NPL-Sentinel: Análise de Sentimento em Avaliações de Produtos

Disciplina: Processamento de Linguagem Natural

Professor: Luciano Barbosa

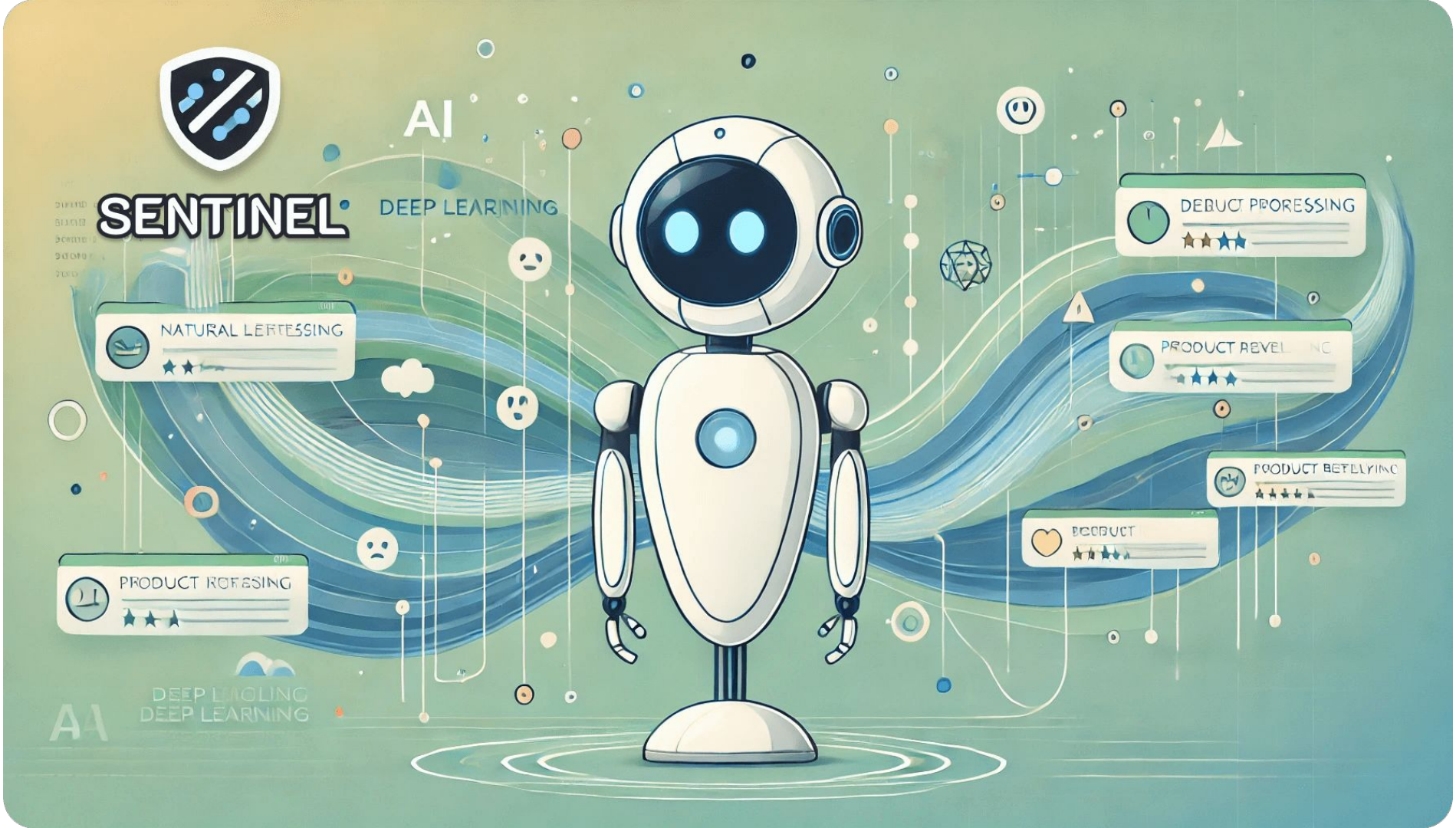
Link Colab: https://colab.research.google.com/drive/1WwAcQrNEa5Md1hhqvQt9R4uDtFJ57JS8?usp=sharing

Alunos:
Paloma Correa Alves | pca2@cin.ufpe.br
Luciano Ayres Farias de Carvalho | lafc@cin.ufpe.br

Especialização
Deep Learning

Centro de Informática UFPE

Universidade Federal de Pernambuco


SENTINEL
AI
DEEP LEARNING
NATURAL LEFESSING
PRODUCT REFESSING
DEBUCT PROFESSING
PRODUCT BEFELYING
PRODUCT
AI
DEEP LEARNING

Introdução

Visão Geral:

- O NPL-Sentinel é um projeto que utiliza técnicas de Processamento de Linguagem Natural (PLN) para analisar sentimentos em avaliações de consumidores sobre um smartphone.

Motivação:

- Compreender a percepção dos consumidores é crucial para aprimorar produtos e estratégias de mercado.
- A análise automatizada de sentimentos permite insights em larga escala e em tempo real.

Objetivos do Projeto

Objetivo Geral:

- Desenvolver modelos capazes de identificar a polaridade emocional (positivo, neutro, negativo) em avaliações textuais de produtos.

Objetivos Específicos:

- Correlacionar a polaridade dos sentimentos com as notas atribuídas pelos consumidores.
- Analisar padrões textuais e temáticos nas avaliações.
- Realizar análises exploratórias para identificar padrões nas avaliações, destacando os aspectos mais comentados em avaliações positivas e negativas.
- Comparar o desempenho de diferentes abordagens de modelagem em PLN.

Metodologia Geral

Etapas do Projeto:

- Coleta e Pré-processamento de Dados
- Análise Descritiva Exploratória
- Desenvolvimento de Modelos de Classificação
- Avaliação e Comparação dos Modelos
- Análise de Resultados e Conclusões

Abordagens Utilizadas:

- Métodos tradicionais de machine learning (SVM).
- Técnicas avançadas de deep learning (BERT).
- In-Context Learning com LLMs (**OpenAI GPT-4 e Google Gemini**).

Fonte de dados

Fonte dos Dados:

- Foi elaborado um CSV com dados extraídos de uma plataforma de e-commerce amplamente utilizada. Foram utilizados dados de reviews de consumidores da plataforma online, com foco em opiniões relacionadas ao produto avaliado.

Características dos Dados:

- Total de Avaliações: 4.197

Campos:

★ review: Comentário do consumidor.
★ rating: Nota de satisfação (1 a 5).

```python
import pandas as pd

url = "https://raw.githubusercontent.com/lucianoayres/npl-sentinel/refs/heads/main/data/re
df = pd.read_csv(url)
df.to_csv("reviews.csv", index=False)
print(df.head())
print("\nTipos de dados de cada coluna:")
df.info()
print("\nDescrição detalhada dos dados:")
print(df.describe(include='all'))
print("\nEstatísticas descritivas para variáveis categóricas:")
print(df.describe(include=['object']))
```

Pré-processamento de Dados

Limpeza de Texto:

- Conversão para minúsculas.
- Remoção de pontuação e caracteres especiais.

Tokenização e Lematização:

- Uso de NLTK e spaCy.

Remoção de Stopwords:

- Eliminação de palavras irrelevantes para a análise.

Conversão de Notas em Sentimentos:

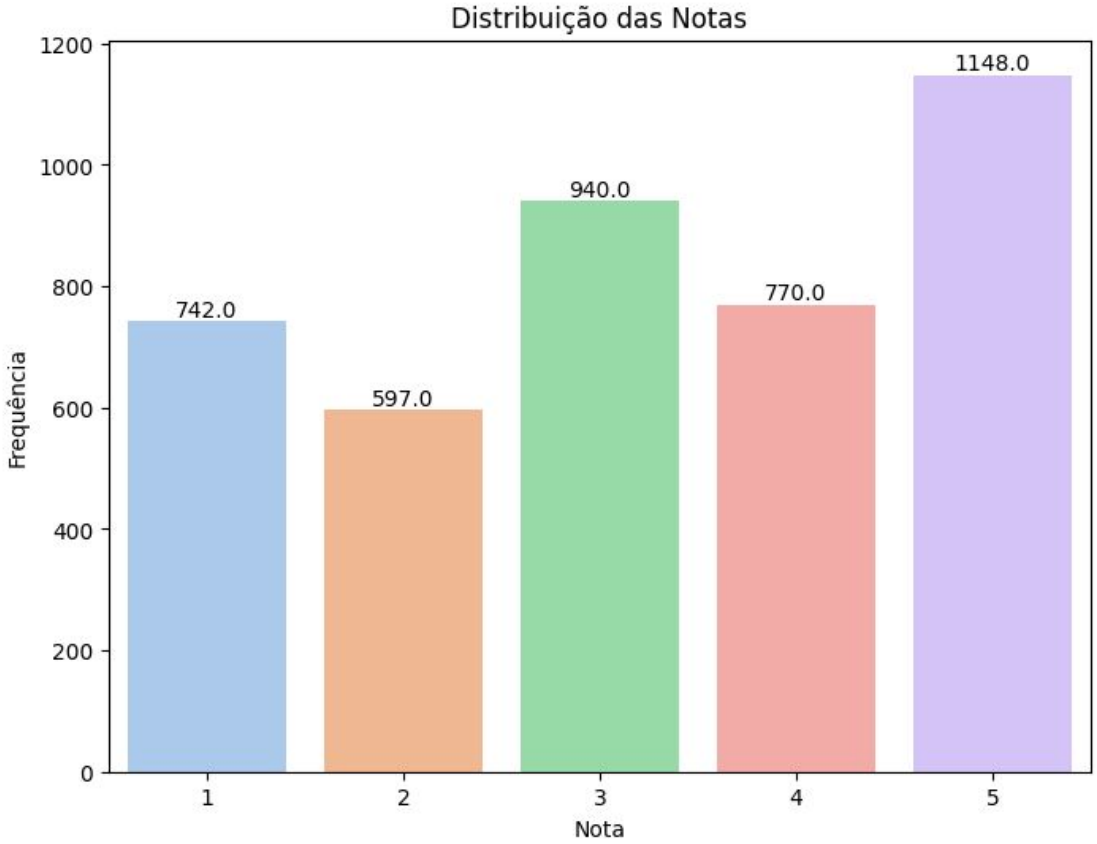
- Notas 1-2: Negativo
- Nota 3: Neutro
- Notas 4-5: Positivo


```
[ ] import nltk
    from sklearn.model_selection import train_test_split
    from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
    from nltk.corpus import stopwords
    import string

    # Faz o download das stopwords
    nltk.download("stopwords")
    stop_words = stopwords.words("portuguese")

    # Função para pré-processar o texto
    def preprocess_text(text):
        text = text.lower()  # Converte para minúsculas
        text = text.translate(str.maketrans('', '', string.punctuation))  # Remove pontuação
        text = "".join([char for char in text if char.isalnum() or char.isspace()])  # Remove caracteres não alfanuméricos
        tokens = [word for word in text.split() if word not in stop_words]  # Remove stopwords
        return " ".join(tokens)  # Junta os tokens de volta em uma string

    # Aplica a função de pré-processamento à coluna 'review'
    df["clean_review"] = df["review"].apply(preprocess_text)

    # Função para converter notas em rótulos de sentimento
    def convert_to_sentiment(rating):
        if rating >= 4:
            return "positivo"
        elif rating <= 2:
            return "negativo"
        else:
            return "neutro"

    # Aplica a função de conversão à coluna 'rating'
    df["sentiment"] = df["rating"].apply(convert_to_sentiment)

    # Verifica e remove linhas com review ou rating vazio
    empty_reviews = df["review"].isnull() | (df["review"].str.strip() == "")
    empty_ratings = df["rating"].isnull()
    df = df[~(empty_reviews | empty_ratings)]

    # Exibe a quantidade de reviews e ratings vazios removidos
    print("\nItens Removidos:")
    print("Reviews vazios:", df["review"].isnull().sum() + (df["review"].str.strip() == "").sum())
    print("Ratings vazios:", df["rating"].isnull().sum())
```

Análise Descritiva - Distribuição das Notas

Gráfico de Barras - Distribuição das Nota

- Nota 1: 742 avaliações
- Nota 2: 597 avaliações
- Nota 3: 940 avaliações
- Nota 4: 770 avaliações
- Nota 5: 1148 avaliações



Obs:
Predominância de avaliações positivas (nota 5).
Distribuição assimétrica indicando tendência geral de satisfação.

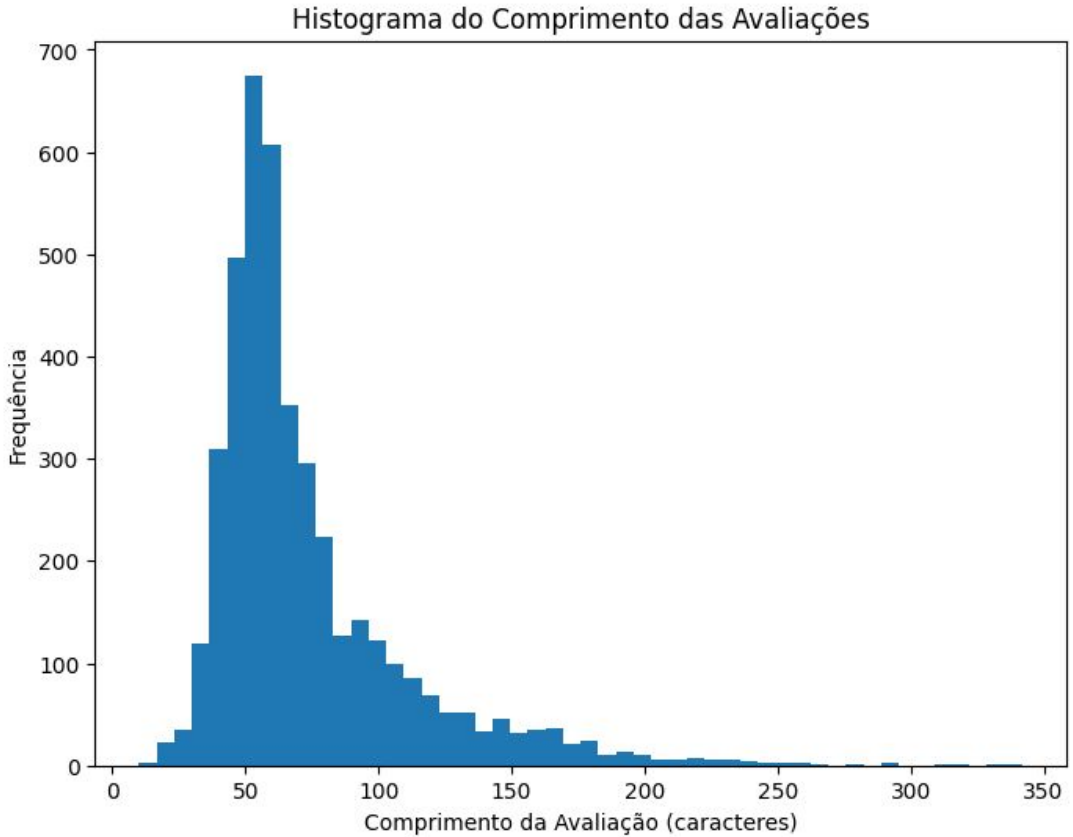
Análise Descritiva - Comprimento das Avaliações

Histograma do Comprimento das Avaliações:

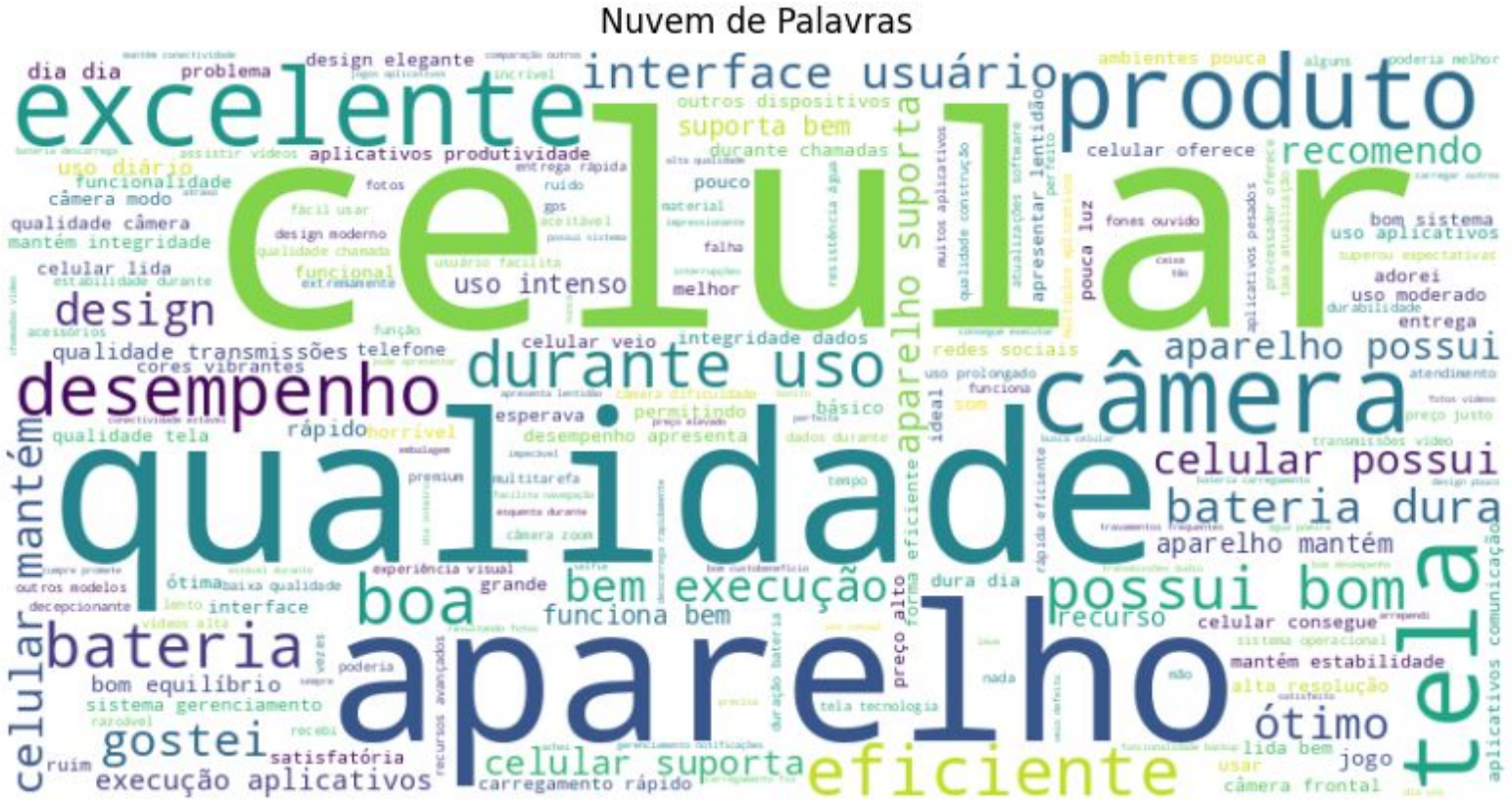
- Comprimento Médio: 73,36 caracteres
- Comprimento Mínimo: 10 caracteres
- Comprimento Máximo: 342 caracteres

Observações:

- A maioria das avaliações concentra-se entre 40 e 60 caracteres.
- Avaliações curtas e objetivas são comuns.



Análise Textual - Frequência de Palavras



Análise Textual - Frequência de Palavras

Palavra	Frequência
celular	1.0000
qualidade	0.4106
aparelho	0.3480
excelente	0.3248
câmera	0.3201
tela	0.3178
produto	0.2807
desempenho	0.2552
boa	0.2482
eficiente	0.2436

-As 10 palavras mais frequentes destacam atributos positivos do produto.

-Indica que os consumidores frequentemente mencionam a qualidade.

-A frequência foi normalizada em relação à palavra mais frequente (celular).

Análise Textual - Entidades Nomeadas mais frequentes

Tópico 4 - Análise

1. **Lematização** dos textos na coluna **review** para simplificar as palavras ao seu radical.

2. **Extração de entidades nomeadas** relevantes (organizações, pessoas, locais e produtos).

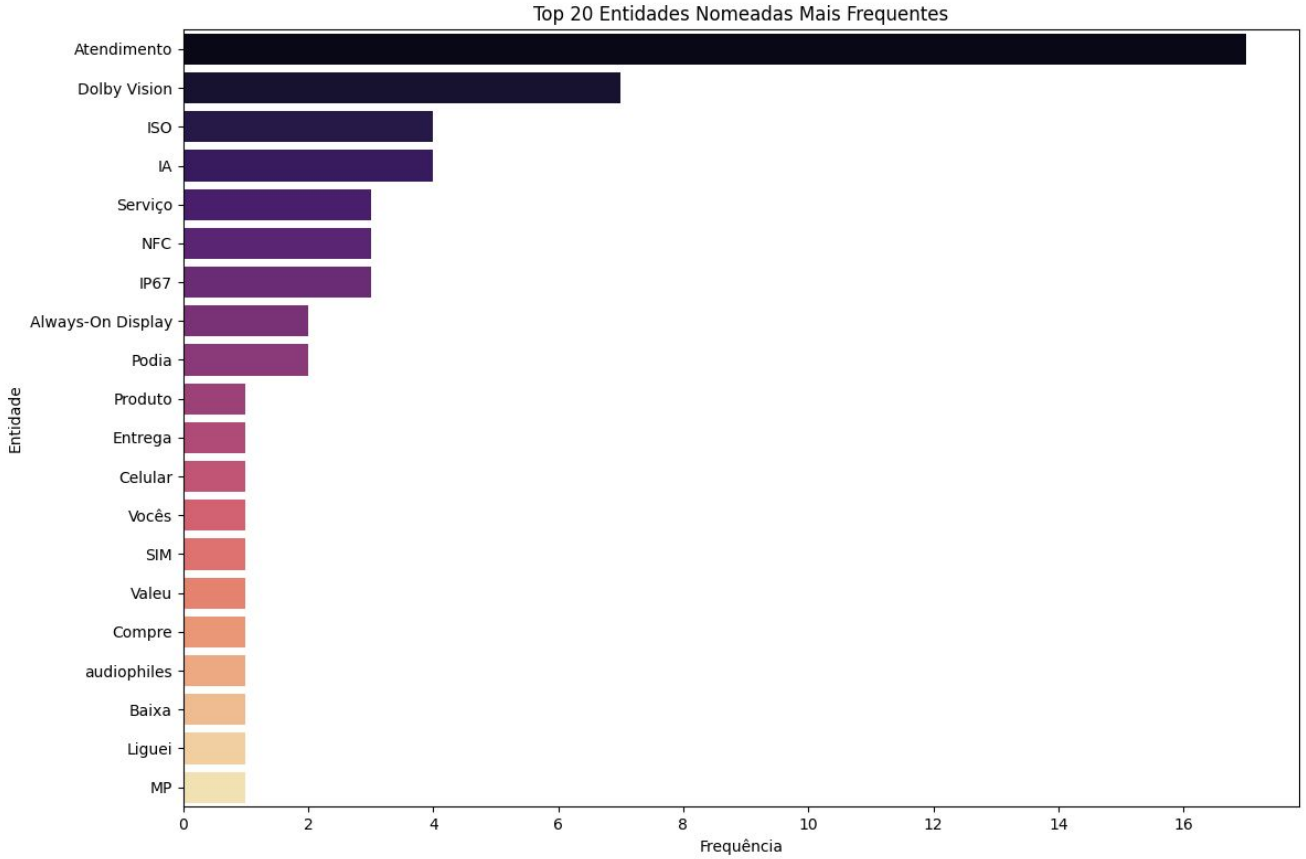
3. **Contagem e análise** das entidades mais frequentes, facilitando a identificação dos principais elementos mencionados nas avaliações.

Essas etapas são essenciais para análises de textos mais avançadas, como identificação de tendências ou padrões nos dados.

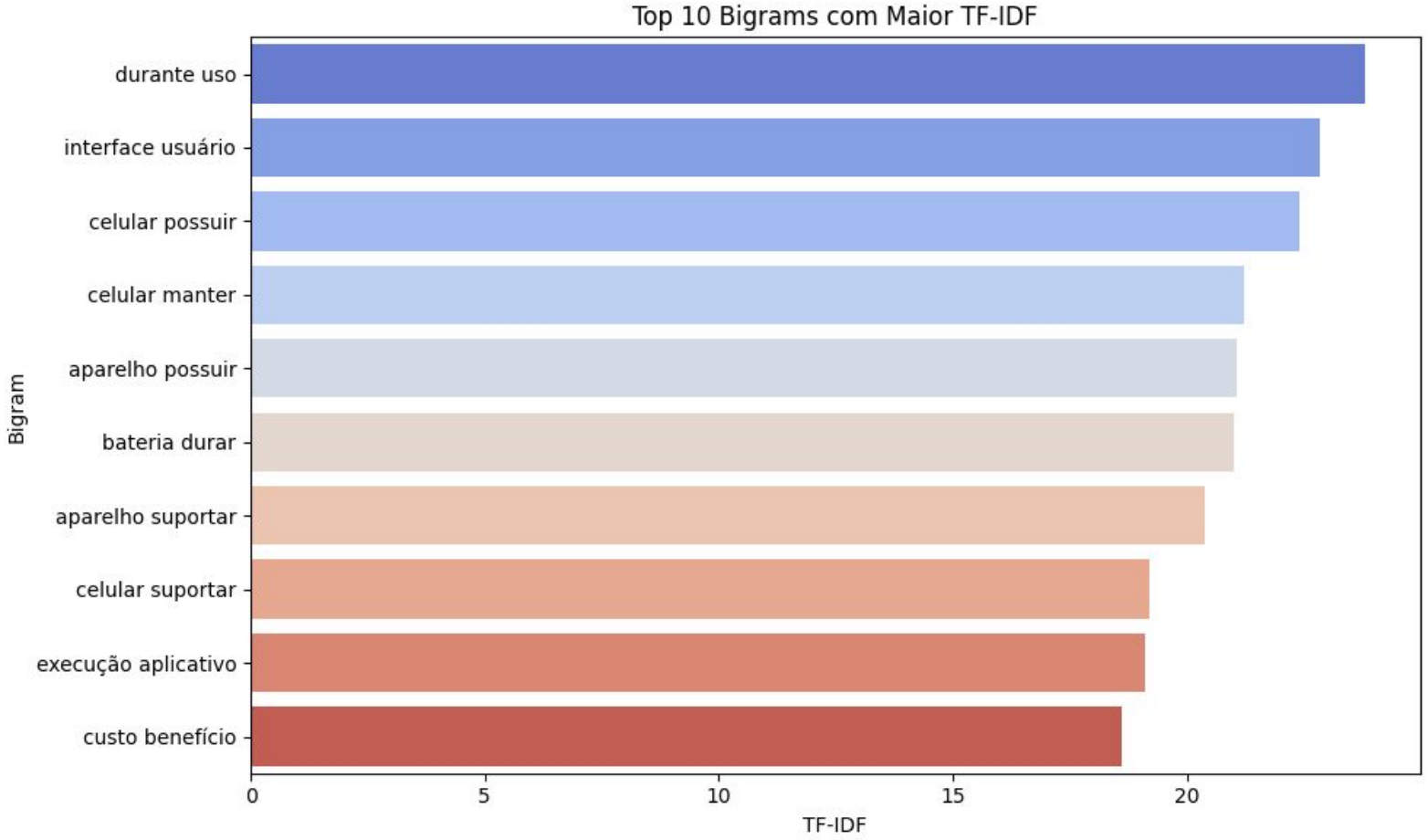
3.4 Frequência das Entidades (review)

```python
# Faz o Download do Modelo de Linguagem PT-BR
!python -m spacy download pt_core_news_md

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from collections import Counter
import spacy

import plotly.express as px

# Carregamento do modelo de linguagem do spaCy
nlp = spacy.load('pt_core_news_md')

# Pré-processamento com spaCy
def preprocess(text):
    doc = nlp(text)
    tokens = [
        token.lemma_.lower() for token in doc
        if not token.is_stop and not token.is_punct and not token.like_num
    ]
    return ' '.join(tokens)

df['clean_review'] = df['review'].astype(str).apply(preprocess)

# Análise de Entidades Nomeadas com spaCy
def extract_entities(text):
    doc = nlp(text)
    return [ent.text for ent in doc.ents if ent.label_ in ['ORG', 'PERSON', 'LOC', 'PRODUCT']]

df['entities'] = df['review'].astype(str).apply(extract_entities)

# Contagem das entidades mais frequentes
all_entities = [entity for sublist in df['entities'] for entity in sublist]
entity_freq = Counter(all_entities).most_common(20)
```

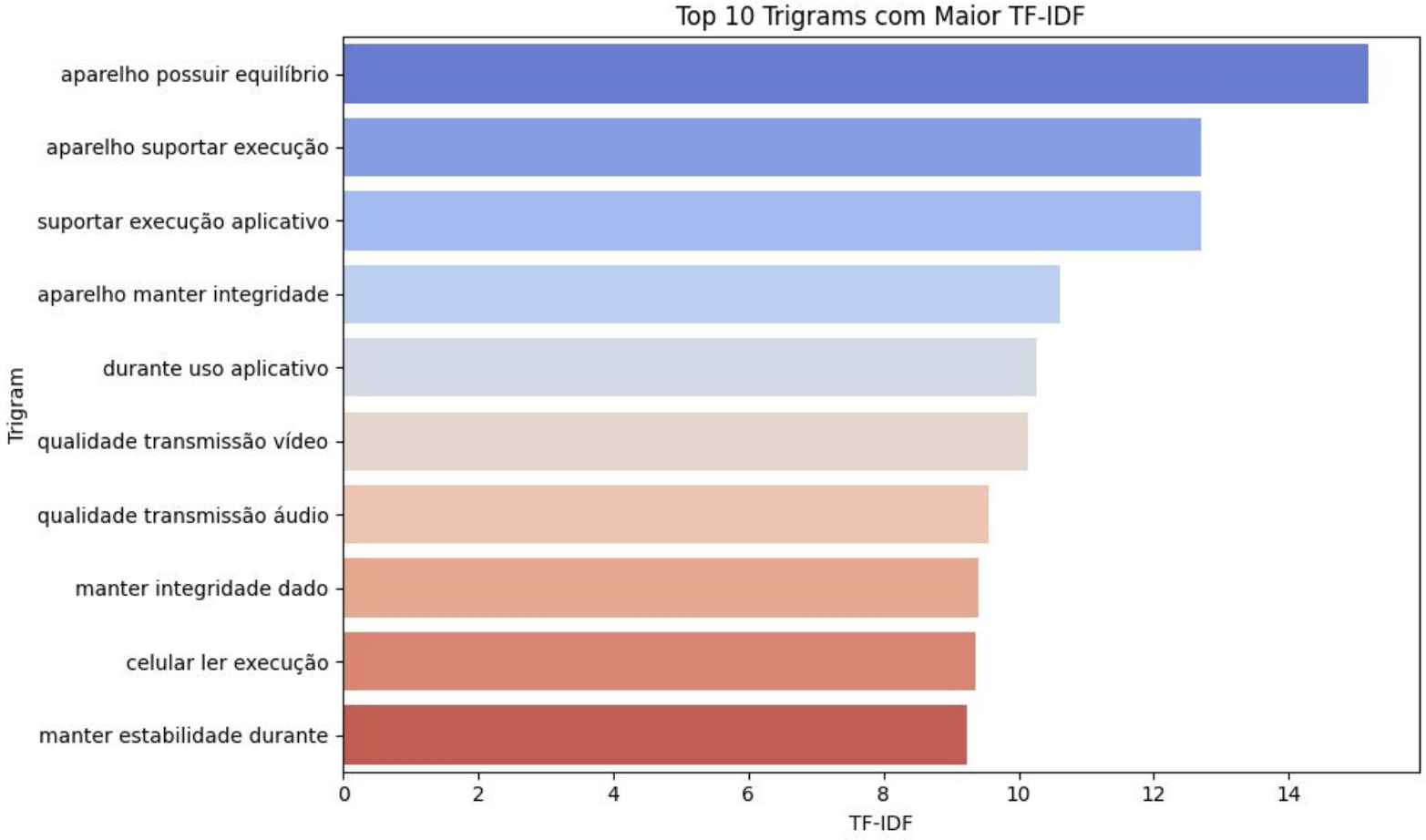
Análise Textual - Entidades Nomeadas mais frequentes Tópico 4 - Análise



Análise Textual - N-grams



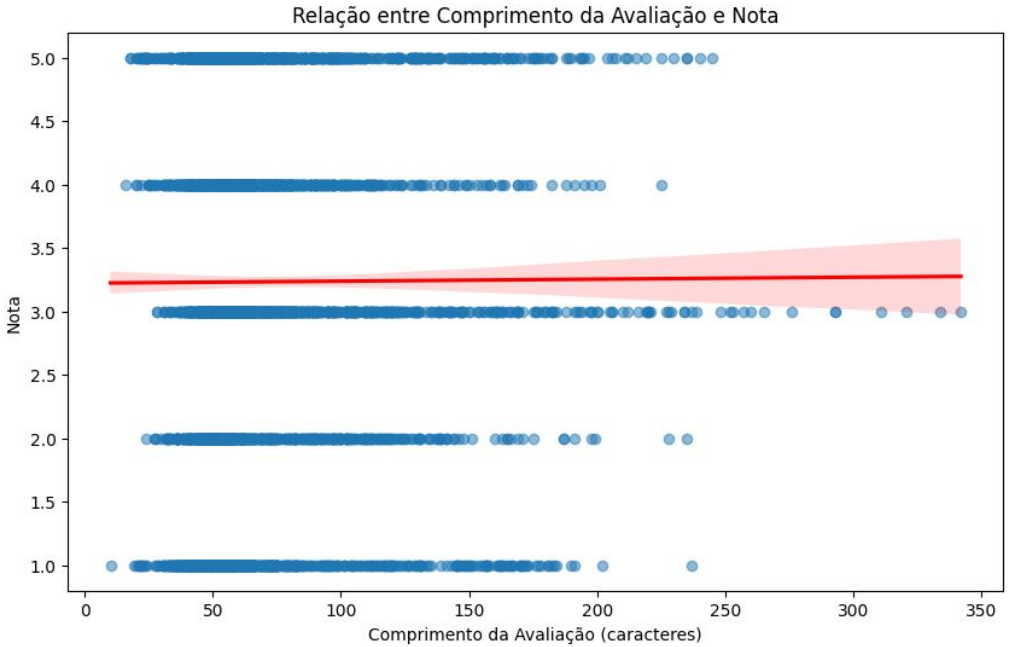
Análise Textual - N-grams



Comprimento das avaliações e as notas

Tópico 4 - Análise

- A linha de tendência indica uma ligeira inclinação positiva;
- O coeficiente de correlação é baixo, mostrando que a relação é fraca;
- Avaliações curtas aparecem em todas as faixas de notas.
- O comprimento do texto não é um forte indicador de satisfação, embora exista uma leve tendência de textos mais longos estarem relacionados a avaliações mais positivas.



Análise de Sentimento - Correlação com Notas

Correlação entre Comprimento da Avaliação e Nota:

- Coeficiente de Pearson: 0,00 (baixa correlação).

```
# Calcular e imprimir a correlação
correlation = df['review_length'].corr(df['rating'])
print("-" * 50)
print(f"Coeficiente de correlação de Pearson: {correlation:.2f}")
print("-" * 50)
```

```
--------------------------------------------------
Coeficiente de correlação de Pearson: 0.00
```

Interpretação:

- O tamanho da avaliação não está significativamente relacionado à nota atribuída.
- Avaliações longas não necessariamente indicam insatisfação ou satisfação extremas.

Desenvolvimento dos Modelos - SVM + Bag of Words

Descrição do Método:

- Transformação das avaliações em vetores numéricos usando Bag of Words (BoW).
- Treinamento de um SVM (Support Vector Machine) para classificação.

Processo:

- Divisão dos dados em treino (80%) e teste (20%).
- Uso do CountVectorizer para criação do BoW.
- Ajuste de hiperparâmetros padrão do SVM.

Motivação:

- Método simples e eficiente para estabelecer uma linha de base.

Resultados - SVM + Bag of Words

Métricas de Avaliação:

- Acurácia: 64,04%
- F1 Score: 63,05%

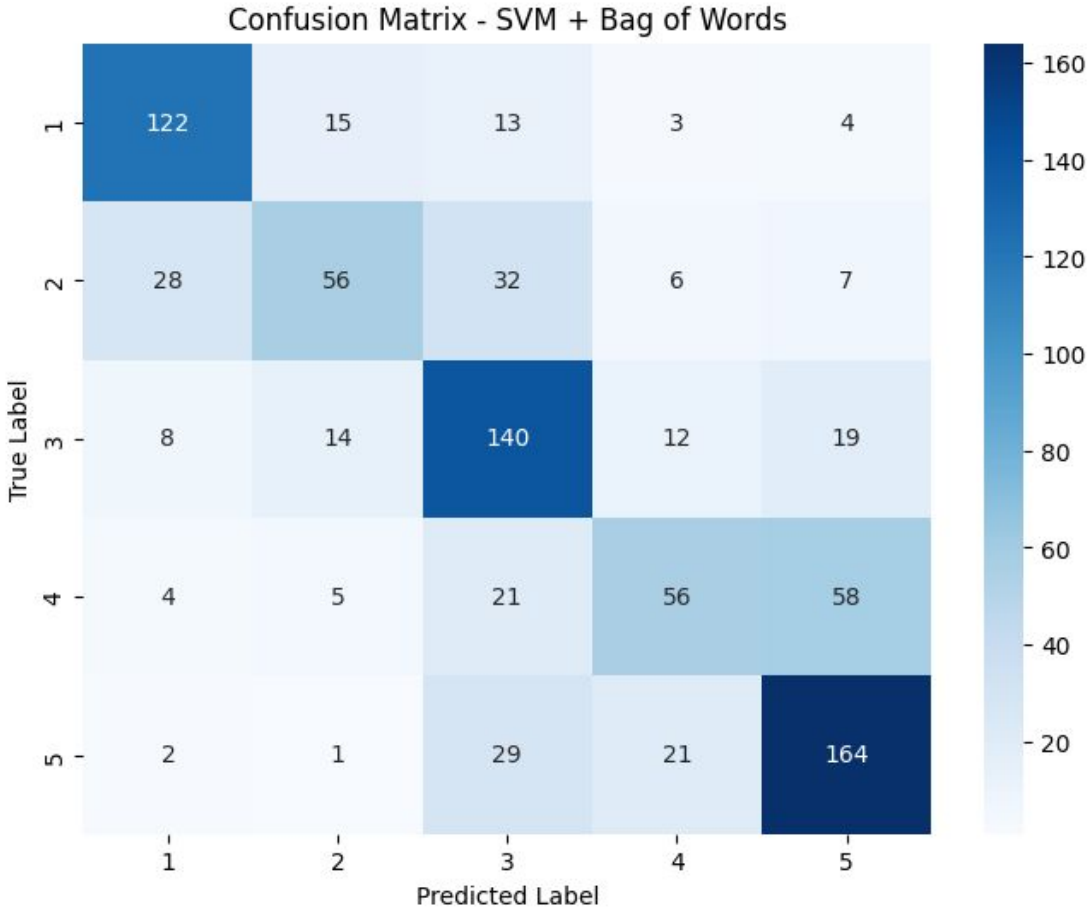
Análise:

- Desempenho modesto, indicando limitações em capturar contexto semântico.
- Dificuldades em diferenciar sentimentos neutros.

```
# Avaliação
print("SVM + Bag of Words")
print(classification_report(y_test, y_pred_bow))

# Calcula a acurácia e F1 score
accuracy_bow = accuracy_score(y_test, y_pred_bow)
f1_bow = f1_score(y_test, y_pred_bow, average='weighted')

print(f"Accuracy: {accuracy_bow}")
print(f"F1-score: {f1_bow}")
```

```
SVM + Bag of Words
              precision    recall  f1-score   support

           1       0.74      0.78      0.76       157
           2       0.62      0.43      0.51       129
           3       0.60      0.73      0.65       193
           4       0.57      0.39      0.46       144
           5       0.65      0.76      0.70       217

    accuracy                           0.64       840
   macro avg       0.64      0.62      0.62       840
weighted avg       0.64      0.64      0.63       840

Accuracy: 0.6404761904761904
F1-score: 0.6305711851281793
```

Resultados - SVM + Bag of Words



Desenvolvimento dos Modelos - SVM + Embeddings

Descrição do Método:

- Utilização de embeddings gerados pelo spaCy para representar textos.
- Treinamento de um SVM com esses vetores densos.

Processo:

- Extração de vetores de dimensão fixa para cada avaliação.
- Normalização dos embeddings.
- Treinamento e validação similares ao modelo anterior.

Motivação:

- Capturar melhor as relações semânticas entre palavras.

Resultados - SVM + Embeddings

Métricas de Avaliação:

- Acurácia: 60,0%
- F1 Score: 58.30%

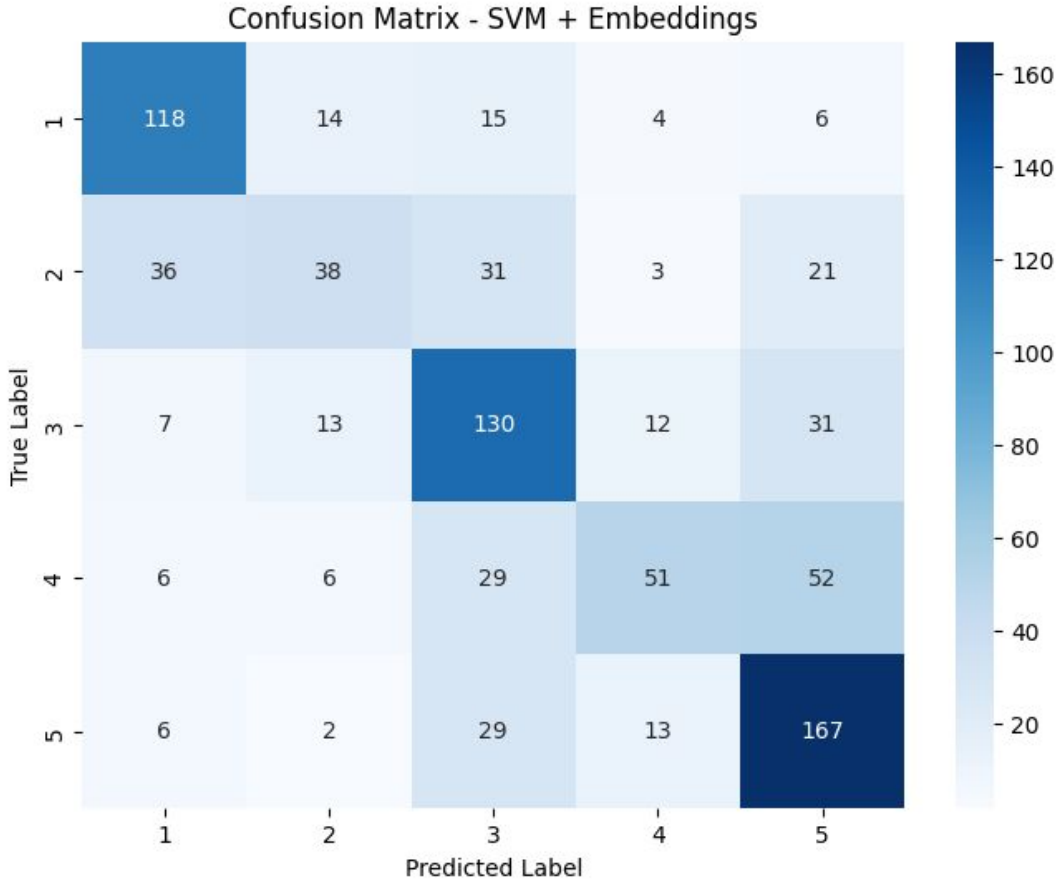
Análise:

- Embeddings contribuíram para uma representação mais rica, mas não suficiente.

```
# Avaliação
print("SVM + Embeddings")
print(classification_report(y_test, y_pred_embed))

# Calcula a acurácia e F1 score
accuracy_embed = accuracy_score(y_test, y_pred_embed)
f1_embed = f1_score(y_test, y_pred_embed, average='weighted')

print(f"Accuracy: {accuracy_embed}")
print(f"F1-score: {f1_embed}")
```

```
SVM + Embeddings
              precision    recall  f1-score   support

           1       0.68      0.75      0.72       157
           2       0.52      0.29      0.38       129
           3       0.56      0.67      0.61       193
           4       0.61      0.35      0.45       144
           5       0.60      0.77      0.68       217

    accuracy                           0.60       840
   macro avg       0.60      0.57      0.57       840
weighted avg       0.60      0.60      0.58       840

Accuracy: 0.6
F1-score: 0.5830386318478644
```

Resultados - SVM + Embeddings



Desenvolvimento dos Modelos - BERT

O modelo BERT usa redes neurais profundas com arquitetura de transformers e busca apreender as relações contextuais entre as palavras e fazer uma representação semântica do texto. Várias variações de modelos surgiram a partir dele (DEVLIN, et al., 2018).

Descrição do Método:

- Uso do modelo pré-treinado BERT específico para o português (neuralmind/bert-base-portuguese-cased).
- Ajuste fino (fine-tuning) para a tarefa de classificação de sentimentos.

Processo:

- Tokenização das avaliações com o BertTokenizer.
- Treinamento por 3 épocas com ajuste dos pesos.
- Otimizador AdamWeightDecay e função de perda de entropia cruzada.

Motivação:

- Aproveitar o aprendizado profundo contextual do BERT para melhorar a precisão.

Resultados - BERT

Metricas de Avaliação:

- Acurácia: 64,28%
- F1 Score: 63,97%

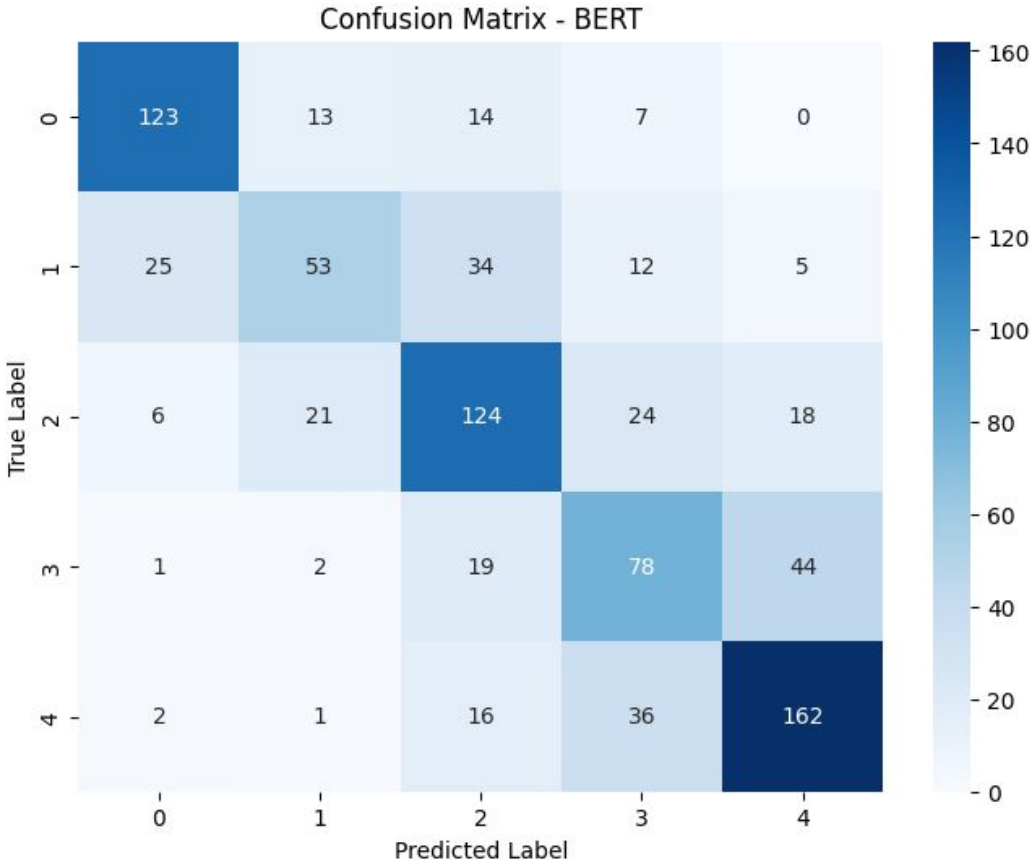
Análise:

- Desempenho superior aos modelos SVM, porém ainda modesto.
- Indica a complexidade da tarefa e possíveis limitações nos dados.

```python
import numpy as np
from sklearn.metrics import accuracy_score, f1_score

# Avaliação
predictions = model(X_test_tokens.data).logits

# Obtém os rótulos previstos a partir dos logits
predicted_labels = np.argmax(predictions, axis=1)

# Ajusta y_test para o intervalo 0-4 para avaliação (se necessário)
y_test_adjusted = y_test - 1

# Calcula as métricas
accuracy_bert = accuracy_score(y_test_adjusted, predicted_labels)
f1_bert = f1_score(y_test_adjusted, predicted_labels, average='weighted')
loss = loss_fn(y_test_adjusted, predictions) # Calcula a perda após o ajuste do y_test

print("\nBERT:")
print(f"Test Loss: {loss:.4f}") # Imprime a perda
print(f"Acurácia: {accuracy_bert:.4f}")
print(f"F1 Score: {f1_bert:.4f}")
```

```
BERT:
Test Loss: 1.1388
Acurácia: 0.6429
F1 Score: 0.6397
```

Resultados - BERT



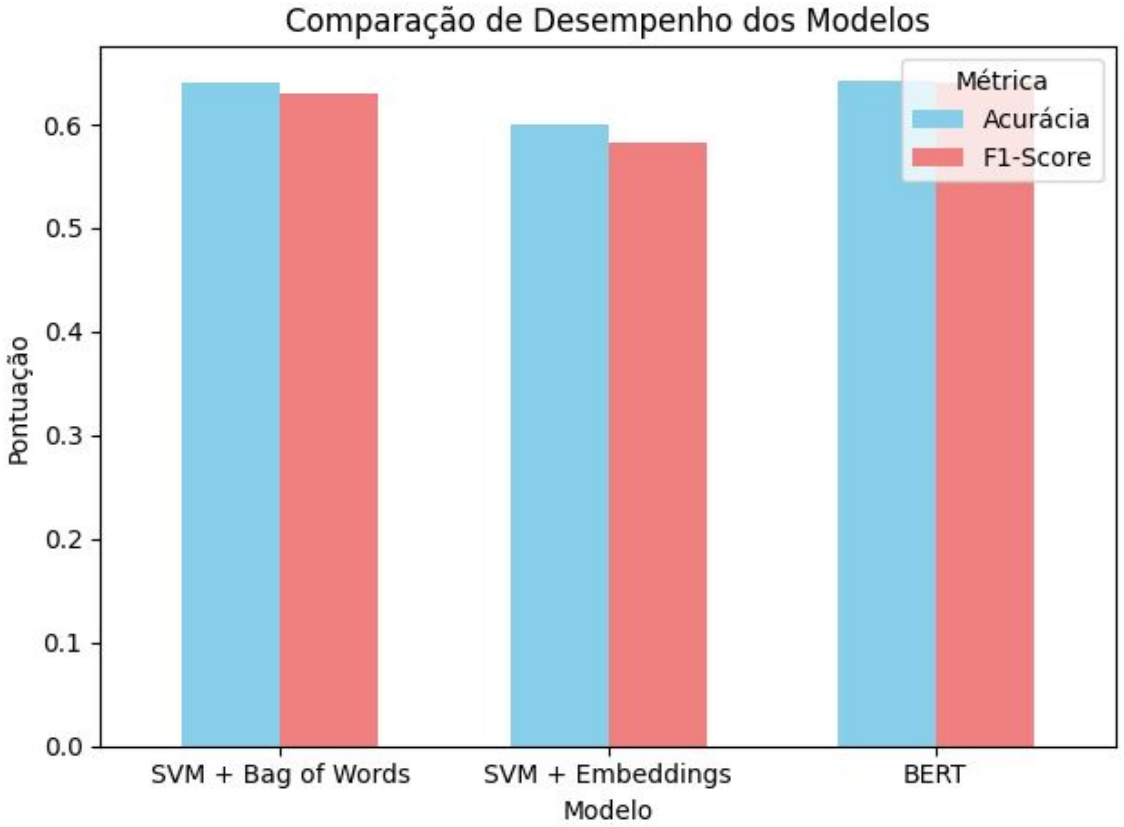
Comparação dos Modelos

Modelo	Acurácia	F1 Score
SVM + Bag of Words	64,04%	63,05%
SVM + Embeddings	60,00%	58,30%
BERT	64,04%	63,97%

Observações:

- O BERT, apesar de mais robusto, não atingiu alta performance.

Comparação dos Modelos



Modelo com In-Context Learning - OpenAI ChatGPT

- **Uso de LLMs (Large Language Models)**:
  - **OpenAI GPT-4**
  - **Google Gemini 1.5**

ChatGPT

Gemini

- **Método**:
  - Aplicação de **In-Context Learning**, fornecendo exemplos no prompt.
  - Classificação direta das avaliações sem treinamento adicional.

6.1 Using OpenAI (GPT-4)

```python
from google.colab import userdata
from openai import OpenAI

# Retorna a chave de API da OpenAI API do userdata
api_key = userdata.get('OPENAI_API_KEY')

# Inicializa o cliente da OpenAI
client = OpenAI(api_key=api_key)

def classify_with_openai(review_text):
    completion = client.chat.completions.create(
        model="gpt-4",
        messages=[
            {"role": "user", "content": prompt_template.format(review_text=review_text)}
        ]
    )
    return completion.choices[0].message.content.strip()

# Classifica 10 reviews aleatórias
random_reviews = df.sample(n=10)

print("Classificação da OpenAI GPT-4")
print("=" * 90)

for index, row in random_reviews.iterrows():
    review_text = row['review']
    classification = classify_with_openai(review_text)

    # Add emoji
    display_classification = add_emoji_to_classification(classification)

    # Compara o resultado com o sentimento correto
    result, is_correct = compare_classification_with_actual(classification, row['sentiment'])

    print(f"\nAvaliação: {review_text}")
    print(f"Classificação da OpenAI: {display_classification}")
    print("-" * 90)
    print(f"Sentimento Real: {row['sentiment'].capitalize()}")
    print(f"Resultado: {result}")
```

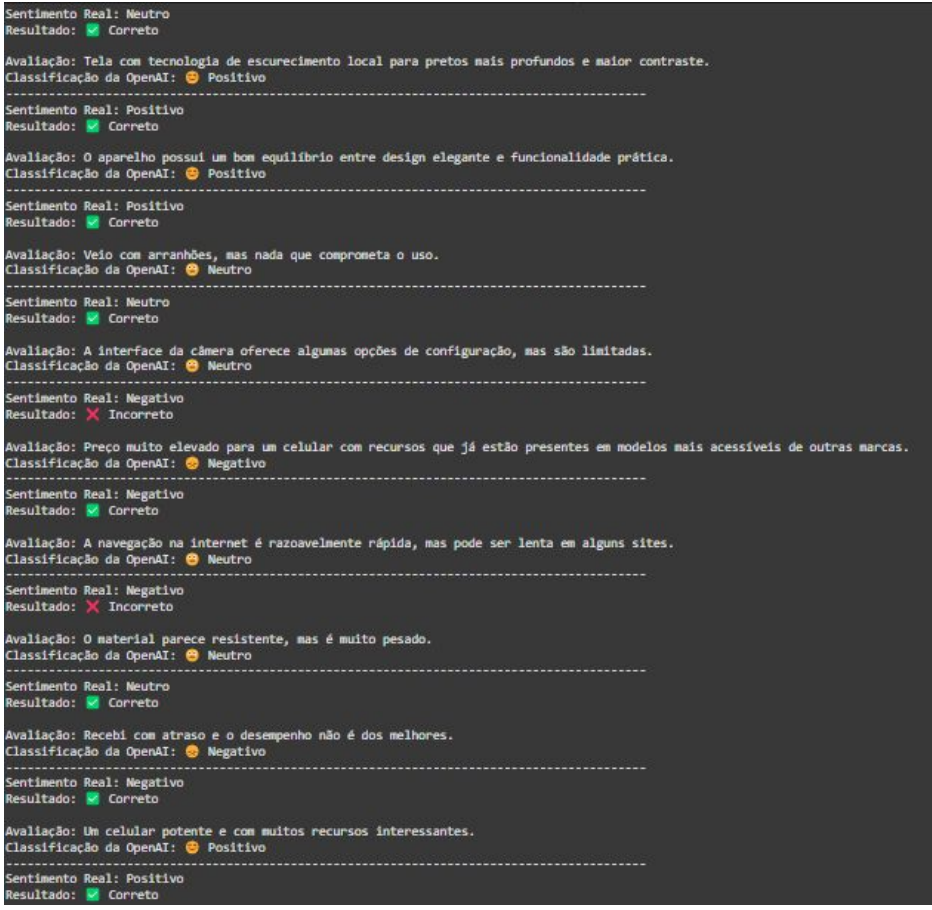
Loading...

Resultados com In-Context Learning - OpenAI ChatGPT

- **Resultados**:
    - Avaliações classificadas com base na compreensão contextual do modelo.
    - A Figura mostra exemplos de classificações com alta precisão do ChatGPT.

Observações:

- Potencial para melhorar a acurácia sem treinamento explícito.
- Dependência de modelos externos e possíveis custos associados.



Análise e Resultados com In-Context Learning - Google Gemini

6.2 Using Google Gemini 1.5

```python
from google.colab import userdata
import google.generativeai as genai

# Retorna a chave de API do Google Gemini do userdata
myKey = userdata.get('GOOGLE_API_KEY')
genai.configure(api_key=myKey)

# Espcifica o modelo do Gemini
model = genai.GenerativeModel("gemini-1.5-flash-latest")

def classify_with_gemini(review_text):
    response = model.generate_content(prompt_template.format(review_text=review_text))
    classification = response.text.strip()
    return classification

# Classifica 10 reviews aleatórias
random_reviews = df.sample(n=10)

print("Classificação do Google Gemini 1.5")
print("=" * 90)

for index, row in random_reviews.iterrows():
    review_text = row['review']
    classification = classify_with_gemini(review_text)

    # Adiciona o emoji
    display_classification = add_emoji_to_classification(classification)

    # Compara o resultado com o sentimento correto
    result, is_correct = compare_classification_with_actual(classification, row['sentiment'])

    print(f"\nAvaliação: {review_text}")
    print(f"Classificação do Gemini: {display_classification}")
    print("-" * 90)
    print(f"Sentimento Real: {row['sentiment'].capitalize()}")
    print(f"Resultado: {result}")
```

```
Classificação do Google Gemini 1.5
==========================================================================================

Avaliação: A embalagem veio danificada, mas o produto está em perfeitas condições.
Classificação do Gemini: 😐 Neutro
------------------------------------------------------------------------------------------
Sentimento Real: Positivo
Resultado: ❌ Incorreto

Avaliação: O celular é resistente a água e poeira, ideal para usar em qualquer ambiente.
Classificação do Gemini: 😊 Positivo
------------------------------------------------------------------------------------------
Sentimento Real: Positivo
Resultado: ✅ Correto

Avaliação: A tela com alta taxa de amostragem de toque oferece uma resposta rápida e precisa aos comandos, ideal
Classificação do Gemini: 😊 Positivo
------------------------------------------------------------------------------------------
Sentimento Real: Positivo
Resultado: ✅ Correto

Avaliação: O aparelho mantém a integridade dos dados durante sincronizações com serviços de armazenamento extern
Classificação do Gemini: 😊 Positivo
------------------------------------------------------------------------------------------
Sentimento Real: Positivo
Resultado: ✅ Correto

Avaliação: O aparelho suporta bem a reprodução de diferentes formatos de mídia.
Classificação do Gemini: 😊 Positivo
------------------------------------------------------------------------------------------
Sentimento Real: Neutro
Resultado: ❌ Incorreto

Avaliação: A câmera frontal não funciona direito para selfies.
Classificação do Gemini: 😠 Negativo
------------------------------------------------------------------------------------------
Sentimento Real: Negativo
Resultado: ✅ Correto

Avaliação: Esperava mais pelo preço que paguei.
Classificação do Gemini: 😠 Negativo
------------------------------------------------------------------------------------------
Sentimento Real: Negativo
Resultado: ✅ Correto

Avaliação: A embalagem estava perfeita e o produto chegou intacto.
Classificação do Gemini: 😊 Positivo
------------------------------------------------------------------------------------------
Sentimento Real: Positivo
Resultado: ✅ Correto

Avaliação: A qualidade do display para leitura de textos é confortável para os olhos, mesmo em longas sessões.
Classificação do Gemini: 😊 Positivo
------------------------------------------------------------------------------------------
Sentimento Real: Positivo
Resultado: ✅ Correto
```

Desafios e Limitações

- **Desbalanceamento de Classes**:
  - Predominância de avaliações positivas pode afetar o treinamento.
  - Necessidade de técnicas para lidar com classes minoritárias.

- **Recursos Computacionais**:
  - Modelos como BERT exigem maior capacidade de processamento.
  - Limitações em ambiente de desenvolvimento podem restringir experimentos.

Conclusões e Trabalhos Futuros

- **Principais Conclusões**:
  - O **BERT** apresentou melhor desempenho, mas ainda aquém do desejado.
  - Modelos simples têm limitações significativas na análise de sentimentos.
- **Sugestões para Melhorias**:
  - **Aumento e Diversificação do Dataset**:
    - Coletar mais dados para enriquecer o conjunto.
  - **Ajuste de Hiperparâmetros e Arquiteturas**:
    - Experimentar com diferentes configurações de modelos.
  - **Técnicas de Data Augmentation**:
    - Aumentar a variedade linguística e semântica das avaliações.
  - **Ensemble de Modelos**:
    - Combinar diferentes modelos para aprimorar a performance.

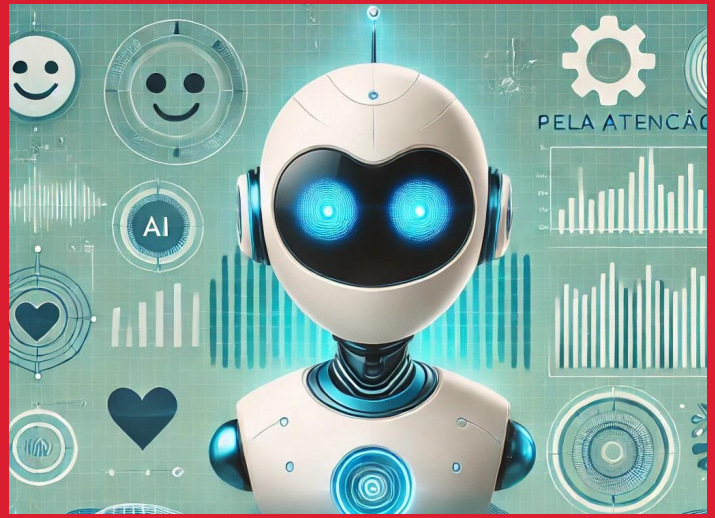
Referências

- **Ferramentas e Bibliotecas**:
  NLTK, spaCy, Scikit-learn, TensorFlow, Transformers.
- **Modelos Pré-treinados**:
  neuralmind/bert-base-portuguese-cased.

- **Fontes de Pesquisa**: Artigos e publicações sobre PLN e análise de sentimentos.

1. CASELI, H. de M.; NUNES, M. das G. V. Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português. 1. ed. São Paulo: Brasileiras em PLN, 2023. 563 p.
2. DEVLIN, J., CHANG, M., LEE, K., et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, CoRR, v. abs/1810.04805, 2018.
3. WEISS, S. M.; INDURKHYA, N.; ZHANG, T. Fundamentals of Predictive Text Mining. 2. ed. London: Springer International Publishing, 2015. 239 p.
4. ZONG, C.; XIA, R.; ZHANG, J. Text Data Mining. 2. ed. Singapore: Springer International Publishing and Tsinghua University Press, 2021. 351 p

Obrigado!



Contatos:

Paloma Correa Alves | pca2@cin.ufpe.br

Luciano Ayres Farias de Carvalho | lafc@cin.ufpe.br

Especialização
Deep Learning

Centro de Informática UFPE

Universidade Federal de Pernambuco