Romanian sub-dialect identification

Machine Learning project

Știrbu Denisa-Elena, grupa 244

Tema proiectului vizează crearea unui program Machine Learning ce se dorește a antrena un model pe tweets, urmând ca acesta să deosebească dialectele moldovenești (clasificate cu 0), de cele românești (clasificate cu 1).

Așadar, am încercat sa organizez proiectul și etapele de lucru în 5 pași simpli pentru a oferi un cod ușor de citit și de înteles, astfel:

1. Import
2. Natural Language Processing
3. Feature extraction
4. Classifier
5. Evaluation of accuracy and precision

1.Import

Pentru început am importat pachetele de care am nevoie: numpy (pentru array-uri), csv (pentru scrierea fișierului în formatul dorit), TfidfVectorizer (pentru feature extraction), f1_score, confusion_matrix, accuracy_score, classification_report (pentru evaluare), MultinomialNB (model).

```
#importarea pachetelor de care am nevoie
import numpy as np
import csv
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
```

2. Natural Language Processing

În privința citirii datelor, am ales să fac acest lucru cu open și să apelez metoda readlines(). De asemenea, ar fi de menționat faptul că am concatenat datele de train și datele de validation, întrucât am dorit sa obțin o acuratețe cât mai mare și să antrenez modelul pe cât mai multe date. Pentru partea de samples aveam nevoie să pun doar partea de text într-o variabilă, parte de text ce aparținea fiecărei propoziții, așa că am despărțit id-ul de text cu funcția de split cu separatorul "\t", totul fiind într-un for și făcându-se append la fiecare nouă propoziție. După ce am folosit

același procedeu atât pentru partea de samples din train, cât și pentru cea din validation, am concatenat cei doi vectori.

```
#procesare
file = open("C:/Users/Denisa/Desktop/ml-2020-unibuc-3 (1)/train_samples.txt",  encoding="utf8")
train1_samples = file.readlines()
file = open("C:/Users/Denisa/Desktop/ml-2020-unibuc-3 (1)/validation_samples.txt",  encoding="utf8")
train2_samples = file.readlines()
```

```
samples1 = []
for x in train1_samples:
    samples1.append(x.split("\t")[1])

samples2 = []
for y in train2_samples:
    samples2.append(y.split("\t")[1])
```

Așadar, am folosit același mecanism și pentru partea de labels din train și validation, separând id-ul de cod (codul fiind 1 sau 0) și concatenând cele doua variabile la final.

```
#procesare
file = open("C:/Users/Denisa/Desktop/ml-2020-unibuc-3 (1)/train_labels.txt",  encoding="utf8")
train1_labels = file.readlines()
file = open("C:/Users/Denisa/Desktop/ml-2020-unibuc-3 (1)/validation_labels.txt",  encoding="utf8")
train2_labels = file.readlines()
```

```
labels1 = []
for x in train1_labels:
    labels1.append(x.split()[1])
labels2 = []
for y in train2_labels:
    labels2.append(y.split()[1])
```

```
data_samples = []
data_samples = samples1 + samples2
data_labels = []
data_labels = labels1 + labels2
```

3. Feature extraction

Pentru acest punct al proiectului am ales să folosesc TFIDF Vectorizer (term frequency–inverse document frequency), transformând textul în vectori caracteristici ce vor fi folosiți ca input pentru modelul de antrenare. Am făcut această alegere întrucat TFIDF Vectorizer este echivalentul lui CountVectorizer, urmat de TfidfTransformer, oferind o performanță mult mai bună comparativ cu CountVectorizer, procedeu încercat și de către mine.

4. Classifier

Cele mai multe modificări ale proiectului au fost în această parte, întrucat am încercat diverse modele, precum: KNeighborsClassifier, RandomForestClassifier, AdaBoostClassifier, SVC, SGDClassifier, LogisticRegression, obținând rezultate destul de diferite, însă cel mai bun rezultat pe Kaggle l-am obținut cu MultinomialNB.

Mai jos sunt câteva exemple de evaluari pe cateva modele anterior menționate:

```
In [460]: #evaluare
          f1_score(test_data_labels,prediction, average = 'macro')
Out[460]: 0.8401358496945046
```

SGDClassifier

```
In [20]: #evaluare
         f1_score(test_data_labels,prediction, average = 'macro')
Out[20]: 0.7652783323817394
```

KNeighborsClassifier

```
In [107]: #evaluare
          f1_score(test_data_labels,prediction, average = 'macro')
Out[107]: 0.6394935760922984
```

RandomForestClassifier

```
In [168]: #evaluare
          f1_score(test_data_labels,prediction, average = 'macro')
Out[168]: 0.6424256636431203
```

SVCClassifier

```
In [217]: #evaluare
          f1_score(testing_labels,prediction, average = 'macro')
Out[217]: 0.8225577914651206
```

MultinomialNB

Pentru această parte, a fost nevoie de puțin mai multă documentare, așa că am accesat surse precum:

- https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
- https://www.youtube.com/watch?v=ppXFoltcX7A
- https://www.youtube.com/watch?v=pXdum128xww
- https://www.youtube.com/watch?v=DCZ3tsQIoGU
- https://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html

Am citit și procesat datele din fisierele validation_samples și validation_labels, urmând să ne folosim de funcțiile append și split pentru a separa partea de text și labels de id, în același fel în care am procedat și cu celelalte fișiere și aplicăm funcția de transform.

5.Evaluation of accuracy and precision

Pentru a testa acuratețea modelului propus, am optat pentru f1 score, fiind interpretat ca o medie ponderată a preciziei și a recall-ului, unde scorul cel mai bun este 1 și cel mai slab atingând punctul 0. De asemenea, m-am folosit de matricea de confuzie, scorul de acurateț̦e și raportul de clasificare. De exemplu, pentru modelul final, MultinomialNB situația este prezentată mai jos:

```
7]: #evaluare
    f1_score(testing_labels,prediction, average = 'macro')

7]: 0.8225577914651206

8]: #evaluare
    print('Confusion Matrix :')
    results = confusion_matrix(testing_labels,prediction)
    print(results)
    print('Accuracy Score :',accuracy_score(testing_labels,prediction))
    print ('Report : ')
    print(classification_report(testing_labels,prediction))

    Confusion Matrix :
    [[1001  300]
     [ 169 1186]]
    Accuracy Score : 0.8234186746987951
    Report :
                  precision    recall  f1-score   support

               0       0.86      0.77      0.81      1301
               1       0.80      0.88      0.83      1355

        accuracy                           0.82      2656
       macro avg       0.83      0.82      0.82      2656
    weighted avg       0.83      0.82      0.82      2656
```

În final, am procesat datele din fișierul test_samples, la fel cum am făcut și cu celelalte date citite din fișiere, după același procedeu, și prelucrăm datele pentru a obține submit_prediction, de care ne vom folosi în afisare (facem transformarea și aplicăm modelul pentru predicție). De menționat ar fi și faptul că ne pastram id-urile separat, pentru a face afișarea în formatul cerut.

```
:  file = open("C:/Users/Denisa/Desktop/ml-2020-unibuc-3 (1)/test_samples.txt",  encoding="utf8")
   samples = file.readlines()

:  submit_samples = []
   id = []
   for x in samples:
       submit_samples.append(x.split('\t')[1])
       id.append(x.split()[0])

:  submit_features = vectorizer.transform(submit_samples)
   submit_prediction = model.predict(submit_features)
```

Afișarea se face în formatul cerut, csv:

```
#scriere in formatul cerul, CSV
with open ('Denisa_ml3.csv', 'w', newline = '') as f:
    filewriter=csv.writer(f)
    filewriter.writerow(['id','label'])
    for i in range(len(submit_prediction)):
        filewriter.writerow([id[i],submit_prediction[i]])
```