Information Retrieval – Assignment 12

# IR13A.py

## Code

# IR13A.py CS5154/6054 cheng 2022  
# use the first and the last 1000 lines of bible.txt as two classes  
# find top terms according to mutual information  
# Usage: python IR13A.py  
  
import numpy as np  
import sklearn.feature\_selection as fs  
from sklearn.feature\_extraction.text import CountVectorizer  
  
f = open("bible.txt", "r")  
docs = f.readlines()  
f.close()  
N =len(docs)  
trainX = np.concatenate([docs[0:1000], docs[N-1000:N]])  
y = np.concatenate([np.zeros(1000, dtype=np.int16), np.ones(1000, dtype=np.int16)])  
  
cv = CountVectorizer(binary=True, max\_df=0.4, min\_df=4)  
X = cv.fit\_transform(trainX).toarray()  
print(X.shape)  
voc = np.array(cv.get\_feature\_names())  
chi2 = fs.chi2(X, y)  
sorted = np.argsort(chi2)[::-1][0]  
for i in range(10):  
 index = sorted[i]  
 print(voc[index], chi2[0][index])  
  
kbest = fs.SelectKBest(fs.chi2)  
kbest.fit(X, y)  
support = np.array(kbest.get\_support())  
print(voc[support])

## Results

Text

Description automatically generated

# IR13C.py

## Code

# IR13C.py CS5154/6054 cheng 2022  
# BernoulliNB classification  
# with feature selection  
# Usage: python IR13C.py  
  
import numpy as np  
import sklearn.feature\_selection as fs  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.naive\_bayes import BernoulliNB  
from sklearn.metrics import accuracy\_score  
  
f = open("bible.txt", "r")  
docs = f.readlines()  
f.close()  
N =len(docs)  
trainX = np.concatenate([docs[0:1000], docs[N-1000:N]])  
y = np.concatenate([np.zeros(1000, dtype=np.int16), np.ones(1000, dtype=np.int16)])  
testX = np.concatenate([docs[1000:1100], docs[N-1100:N-1000]])  
testY = np.concatenate([np.zeros(100, dtype=np.int16), np.ones(100, dtype=np.int16)])  
  
cv = CountVectorizer(binary=True, max\_df=0.4, min\_df=4)  
X = cv.fit\_transform(trainX).toarray()  
print(X.shape)  
voc = cv.get\_feature\_names()  
T = cv.transform(testX).toarray()  
  
kbest = fs.SelectKBest(fs.chi2, k=700)  
kbest.fit(X, y)  
X2 = kbest.transform(X)  
T2 = kbest.transform(T)  
  
model = BernoulliNB()  
model.fit(X2, y)  
pred = model.predict(T2)  
print(pred)  
print ('Accuracy Score - ', accuracy\_score(testY, pred))

## Results

With No Changes  
A screenshot of a computer

Description automatically generated with medium confidence

### With chi2

A screenshot of a computer

Description automatically generated with medium confidence

### With Chi 2 and k = 700

A screenshot of a computer

Description automatically generated with medium confidence

As you can see in the above screenshots there is a noticeable increase in the accuracy score on switching to chi2