EducaCiência FastCode

Implementação de Análise de Sentimento com Naive Bayes usando WEKA em Java

Este documento detalha o passo a passo para criar e treinar um classificador de análise de sentimento utilizando a biblioteca WEKA em Java.

O classificador é baseado no algoritmo Naive Bayes e realiza a conversão de texto em vetores de palavras (bag-of-words) para realizar a classificação.

Objetivo

O objetivo deste código é classificar frases fornecidas pelo usuário em sentimentos, como positivo ou negativo, utilizando um modelo Naive Bayes treinado previamente com dados rotulados.

Passo a Passo de Implementação

1. Configuração Inicial

Certifique-se de ter o NetBeans IDE e o Java 8 instalados. Além disso, adicione as bibliotecas do WEKA ao seu projeto, caso ainda não tenha feito isso.

- Baixe a biblioteca WEKA em: https://waikato.github.io/weka-wiki/downloading_weka/

- Adicione o arquivo .jar da biblioteca WEKA às dependências do seu projeto no NetBeans.

2. Estrutura do Código

Importações

O código importa as classes necessárias da biblioteca WEKA para carregar e processar o conjunto de dados, aplicar filtros, treinar o classificador Naive Bayes e realizar a classificação de novas instâncias.

```
import java.util.Scanner;
import weka.classifiers.bayes.NaiveBayes;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.StringToWordVector;
import weka.core.DenseInstance;
```

3. Carregando o Conjunto de Dados

O arquivo .arff contendo os dados rotulados de sentimentos é carregado para o programa. Este arquivo deve conter textos já classificados como positivos ou negativos.

```
DataSource source = new
DataSource("C:\\caminho\\para\\seu\\arquivo.arff");
Instances dataset = source.getDataSet();
```

4. Definindo o Atributo de Classe

O código define que o último atributo no conjunto de dados será o atributo de classe (ou seja, o rótulo que representa o sentimento).

dataset.setClassIndex(dataset.numAttributes() - 1);

5. Convertendo Texto em Vetores de Palavras

O texto do dataset precisa ser convertido em um formato que o algoritmo Naive Bayes consiga processar. Para isso, usamos o filtro StringToWordVector, que transforma o texto em vetores de palavras (bag-of-words).

StringToWordVector filter = new StringToWordVector();
filter.setInputFormat(dataset);
Instances filteredData = Filter.useFilter(dataset, filter);

6. Treinando o Classificador Naive Bayes

Depois que o texto foi convertido, o classificador Naive Bayes é criado e treinado com o conjunto de dados filtrado.

NaiveBayes classifier = new NaiveBayes();
classifier.buildClassifier(filteredData);

7. Classificando uma Nova Frase

O código permite que o usuário digite uma nova frase no console para ser classificada pelo modelo. A frase é transformada em uma nova instância, que é adicionada ao conjunto de dados filtrado e processada pelo classificador.

Scanner scanner = new Scanner(System.in);
System.out.println("Digite uma frase para classificar:");

String newText = scanner.nextLine();
scanner.close();

A nova frase é então convertida no mesmo formato do dataset original (vetores de palavras) e classificada pelo modelo Naive Bayes.

Instance newInstance = new
DenseInstance(emptyDataset.numAttributes());
newInstance.setDataset(emptyDataset);
newInstance.setValue(emptyDataset.attribute(0), newText);

O filtro StringToWordVector é aplicado ao novo texto antes da classificação:

Instances filteredTestData = Filter.useFilter(emptyDataset, filter);

8. Exibindo o Resultado da Classificação

Por fim, o código exibe a classificação do novo texto (positivo ou negativo).

double label = classifier.classifyInstance(filteredInstance);
System.out.println("Classificação prevista: " +
dataset.classAttribute().value((int) label));

Output do Código

Quando o código é executado, ele exibe a seguinte sequência de etapas no console:

Carregando o conjunto de dados...

Conjunto de dados carregado com sucesso!
Definindo o atributo que será classificado...
Atributo de classe definido: sentiment
Convertendo atributos de texto em vetores de palavras...
Atributos de texto convertidos com sucesso!
Treinando o modelo NaiveBayes...
Modelo treinado com sucesso!
Digite uma frase para classificar:
adorei o produto
Classificando o novo exemplo de texto: ***adorei o produto***
Classificação prevista: positive

Exemplo de Execução

O código pode ser testado inserindo frases no console.

A classificação será exibida de acordo com o modelo treinado:

- Entrada: adorei o produto
- Saída: positive
- Entrada: não gostei do serviço
- Saída: negative

SentimentAnalysis.java

Source | History

```
33          System.out.println("Atributos de texto convertidos com sucesso!");
34
35          // Criar e treinar o classificador NaiveBayes
36          System.out.println("Treinando o modelo NaiveBayes...");
37          NaiveBayes classifier = new NaiveBayes();
38          classifier.buildClassifier(filteredData);
39          System.out.println("Modelo treinado com sucesso!");
40
41          // Ler nova frase do console
42          Scanner scanner = new Scanner(System.in);
43          System.out.println("Digite uma frase para classificar:");
44          String newText = scanner.nextLine();
45          scanner.close();
46
47          System.out.println("Classificando o novo exemplo de texto: " + newText);
48
49          // Criar uma instância para o novo texto
50          Instances emptyDataset = new Instances(dataset, 0); // Criar um dataset vazio com
51          emptyDataset.setClassIndex(dataset.classIndex()); // Definir o atributo de classe
```

Output - Java_argumento_generative_ai (run)

```
run:
Carregando o conjunto de dados...
Conjunto de dados carregado com sucesso!
Definindo o atributo que será classificado...
Atributo de classe definido: sentiment
Convertendo atributos de texto em vetores de palavras...
Atributos de texto convertidos com sucesso!
Treinando o modelo NaiveBayes...
Modelo treinado com sucesso!
Digite uma frase para classificar:
adorei o produto
Classificando o novo exemplo de texto: adorei o produto
Classificação prevista: positive
BUILD SUCCESSFUL (total time: 5 seconds)
```

Este documento descreve a implementação de um modelo de análise de sentimento utilizando o algoritmo Naive Bayes com WEKA em Java.
O modelo permite que o usuário classifique novas frases em sentimentos, como positivo ou negativo, com base em um conjunto de dados previamente rotulado.

O sistema é simples e extensível.

Futuramente, você pode aprimorar o código ao incluir mais opções de algoritmos de aprendizado de máquina ou integrar com diferentes conjuntos de dados para análise de sentimento.

Link do Projeto :
https://github.com/perucello/Java_Args_Generative_AI

Abraços,
EducaCiência FastCode