Information Retrieval – Assignment 14

# IR14A.py

## Code

# IR15A.py CS5154/6054 cheng 2022  
# Comparing classifiers on documents as binary, count, or tfidf vector  
# on two random segments of bible.txt from the first third and last third  
# 100 test documents are at the center of 1000 training documents  
# Only the four in IIR Chapters 13 and 14 are implemented  
# you need to add the ten others imported, too  
# Usage: python IR15A.py  
  
import numpy as np  
import random  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.naive\_bayes import BernoulliNB  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.neighbors import NearestCentroid  
from sklearn.linear\_model import LogisticRegression  
from sklearn.svm import LinearSVC  
from sklearn.svm import SVC  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.tree import ExtraTreeClassifier  
from sklearn.ensemble import ExtraTreesClassifier  
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.linear\_model import Perceptron  
from sklearn.neural\_network import MLPClassifier  
from sklearn.metrics import accuracy\_score  
  
f = open("bible.txt", "r")  
docs = f.readlines()  
f.close()  
  
N =len(docs)  
N1 = N // 3 - 1100  
c0 = random.randrange(N1)  
c1 = N - 1100 - random.randrange(N1)  
print('Random segments at -', c0, c1)  
  
trainX = np.concatenate([docs[c0:c0+500], docs[c0+600:c0+1100],  
docs[c1:c1+500], docs[c1+600:c1+1100]])  
y = np.concatenate([np.zeros(1000, dtype=np.int16), np.ones(1000, dtype=np.int16)])  
testX = np.concatenate([docs[c0+500:c0+600], docs[c1+500:c1+600]])  
testY = np.concatenate([np.zeros(100, dtype=np.int16), np.ones(100, dtype=np.int16)])  
  
# documents as binary vectors  
cv = CountVectorizer(binary=True, max\_df=0.4, min\_df=4)  
X0 = cv.fit\_transform(trainX).toarray()  
T0 = cv.transform(testX).toarray()  
  
# documents as count vectors  
cv = CountVectorizer(max\_df=0.4, min\_df=4)  
X1 = cv.fit\_transform(trainX).toarray()  
T1 = cv.transform(testX).toarray()  
  
# documents as tfidf vectors  
cv = TfidfVectorizer(max\_df=0.4, min\_df=4)  
X2 = cv.fit\_transform(trainX).toarray()  
T2 = cv.transform(testX).toarray()  
  
model = BernoulliNB()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
print ('BernoulliNB -', A0)  
  
model = MultinomialNB()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
print ('MultinomialNB -', A0, A1)  
  
model = KNeighborsClassifier()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('KNN -', A0, A1, A2)  
  
model = NearestCentroid()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Rocchio -', A0, A1, A2)  
  
model = LogisticRegression()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Logistic Regression -', A0, A1, A2)  
  
model = LinearSVC()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('LinearSVC -', A0, A1, A2)  
  
model = SVC()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('SVC -', A0, A1, A2)  
  
model = DecisionTreeClassifier()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Decision Tree Classifier -', A0, A1, A2)  
  
model = ExtraTreeClassifier()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Extra Tree Classifier -', A0, A1, A2)  
  
model = ExtraTreesClassifier()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Extra Trees Classifier -', A0, A1, A2)  
  
model = AdaBoostClassifier()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Ada Boost Classifier -', A0, A1, A2)  
  
model = RandomForestClassifier()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Random Forest Classifier -', A0, A1, A2)  
  
model = Perceptron()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('Perceptron -', A0, A1, A2)  
  
model = MLPClassifier()  
model.fit(X0, y)  
A0 = accuracy\_score(testY, model.predict(T0))  
model.fit(X1, y)  
A1 = accuracy\_score(testY, model.predict(T1))  
model.fit(X2, y)  
A2 = accuracy\_score(testY, model.predict(T2))  
print ('MLP Classifier -', A0, A1, A2)

## Results

### Iteration 1

Text

Description automatically generated

Winning combination = Perceptron with documents as binary vectors

### Iteration 2

Text

Description automatically generated

Winning combination = MultinomialNB with documents as binary vectors

### Iteration 3

Text

Description automatically generated

Winning combination = Linear SVC with documents as tf-idf vectors or SVC with documents as tf-idf vectors

### Iteration 4

Text

Description automatically generated

Winning combination = Extra Trees Classifier with documents as count vectors

### Iteration 5

Text

Description automatically generated

Winning combination = Perceptron with documents tf-idf vectors