UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LARISSA MAYUMI BARELA HONDO

DETECÇÃO DE *FAKE NEWS* EM PORTUGUÊS UTILIZANDO INFERÊNCIA DE LINGUAGEM NATURAL

BAURU
NOVEMBRO/2023

LARISSA MAYUMI BARELA HONDO

DETECÇÃO DE *FAKE NEWS* EM PORTUGUÊS UTILIZANDO INFERÊNCIA DE LINGUAGEM NATURAL

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Assoc. Aparecido Nilceu Marana

BAURU
NOVEMBRO/2023

Larissa Mayumi Barela Hondo

Detecção de *Fake News* em Português Utilizando Inferência de Linguagem Natural

Banca Examinadora

Prof. Assoc. Aparecido Nilceu Marana
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profª. Drª. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Profª. Drª. Andrea Carla Gonçalves Vianna
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Bauru, 14 de Novembro de 2023.

Agradecimentos

Agradeço ao professor Aparecido Nilceu Marana, à UNESP e à minha família pelo apoio à realização deste trabalho.

Resumo

Com o advento da *Internet*, é possível obter informações a partir de quaisquer fontes sobre eventos ao redor do mundo, inclusive informações imprecisas e até mesmo falsas, sendo estas chamadas de *fake news*. A divulgação e a proliferação de *fake news* podem causar prejuízos e danos seríssimos para as pessoas e para a sociedade de modo geral, salientando a extrema importância de sua identificação. Isso pode ser feito com o uso do Processamento de Linguagem Natural, que abrange técnicas computacionais para a análise automática e à representação de linguagens antropológicas. Por exemplo, a detecção de *fake news* pode ser feita com a Inferência de Linguagem Natural, uma sub-área do Processamento de Linguagem Natural focada na verificação da implicação ou contradição em um par de sentenças, que pode ser utilizada neste contexto considerando notícias já confirmadas como sendo verdadeiras como a premissa e uma notícia suspeita como a hipótese, dessa forma, caso haja contradição entre a hipótese e a premissa, a notícia suspeita é considerada falsa, caso contrário, ela é rotulada como verdadeira. Visto isso, o objetivo deste trabalho é propor um método de detecção automática de *fake news* para a língua portuguesa utilizando a Inferência de Linguagem Natural. Como não há uma base de dados em português que utilize a Inferência de Linguagem Natural para esta aplicação, é possível traduzir dados já existentes para este idioma, em vista disso, neste trabalho utilizou-se a base de dados em inglês denominada *FNID-FakeNewsNet*. Com base nos resultados obtidos para o conjunto de teste, observa-se um desempenho melhor quando a premissa é utilizada junto à hipótese. Além disso, de modo geral, é possível observar que os resultados não foram afetados significativamente pela tradução dos dados em inglês para o português.

Palavras-chave: *Fake news*. Inferência de Linguagem Natural. Processamento de Linguagem Natural.

Abstract

With the Internet, it is possible to obtain news about global events regardless of the source, which means that inaccurate and even false information, called fake news, is shared. This dissemination and proliferation of fake news can cause harm and damage to people and society, highlighting the extreme importance of their identification. One way to approach this problem is by using Natural Language Processing, which encompasses computational techniques for the automatic analysis and representation of anthropological languages. For example, the detection of fake news is possible with Natural Language Inference, a sub-area of Natural Language Processing focused on checking the implication or contradiction in a pair of sentences, which can be used in this context considering reliable news as the premise and suspicious news as the hypothesis, thus, if there is a contradiction between them, the investigated news is considered false, otherwise it is labeled as true. Given this, the objective of this work is to propose a method for the automatic detection of fake news in the Portuguese language using Natural Language Inference. As no database in Portuguese uses Natural Language Inference for this application, it is possible to translate existing data into this language. Therefore, in this work, we used the database in English called FNID-FakeNewsNet. Based on the results obtained from the test set, better performance occurs with the simultaneous use of premise and hypothesis. Furthermore, it is also possible to observe that the results were not significantly affected by the translation from English to Portuguese.

Keywords: Fake news. Natural Language Inference. Natural Language Processing.

Lista de figuras

Lista de quadros

Lista de abreviaturas e siglas

API	*Application Programming Interface*
BERT	*Bidirectional Encoder Representations from Transformers*
BiLSTM	*Bidirectional Long Short-Term Memory*
BoW	*Bag of Words*
brWaC	*Brazilian Portuguese Web as Corpus*
CNN	*Convolutional Neural Network*
FN	*False Negative*
FP	*False Positive*
IA	Inteligência Artificial
IDF	*Inverse Document Frequency*
ILN	Inferência de Linguagem Natural
LR	*Logistic Regression*
LSTM	*Long Short-Term Memory*
mBERT	BERT multilíngue
NB	Naive Bayes
NLTK	*Natural Language Toolkit*
PLN	Processamento de Linguagem Natural
RF	*Random Forest*
RNN	*Recurrent Neural Network*
SVC	*C-Support Vector Classification*
SVM	*Support Vector Machine*
TI	Tecnologia da Informação
TF	*Term Frequency*
TN	*True Negative*
TP	*True Positive*

Sumário

1 Introdução

As notícias e informações são as ferramentas e bases da conscientização e das ações da sociedade e, embora sua transmissão seja tradicionalmente por agências de notícias, atualmente, na era da informação, com o rápido crescimento e atratividade das mídias sociais *online*, tais como redes sociais, aplicativos de mensagens e *blogs*, tem-se uma quantidade significativa de notícias transmitidas e divulgadas nessas plataformas, abrangendo diversos assuntos, como políticos, sociais, econômicos, de saúde e artísticos (SADEGHI; BIDGOLY; AMIRKHANI, 2022). Com isso, é possível obter informações a partir de quaisquer fontes sobre eventos ao redor do mundo, inclusive informações imprecisas e até mesmo falsas, sendo estas chamadas de *fake news*.

A divulgação e a proliferação de *fake news* podem causar prejuízos e danos seríssimos para as pessoas e para a sociedade de modo geral, o que torna de extrema importância a sua detecção (SABARMATHI; GOWTHAMI; KUMAR, 2021), que pode ser realizada, atualmente, de forma automática, por meio de técnicas de Processamento de Linguagem Natural (PLN).

O PLN é uma sub-área da Inteligência Artificial (IA) que ajuda a máquina a entender a linguagem natural em forma de texto ou fala, sendo uma coleção de técnicas computacionais para análise automática e representação de linguagens antropológicas, como recuperação de informações, obtenção de respostas às perguntas, extração de informação de textos, tradução de linguagem natural, resumos de textos, modelagem de tópicos e mineração de textos (CHOWDHARY, 2020). Além disso, o PLN é multidisciplinar, incluindo áreas como a Computação Linguística, o Aprendizado de Máquina, o Aprendizado Profundo e Estatísticas, e auxilia especialmente na eficiência pela automatização de processos que requerem revisão ou análise textuais. Ademais, o PLN também inclui a Inferência de Linguagem Natural (ILN) (do inglês, *Natural Language Inference*), que consiste em detectar implicações ou contradições em um par de sentenças, sendo importante especialmente em tarefas de recuperação de informações (KIM; JANG; ALLAN, 2020). A inferência tem sido um tópico central na IA desde seu início, obtendo progresso no desenvolvimento de métodos automáticos para a dedução formal (MACCARTNEY, 2009).

O estado-da-arte da ILN inclui métodos baseados em Aprendizado Profundo com o intuito de extrair relações de inferência (de implicação, de contradição ou de estado neutro) entre os textos, considerando um texto como premissa (p) que leva a um texto hipótese (h), $p \vdash h$, como na base de dados *Stanford Natural Language Inference* de Bowman et al. (2015).

Já no estado-da-arte da detecção de *fake news* com ILN, geralmente uma notícia confiável é considerada como uma premissa, e, a partir dela, uma notícia suspeita, a hipótese, é comparada, sendo que, se a hipótese contradizer a premissa, então a notícia suspeita é falsa,

senão é verdadeira. Contudo, os estudos sobre a detecção de *fake news* por ILN são limitados a algumas linguagens, como a chinesa no caso de Yang, Niven e Kao (2019) e a inglesa no de Sabarmathi, Gowthami e Kumar (2021).

1.1 Problema

A publicação de notícias de maneira não profissional, sem supervisão e sem verificação pode acarretar a divulgação de informações falsas intencional ou acidentalmente. Portanto, reconhecer a veracidade das notícias tornou-se um desafio nas mídias sociais *online*, já que indivíduos e organizações podem espalhar *fake news* para fins lucrativos, competitivos ou de entretenimento, pois costumam ser mais interessantes do que as notícias verdadeiras (SADEGHI; BIDGOLY; AMIRKHANI, 2022). As *fake news* podem causar danos irreparáveis a indivíduos, organizações e governos, acarretando efeitos devastadores, como aumento da ansiedade social, redução da produtividade de uma empresa e paralisação do ciclo econômico (SADEGHI; BIDGOLY; AMIRKHANI, 2022). Além disso, não há muitos trabalhos utilizando a ILN aplicada à apuração de *fake news* como uma alternativa à análise das notícias por seu conteúdo e local onde foram publicadas, especialmente na língua portuguesa.

1.2 Justificativa

A solução para o problema apresentado (detecção de *fake news*) possui relevância dentro da área técnica e acadêmica, já que um dos métodos de se detectar *fake news* consiste em utilizar ILN, imitando a maneira como os especialistas analisam as informações, ou seja, tentando encontrar uma contradição ou implicação entre uma determinada notícia e outras existentes (SADEGHI; BIDGOLY; AMIRKHANI, 2022). Dessa forma, se uma notícia nova contradiz notícias já confirmadas como sendo verdadeiras, ela é considerada falsa. Caso contrário, se ela concordar com as informações das notícias confiáveis, ela é rotulada como verdadeira (SADEGHI; BIDGOLY; AMIRKHANI, 2022).

1.3 Objetivos

A seguir, os objetivos geral e específicos serão apresentados.

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é propor um método de detecção automática de *fake news* na língua portuguesa utilizando ILN. Neste método, dados dois textos de entrada, um sendo uma notícia já confirmada como verdadeira e outro sendo uma notícia suspeita, é retornada uma relação de inferência a fim de auxiliar a identificação de *fake news* com o uso

de ILN, de tal forma que o texto confiável é considerado como premissa e o texto suspeito é considerado como hipótese, assim, se a hipótese estabelecer uma contradição com a premissa, a notícia suspeita é considerada falsa, senão é considerada verdadeira.

1.3.2 Objetivo Específico

Os objetivos específicos incluem:

1. Traduzir a base de dados de *fake news* escolhida para o português;
2. Realizar um estudo sobre o conceito e técnicas de PLN, especialmente aplicados em tarefas de ILN;
3. Fazer uma revisão da literatura e estudo acerca de métodos já propostos para a classificação de *fake news* com o uso de ILN, verificando os resultados obtidos;
4. Implementar modelos para a detecção de *fake news* com e sem o uso de ILN e comparar seus resultados.

1.4 Organização do Trabalho

Os próximos capítulos abrangem os seguintes pontos:

Capítulo 2: Apresentação da fundamentação teórica que foi utilizada para o desenvolvimento deste trabalho;

Capítulo 3: Apresentação de trabalhos correlatos já realizados;

Capítulo 4: Explicação da Metodologia adotada no decorrer do desenvolvimento;

Capítulo 5: Apresentação e discussão dos resultados obtidos;

Capítulo 6: Considerações finais sobre o trabalho desenvolvido.

2 Fundamentação Teórica

Neste capítulo são apresentados os conhecimentos teóricos necessários para o entendimento do trabalho proposto, bem como as ferramentas utilizadas para o seu desenvolvimento.

2.1 *Fake News*

As *fake news* são artigos fictícios fabricados deliberadamente, que podem enganar os leitores, e já existiam antes mesmo da *Internet* (ALDWAIRI; ALWAHEDI, 2018), sendo que eram transmitidas pela tradição oral, em forma de rumores ou comentários acerca de outras pessoas ou de organizações rivais (MONTEIRO et al., 2018).

Atualmente, as mídias sociais se tornaram os principais meios para o compartilhamento e consumo de notícias, todavia, a qualidade das notícias compartilhadas frequentemente é menor comparada a de fontes tradicionais de notícias, por conta das *fake news* (MONTEIRO et al., 2018).

Esse problema é agravado pelo fato de que as *fake news* podem até mesmo influenciar às tomadas de decisão e os comportamentos interpessoais em várias situações, desde a área de saúde, revelando medicações "milagrosas", até as de política e economia, como no escândalo de dados *Facebook–Cambridge Analytica* (MONTEIRO et al., 2018), já que as redes sociais e os meios de comunicação publicam notícias falsas para aumentar o número de leitores (ALDWAIRI; ALWAHEDI, 2018).

Além das *fake news* serem um problema relacionado ao *marketing* e às relações públicas, também são consideradas cada vez mais como parte das responsabilidades associadas às tecnologias da informação (TI) (ALDWAIRI; ALWAHEDI, 2018).

Logo, vários esforços foram feitos para mitigar este problema, desenvolvendo inúmeras pesquisas acerca da detecção de *fake news* (MONTEIRO et al., 2018). Assim, organizações para a checagem de notícias foram criadas, como a Agência Lupa e Boatos.org, e pesquisadores também uniram esforços ao estudar como as notícias se espalham, o comportamento dos usuários que as produzem e leem as *fake news*, e como as características linguísticas são empregadas em seu uso. Para o último caso mencionado, emprega-se principalmente o uso de PLN (MONTEIRO et al., 2018).

2.2 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) consiste em uma coleção de técnicas computacionais para análise automática e representação de linguagens antropológicas

(CHOWDHARY, 2020). Alguns exemplos de aplicações de PLN são: recuperação de informações, obtenção de respostas às perguntas, extração de informação de textos, tradução de linguagem natural, resumos de textos, modelagem de tópicos e mineração de textos (CHOWDHARY, 2020).

Atualmente, novos modelos computacionais tentam emular os processos do cérebro humano para o processamento de linguagem, com abordagens que dependem de características semânticas que não podem ser explicitamente expressas por meio do texto (CHOWDHARY, 2020). Esses modelos computacionais são úteis para fins teóricos e estudos científicos, como explorar a natureza da comunicação linguística e suas propriedades, e também para aplicações práticas e industriais (CHOWDHARY, 2020).

2.2.1 Inferência de Linguagem Natural

Os humanos usam uma variedade de conhecimentos e raciocínios para auxiliar o entendimento dos significados da linguagem, como, por exemplo, por Inferência de Linguagem Natural (ILN) (STORKS; GAO; CHAI, 2019). Baseado nisso, segundo Bowman et al. (2015), ILN é a tarefa de determinar se uma "hipótese" é verdadeira (implicação), falsa (contradição) ou indeterminada (neutra), quando comparada a uma "premissa", por exemplo:

- Uma contradição seria: "Um homem inspeciona o uniforme de uma figura em algum país do Leste Asiático. $\vdash$ Um homem está dormindo."
- Uma relação neutra seria: "Um homem mais velho e mais jovem sorrindo. $\vdash$ Dois homens sorriem e riem dos gatos brincando no chão."
- Uma implicação seria: "Um jogo de futebol com vários homens jogando. $\vdash$ Alguns homens estão praticando um esporte."

Compreender se uma hipótese é uma implicação ou uma contradição de uma premissa é fundamental para a compreensão da linguagem natural, já que sua inferência é valiosa para o desenvolvimento de representações semânticas (BOWMAN et al., 2015).

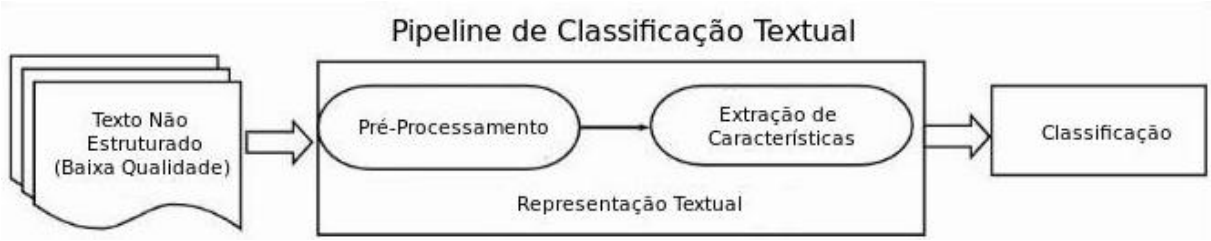
Para apoiar a capacidade de inferência das máquinas, conhecimentos auxiliares se tornam importantes, pois é preciso compreender além do que é explicitamente expresso, baseando-se em informações inferidas a partir de informações sobre como o mundo funciona (STORKS; GAO; CHAI, 2019). Por exemplo, ao falar que "um objeto X não cabe em um Y porque *ele* é muito grande", somente com as análises linguísticas não se consegue saber o que é muito grande, precisa-se de um conhecimento externo, isto é, saber que um objeto deve ser maior que outro objeto para contê-lo, concluindo, assim, que *ele* se associa a X e não a Y.

Com isso, a resolução bem-sucedida de tarefas como respostas às perguntas exige que as máquinas ultrapassem o contexto linguístico e dependam de raciocínio e conhecimento que não estão explicitamente declarados no texto (STORKS; GAO; CHAI, 2019).

2.2.2 Representação de Palavras

Como pode ser observado na Figura 1, para a realização da classificação textual pela máquina, o texto deve ser convertido em um formato que o computador possa entender, já que o texto não estruturado é a forma do texto bruto, disperso e esparso. Normalmente, são as informações geradas pelos usuários em postagens, documentos, *e-mails* ou mensagens nas redes sociais (NASEEM et al., 2020).

Figura 1 – *Pipeline* da classificação textual.



Fonte: Adaptada de Naseem et al. (2020).

2.2.2.1 Pré-processamento Textual

Alguns dos principais processos, segundo Naseem et al. (2020), são:

- Tokenização, que consiste em separar o texto em pequenos pedaços, *tokens*;
- Remoção de pontuações, como "!" e "?", e números, tornando o texto mais simples;
- Conversão de todos os caracteres para minúsculos, já que diversidade de letras maiúsculas dentro do corpus pode causar problemas durante a tarefa de classificação, diminuindo seu o desempenho;
- Remoção de *stop words*, palavras que não acrescentam uma informação crítica ao texto, como "somos", "as", "os", "fosse", "suas', "seus", "uma", "este", etc.
- Lematização para obter o lema de cada palavra, por exemplo, substantivos femininos são transformados em masculinos, palavras no plural são colocadas no singular e verbos são conjugados no infinitivo.

2.2.2.2 Extração de Características

A análise de texto é uma aplicação importante para os algoritmos de aprendizado de máquina e isso é feito com a extração de características, permitindo que bases de dados contendo textos sejam transformadas em formatos numéricos e de tamanho fixo suportados por algoritmos de aprendizado de máquina, já que os dados brutos, sendo sequências de símbolos com tamanhos variáveis, não são esperados (Scikit-Learn, 2023). Normalmente, esta etapa de extração de recursos para transformar dados brutos é chamada de vetorização (NASEEM et al., 2020).

2.2.2.2.1 *Bag of Words*

Uma das formas mais comuns de realizar essa extração consiste em utilizar o *Bag of Words* (BoW), contando as ocorrências dos *tokens* em cada documento analisado, considerando cada frequência individual como uma característica (Scikit-Learn, 2023).

Dessa forma, ocorre o processo de vetorização, transformando uma coleção de documentos de texto em vetores de características numéricas. Os documentos são descritos por ocorrências de palavras, ignorando completamente as informações de posição relativa das palavras no documento (Scikit-Learn, 2023). Um corpus de documentos pode, portanto, ser representado por uma matriz com uma linha por documento e uma coluna por *token* (Scikit-Learn, 2023).

Observa-se que, como a maioria dos documentos normalmente usa um subconjunto muito pequeno de palavras, a matriz resultante terá muitos valores de características que são zeros (Scikit-Learn, 2023).

2.2.2.2.2 *Term Frequency - Inverse Document Frequency*

Algumas palavras estarão muito presentes, transportando muito pouca informação significativa sobre o conteúdo real do documento, obscurecendo as frequências de termos mais raros, porém mais importantes (Scikit-Learn, 2023).

Para isso, a Frequência do Termo, do inglês *Term Frequency* (TF), e a Frequência Inversa do Documento, do inglês *Inverse Document Frequency* (IDF), analisam o quão comum (ou incomum) um *token* é no *corpus*, porém não carregam o significado semântico das relações entre os *tokens* (Scikit-Learn, 2023).

O TF-IDF é calculado pela Equação 2.1, com $TF(t, d)$ representando a TF sendo analisada (Scikit-Learn, 2023).

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t) \tag{2.1}$$

A IDF é calculada pela Equação 2.2, onde n é o número total de documentos no conjunto de documentos e $DF(t)$ é o número de documentos no conjunto de documentos que contêm o termo t, sendo que a importância de um termo é inversamente relacionada à sua frequência nos documentos (Scikit-Learn, 2023).

$$\text{IDF}(t) = \log \frac{n}{1 + \text{DF}(t)}. \tag{2.2}$$

2.2.3 Mecanismo de Atenção

À medida que os modelos de aprendizagem profunda se tornam mais sofisticados, a necessidade de métodos eficazes de processamento de grandes quantidades de dados tornou-se cada vez mais importante. Um desses métodos é o mecanismo de atenção, que permite que um modelo se concentre nas informações mais relevantes ao fazer previsões (BAHDANAU; CHO; BENGIO, 2014).

O mecanismo de atenção foi introduzido em 2014 por Bahdanau, Cho e Bengio (2014), com a arquitetura *sequence-to-sequence* (seq2seq), e pertence à família dos codificadores-decodificadores, do inglês *encoders-decoders*, sendo que o *encoder* lê e codifica a frase original em um vetor de tamanho fixo, enquanto que o *decoder* produz a tradução desse vetor codificado.

Com o mecanismo de atenção, em cada passo do *decoder*, um pedaço específico da sequência inicial recebe foco, assim, procura-se por um conjunto de palavras de entrada ao gerar cada palavra de destino, não precisando codificar cada frase original inteiramente em um vetor de comprimento fixo, dando atenção apenas na informação relevante para a próxima palavra-alvo, melhorando a tradução de sequências longas ou complexas (BAHDANAU; CHO; BENGIO, 2014).

É importante salientar que o mecanismo de atenção não tenta codificar uma frase de entrada inteira em um único vetor de tamanho fixo. Em vez disso, ele a codifica em uma sequência de vetores e escolhe um subconjunto desses vetores de forma adaptativa enquanto decodifica a tradução, com isso, não precisa comprimir toda a mensagem em um único vetor (BAHDANAU; CHO; BENGIO, 2014).

2.2.4 *Transformers*

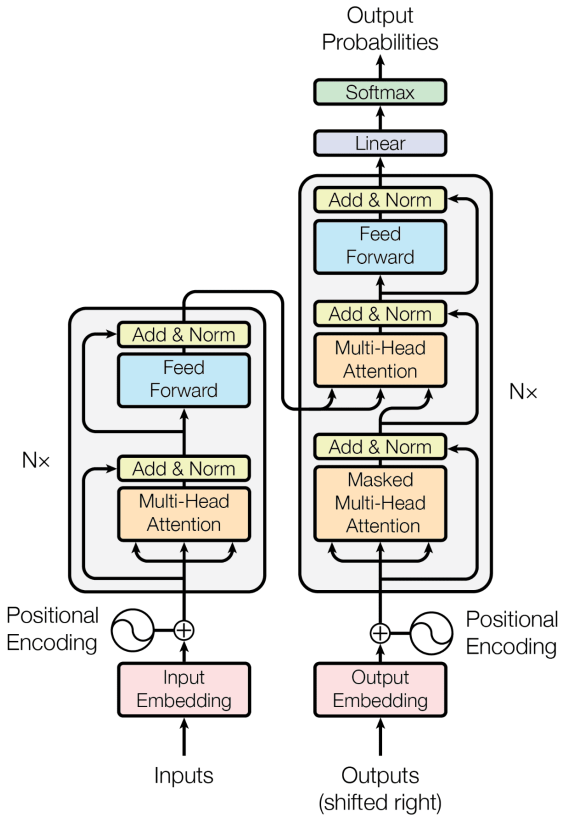
Segundo Aitken et al. (2021), configurações de *encoder-decoder* já provaram seu êxito em tarefas de *seq2seq*, como em modelagem de linguagem, tradução de idiomas e conversão de voz para texto. Muitas dessas arquiteturas, como os *Transformers*, são completamente baseados em mecanismos de atenção.

O *Transformer* é o primeiro modelo de tradução baseado inteiramente em *self-attention* para computar representações de sua entrada e saída sem usar Redes Neurais Recorrentes, do inglês *Recurrent Neural Networks* (RNNs), ou Redes Neurais Convolucionais, do inglês

Convolutional Neural Networks (CNNs), alinhadas à sequência (VASWANI et al., 2017). Para tal, usa o *self-attention* como um mecanismo de atenção que relaciona diferentes posições de uma única sequência para calcular uma representação da sequência. Dessa forma, o *Transformer* pode ser usado com sucesso em uma variedade de tarefas, incluindo compreensão de leitura, resumos e implicação textual (VASWANI et al., 2017).

Dessa forma, Vaswani et al. (2017) explicam que os *Transformers* dispensam RNNs e LSTM inteiramente, permitindo modelar dependências sem considerar sua distância nas sequências de entrada ou saída com a paralelização do mecanismo de atenção, que geram múltiplas representações dos *tokens*, em que cada representação pode se referir a uma relação contextual diferente. Dessa forma, ao contrário das RNNs que processam texto em sequência, os *Transformers* aplicam o *self-attention* para calcular um peso de atenção em paralelo para cada palavra do texto de entrada, medindo a influência que cada palavra tem sobre a outra, permitindo mais paralelização do que RNNs para o treinamento do modelo em grande escala, implicando em uma melhor eficiência do pré-treinamento do modelo, incrementando seu desempenho significativamente (VASWANI et al., 2017).

Figura 2 – Arquitetura do modelo *Transformer*.



Fonte: Vaswani et al. (2017).

Com isso, nos modelos RNNs e CNN, o número de operações necessárias para relacionar

sinais de duas posições arbitrárias de entrada ou saída cresce na distância entre as posições, enquanto que no *Transformer* isso é reduzido a um número constante de operações, obtendo boa performance especialmente com o *multi-head attention* (VASWANI et al., 2017).

O *encoder* do *Transformer* mapeia uma sequência de entrada de representações de símbolos $(x_1, ..., x_n)$ para uma sequência de representações contínuas $z = (z_1, ..., z_n)$ (VASWANI et al., 2017). Dado z, o *decoder* gera uma sequência de saída $(y_1, ..., y_m)$ de símbolos, um elemento por vez, de forma auto-regressiva, consumindo os símbolos gerados anteriormente como entrada adicional ao gerar o próximo (VASWANI et al., 2017). Sua arquitetura pode ser visualizada na Figura 2.

Uma função de atenção pode ser descrita como um mapeamento de uma *query* (representação vetorial de uma palavra na sequência) e um conjunto de pares *key-value* (representações vetoriais de todas as palavras na sequência) para uma saída, onde *query, keys, values*, e a saída são todos vetores. A saída é calculada como uma soma ponderada dos *values*, onde o peso atribuído a cada *value* é calculado por uma função de compatibilidade da *query* com sua *key* correspondente (VASWANI et al., 2017).

2.2.4.1 BERT

Conforme citam Devlin et al. (2018), BERT significa *Bidirectional Encoder Representations from Transformers* (Representações de Codificadores Bidirecionais de *Transformers*) e propõe ser diferente dos modelos padrões de linguagem que são unidirecionais, como o OpenAI GPT, os quais usam uma arquitetura que trabalha com a sequência da esquerda para a direita, permitindo os *tokens* se relacionarem apenas com outros anteriores na camada de *self-attention*, tendo um péssimo desempenho em abordagens como resposta às perguntas, quando é preciso ter o contexto de ambas as direções.

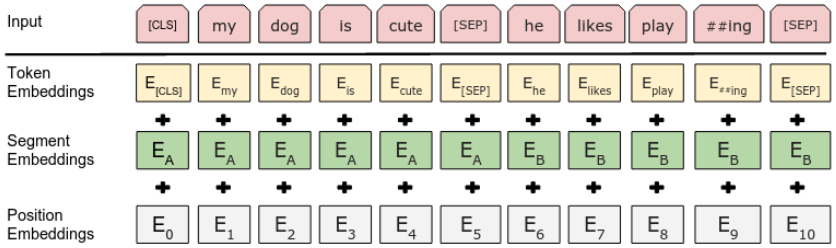
No artigo de Devlin et al. (2018), o exemplo implementado tem como base de dados perguntas e respostas. Assim, o modelo recebeu pares de frases como entrada, sendo que, para as representações de entrada e saída, conforme a Figura 3, tem-se:

- O primeiro *token* sempre é [CLS];
- O estado oculto final correspondente a este *token* é usado como representação da sequência para tarefas de classificação;
- Os pares de sentenças são agrupados em uma única sequência:
  - São separados com o *token* especial [SEP];
  - É adicionado um *embedding* a cada *token* indicando se ele pertence à sentença A ou à sentença B;

– Conforme mostrado na Figura 4, o *embedding* de entrada é denotado como E, o vetor oculto final do *token* especial [CLS] como C $\in \mathbb{R}^H$, e o vetor oculto final para o i-ésimo *token* de entrada como $T_i \in \mathbb{R}^H$.

- Para um determinado *token*, sua representação de entrada é construída somando os *tokens*, segmentos e posições de *embedding* correspondentes;

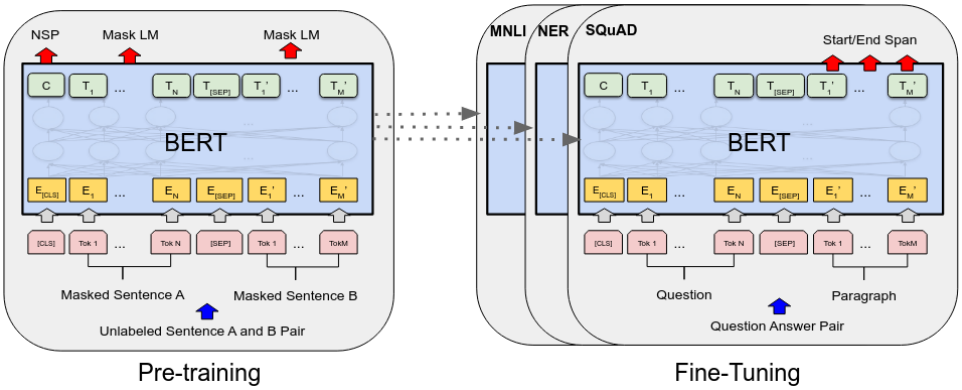
Figura 3 – Representação da entrada do BERT.



Fonte: Devlin et al. (2018).

Para o pré-treinamento, Devlin et al. (2018) fizeram a primeira tarefa com modelo de linguagem mascarada, que oculta o valor de alguns *tokens* de entrada e possui como objetivo prever o *id* do vocabulário original da palavra mascarada com base apenas em seu contexto. Com isso, permite que a representação mescle o contexto esquerdo e direito, criando uma arquitetura bidirecional. Assim, o BERT pode obter bom desempenho em tarefas no nível de frases e também de *tokens*. Para isso, foram mascarados 15% dos *tokens* de entrada aleatoriamente, com os vetores ocultos finais correspondentes aos *tokens* mascarados alimentados em uma saída *softmax* sobre o vocabulário. Contudo, como não há o *token* [MASK] no *fine-tuning*, no pré-treinamento Devlin et al. (2018) substituíram palavras mascaradas com o *token* especial [MASK] em 80% das vezes, em 10% substituíram com um *token* aleatório de vocabulário e nos 10% restantes não mudaram o *token*.

Figura 4 – Procedimentos gerais de pré-treinamento e *fine-tuning* para o BERT.



Fonte: Devlin et al. (2018).

A segunda tarefa do pré-treinamento de Devlin et al. (2018) foi prever a próxima frase, sendo que 50% das vezes a frase B é a próxima que segue A (rotulada como *IsNext*), e 50% das vezes é uma frase aleatória do texto (rotulada como *NotNext*). Como mostrado na Figura 4, C é usado como próxima sentença de predição, denotado como *Next Sentence Prediction* (NSP).

2.2.4.1.1 BERTimbau

Embora haja o BERT multilíngue[1] (mBERT), treinado em 104 línguas, há esforços sendo empregados em modelos de uma única linguagem, como espanhol, italiano, francês, holandês, entre outras, pois, embora seja inviável treinar modelos monolíngues para todos os idiomas, esses trabalhos são motivados pelo desempenho superior e pela eficiência de recursos dos modelos monolíngues em comparação com o mBERT (SOUZA; NOGUEIRA; LOTUFO, 2020). Além disso, modelos pré-treinados são especialmente importantes para idiomas que possuem poucos recursos anotados, mas abundantes dados não rotulados, como o português (SOUZA; NOGUEIRA; LOTUFO, 2020).

Segundo Souza, Nogueira e Lotufo (2020), foram treinados modelos BERTimbau em dois tamanhos, *base*, que conta com 12 camadas, 768 dimensões ocultas, 12 cabeças de atenção, do inglês *attention heads*, e 110.000.000 parâmetros, e *large*, com 24 camadas, 1024 dimensões ocultas, 16 *multi-head attention* e 330.000.000 parâmetros. O comprimento máximo da sentença é definido como 512 *tokens*. A diferença entre letras minúsculas e maiúsculas foi considerada porque se focou na criação de modelos de uso geral e a capitalização é relevante para tarefas como reconhecimento de nomes próprios.

Para treinar os modelos, utilizou-se a base de dados *Brazilian Portuguese Web as Corpus* (brWaC), contendo 3.53 milhões de documentos, que foi o maior corpus português aberto até à data de implementação do BERTimbau. Além de seu tamanho, o brWaC é composto por documentos inteiros e sua metodologia garante alta diversidade de domínios e qualidade de conteúdo, características desejáveis para o pré-treinamento de BERT (SOUZA; NOGUEIRA; LOTUFO, 2020).

Há uma grande diferença de desempenho entre o BERTimbau e o mBERT, o que reforça as vantagens dos modelos monolíngues pré-treinados em dados de vários domínios em comparação com o mBERT, que é treinado apenas em artigos da Wikipédia. Este resultado está no mesmo nível de outros trabalhos monolíngues do BERT (SOUZA; NOGUEIRA; LOTUFO, 2020).

[1] https://huggingface.co/bert-base-multilingual-cased

2.3 Ferramentas

A seguir, serão apresentadas as ferramentas utilizadas para o desenvolvimento do projeto proposto.

2.3.1 Linguagem e Bibliotecas

A linguagem de programação utilizada foi o Python na versão 3.10.12, que permite trabalhar com ótima performance e integrar sistemas de forma mais eficaz, tendo aplicações nas áreas de desenvolvimento *web*, computações científica e numérica, educação para ensino de programação, desenvolvimento de *software* e sistemas de ERP e *e-commerce* (ROSSUM; DRAKE, 2009).

As principais bibliotecas utilizadas para a implementação do projeto foram:

- O *Googletrans*, que é uma biblioteca gratuita e ilimitada que implementa a *Application Programming Interface* (API) do Google Tradutor, fazendo requisições como detecção de linguagem e tradução (PyPI, 2023);
- O *Natural Language Toolkit* (NLTK), que é a principal plataforma para a implementação de programas escritos em Python que lidam com a linguagem humana, proporcionando fácil acesso a interfaces de recursos léxicos e de processamento textual, contendo bibliotecas para a classificação, classificação, tokenização, análise e raciocínio semântico (NLTK, 2023);
- O *TensorFlow*, que é uma plataforma de código aberto que, com sua API Keras, facilita a criação, treinamento e implantação de modelos de Aprendizado Profundo e é fácil de usar e entender, além de proporcionar flexibilidade e ser compatível com um poderoso ecossistema de modelos e bibliotecas de complemento, como o BERT (TensorFlow, 2023);
- O Numpy, que é um projeto de código aberto que permite computação numérica e científica com o Python, fornecendo objetos de matrizes multidimensionais, rotinas para operações rápidas em matrizes, como manipulação de formas, classificação, seleção, álgebra linear básica, operações estatísticas básicas, simulação aleatória entre outros (Numpy, 2023);
- O Pandas, que possui código aberto e é uma ferramenta de análise e manipulação de dados rápida, poderosa, flexível e fácil de usar, permitindo a criação e manipulação de um objeto *DataFrame* rápido e eficiente para manusear dados com indexação integrada (MCKINNEY, 2010);

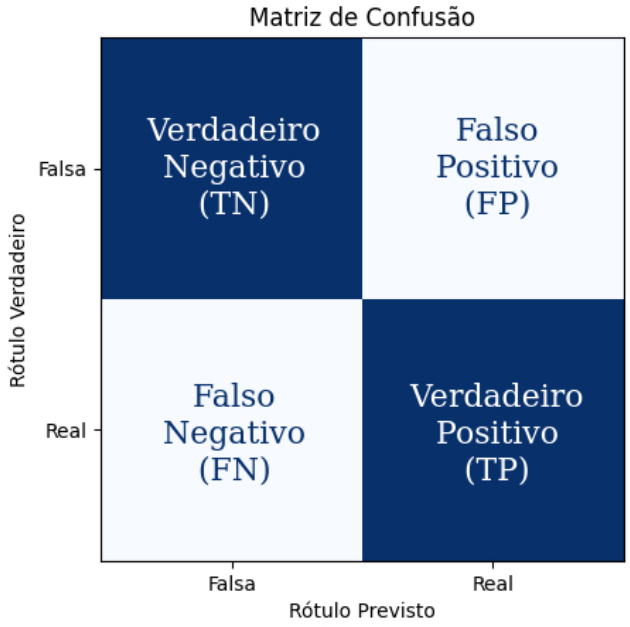
- O *Scikit-Learn*, que oferece ferramentas para a análise e predição de dados, sendo reutilizável em vários contextos, construído em NumPy, SciPy e matplotlib e tendo código aberto (BUITINCK et al., 2013);
- O *Hugging Face*, para construir, treinar e implementar modelos de última geração em aprendizado de máquina, sendo que *Transformers* é uma biblioteca de processamento de linguagem natural e seu *hub* suporta boa parte dos modelos de aprendizado de máquina (HuggingFace, 2023).

2.3.2 Cálculos das Métricas e da Perda

A métrica utilizada nos trabalhos correlatos foi o *macro-averaged F1 score*. Para entendê-la, deve-se considerar:

- *True Positive* (TP): verdadeiro positivo, quando o modelo prevê a classe Positivo e acerta;
- *True Negative* (TN): verdadeiro negativo, quando o modelo prevê a classe Negativo e acerta;
- *False Positive* (FP): falso positivo, quando o modelo prevê a classe Positivo e erra;
- *False Negative* (FN): falso negativo, quando o modelo prevê a classe Negativo e erra;

Figura 5 – Matriz de confusão.



Fonte: Elaborada pela autora.

Tais conceitos podem ser melhor visualizados em uma matriz de confusão, uma tabela que indica os erros e acertos do modelo. Isso poder ser alcançado utilizando a matriz de confusão, como na Figura 5, onde se pode observar com as métricas TP, FN, FP e TN definidas.

Primeiro, deve-se calcular a precisão, do inglês *precision*, dada pela Equação 2.3, que realça a quantidade de acertos de todos da classe Positivo, embora não seja penalizada se classificar um Positivo como Negativo (SOKOLOVA; JAPKOWICZ; SZPAKOWICZ, 2006).

$$\text{precisão} = \frac{TP}{TP + FP} \tag{2.3}$$

Depois, calcula-se a revocação, do inglêS *recall*, que checa dentre todas as situações da classe Positivo com o valor esperado, de todos que realmente são positivos, quantos o modelo conseguiu encontrar, embora não seja penalizada se erroneamente identificar negativos como positivos (SOKOLOVA; JAPKOWICZ; SZPAKOWICZ, 2006).

$$\text{revocação} = \frac{TP}{TP + FN} \tag{2.4}$$

Observa-se que há um *trade-off* entre as métricas *precision* e *recall*.

Por fim, o *F1 score* é a média harmônica entre o *precision* e o *recall*, dado pela Equação 2.5. Se esta métrica estiver baixa, as duas métricas que o compõem está baixa, e se o valor está mediano, uma das métricas está baixa (SOKOLOVA; JAPKOWICZ; SZPAKOWICZ, 2006).

$$F_1Score = 2 \cdot \frac{\text{precisão} \cdot \text{revocação}}{\text{precisão} + \text{revocação}} \tag{2.5}$$

Para calcular o *macro-averaged F1 score*, soma-se todas as pontuações *F1 score* de cada classe i e se divide pelo número de classes N, conforme a Equação 2.6.

$$F_1Score_{macro-avg} = \frac{\sum_{i=1}^{N} F_1Score_i}{N} \tag{2.6}$$

Para calcular a perda L por entropia binária, *binary cross entropy*, é utilizada a Equação 2.7, considerando M a quantidade de amostras analisadas, y_i sendo a classe verdadeira, 0 ou 1, para a amostra i e $p(y_i)$ a probabilidade que o modelo previu de ser a classe verdadeira. As partes $-y_i log(p(y_i))$ e $-(1 - y_i) log(1 - p(y_i))$ estão sendo multiplicadas por -1 pois o log de decimais é negativo e deve ser convertido em positivo para a soma final. Observa-se que quando y_i é 0, $-y_i log(p(y_i))$ resulta em 0, retornando apenas a parte $-(1 - y_i) log(1 - p(y_i))$, que vai diminuindo conforme $p(0)$ aumenta, tendo uma perda baixa. Por outro lado, para y_i igual a 1, $-(1 - y_i) log(1 - p(y_i))$ resulta em 0, retornando apenas $-y_i log(p(y_i))$, que retorna em valores baixos conforme $p(1)$ aumenta (AUFFARTH, 2020). Considerando que entropia

significa que há incerteza, quanto mais espalhadas as probabilidades entre as classes são, maior a entropia é.

$$L = \frac{1}{M}\sum_{i=1}^{M} -y_i log(p(y_i)) - (1-y_i) log(1-p(y_i)) \tag{2.7}$$

2.3.3 Classificadores

A seguir, os classificadores utilizados serão apresentados.

2.3.3.1 Regressão Logística

Segundo Buitinck et al. (2013), a Regressão Logística, do inglês *Logistic Regression* (LR), é um modelo estatístico utilizado para análises de classificação e predição, estimando a probabilidade de um evento ocorrer, baseando-se em um conjunto de variáveis independentes. A função *logit* é aplicada sobre a probabilidade π, como é observado na Equação 2.8, com ln como o logaritmo natural e α e β os parâmetros do modelo. π assume o valor de α quando X é zero, e β ajusta a rapidez com que a probabilidade muda com a mudança de X. Como a relação entre X e π não é linear, β não tem uma interpretação direta neste modelo como acontece na regressão linear comum.

$$logit(\pi) = ln(\frac{\pi}{1-\pi}) = \alpha + \beta_1 x_1 + ... + \beta_k x_k = \alpha + \beta X \tag{2.8}$$

Com isso, mapeia-se a probabilidade como uma função sigmoide de X e pode-se obter π simplificando as Equações 2.9 e 2.10, obtendo a Equação 2.11 (BUITINCK et al., 2013).

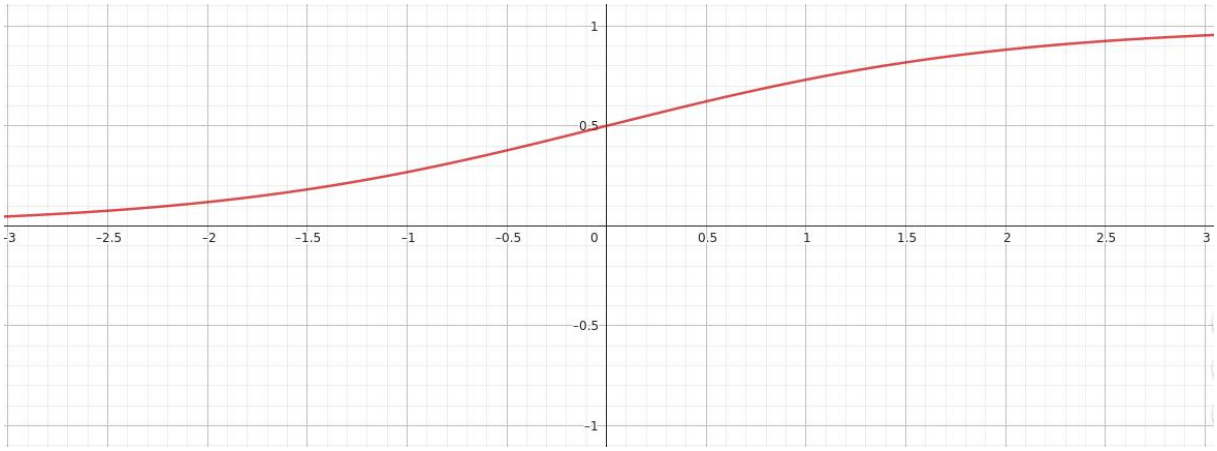
$$e^{\alpha+\beta X} = \frac{\pi}{1-\pi} \tag{2.9}$$

$$e^{\alpha+\beta X} - e^{\alpha+\beta X}\pi = \pi \tag{2.10}$$

$$\pi = \frac{e^{\alpha+\beta X}}{1+e^{\alpha+\beta X}} \tag{2.11}$$

Além disso, como o resultado é uma probabilidade, a variável dependente está entre $[0-1]$ e sua curva pode ser vista conforme a Figura 6.

Figura 6 – Curva de probabilidade da LR.



Fonte: Elaborada pela autora.

2.3.3.2 Naive Bayes

Naive Bayes (NB) abrange um conjunto de algoritmos de aprendizagem supervisionada baseado na aplicação do teorema de Bayes de independência condicional entre cada par de recursos. O teorema de Bayes afirma a relação da Equação 2.12, dada a variável de classe y e o vetor de características x_i dependentes através de x_n (BUITINCK et al., 2013).

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{2.12}$$

Utilizando a suposição de independência condicional da Equação 2.13 para todo i, a relação é simplificada para a Equação 2.14 (BUITINCK et al., 2013).

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{2.13}$$

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{2.14}$$

Como $P(x_1, \ldots, x_n)$ é constante, pode-se usar a regra de classificação da Equação 2.15 (BUITINCK et al., 2013).

$$\begin{gathered} P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \\ \Downarrow \\ \hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y), \end{gathered} \tag{2.15}$$

Os diferentes classificadores de Bayes diferem principalmente pelas suposições que fazem em relação à distribuição de $P(x_i \mid y)$ (BUITINCK et al., 2013).

Os classificadores NB funcionam muito bem em muitas situações do mundo real, como a classificação de documentos e a filtragem de *spam*, requirindo uma pequena quantidade de dados de treinamento para estimar os parâmetros necessários (BUITINCK et al., 2013).

2.3.3.2.1 Naive Bayes Multinomial

Implementa o algoritmo para dados distribuídos multinomialmente e é uma das duas variantes clássicas utilizadas na classificação de texto (onde os dados são normalmente representados como vetores com as frequências dos *tokens*). A distribuição é parametrizada por vetores $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ para cada classe y, onde n é o número de características, que na classificação de textos corresponde ao tamanho do vocabulário, e θ_{yi} é a probabilidade $P(x_i \mid y)$ da característica i aparecer em uma amostra pertencente à classe y (BUITINCK et al., 2013).

O parâmetro θ_y é estimado por uma versão suavizada da contagem de frequência relativa, conforme a Equação 2.16, onde $N_{yi} = \sum_{x \in T} x_i$ é o número de vezes que a característica i aparece em uma amostra da classe y no conjunto de treinamento T, com $N_y = \sum_{i=1}^{n} N_{yi}$ sendo o total de características da classe y. Já $\alpha \geq 0$ leva em conta recursos não presentes nas amostras de aprendizagem e evita probabilidades zero em cálculos adicionais. A configuração $\alpha = 1$ é chamada de suavização de Laplace, enquanto $\alpha < 1$ é chamada de suavização de Lidstone (BUITINCK et al., 2013).

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{2.16}$$

2.3.3.3 Árvore de Decisão

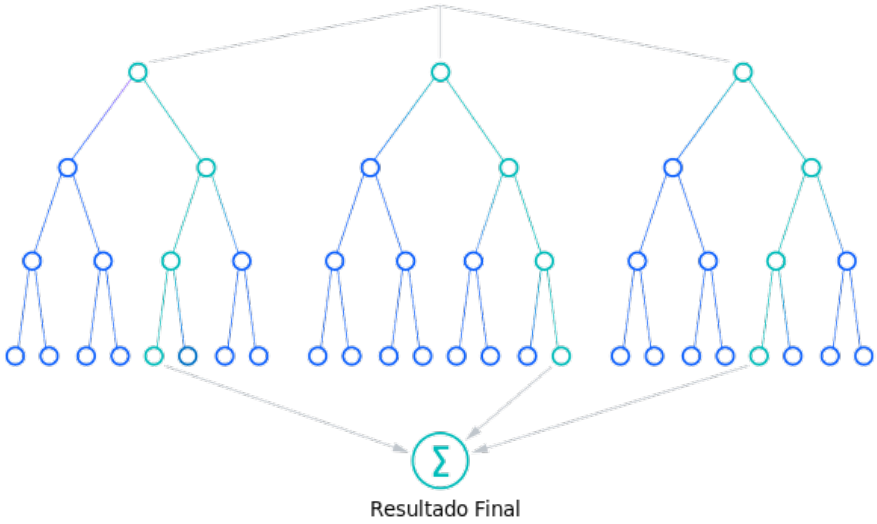
Árvore de decisão é um método de aprendizagem supervisionado usado para classificação e regressão, tendo como objetivo a criação de um modelo que preveja o valor de uma variável alvo, aprendendo regras de decisão simples inferidas a partir das características dos dados (IBM, 2023).

2.3.3.3.1 Floresta Aleatória

Floresta Aleatória, do inglês *Random Forest* (RF), faz parte dos métodos de *ensembled learning*, que são compostos por um conjunto de classificadores e suas previsões são agregadas para identificar o resultado mais popular. Um dos métodos mais conhecidos é o *bagging* (IBM, 2023). Neste método, uma amostra aleatória de dados em um conjunto de treinamento é selecionada com substituição (IBM, 2023).

Depois que várias amostras de dados são geradas, esses modelos são treinados de forma independente, já que essa abordagem é comumente usada para reduzir a variação em um conjunto de dados ruidoso (IBM, 2023). Para uma tarefa de regressão, será calculada a média das árvores de decisão individuais e, para uma tarefa de classificação, será retornada a variável categórica mais frequente como a classe prevista (IBM, 2023), como pode ser observado na Figura 7.

Figura 7 – Representação da RF.



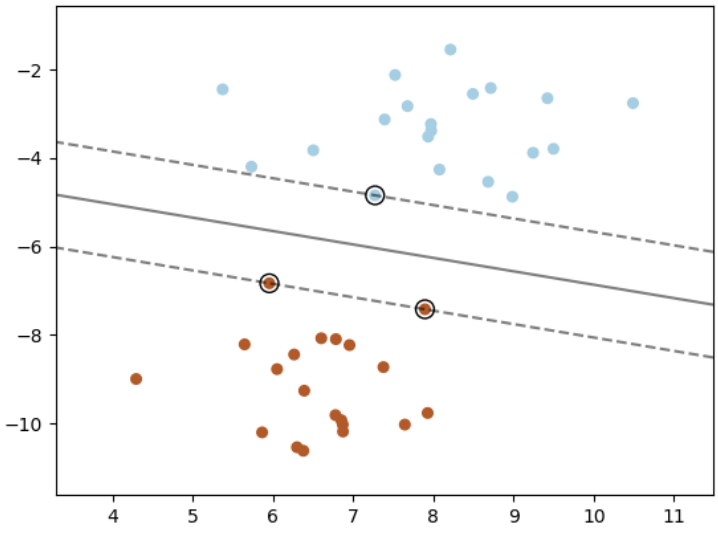
Fonte: Adaptada de IBM (2023).

2.3.3.4 Máquina de Vetores de Suporte

De acordo com Buitinck et al. (2013), uma Máquina de Vetores de Suporte, do inglês *Support Vector Machine* (SVM), constrói um hiperplano ou conjunto de hiperplanos em um espaço de dimensão infinita, que pode ser usado para tarefas como classificação e regressão. Uma boa separação é alcançada pelo hiperplano que possui a maior distância aos pontos de dados de treinamento mais próximos de qualquer classe, pois em geral, quanto maior a margem, menor o erro de generalização do classificador.

A Figura 8 mostra a função de decisão para um problema linearmente separável, com três amostras nos limites da margem, chamados de "vetores de suporte". Em geral, quando o problema não é linearmente separável, os vetores de suporte são as amostras dentro dos limites da margem.

Figura 8 – Exemplos de *kernel* para o SVM.



Fonte: Buitinck et al. (2013).

2.3.3.4.1 Classificação de Vetores de Suporte

A Classificação de Vetores de Suporte-C, do inglês *C-Support Vector Classification* (SVC), é uma classe de SVM capaz de performar classificações binárias ou multi-classificações em uma base de dados (BUITINCK et al., 2013).

Para os vetores $x_i \in IR^P, i = 1, ..., n$, do conjunto de treinamento em duas classes e um vetor $y \in \{1, -1\}^n$, deve-se encontrar $w \in IR^P$ e $b \in IR$, com $sinal(w^T\phi(x) + b)$ sendo correto para a maioria das amostras (BUITINCK et al., 2013). Para isso, deve-se resolver 2.17.

$$\begin{aligned} &\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ &\text{sujeito a } y_i(w^T\phi(x_i) + b) \geq 1 - \zeta_i, \\ &\zeta_i \geq 0, i = 1, ..., n \end{aligned} \tag{2.17}$$

Está tentando maximizar a margem, minimizando $||w||^2 = w^T w$, enquanto aplicando uma penalidade quando uma amostra é classificada erroneamente ou dentro do limite da margem. O valor $y_i(w^T\phi(x_i) + b)$ deveria ser ≥ 1 para todas as amostras, indicando uma predição perfeita, mas os problemas geralmente não são sempre separáveis com um hiperplano, logo, algumas amostras podem estar a uma distância de ζ_i do seu limite da margem correto. O termo C controla a intensidade dessa penalidade, atuando como um parâmetro regularizador inverso.

Deve-se satisfazer 2.18, onde e é o vetor de todos e Q é uma matriz semidefinida de $n \times n$, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, onde $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ é o *kernel* (BUITINCK et al., 2013). Os termos α_i são os coeficientes e seu limite superior é limitado por C.

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ &sujeito\ a\ y^T\alpha = 0 \\ &0 \leq \alpha_i \leq C, i = 1, ..., n \end{aligned} \tag{2.18}$$

Depois do problema de otimização ser resolvido, a saída para uma amostra x é dada por 2.19.

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b, \tag{2.19}$$

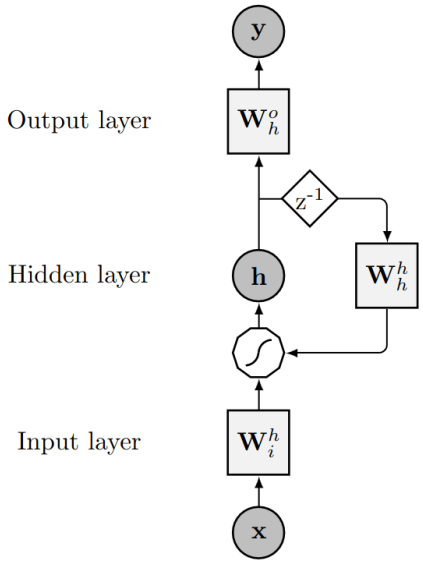
A classe predita corresponde ao seu sinal, sendo preciso somar sobre os vetores de suporte, por exemplo, as amostras dentro da margem, porque os coeficientes duplos α_i são zero para as outras amostras (BUITINCK et al., 2013).

2.3.4 RNN

No contexto de predição, uma RNN é treinada em dados temporais de entrada $x(t)$ para reproduzir uma saída temporal desejada $y(t)$, sendo que $y(t)$ pode ser qualquer série temporal relacionada à entrada e até mesmo uma mudança temporal do próprio $x(t)$ (BIANCHI et al., 2017). A função objetivo a ser minimizada é uma função de perda, que depende do erro entre a saída estimada $\hat{y}(t)$ e a saída real da rede $y(t)$ (BIANCHI et al., 2017).

Na Figura 9, tem-se uma arquitetura simples de RNN, segundo Bianchi et al. (2017). Os círculos representam os nós de entrada (x), ocultos (h) e de saída (y). Os quadrados W_i^h, W_h^h e W_h^o são as matrizes que representam os pesos de entrada, ocultos e de saída respectivamente. Seus valores são comumente ajustados na fase de treinamento por meio de gradiente descendente. O polígono representa a transformação não linear realizada pelos neurônios e $z-1$ é o operador de atraso unitário de n passos entre um nó de origem e um de destino. Os nós de entrada não possuem conexões de entrada e os nós de saída não possuem conexões de saída, mas os nós ocultos possuem ambas.

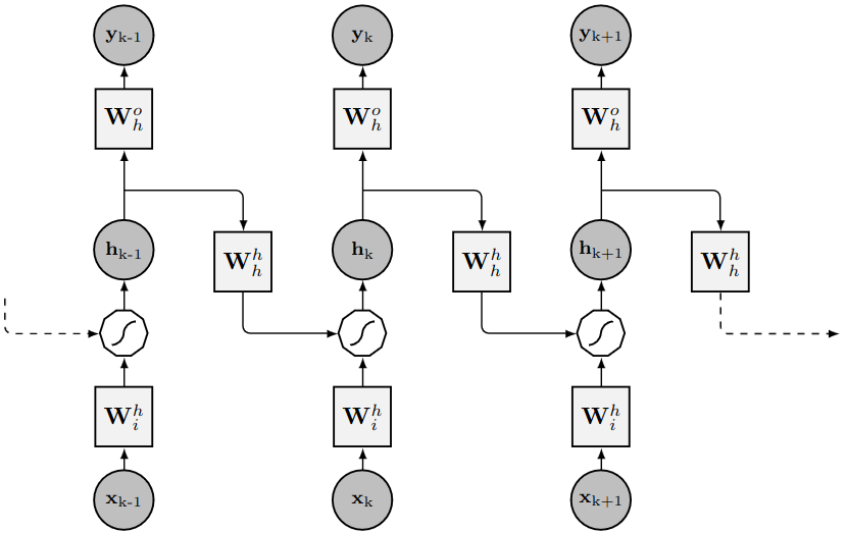
Figura 9 – Arquitetura da RNN.



Fonte: Bianchi et al. (2017).

A aprendizagem baseada em gradiente por Retropropagação ao Longo do Tempo, do inglês *Back Propagation Through Time*, requer uma relação fechada entre os parâmetros do modelo e a função de perda, permitindo propagar a informação do gradiente calculado na função de perda de volta aos parâmetros do modelo (BIANCHI et al., 2017).

Figura 10 – Desdobramento da RNN.



Fonte: Bianchi et al. (2017).

Portanto, conforme Bianchi et al. (2017) salientam, deve-se tomar cuidado com a RNN, já que é cíclica, logo, a RNN é representada como um gráfico equivalente infinito, acíclico e direcionado através de um processo denominado "desdobramento" e consiste em replicar a estrutura da camada oculta da rede para cada intervalo de tempo, conforme a Figura 10. Cada

entrada x_t e saída y_t são relativas a intervalos de tempo diferentes e uma RNN compartilha os mesmos pesos W_i^h, W_h^h e W_h^o em cada intervalo de tempo (BIANCHI et al., 2017).

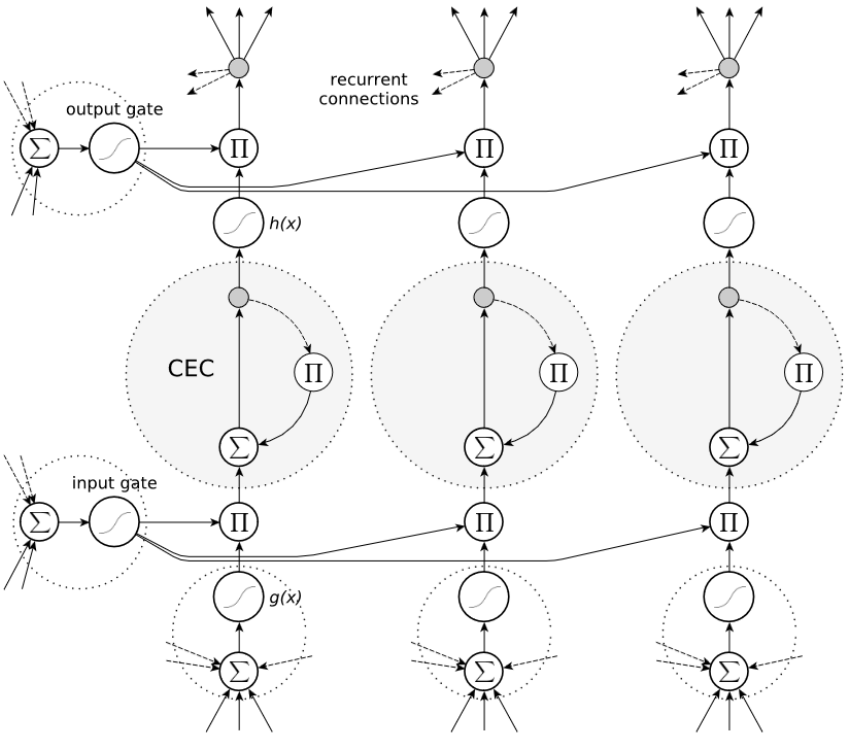
No entanto, embora a estrutura da rede possa ser replicada *ad infinitum*, na prática o desdobramento é sempre truncado após um número finito de instantes de tempo, ocorrendo o problema da dissipação ou explosão do gradiente, que tende a crescer ou a diminuir a cada passo (BIANCHI et al., 2017).

2.3.4.1 *Long Short-Term Memory*

A Memória Longa de Curto Prazo, do inglês *Long Short-Term Memory* (LSTM), é um tipo de RNN utilizada no PLN em aplicações como modelagem de linguagem, transcrição de fala para texto, tradução automática, entre outras, sendo uma evolução da RNN, já que esta possui memória limitada sobre informações anteriores por conta da dissipação do gradiente (STAUDEMEYER; MORRIS, 2019).

Para contornar este problema, a LSTM usa o Carrossel de Erro Constante, do inglês *Constant Error Carousel*, que impõe um fluxo de erro constante dentro dos blocos de memória (STAUDEMEYER; MORRIS, 2019). Pode-se observar um exemplo de três blocos de memórias LSTM na Figura 11.

Figura 11 – Três blocos de memórias LSTM.



Fonte: Staudemeyer e Morris (2019).

Seus blocos de memória possuem controle de acesso na forma de portas de entrada

(*input gates*) e saída (*output gates*), barrando a entrada ou a saída de informações do bloco de memória (STAUDEMEYER; MORRIS, 2019). Staudemeyer e Morris (2019) complementam que os blocos de memória também possuem uma porta de esquecimento (*forget gate*) que permite a reinicialização do bloco de memória sempre que a informação anterior se tornar irrelevante.

Por fim, com o LSTM Bidirecional, BiLSTM, pode-se analisar tanto o futuro quanto o passado de um determinado ponto no contexto do LSTM, ou seja, a entrada é apresentada para frente e para trás em duas redes LSTM separadas, ambas conectadas à mesma camada de saída (STAUDEMEYER; MORRIS, 2019).

3 Trabalhos Correlatos

Neste capítulo são apresentados alguns trabalhos correlatos que também visam à detecção de *fake news* em português e o uso de ILN para esta finalidade.

3.1 *Fake News* em Português

Um dos principais trabalhos abordando a detecção de *fake news* em português é o de Monteiro et al. (2018), que criou a base de dados *Fake.Br Corpus*, a primeira base de dados pública em português com notícias rotuladas como sendo *fake news* ou verdadeiras. Primeiro, separou-se as notícias falsas no período de 2016 até 2018, sendo coletadas manualmente, checando seus detalhes e se eram de fato *fake news*, e, para cada *fake news*, sua respectiva notícia verdadeira foi pesquisada de forma semiautomática utilizando uma similaridade lexical baseada nas palavras-chave, obtendo 7.200 amostras no total. Vários modelos foram treinados, como exemplo de resultado para essa base de dados, foi utilizado o classificador SVM, com uma implementação do *Scikit-learn*, com seus parâmetros padrão. Somente com um BoW já se alcançou bons resultados, 88% do *F1 score*, tanto para notícias verdadeiras quanto falsas.

Já no trabalho de Garcia, Afonso e Papa (2022), coletou-se as notícias entre 2019 e 2021, sendo que as reais são de páginas de agências de notícias conhecidas, como G1, UOL e Extra. Por outro lado, as falsas são de páginas de checagem de notícias, como Aos fatos, Boatos.org, Estadão Verifica e E-farsas. Totalizou-se 11.902 notícias e o melhor resultado com o BoW foi alcançado pela rede neural *Multi-Layer Perceptron*, com 0,930 na *F1 score*, seguido pelo SVM, com 0,926, e RF, com 0,922. Os resultados mostram que mesmo utilizando uma técnica padrão de processamento de linguagem natural, ou seja, BoW, os resultados alcançados foram satisfatórios, embora os melhores resultados ultrapassaram 0,94 utilizando uma arquitetura mais robusta, como *FastText* com CNN.

3.2 ILN aplicada à Detecção de *Fake News*

Sadeghi, Bidgoly e Amirkhani (2022) propuseram um método para a detecção de *fake news* baseado na abordagem de ILN para a base de dados *FakeNewsNet* de Shu et al. (2019), criando a base de dados *FNID-FakeNewsNet*, onde se imita a forma como especialistas apuram notícias falsas, comparando uma notícia suspeita com uma verdadeira, dessa forma, se houver contradição sobre a notícia já verificada, a suspeita será rotulada como falsa, porém, se concordar com a notícia confirmada, é rotulada como verdadeira. O maior resultado obtido por Sadeghi, Bidgoly e Amirkhani (2022), considerando somente as notícias suspeitas, foi obtido por um modelo com o uso de BERT e BiLSTM, resultando em 0,7514 na métrica

macro-averaged F1 score, por outro lado, para o mesmo modelo, ao se utilizar as notícias já verificadas, foi obtido 0,8548 na mesma métrica, melhorando significativamente seu resultado.

No trabalho de Yang, Niven e Kao (2019), utilizou-se uma base de dados em chinês da competição *Web Search and Data Mining 2019 Fake News Classification*, onde são comparados pares de títulos de notícias e se deve averiguar se a segunda notícia concorda, discorda ou não há relação com a primeira. Foram treinados vários modelos de ILN, como RNN e CNN, e, com o uso de *ensemble*, juntando-se os modelos, atingiu-se uma precisão de 88,063% no conjunto de teste.

4 Material e Metodologia

Neste capítulo são apresentadas a base de dados e a metodologia utilizadas para o desenvolvimento do trabalho, bem como o método proposto para a detecção de *fake news* utilizando ILN.

4.1 Base de Dados

Primeiramente, as amostras existentes do *site* do *PolitiFact* foram agrupadas por Sadeghi, Bidgoly e Amirkhani (2022), utilizando sua API até o dia 26 de Abril de 2020. O *PolitiFact* é uma fonte respeitável de apuramento de fatos, criada em 2007, em que uma equipe de repórteres e peritos avaliam artigos de notícias políticas publicados em várias fontes, como a CNN, a BBC e o Facebook. Segundo PolitiFact (2023), o *site PolitiFact* conta com jornalistas que analisam discursos, conferências, folhetos de campanha política, programas de TV e mídias sociais, onde não são checadas opiniões, mas sim declarações duvidosas, que podem ser espalhadas a diante pelos ouvintes.

Posteriormente, Sadeghi, Bidgoly e Amirkhani (2022) pré-processaram os dados recolhidos para remover as partes do texto que representam o rótulo da veracidade das notícias, que geralmente se localizam ao fim do texto, após expressões como "*Our Ruling...*" ou "*We Rate....*"

Sadeghi, Bidgoly e Amirkhani (2022) também mapearam os dados de outras bases de dados, uma destas sendo o *FakeNewsNet* de Shu et al. (2019), a fim de criar uma compatibilidade entre ambas bases. Para isso, as classes foram separadas em *"real"* e *"fake"*, dividindo as amostras em 15.212 para o treinamento (com 7.621 amostras *fakes* e 7.591 verdadeiras) 1.058 para a validação (com 518 amostras *fakes* e 540 verdadeiras) e 1.054 para o teste (com 418 amostras *fakes* e 636 verdadeiras).

Com isso, Sadeghi, Bidgoly e Amirkhani (2022) criaram a base de dados *FNID-FakeNewsNet*, fornecida para a tarefa de detecção de *fake news*, podendo ser especificamente utilizada para detectar *fake news* utilizando a Inferência de Linguagem Natural. Além disso, detém a licença *Creative Commons Attribution 4.0*, permitindo seu compartilhamento e adaptação por atribuição aos autores.

A detecção de *fake news* por humanos baseia-se na inferência da veracidade utilizando um conjunto de notícias confiáveis e não apenas em características estatísticas do conteúdo ou do contexto das notícias (SADEGHI; BIDGOLY; AMIRKHANI, 2022). Cada notícia suspeita pode ser considerada como uma hipótese e o conjunto disponível de notícias confiáveis como uma premissa. A relação de inferência entre a premissa e a hipótese revela a fiabilidade da

notícia suspeita.

4.1.1 Análise e Preparação da Base de Dados

Embora haja bases de dados em português para a detecção de *fake news*, como a *Fake.Br Corpus* de Monteiro et al. (2018) e a *FakeRecogna* de Garcia, Afonso e Papa (2022), não há nenhuma tratando sobre a ILN na apuração de fatos nesta linguagem.

Dessa forma, utilizou-se a base de dados coletada por Sadeghi, Bidgoly e Amirkhani (2022) com compatibilidade à *FakeNewsNet*, *FNID-FakeNewsNet*, tendo apenas duas classes, *"real"* e *"fake"*, em que cada amostra contém os atributos especificados no Quadro 1.

Quadro 1 – Atributos presentes na base de dados *FNID-FakeNewsNet*.

Atributo	**Descrição**
Statement	Declaração investigada no *PolitiFact* feita por uma pessoa ou organização.
Title	Título do artigo publicado no *PolitiFact*.
Time	Data da publicação do artigo.
Speaker	Pessoa ou organização que fez a declaração.
FullText-based-content	Contém textos de notícias relacionadas à notícia em análise.
Paragraph-based-content	Contém uma lista com os parágrafos dos textos de notícias relacionadas à notícia em análise.
Sources	URLs das fontes de informação utilizadas pelo *PolitiFact*.
Label	Se a notícia é "*fake*" ou "*real*"

Fonte: Adaptado de Sadeghi, Bidgoly e Amirkhani (2022).

Porém, essa base de dados está em inglês, sendo necessário traduzi-la. Para isso, utilizou-se a biblioteca *Googletrans*, possibilitando o uso da API do Google Tradutor para traduzir os textos do inglês para o português. Contudo, observa-se que nesta biblioteca só há uma opção de português, então algumas vezes há inconsistências na tradução para o português brasileiro, sendo utilizada a grafia de outros países, como no uso da palavra "económica" em vez de "econômica" e "actual" em vez de "atual".

Considerou-se o campo "*Statement*" como a hipótese e o "*FullText-based-content*" como a premissa para cada amostra, conforme o trabalho de Sadeghi, Bidgoly e Amirkhani (2022).

Como salientam Monteiro et al. (2018), geralmente as notícias verdadeiras são maiores, criando um viés no treinamento. Assim, a fim de evitar uma discrepância nas diferenças de tamanho das premissas e hipóteses, para as premissas do conjunto de treinamento, tentou-se limitar o tamanho máximo para a tradução em aproximadamente 505 palavras, por outro lado, observa-se que o tamanho das hipóteses é naturalmente semelhante, não sendo necessária nenhuma interferência. As médias finais de palavras e caracteres por premissa e hipótese, separadas por seus rótulos, para o conjunto de treinamento, podem ser vistas no Quadro 2.

Quadro 2 – Média de caracteres e palavras nas premissas e hipóteses.

Rótulo	Análise	Média
Real	Caracteres nas premissas	3154,3997
	Palavras nas premissas	512,3448
	Caracteres nas hipóteses	125,8918
	Palavras nas hipóteses	20,8673
Falsa	Caracteres nas premissas	3063,3925
	Palavras nas premissas	497,2056
	Caracteres nas hipóteses	121,3452
	Palavras nas hipóteses	19,9495

Fonte: Elaborado pela autora.

Na Figura 12, pode-se verificar o *word cloud* com as 1.000 palavras mais utilizadas nos textos das declarações falsas, ignorando as *stop words*.

Figura 12 – *Word cloud* das declarações falsas do conjunto de treinamento.

Fonte: Elaborada pela autora.

Já na Figura 13, pode-se verificar o *word cloud* com as 1.000 palavras mais utilizadas nos textos das declarações verdadeiras, também ignorando as *stop words*.

Figura 13 – *Word cloud* das declarações verdadeiras do conjunto de treinamento.

Fonte: Elaborada pela autora.

Verifica-se que o tema principal para ambas as classes é o cenário político estadunidense. Além disso, para as declarações falsas, a concentração de palavras mais utilizadas é mais dispersa, enquanto que, para as verdadeiras, é mais concentrada.

4.2 Classificadores

Fez-se o pré-processamento do texto, como visto na Subseção 2.2.2.1. Por exemplo, a primeira declaração do conjunto de treinamento é "*A national organization says Georgia has one of America's toughest ethics laws.*" Após a tradução com o GoogleTrans, obtém-se "Uma organização nacional afirma que a Geórgia tem uma das leis éticas mais rígidas da América." Ao tirar os acentos e a pontuação, converter as letras maiúsculas em minúsculas e aplicar a lematização, o resultado é: "organizacao nacional afirmar georgia lei etico rigido america". As premissas foram concatenadas às suas respectivas hipóteses.

A classe *CountVectorizer* da biblioteca *Scikit-learn* implementa a tokenização e contagem de ocorrências em uma única classe, além de também permitir passar parâmetros indicando que o texto deve ser mantido em letra minúscula e a lista de *stop words* a ser ignorada. A lista de *stop words* em português foi fornecida através da biblioteca NLTK. Para a conversão aos lemas, utilizou-se uma lista pública[1] com as palavras em português.

Os parâmetros foram ajustados com o auxílio da classe *GridSearchCV* na biblioteca *Scikit-learn*.

Para o modelo de LR sem premissas, utilizou-se:

- *CountVectorizer* com *ngram_range=(1, 2)*;
- *TfidfTransformer* com *use_idf=True* e *sublinear_tf=True*;
- *LogisticRegression* com *penalty='l2'* e *C=1.6238*.

Para o modelo de LR com premissas, utilizou-se:

- *CountVectorizer* com *ngram_range=(1, 2)*;
- *TfidfTransformer* com *use_idf=True* e *sublinear_tf=True*;
- *LogisticRegression* com *penalty='l2'* e *C=29.7635*.

Para os modelos de NB sem e com premissas, utilizou-se:

- *CountVectorizer* com *ngram_range=(1, 2)*;
- *TfidfTransformer* com *use_idf=True* e *sublinear_tf=True*;
- *MultinomialNB* com os parâmetros padrões.

Para o modelo de RF sem premissas, utilizou-se:

[1] https://github.com/michmech/lemmatization-lists/raw/master/lemmatization-pt.txt

- *CountVectorizer* com *ngram_range=(1, 2)*;
- *TfidfTransformer* com *use_idf=True* e *sublinear_tf=True*;
- *RandomForestClassifier* com *max_depth=20* e *n_estimators=500*.

Para o modelo de RF com premissas, utilizou-se:

- *CountVectorizer* com os valores padrões;
- *TfidfTransformer* com *use_idf=True* e *sublinear_tf=True*;
- *RandomForestClassifier* com *max_depth=20* e *n_estimators=500*.

Para os modelos de SVM com e sem premissas, utilizou-se os valores padrões de *CountVectorizer* e *SVC*. Para o *TfidfTransformer*, utilizou-se *use_idf=True* e *sublinear_tf=True*.

4.3 BERTimbau e BiLSTM

Utilizou-se o modelo pré-treinado *TFBertModel*, disponível no *Hugging Face Hub*, que permite inicializar um modelo que já possui os mesmos pesos treinados divulgados pelos autores do BERTimbau, Souza, Nogueira e Lotufo (2020), sendo compatível com o *TensorFlow*. Inicializou-se o BERTimbau-*Large*[2].

Para o modelo com o BERTimbau, não houve um pré-processamento do texto igual ao dos modelos anteriores, já que o BERTimbau foi treinado em textos com pontuação, letras maiúsculas e sem lematização, mas sim se passou todos os pares de premissa e hipótese referentes a cada amostra no *tokenizer* do BERTimbau. O BERTimbau consegue trabalhar com até 512 *tokens* (SOUZA; NOGUEIRA; LOTUFO, 2020), que devem ser a soma da quantidade dos *tokens* da premissa e hipótese juntas, dessa forma, precisou-se limitar o tamanho da premissa, que é um conjunto de parágrafos, dessa forma, foram escolhidos e concatenados os cinco maiores parágrafos. No *tokenizer*:

- São adicionados os *tokens* especiais [CLS] e [SEP], a sentença é separada em *tokens* e se mapeia tais *tokens* em seus IDs, obtendo o vetor *input_ids*, com o valor numérico de cada *token*;
- É aplicado o *padding*, preenchendo o vetor referente ao *input_ids* com o *token* especial [PAD] caso seu tamanho seja menor que 512. Por outro lado, caso o tamanho passe 512, os IDs são truncados. Dessa forma, o tamanho do vetor obtido é sempre 512;

[2] https://huggingface.co/neuralmind/bert-large-portuguese-cased

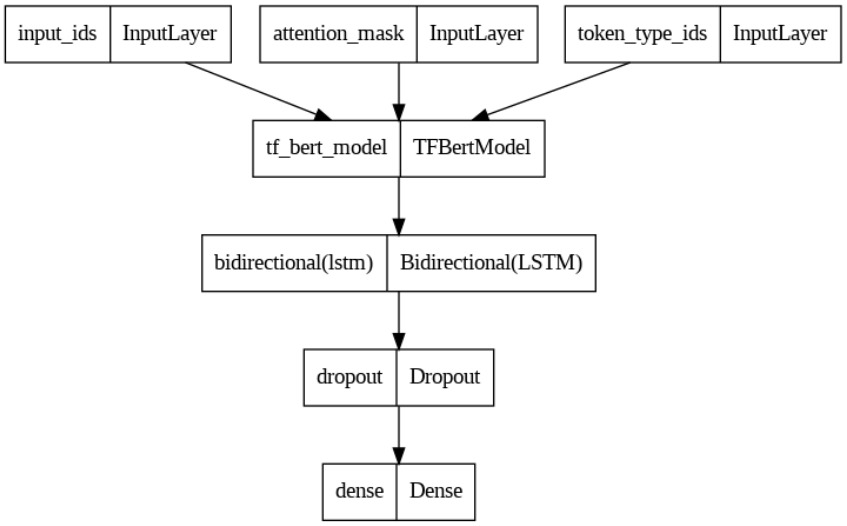
- Cria-se um vetor *attention_mask* de tamanho 512 para diferenciar os *tokens* de *padding*, sendo que se for de *padding* seu valor é 0, senão é 1, indicando que o modelo deve dar uma atenção maior se o *token* não for [PAD];
- Também é criado um vetor *token_type_ids*, com o mesmo tamanho de 512, que indica se o *token* no *input_ids* é da primeira sentença ou da segunda, armazenando 0 ou 1 respectivamente.

Para treinar o modelo com só as hipóteses, realizou-se o mesmo processo descrito, mas limitou-se em 100 *tokens*, passando somente as sentenças das declarações.

Dessa forma, para cada amostra, os modelos com o BERTimbau recebem como entrada três vetores e transformam as sentenças em vetores de *embeddings*. Os pesos do BERTimbau sempre foram mantidos congelados. A saída do BERTimbau utilizada foi o *last_hidden_state*, que contém a representação para cada *token* em cada sequência.

Em seguida, os *embeddings* passam para uma camada de BiLSTM. Para o modelo que considera as premissas, seu BiLSTM possui 32 unidades. Já para o modelo que considera somente as hipóteses, seu BiLSTM possui 16 unidades, pois os *embeddings* para o limite de 100 *tokens* são de tamanho menor, não precisando de uma rede neural mais complexa. Após isso, colocou-se uma camada de *dropout* com regularização de 0,2, ignorando aleatoriamente 20% dos neurônios no treinamento. Por fim, retorna-se a saída em uma camada *dense* de uma unidade com ativação sigmoide. A arquitetura desses modelos pode ser vista na Figura 14.

Figura 14 – Arquitetura dos modelos criados com os *embeddings* do BERTimbau.



Fonte: Elaborada pela autora.

A perda utilizada foi a entropia binária, de acordo com a Equação 2.7, o otimizador utilizado foi o Adam, a taxa de aprendizado adotada foi de 4×10^{-4} e o tamanho do *batch* foi 32. Para o modelo sem o uso das premissas, o modelo foi treinado por 4 épocas, para o com as premissas, o modelo foi treinado por 5 épocas.

Se a saída é menor do que 0,5, considerou-se a notícia suspeita como falsa, senão foi considerada como verdadeira.

5 Resultados e Discussão

Dentre os modelos treinados por Sadeghi, Bidgoly e Amirkhani (2022), alguns são NB, RF, LR, SVM e BiLSTM, junto com os *embeddings* do GloVe e do BERT. Os autores apontaram que se utilizou os parâmetros padrões do *Scikit-learn* para os classificadores. Além disso, Sadeghi, Bidgoly e Amirkhani (2022) não citaram se houve um pré-processamento textual.

As pontuações de Sadeghi, Bidgoly e Amirkhani (2022) para a métrica *macro-averaged F1 score* para os dados sem premissa na base de dados em inglês podem ser checadas no Quadro 3.

Quadro 3 – Resultados de Sadeghi, Bidgoly e Amirkhani (2022), sem o uso de premissas, na base de dados *FNID-FakeNewsNet*, para o conjunto de teste.

Modelo	**GloVe**	**BERT**
NB	0,6923	0,6691
RF	0,6520	0,6567
LR	0,6850	0,6949
SVM	0,7001	0,6766
BiLSTM	0,6170	0,7514

Fonte: Adaptado de Sadeghi, Bidgoly e Amirkhani (2022).

Já as pontuações para as hipóteses com premissas podem ser checadas no Quadro 4.

Quadro 4 – Resultados de Sadeghi, Bidgoly e Amirkhani (2022), com o uso de premissas, na base de dados *FNID-FakeNewsNet*, para o conjunto de teste.

Modelo	**GloVe**	**BERT**
NB	0,6778	0,6623
RF	0,6666	0,7132
LR	0,7351	0,8007
SVM	0,7322	0,7607
BiLSTM	0,8512	0,8548

Fonte: Adaptado de Sadeghi, Bidgoly e Amirkhani (2022).

Os resultados obtidos neste projeto sem o uso de premissas podem ser conferidos no Quadro 5.

Quadro 5 – Resultados obtidos com o método proposto, sem o uso de premissas, na base de dados *FNID-FakeNewsNet* traduzida, para o conjunto de teste.

Modelo	**Representação das Palavras**	**Macro-averaged F1 score**
NB	BoW e TF-IDF	0,7162
RF	BoW e TF-IDF	0,6599
LR	BoW e TF-IDF	0,7125
SVM	BoW e TF-IDF	0,6885
BiLSTM	BERTimbau	0,7205

Fonte: Elaborado pela autora.

Os resultados obtidos neste projeto com o uso de premissas podem ser conferidos no Quadro 6.

Quadro 6 – Resultados obtidos com o método proposto, com o uso de premissas, na base de dados *FNID-FakeNewsNet* traduzida, para o conjunto de teste.

Modelo	**Representação das Palavras**	**Macro-averaged F1 score**
NB	BoW e TF-IDF	0,8102
RF	BoW e TF-IDF	0,7732
LR	BoW e TF-IDF	0,8309
SVM	BoW e TF-IDF	0,8234
BiLSTM	BERTimbau	0,8199

Fonte: Elaborado pela autora.

No geral, as pontuações obtidas foram maiores comparadas às do trabalho de Sadeghi, Bidgoly e Amirkhani (2022). Isso pode ser atribuído ao ajuste dos parâmetros, ao pré-processamento textual e à diferença na forma de representação das palavras utilizadas.

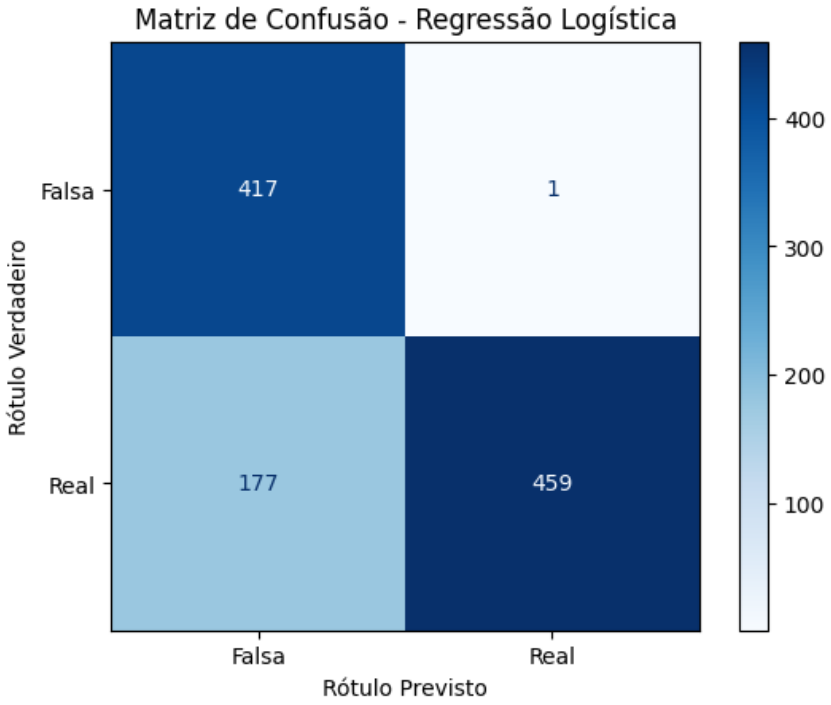
Observa-se que a pontuação para o modelo com o BERTimbau com premissa poderia ser melhor, provavelmente por causa de erros na tradução causados pela biblioteca *Googletrans*, que nem sempre fornece traduções corretas, que façam sentido, e para o português brasileiro. Além disso, por conta da limitação de 512 *tokens*, utilizou-se somente os cinco maiores parágrafos, que nem sempre possuem relação entre si, para a premissa.

Todos os modelos implementados com o uso de premissa conseguiram superar seus respectivos resultados quando usada somente a hipótese, inclusive com o NB, que no trabalho de Sadeghi, Bidgoly e Amirkhani (2022) foi o único a obter uma pontuação melhor para as hipóteses sem suas premissas.

De acordo com Monteiro et al. (2018), classificar incorretamente e, consequentemente, filtrar notícias verdadeiras é mais prejudicial do que não detectar algumas notícias falsas, sendo a mesma lógica utilizada para a detecção de *spam*. Observando, na Figura 15, a matriz de confusão do modelo da LR com premissas, que obteve o melhor resultado, verifica-se que o número de vezes que o modelo prevê uma notícia suspeita como falsa e na verdade é verdadeira é $\frac{177}{1054} = 16,79\%$ do total de amostras, dessa forma, raramente as notícias verdadeiras são

perdidas, corroborando para a eficácia do uso de premissas também na detecção de *fake news* na língua portuguesa, mesmo que a base de dados não seja originalmente neste idioma.

Figura 15 – Matriz de confusão da LR com premissas.



Fonte: Elaborada pela autora.

6 Conclusão

Como problemática, explicou-se que a divulgação de informações não profissionais, sem supervisão ou verificação, pode resultar na disseminação deliberada de notícias falsas. Portanto, a identificação da veracidade das notícias representa um desafio nas redes sociais *online*. Isso ocorre porque tanto indivíduos quanto organizações podem compartilhar *fake news* nessas mídias, muitas vezes com objetivos lucrativos, competitivos ou somente por entretenimento, já que essas notícias frequentemente atraem mais atenção do que as notícias verdadeiras. As notícias falsas podem causar danos irreparáveis a pessoas, empresas e governos, resultando em efeitos devastadores, incluindo o aumento da ansiedade social, a diminuição da produtividade de organizações e a interrupção dos ciclos econômicos.

Diante disso, observou-se que há uma carência de estudos que apliquem a ILN para investigar *fake news* como uma alternativa à análise com base somente em seu conteúdo e na fonte de publicação.

Assim, a proposta de trabalho é significativa no âmbito técnico e acadêmico e consiste em utilizar a ILN como método para identificar as notícias falsas, especialmente no contexto da língua portuguesa, simulando o processo de análise realizado por especialistas. Em outras palavras, busca-se identificar contradições ou implicações entre uma notícia suspeita, que é considerada como hipótese, e notícias já confirmadas como verdadeiras, que são modeladas como as premissas. Portanto, quando uma notícia suspeita entra em conflito com informações previamente confirmadas como verdadeiras, é classificada como *fake*. Caso contrário, se estiver em concordância com informações previamente confirmadas como verídicas, é considerada verdadeira.

O propósito geral deste estudo foi alcançado, já que foram propostos e avaliados modelos utilizando BoW e TF-IDF para a representação das palavras em combinação com os classificadores LR, NB, RF e SVM. Além disso, também se testou BiLSTM com o BERTimbau, que é derivado do BERT, mas pré-treinado exclusivamente em português brasileiro. Verificou-se que, para todos os modelos, obtém-se um resultado melhor quando se utiliza as notícias verdadeiras juntamente com as suspeitas. O melhor resultado foi obtido com o LR, conseguindo uma pontuação de 0,83 na métrica *macro-averaged F1 score*.

Assim, finalizou-se o objetivo de desenvolver um método automatizado para detectar notícias falsas utilizando ILN.

6.1 Trabalhos Futuros

Em um trabalho futuro, seria interessante testar a base de dados *FNID-FakeNewsNet* em outros idiomas e checar se os resultados também melhoram com o uso das premissas.

Além disso, no caso do BERTimbau, uma possível forma de aumentar sua pontuação é contornar o problema do limite de 512 *tokens*, por exemplo, passando cada parágrafo para o seu decodificar e concatenando os *embeddings* para cada amostra.

Referências

AITKEN, K.; RAMASESH, V.; CAO, Y.; MAHESWARANATHAN, N. Understanding how encoder-decoder architectures attend. *Advances in Neural Information Processing Systems*, v. 34, p. 22184–22195, 2021.

ALDWAIRI, M.; ALWAHEDI, A. Detecting fake news in social media networks. *Procedia Computer Science*, Elsevier, v. 141, p. 215–222, 2018.

AUFFARTH, B. *Artificial Intelligence with Python Cookbook: Proven recipes for applying AI algorithms and deep learning techniques using TensorFlow 2.x and PyTorch 1.6*. Packt Publishing, 2020. ISBN 9781789137965. Disponível em: <https://books.google.com.br/books?id=5agGEAAAQBAJ>.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

BIANCHI, F. M.; MAIORINO, E.; KAMPFFMEYER, M.; RIZZI, A.; JENSSEN, R. Recurrent neural network architectures. In: _____. [S.l.: s.n.], 2017. p. 23–29. ISBN 978-3-319-70337-4.

BOWMAN, S. R.; ANGELI, G.; POTTS, C.; MANNING, C. D. A large annotated corpus for learning natural language inference. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.]: Association for Computational Linguistics, 2015. Disponível em: <https://nlp.stanford.edu/projects/snli/>. Acesso em: 09 de abril de 2023.

BUITINCK, L.; LOUPPE, G.; BLONDEL, M.; PEDREGOSA, F.; MUELLER, A.; GRISEL, O.; NICULAE, V.; PRETTENHOFER, P.; GRAMFORT, A.; GROBLER, J.; LAYTON, R.; VANDERPLAS, J.; JOLY, A.; HOLT, B.; VAROQUAUX, G. API design for machine learning software: experiences from the scikit-learn project. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. [S.l.: s.n.], 2013. p. 108–122.

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>. Acesso em: 09 de abril de 2023.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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.

HuggingFace. *The AI community building the future.* 2023. Disponível em: <https://huggingface.co/>. Acesso em: 04 de Abril de 2023.

IBM. *What is random forest?* 2023. Disponível em: <https://www.ibm.com/topics/random-forest>. Acesso em: 21 de outubro de 2023.

KIM, Y.; JANG, M.; ALLAN, J. Explaining text matching on neural natural language inference. *ACM Trans. Inf. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 4, sep 2020. ISSN 1046-8188. Disponível em: <https://doi.org/10.1145/3418052>. Acesso em: 09 de abril de 2023.

MACCARTNEY, B. *Natural language inference*. [S.l.]: Stanford University, 2009.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). *Proceedings of the 9th Python in Science Conference*. [S.l.: s.n.], 2010. p. 56 – 61.

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.

NASEEM, U.; RAZZAK, I.; KHAN, S. K.; PRASAD, M. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *CoRR*, abs/2010.15036, 2020. Disponível em: <https://arxiv.org/abs/2010.15036>.

NLTK. *Natural Language Toolkit*. 2023. Disponível em: <https://www.nltk.org/>. Acesso em: 21 de outubro de 2023.

Numpy. *What is NumPy?* 2023. Disponível em: <https://numpy.org/doc/stable/user/whatisnumpy.html>. Acesso em: 09 de abril de 2023.

PolitiFact. *The Principles of the Truth-O-Meter: PolitiFact's methodology for independent fact-checking*. 2023. Disponível em: <https://www.politifact.com/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/>. Acesso em: 18 de outubro de 2023.

PyPI. *GoogleTrans*. 2023. Disponível em: <https://pypi.org/project/googletrans/>. Acesso em: 21 de outubro de 2023.

ROSSUM, G. V.; DRAKE, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

SABARMATHI, K. R.; GOWTHAMI, K.; KUMAR, S. S. Fake news detection using machine learning and natural language inference (nli). *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, v. 1084, n. 1, p. 012018, mar 2021. Disponível em: <https://dx.doi.org/10.1088/1757-899X/1084/1/012018>. Acesso em: 09 de abril de 2023.

SADEGHI, F.; BIDGOLY, A. J.; AMIRKHANI, H. Fake news detection on social media using a natural language inference approach. *Multimedia Tools and Applications*, v. 81, n. 23, p. 33801–33821, Sep 2022. ISSN 1573-7721. Disponível em: <https://doi.org/10.1007/s11042-022-12428-8>. Acesso em: 09 de abril de 2023.

Scikit-Learn. *Feature extraction*. 2023. Disponível em: <https://scikit-learn.org/stable/modules/feature_extraction.html>. Acesso em: 20 de outubro de 2023.

SHU, K.; MAHUDESWARAN, D.; WANG, S.; LEE, D.; LIU, H. *FakeNewsNet: A Data Repository with News Content, Social Context and Spatialtemporal Information for Studying Fake News on Social Media*. 2019.

SOKOLOVA, M.; JAPKOWICZ, N.; SZPAKOWICZ, S. Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. In: . [S.l.: s.n.], 2006. Vol. 4304, p. 1015–1021. ISBN 978-3-540-49787-5.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Bertimbau: Pretrained bert models for brazilian portuguese. In: CERRI, R.; PRATI, R. C. (Ed.). *Intelligent Systems*. Cham: Springer International Publishing, 2020. p. 403–417. ISBN 978-3-030-61377-8.

STAUDEMEYER, R. C.; MORRIS, E. R. Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. *CoRR*, abs/1909.09586, 2019. Disponível em: <http://arxiv.org/abs/1909.09586>.

STORKS, S.; GAO, Q.; CHAI, J. Y. Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *CoRR*, abs/1904.01172, 2019. Disponível em: <http://arxiv.org/abs/1904.01172>.

TensorFlow. *Why TensorFlow*. 2023. Disponível em: <https://www.tensorflow.org/about>. Acesso em: 09 de abril de 2023.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. *CoRR*, abs/1706.03762, 2017. Disponível em: <http://arxiv.org/abs/1706.03762>.

YANG, K.; NIVEN, T.; KAO, H. Fake news detection as natural language inference. *CoRR*, abs/1907.07347, 2019. Disponível em: <http://arxiv.org/abs/1907.07347>.