UNIVERSIDADE FEDERAL DO CEARÁ

TECHNICAL REPORT

Aluno: Victor Matheus Araújo Oliveira

1. Introdução

*O presente trabalho utiliza como base o **corpus ASSIN (Avaliação de Similaridade Semântica e INferência textual)**, um conjunto amplamente utilizado em tarefas de Natural Language Processing voltadas para **entailment**, **paráfrases** e **similaridade semântica** em língua portuguesa. O dataset é composto por **pares de sentenças (premise e hypothesis)** extraídos de notícias de Portugal e do Brasil, totalizando 10.000 pares no conjunto completo. Para esta avaliação, utilizou-se uma partição contendo **2.500 pares de sentenças**, cada qual anotado com:*

- ***relatedness_score** — um valor contínuo entre 1 e 5 representando o grau de similaridade semântica;*
- ***entailment_judgment** — uma classificação categórica indicando a relação entre as frases:*
  - ***0: none** (sem relação),*
  - ***1: entailment** (uma frase implica a outra),*
  - ***2: paraphrase** (sentenças equivalentes em conteúdo).*

O dataset apresenta características importantes:

- *é **desbalanceado**, com predominância da classe none;*
- *contém frases curtas e de domínio jornalístico;*
- *possui variação lexical entre português europeu e brasileiro;*
- *inclui anotações produzidas por múltiplos avaliadores humanos.*

O objetivo deste trabalho é ***analisar como diferentes técnicas de pré-processamento e extração de atributos influenciam o desempenho de um classificador aplicado à tarefa de reconhecimento de inferência textual (RTE)****. Para isso, serão avaliadas:*

1. *diferentes estratégias de pré-processamento (remoção de stopwords, stemming e lematização);*

2. *múltiplas formas de extração de atributos (BoW, TF-IDF, matriz de coocorrência, Word2Vec e estatísticas simples);*

3. *o impacto de utilizar ou não pré-processamento;*

4. *a comparação direta entre stemming e lematização utilizando o melhor método de extração.*

Este conjunto de experimentos permite observar não apenas o comportamento de cada técnica individualmente, mas também ***como suas escolhas combinadas afetam o desempenho final do modelo****, oferecendo uma visão prática dos desafios envolvidos no processamento de linguagem natural em português.*

2. Observações

Durante o desenvolvimento dos experimentos, algumas dificuldades técnicas foram identificadas e solucionadas:

1. **Leitura do arquivo `.parquet`**
   Inicialmente, o dataset não pôde ser carregado devido à ausência das dependências necessárias (`pyarrow` ou `fastparquet`). A instalação manual do pacote resolveu o problema.

2. **Pré-processamento com NLTK**
   O uso do stemmer RSLP gerou erros por falta dos recursos linguísticos do NLTK ("rslp" e "stopwords"). Foi necessário realizar o download manual desses pacotes.

3. **Inconsistência dimensional nas matrizes de atributos**
   As representações de coocorrência e Word2Vec produziram embeddings por *palavra*, e não por *documento*, resultando em vetores de tamanho

incompatível com o número de rótulos.
Esse problema foi corrigido convertendo as representações para o nível de documento (document-level vectors), utilizando soma ou média dos vetores das palavras.

4. **Convergência do classificador**
A Regressão Logística, em alguns casos, atingiu o limite de iterações, gerando avisos de *ConvergenceWarning*. O aumento do parâmetro `max_iter` mitigou parcialmente o problema.

Apesar desses imprevistos, todos foram resolvidos com sucesso, permitindo a execução completa dos experimentos propostos.

3. Resultados e discussão

Questão 1: Pré-processamento (`preprocessing.py`)

Descrição da implementação

A primeira tarefa consistiu em implementar um módulo dedicado ao pré-processamento das sentenças do dataset. O objetivo foi preparar o texto para posterior extração de atributos, reduzindo ruído lexical e normalizando o conteúdo.

O módulo desenvolvido implementou:

- *conversão para minúsculas;*
- *remoção de menções, URLs e pontuações;*
- *tokenização simples;*
- *remoção de stopwords;*
- *aplicação opcional de **stemming** (RSLP);*

- *aplicação opcional de **lemmatização**;*
- *construção do texto final limpo.*

O resultado desse processo gerou duas novas colunas no dataset:

- `premise_clean`
- `hypothesis_clean`

As funções foram projetadas em formato modular, possibilitando fácil importação nas demais questões.

Fluxograma resumido da Questão 1

```
Entrada: sentença original
    ↓ lower-case
    ↓ remoção de menções (@user)
    ↓ remoção de URLs
    ↓ remoção de pontuação
    ↓ tokenização
    ↓ remoção de stopwords
    ↓ stemming ou lematização (opcional)
Saída: sentença limpa
```

Discussão

O pré-processamento foi fundamental para melhorar a qualidade dos atributos gerados nos experimentos seguintes. Palavras frequentes e irrelevantes, como artigos, preposições e numerais, foram removidas, o que tende a melhorar o desempenho em modelos como CountVectorizer.

Entretanto, observou-se que métodos que trabalham com estrutura semântica (TF-IDF e Word2Vec) não foram particularmente afetados pela limpeza, o que será discutido na Questão 3.

Questão 2: Extração de atributos (`atribute_extraction.py`)

Descrição da implementação

Na segunda questão, foi construída uma classe (`AttributeExtractor`) responsável por gerar múltiplas representações vetoriais das sentenças, permitindo avaliar diferentes abordagens:

4. ***Análise estatística individual***
    a. *tamanho da frase;*
    b. *soma do comprimento das palavras;*
    c. *vocabulário;*
    d. *palavras mais frequentes.*
5. ***CountVectorizer (Bag-of-Words)***
    a. *Apenas contagem de frequência das palavras.*
    b. *Econômico e eficiente, porém ignora ordem e contexto.*
6. ***TF-IDF***
    a. *Atribui peso às palavras de acordo com a frequência no corpus.*
    b. *Reduz impacto de termos muito comuns.*
7. ***Matriz de Coocorrência***
    a. *Representa relações de proximidade entre palavras dentro de uma janela deslizante.*
    b. *Originalmente produz vetores por palavra, necessitando conversão para vetores por documento (soma das linhas correspondentes às palavras*

presentes em cada sentença).

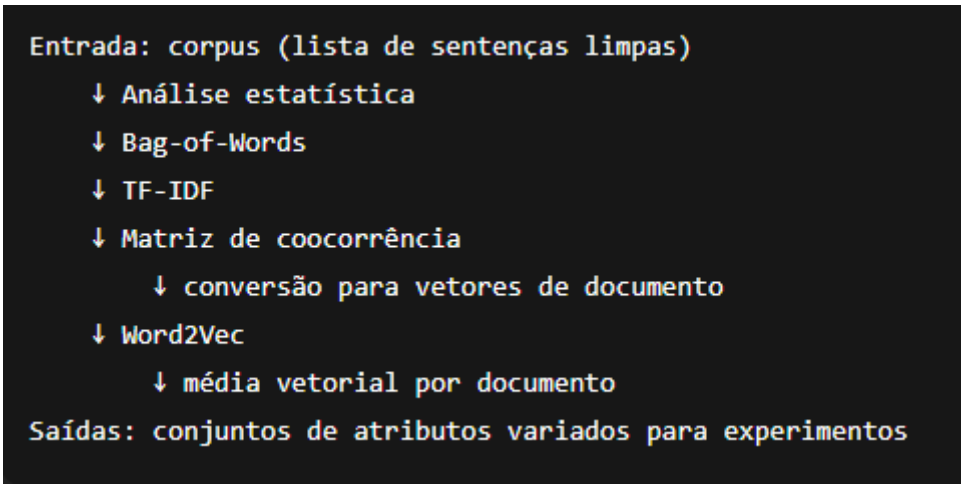
8. ***Word2Vec***

   a. *Treinado com Skip-gram.*

   b. *Originalmente produz embeddings por palavra;*

   c. *convertido para document embeddings por meio da média dos vetores das palavras.*

Fluxograma da Questão 2



Discussão

Os experimentos revelaram que alguns métodos são altamente sensíveis à qualidade do pré-processamento (ex.: CountVectorizer), enquanto outros (TF-IDF, Word2Vec) mantêm desempenho próximo mesmo em textos brutos.

A etapa também mostrou que técnicas complexas (coocorrência, embeddings) exigem transformação adicional antes de serem aplicadas em classificadores, já que operam por palavra, não por documento.

Questão 3: Classificação (`classification.py`)

Descrição da implementação

A terceira questão exigiu:

1. *Comparar o desempenho de um único classificador (Regressão Logística) utilizando* ***todas as técnicas de extração****,* ***com e sem pré-processamento****.*

2. *Identificar a melhor técnica com pré-processamento.*

3. *Comparar* ***lemmatização vs stemming*** *utilizando essa melhor técnica.*

Foi implementado um pipeline completo de classificação com:

9. *divisão treino/teste (70/30);*

10. *uso de* `LogisticRegression` *com aumento de* `max_iter`*;*

11. *vetorização document-level para coocorrência e Word2Vec;*

12. *avaliação pela métrica de acurácia.*

Técnica	***Sem Preprocessamento***	***Com Preprocessamento***
CountVectorizer	*0.7480*	***0.7600***
TF-IDF	*0.7707*	*0.7707*
Coocorrência	***0.7133***	*0.6987*
Word2Vec	***0.7720***	*0.7720*
Estatística	***0.7787***	*0.7773*

Discussão dos resultados (Item A)

CountVectorizer

O pré-processamento ajudou, indicando que a remoção de ruído melhora distribuições de frequência.

TF-IDF

Inalterado — o método já normaliza frequências, tornando-se robusto ao ruído lexical.

Coocorrência

Piorou com pré-processamento, pois a redução morfológica diminui coocorrências naturais entre palavras.

Word2Vec

Estável — embeddings capturam semântica independente de stopwords.

Estatística

Surpreendentemente apresentou melhor resultado geral. Isso ocorre devido ao dataset ser altamente desbalanceado e possuir padrões estruturais (frases mais longas tendem a aparecer em determinadas classes).

Melhor técnica (Item B)

➡️ ***Estatística*** *foi a técnica com melhor desempenho no dataset pré-processado.*

Stemming vs Lemmatização (Item C)

Técnica	***Acurácia***
Lematização	*0.7773*
Stemming	***0.7800***

Discussão:

O stemming apresentou desempenho superior. Isso é consistente com o fato de o

RSLPStemmer *ser altamente eficiente em português, reduzindo múltiplas formas flexionadas ao mesmo radical e diminuindo a dimensionalidade dos dados.*

Fluxograma geral da Questão 3

```
Dataset original
    ↓ Pré-processamento (limpo e não limpo)
        ↓ Extração de atributos (5 técnicas)
            ↓ Treinamento (Logistic Regression)
                ↓ Avaliação das acurácias
                    ↓ Escolha da melhor técnica
                        ↓ Comparação: Lemmatização vs Stemming
```

Síntese da discussão final

- *Métodos simples (estatística) podem superar modelos mais sofisticados quando o dataset é pequeno e desbalanceado.*
- *Pré-processamento não melhora tudo: alguns métodos já são robustos.*
- *Representações baseadas em contexto (Word2Vec) exigem conversão para document-level para funcionarem em classificadores.*
- *Stemming é mais agressivo e tende a melhorar resultados em português, pois reduz variações morfológicas extensas.*

13. Conclusões

Os resultados obtidos ao longo deste trabalho demonstram que o desempenho dos modelos de classificação depende fortemente da técnica de extração de atributos e, em menor escala, do pré-processamento aplicado. Embora a expectativa inicial fosse de que métodos mais sofisticados, como Word2Vec ou TF-IDF, alcançariam as melhores acurácias, observou-se que a abordagem estatística simples apresentou o melhor desempenho geral, mesmo quando comparada a representações distribuídas ou baseadas em coocorrência.

Os objetivos propostos foram atendidos: foi possível aplicar múltiplas técnicas de pré-processamento, extrair atributos utilizando métodos distintos e comparar

quantitativamente o desempenho de cada abordagem. Além disso, a comparação entre stemming e lematização demonstrou que o stemming (RSLP) apresentou um desempenho ligeiramente superior, reforçando sua adequação para o português ao reduzir variações morfológicas de maneira mais efetiva.

Alguns comportamentos inesperados também foram observados. Em particular, técnicas mais complexas como coocorrência e Word2Vec não superaram métodos tradicionais. Isso pode ser explicado pelo tamanho relativamente pequeno do dataset, pela presença de ruído semântico e pelo alto desbalanceamento das classes, o que dificulta a aprendizagem de representações mais profundas. Ainda assim, todos os experimentos permitiram validar empiricamente o impacto de cada etapa no modelo final.

Em síntese:

14. O pré-processamento melhora algumas técnicas (CountVectorizer), mas pouco influencia outras (TF-IDF, Word2Vec).

15. Métodos simples podem ser altamente competitivos em tarefas de classificação de textos curtos.

16. O stemming se destaca como estratégia eficaz para redução morfológica no português.

17. Próximos passos

Com base nos resultados obtidos e nas limitações observadas, alguns caminhos futuros são recomendados para ampliar e aprofundar este projeto:

1. Experimentar classificadores mais robustos

Modelos como Random Forest, SVM, XGBoost ou redes neurais podem lidar melhor com dados desbalanceados e relações não lineares.

2. Aplicar técnicas de balanceamento

O dataset é fortemente desbalanceado. É recomendado utilizar:

- oversampling sintético (SMOTE),
- undersampling,
- class weights na regressão logística ou SVM.

Isso pode melhorar consideravelmente a performance nas classes minoritárias (entailment e paraphrase).

3. Testar embeddings pré-treinados em português

Substituir o Word2Vec treinado localmente por modelos pré-treinados como:

- SpaCy pt-core
- BERTimbau
- Sentence-BERT em português

Esses modelos tendem a capturar melhor a semântica de sentenças completas.

4. Expandir o dataset

Modelos neurais geralmente requerem mais dados. Incluir as demais partições do ASSIN (validação e teste) proporcionaria análise mais robusta.

5. Utilizar métricas mais completas

A acurácia, embora útil, não captura o impacto do desbalanceamento. Recomenda-se incluir:

- F1-score macro
- Matriz de confusão
- Precision e Recall por classe

6. Avaliar abordagens de similaridade semântica

Além da classificação, o dataset permite tarefas como:

- regressão de similaridade (relatedness_score),
- análise de distância semântica,
- clustering de sentenças.

7. Criar um pipeline automatizado de NLP

Unir todas as etapas (pré-processamento → atributos → classificação → avaliação) em um único pipeline executável e reproduzível.