Furculesteanu Bianca

Grupa 241

Proiect IA – Identificarea subdialectelor romanesti

Cerinta proiectului:

In acest proiect se doreste antrenarea unui model care sa clasifice daca anumite tweeturi se incadreaza in familia cuvintelor specific romanesti sau se pot clasifica mai de graba ca facand parte din familia cuvintelor specifice dialectului moldovenesc, avand ca date de antrenare fragmente din stiri ce apartin ambelor categori.

Abordarea folosita pentru invatarea automata:

Am abordat proiectul folosindu-ma de un model bag-of-words(BoW), in care am extras caracteristicile textului in urmatorul mod:

-Am creat un dictionar(dict_tr) in care ofer fiecarui cuvant din training_samples un id(in dictionar nu vor aparea duplicate). (tododata a mai fost nevoie sa mai fac un vector care sa retina fiecare cuvant in parte pentru a ma putea asigura ca atunci cand construiesc matricea de features pentru datele din test_samples, nu exista cumva acolo un cuvant care sa nu fie in dictionar, iar daca acest cuvant lipsa exista, am optat pentru ignorarea)

```
dict_tr = {}#dictionarul in care am pastrat id-ul pentru fiecare cuvant
dict_v = []# vector ce contine fircare cuvant ce se afla in dictionar
i = 0       #id-ul care va fi atribuit cuvantului
for sent in train_samples:#parcurgerea propozitiilor din train samples
  aux = sent.split()   # spargerea propozitiilor in cuvinte
  for w in aux:        # parcurgerea fiecarui cuvant
    if w not in dict_tr:     # daca cuvantul nu se afla in dictionar i
      dict_tr[w] = i         # va atribui un id
      i = i + 1              # incrementarea id-ului
      dict_v.append(w)       # se adauga cuvantul in vectorul de cuvinte
```

-Am facut doua matrice de features(de marimea: len(sample) x len(dict_tr)), una pentru propozitiile din train_samples.txt(result_tr) si una pentru cele din test_samples.txt(result_te) in care prima linie reprezinta fiecare propozitie din train_samples.txt, respectiv test_samples.txt, iar coloanele reprezinta fiecare cuvant din dictionar. (spre exemplu

result_te[5][dict_tr[‘2@^#’] = 1, inseamna ca acel cuvant, 2@^# apare o data in propozitia respectiva)

```
i = 0                                   # indicele propozitiei
for sent in train_samples:              # parcurgem
    aux = sent.split()              # impartim propozitiile in cuvinte
    for w in aux:
      result_tr[i][dict_tr[w]] += 1     # se numara de cate ori exista
un cuvant intr-o propozitie
    i = i + 1                       # se trece la urmatoarea propozitie
```

```
[[1. 1. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
```

-Am folosit StandardScaler() pentru a standardiza datele

```
scaler = preprocessing.StandardScaler()
scaler.fit(result_tr)
result_tr = scaler.transform(result_tr)
result_te = scaler.transform(result_te)
```

-Am utilizat LogisticRegression(aka logit, MaxEnt) pentru a face predictiile, utilizand parametrii default(penalty = ‘l2’, dual = False, tol = 0.001, C = 1.0, fir_intercept = True, intercept_scaling = 1, class_weight = None, random_state = None, solver = ‘lbfgs’, max_iter = 100, multi_class = ‘auto’, verbose = 0, warm_start = False, n_jobs = None, l1_ratio = None), implementat deja in python. Acesta pune in evidenta problema gasirii unei functii de forma:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

care interpreteaza cem mai bine un set de exemple:

$$S = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x}_\ell, y_\ell)\}$$

(In cazul algoritmului scris de mine, ce se afla in result_tr cu label-urile aferente)

```
clf = LogisticRegression()
clf.fit(result_tr, tr_l)
prediction = clf.predict(result_te)
```

Totodata pentru a doua submisie m-am folosit de Naïve Bayes cu ajutorul functiilor deja implementate in Python.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

```
model = GaussianNB()

model.fit(features,label)
```

Matricea de confuzie :

	Prezis Ro	Prezis MD
Ro	795.	456.
MD	506.	899.