Tema practica

Taga Stefan Razvan - 3A2

January 10, 2024

Algorithm 1 Soluționarea Problemei de Clasificare a Fișierelor Text

Pregătirea Datelor:
for fiecare subdir in directorul "lemn" **do**
    for fiecare fișier cu extensia ".txt" în subdir **do**
        citeste conținutul fișierului
        adaugă conținutul la lista de date (data)
        adaugă subdir la lista de etichete (labels)
    end for
end for
Extragerea de Caracteristici:
folosește un vectorizator pentru a extrage caracteristici din date
Implementarea Algoritmului de Învățare Automată:
împarte datele în seturi de antrenare și testare
inițializează un clasificator Naive Bayes (MultinomialNB)
antrenează clasificatorul pe setul de antrenare
Antrenarea Modelului:
folosește setul de testare pentru a face predicții
Evaluarea Modelului:
calculează acuratețea modelului
Predicții:
folosește modelul antrenat pentru a face predicții pentru noi fișiere
Analiza și Interpretare:
afișează raportul de clasificare

Vom utiliza un clasificator Naive Bayes, care este potrivit pentru problemele de clasificare a textelor.

```
import os
from sklearn.feature_extraction.text import
    CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,
    classification_report
```

```
6
7  # Preg tirea Datelor
8  data = []
9  labels = []
10
11 directory = "lemn"
12
13 for subdir in os.listdir(directory):
14     if os.path.isdir(os.path.join(directory, subdir)):
15         for filename in
               os.listdir(os.path.join(directory,
               subdir)):
16             if filename.endswith(".txt"):
17                 with open(os.path.join(directory,
                       subdir, filename), "r") as file:
18                     content = file.read()
19                     data.append(content)
20                     labels.append(subdir)
21
22 # Extragerea de Caracteristici
23 vectorizer = CountVectorizer()
24 X = vectorizer.fit_transform(data)
25
26 # Implementarea Algoritmului de    nvare    Automat
27 X_train, X_test, y_train, y_test =
       train_test_split(X, labels, test_size=0.2,
       random_state=42)
28
29 classifier = MultinomialNB()
30 classifier.fit(X_train, y_train)
31
32 # Antrenarea Modelului
33 y_pred = classifier.predict(X_test)
34
35 # Evaluarea Modelului
36 accuracy = accuracy_score(y_test, y_pred)
37 print(f"Accuracy: {accuracy:.2f}")
38
39 # Analiza  i  Interpretare
40 print("\nClassification Report:")
41 print(classification_report(y_test, y_pred))
```

Listing 1: Implementarea algoritmului în Python

Justificarea Alegerii Clasificatorului Naive Bayes

1. **Text Classification:** Problema pare a fi una de clasificare a textului, deoarece dorim să clasificăm fișierele text în funcție de numele lor.

2. **Simplicitate:** Algoritmii Naive Bayes sunt cunoscuți pentru simplitatea lor și funcționează bine în multe probleme de clasificare a textului. Aceștia sunt eficienți și nu necesită multe eforturi de ajustare a parametrilor.

3. **Număr de Caracteristici Limitat:** În acest exemplu, am ales să extragem doar numărul de apariții ale cuvintelor (termenilor) din fișierele text, ceea ce este potrivit pentru un clasificator Naive Bayes.

4. **Scalabilitate:** Naive Bayes este scalabil și poate funcționa bine și cu seturi de date mai mari, ceea ce îl face potrivit pentru probleme de dimensiuni moderate.

5. **Bun Început:** Naive Bayes este adesea considerat un "bun început" pentru problemele de clasificare a textului, iar rezultatele pot fi satisfăcătoare, mai ales dacă datele îndeplinesc asumpțiunile naive ale algoritmului.