



MODELOS DE CLASSIFICAÇÃO DE TEXTO PARA DETECÇÃO DE NOTÍCIAS FALSAS COM O FAKE.BR CORPUS

Trabalho de Conclusão de Curso

Engenharia da Computação

Aldo Ferreira de Souza Monteiro
Orientador: Prof. Bruno José Torres Fernandes



**Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação**

**ALDO FERREIRA DE SOUZA
MONTEIRO**

**MODELOS DE CLASSIFICAÇÃO DE
TEXTO PARA DETECÇÃO DE
NOTÍCIAS FALSAS COM O FAKE.BR
CORPUS**

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, 05/2021

Monteiro, Aldo Ferreira de Souza

Modelos de Classificação de Texto para Detecção de Notícias Falsas com o Fake.Br Corpus / Aldo Ferreira de Souza Monteiro. – Recife - PE, 2021.

xiii, 21 f. : il. ; 29 cm.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) Universidade de Pernambuco, Escola Politécnica de Pernambuco, Recife, 2021.

Orientador: Prof. Dr. Bruno José Torres Fernandes.

Inclui referências.

1. Classificação de texto. 2. Fake News. 3. Deep Learning. I. Título. II. Fernandes, Bruno José Torres. III. Universidade de Pernambuco.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 11/5/2021, às 08h30min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **ALDO FERREIRA DE SOUZA MONTEIRO**, orientado(a) pelo(a) professor(a) **BRUNO JOSÉ TORRES FERNANDES**, sob título **MODELOS DE CLASSIFICAÇÃO DE TEXTO PARA DETECÇÃO DE NOTÍCIAS FALSAS COM O FAKE.BR CORPUS**, a banca composta pelos professores:

JOSÉ PAULO G. DE OLIVEIRA (PRESIDENTE)

BRUNO JOSÉ TORRES FERNANDES (ORIENTADOR)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

Aprovada Aprovada com Restrições* Reprovada

e foi-lhe atribuída nota: 10,0 (dez)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

AVALIADOR 1: Prof (a) **JOSÉ PAULO G. DE OLIVEIRA**

AVALIADOR 2: Prof (a) **BRUNO JOSÉ TORRES FERNANDES**

AVALIADOR 3: Prof (a)

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Agradecimentos

Aos familiares e amigos por todo o apoio e ajuda, que muito contribuíram para a realização deste trabalho. E a todos professores e funcionários da POLI, essenciais no meu processo de formação profissional, pela dedicação, e por tudo o que aprendi ao longo dos anos do curso.

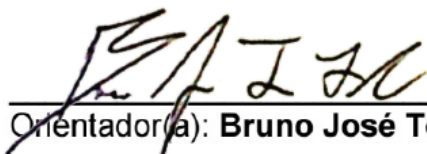
Autorização de publicação de PFC

Eu, **Aldo Ferreira de Souza Monteiro** autor(a) do projeto de final de curso intitulado: **MODELOS DE CLASSIFICAÇÃO DE TEXTO PARA DETECÇÃO DE NOTÍCIAS FALSAS COM O FAKE.BR CORPUS**; autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.



Aldo Ferreira de Souza Monteiro



Orientador(a): **Bruno José Torres Fernandes**

Coorientador(a):



Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 11/5/2021

Resumo

A grande facilidade de produzir, acessar e distribuir conteúdo nos dias atuais favoreceu a rápida disseminação de informações falsas. O mau hábito de compartilhar notícias sem nem mesmo checar a fonte, tornou as chamadas fake news um grande problema moderno. Nesse contexto, uma ferramenta capaz de detectar automaticamente textos enganosos seria de grande utilidade. Uma possível abordagem para tal detecção automática é por meio de técnicas de processamento de linguagem natural. Neste trabalho foram utilizados os dados do Fake.Br Corpus, para treinamento e avaliação de diversos algoritmos clássicos de machine learning e também o modelo de *deep learning bidirectional encoder representations from transformers* (BERT), com o objetivo de identificar as melhores abordagens para construção de um sistema de detecção automática de *fake news*.

Palavras-chave: Fake news; NLP; BERT; Fake.Br Corpus; Classificação de texto.

Abstract

The great ease of producing, accessing and distributing content nowadays has favored the rapid dissemination of false information. The bad habit of sharing news without even checking the source has made so-called fake news a big modern problem. In this context, a tool capable of automatically detecting misleading texts would be of great use. A possible approach for such automatic detection is through natural language processing techniques. In this work, Fake.Br Corpus data was used for training and evaluation of several classic machine learning algorithms and also a deep learning model called bidirectional encoder representations from transformers (BERT), in order to identify the best approaches for building an automatic fake news detection system.

Keywords: Fake news; NLP; BERT; Fake.Br Corpus; Text classification.

Índice de Figuras

Figura 1.	Gráfico de barras da quantidade de notícias por categoria e por classe (verdadeiras em azul e falsas em vermelho).	9
Figura 2.	Histograma empilhado normalizado do número de palavras das notícias por classe (verdadeiras em azul e falsas em vermelho). Pode-se observar no gráfico que a partir de um certo tamanho de notícia, todas são verdadeiras.	9
Figura 3.	Histograma empilhado normalizado do número de palavras das notícias por classe (verdadeiras em azul e falsas em vermelho) na base de treino após processo de balanceamento	12
Figura 4.	Arquitetura utilizada para classificação usando BERT.	14
Figura 5.	<i>Boxplot</i> das acurácias médias de cada combinação do <i>grid search</i> , agrupando-se por classificador: SVM (azul), <i>random forest</i> (vermelho), <i>naive bayes</i> (laranja), KNN (verde) e árvore de decisão (roxo). Observa-se que a SVM obteve resultados bem superiores aos demais algoritmos.	16

Índice de Tabelas

Tabela 1.	10 melhores resultados obtidos no grid search com algoritmos tradicionais de machine learning.	17
Tabela 2.	Resultados do modelo selecionado na grid search no conjunto de teste.	17
Tabela 3.	Resultados do modelo BERT no conjunto de teste.	18

Índice de Siglas

BERT – bidirectional encoder representations from transformers

BoW – bag of words (saco de palavras)

KNN – k-nearest neighbors (k-vizinhos mais próximos)

MLM – masked language model

MLP – multilayer perceptron

NLP – natural language processing (processamento de linguagem natural)

NLTK – natural language toolkit

NSP – next sentence prediction

SVM – support vector machine (Máquina de vetor de suporte)

TF-IDF – term frequency inverse document frequency

Sumário

Introdução	1
Caracterização do Problema	1
Objetivos	2
Objetivos Específicos	2
Estrutura de Capítulos	2
Referencial Teórico	3
Fake News	3
Impactos	3
Classificação de Texto Automática	3
O Processo de Classificação	4
Pré-processamento	4
Extração de Características	5
Classificação	5
Deep Learning	6
BERT	6
Fake.Br Corpus	8
Visão Geral	8
Análises e Caracterização	8
Arranjo Experimental	11
Preparação do Conjunto de Dados para Treino e Teste	11
Pré-processamento	12
Extração de Características	13
Classificação	13
Grid Search	14
BERT	14
Resultados	16
Grid Search	16
BERT	17
Comparação com Trabalhos Anteriores	18
Considerações Finais	19
Conclusão	19

Trabalhos Futuros	19
Referências	19

Capítulo 1

Introdução

1.1 Caracterização do Problema

Através das redes sociais e sua crescente difusão, a informação, nos dias atuais, circula num ritmo nunca antes visto. Essa facilidade na produção e distribuição de conteúdo, se por um lado proporcionou uma maior democratização do conhecimento, por outro lado favoreceu a disseminação de informações falsas. O mau hábito de compartilhar notícias sem nem mesmo checar a fonte, tornou as chamadas *fake news* um grande problema moderno. Este problema, apesar de aparentemente inofensivo, como explicado por Bussular [2], possui sérios impactos para a sociedade no mundo todo.

Nesse contexto, uma forma de identificar notícias falsas de forma automática seria de grande ajuda, possibilitando, por exemplo, alertar os internautas sobre a informação duvidosa antes de compartilhá-la. Entretanto, esta tarefa é longe de trivial, uma vez que, além de compreensão do texto, ela também envolve conhecimentos externos ao mesmo. Além disso, uma notícia falsa pode estar repleta de informações verdadeiras, não sendo nada fácil separar o que é verdadeiro do que é falso até mesmo para um ser humano. Outro desafio para a construção deste classificador, como apontado por Monteiro et al [6], é a escassez de dados rotulados de *fake news* em língua portuguesa.

Apesar de existirem vários projetos buscando esta detecção automática, a literatura, segundo Okano [8,p.27], ainda carece de modelos para identificação de textos enganosos em língua portuguesa.

1.2 Objetivos

Este trabalho tem como objetivo propor um modelo de classificação de texto para identificação automática de notícias falsas em língua portuguesa, analisando diferentes abordagens e comparando com os resultados atualmente obtidos na literatura.

1.2.1 Objetivos Específicos

- Analisar a base escolhida, o Fake.Br Corpus;
- Aplicar algoritmos clássicos de *machine learning* com diferentes parametrizações;
- Aplicar algoritmos de *deep learning*;
- Analisar resultados obtidos e compará-los com os alcançados em trabalhos anteriores.

1.3 Estrutura de Capítulos

Este trabalho está dividido em 6 capítulos. No primeiro, são introduzidas a problemática e os objetivos. No segundo capítulo, é trazida uma visão geral sobre os temas abordados, introduzindo alguns conceitos chave para a compreensão do restante da monografia. O terceiro introduz a base de dados utilizada no trabalho, analisando suas características. No quarto capítulo são detalhadas a metodologia e as técnicas aplicadas para a solução do problema proposto. O capítulo cinco traz os resultados obtidos a partir da aplicação dos métodos descritos no capítulo 4. No sexto e último capítulo é feito o fechamento do trabalho, comparando os resultados e propondo possíveis abordagens para trabalhos futuros.

Capítulo 2

Referencial Teórico

2.1 Fake News

A questão das notícias falsas, ou *fake news*, vem recebendo cada vez mais destaque nos últimos anos. Praticamente qualquer área da vida em sociedade pode ser alvo de informações intencionalmente enganosas: política, saúde, religião, economia, entre muitas outras, são temas constantes de *fake news*. Seja para conseguir algum tipo de vantagem ou simplesmente causar o caos, notícias falsas são forjadas todos os dias ao redor do mundo. Apesar de parecer recente, esta problemática é bem mais antiga do que parece: desde o final do século XIX o termo *fake news* já era utilizado [1], mas foi com a facilidade de acesso e compartilhamento de conteúdo, apenas possível nos dias atuais, que tal problemática atingiu as dimensões que vemos hoje.

2.1.1 Impactos

A disseminação de informações falsas impacta toda a sociedade em diversas áreas, gerando confusão e induzindo as pessoas ao erro. Imagem de pessoas e empresas, rumo de eleições e políticas de vacinação são apenas alguns exemplos de coisas que são impactadas pelas *fake news*.

2.2 Classificação de Texto Automática

Em nosso dia a dia, estamos em constante contato com textos das mais diversas naturezas: livros, *e-mails*, notícias, *websites*, redes sociais entre muitas outras, que cobrem boa parte da informação que consumimos. O tratamento manual de tanta informação pode ser inviável, principalmente quando o volume de dados se torna maciço. Nesse contexto, surge a área de processamento de linguagem natural (NLP, do inglês *natural language processing*) que visa automatizar várias tarefas envolvendo linguagem humana (também chamada de linguagem natural), possibilitando que máquinas possam executá-las. Ler, interpretar e até mesmo

escrever textos são exemplos de tarefas, antes exclusivamente humanas, que agora podem ser realizadas por máquinas utilizando técnicas de NLP.

Uma das áreas mais conhecidas da NLP é a de classificação de texto, que visa categorizar textos, automaticamente, em uma ou mais classes pré-definidas. Uma aplicação dessa técnica já bem presente no nosso cotidiano são os famosos filtros de *spam* que classificam, automaticamente, os *e-mails* como “normais” ou “suspeitos”, livrando nossas caixas de entrada dos mal intencionados. No contexto deste trabalho, são utilizadas técnicas de NLP para classificar notícias como “verídicas” ou “falsas”.

2.2.1 O Processo de Classificação

Em geral, um sistema de classificação de texto pode ser dividido nas seguintes etapas:

- Pré-processamento
- Extração de características
- Classificação

2.2.2 Pré-processamento

A primeira etapa do processo consiste em realizar uma série de tratamentos, limpezas ou transformações no texto para possibilitar o bom funcionamento das etapas posteriores. O tipo de processamento a ser realizado vai depender das técnicas aplicadas nas próximas etapas. Exemplos de pré-processamentos comumente utilizados incluem:

- Remoção de números e símbolos;
- Remoção das *stopwords* - palavras que trazem pouco ou nenhum conteúdo semântico (apesar de serem essenciais do ponto de vista sintático). Artigos, conjunções e pronomes são exemplos de *stopwords*;
- Lematização - processo de redução das palavras flexionadas do texto as suas respectivas formas base (ou lema). Por exemplo: as palavras andando, andei, andaria e andará seriam todas substituídas por “andar”.

2.2.3 Extração de Características

Para realizar a categorização, o classificador precisa de um conjunto de critérios ou características (normalmente quantitativos) que permitam diferenciar bem as classes. Em um sistema para classificação de espécies de flores, por exemplo, as características (ou *features*) poderiam ser as larguras e comprimentos de pétalas e sépalas. Dessa forma, cada flor seria representada por esses quatro valores para então serem analisadas pelo classificador, que, baseando-se apenas nessas características, poderá determinar as espécies de cada flor.

A extração de características, ou *feature extraction*, é uma etapa fundamental para o processo de classificação. É nela que o texto é convertido em um conjunto de características representativas do mesmo, que possam ser utilizadas pelo classificador. Para *feature extraction de texto*, pode-se citar:

- *Bag of Words* (BoW): o texto é representado pela contagem das ocorrências de cada palavra ou *n-gram* (sequência de n palavras).
- Term frequency inverse document frequency (TF-IDF): baseado no BoW, com a diferença de que são usados valores relativos para as contagens dos termos (percentual de ocorrências no texto - *term frequency*) além de um fator inversamente proporcional a frequência do termo nos demais documentos da base (*inverse document frequency*). Dessa forma, palavras que aparecem frequentemente em todos os textos perdem relevância enquanto as que aparecem especificamente em alguns textos ganham maior peso.

2.2.4 Classificação

Nesta última etapa do processo, as *features* extraídas são analisadas por um classificador para determinar a que classe o texto pertence. Para construir esse classificador, comumente são utilizados algoritmos de aprendizagem de máquina (ML, do inglês *machine learning*) para, a partir de exemplos, identificar padrões consistentes entre as *features* e as classes de modo a generalizar para novos textos. Dentre os algoritmos mais utilizados pode-se citar:

- Árvore de decisão: encontra o ponto em cada *feature* que ocasiona o maior ganho de informação para construir uma árvore de condições [7,p.72].

- *Random forest*: utiliza várias árvores de decisão, cada uma treinada em um subconjunto aleatório de *features*. O resultado final da classificação é obtido pela média das predições individuais de cada árvore de decisão [7,p.85].
- *Naive Bayes*: se baseia no teorema de Bayes para calcular a probabilidade do pertencimento à cada classe [7,p.70].
- Máquina de vetor de suporte (SVM, do inglês *support vector machine*): determina o hiperplano que separa as classes com a maior margem possível. A princípio, capaz apenas de lidar com dados linearmente separáveis, mas, a partir da utilização do chamado *kernel trick* (elevação da dimensionalidade dos dados para torná-los linearmente separáveis), passa a lidar muito bem com padrões não lineares [7,p.94].
- K-vizinhos mais próximos (KNN, do inglês *k-nearest neighbors*): usa os k pontos mais próximos (no espaço de *features*) ao ponto a ser classificado para determinar sua classe [7,p.37].

2.2.5 Deep Learning

Uma classe de algoritmos que vem obtendo sucessos nas mais diversas áreas são as redes neurais artificiais, que utilizam modelos matemáticos dos neurônios e os interconectam formando uma rede semelhante ao que ocorre no cérebro humano. As bases teóricas para o treinamento de redes neurais com uma grande quantidade de camadas de neurônios (aprendizado profundo, do inglês *deep learning*), apesar de já existirem há um bom tempo, só recentemente, graças à grande disponibilidade de dados e poder computacional, vem sendo utilizadas com sucesso em diversas aplicações práticas, como explicado por Goodfellow et al [4,p.12].

2.2.6 BERT

No contexto do processamento de linguagem natural, um algoritmo de *deep learning* que têm obtido grandes sucessos em diferentes aplicações é o *bidirectional encoder representations from transformers* (BERT), proposto em 2018 por Devlin et al [3]. Sua arquitetura consiste basicamente em uma sucessão de camadas de *encoders* do *transformer*, que é outra rede neural, proposta em 2017 por Vaswani [10]. O *transformer* é formado por duas partes: o *encoder* e o *decoder*.

Primeiramente, o BERT é pré-treinado de modo não supervisionado em um grande volume de texto. Uma vez pré-treinado, o BERT pode ser treinado em tarefas de domínio específico em um processo chamado ajuste fino (do inglês *fine tuning*). A etapa de *pre-training* (pré-treinamento) é composta por dois objetivos de treinamento:

- *Masked Language Model* (MLM): Algumas palavras do texto de entrada, selecionadas aleatoriamente, são mascaradas, ou seja, ficam ocultas para o algoritmo. O BERT deve então aprender como prever as palavras que foram ocultadas baseando-se nas demais palavras. Nesse processo o BERT aprende como modelar a linguagem, ou seja, como representar cada palavra dentro do contexto em que estão inseridas.
- *Next Sentence Prediction* (NSP): Dadas duas sentenças, o BERT deve aprender a identificar se a segunda sucede imediatamente à primeira no texto de onde elas foram extraídas. Neste processo, o BERT aprende a identificar a relação entre duas sentenças, algo bastante útil para algumas aplicações.

No processo de *fine tuning*, o BERT, com mínima alteração em sua arquitetura (comumente com alguma estrutura extra para adaptá-lo à tarefa a ser executada), é treinado no conjunto de dados específico do problema a ser resolvido (por exemplo, detecção de *fake news*). Nesse processo, tanto os parâmetros da nova estrutura que foi nele plugada, como os parâmetros obtidos durante o pré-treinamento são ajustados, ou seja, para aprender como executar a nova tarefa, ele também ajusta sua forma de modelar a linguagem, captando as peculiaridades linguísticas do domínio de conhecimento específico onde ele está sendo aplicado.

Uma limitação do BERT é sua incapacidade de lidar com textos muito longos. O tamanho máximo da sequência de tokens (unidades mínimas do vocabulário do BERT) que ele é capaz de processar é 512.

Capítulo 3

Fake.Br Corpus

3.1 Visão Geral

O Fake.Br Corpus foi o primeiro conjunto de dados rotulados sobre notícias falsas em língua portuguesa, proposto em 2018 por Monteiro et al [6]. A base de dados é composta por 7200 notícias, sendo 3600 verdadeiras e 3600 falsas, coletadas entre janeiro de 2016 a janeiro de 2018. Todas as 7200 notícias, além de rotuladas como “verdadeiras” ou “falsas”, também foram divididas manualmente em 6 categorias de acordo com seu tema:

- Política (58%)
- TV e celebridades (21.4%)
- Sociedade e cotidiano (17.7%)
- Ciência e Tecnologia (1.5%)
- Economia (0.7%)
- Religião (0.7%)

Os autores do *corpus* também tiveram o cuidado de filtrar as *fake news* com meias-verdades, mantendo apenas as completamente falsas.

3.2 Análises e Caracterização

Uma característica bem positiva do Fake.Br, para treinamento de modelos preditivos, é seu balanceamento de classes, isto é, sua igualdade no número de exemplos para cada classe (“verdadeira” ou “falsa”). Este balanceamento é encontrado também dentro de cada uma das 6 categorias, como pode ser observado no gráfico de barras da Figura 1.

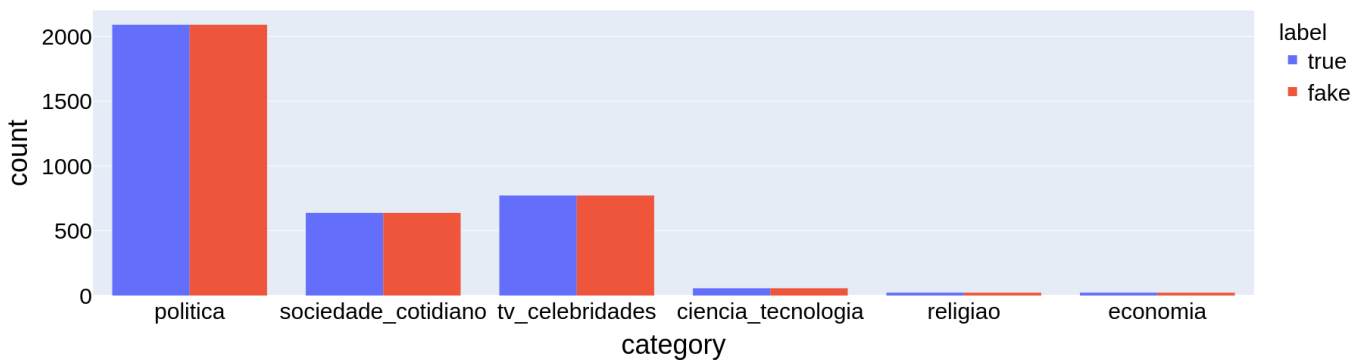


Figura 1. Gráfico de barras da quantidade de notícias por categoria e por classe (verdadeiras em azul e falsas em vermelho).

Esta característica é importante para evitar viés no treinamento de modelos de classificação. Caso houvesse discrepância entre o número de notícias verdadeiras e falsas dentro de alguma categoria, o classificador poderia ficar enviesado ao classificar textos dessa categoria.

Apesar da base ser balanceada em relação às classes, não se pode afirmar o mesmo em relação aos tamanhos das notícias. As verdadeiras são significativamente maiores que as falsas, como mostra o histograma empilhado normalizado da Figura 2.

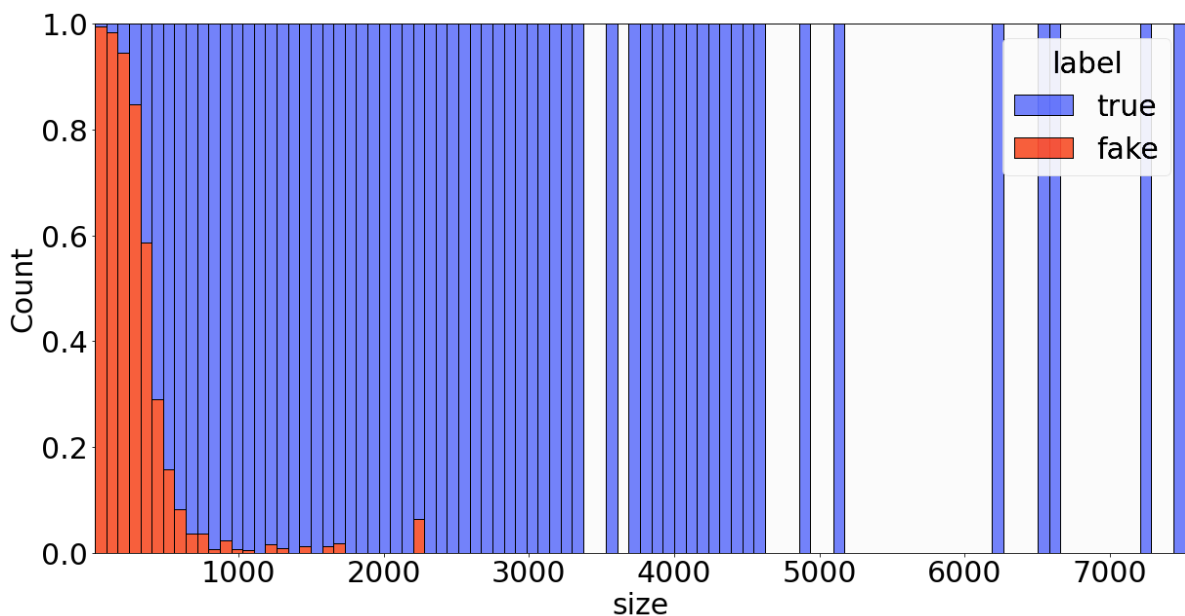


Figura 2. Histograma empilhado normalizado do número de palavras das notícias por classe (verdadeiras em azul e falsas em vermelho). Pode-se

observar no gráfico que a partir de um certo tamanho de notícia, todas são verdadeiras.

Essa discrepância entre o tamanho das notícias verdadeiras e falsas pode ser altamente prejudicial para a capacidade de generalização de um modelo de classificação: embora este corpus possua essa característica, este padrão não se repete, necessariamente, em outras amostras. Um modelo treinado com esta base poderia ter ótimos resultados nela (simplesmente classificando textos longos como verdadeiros e pequenos como falsos), mas não funcionar bem na prática. Para evitar esse problema, pode ser necessário aplicar algum tratamento para equilibrar os tamanhos dos textos antes de treinar o modelo.

Capítulo 4

Arranjo Experimental

Neste capítulo é descrito todo o processo experimental desenvolvido neste trabalho, incluindo tratamentos da base, pré-processamento dos textos, algoritmos de extração de características e classificação, métodos de avaliação e seleção de modelos e hiperparâmetros utilizados.

4.1 Preparação do Conjunto de Dados para Treino e Teste

A base utilizada neste trabalho foi o Fake.Br Corpus. Para verificar a capacidade de generalização do modelo final proposto, foram separados 15% das notícias, selecionadas aleatoriamente, para avaliação final da solução, tal subconjunto recebe o nome de conjunto de teste. O restante (conjunto de treino) foi utilizado para treinamento, validação e seleção de modelos.

O Fake.Br Corpus, conforme explicado no capítulo 3, sofre com um forte desbalanceamento em relação ao comprimento dos textos (notícias falsas são significativamente menores que as verdadeiras). Para evitar um possível viés da classificação em relação ao tamanho das notícias, foi aplicado no conjunto de treino um processo de truncagem dos textos da seguinte forma:

- É inicializada uma lista vazia chamada “resultado”, para comportar as notícias truncadas
- Para cada notícia falsa (em ordem crescente de número de palavras):
 - Se ainda restar alguma notícia verdadeira:
 - É extraída da base a menor notícia verdadeira;
 - A notícia verdadeira extraída é truncada para ter o mesmo número de palavras que a falsa;
 - Ambas, a falsa e a verdadeira, são inseridas na lista “resultado”.

- Caso contrário:
 - Encerra o laço de repetição.
- A lista “resultado”, agora contendo as notícias truncadas, é salva em disco para posterior utilização como conjunto de treino.

Neste processo, não há modificação nos textos, apenas remoção de parte deles. Entretanto, é possível que haja alteração no sentido do texto caso a parte removida seja relevante. Por conta disso, este processo foi aplicado apenas na base de treino, deixando a de teste isenta de qualquer manipulação. A Figura 3 mostra a distribuição de tamanhos das notícias na base de treino após aplicação do processo de balanceamento.

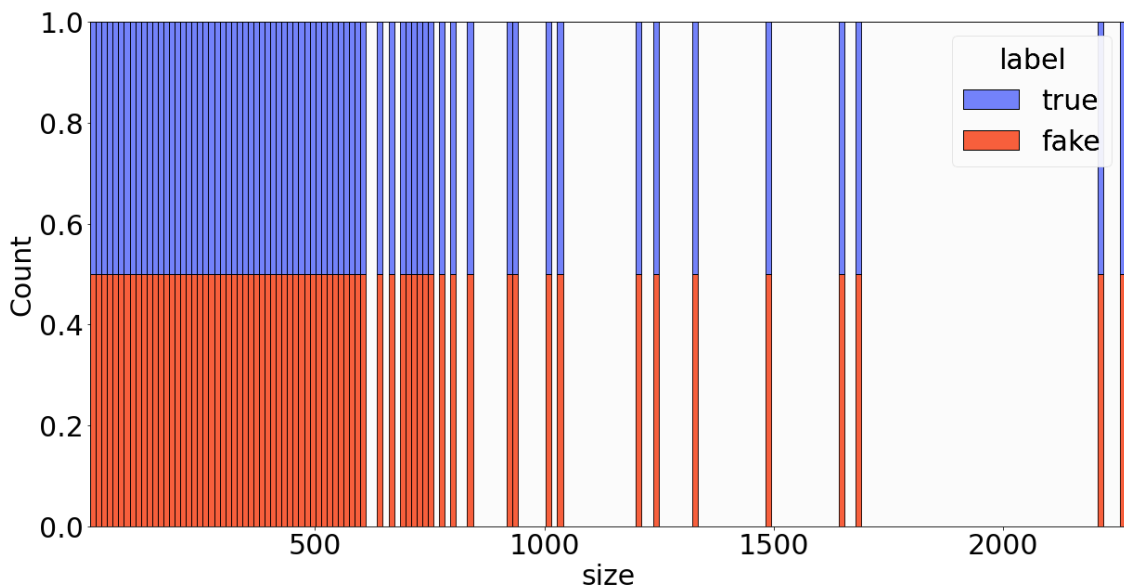


Figura 3. Histograma empilhado normalizado do número de palavras das notícias por classe (verdadeiras em azul e falsas em vermelho) na base de treino após processo de balanceamento.

4.2 Pré-processamento

Foi utilizado um pré-processamento bem simples, consistindo apenas em converter o texto para caixa baixa e a remoção de *stopwords* e números. A lista de *stopwords* utilizada foi obtida a partir da biblioteca NLTK (*natural language toolkit*).

4.3 Extração de Características

Foi utilizada a implementação do TF-IDF da biblioteca *scikit-learn* com diferentes configurações:

- *max features* (número máximo de dimensões na representação final):
 - 1000
 - 10000
 - 20000
- *ngram range* (faixa de tamanho do *n-gram* - número mínimo e máximo de palavras para formação de um *n-gram*):
 - 1-3
 - 1-5
 - 1-7

4.4 Classificação

Também da biblioteca *scikit-learn*, foram utilizados diferentes classificadores com as seguintes configurações:

- **SVM**
 - C (fator de regularização)
 - 1.0
 - 0.75
 - 0.5
 - kernel
 - linear
 - poly
 - rbf
 - sigmoid
- **Naive Bayes** (Multinomial)
- **Árvore de Decisão**
- **Random Forest**
- **KNN**
 - K (número de vizinhos)

■ 5

4.5 Grid Search

Foi executada uma *grid search*, ou seja, uma busca exaustiva ao longo de cada combinação entre configurações da extração de características e da classificação. Cada combinação foi avaliada por meio de um *5-fold cross validation*, usando como métrica a acurácia. Neste processo, foi utilizado apenas o conjunto de treino, deixando o de teste apenas para avaliação final do modelo selecionado.

4.6 BERT

Além dos algoritmos clássicos de *machine learning*, também foram realizados experimentos com algoritmos de *deep learning*. Mais especificamente, foi utilizado um modelo BERT pré-treinado (*multi_cased_L-12_H-768_A-12*) como camada de *embedding* de texto para o classificador. A Figura 4 mostra a arquitetura completa, desde o texto bruto da notícia até as probabilidades da mesma ser “fake” ou “verdadeira”. O texto bruto da notícia é “tokenizado”, isto é, quebrado em unidades mínimas do vocabulário do BERT. Em seguida, é atribuída a cada token uma representação vetorial baseada na semântica das palavras dentro do contexto em que elas estão incluídas. O token “[CLS]”, sempre fixo no início na lista de tokens, ganha uma representação referente ao documento como um todo (“C”), uma vez que, durante o pré-treinamento, foi ensinado a partir de NSP ao invés de MLM como nos demais tokens. O vetor “C” é então utilizado como features para um MLP.

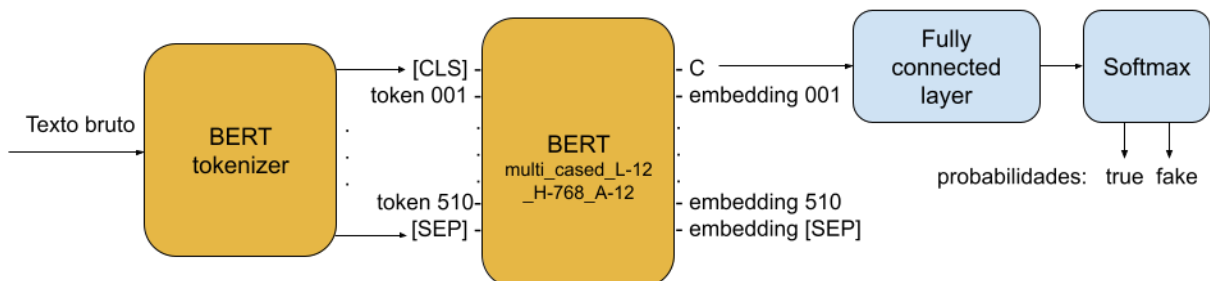


Figura 4. Arquitetura utilizada para classificação usando BERT.

Como o modelo BERT utilizado só consegue lidar com sequências de até 512 tokens, textos muito longos podem ser parcialmente ignorados pelo classificador, que irá desprezar o restante do texto que excede os 512 tokens.

Este classificador foi implementado utilizando a biblioteca *ktrain*, que encapsula o *TensorFlow*, abstraindo detalhes de implementação e agilizando consideravelmente o processo de desenvolvimento [5]. O modelo foi treinado utilizando a *1 Cycle Policy*, proposta por Smith [9], por 1 *epoch* em 90% do conjunto de treino (10% foram usados para validação).

Capítulo 5

Resultados

Neste capítulo são apresentados os resultados dos experimentos detalhados no capítulo 4, além de um comparativo com trabalhos anteriores.

5.1 Grid Search

Cada combinação entre configurações da *feature extraction* e da classificação foi treinada e avaliada em um *5-fold cross validation* no conjunto de treino para encontrar a melhor combinação de hiperparâmetros. Na Figura 5 são exibidos os resultados da *grid search* agrupando-os pelo algoritmo de classificação.



Figura 5. *Boxplot* das acurácias médias de cada combinação do *grid search*, agrupando-se por classificador: SVM (azul), *random forest* (vermelho), *naive bayes* (laranja), KNN (verde) e árvore de decisão (roxo). Observa-se que a SVM obteve resultados bem superiores aos demais algoritmos.

A Tabela 1 mostra os 10 melhores resultados obtidos durante o *grid search* com os algoritmos clássicos de extração de características e classificação.

Tabela 1. 10 melhores resultados obtidos no *grid search* com algoritmos tradicionais de machine learning.

Extração de características			Classificação			acurácia média
algoritmo	max_features	ngram_range	algoritmo	C	kernel	
TFIDF	20000	(1, 3)	SVC	1.0	rbf	90,97%
TFIDF	20000	(1, 3)	SVC	0.75	linear	90,91%
TFIDF	20000	(1, 7)	SVC	1.0	rbf	90,89%
TFIDF	20000	(1, 5)	SVC	1.0	rbf	90,84%
TFIDF	10000	(1, 3)	SVC	1.0	rbf	90,79%
TFIDF	20000	(1, 3)	SVC	0.75	sigmoid	90,78%
TFIDF	10000	(1, 7)	SVC	1.0	rbf	90,76%
TFIDF	10000	(1, 5)	SVC	1.0	rbf	90,76%
TFIDF	20000	(1, 3)	SVC	1.0	linear	90,66%
TFIDF	20000	(1, 3)	SVC	0.75	rbf	90,63%

A configuração que obteve a melhor acurácia (primeira linha da Tabela 1) foi então avaliada no conjunto de teste e obteve 91,67% de acurácia, como mostrado na Tabela 2.

Tabela 2. Resultados do modelo selecionado na *grid search* no conjunto de teste.

	falsa	verdadeira
precision	0,91	0,92
recall	0,93	0,91
f1-score	0,92	0,91
support	553	527
accuracy	91,67%	

5.2 BERT

O modelo foi treinado com 90% do conjunto de treino durante 1 *epoch* e obteve 98,85% de acurácia na validação (10% restantes do conjunto de treino). Em seguida foi aplicado no conjunto de teste onde obteve 97,13% de acurácia, como mostrado na Tabela 3.

Tabela 3. Resultados do modelo BERT no conjunto de teste.

	falsa	verdadeira
precision	0,96	0,99
recall	0,99	0,95
f1-score	0,97	0,97
support	553	527
accuracy	97,13%	

5.3 Comparação com Trabalhos Anteriores

Em um primeiro esforço para a construção de um detector de notícias falsas em língua portuguesa, Monteiro et al [6], construíram o Fake.Br corpus e realizaram os primeiros experimentos nesta base. Utilizando principalmente BoW com uma SVM linear, os autores obtiveram 89% de acurácia em textos truncados. Outro avanço foi realizado por Okano [8] que, utilizando *char n-gram* e testando vários classificadores com diferentes valores para *n*, obteve, no melhor resultado em textos truncados, uma acurácia de 92,32%.

Capítulo 6

Considerações Finais

6.1 Conclusão

Neste trabalho foram realizados experimentos com diversos algoritmos para detecção de notícias falsas em língua portuguesa utilizando o Fake.Br Corpus. O melhor resultado, que foi obtido utilizando um modelo BERT pré-treinado como camada de *embedding*, foi de 97,13% de acurácia em textos truncados, bem superior aos alcançados com algoritmos clássicos de extração de características e classificação. Isso indica um grande potencial dos modelos baseados em *deep learning* para detecção de texto enganosos.

6.2 Trabalhos Futuros

Para trabalhos futuros, pode-se estudar formas para aprimorar o modelo proposto. Uma possibilidade seria ajustar sua arquitetura para permitir a leitura de textos de tamanho arbitrário. Outro ponto importante, é a interpretabilidade do classificador, ou seja, implementar um meio de explicar o motivo da classificação, identificando que parte do texto o levou a crer que a notícia é falsa ou verdadeira. Essa interpretabilidade poderia então ser aplicada para aprimorar ainda mais a assertividade do sistema. Com a extração dos trechos-chave do texto, é possível utilizá-los numa consulta automática em algum motor de busca como Google ou Bing. Os resultados da busca poderiam ser então utilizados como *features* valiosas por um segundo classificador.

Referências

- [1] BATISTA, Rafael. O que são fake news? Origem e perigos. **Mundo Educação**. Disponível em: <<https://mundoeducacao.uol.com.br/curiosidades/fake-news.htm>>. Acesso em: 10 de abril de 2021.
- [2] BUSSULAR, Luis Filipe. O impacto das Fake News na vida em sociedade. **Jusbrasil**, 2018. Disponível em: <<https://fbussular.jusbrasil.com.br/artigos/577903609/o-impacto-das-fake-news-na-vida-em-sociedade>>. Acesso em: 3 de março de 2021.
- [3] DEVLIN, Jacob et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **arXiv preprint arXiv:1810.04805**, 2018. Disponível em: <<https://arxiv.org/pdf/1810.04805.pdf>>. Acesso em: 01 de maio de 2021.
- [4] GOODFELLOW, Ian, et al. **Deep Learning**. MIT Press, 2016. Disponível em: <<http://www.deeplearningbook.org>>. Acesso em: 14 de maio de 2021.
- [5] MAIYA, Arun S.. Ktrain: A Low-Code Library for Augmented Machine Learning. **arXiv preprint arXiv:2004.10703**, 2020. Disponível em: <<https://arxiv.org/pdf/2004.10703.pdf>>. Acesso em: 25 de abril de 2021.
- [6] MONTEIRO, Rafael, et al. **Contributions to the Study of Fake News in Portuguese: New Corpus and Automatic Detection Results**. In: PROPOR, 2018, Canela.
- [7] MÜLLER, Andreas C.; GUIDO, Sarah. **Introduction to Machine Learning with Python: A Guide for Data Scientists**. Sebastopol, CA: O'Reilly, 2016.
- [8] OKANO, Yoshiaki Emerson. **Análise e caracterização de textos intencionalmente enganosos escritos em português usando métodos de processamento de textos**. 2020. 109 f. Tese de Mestrado, Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto (FFCLRP) da USP, Ribeirão Preto.

- [9] SMITH, Leslie N.. No More Pesky Learning Rate Guessing Games. **arXiv preprint arXiv:1506.01186v2**, 2015. Disponível em: <<https://arxiv.org/pdf/1506.01186v2.pdf>>. Acesso em: 27 de abril de 2021.
- [10] VASWANI, Ashish et al. Attention is All you Need. **arXiv preprint arXiv:1706.03762v5**, 2017. Disponível em: <<https://arxiv.org/pdf/1706.03762.pdf>>. Acesso em: 01 de maio de 2021.