Análise de Desempenho de Modelos de Embeddings Multilínguas na Classificação de Notícias Falsas

Alison I. O. de Moura[1], **Bruno M. Nogueira**[1]

[1]Faculdade de Computação - Universidade Federal de Mato Grosso do Sul (UFMS)

alison.moura@ufms.br, bruno@facom.ufms.br

Abstract. *The fake news classification introduces some challenges for Natural Language Processing (NLP). Other non-textual representations, such as text embedding, can influence the performance of those classifications. The goal of this work is to evaluate these performances on two corpus databases in the context of fake news classification. Five different word embedding algorithms were applied, and subsequently, the data were classified by four classification algorithms. Our study demonstrates satisfactory performance for the representations, with the Multilingual-E5-large algorithm standing out with superior performance compared to the other tested models. This work may contribute to understanding the influence of multilingual word embedding algorithms on the performance of text classification algorithms. The experimental results can serve as a performance reference to guide the use of these models for text classification in similar studies.*

Resumo. *A classificação de notícias falsas introduzem alguns desafios para o Processamento de Linguagem Natural (PLN). Outras representações não-textuais, como text embedding podem influenciar o desempenho das classificações. O objetivo deste trabalho é avaliar esses desempenhos em duas bases de dados textuais, no contexto de classificação de notícias falsas, onde foram aplicados cinco algoritmos de word embedding diferentes e posteriormente classificados por quatro algoritmos de classificação. Nosso trabalho apresenta um desempenho satisfatório para as representações, com destaque do algoritmo Multilingual-E5-large com performance superior aos demais modelos testados. Este trabalho pode contribuir para o entendimento da influência dos algoritmos de word embedding multilíngue no desempenho dos algoritmos de classificação de textos. Os resultados experimentais podem servir como uma referência de desempenho para guiar a utilização desses modelos para classificação de textos em trabalhos similares.*

1. Introdução

Nos últimos anos, tem se observado uma tendência a representações de linguagem pré-treinadas de sistemas de processamento de linguagem natural (PLN) [Brown et al. 2023]. A técnica tem ganhado popularidade, em uma variedade de domínios, incluindo: semântica e linguística, correção automática de texto, *Named Entity Recognition* (NER), bibliometria, cibersegurança, mecânica quântica, química, estudo de gêneros ou ortodontia [Sawicki et al. 2023].

Word embedding ou *text embedding* é uma repressentação vetorial de baixa-dimensão para textos de tamanhos arbitrários e possui um papel-chave em muitas tarefas

de PLN [Wang et al. 2022]. Essas representações podem ser dividas entre contextualizada e não-contextualizada. A diferença entre as duas é a análise ou não do contexto da palavra, um exemplo é a palavra BERT, que para um contexto televisivo infantil significa um personagem. Já no contexto de NLP é modelo de *embbeding* de texto [Sawicki et al. 2023].

O objetivo deste trabalho é avaliar o desempenho de algoritmos de classificação de textos baseados em duas bases de dados textuais, no contexto de notícias falsas na língua portuguesa. Dado a complexidade e desafios na classificação de notícias falsas [Souza et al. 2021], a hipótese que queremos testar é se os algoritmos de *word embedding* terão desempenho satisfatório nesse contexto, e se existirá alguma diferença significativa de desempenho entre eles.

Para realização dos teste, foram aplicados cinco algoritmos de *word embedding* diferentes. Os resultados foram utilizados para realização do treinamento de quatro classificadores distintos, e posteriormente, a coleta de métricas do desempenho das classificações também foi realizada. Os resultados coletados nos experimentos deste trabalho apontam para a viabilidade do uso dessas representações para a classificação de notícias falsas. Sendo o modelo de *embedding* Multilingual-E5-large, que apresentou 89,19% de média de acurácia, o modelo com o melhor desempenho performado no trabalho.

A seção 2 deste artigo tem como objetivo discorrer sobre a metodologia experimental utilizada. Nela será discutida quais bases de dados foram utilizadas, seu tamanho e atributos. Também serão apresentados os algoritmos de *word embedding* utilizados. Por fim, também será apresentado quais classificadores foram utilizados, e todo o processo de treinamento, teste e validação.

A seção 3 deste artigo apresenta os resultados e algumas interpretações dos autores sobre. Serão apresentados os valores coletados ao longo do trabalho, exibidos em gráficos de barras, com seus valores e desvio padrão.

Por fim, a seção 4 destina-se à discussão dos resultados coletados, dificuldades enfrentadas ao longo do trabalho, possíveis melhorias e trabalhos futuros que podem ser realizados.

2. Metodologia Experimental

Nesta seção, descrevemos a metodologia experimental utilizada neste trabalho. Abordaremos quais bases de dados textuais foram utilizadas, e como foram estruturadas. Também abordaremos os algoritmos de *word embeddings* e o processo da transformação textual (*text transform*). Por fim, discorreremos sobre o processo de classificação, utilizando os dados do passo anterior, e também dos algoritmos utilizados e métricas coletadas.

2.1. Bases de Dados

Primeiramente, sobre as bases de dados, foram utilizados dois *datasets* textuais para os experimentos. Ambas bases contém amostras de texto (*corpus*) de notícias falsas e notícias verdadeiras, a Tabela 1 exibe as características e distribuição dos dados de cada uma. O processo, no geral, consiste basicamente de: importação da base, execução do processo de transformação de texto com os algoritmos de *word embbeding*; e treinamento

e coleta de métricas dos algoritmos de classificação. Esse procedimento foi executado separadamente para cada base, que serão descritas com mais detalhes abaixo:

A primeira base de dados **Fake.BR (FBR)** foi a primeira base de texto para detecção de *fake news* na língua portuguesa [Silva et al. 2020]. A segunda base **Fact-checked news (FCN)** foi coletado a partir de cinco sites de verificação de notícias brasileiras [Souza et al. 2021]. Os textos estavam inseridos em arquivos em formato TSV, ou seja, valores separados por tabulação.

Base de dados	**Tema**	**Qtd. de notícias falsas**	**Qtd. de notícias verdadeiras**
FBR	Política, tecnologia, cotidiano, celebridades, economia, religião	3.600	3.600
FCN	Política	1.044	1.124

Table 1. Informações detalhadas sobre as base de dados

A importação da base foi realizada no ambiente Google Colab com *runtime* Tesla V100-SXM2-16GB, utilizando Python 3.10 com a biblioteca Pandas. Logo após a importação, as colunas foram renomeadas para: *file_name*, *text* e *toxic*, representando respectivamente: nome/identificação do texto, a sentença (*corpus*) em si e, por fim, a classificação (*label*) do texto.

Após realização dos passos anteriores, com a base de dados importada e já pronta para uso, o passo seguinte foi a execução do processo de transformação textual. Para essa etapa foram utilizados cinco algoritmos de *word embedding* sendo eles:

- Multilingual-E5-base: *text embedding* por pré-treinamento contrastivo fracamente supervisionado, contendo 12 camadas e tamanho 786 unidades (*embedding*) [Wang et al. a].
- Multilingual-E5-large: *text embedding* por pré-treinamento contrastivo fracamente supervisionado, contendo 24 camadas e tamanho 1024 unidades (*embedding*) [Wang et al. b]
- Multilingual-E5-small: *text embedding* por pré-treinamento contrastivo fracamente supervisionado, contendo 12 camadas e tamanho 384 unidades (*embedding*) [Wang et al. c]
- Distiluse-base-multilingual-cased-v2: um modelo transformador de sentenças (*sentence transform*), mapeia sentenças e parágrafos para um denso espaço vetorial de 512 dimensões [Reimers and Gurevych] e [Reimers and Gurevych 2019].
- Text2vec-base-multilingual: é um modelo CoSENT(Cosine Sentence), mapeia sentenças para um denso espaço vetorial de 384 dimensões [Xu] e [Xu 2023].

2.2. Multilingual-E5

O algoritmo de *word embedding* Multilingual-E5 é um modelo de linguagem com pré-treinamento contrastivo que representa palavras como vetores de números reais. Esses

vetores são projetados para capturar a relação semântica entre as palavras, ou seja, a semelhança de significado entre elas. O método visa prover *text embedding* pronto para uso para quaisquer tarefas que requerem representações de vetor único com ou sem *fine tunning*. Para isso, ao invés de ter uma dependência em um conjunto de dados rotulados ou pares sintéticos de texto de baixa qualidade, foi utilizado um métodos contrastivo para o treinamento. Para o treinamento foi utilizado o *dataset* CCPairs (*Colossal Clean text Pairs*) uma larga coleção de pares de texto não rotulados [Wang et al. 2022].

A principal diferença entre os modelos Multilingual-E5-base, Multilingual-E5-small e Multilingual-E5-large é o tamanho do modelo. A exemplo, o Multilingual-E5-base possui 12 camadas de transformadores e 768 unidades de processamento por camada. O Multilingual-E5-small tem 12 camadas de transformadores e 384 unidades de processamento por camada. O Multilingual-E5-large tem 24 camadas de transformadores e 1024 unidades de processamento por camada [Wang et al. 2022].

O pré-treinamento do algoritmo foi realizado em três modelos de tamanhos: E5_{small}, E5_{base} e E5_{large}, inicializado a partir do MiniLM [Wang et al. 2020], *bert-base-uncased* e *bert-large-uncased-whole-word-masking* respectivamente.

2.3. Distiluse

O algoritmo de *word embedding* Distiluse-base-multilingual é um modelo de *sentence transform* pré-treinado em mais de 50 diferentes idiomas [Reimers and Gurevych 2020a]. O algoritmo foi treinado usando uma abordagem de destilação de conhecimento multilíngue, onde um modelo novo "aluno" destila o conhecimento de um modelo "professor" [Reimers and Gurevych 2020a].

Assim como o Multilingual-E5-base, o Distiluse-base-multilingual é composto por 12 camadas de transformadores, cada uma com 768 unidades de processamento. As unidades de processamento são responsáveis por calcular a relação entre as palavras em uma sentença ou frase [Sanh et al. 2019].

2.4. Text2vec

Text2vec-base-multilingual é um modelo CoSENT(*Cosine Sentence*) que mapeia sentenças para um vetor espacial de 384 dimensões e pode ser utilizado para atividades de *sentence embedding*, *text matching* ou *semantic search* [Xu]. Para o pré-treinamento, foi utilizado o modelo pré-treinado MiniLM [Wang et al. 2020] e ajustes finos foram realizados usando um objetivo contrativo e computando a similaridade de cosseno [Xu].

2.5. Importação dos Modelos e Transformação de Texto

Todos os algoritmos utilizados, são modelos pré-treinados disponibilizados pelo repositório de modelos da biblioteca *SentenceTransformers* [Reimers and Gurevych 2019] e [Reimers and Gurevych 2020b], chamado *Hugging Face*. Na plataforma, é possível encontrar mais de 500 modelos adicionais, além dos já disponibilizados oficialmente pela biblioteca. Nela os autores informam detalhes do funcionamento, citam pesquisas e disponibilizam o modelo. Também é possível criar novos modelos e publicá-los, assim como os próprios *embeddings* obtidos.

Para a transformação do texto, o primeiro passo foi carregar o modelo pré-treinado, utilizando o *SentenceTransform*. Feito isso, aplicamos o atributo *text* da base

de dados, que contém os *corpus*, ao modelo, fazendo com que todo o texto original passasse pelo processo de *word embedding*. Como resultado dessa execução, o algoritmo nos retorna o resultado, ou seja, o texto com a trasformação já aplicada. O resultado foi armazenado para utilização posterior, durante a classificação.

2.6. Classificação

Após a transformação dos textos em representações numéricas, utilizando os algoritmos de *word embedding*, a etapa seguinte foi a classificação. Para isso, foram utilizados quatro algoritmos de classificação: *Logistic Regression*, *Support Vector Machines*, *Decision Trees* e *Multi-Layer Perceptron*.

A metodologia foi a mesma aplicada para cada algoritmo, onde foi feito a otimização de hiper-parâmetros utilizando *GridSearch* [ScikitLearn g] com a acurácia como *scoring* do processo (as informações de hiper-parâmetros estão descritos na Tabela 2). Para o treinamento, as bases de dados foram separadas (em treinamento e teste) utilizando os índices fornecidos pelo *KFold* [ScikitLearn c] e posteriormente aplicando o *Cross Validation* [ScikitLearn a] para realização da coleta das métricas. Ambos as funcionalidades são fornecidos pela biblioteca *Scikit Learn*.

As rodadas foram executadas utilizando as configurações supracitadas, em cada iteração foi coletado os valores de: acurácia (*accuracy*), precisão (*precision*) e revocação (*recall*), como métricas do treinamento. Para cada *embbeding*, o processo de treinamento e coleta foi executado em iterações de 5-*fold* do *KFold* para cada um dos quatro algoritmos de classificação. E esse processo foi repetido para execução de cada base, ou seja, duas execuções.

Logistic Regression utiliza um modelo linear de regressão logística para classificação dos dados [ScikitLearn d]. *Support Vector Machines* (SVM) cria um hiperplano de decisão ótimo que separa as instâncias de diferentes classes no espaço de características [ScikitLearn f]. **Decision Trees**: utiliza-se de uma estrutura de árvore de decisão para executar tarefas de classificação e regressão, com aprendizado supervisionado [ScikitLearn b]. **Multi-Layer Perceptron (MLP)** cria uma rede neural, composto por camadas de neurônios que se comunicam entre si [ScikitLearn e]. Para a otimização dos hiper-parâmetros com *GridSearch* os seguintes parâmetros e valores foram configurados, como exposto na tabela abaixo:

3. Discussão dos Resultados

Nessa seção iremos apresentar e discutir os resultados obtidos pelo procedimento descrito na seção anterior. Será apresentado os resultados para cada algoritmo de *word embedding* contendo as médias de acurácia, precisão e *recall* respectivamente para cada classificador.

3.1. *Fake.BR* (FBR)

Começando pela primeira base de dados ***Fake.BR***, iremos analisar as métricas de desempenho de cada algoritmo de classificação para as cinco estratégias de *web embedding*. As métricas coletadas (acurácia, precisão e revocação) serão exibidas em formato de gráfico de barras com o seus respectivos desvios-padrões, e também seus valores

	Parâmetro	Valores
Logistic Regression		
	solver	lbfgs, newton-cg, newton-cholesky, sag, saga
	multi_class	auto, ovr, multinomial
SVM		
	kernel	linear, poly, rbf, sigmoid
	gamma	scale, auto
	decision_function_shape	ovo, ovr
Decision Trees		
	criterion	gini, entropy, log_loss
	splitter	best, random
MLP		
	activation	identity, logistic, tanh, relu
	solver	lbfgs, sgd, adam
	learning_rate	constant, invscaling, adaptive

Table 2. Hiperparâmetros utilizando no *GridSearch*

contínuos exibidos na Tabela 3, para cada classificador. Ao final da seção discutiremos os resultados.

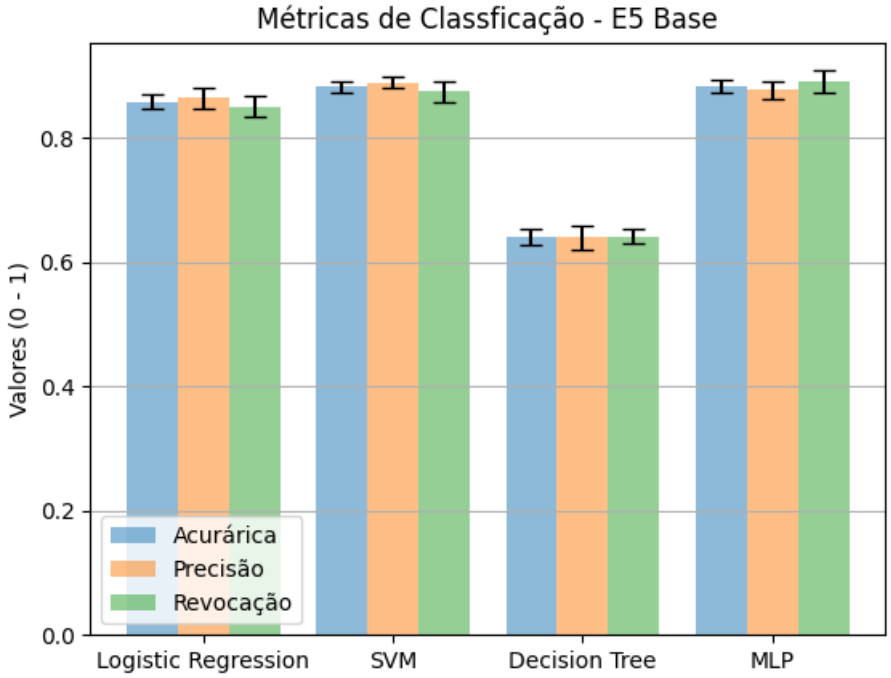

Figure 1. Métricas de classificação - Multilingual-E5-base - FBR

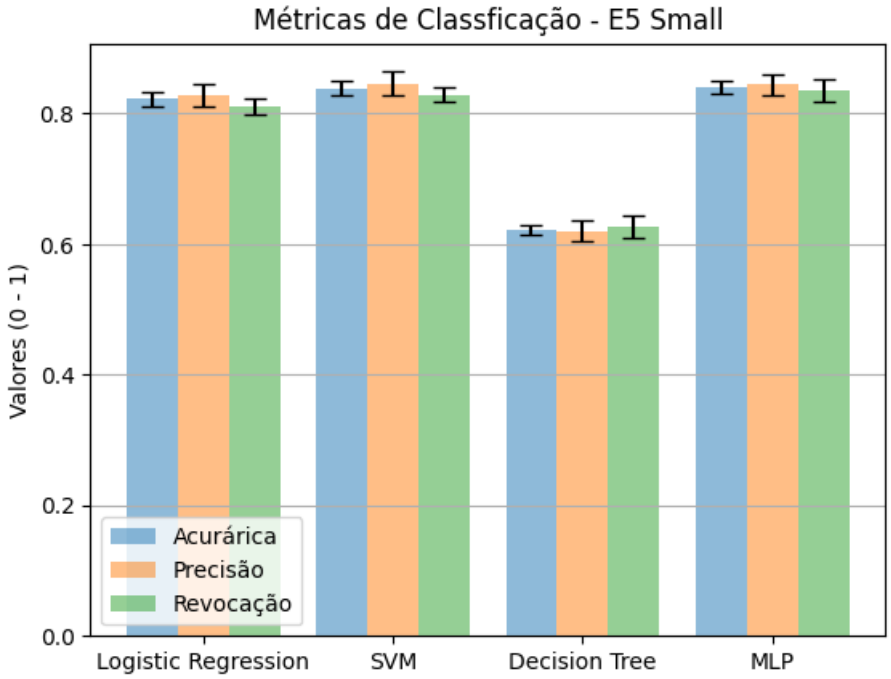

Figure 2. Métricas de classificação - Multilingual-E5-small - FBR

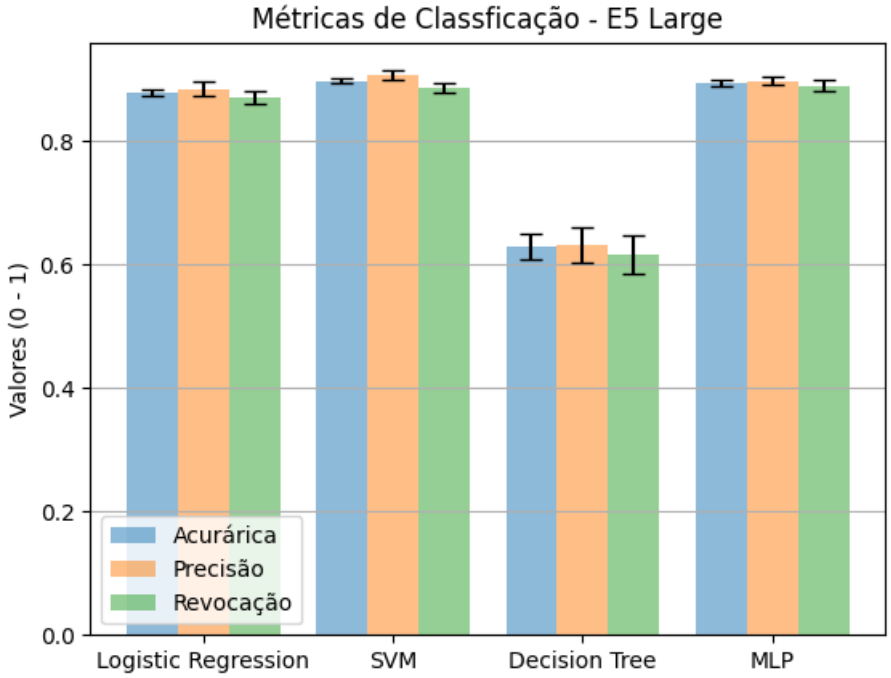

Figure 3. Métricas de classificação - Multilingual-E5-large - FBR

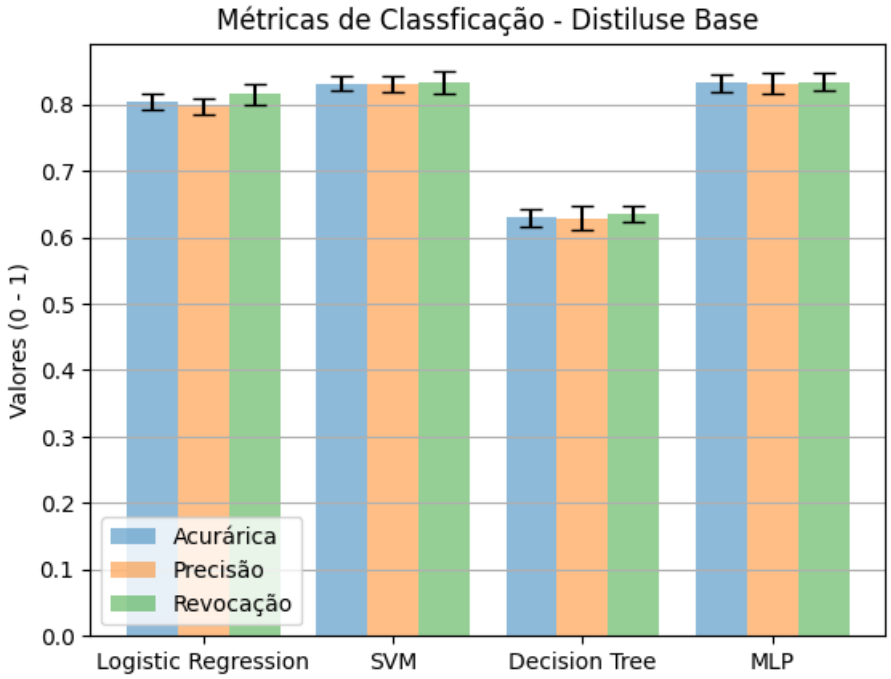

Figure 4. Métricas de classificação - Distiluse-base - FBR

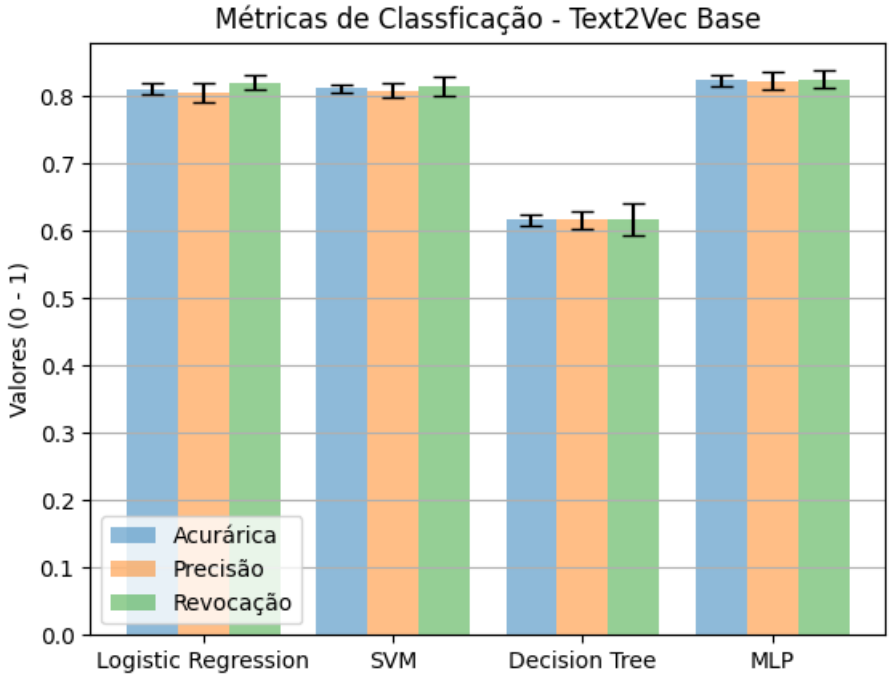

Figure 5. Métricas de classificação - Text2Vec-base - FBR

	Acurácia	Precisão	Revocação
Multilingual-E5-base			
Logistic Regression	0.8583	0.8641	0.8503
SVM	0.8823	0.8887	0.8742
DecisionTrees	0.6402	0.6399	0.6414
MLP	0.8827	0.8769	0.8903
Multilingual-E5-small			
Logistic Regression	0.8221	0.8289	0.8119
SVM	0.8389	0.8461	0.8286
DecisionTrees	0.6216	0.6203	0.6278
MLP	0.8409	0.8448	0.8356
Multilingual-E5-large			
Logistic Regression	0.8776	0.8834	0.8698
SVM	0.8969	0.9059	0.8857
DecisionTrees	0.6283	0.6317	0.6154
MLP	0.8929	0.8963	0.8886
Distiluse-base			
Logistic Regression	0.8042	0.7977	0.8153
SVM	0.8316	0.8308	0.8329
DecisionTrees	0.6301	0.6291	0.6350
MLP	0.8325	0.8318	0.8341
Text2Vec-base			
Logistic Regression	0.8107	0.8049	0.8204
SVM	0.8111	0.8087	0.8151
DecisionTrees	0.6160	0.6161	0.6167
MLP	0.8234	0.8227	0.8246

Table 3. Métricas de classificação por *word embedding* - FBR

Podemos verificar que a grande maioria dos algoritmos de classificação tiveram um bom desempenho no geral, com métricas de acurácia, precisão e revocação próximos de 80%. A exceção seria apenas o algoritmo *Decision Trees* (Árvores de Decisão), onde suas métricas ficaram mais próximas de 60%, um possível indicativo que o modelo não conseguiu aprender o nessário para alcançar a métrica dos outros modelos de classificação. Vale a pena destacar também o desempenho dos algoritmos SVM (*Support Vector Machines*) e *Decision Trees*.

Com relação aos algoritmos de *word embedding* não vimos nenhuma variação significativa nas métricas, porém os resultados também não indicam uma influência negativa no desempenho dos classificadores, indicando uma possível viabilidade no uso para esta base de dados. Como destaque vimos que o modelo Multilingual-E5-large propiciou métricas levemente maiores nas classificações.

3.2. *Fact-checked news* (FCN)

Na segunda base de dados ***Fact-checked news***, durante a execução dos algoritmos de classificação, tivemos os seguintes resultados por cada processo de transformação de texto (*word embedding*). As métricas coletadas (acurácia, precisão e revocação) serão exibidas em formato de gráfico de barras com o seus respectivos desvios-padrões, e também seus valores contínuos exibidos em tabela, para cada classificador. Ao final da seção discutiremos os resultados.

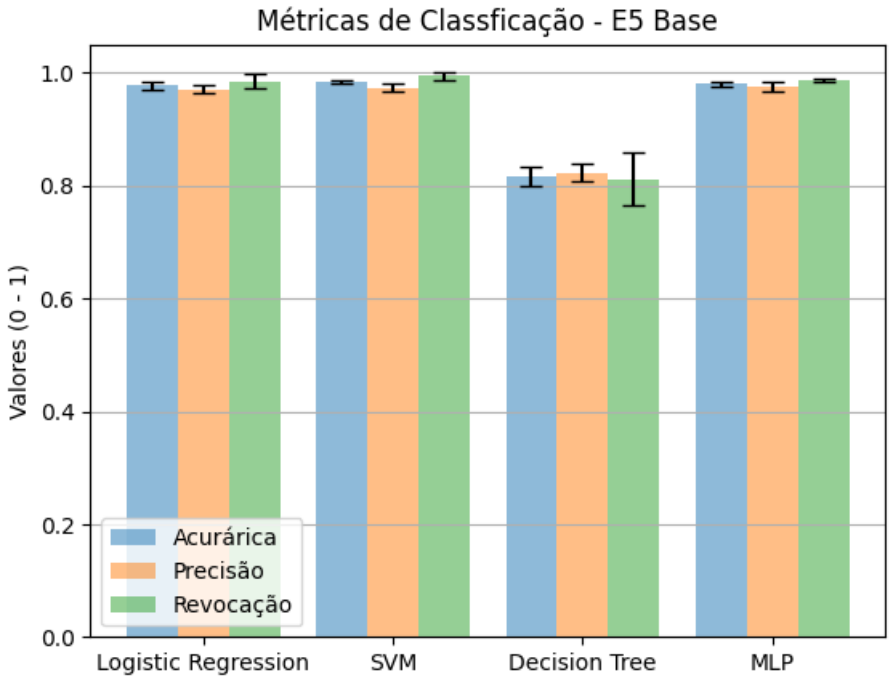

Figure 6. Métricas de classificação - Multilingual-E5-base - FCN

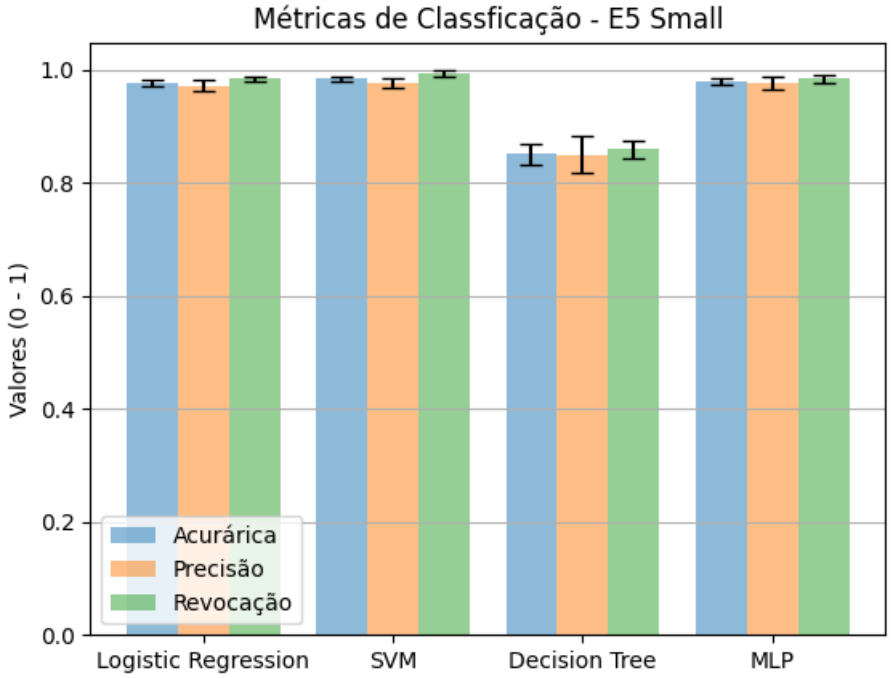

Figure 7. Métricas de classificação - Multilingual-E5-small - FCN

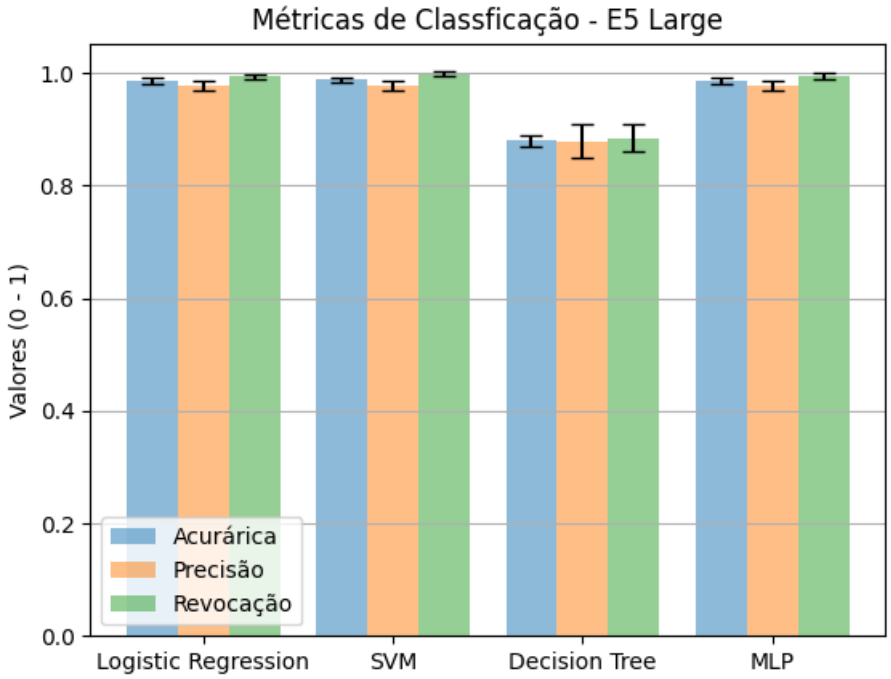

Figure 8. Métricas de classificação - Multilingual-E5-large - FCN

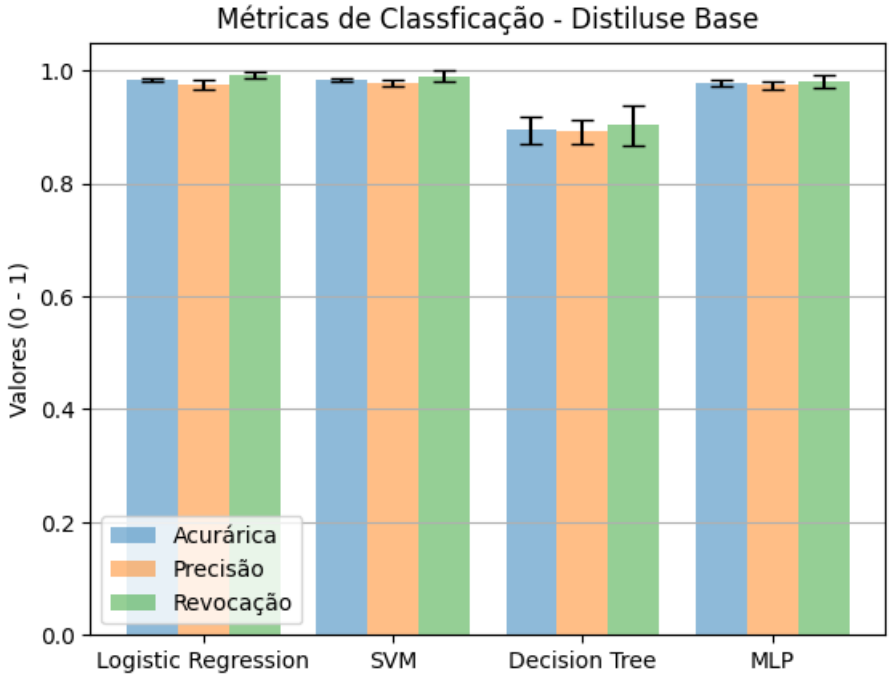

Figure 9. Métricas de classificação - Distiluse-base - FCN

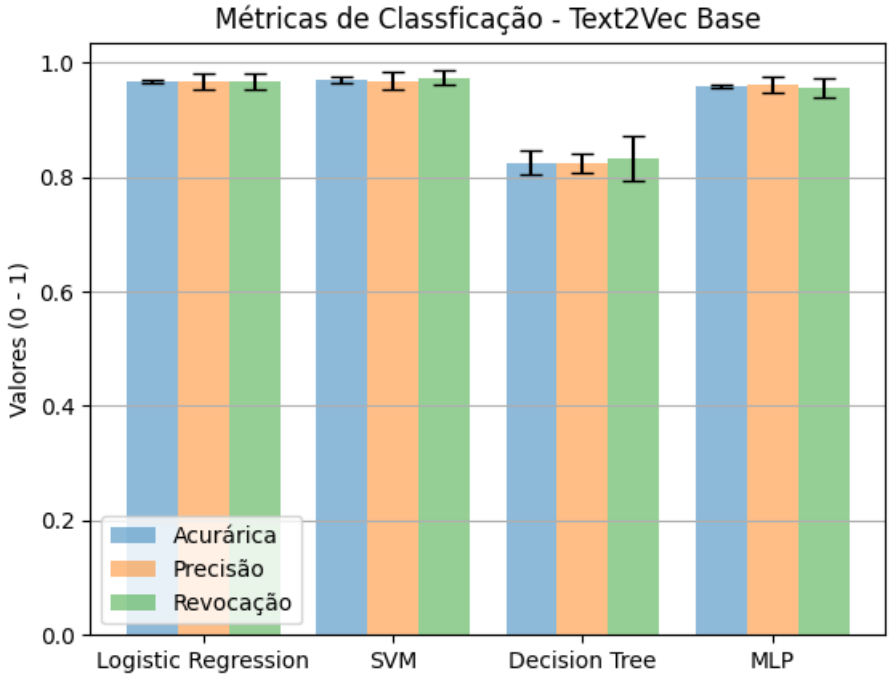

Figure 10. Métricas de classificação - Text2Vec-base - FCN

	Acurácia	Precisão	Revocação
Multilingual-E5-base			
Logistic Regression	0.9772	0.9706	0.9850
SVM	0.9835	0.9737	0.9944
DecisionTrees	0.8169	0.8234	0.8121
MLP	0.9801	0.9753	0.9857
Multilingual-E5-small			
Logistic Regression	0.9772	0.9715	0.9838
SVM	0.9845	0.9765	0.9935
DecisionTrees	0.8517	0.8502	0.8591
MLP	0.9806	0.9772	0.9848
Multilingual-E5-large			
Logistic Regression	0.9859	0.9782	0.9942
SVM	0.9879	0.9783	0.9980
DecisionTrees	0.8798	0.8794	0.8846
MLP	0.9860	0.9782	0.9941
Distiluse-base			
Logistic Regression	0.9835	0.9756	0.9923
SVM	0.9835	0.9773	0.9904
DecisionTrees	0.8953	0.8917	0.9028
MLP	0.9777	0.9745	0.9817
Text2Vec-base			
Logistic Regression	0.9661	0.9664	0.9665
SVM	0.9700	0.9678	0.9733
DecisionTrees	0.8251	0.8243	0.8322
MLP	0.9578	0.9619	0.9550

Table 4. Métricas de classificação por *word embedding* - FCN

Podemos verificar que os valores das métricas, no geral, ficaram levemente maiores com os dados da base **Fact-checked-news.tsv** comparadas a base **Fakebr.tsv**. Algumas hipóteses podem ser levantadas, como: a segunda base de dados possui dados mais refinados, ou o tema das notícias coletadas foi mais especialista.

Adicionalmente, podemos observar também que, assim como na primeira base de dados, aqui o classificador *Decision Trees* (Árvores de Decisão) também teve um desempenho menor, comparado com os outros classificadores.

Sobre os algoritmos de *word embedding* podemos repetir nossas considerações levantadas anteriormente. Os modelos não representaram uma grande diferença entre si, porém as métricas de classificação, no geral, foram satisfatórias e o processo de transformação textual não influenciou negativamente os resultados dos classificadores.

3.3. Resultados Gerais

Para termos uma visão global do desempenho dos algoritmos, tanto os *word embeddings* quanto os classificadores, vamos realizar a média de acurácia dos algoritmos de classificação, somando as métricas de ambos resultados das duas bases, em duas visões: **média de acurácia por *word embedding*** e **média de acurácia por classificador**.

O gráfico abaixo exibe a média de acurácia por *word embedding*, onde podemos ver que nessa classificação o algoritmo de *embedding* **Multilingual-E5-large** teve a melhor média de desempenho:

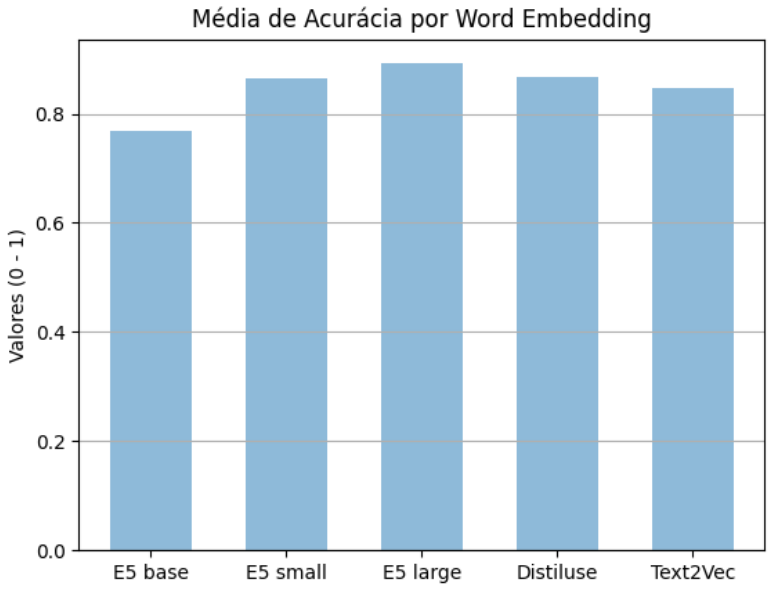

Figure 11. Média de Acurácia por Word Embedding

Em seguida, o gráfico abaixo exibe a média de acurácia por algoritmo de classificação, onde podemos ver que nesse cenário o classificador **Support Vector Machines (SVM)** teve a melhor média de desempenho:

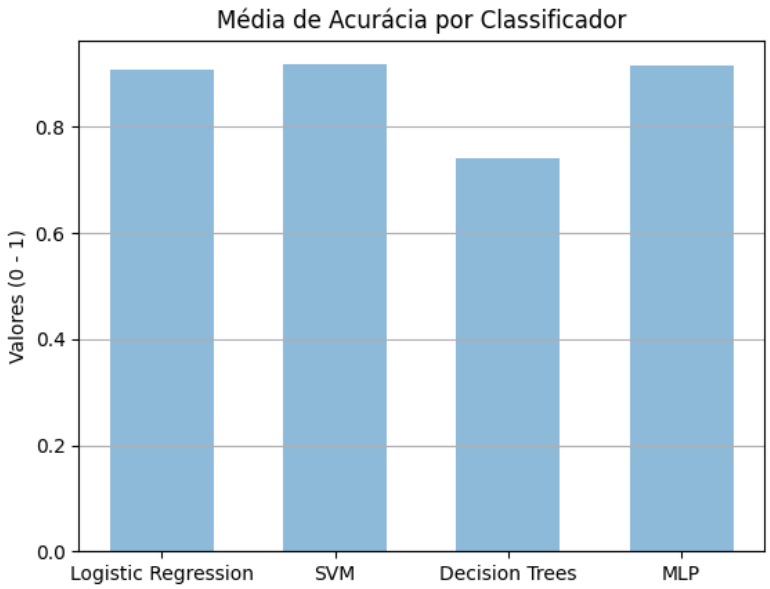

Figure 12. Média de Acurácia por Classificador

4. Conclusão

Antes da conclusão final deste trabalho, algumas considerações serão descritas na sequência, com relatos de problemas e decisões tomadas durante a execução dos experimentos e na posteriormente a conclusão final do artigo será apresentada.

Durante a execução dos experimentos, algumas dificuldades foram encontradas, uma delas está relacionada ao tempo de execução demandado para o treinamento dos algoritmos. Para execução do treinamento, tanto dos algoritmos de *word embedding* quanto os classificadores, foi demandado mais de 10 horas de execução, para ambas as bases de dados, sem levar em considerações retrabalhos ao longo das execuções. O tempo teve uma redução significativa após o uso da *runtime* Tesla V100-SXM2-16GB, porém o consumo de unidades computacionais também é bem elevado.

Outra consideração importante diz respeito a alguns algoritmos que sua utilização foi planejada, porém impedimentos técnicos e limitações de tempo, fizeram com que ficassem fora dos experimentos do trabalho realizado. Dois modelos de *word embedding*, XLM-RoBERTa-base [Conneau et al. a] e XLM-RoBERTa-large [Conneau et al. b] introduzidos pelo *paper* [Conneau et al. 2019], não foram utilizados nos experimentos pois foram apresentados erros durante a execução do processo de transformação textual. Um algoritmo de classificação XGBoost (Extreme Gradient Boost) [XGBoost] também não foi incluso nos experimentos desse trabalho pelo mesmo motivo do XLM-RoBERTa.

Como podemos observar, pelos resultados apresentados durante os experimentos deste trabalho, concluímos que os algoritmos de *Word Embedding Multilingual* apresentados podem ser utilizados na classificação de notícias falsas, sem ônus aos resultados obtidos. Os resultados também mostram que *embeddings* com pré-treinamento utilizando lotes de texto maiores (ex: Multilingual-E5-large) podem levar a resultados levemente superiores. Porém é válido considerar o fato de que o tamanho do modelo também será maior, e deve ser analisado conforme o contexto da utilização.

É válido considerar o fato do trabalho ter utilizado apenas modelos de *embbedings* multilíngue, ou seja, com pré-treinamento em uma variedade de línguas do mundo, diferentemente de modelos especializados em uma língua apenas. *Word Embeddings* especializados podem apresentar desempenhos diferentes, porém não foi objeto de experimentação deste trabalho, restando aqui um incentivo para trabalhos futuros.

Em conclusão, este estudo contribui para a compreensão experimental do desempenho de algoritmos de classificação em dados textuais, com utilização de cinco técnicas de *Word Embedding Multilingual* aplicados em dados para classificação de notícias falsas. Os resultados obtidos proporcionaram *insights* sobre a capacidade discriminativa e generalização desses modelos, além da viabilidade de sua utilização para o domínio proposto do trabalho.

Diante disso, este trabalho expões os resultados práticos, da utilização dos classificadores e a influência que o processo de transformação textual (*Word Embedding Multilingual*) exerce no desempenho das classificações.

References

[Brown et al. 2023] Brown, T. B., Mann, B., Ryder, N., Kaplan, M. S. J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Ariel, S. A., Herbert-Voss, Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2023). Language models are few-shot learners. *arXiv:2005.14165v4 [cs.CL]*.

[Conneau et al. a] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. Xlm-roberta (base-sized model). `https://huggingface.co/xlm-roberta-base`. Accessed: 2023-11-25.

[Conneau et al. b] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. Xlm-roberta (large-sized model). `https://huggingface.co/xlm-roberta-large`. Accessed: 2023-11-25.

[Conneau et al. 2019] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

[Reimers and Gurevych] Reimers, N. and Gurevych, I. Distiluse-base-multilingual-cased-v2 - hugging face. `https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v2`. Accessed: 2023-11-23.

[Reimers and Gurevych 2019] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[Reimers and Gurevych 2020a] Reimers, N. and Gurevych, I. (2020a). Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv:2004.09813 [cs.CL] 21 Apr 2020*.

[Reimers and Gurevych 2020b] Reimers, N. and Gurevych, I. (2020b). Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[Sanh et al. 2019] Sanh, V., Debut, L., Chaumond, J., and Dozat, T. (2019). Distilbert: A distilled version of bert for natural language processing. *arXiv preprint arXiv:1910.01108*.

[Sawicki et al. 2023] Sawicki, J., Ganzha, M., and Paprzycki, M. (2023). The state of the art of natural language processing – a systematic automated review of nlp literature using nlp techniques.

[ScikitLearn a] ScikitLearn. Cross-validation: evaluating estimator performance. `https://scikit-learn.org/stable/modules/cross_validation.html`. Accessed: 2023-11-23.

[ScikitLearn b] ScikitLearn. Decision trees. `https://scikit-learn.org/stable/modules/tree.html#decision-trees`. Accessed: 2023-11-23.

[ScikitLearn c] ScikitLearn. Kfold documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html#sklearn.model_selection.KFold`. Accessed: 2023-11-24.

[ScikitLearn d] ScikitLearn. Logistic regression. `https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`. Accessed: 2023-11-23.

[ScikitLearn e] ScikitLearn. Multi-layer perceptron. `https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron`. Accessed: 2023-11-23.

[ScikitLearn f] ScikitLearn. Support vector machines. `https://scikit-learn.org/stable/modules/svm.html#support-vector-machines`. Accessed: 2023-11-23.

[ScikitLearn g] ScikitLearn. Tuning the hyper-parameters of an estimator. `https://scikit-learn.org/stable/modules/grid_search.html#grid-search`. Accessed: 2023-11-23.

[Silva et al. 2020] Silva, R. M., Santos, R. L. d. S., Almeida, T. A., and Pardo, T. A. S. (2020). Towards automatically filtering fake news in portuguese. *Expert Systems with Applications*.

[Souza et al. 2021] Souza, M., Nogueira, B., Rossi, R., Marcacini, R., and Rezende, S. (2021). A heterogeneous network-based positive and unlabeled learning approach to detect fake news. In *Anais da X Brazilian Conference on Intelligent Systems*, Porto Alegre, RS, Brasil. SBC.

[Wang et al. a] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Zhang, W., and Wei, F. Multilingual-e5-small - hugging face. `https://huggingface.co/intfloat/multilingual-e5-base`. Accessed: 2023-11-23.

[Wang et al. b] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Zhang, W., and Wei, F. Multilingual-e5-small - hugging face. https://huggingface.co/intfloat/multilingual-e5-large. Accessed: 2023-11-23.

[Wang et al. c] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Zhang, W., and Wei, F. Multilingual-e5-small - hugging face. https://huggingface.co/intfloat/multilingual-e5-small. Accessed: 2023-11-23.

[Wang et al. 2022] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Zhang, W., and Wei, F. (2022). Text embeddings by weakly-supervised contrastive pre-training. *arXiv:2212.03533v1 [cs.CL] 7 Dec 2022*.

[Wang et al. 2020] Wang, W., Bao, H., Huang, S., and Li Dong, F. W. (2020). Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. *arXiv:2012.15828 [cs.CL] 31 Dec 2020*.

[XGBoost] XGBoost. Xgboost documentation. https://xgboost.readthedocs.io/en/stable/. Accessed: 2023-11-23.

[Xu] Xu, M. Text2vec-base-multilingual. https://huggingface.co/shibing624/text2vec-base-multilingual. Accessed: 2023-11-23.

[Xu 2023] Xu, M. (2023). text2vec: A tool for text to vector. https://github.com/shibing624/text2vec. Accessed: 2023-11-23.