CO544 – Data Mining & Machine Learning Assignment

WIMALASIRI KPGP

E/14/403

SEMESTER 6

14/01/2019

Text Classification

Text classification is an example of supervised machine learning and done for many practical purposes as,

- Web pages to subject categories
- Research papers to subject categories
- Understanding audience sentiment from social media
- Detection of spam and non-spam emails
- Auto tagging of customer queries
- Categorization of news articles into defined topics.

Here, according to the text in each documents classification is done. In order to perform this classification to documents those text documents following steps needed to be followed.

1. Dataset preparation
2. Feature processing
3. Model training

Before all the steps above importing libraries has to be done.

```
import re
import nltk
#nltk.download()
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import csv
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from sklearn import metrics
```

In order to prepare the data set for the processing given data in the txt file needed to be converted into standard file. (arff or csv)

And also other preprocessing techniques have to be performed here. (stop words, stemming, removing special characters, replace multiple spaces with single spaces, converting into lowercase letters and etc.)

```
23
24  while i < len(lines):       #doing following operation line by line
25      X = lines[i].split('\t')        #split by the tab (four main attributes '<CLASS> \t <TITLE> \t <DATE> \t <BODY>')
26      Y.append([])        #2D array
27      for j in range(4):
28          Y[i].append(X[j])       #appending all four attributes to each line
29      i+=1
30
31  def preprocess(attribute):
32
33      attribute = re.sub(r"[^a-zA-Z0-9]+", ' ', attribute)    #Remove all the special characters except space
34      attribute = re.sub(r'\s+', ' ', attribute, flags=re.I)  #Substituting multiple spaces with single space
35      attribute = attribute.lower()
36
37      all_words = attribute.split()
38      en_stops = set(stopwords.words('english')) #stopwords for english
39      attribute = ""
40      ps = PorterStemmer()
41
42      for word in all_words:
43          if word not in en_stops:
44              attribute +=  ps.stem(word) +" "
45
46      return attribute
47
```

```
48
49  YProcessed = []
50  i = 0
51
52  while i <len(lines):
53      YProcessed.append([])
54      YProcessed[i].append(Y[i][0])
55
56      title= preprocess(Y[i][1])
57      date= preprocess(Y[i][2])
58      body= preprocess(Y[i][3])
59
60      YProcessed[i].append(title+date+body)
61
62      #print(YProcessed[i])
63      i+=1
64
65
66  with open("YProcessed.csv", "w") as file:
67      writer = csv.writer(file)
68      writer.writerows(YProcessed)
```

By NLTK stop word, most common words in English (such as “the”, “a”, “an”, “in”) is removed as they going to be affective for the final result. By using stemmer suffixes are removed. (eg: house, houses both are treated as the same)

By using re library (regex) removing special characters, replace multiple spaces with single spaces, converting into lowercase letters has been done.

After doing the preprocessing Bag-of-words representation is used to have count of words in the document. “TfidfVectorizer” is imported for this purpose. In this case there are two terms as min_df and max_df. max_df is used for removing terms that appear too frequently. max_df = 0.8 means ignore terms that appear in more than 80% of the documents. min_df is used for removing terms that appear too infrequently. min_df = 0.05 means "ignore terms that appear in less than 5% of the documents. To choose these values classification test had to be done. Therefore, train data set split has performed. By using that min_df and max_df values are adjusted.

```
70
71  df=pd.read_csv('YProcessed.csv',sep=',',names=['class','data'])
72  df_x=df["data"]
73  df_y=df["class"]
74  x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.333, random_state=0)
75
76  cv = TfidfVectorizer(min_df=0.05, max_df=0.8)
77  x_traincv=cv.fit_transform(x_train)
78
```

As MultinomialNB is known to be a popular text classifier it is used here for the classification. Finally “testsetwithoutlabels.txt” has used and predicted the classes.

```
88  text_file = open("testsetwithoutlabels.txt", "r")
89  test_line = text_file.readlines()
90  test=cv.transform(test_line)
91  predict=clf.predict(test)
92
93  #output = open("e14403.txt","w+")
94
95  i=0
96  while i < len(test_line):
97      #f.write("%f" %predict[i])
98      print(predict[i])
99      i+=1
```

APPENDIX

CODE

```
import re
import nltk
#nltk.download()
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import csv
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from sklearn import metrics

f = open("trainset.txt","r")                    #opening the text file for reading
#contents = f.read()
#print(contents)
lines = f.readlines()                  #reading text file into an array

Y = []
i = 0

while i < len(lines):                  #doing following operation line by line
```

```
        X = lines[i].split('\t')                #split by the tab (four main attributes '<CLASS> \t <TITLE> \t
<DATE> \t <BODY>')
        Y.append([])            #2D array
        for j in range(4):
                Y[i].append(X[j])                #appending all four attributes to each line
        i+=1


def preprocess(attribute):

   attribute = re.sub(r"[^a-zA-Z0-9]+", ' ', attribute)        #Remove all the special characters except space
   attribute = re.sub(r'\s+', ' ', attribute, flags=re.I)        #Substituting multiple spaces with single space
   attribute = attribute.lower()

   all_words = attribute.split()
   en_stops = set(stopwords.words('english')) #stopwords for english
   attribute = ""
   ps = PorterStemmer()

   for word in all_words:
      if word not in en_stops:
         attribute +=  ps.stem(word) +" "

   return attribute



YProcessed = []
i = 0

while i <len(lines):
```

```
    YProcessed.append([])
    YProcessed[i].append(Y[i][0])

    title= preprocess(Y[i][1])
    date= preprocess(Y[i][2])
    body= preprocess(Y[i][3])

    YProcessed[i].append(title+date+body)

    #print(YProcessed[i])
    i+=1



with open("YProcessed.csv", "w") as file:
    writer = csv.writer(file)
    writer.writerows(YProcessed)



df=pd.read_csv('YProcessed.csv',sep=',',names=['class','data'])
df_x=df["data"]
df_y=df["class"]
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.333, random_state=0)

cv = TfidfVectorizer(min_df=0.05, max_df=0.8)
x_traincv=cv.fit_transform(x_train)

clf = MultinomialNB()
clf.fit(x_traincv,y_train)
x_testcv=cv.transform(x_test)
```

```
predictions=clf.predict(x_testcv)


actual=np.array(y_test)
accuracy = metrics.accuracy_score(actual, predictions)
print('Accuracy: ',accuracy)


text_file = open("testsetwithoutlabels.txt", "r")
test_line = text_file.readlines()
test=cv.transform(test_line)
predict=clf.predict(test)


#output = open("e14403.txt","w+")


i=0
while i < len(test_line):
    #f.write("%f" %predict[i])
    print(predict[i])
    i+=1
```