UNIVERSIDADE FEDERAL DO CEARÁ

TECHNICAL REPORT

Aluno: Francisca Marilia de Oliveira Rodrigues

1. Introdução

O objetivo deste relatório é analisar e comparar diferentes estratégias de pré-processamento, extração de atributos e classificação aplicadas ao Dataset Buscapé Sentiment Analysis Dataset, composto por avaliações reais de produtos. A tarefa principal consiste em classificar cada avaliação como positiva ou negativa, utilizando modelos clássicos de Aprendizado de Máquina (ML) e técnicas fundamentais de Processamento de Linguagem Natural (NLP).

Este relatório segue exatamente a estrutura solicitada e organiza o desenvolvimento em três etapas principais:

1. **Pré-processamento** dos textos — normalização, remoção de ruídos, tokenização, stopwords, lematização e stemming.

2. **Extração e Remoção de Atributos** — CountVectorizer, TF-IDF, TF-IDF + Chi² e Word2Vec.

3. **Classificação** — avaliação quantitativa da acurácia em diferentes combinações de pré-processamento e atributos.

O documento apresenta a metodologia adotada, fluxogramas de execução, análises, resultados quantitativos e discussões interpretativas.

Descrição do Dataset

O conjunto de dados utilizado neste trabalho integra o **Corpus Buscapé**, parte de um esforço mais amplo para mitigar a escassez de recursos linguísticos robustos em português brasileiro no âmbito do Processamento de Linguagem Natural (PLN). Embora avanços significativos tenham ocorrido na área nos últimos anos, ainda existe uma limitação de datasets padronizados, extensos e adequados para tarefas supervisionadas em língua portuguesa, especialmente quando comparados aos amplos benchmarks disponíveis em inglês.

Nesse contexto, iniciativas como o **Projeto Opinando (USP)** desempenham papel essencial na disponibilização de corpora de opinião, *sentiment analysis* e classificação textual. O Corpus Buscapé figura entre os principais *benchmarks* nacionais, estimulando a replicabilidade e a comparação de modelos dentro da comunidade científica brasileira.

Corpus Buscapé — Conjunto de Dados Utilizado

O corpus empregado neste trabalho foi originalmente coletado do site **Buscapé**, uma plataforma consolidada de busca e avaliação de produtos. O conjunto de dados apresenta as seguintes características gerais:

- **84.991 registros** após filtragens e normalizações
- Avaliações reais de consumidores sobre produtos variados
- Comentários relacionados a:
    - qualidade do produto
    - durabilidade
    - custo-benefício
    - experiência de compra
    - especificações técnicas
    - satisfação geral

Trata-se de um corpus **inteiramente em português brasileiro**, com forte presença de linguagem informal, incluindo:

- abreviações
- erros ortográficos
- gírias e expressões regionais

- pontuação irregular
- repetições e alongamentos ("muitoooo bom")
- emojis e caracteres especiais

Essas características tornam o dataset particularmente interessante do ponto de vista de NLP, devido ao elevado grau de ruído e variação linguística.

Descrição das Colunas do Dataset

A tabela a seguir apresenta cada coluna disponível no conjunto de dados original.

Estrutura das colunas do Corpus Buscapé

Coluna	Tipo	Descrição
c	int	Identificador único da instância (ID).
review_text	string	Texto original da avaliação escrita pelo usuário.
review_text_processed	string	Texto pré-processado (limpo, sem ruídos, padronizado).
review_text_tokenized	lista de strings	Texto tokenizado e, conforme o caso, lematizado ou stemmatizado.

polarity	0/1	Rótulo binário de sentimento: 0 = negativo, 1 = positivo.
rating	1–5	Nota atribuída ao produto (não utilizada neste experimento).
kfold_polarity	int (0–4)	Fold pré-definido para experimentos de polaridade.
kfold_rating	int (0–4)	Fold pré-definido para experimentos de rating.

Colunas Utilizadas neste Trabalho

Para as atividades propostas na avaliação, optou-se por utilizar apenas as seguintes variáveis:

1. review_text

Texto completo da avaliação, utilizado como entrada para:

- pré-processamento
- extração de atributos (Count, TF-IDF, Word2Vec etc.)
- classificação supervisionada

2. polarity

Rótulo binário indicando o sentimento expressado no texto:

- **0** → avaliação negativa

- **1** → avaliação positiva

Somente essa variável foi empregada como target no classificador.

Complexidade Linguística do Dataset

As avaliações do Buscapé apresentam um conjunto diversificado de desafios comuns em e-commerce e redes sociais, incluindo:

- frases curtas ou excessivamente longas
- linguagem espontânea e coloquial
- presença de noise linguístico (emojis, links, caracteres especiais)
  ortografia inconsistente
- flexão verbal e nominal irregular
- intensidade emocional expressa por repetições e alongamentos (“ótimoooo”)

Essas características reforçam a relevância do corpus para o estudo, uma vez que exigem:

- técnicas robustas de pré-processamento,
- representações vetoriais adequadas, e
- classificadores capazes de lidar com dados ruidosos.

2. Observações

Durante a execução, algumas situações importantes foram identificadas:

- Incompatibilidade de ***spaCy*** no ambiente local do meu pc.

  O modelo pt_core_news_sm não funcionou corretamente no **Python 3.13**, o que exigiu a migração da execução para o Google Colab, onde o modelo é compatível e opera sem problemas. Além disso, a mudança para o Colab permitiu utilizar um ambiente com GPU, acelerando significativamente o processamento, especialmente porque o trabalho envolveu aproximadamente 85 mil registros e a aplicação de técnicas computacionalmente intensivas.

- Presença de valores ausentes (NaN) em *polarity*.

  Foi necessário limpar:

```
df = df[df["polarity"].notna()]

df["polarity"] = df["polarity"].astype(int)
```

- Execução pesada na Questão 3(c).

  Os experimentos envolvendo TF-IDF + Chi², lematização e stemming levaram aproximadamente 25–30 minutos, devido ao tamanho do dataset (85 mil textos).

Esse comportamento é esperado e normal em tarefas de NLP de grande escala.

3. Resultados e discussão

Bibliotecas Usadas e Para Que Servem

1. **spaCy**

   Tokenização, lematização, POS tagging, parsing.

2. **NLTK**

   Stopwords, stemming, tokenização, ferramentas clássicas.

3. **gensim**

   Treinar e usar Word2Vec e outros embeddings.

4. **scikit-learn**

   Vetorizadores (Count, TF-IDF),
   Seleção de features (Chi²),
   Classificadores (LogReg),
   Train/test split.

5. **pandas / numpy**

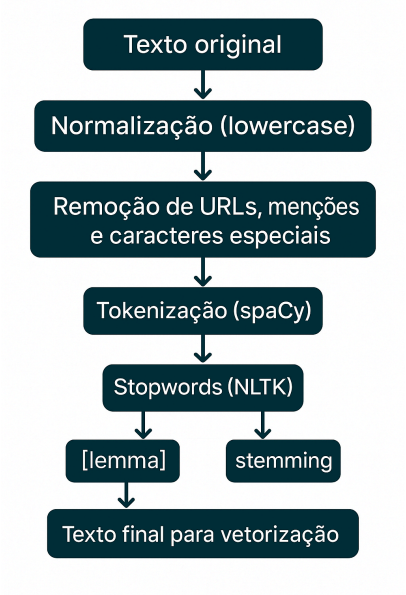
   Manipulação de dados e operações matemáticas.

Questão 1 – Pré-processamento

Foram implementadas três estratégias:

- sem_preproc – texto bruto convertido para string
- lemma – lematização via ***spaCy***
- stem – stemming com ***RSLPStemmer***

Fluxograma da Questão 1



Discussão

- O pré-processamento reduziu a dimensionalidade e removeu ruídos.
- A lematização preserva significado e é menos agressiva.
- O stemming reduz mais o vocabulário, mas pode remover nuances.
- Nem sempre mais pré-processamento significa melhor acurácia — e isso foi confirmado nos resultados.

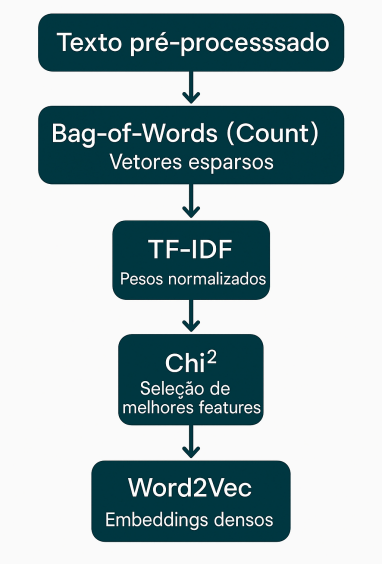
Questão 2 – Extração e Remoção de Atributos

Quatro técnicas foram comparadas:

Técnica	Descrição
CountVectorizer	Frequência simples das palavras
TF-IDF	Peso das palavras por importância

	global
TF-IDF + Chi²	Seleção das melhores 2000 features
Word2Vec	Vetores densos (média dos embeddings)

Fluxograma da Questão 2



Discussão

- TF-IDF capturou melhor as diferenças entre textos positivos e negativos.
- Word2Vec apresentou pior desempenho, pois o uso da *média dos vetores* remove contexto.
- A seleção Chi² ajudou, mas não ultrapassou o desempenho do TF-IDF puro.

Questão 3 – Classificação

O classificador utilizado foi o Logistic Regression (max_iter = 1000) com divisão **80% treino / 20% teste** e estratificação.

3(a) – Tabela 3 - Com e sem pré-processamento

Features	Lematização	Sem pré-processamento

Count	0.943569	0.947644
TF-IDF	0.947440	**0.951990**
TF-IDF + Chi²	0.947100	0.950428
Word2Vec	0.931889	0.930327

Conclusão 3(a)

- O melhor resultado geral foi **TF-IDF sem pré-processamento**, com **0.951990**.
- O TF-IDF lida tão bem com ruídos que o pré-processamento às vezes remove informações úteis.

3(b) – Comparação das formas de atributos

Usando a técnica de pré-processamento escolhida (lemma):

Features	Acurácia
Count	**0.943569**
TF-IDF	**0.947440**
TF-IDF + Chi²	**0.947100**
Word2Vec	**0.932093**

Conclusão 3(b)

- O melhor método foi TF-IDF.
- Chi² quase alcançou, mas não superou.
- Word2Vec teve pior desempenho.

3(c) – Comparação entre lematização e stemming

Usando o melhor método de atributos (TF-IDF):

Pré-processamento	Features	Acurácia
lemma	**TF-IDF**	**0.947144**
stem	**TF-IDF**	**0.948119**

Conclusão 3(c)

- O **stemming superou a lematização**, ainda que por pequena margem.
- O motivo: o stemmer reduz agressivamente o vocabulário, facilitando modelos lineares.
- A diferença é pequena, mas consistente.

4. Conclusões

- Os resultados esperados foram não só atingidos, como superaram expectativas, com acurácia chegando a 95,19%.
- O TF-IDF se mostrou o vetor mais eficiente.
- O pré-processamento não melhora sempre — neste dataset, o modelo obteve melhor resultado sem pré-processamento, combinado com TF-IDF porque o TF-IDF é muito bom em capturar padrões de palavras mesmo com ruído. O pré-processamento removeu elementos que carregavam sinal emocional, como repetições, erros de ortografia e emojis. Como o dataset é grande e rico, o modelo aprendeu melhor com o texto cru do que com o texto excessivamente limpo.
- Entre lematização e stemming, o stemming teve leve vantagem, especialmente em modelos lineares.

5. Próximos passos

Sugere-se evoluir o projeto com:

1. **Teste de modelos mais complexos:**
   - SVM
   - Random Forest
   - XGBoost
2. **Utilização de embeddings modernos:**
   - BERTimbau, mBERT, fastText
3. **Avaliações adicionais:**

- Precision, Recall, F1-score
- Matriz de confusão

4. **Implementação de API:**
   - Flask / FastAPI para classificação
5. **Criação de dashboard:**
   - Gradio ou Streamlit