Classificação de Notícias Falsas

Luan Matheus Trindade Dalmazo (20211077)
Thalita Maria do Nascimento (20211079)

Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil

I. Introducão

O fácil acesso aos meios de comunicação na internet facilitou que qualquer pessoa possa divulgar notícias, até mesmo falsas, as chamadas *fake news*. O Instituto Mundial de Pesquisa (IPSO) trouxe dados alarmantes em 2018 de que 62% dos entrevistados brasileiros já acreditaram em uma *fake news*, uma porcentagem muito acima da média mundial de 48%. Isso tem impactos danosos reais na sociedade, como o retorno do movimento antivacina, que resultou no aumento de casos de Sarampo no mesmo ano [1]. O tema em questão, portanto, foi escolhido em razão da importância e impacto social no Brasil.

II. Materiais e Métodos

Essa seção tem por objetivo detalhar os métodos e o conjunto de dados utilizados. Inicialmente, apresenta-se a descrição do dataset na Seção II-A. Em seguida, a Seção II-B aborda as estratégias de pré-processamento adotadas. A Seção II-C descreve os modelos avaliados e, por fim, a Seção II-D apresenta as métricas utilizadas para a avaliação. O código desenvolvido para os experimentos está disponível como um repositório público [1].

A. Dataset

O trabalho desenvolvido por Garcia et al. (2022) [2] disponibiliza um conjunto de dados denominado FakeRecogna, o qual contém notícias falsas e verdadeiras em português. O dataset inclui os campos título, subtítulo, notícia, categoria, auto, data, URL e classe, sendo 0 para notícias falsas e 1 para verdadeiras.

O conjunto de dados balanceado (5.951 amostras da classe falsa e 5.951 amostras da classe verdadeira) possui, ao todo, 11.902 amostras distribuídas conforme representado na Tabela I. As notícias verdadeiras foram coletadas de meios como *G1* e *UOL*, enquanto as falsas foram obtidas a partir de agências de checagem, como a *UOL Confere*, responsáveis pela verificação da veracidade dos fatos.

B. Pré-processamento

Previamente ao treinamento dos modelos, foi realizada a etapa de pré-processamento dos dados, com o objetivo de evitar vieses, padronizar o formato e remover informações consideradas desnecessárias para a classificação. As etapas adotadas são apresentadas na Figura 1, e foram fortemente baseadas nas descritas por Garcia et al. (2022) [2].

[1]https://github.com/luandalmazo/ml-work

Tabela I: Distribuição das categorias de notícias, informações disponibilizadas por Garcia et al. (2022) [2].

Categoria	# Notícias	%
Brazil	904	7.6
Entertainment	1,409	12.0
Health	4,456	37.4
Politics	3,951	33.1
Science	602	5.1
World	580	4.9
Total	**11,902**	**100.0**

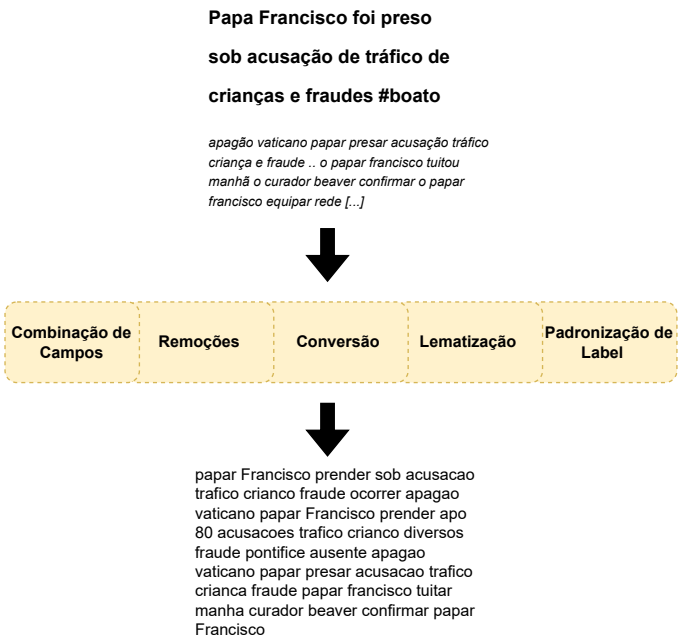

Figura 1: Etapas de pré-processamento adotadas.

A etapa de combinação de campos consiste em unir os campos Título, Subtítulo e Notícia em um único campo

denominado *text*, com o objetivo de facilitar a classificação, reunindo todas as informações relevantes em um só local e agregando elementos considerados importantes.

Em seguida, são realizadas remoções, iniciando pela exclusão de URLs, caracteres especiais e pontuação. Em seguida, são removidos os acentos e termos considerados enviesados. Esses termos compõem uma lista denominada `BIAS_TERMS`, que inclui as palavras: "enganoso", "boato" e "#fake". A remoção desses termos visa reduzir possíveis vieses na classificação, uma vez que sua presença pode influenciar o modelo a associar diretamente essas palavras à classe de notícia falsa. Também foi realizada a remoção de *stopwords*, utilizando o módulo *stopwords* do NLTK para o idioma português. Por fim, são eliminados espaços duplicados.

Após, o texto é então convertido para letras minúsculas e submetido ao processo de lematização, técnica amplamente utilizada para reduzir palavras à sua forma canônica. E então, é realizada a padronização do campo de rótulo: a coluna "Classe" é renomeada para "label" e convertida para o formato numérico inteiro.

C. Modelos Avaliados

Esta seção apresenta os modelos e estratégias utilizadas para o treinamento e avaliação. Será realizado um comparativo entre uma *baseline* e o modelo *BERTimbau*. Ressalta-se que, em todos os treinamentos do modelo *BERTimbau*, foi empregada a técnica de *early stopping* com um fator de paciência igual a 2, a fim de evitar *overfitting*, utilizando a *loss* de validação como métrica de monitoramento. Além disso, todos os treinamentos adotaram a técnica de validação cruzada (*K-Fold Cross-Validation*) com $K = 5$.

1) Baseline: A *baseline* visou aplicar no *dataset* os mesmos modelos clássicos aplicados por seus criadores, mas em uma estratégia de representação de texto diferente. Enquanto Garcia et al. (2022) [2] utilizou Bag-of-Words (BoW) e FastText, nós utilizamos a TF-IDF. A *Bag-of-Words* representa a frequência de cada palavra (Frequência do Termo); o *FastText* é a representação por *word embeddings*; o TF-IDF, por sua vez, é o produto entre a Frequência do Termo pela Frequência Inversa do Documento, ou seja, dá maior importância a termos que ocorrem mais raramente entre documentos (em nosso caso, entre amostras)[2].

Os classificadores escolhidos por Garcia et al. (2022) [2] foram *Multi-Layer Perceptron* (MLP), *Naive Bayes* (NB), *Random Forest* (RF), *Support Vector Machines* (SVM), *Optimum-Path Forest* (OPF) e *Convolutional Neural Networks* (CNN). O presente trabalho foi desenvolvido apenas utilizando os quatro primeiro modelos, pois todos estão disponíveis para uso na biblioteca *scikit-learn* e porque já seria utilizado outro modelo de multicamadas mais apropriado para texto que o CNN, o *BERTimbau*.

Todos os quatro modelos foram treinados por Garcia et al. (2022) [2] na configuração padrão de parâmetros. O presente trabalho, no entanto, apresenta o resultado de duas rodadas de testes: tanto a configuração padrão dos modelos, a fim de comparar os resultados com o trabalho original, quanto experimentos de ajuste de parâmetros, a fim de conseguir métricas melhores do que os modelos em sua versão padrão.

Uma importante distinção entre metodologias é necessário de ser destacada. Enquanto na representação *FastText* o número máximo de *tokens* foi 1000, não fica claro a quantidade utilizada na representação de texto BoW, por isso, decidiu-se padronizar a quantia 256 como o máximo de *tokens* para a *baseline*, para que fosse possível realizar uma comparação mais adequada com os resultados do *BERTimbau*.

Os modelos utilizados correspondem as seguintes classes do *scikit-learn*: MultinomialNB, MLPClassifier, RandomForestClassifier e SVC para *Support Vector Machines*. A segunda rodada de testes obteve os melhores resultados após 4 rodadas de NB, 14 de MLP, 9 de RF e 18 de SVM. Os parâmetros que resultaram nas melhores métricas foram:

- *Naive Bayes*: alpha=0, force_alpha=True
  - esta combinação de parâmetros retira o parâmetro de suavização[3]
- *Multi-Layer Perceptron*: hidden_layer_sizes=(256), max_iter=400, alpha=0.01
  - uma camada oculta com 256 neurônios; o algoritmo de otimização *solver* itera até convergir ou até o número máximo de 400 iterações; parâmetro *alpha* para combater *overfitting*[4]
- *Random Forest*: n_estimators=75, max_depth=64, criterion='entropy', bootstrap=False
  - número de árvores igual a 75; máxima profundidade da árvore igual a 64; função de entropia para medir qualidade do *split*; todo o *dataset* é utilizado para criar cada uma das árvores[5]
- *Support Vector Machines*: C=5
  - parâmetro de regularização[6]

2) BERTimbau: O *BERTimbau*, proposto por Souza et al. (2020) [3], é um modelo pré-treinado no idioma português. A escolha deste modelo se deve tanto ao idioma do conjunto de dados utilizado quanto ao elevado desempenho que modelos baseados em *transformers* costumam apresentar em tarefas de classificação textual.

A versão utilizada, `bert-base-portuguese-cased`[7], possui aproximadamente 110 milhões de parâmetros distribuídos em 12 camadas. Durante o treinamento, foram adotados os seguintes hiperparâmetros: *learning rate* de 2×10^{-5}, *batch size* igual a 8, número de épocas igual a 10 e *max length* (tamanho máximo da sequência de tokens considerada) igual a 256.

[2]https://pt.wikipedia.org/wiki/Tf–idf

[3]https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[4]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[5]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[6]https://scikit-learn.org/stable/auto_examples/svm/plot_svm_scale_c.html

[7]https://huggingface.co/neuralmind/bert-base-portuguese-cased

D. Métricas de Avaliação

Para a avaliação dos resultados, foram considerados seis diferentes métodos de análise. Inicia-se pela precisão, que mede a proporção de verdadeiros positivos (TP) em relação à soma de verdadeiros e falsos positivos (FP), conforme apresentado na Equação 1. Em seguida, é utilizada a sensibilidade (*recall*), responsável por medir a proporção de verdadeiros positivos em comparação à soma de verdadeiros positivos e falsos negativos (FN), conforme a Equação 2. Por fim, para equilibrar ambas as métricas, é empregada a F1-score (F_1), definida pela Equação 3.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{3}$$

Por fim, para uma avaliação mais detalhada do desempenho do modelo, realiza-se a análise da função de *loss* obtida para o melhor modelo *BERTimbau*, bem como da matriz de confusão, a fim de permitir uma visualização dos acertos e erros durante a classificação[8]. Para os modelos da *baseline*, usou-se a função log_loss do *scikit-learn*, que visa medir o quanto as probabilidades previstas se aproximam do valor real, e quanto mais próximo de zero, melhor [4]. Vale destacar que os criadores do *dataset* apresentaram a métrica acurácia (*accuracy*) invés de *loss*. A acurácia mede a proporção de previsões corretas em relação ao número total de previsões.

III. Resultados e Discussões

A. Baseline

Iniciemos a discussão a partir de os resultados obtidos por Garcia et al. (2022) [2]. Fora a CNN na representação *FastText*, o modelo que melhor performou foi o MLP na representação BoW, segundo a Figura 2, que apresenta a média da validação cruzada usando 5-fold. O classificador MLP foi superior em todas as métricas, obtendo precisão de 0.931, *recall* de 0.931, F1-score de 0.930 e acurácia de 93.1%.

classifier	precision		recall		f1-score		accuracy	
	BoW	FastText	BoW	FastText	BoW	FastText	BoW	FastText
MLP	**0.931**	**0.850**	**0.931**	**0.848**	**0.930**	**0.848**	**93.1%**	**84.8%**
NB	0.896	0.712	0.898	0.680	0.897	0.666	89.7%	68.1%
OPF	0.834	0.784	0.834	0.784	0.834	0.782	83.4%	78.4%
RF	0.924	0.840	0.922	0.840	0.922	0.840	92.3%	84.0%
SVM	0.926	0.832	0.925	0.810	0.926	0.804	92.5%	80.8%
CNN	-	**0.942**	-	**0.942**	-	**0.942**	-	**94.28%**

Figura 2: Resultados por Garcia et al. (2022) [2]

Em relação aos resultados da *baseline* em si, o *Naive Bayes* teve o desempenho mais baixo consistentemente. Em razão de as métricas obtidas serem todas muito próximas, utilizou-se o critério de menor *loss* para selecionar o melhor modelo, e, nesse caso, tanto o SVM na configuração padrão (vide Tabela II) quanto na melhor configuração de parâmetros (vide Tabela III) performaram melhor. O SVM padrão apresentou precisão de 0.9151, *recall* de 0.9244, F1-score de 0.9197 e *loss* de 0.2307. O SVM na melhor configuração encontrada, por sua vez, obteve precisão de 0.9195, *recall* de 0.9289, F1-score de 0.9242 e *loss* de 0.2342.

Tabela II: Médias das métricas de desempenho por modelo padrão

Modelo	Precisão	Recall	F1	Loss
NB	0.8363	0.8941	0.8642	0.3495
MLP	0.9196	0.9227	0.9211	0.3257
RF	0.9095	0.9114	0.9105	0.2766
SVM	0.9151	0.9244	0.9197	0.2307

Tabela III: Média das métricas de desempenho pelo melhor modelo

Modelo	Precisão	Recall	F1	Loss
NB	0.8371	0.8943	0.8647	0.3439
MLP	0.9211	0.9251	0.9230	0.2411
RF	0.9078	0.9163	0.9120	0.2543
SVM	0.9195	0.9289	0.9242	0.2342

A seguir, apresentamos as matrizes de confusão do SVM padrão na Figura 3 e do melhor SVM na Figura 4. É possível de se observar que os erros de falso negativo (FN) são os mais comuns em ambas as matrizes (lembrando que as figuras estão com as classes invertidas).

Apesar de as métricas obtidas na *baseline* serem muito próximas, os resultados originais apresentados por Garcia et al. (2022) [2] são superiores em aproximadamente 1%. Essa discrepância pode ter origem tanto em pequenas etapas do pré-processamento, quanto na quantidade máxima de *tokens* utilizadas.

B. BERTimbau

A Tabela IV apresenta os resultados obtidos para cada fold. Em todas as métricas analisadas, o Fold 1 apresentou o melhor desempenho, com precisão de 0.9819, *recall* de 0.9819, F1-score e 0.9819 e *loss* de 0.0874. Observa-se ainda o número reduzido de épocas necessárias para atingir resultados satisfatórios. Conforme ilustrado na Figura 5, todos os folds convergiram em aproximadamente três épocas, mesmo com o número máximo definido em dez.

Tabela IV: Resultados de desempenho por fold na validação cruzada (BERTimbau)

Fold	Precisão	Recall	F1	Loss
1	0.9819	0.9819	0.9819	0.0874
2	0.9724	0.9723	0.9723	0.1219
3	0.9782	0.9782	0.9782	0.1205
4	0.9726	0.9723	0.9723	0.1274
5	0.9713	0.9710	0.9710	0.1613

De modo geral, o modelo *BERTimbau* apresentou desempenho excelente na classificação de notícias falsas e verdadeiras.

[8] Aqui fazemos uma pequena observação que todas as figuras das matrizes de confusão estão apresentando as classes ao contrário

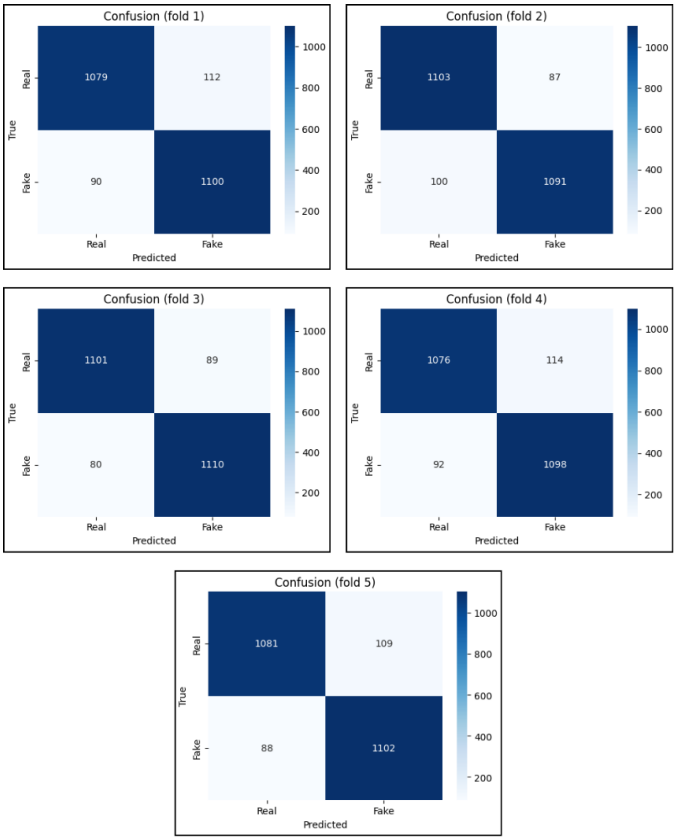

Figura 3: Matriz de confusão do SVM padrão.

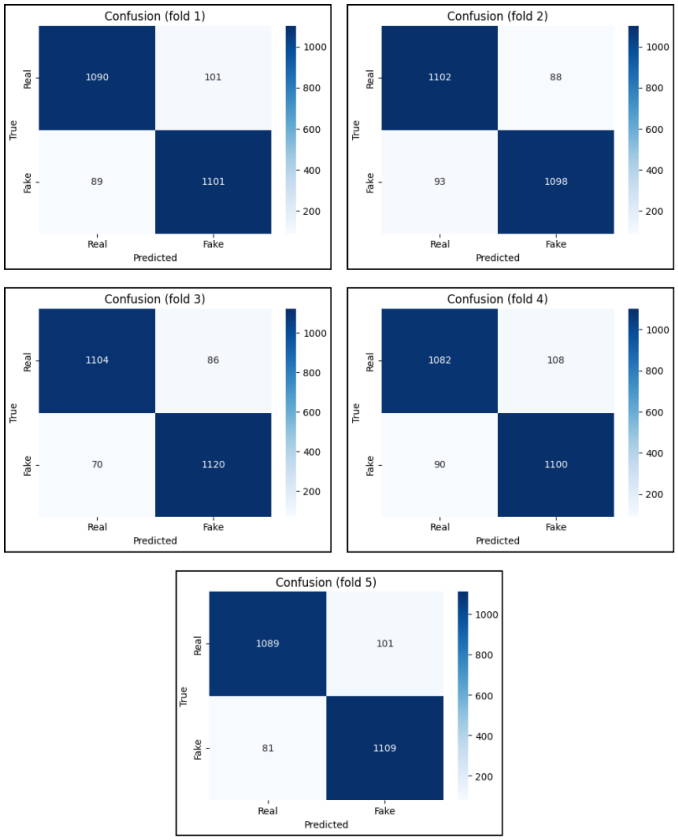

Figura 4: Matriz de confusão do melhor SVM.

Conforme a Tabela V, a precisão média foi de 0.975, o *recall* de 0.9751, o F1-score de 0.9751 e a *loss* média de 0.1237, com baixíssimo desvio-padrão entre os folds.

Tabela V: Média e desvio-padrão das métricas de desempenho (BERTimbau)

Estatística	Precisão	Recall	F1	Loss
Média	0.9753	0.9751	0.9751	0.1237
Desvio Padrão	0.0046	0.0047	0.0047	0.0263

Em relação às matrizes de confusão apresentadas na Figura 6, observa-se que as classificações mantêm um padrão consistente entre os diferentes folds, com variações pouco expressivas. As diferenças mais notáveis concentram-se nas predições incorretas, especialmente no número de falsos positivos e falsos negativos.

C. Desafios enfrentados

Foram poucos os desafios enfrentados na realização da tarefa de classificação. Entre os principais, destacam-se a definição dos hiperparâmetros a serem utilizados, a escolha de um modelo adequado e a seleção das métricas de avaliação mais relevantes.

IV. Trabalhos Relacionados

Um projeto final de curso realizado por Costa (2024) [5] utilizou o *dataset* FakeRecogna combinado ao *Corpus FAKE.BR* para treinamento de um modelo SVM com *kernel* polimonial e a representação de texto BoW. Em razão de o

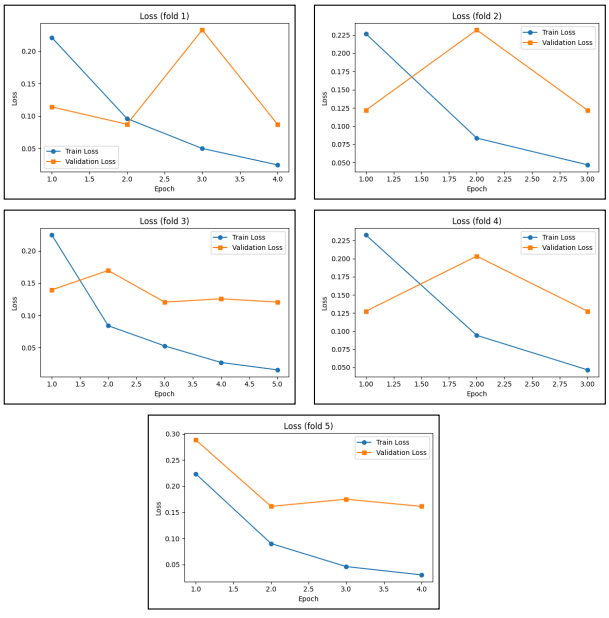

Figura 5: Curva Loss do BERTimbau.

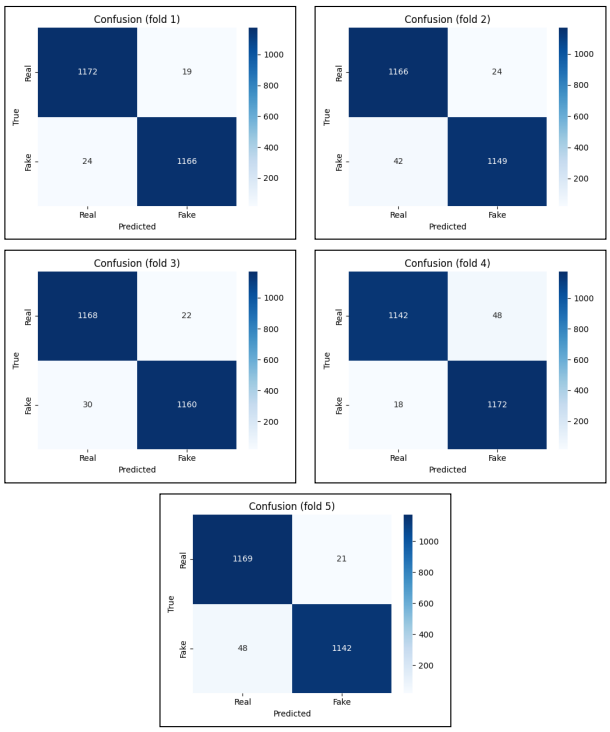

Figura 6: Matriz de confusão do BERTimbau.

dataset não ser o mesmo, uma comparação com os resultados obtidos neste trabalho não pode ser feita.

Outro trabalho de conclusão de curso (TCC), desenvolvido por Silva Júnior (2024) [6] teve como objetivo analisar separadamente o *dataset FAKE.BR* e o *FakeRecogna* em uma alta diversidade de modelos clássicos e de aprendizado profundo, verificando como os modelos performavam com ou sem *stopwords* e com ou sem *stemming* (mapear palavras ao seu radical), estratégia essa diferente da adotada pelo presente trabalho, a lematização. Os resultados do TCC podem ser conferidos na Figura 7, em que RNF é o *recall* e PNF é a precisão. O melhor resultado encontrado foi utilizando-se do modelo *BERTimbau*, com *stopwords* e uso de *stemming*. A quantidade de *tokens* máxima não foi informada pelo autor e os hiperparâmetros foram *batch size* igual a 16, *learning rate* igual a 0.00005, *epochs* igual a 10. Ele obteve *recall* de 0.9453, precisão de 0.9550, acurácia de 0.9504 e F1-score de 0.9521.

Por fim, o Pires e Silva (2024) [7] realizaram experimentos com modelos *BERT* (*BERT*, *mBERT* e *BERTimbau*) em diferentes *datasets*: *FAKE.BR, FakeTrueBR, FakeRecogna* e *Fakepedia*. O pré-processamento realizado pelos autores foi retirar *stopwords* e os hiperparâmetros foram otimizar AdamW, número máximo de *tokens* igual a 128, *learning rate* igual a 0.05, *batch size* igual a 128, com técnica de *early stopping* e 100 *epochs*. Os resultados obtidos podem ser conferidos na Figura 8. O *BERTimbau* foi o modelo com

Rank	Experimento	RNF	PNF	Acurácia	F1-Score
1	BERT, BERTIMBAU, SW, FINETUNING	0.9453	0.9550	0.9504	0.9521
2	SGDC, TF-IDF	0.9395	0.9419	0.9408	0.9407
3	LSTM, W2V, SW, FINETUNING	0.9412	0.9372	0.9391	0.9392
4	LSTM, FT, SW, FINETUNING	0.9395	0.9387	0.9391	0.9391
5	SGDC, TF-IDF, SW	0.9387	0.9394	0.9391	0.9391
6	DENSA, W2V, SW, FINETUNING	0.9370	0.9386	0.9378	0.9377
7	SVM, TF-IDF	0.9319	0.9430	0.9378	0.9374
8	SVM, TF-IDF, SW	0.9311	0.9406	0.9362	0.9358
9	SGDC, TF-IDF, SW, ST	0.9303	0.9397	0.9353	0.9350
10	LR, TF-IDF	0.9353	0.9283	0.9315	0.9318
11	LR, TF-IDF, SW, ST	0.9294	0.9341	0.9320	0.9318
12	DENSA, GV, SW, FINETUNING	0.9286	0.9349	0.9319	0.9317
13	LR, TF-IDF, SW	0.9294	0.9310	0.9303	0.9302
14	SVM, TF-IDF, SW, ST	0.9244	0.9346	0.9299	0.9294
15	RF, TF-IDF	0.9294	0.9271	0.9282	0.9282
16	LSTM, GV, SW	0.9235	0.9289	0.9265	0.9262
17	LSTM, W2V, SW	0.9159	0.9332	0.9252	0.9245
18	LSTM, FT, SW	0.9159	0.9324	0.9248	0.9241
19	LSTM, GV, SW, FINETUNING	0.8849	0.9616	0.9248	0.9216
20	DENSA, W2V, SW	0.9050	0.8968	0.9004	0.9008
21	DENSA, GV, SW	0.8891	0.9058	0.8983	0.8973
22	DENSA, FT, SW	0.9042	0.8805	0.8908	0.8922
23	SVM, GV	0.8613	0.9007	0.8832	0.8806
24	SVM, FT	0.8622	0.8969	0.8816	0.8792
25	SVM, GV, SW	0.8580	0.9004	0.8816	0.8787
26	LR, GV	0.8563	0.8946	0.8778	0.8751
27	SGDC, GV, SW	0.8395	0.9132	0.8799	0.8748
28	SGDC, FT, SW	0.8538	0.8967	0.8778	0.8747
29	LR, GV, SW	0.8563	0.8939	0.8774	0.8747
30	SVM, FT, SW	0.8597	0.8888	0.8761	0.8740
31	SVM, W2V, SW	0.8555	0.8922	0.8761	0.8734
32	SVM, W2V	0.8521	0.8942	0.8757	0.8726
33	DENSA, FT, SW, FINETUNING	0.7168	0.9827	0.8521	0.8289

Figura 7: Resultados obtidos por Silva Júnior.

melhor performance, obtendo acurácia de 0.970, precisão de 0.971, *recall* de 0.968 e F1-score de 0.969.

Comparando os trabalhos relacionados com os resultados obtidos no presente trabalho pelo modelo *BERTimbau*, as métricas de precisão, *recall* e F1-score foram consistentemente todas superiores, e o uso de lematização e número máximo de tokens como 256 são os maiores destaques a serem considerados.

Dataset	Modelo	Média ± Desvio Padrão			
		Acurácia	*Precisão*	*Recall*	*F1-score*
Fake.Br	BERT	0,897 ± 0,005	0,896 ± 0,013	0,896 ± 0,014	0,896 ± 0,009
	mBERT	0,938 ± 0,005	0,937 ± 0,013	0,939 ± 0,013	0,938 ± 0,005
	BERTimbau	**0,950 ± 0,002**	**0,955 ± 0,007**	**0,945 ± 0,009**	**0,950 ± 0,003**
FakeTrueBR	BERT	0,965 ± 0,004	0,963 ± 0,011	0,965 ± 0,007	0,964 ± 0,004
	mBERT	0,974 ± 0,006	0,973 ± 0,015	0,976 ± 0,009	0,975 ± 0,006
	BERTimbau	**0,981 ± 0,003**	**0,979 ± 0,007**	**0,984 ± 0,007**	**0,982 ± 0,003**
FakeRecogna	BERT	0,951 ± 0,004	0,949 ± 0,009	0,954 ± 0,007	0,951 ± 0,004
	mBERT	0,963 ± 0,003	0,962 ± 0,008	0,963 ± 0,008	0,963 ± 0,003
	BERTimbau	**0,970 ± 0,002**	**0,971 ± 0,006**	**0,968 ± 0,004**	**0,969 ± 0,002**
Fakepedia	BERT	0,969 ± 0,003	0,969 ± 0,006	0,969 ± 0,004	0,969 ± 0,003
	mBERT	0,981 ± 0,003	0,982 ± 0,003	0,980 ± 0,005	0,981 ± 0,003
	BERTimbau	**0,985 ± 0,002**	**0,985 ± 0,005**	**0,986 ± 0,004**	**0,985 ± 0,002**

Figura 8: Resultados obtidos por Pires e Silva.

V. Conclusão

O presente trabalho utilizou o *dataset FakeRecogna* para treinar modelos clássicos de *machine learning* (*baseline*), assim como o modelo de aprendizado profundo *BERTimbau*, ambos com o mesmo pré-processamento. Os modelos clássicos

foram treinados de modo a comparar os resultados obtidos pelo *BERTimbau* e pelos criadores do *FakeRecogna* ao utilizarem representações de texto distintas (BoW, *FastText* e TF-IDF). Os modelos de TF-IDF apresentados neste trabalho não perfomaram melhor que em BoW, uma representação mais simples. No entanto, o modelo *BERTimbau* obteve métricas excelentes, com métricas aproximadamente 5% superiores às da *baseline*, performando melhor até mesmo que trabalhos relacionados que também utilizaram-se do mesmo modelo para o mesmo *dataset*. Para trabalhos futuros, sugere-se de procurar remover outras possíveis *stopwords*, de combinar o *FakeRecogna* com outras bases ou performar análises por categorias de notícias.

REFERÊNCIAS

[1] 2ª Vice-Presidência do TJPR, "O perigo das fake news," 06 2020. Disponível em: https://www.tjpr.jus.br/noticias-2-vice/-/asset_publisher/sTrhoYRKnlQe/content/o-perigo-das-fake-news/14797.

[2] G. Garcia, L. Afonso, and J. Papa, *FakeRecogna: A New Brazilian Corpus for Fake News Detection*, pp. 57–67. 03 2022.

[3] F. Souza, R. Nogueira, and R. Lotufo, "BERTimbau: pretrained BERT models for Brazilian Portuguese," in *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020.

[4] M. Filho, "Guia completo da log loss (perda logarítmica) em machine learning," 2023. Disponível em: https://mariofilho.com/guia-completo-da-log-loss-perda-logaritmica-em-machine-learning/.

[5] L. A. Costa, "Uma ferramenta baseada em aprendizado de máquina para detecção de notícias falsas," 2024. Disponível em: https://bdm.unb.br/bitstream/10483/41009/1/2024_LucasAquinoCosta_tcc.pdf.

[6] F. J. d. Silva Júnior, "Detecção de notícias falsas usando técnicas de aprendizado de máquina e aprendizado profundo," 2024. Disponível em: https://repositorio.ufc.br/bitstream/riufc/79291/1/2024_tcc_fjsilvajunior.pdf.

[7] V. Pires and D. G. e Silva, "Portuguese fake news classification with bert models," in *Anais do XXI Encontro Nacional de Inteligência Artificial e Computacional*, (Porto Alegre, RS, Brasil), pp. 834–845, SBC, 2024.