Desenvolvimento de Sistema de Análise de Sentimento Probabilístico com Java (NLP)

Este documento tem como objetivo apresentar uma aplicação em Java que analisa o sentimento de um texto em português e identifica entidades nomeadas utilizando técnicas básicas de Processamento de Linguagem Natural (NLP).

O sistema funciona com base em contagem de palavras positivas e negativas, além de um processo manual de pré-processamento textual.

Tecnologias Utilizadas

- **Linguagem de Programação:** Java 8+
- **IDE Sugerida:** Eclipse, IntelliJ ou VS Code com extensão Java
- **Bibliotecas externas:** Nenhuma (100% Java nativo)

Estrutura do Projeto

Pacote: com.nlp.teste
Classe Principal: Nlp_Probabilistica
Módulos internos (métodos):

- processText(String text) - Pré-processamento textual
- countWordFrequency(List<String> words) - Contagem de palavras
- sortByFrequency(Map<String, Integer> wordFrequency) - Ordenação por frequência
- stem(String word) - Redução de palavras (stemming)
- analyzeSentiment(List<String> words) - Análise de sentimento
- extractEntities(String text) - Extração de entidades

Fluxo de Desenvolvimento e Explicação Técnica

Entrada de Dados

- **Tipo:** String com múltiplas sentenças

Exemplo:

String text = "A inteligência artificial é incrível! Programar NLP com Java é desafiador, mas gratificante.";

- **Função:** Contextualizar o sistema com um texto real para ser analisado.

Etapas do Processamento

Etapa 1: Impressão do Texto Original

- Apenas imprime o texto recebido:

System.out.println("Texto: " + text);

Etapa 2: Pré-processamento de Texto (processText)

- **Objetivo:** Normalizar, limpar e segmentar o texto.

Código Base:

```
private static List<String> processText(String text) {
    text = text.toLowerCase();
    text = text.replaceAll("[^a-zA-Záéíóúãõâêîôûç ]", "");
    String[] tokens = text.split("\\s+");
    List<String> processedWords = new ArrayList<>();
    for (String word : tokens) {
        if (!STOPWORDS.contains(word)) {
            String stemmedWord = stem(word);
            processedWords.add(stemmedWord);
        }
    }
    return processedWords;
}
```

Etapa 3: Contagem de Frequência (countWordFrequency)

Código Base:

```
private static Map<String, Integer> countWordFrequency(List<String> words) {
    Map<String, Integer> wordCount = new HashMap<>();
    for (String word : words) {
        wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
    }
    return wordCount;
}
```

Etapa 4: Ordenação por Frequência (sortByFrequency)

Código Base:

```
private static List<Map.Entry<String, Integer>> sortByFrequency(Map<String, Integer>
wordFrequency) {
    List<Map.Entry<String, Integer>> sortedWords = new
ArrayList<>(wordFrequency.entrySet());
    sortedWords.sort((e1, e2) -> e2.getValue().compareTo(e1.getValue()));
    return sortedWords;
}
```

Etapa 5: Análise de Sentimento (analyzeSentiment)

Código Base:

```
private static String analyzeSentiment(List<String> words) {
    int positiveCount = 0, negativeCount = 0;
    for (String word : words) {
        if (POSITIVE_WORDS.contains(word)) positiveCount++;
        else if (NEGATIVE_WORDS.contains(word)) negativeCount++;
    }
    if (positiveCount > negativeCount) return "Positivo 😊";
    else if (negativeCount > positiveCount) return "Negativo 😞";
    else return "Neutro 😐";
}
```

Etapa 6: Extração de Entidades Nomeadas (extractEntities)

Código Base:

```
private static Set<String> extractEntities(String text) {
    Set<String> namedEntities = new HashSet<>();
    String[] words = text.split("\\s+");
    for (String word : words) {
        if (Character.isUpperCase(word.charAt(0)) && word.length() > 1) {
            namedEntities.add(word.replaceAll("[^a-zA-Záéíóúãõâêîôûç]", ""));
        }
    }
    return namedEntities;
}
```

Etapa 7: Exibição dos Resultados (Main)

```
System.out.println("Análise de Sentimento: " + sentiment);
System.out.println("Entidades Nomeadas: " + namedEntities);
```

Saídas Esperadas (Outputs)

No console:

- Texto original
- Lista processedWords
- Mapa wordFrequency
- Lista sortedWords
- Resultado da função analyzeSentiment
- Resultado da função extractEntities
- Listas de referência (POSITIVE_WORDS, NEGATIVE_WORDS, STOPWORDS)

Considerações

Este sistema utiliza um modelo simplificado de bag-of-words (saco de palavras), onde a análise textual é feita com base na contagem de ocorrências de palavras específicas sem considerar a posição, o contexto ou a estrutura sintática das sentenças. A estratégia é complementada com regras manuais como listas de palavras positivas e negativas, além de uma abordagem simples para a detecção de entidades nomeadas com base na capitalização.

Por sua simplicidade e clareza, este projeto é altamente indicado como ponto de partida para fins didáticos e experimentação inicial em NLP com Java, especialmente para iniciantes que desejam compreender os fundamentos do pré-processamento, análise de frequência e classificação de sentimento.

Sugestões de Melhoria:

1. **Lematização com Bibliotecas Externas:**
   - Em vez de aplicar regras manuais de stemming, utilizar bibliotecas como *OpenNLP*, *Lucene* ou *Morfologik* para realizar lematização real (redução da palavra à sua forma canônica).

Exemplo:

```
LemmatizerME lemmatizer = new LemmatizerME(lemmatizerModel);
String[] lemmas = lemmatizer.lemmatize(tokens, posTags);
```

2. **Análise Semântica Baseada em Contexto:**
   - Utilizar análise de dependência ou modelos baseados em embeddings (ex: Word2Vec, GloVe) para entender o contexto semântico das palavras.

Pode-se também empregar regras como:

```
if (word.equals("não") && nextWord.equals("gostei")) {
    negativeCount++;
}
```

3. **Classificação com Algoritmos Supervisionados:**
   - Substituir as regras fixas por um modelo treinado com dados rotulados.
   - Pode-se usar bibliotecas como Weka (Java) ou treinar modelos em Python e integrar via API.

Exemplo conceitual:

```
String sentiment = model.predict(features);
```

4. **Ferramentas de NLP Consolidadas:**
   - Apache OpenNLP ou Stanford CoreNLP oferecem modelos prontos para tokenização, POS tagging, lematização, análise sintática e NER (Named Entity Recognition).

Caso o projeto seja estendido via Python, recomenda-se usar **spaCy** e integrá-*lo por REST:*

```
HttpRequest request = HttpRequest.newBuilder()
.uri(URI.create("http://localhost:8000/spacy"))
.POST(HttpRequest.BodyPublishers.ofString(text))
.build();
```

EducaCiência FastCode para a comunidade