Automatic Misogyny Identification

Descriere

Programele în python presupun pregătirea unui clasificator pentru a prezice dacă unele postări de pe Twitter scrise în limba italiană au un conţinut misogin. Datele constau în 6000 de postări, 5000 fiind de antrenare şi 1000 de testare. Pentru rezolvarea problemei s-au folosit 2 clasificatori, KNN (k-nearest neighbors) şi SVM (Support vector machines).

Realizare clasificatori

Programele în python prezintă o parte comună, cea de prelucrare a datelor, fiind diferenţiate doar de clasificator. Pentru simplificarea textelor folosite în procesarea limbajului şi în regasirea informaţiei se utilizează o reprezentare după modelul Bag-of-Words.

Pentru citirea csv-urilor se foloseşte pandas:

```
train_df = pd.read_csv('train.csv')
test_df  = pd.read_csv('test.csv')
```

Csv-urile sunt sub forma:

id	text	label
1	@KassemAmin4 @Laylasexgdr Fatti trovare te lo do volentieri e ti sborro in bocca,	1
2	@meb Tu dovresti ricominciare dai semafori tanto il vitalizio ti resterà in gola,	0
	…	

(1 – e misogin, 0 – nu e misogin)

Se foloseşte nltk, mai exact Tweet Tokenizer pentru a împarţi textul în cuvinte, având parametrii:

```
strip_handles  = True        - eliminarea @x şi a emoticoanelor
preserve_case = False        - trecerea literelor mari în litere mici în cadrul cuvintelor
```

Textele devin astfel un vector de cuvinte ignorand #-urile, @x (userii) si adresele:

```
['fatti', 'trovare', 'te', 'lo', 'do', 'volentieri', 'e', 'ti', 'sborro', 'in', 'bocca']
['tu', 'dovresti', 'ricominciare', 'dai', 'semafori', 'a','tanto', 'il', 'vitalizio', 'ti', 'resterà', 'in', 'gola']
```

Mai departe, pentru crearea unui “vector de frecvenţă” se utilizează Counter din collections:

```
Counter({'che': 2927, ',': 2429, 'di': 2142, 'e': 1990, '?': 1944, 'puttana': 1928, 'porca': 1894…
```

Se doreşte însă obţinerea unui dicţionar cu primele n cuvinte ca şi frecvenţă pentru a putea reprezenta un text sub formă de vector:

```
{'che': 0, ',': 1, 'di': 2, 'e': 3, '?': 4, 'puttana': 5, 'porca': 6, '.': 7, 'la': 8, '!': 9, 'non': 10 …
```

Funcţia text_to_bow realizează pentru fiecare text trecerea într-un vector de frecvenţă:

	che	,	di	e	?	puttana	porca	.
text	0	1	2	0	0	2	1	0

Prin normalizarea l2 se reduce diferenţele realizate de lungimile diferite ale textelor:

$$norm = \sqrt{0^2+1^2+2^2+0^2+0^2+2^2+1^2+0^2} = \sqrt{10}$$

	che	,	di	e	?	puttana	porca	.
text	$\frac{0}{\sqrt{10}}$	$\frac{1}{\sqrt{10}}$	$\frac{2}{\sqrt{10}}$	$\frac{0}{\sqrt{10}}$	$\frac{0}{\sqrt{10}}$	$\frac{2}{\sqrt{10}}$	$\frac{1}{\sqrt{10}}$	$\frac{0}{\sqrt{10}}$

Funcția corpus_to_bow utilizează text_to_bow în realizarea unui vector de vectori atat pentru train_df cat și pentru test_df cu scopul de a fi utlizate de catre clasificatori:

```
[[0.17407766 0.17407766 0.17407766 ... 0. 0. 0. ]
 [0.60302269 0. 0. 0. ... 0. 0. 0. ]
 [0.38490018 0. 0.38490018 … 0.19245009 0. 0. ]
  ...
```

Nota: Ambele fișiere python (submisii) prezintă verificarea pe datele de antrenare cu kfold și afișare matricei de confuzie într-un fragment de cod comentat.

Clasificatorul kNN

Acest clasificator:

- utilizează default metrica l2 si primele 70 de cuvinte
- nu are parametrii
- hiperparametrul este k, reprezentând numarul de vecini (ales dupa performanța pe teste)
- verificarea pe datele de train folosind kfold și printarea acurateții, fscorului mediu si a matricei de confuzie durează aproximativ 25s
- performanța pe kaggle este de 75.225%

Acuratețea: 0.8324 Fscore: 0.8180 Matricea de confuzie: [[1889 448]
[390 2273]]

```
from sklearn.neighbors import KneighborsClassifier          - importarea clasificatorului
clf = KNeighborsClassifier(n_neighbors = 7)                 - inițializarea lui
clf.fit(data, labels)                                       - antrenare pe datele de train
preds = clf.predict(test_data)                              - obținerea predicțiilor
write_prediction('352_TroianStefan_submisie1.csv', preds)   - scrierea acestora
```

Clasificatorul, pentru fiecare vector din test_data caută cei mai apropiați k vectori din data (diferența element cu element dupa l2) și preia label-ul predominant dintre 0 și 1.

Clasificatorul SVM

Acest clasificator:

- utilizează default un kernel linear și primele 70 de cuvinte
- are ca hiperparametru pe c, reprezentând parametrul de penalitate, care e default 1.0
- verificarea pe datele de train folosind kfold și printarea acurateții, fscorului mediu și a matricei de confuzie durează aproximativ 30s
- performanța pe kaggle este de 74.105%

Acuratețea: 0.8606 Fscore:0.8558 Matricea de confuzie: [[2071 266]
[431 2232]]

```
from sklearn import svm                                     - importarea clasificatorului
clf = svm.SVC()                                             - inițializarea lui
clf.fit(data, labels)                                       - antrenare pe datele de train
preds = clf.predict(test_data)                              - obținerea predicțiilor
write_prediction('352_TroianStefan_submisie2.csv', preds)   - scrierea acestora
```

Prin parametrii default și datele preluate de clasificator, data și labels, acesta realizează o dreaptă (“granița”) între cele două clase. În momentul în care se fac predicțiile se aleg valorile 0 sau 1 în funcție de acea dreaptă.