Petre Andrei, 342 C3

Classifier KNN & Naive Bayes

Rezultate

Dupa implementarea celor doi algoritmi am compus urmatorul tabel:

	Naive	5-NN	7-NN	10-NN
Training	700 SPAM 1000 SPAM_2 700 HAM 1000 HAM_2 170 HAM_HARD	700 SPAM 1000 SPAM_2 700 HAM 1000 HAM_2 170 HAM_HARD	700 SPAM 1000 SPAM_2 700 HAM 1000 HAM_2 170 HAM_HARD	700 SPAM 1000 SPAM_2 700 HAM 1000 HAM_2 170 HAM_HARD
SPAM	31/300 WRONG 21/300 WRONG	14/300 WRONG 45/300 WRONG	12/300 WRONG 41/300 WRONG	6/300 WRONG 21/300 WRONG
SPAM_2	54/397 WRONG 35/397 WRONG	62/397 WRONG 102/397 WRONG	41/397 WRONG 97/397 WRONG	10/397 WRONG 53/397 WRONG
HAM	1/300 WRONG 1/300 WRONG	36/300 WRONG 20/300 WRONG	52/300 WRONG 23/300 WRONG	149/300 WRONG 46/300 WRONG
HAM_2	0/397 WRONG 0/397 WRONG	70/397 WRONG 33/397 WRONG	80/397 WRONG 46/397 WRONG	135/397 WRONG 59/397 WRONG
HAM_HARD	29/80 WRONG 20/80 WRONG	30/80 WRONG 11/80 WRONG	40/80 WRONG 19/80 WRONG	62/80 WRONG 25/80 WRONG

Pentru Rotten Tomatoes nu am putut face un tabel pentru ca nu cunosteam clasele adevarate ca sa estimez cat e bine. Am gasit insa cateva repere care indica faptul ca algoritmul functioneaza corect, iar tag-urile se pot vedea in fisierele de iesire pastrate (**rotten_naive** si **rotten_knn_5**) in root.
Functioneaza corect pentru ca:

- “great” e clasificat ca 4 in KNN si ca 3 in Naive (el in train e 4)
- “disservice” e clasificat ca 1 in KNN si in Naive (el in train e 1)
- impreuna [“great”, “disservice”] sunt clasificate ca 1 de KNN si de Naive

Naive gaseste ca probabiltatea de a fi in clasa 3 este mai mare pentru ca sunt mai multe aparitii ale lui “great” in clasa 3, iar cu cat numitorul (cu laplace smoothing cu tot) e mai mare, numaratorul devine mai semnificativ => iese 3.

KNN insa va tine cont ca a gasit “exact” great cu clasa 4 in train si deci va obtine tot 4. Tot KNN va obtine 1 pentru cele doua cuvinte impreuna pentru ca “great” apare de mai multe ori, deci are o pondere redusa comparativ cu “disservice” care apare de maxim 10 ori in 150k fraze.

Implementare

Naive Bayes gaseste clasa cu probabiltatea cea mai mare, unde probabilitatea ca o clasa sa fie potrivita pentru un input e calculata cu formula lui Bayes si se simplifica prin faptul ca cuvintele sunt independente intre ele (paste din comentarii cod):

```
=> p(class|data) = p(data|class) * p(class)
                 = p(w1, w2, ... wn | class) * p(class),
   where the data (e.g. a mail) can be composed of words.
                 = p(w1 | class) * ... p(wn | class) * p(class),
   and the above step is the _naive_ assumption.
```

KNN gaseste cele mai apropiate k fraze si ia clasa dominanta dintre acei vecini gasiti.

Totodata in implementare am folosit pachetul “nltk” din python care ofera facilitati ca stemming (am folosit SnowballStemmer), si word_tokenize. Pentru preprocesarea cuvintelor am urmat pasii in ordine:

- am tinut doar litere ascii in cuvinte (eliminat caractere unicode, numere, punctuatie etc.)
- le-am transformat in litere mici
- am scos cuvinte comune din engleza
- am scos cuvinte cu lungime <= 2
- am stemm-uit cuvintele
- am scos cuvintele duplicate

Cuvintele obtinute le-am retinut pentru fiecare mail (sau fraza din rotten tomatoes) in parte.