Filip Ioana Alexandra

Grupa 243

Documentație Proiect - Romanian sub-dialect identification

1. Citirea datelor din fișiere

La citire am folosit importarea datelor prin genfromtxt, funcție din NumPy care preia datele linie cu linie, le împarte după delimitatorul tab (\t) în perechi și returnează un ndarray de aceste perechi. Acestea reprezintă fie o pereche cheie-text (pentru fișierele de tip samples), fie o pereche cheie-dialect (pentru fișierele de tip labels).
Am constatat că nu preluam întregul text corespunzător fiecarei linii, deoarece odată ce se întâlnea caracterul #, tot ce urma după el era considerat comentariu, așa că am adaugat la citire și argumentul comments = None.

2. Preluarea datelor

Din perechile de date create în urma citirii iau următoarele date:

- de test (test_samples_keys = cheile și test_samples_texts = textele);
- de antrenare (train_samples_texts = textele și train_labels_dialects = dialectele);
- de validare (validation_samples_texts = textele și validation_labels_dialects = dialectele).

3. **Antrenarea și validarea** – aici am antrenat clasificatorul pe datele de antrenare și am făcut predicțiile pe datele de validare pentru a vedea eficiența sa de a prezice corect dialectele unui set de texte.

CountVectorizer convertește colecția de texte într-o matrice bazată pe numărul de aparițiii al unui cuvânt în fiecare text. Implicit acesta transformă toate literele în litere mici, împarte textul în cuvinte, ignorând cuvintele de lungime 1, iar numărul de features va fi egal cu numărul cuvintelor din vocabular (numărul de cuvinte întâlnite cel puțin o dată în cel puțin unul dintre texte). Aplic pe următoarele:

- train_data_matrix = CountVectorizer.fit_transform(train_samples_texts) pentru a învăța vocabularul și a returna matricea respectivă acestui set de date. Ea va fi de mărime (număr linii = mărime train_sample_texts, număr coloane = număr de features/cuvinte din vocabular).
- validation_data_matrix = CountVectorizer.transform(validation_samples_texts) pentru a returna matricea respectivă acestui set de date, iar aceasta va avea mărimea (număr linii = mărime validation_samples_texts, număr coloane = număr de features/cuvinte din vocabular).

Apoi am folosit pentru clasificare **MultinomialNB**. Acesta este un clasificator a cărui distribuție multinomială necesită date de tip întreg ce reprezintă frecvența fiecărui feature (cuvânt din vocabular), precum cele create de CountVectorizer.

- MultiomialNB.fit(train_data_matrix, train_labels_dialects) face fit pe un clasificator Naïve Bayes astfel antrenându-l pe setul de date de antrenare.
- MultinomialNB.predict(validation_data_matrix) face efectiv clasificarea pe setul de date de validare, returnând un set de predicții cu ajutorul cărora ulterior să putem vedea eficiența clasificatorului.

Pentru a vedea eficiența clasificării realizate am calculat:

- **Acuratețea**: accuracy_score, ce a rezultat a fi aproximativ 0,66;
- **Scorul F1**: f1_score aproximativ egal cu 0,7;
- **Raportul clasificării**: classification_report ce reprezintă principalele măsurători specifice clasificărilor (pe lângă accuracy și f1 conține și recall, support):

	precision	recall	f1-score	support
0	0.69	0.54	0.61	1301
1	0.64	0.77	0.70	1355
accuracy			0.66	2656
macro avg	0.67	0.66	0.65	2656
weighted avg	0.67	0.66	0.65	2656

- **Matricea de confuzie**: confusion_matrix, unde un element $x_{i,j}$ este egal cu numărul de texte știute ca având dialectul i și prezise ca având dialectul j. Din aceasta reiese că din dialectul 0 (moldovensec) am prezis: 706 bine și 595 prost, iar din dialectul 1 (românesc) am prezis: 1045 bine și 310 prost:

```
[[ 706  595]
[ 310 1045]]
```

4. **Testarea** – am aplicat clasificatorul antrenat si "validat" anterior, pe datele de test.
   - test_data_matrix = CountVectorizer.transform(test_samples_texts) pentru a returna matricea respectivă acestui set de date, ce va avea marimea (mărime test_samples_texts, număr de features/cuvinte din vocabular creat anterior la punctul 3.).
   - MultinomialNB.predict(test_data_matrix) face efectiv clasificarea pe setul de date de testare, creând un set de predicții.

5. **Scrierea predicțiilor** – creez un fișier "predictions.txt" în care scriu perechile de chei din test_samples_keys și predicția de dialect corespunzătoare acesteia, din setul de predicții. Deoarece nu am modificat ordinea textelor din test_samples_texts, indicele cheii corespunde cu indicele dialectului (predicției).