Documentatie proiect inteligenta artificiala

Buturuga George-Alexandru

Grupa 364

Universitatea din Bucuresti

1. **Linear Support Vector Classification.**

Similar cu SVC cu parametrul kermel setat ca ‘linear’ . Acesta are o flexibilitate mai mare in alegerea penalitatilor si a functilor de pierdere si de asemenea se pliaza mai bine pe un numar mare de date.

Am implementat codul folosind urmatorii parametrii, regasiti si in urmatoarea secventa de cod:

C = 0.1 - Parametru de regularizare.

Max\_Iter = 1000 - Numărul maxim de iterații care trebuie rulate.

model = svm.LinearSVC(C=0.1, max\_iter=1000)

In urma rularii, programul a optinut o precizie de aproximativ 64%.

Timpul de executie este de aproximativ 2 minute.

1. **Naive Bayes classifier for multinomial models.**

Clasificatorul multinomial Naïve Bayes este potrivit pentru clasificarea cu caracteristici discrete. Distributia miltinomiala necesita in mod normal numarari intregi de caracteristici.

Alpha = 0.5 - parametru de netezire

Fit\_prior = True - Dacă să înveți sau nu probabilitățile anterioare ale clasei. Dacă este fals, se va folosi un prealabil uniform.

Max\_Iter = 1000 - Numărul maxim de iterații care trebuie rulate.

model = MultinomialNB(alpha=0.5,fit\_prior=True)

In urma rularii, programul a optinut o precizie de aproximativ 58%.

Timpul de executie este de aproximativ 2 minute.

**Desfasurare proiect**

**Importuri necesare**

import os  
import re  
import numpy as np  
import nltk  
from collections import Counter  
from sklearn import svm  
from sklearn.naive\_bayes import MultinomialNB  
nltk.download('punkt')  
import pandas as pd

## Citirea datelor

data\_path = '.'  
train\_data\_df = pd.read\_csv(os.path.join(data\_path, 'train\_data.csv'))  
test\_data\_df = pd.read\_csv(os.path.join(data\_path, 'test\_data.csv'))

## Codificarea etichetelor din string in int

etichete\_unice = train\_data\_df['label'].unique()  
label2id = {}  
id2label = {}  
for idx, eticheta in enumerate(etichete\_unice):  
 label2id[eticheta] = idx  
 id2label[idx] = eticheta

## Pre-procesarea datelor

def proceseaza(text):  
 text = re.sub("[-.,;:!?\"\'\/()\_\*=`]", "", text)  
 text = text.lower()  
 text = text.replace('\n', ' ').strip().lower()  
 text\_in\_cuvinte = text.split(' ')  
 # text\_in\_cuvinte = word\_tokenize(text)  
 list = []  
 for word in text\_in\_cuvinte:  
 if len(word) > 3:  
 list.append(word)  
 return list

data = train\_data\_df['text'].apply(lambda text: proceseaza(text))  
data2 = test\_data\_df['text'].apply(lambda text: proceseaza(text))

## Bag of Words

### Frecventa cuvintelor din setul de antrenare

def count\_most\_common(how\_many, texte\_preprocesate):  
 counter = Counter()  
 for text in texte\_preprocesate:  
 counter.update(text)  
 cuvinte\_caracteristice = []  
 for cuvant, frecventa in counter.most\_common(how\_many):  
 if cuvant.strip():  
 cuvinte\_caracteristice.append(cuvant)  
 return cuvinte\_caracteristice

### Reprezentarea datelor sub forma vectoriala

def build\_id\_word\_dicts(cuvinte\_caracteristice):  
 word2id = {}  
 id2word = {}  
 for idx, cuv in enumerate(cuvinte\_caracteristice):  
 word2id[cuv] = idx  
 id2word[idx] = cuv  
  
 return word2id, id2word

### Frecventa cuvintelor

def featurize(text\_preprocesat, id2word):  
 ctr = Counter(text\_preprocesat)  
 features = np.zeros(len(id2word))  
 for idx in range(0, len(features)):  
 cuvant = id2word[idx]  
 features[idx] = ctr[cuvant]  
 return features

### Frecventa cuvintelor sub forma de matrice

def featurize\_multi(texte, id2word):  
 all\_features = []  
 for text in texte:  
 all\_features.append(featurize(text, id2word))  
 return np.array(all\_features)

## Scrierea datelor in table

with open('submission.csv', 'w') as f:  
 f.write('id,label\n')  
 for id, label in enumerate(model.predict(X\_test)):  
  
 if id == 13860:  
 break  
 else:  
 f.write(f'{id + 1},{label}\n')  
  
print("gata")