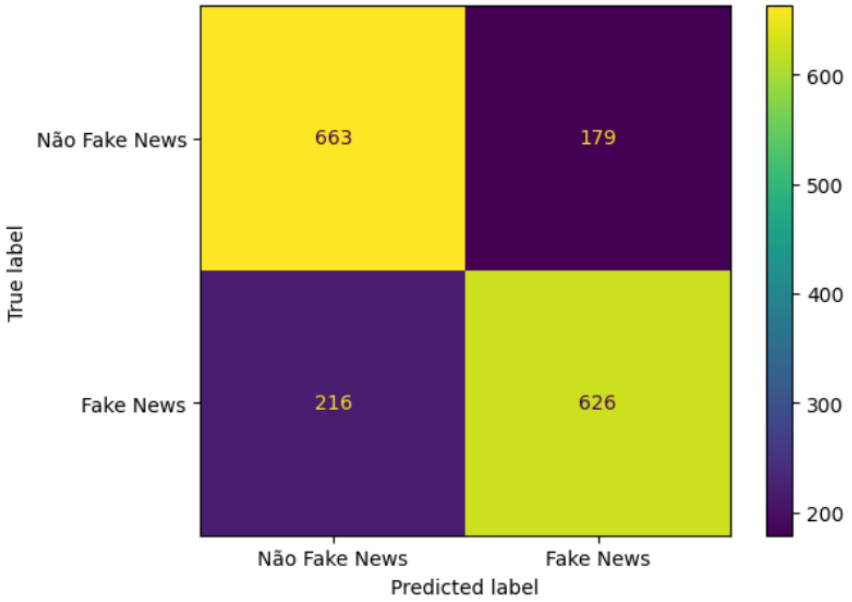
[10] Guia TensorFlow de classificação de texto com RNNs. Disponível em: <https://www.tensorflow.org/text/tutorials/text_classification_rnn>

[11] *Notebook* de treino LSTM. Disponível em: <https://github.com/marcosgtavares/TCC2-FGA/blob/v1/notebooks/TCC_TRAIN_LSTM.ipynb>

Parâmetros	Valores possíveis
Learning Rate	Float, entre 1×10^{-5} e 1×10^{-2}
Epsilon	Float, entre 1×10^{-8} e 1×10^{-5}
Tokens maximos	Int, entre 64 e 512 com passo de 64
Epochs	Int, 20 com *early stopping*
Batch size	Int, entre 8 e 64 com passo de 8
Word2Vec embeddings	Cátegorico, com os valores possíveis sendo CBOW e Skip-Gram
Conjunto de treino	Cátegorico, com os valores possiveis sendo Dataset de treino sem aumento de dados, Dataset de treino com aumento de dados com a técnica *Backtranslation*, Dataset de treino com aumento de dados com a técnica Parafraseamento e Dataset de treino com aumento de dados com as técnicas de *Backtranslation* e Parafraseamento.

TABELA 11 – Hiperparâmetros sendo refinados com Optuna no modelo LSTM

FIGURA 12 – Matriz de confusão do melhor modelo LSTM



4.8 Realizar comparação quantitativa dos resultados dos modelos

A natureza dos dados de Fake News exige uma análise cuidadosa dos resultados dos modelos de classificação, pois tanto falsos positivos quanto falsos negativos podem ter consequências prejudiciais. Considerando que o conjunto de teste é balanceado, com igual número de itens em cada classe, a acurácia se torna uma métrica relevante para avaliar o desempenho dos modelos. É crucial ressaltar que os modelos aqui treinados podem apresentar vieses decorrentes dos dados utilizados no treinamento, onde a natureza política de diversos itens foi observada. Também é válido mencionar o amplo espectro do que

conhecemos como Fake News, que pode incluir vários subtipos, tais como as Fake News de natureza política, as relacionadas a saúde, as de caráter científico, entre outras. Portanto, os resultados gerados pelos modelos produzidos neste trabalho devem ser interpretados com cautela e utilizados apenas como referência.

O modelo LSTM apresentou, em geral, resultados inferiores aos demais, com valores de métricas iguais ou inferiores aos dos outros modelos. O modelo BERT, apesar de ter acurácia inferior ao modelo CNN, obteve a melhor *precision* para Não Fake News e o melhor *recall* para Fake News, indicando uma alta taxa de falsos positivos e verdadeiros positivos, respectivamente. Já o modelo CNN apresentou a melhor acurácia geral e, de acordo com a natureza dos dados discutida anteriormente, parece ser o modelo superior entre os modelos treinados neste trabalho. A TABELA 12 permite comparar diretamente os resultados de forma mais acessível com a média das métricas obtidas no conjunto de teste.

Métricas	BERT	CNN	LSTM
Precision	0.82	0.81	0.77
Recall	0.81	0.81	0.77
F1-Score	0.80	0.81	0.77
Acurácia	80.58%	81.05%	76.54%

TABELA 12 – Média das métricas de todos os modelos (BERT, CNN, LSTM)

4.9 Conclusão e discussão quanto aos resultados do trabalho

Este trabalho foi elaborado com o objetivo de treinar múltiplos modelos de classificação supervisionada de várias arquiteturas acerca do tópico de classificação de *Fake News* para que fosse possível comparar os resultados dos modelos. O projeto foi construído de forma que fossem utilizadas várias tecnologias a fim de experimentar com PLN no contexto de *deep learning*. Ele foi executado de acordo com uma metodologia detalhada nos métodos e materiais.

No inicio, foi-se utilizado apenas um dos conjuntos de dados apresentado, sendo ele o **FakeWhatsApp.BR**. Testes iniciais levaram a realização de que os dados apesar de abundantes ainda eram insuficientes para os motivos dos trabalho. Após isso, foram encontrados outros três conjuntos de dados que atendiam os requisitos do projeto. Para mitigar ainda mais o problema inicial, também foi decidido utilizar técnicas de aumento de dados para incrementar ainda mais a base a ser utilizada. Essas técnicas foram escolhidas de forma a utilizar outros modelos de *deep learning*.

Como a iteração manual dos hiperparâmetros presenciada no estado inicial do projeto levou a resultados não ótimos e difíceis de reproduzir, foi introduzido o uso de uma ferramenta de otimização de hiperparâmetros (Optuna). O aumento da complexidade

neste estágio levou o uso de outra ferramenta para gerenciar o treinamento de diversos modelos (MLflow).

Devido as demandas computacionais consequentes das mudanças introduzidas, o projeto que antes estava sendo progredido nos *notebooks* do Google Colab passou a ser executado localmente com *Jupyter*. Essa mudança, embora necessária, apresentou o desafio de adaptar o ambiente para as tarefas de manipulação de dados e treino a seguir.

Algumas das ideias inicias foram sujeitas de falha e elas foram refinadas ao longo do projeto. Novos técnicas também foram introduzidas para mitigar problemas encontrados durante o desenvolvimento. No fim, o projeto foi um processo iterativo de melhoria e aprendizado. Apesar das dificuldade encontradas durante a jornada de desenvolvimento, o projeto foi capaz de alcançar seus objetivos de obter modelos de *deep learning* capazes de classificar *Fake News* com elementos textuais e comparar seus resultados. Os obstáculos encontrados durante o progresso do projeto apenas incrementaram ao seu conteúdo e aumentaram a qualidade de seu resultado.

Referências

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado na página 30.

AI@META. Llama 3 model card. 2024. Disponível em: <https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md>. Citado na página 37.

AKIBA, T. et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. arXiv, 2019. ArXiv:1907.10902 [cs, stat]. Disponível em: <http://arxiv.org/abs/1907.10902>. Citado 2 vezes nas páginas 25 e 30.

BIRD, S.; KLEIN, E.; LOPER, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. [S.l.]: "O'Reilly Media, Inc.", 2009. Citado 2 vezes nas páginas 15 e 16.

BISCHL, B. et al. *Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges*. arXiv, 2021. ArXiv:2107.05847 [cs, stat]. Disponível em: <http://arxiv.org/abs/2107.05847>. Citado na página 22.

BISONG, E. Google colaboratory. In: _____. *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, 2019. p. 59–64. ISBN 978-1-4842-4470-8. Disponível em: <https://doi.org/10.1007/978-1-4842-4470-8_7>. Citado na página 27.

CHARLES, A. C.; RUBACK, L.; OLIVEIRA, J. Fakepedia Corpus: A Flexible Fake News Corpus in Portuguese. In: PINHEIRO, V. et al. (Ed.). *Computational Processing of the Portuguese Language*. Cham: Springer International Publishing, 2022. v. 13208, p. 37–45. ISBN 978-3-030-98304-8 978-3-030-98305-5. Series Title: Lecture Notes in Computer Science. Disponível em: <https://link.springer.com/10.1007/978-3-030-98305-5_4>. Citado na página 32.

CHAVARRO, J. et al. Faketruebr: Um corpus brasileiro de notícias falsas. In: *Anais da XVIII Escola Regional de Banco de Dados*. Porto Alegre, RS, Brasil: SBC, 2023. p. 108–117. ISSN 2595-413X. Disponível em: <https://sol.sbc.org.br/index.php/erbd/article/view/24352>. Citado na página 32.

CHOLLET, F. *Keras*. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>. Citado na página 30.

CUNHA, L. C. C. d. FakeWhatsApp.BR: Detecção de Desinformação e Desinformadores em Grupos Públicos do WhatsApp em PT-BR. *130 f. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal do Ceará, Fortaleza*, 2021. Disponível em: <http://www.repositorio.ufc.br/handle/riufc/63379>. Citado na página 32.

DAHL, G. E. et al. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, v. 20, n. 1, p. 30–42, jan. 2012. ISSN 1558-7916, 1558-7924.

Disponível em: <http://ieeexplore.ieee.org/document/5740583/>. Citado 2 vezes nas páginas 19 e 20.

DEVLIN, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv, 2019. ArXiv:1810.04805 [cs]. Disponível em: <http://arxiv.org/abs/1810.04805>. Citado na página 21.

FACE, H. *NLP Course*. 2021. Disponível em: <https://huggingface.co/course/pt/chapter1/2?fw=pt>. Citado na página 15.

FENG, S. Y. et al. *A Survey of Data Augmentation Approaches for NLP*. arXiv, 2021. ArXiv:2105.03075 [cs]. Disponível em: <http://arxiv.org/abs/2105.03075>. Citado na página 21.

GitHub: Let's build from here. 2023. Disponível em: <https://github.com/>. Citado na página 27.

GODBOLE, S.; SARAWAGI, S. Discriminative Methods for Multi-labeled Classification. In: KANADE, T. et al. (Ed.). *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. v. 3056, p. 22–30. ISBN 978-3-540-22064-0 978-3-540-24775-3. Series Title: Lecture Notes in Computer Science. Disponível em: <http://link.springer.com/10.1007/978-3-540-24775-3_5>. Citado na página 16.

HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>. Citado na página 29.

HARTMANN, N. et al. *Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks*. arXiv, 2017. ArXiv:1708.06025 [cs]. Disponível em: <http://arxiv.org/abs/1708.06025>. Citado 2 vezes nas páginas 39 e 41.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007. Citado na página 29.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. (Ed.). *Automated Machine Learning: Methods, Systems, Challenges*. Cham: Springer International Publishing, 2019. (The Springer Series on Challenges in Machine Learning). ISBN 978-3-030-05317-8 978-3-030-05318-5. Disponível em: <http://link.springer.com/10.1007/978-3-030-05318-5>. Citado na página 22.

KALMUKOV, Y. USING WORD CLOUDS FOR FAST IDENTIFICATION OF PAPERS' SUBJECT DOMAIN AND REVIEWERS' COMPETENCES15. v. 60, 2021. Citado na página 17.

KHURANA, D. et al. Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, v. 82, n. 3, p. 3713–3744, jan. 2023. ISSN 1573-7721. Disponível em: <https://doi.org/10.1007/s11042-022-13428-4>. Citado na página 15.

Kluyver, T. et al. Jupyter Notebooks—a publishing format for reproducible computational workflows. In: *IOS Press*. [S.l.: s.n.], 2016. p. 87–90. Citado na página 28.

LEE, D.; Huei Chuang; SEAMONS, K. Document ranking and the vector-space model. *IEEE Software*, v. 14, n. 2, p. 67–75, abr. 1997. ISSN 07407459. Disponível em: <http://ieeexplore.ieee.org/document/582976/>. Citado na página 18.

LI, Z. et al. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, v. 33, n. 12, p. 6999–7019, dez. 2022. ISSN 2162-237X, 2162-2388. Disponível em: <https://ieeexplore.ieee.org/document/9451544/>. Citado na página 20.

MANNING, C.; RAGHAVAN, P.; SCHUETZE, H. Introduction to Information Retrieval. 2009. Citado na página 17.

MIKOLOV, T. et al. *Efficient Estimation of Word Representations in Vector Space*. arXiv, 2013. ArXiv:1301.3781 [cs]. Disponível em: <http://arxiv.org/abs/1301.3781>. Citado 2 vezes nas páginas 17 e 18.

MORAIS, J. I. D. et al. Deciding among Fake, Satirical, Objective and Legitimate news: A multi-label classification system. In: *Proceedings of the XV Brazilian Symposium on Information Systems*. Aracaju Brazil: ACM, 2019. p. 1–8. ISBN 978-1-4503-7237-4. Disponível em: <https://dl.acm.org/doi/10.1145/3330204.3330231>. Citado na página 32.

NETO, M. et al. FAKE NEWS NO CENÁRIO DA PANDEMIA DE COVID-19. *Cogitare Enfermagem*, v. 25, abr. 2020. ISSN 2176-9133, 1414-8536. Disponível em: <https://revistas.ufpr.br/cogitare/article/view/72627>. Citado na página 13.

OLLAMA. 2024. Disponível em: <https://ollama.com>. Citado na página 28.

O'SHEA, K.; NASH, R. *An Introduction to Convolutional Neural Networks*. arXiv, 2015. ArXiv:1511.08458 [cs]. Disponível em: <http://arxiv.org/abs/1511.08458>. Citado na página 20.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 2 vezes nas páginas 18 e 30.

SONI, S.; CHOUHAN, S. S.; RATHORE, S. S. *TextConvoNet:A Convolutional Neural Network based Architecture for Text Classification*. arXiv, 2022. ArXiv:2203.05173 [cs]. Disponível em: <http://arxiv.org/abs/2203.05173>. Citado na página 39.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: pretrained BERT models for Brazilian Portuguese. In: *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*. [S.l.: s.n.], 2020. Citado na página 38.

TANG, Y. et al. Multilingual translation with extensible multilingual pretraining and finetuning. 2020. Citado na página 37.

TEAM, T. pandas development. *pandas-dev/pandas: Pandas*. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>. Citado na página 28.

WASKOM, M. L. seaborn: statistical data visualization. *Journal of Open Source Software*, The Open Journal, v. 6, n. 60, p. 3021, 2021. Disponível em: <https://doi.org/10.21105/joss.03021>. Citado na página 29.

WOLF, T. et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. Citado na página 27.

WORDCLOUD · PyPI. 2023. Disponível em: <https://pypi.org/project/wordcloud/>. Citado na página 29.

ZAHARIA, M. et al. Accelerating the Machine Learning Lifecycle with MLflow. In: . Santa Clara, CA: USENIX Association, 2019. Citado 2 vezes nas páginas 25 e 31.

Apêndices

FIGURA 13 – Os 20 melhores resultados da otimização de hiperparâmetros no modelo BERT

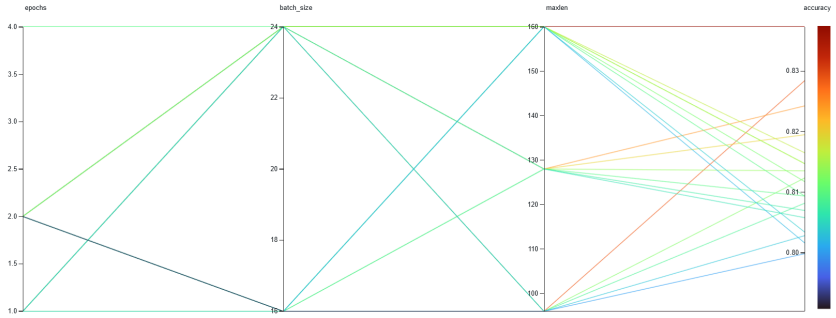


FIGURA 14 – Os 20 melhores resultados da otimização de hiperparâmetros no modelo CNN

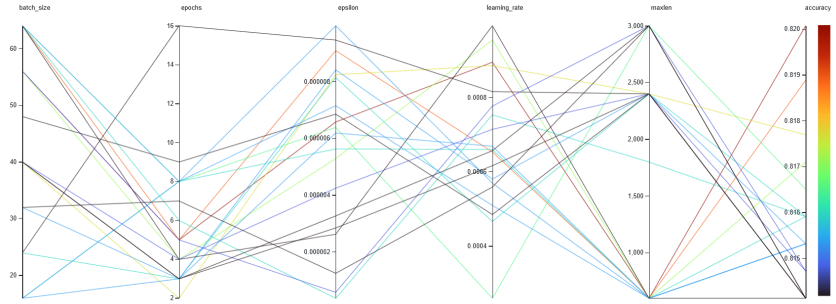


FIGURA 15 – Os 20 melhores resultados da otimização de hiperparâmetros no modelo LSTM