UNIVERSIDADE FEDERAL DO CEARÁ

TECHNICAL REPORT

Aluno: Luis Sávio Gomes Rosa

1. Introdução

O dataset utilizado neste trabalho são sobre notícias que foram classificadas como fake news e não fake news. Inicialmente, foram dados 2 datasets, ambos com 300 linhas cada, um com apenas fake news, e outro sem fake news. Os datasets possuíam a mesma estrutura, caracterizadas pelas seguintes colunas:

- ***id**: Um identificador único para cada registro.*
- ***news_url**: O URL da notícia.*
- ***title**: O título da notícia.*
- ***tweet_ids**: Identificadores de tweets associados à notícia.*

O objetivo principal deste relatório é avaliar as técnicas de extração de atributos e pré-processamento textual aplicadas às notícias, com vistas a construir modelos preditivos eficazes. As etapas deste estudo incluem:

***Exploração do Dataset:** Análise inicial das características do dataset, como distribuição de rótulos e estatísticas dos títulos.*

***Pré-Processamento Textual:** Limpeza e transformação dos títulos por meio de técnicas como remoção de stopwords, stemming e lematização.*

***Extração de Atributos**: Aplicar métodos CountVectorizer, TF-IDF, matriz de coocorrência e Word2Vec para representação dos dados textuais.*

***Avaliação de Modelos:** Comparar o desempenho de classificação utilizando diferentes formas de extração de atributos e pré-processamento textual.*

Assim, este trabalho busca contribuir para a compreensão e desenvolvimento de soluções eficazes no campo da detecção de fake news.

2. Observações

Como falado no tópico anterior, inicialmente foram dois datasets, logo, como o objetivo se trata de uma classificação, foi-se necessário a atribuição de uma nova coluna "fake_news", coluna essa do tipo booleana, onde 1 representa que a notícia é fake news e 0 significa que não. A partir disso foi-se necessário a concatenação dos dois datasets.

3. Resultados e discussão

a. Pré-processamento

No pré-processamento textual, foi utilizada a função *preprocess_text* para remover URLs, menções, caracteres especiais e números, bem como converter os textos para minúsculas. Stopwords em português foram eliminadas com a função *remove_stopwords*. Além disso, foram aplicadas técnicas de stemming, com o algoritmo *PorterStemmer*, e lematização, com o lematizador *WordNetLemmatizer*, ambas da biblioteca nltk. *A Tabela abaixo mostra um exemplo do feito usando uma frase de exemplo do dataset.*

Pré Processamento	
Texto Original	*"Contra fotos ops fatos, não há argumentos Juiz Marco Aurélio e seu amigo Fidel."*
preprocess_text	*contra fotos ops fatos h argumentos juiz marco aurlio amigo fidel*
apply_stemming	*contra foto op fato h argumento juiz marco aurlio amigo fidel*
apply_lemmatization	*contra fotos ops fatos h argumentos juiz marco aurlio amigo fidel*

b. Extração de Atributos

Os processos executados neste trabalho foram ampliados com a utilização de técnicas de extração de atributos, implementadas por meio de uma classe chamada FeatureExtraction. Essa classe permite a aplicação de diferentes métodos para explorar e representar os dados textuais do dataset processado. As etapas realizadas incluem:

Análise Estatística:

O msétodo statistical_analysis foi utilizado para calcular estatísticas descritivas dos texto, como o total de palavras, o comprimento médio das palavras e o número total de caracteres em cada texto. Esses atributos foram adicionados como novas colunas no dataset original, fornecendo informações básicas sobre a estrutura textual.

Count Vectorizer:

A função count_vectorizer gerou uma matriz de frequência de palavras com base nos títulos, considerando um máximo de 500 palavras mais frequentes. Essa representação vetorial permite avaliar a ocorrência de termos em todo o dataset.

TF-IDF (Term Frequency-Inverse Document Frequency):

A função tfidf_vectorizer criou uma matriz baseada na importância relativa de cada palavra no conjunto de dados, equilibrando a frequência do termo com sua relevância global. Esse método destaca palavras mais informativas para a análise.

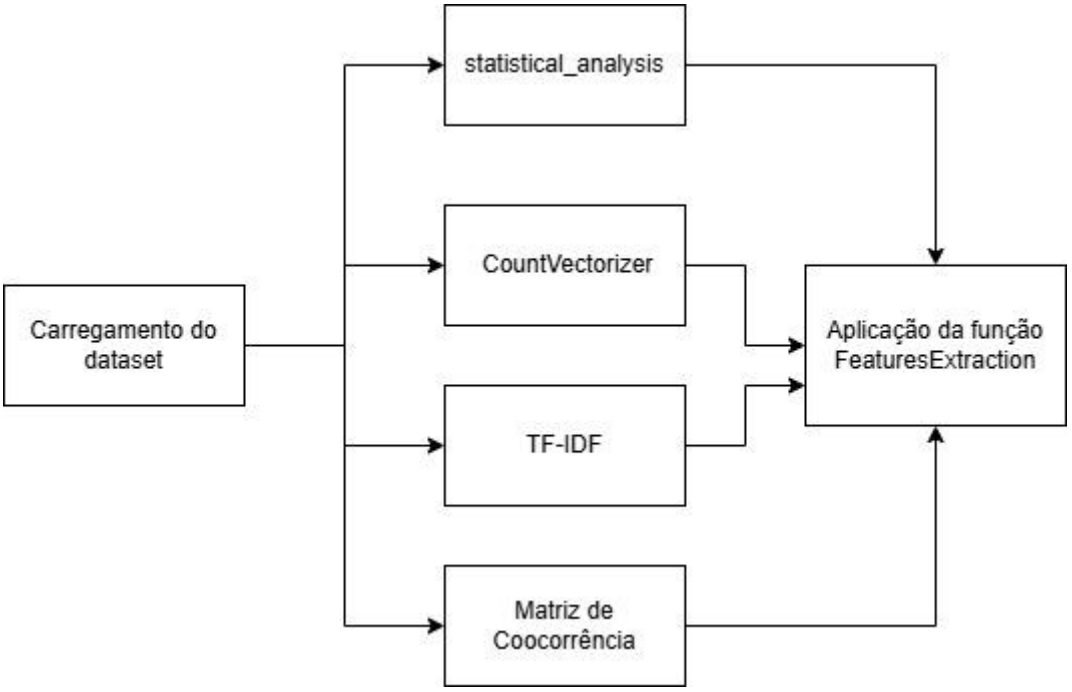
Matriz de Coocorrência:

A função cooccurrence_matrix construiu uma matriz que relaciona a frequência de palavras que aparecem juntas dentro de uma janela de contexto definida (tamanho 2). Essa representação auxilia na identificação de relações semânticas entre palavras. Para melhorar o desempenho e evitar alta dimensionalidade, foram utilizadas técnicas de redução, a técnica foi o **PCA (Principal Component Analysis).**

Word2Vec:

A função *word2vec* utilizou o modelo Word2Vec para gerar representações vetoriais densas de palavras, onde cada termo é mapeado para um vetor em um espaço dimensional. Essa abordagem captura relações semânticas mais profundas entre palavras baseando-se em seu contexto.

Fluxograma



c. Classificação

Nesta etapa, foi implementada uma abordagem para treinamento, avaliação e comparação de modelos preditivos utilizando diferentes formas de extração de atributos e pré-processamento textual. Foi implementado um pipeline para treinamento e avaliação de um modelo de classificação LogisticRegression, utilizando diferentes formas de extração de atributos textuais. O processo incluiu as seguintes etapas:

Função train_and_evaluate:

A função foi responsável por treinar e avaliar o modelo de classificação. Quando nenhum modelo era fornecido, a função utilizava o LogisticRegression como padrão. Os dados foram divididos em conjuntos de treino e teste (70% para treino e 30% para teste), e a função retornava a acurácia e um relatório de classificação detalhado.

Extração de Atributos:

Foram aplicadas diferentes formas de extração de atributos textuais utilizando a classe FeatureExtraction, incluindo:

- *statistical_analysis: Gera estatísticas como contagem de palavras e caracteres.*
- *count_vectorizer e tfidf_vectorizer: Geram representações vetoriais baseadas na frequência e na importância relativa dos termos.*
- *cooccurrence_matrix: Cria uma matriz relacionando palavras que aparecem próximas umas das outras.*
- *Word2Vec:* Utiliza o modelo Word2Vec para gerar representações vetoriais densas de palavras

Essas técnicas foram aplicadas tanto com quanto sem pré-processamento textual, permitindo uma comparação abrangente.

Tabela com os resultados das acurácias:

Modelo	**acurácia**
cooccurrence_matrix_no_pre	**0.950000**
cooccurrence_matrix_pre	**0.950000**
tfidf_vectorizer_no_pre	**0.933333**
count_vectorizer_no_pre	**0.927778**
count_vectorizer_pre	**0.922222**
tfidf_vectorizer_pre	**0.922222**
statistical_analysis_no_pre	**0.561111**
statistical_analysis_pre	**0.538889**
word2vec_no_pre	**0.455556**
word2vec_pre	**0.455556**

***Caso A**) A **Matriz de Coocorrência** obteve a melhor acurácia geral, alcançando **95,00%** tanto com quanto sem pré-processamento. Os métodos baseados em **TF-IDF** e **CountVectorizer** também apresentaram alta acurácia, com **93,33%** e **92,78%**,*

respectivamente, nos dados sem pré-processamento. No entanto, esses métodos tiveram uma ligeira redução de desempenho ao aplicar o pré-processamento, sugerindo que a transformação inicial dos textos pode não ser essencial para sua eficácia. Por outro lado, a ***Análise Estatística****, que utiliza atributos simples como contagem de palavras e caracteres, teve um desempenho significativamente inferior, com acurácia de* ***56,11%*** *sem pré-processamento e* ***53,89%*** *com pré-processamento. O* ***Word2Vec****, por sua vez, apresentou os piores resultados, com* ***45,56%*** *em ambos os casos, indicando que os vetores médios calculados não foram capazes de capturar as características discriminantes necessárias*

Caso b) *A cooccurrence_matrix foi identificado como a melhor forma de extração de atributos.*

Caso c)*, A aplicação de* ***Stemming*** *e* ***Lemmatização*** *no pré-processamento apresentou resultados idênticos, com acurácia de* ***95%****, quando usada em conjunto com a Matriz de Coocorrência.*

4. Conclusões

A análise realizada permitiu avaliar o desempenho de diferentes métodos de extração de atributos e técnicas de pré-processamento em um modelo de classificação para detecção de fake news. Os resultados demonstraram que a ***Matriz de Coocorrência****, tanto com quanto sem pré-processamento, foi a técnica mais eficaz, alcançando uma acurácia de* ***95%****. Isso evidencia sua capacidade de capturar relações semânticas entre palavras, especialmente após a redução de dimensionalidade com PCA. Os métodos* ***TF-IDF*** *e* ***CountVectorizer*** *também apresentaram desempenhos próximos, destacando-se como alternativas robustas para análise textual, embora tenham registrado ligeira queda de desempenho com a aplicação do pré-processamento. Por outro lado, a* ***Análise Estatística*** *e o* ***Word2Vec*** *não foram capazes de fornecer informações suficientemente discriminantes para obter bons resultados, mostrando limitações significativas no contexto deste dataset.*

Ao comparar as técnicas de pré-processamento, ***Stemming*** *e* ***Lemmatização*** *apresentaram desempenhos equivalentes, indicando que, para os métodos avaliados, ambas são igualmente adequadas. No entanto, sua aplicação mostrou-se mais relevante para métodos específicos, como a Matriz de Coocorrência, do que para TF-IDF e CountVectorizer.*

Esses resultados reforçam a importância de selecionar métodos de extração de atributos adequados ao problema e ao tipo de dados. Para aplicações futuras, recomenda-se explorar embeddings pré-treinados, como Word2Vec ou GloVe, e investigar o impacto de modelos mais complexos, como redes neurais ou ensembles, para potencializar o desempenho da classificação. Além disso, a otimização dos parâmetros da redução de dimensionalidade e uma análise mais aprofundada do pré-processamento podem ajudar a refinar ainda mais os resultados.

Fluxograma

5. Próximos passos

Exploração de Modelos Mais Sofisticados:

Avaliar o desempenho de algoritmos mais complexos, como Random Forest, Gradient Boosting (XGBoost, LightGBM) e Redes Neurais, para verificar se eles trazem melhorias significativas.

Análise de Sensibilidade ao Pré-Processamento:

Investigar como diferentes combinações de pré-processamento (e.g., remoção de stopwords, uso combinado de stemming e lematização) afetam o desempenho dos modelos.

Validação em Dados Reais:

Testar os modelos em novos conjuntos de dados ou dados do mundo real para avaliar sua generalização e robustez.

Otimização de Hiperparâmetros:

Realizar buscas de hiperparâmetros (Grid Search ou Random Search) nos modelos mais promissores para maximizar o desempenho.