

ANO
2025



**CADERNO DE RESPOSTAS DA
ATIVIDADE PRÁTICA DE:**

NLP

ALUNA+: JOSELINA DE ALMEIDA MARRA

RU: 4688889

Caderno de Resposta Elaborado por:
Prof. MSc. Guilherme Ditzel Patriota

Prática 01 – Criação de modelo de classificação supervisionado para análise de Fake News.

Questão 01 – Apresente aqui o código referente ao modelo gerado e a nuvem de palavras que foram usadas para identificar textos VERDADEIROS (Estes códigos devem estar completos, incluindo as importações, os processamentos, os treinamentos e a geração da nuvem de palavras específica solicitada.)

ENUNCIADO: Veja o Roteiro da Atividade Prática para mais detalhes.

I. Apresentação do Código (não esquecer do identificador pessoal):

```
# RU: 4688889

import pandas as pd
import nltk
from nltk import ngrams

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

from funcoes_auxiliares import gerar_nuvem_palavras

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import RSLPStemmer
import string

# Baixar recursos do NLTK
nltk.download("all")

# 1. Criação do dataframe a partir do CSV
df = pd.read_csv('pre-processed.csv')

# 2. Configuração stemmer e stopwords
stemmer = RSLPStemmer()
nltk.download('stopwords')

# Baixar o tokenizer
nltk.download('punkt')

# 3. Função de pré-processamento do texto
def preprocess_text(text):

    # Tokenização
    tokens = word_tokenize(text, language='portuguese')

    # Remover acentos e números, deixar minúsculas
    tokens = [t.lower() for t in tokens if t.isalpha()]

    # Remover stopwords e pontuações
    stop_words = set(stopwords.words('portuguese'))
    tokens = [t for t in tokens if t not in stop_words
              and t not in string.punctuation]

    # Remover palavras muito curtas
    tokens = [t for t in tokens if len(t) > 2]
```

II. Apresentação das Imagens/Print do resultado (não esquecer do identificador):

Acurácia do modelo: 94,67%
 RU: 4688889
 Palavras: 72, Bigramas: 1367, Trigramas: 2161
 * Um total de 3600 tokens foram computadas a partir do conjunto de dados.

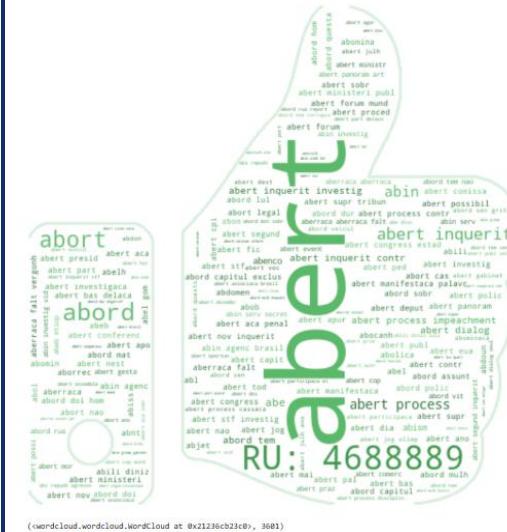


Figura 2 : Nuvem de palavras com identificador de verdadeiro. RU: 4688889 inserido do lado direito, na parte inferior da imgem.

```

# Aplicar o stemmer
tokens = [stemmer.stem(t) for t in tokens]
return ' '.join(tokens)

# 4. Pré-processamento no dataframe
df['preprocessed_news'] = df['preprocessed_news'].apply(preprocess_text)

# 5. Trunca os pares de notícias verdadeiras com falsas
# para normalizar quantidade de palavras
df_truncated = pd.concat(
    [
        df[df['label'] == 'true'].head(len(df[df['label'] == 'fake'])),
        df[df['label'] == 'fake']
    ]
)

# 6. Remonta as notícias em string e cria coluna no dataframe
# para o resultado do pré-processamento
df_truncated['processed_news'] = df_truncated['preprocessed_news']
print(df_truncated[['label', 'processed_news']])

# 7. Criação do modelo de classificação (Regressão Logística)
from sklearn.linear_model import LogisticRegression

# 7.1 Criar matriz de frequências TF-IDF com n-gramas de 1 a 3 palavras
vectorizer = TfidfVectorizer(ngram_range=(1, 3))
X = vectorizer.fit_transform(df_truncated['processed_news'])
y = df_truncated['label']

# 7.2 Divide o corpus pré-tratado em 75% para treinamento e 25% para testes
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42)

# 7.3 Faz regressão Logística com solver='Lbfgs'
logreg_model = LogisticRegression(solver='lbfgs')
logreg_model.fit(X_train, y_train)

# 7.4 Realizar predição dos textos de teste com o método predict_proba
proba_predictions = logreg_model.predict_proba(X_test)

# 7.5 Calcular a acurácia geral do algoritmo
predicted_labels = logreg_model.predict(X_test)
accuracy = accuracy_score(y_test, predicted_labels)
print(f'Acurácia do modelo: {accuracy:.2%}')

```

```

# 8. Separar frequências para textos rotulados como verdadeiros
# Frequência total dos tokens no conjunto de treinamento
frequencias_treinamento = dict(
    zip(vectorizer.get_feature_names_out(),
        X_train.sum(axis=0).A1))

# Índices dos textos verdadeiros
verdadeiras_indices = df_truncated[df_truncated['label'] == 'true'].index

# Features relevantes para textos verdadeiros
frequencias_verdadeiras = {
    token: frequencias_treinamento[token]
    for token in set(vectorizer.get_feature_names_out())
    [verdadeiras_indices].intersection(frequencias_treinamento)}

# Converte para dicionário
frequencias_verdadeiras = dict(frequencias_verdadeiras)

# 9. Contagem de palavras, bigramas e trigramas
num_palavras = sum(1 for token in frequencias_verdadeiras.keys()
    if ' ' not in token)
num_bigramas = sum(1 for token in frequencias_verdadeiras.keys()
    if ' ' in token and token.count(' ') == 1)
num_trigramas = sum(1 for token in frequencias_verdadeiras.keys()
    if ' ' in token and token.count(' ') == 2)

print(f'RU: 4688889')
print(f'Palavras: {num_palavras}, bigramas: {num_bigramas}, Trigramas: {num_trigramas}')

# Adiciona o RU ao dicionário com frequência 1
frequencias_verdadeiras[f'RU: 4688889'] = 10

# 10. Geração da nuvem de palavras usando a função auxiliar
gerar_nuvem_palavras(
    arquivo_mascara="thumbs_up_mask.png",
    dicionario_tokens_e_frequencia=frequencias_verdadeiras
)

```

Figura 1: Etapas de preparação do modelo: instalação das dependências, carga e pré-processamento do corpus, balanceamento entre textos verdadeiros e falsos, extração das features relevantes e inserção do RU no dicionário final de frequências, seguido da geração da nuvem de palavras utilizada na identificação dos textos reais.

III. Responda à pergunta: Quantas palavras, bigramas e trigramas foram usados dos textos rotulados como REAL para a criação de seu modelo e qual a acurácia?

Resposta: 72 Palavras, 1367 Bigramas e 2161 Trigramas. Acurácia de 94.67%.80

Prática 01 – Criação de modelo de classificação supervisionado para análise de Fake News.

Questão 02 – Apresente aqui o código referente ao modelo gerado e a nuvem de palavras que foram usadas para identificar textos FALSOS. (Estes códigos devem estar completos, incluindo as importações, os processamentos, os treinamentos e a geração da nuvem de palavras específica solicitada.)

ENUNCIADO: Veja o Roteiro da Atividade Prática para mais detalhes.

- I. Apresentação do Código (não esquecer do identificador pessoal): II. Apresentação das Imagens/Print do resultado (não esquecer do identificador):

```

# RU: 4688889

# 1. Frequências já calculadas no treino (mesmo dicionário)
frequencias_treinamento = dict(
    zip(vectorizer.get_feature_names_out(), X_train.sum(axis=0).A1)
)

# 2. Índices dos textos FALSOS no dataframe truncado
falsas_indices = df_truncated[df_truncated['label'] == 'fake'].index

# 3. Features relevantes para textos FALSOS
frequencias_falsas = {
    token: frequencias_treinamento[token]
    for token in set(vectorizer.get_feature_names_out()).intersection(
        frequencias_treinamento
    )
}

# 4. Contagem de palavras, bigramas e trigramas (dicionário - FALSAS)
num_palavras_false = sum(
    1 for token in frequencias_falsas.keys() if " " not in token
)
num_bigramas_false = sum(
    1 for token in frequencias_falsas.keys()
    if " " in token and token.count(" ") == 1
)
num_trigramas_false = sum(
    1 for token in frequencias_falsas.keys()
    if " " in token and token.count(" ") == 2
)

print(f"RU: 4688889")
print(
    f"Palavras: {num_palavras_false}, "
    f"bigramas: {num_bigramas_false}, "
    f"Trigramas: {num_trigramas_false}"
)

```

5. Coloca o RU no dicionário de frequências FALSAS

frequencias_falsas["RU: 4688889"] = 50



Figura 4: Nuvem de palavras em formato de mão negativa com termos usados na classificação de notícias FAKE. O RU 4688889 aparece em vermelho na parte superior da nuvem.

Figura 3 : Etapas de seleção das features para textos rotulados como FAKE, construção do dicionário de frequências, inserção do RU 4688889 no conjunto de tokens e chamada da função auxiliar para gerar a nuvem de palavras negativa

- III. Responda à pergunta: Quantas palavras, bigramas e trigramas foram usados dos textos rotulados como FAKE para a criação de seu modelo, quais foram as técnicas usadas em seu pré-processamento e qual tipo de modelo foi escolhido para este classificador?

Resposta: 36561 palavras, 1176204 bigramas e 2161216 trigramas.

O pré-processamento foi feito com tokenização, remoção de acentos, números e pontuações, conversão de todo o texto para minúsculas e remoção de stopwords em português usando a biblioteca NLTK. Em seguida, os textos foram normalizados (balanceamento entre REAL e FAKE) e representados com TF-IDF de 1 a 3-gramas para treinar o modelo. O classificador escolhido foi uma **Rregressão Logística**, que obteve acurácia de 94.67%.

LINK do seu projeto, se existir (google colab de acesso aberto

ou repositório github público):

🔗 https://github.com/JosyMarra/NLP_Projeto