data_analytics_ass-1

August 17, 2021

0.1 Q1

[56]:
```
## 1. File method

import os
fileurl = ''
filenames = os.listdir(fileurl)
content = []

for i in range(len(filenames)):
    f = open(fileurl+filenames[i], 'r')
    text = f.read()
    content.append(text)
    f.close()

# print(content)
```

[ ]:
```

```

[4]:
```
## 2. Using PlaintextCorpusReader

from nltk.corpus import PlaintextCorpusReader
a = PlaintextCorpusReader(fileurl, ".*")
```

[57]:
```
# a.sents()
```

[58]:
```
# len(a.sents())
```

[7]:
```
content1 = []
for i in range(len(a.sents())):
    b = ' '.join(a.sents()[i])
    b = b.replace("< br /><br />","")
    content1.append(b)

# print(content1)
```

0.2 Q2

[29]:
```
lower=[]
alpha=[]
for i in range(len(content)):
    lower.append(' '.join([word.lower() for word in content[i].replace("<br /
 ↪><br />"," ").split()]))
# print(lower)
for i in range(len(lower)):
    alpha.append(' '.join([word for word in lower[i].split() if word.
 ↪isalpha()]))
# print(alpha)
```

[30]:
```
import nltk
tokenized=[]
for i in range(len(alpha)):
    tokenized.append(nltk.word_tokenize(alpha[i]))
# print(tokenized)
```

[31]:
```
stopword_list=nltk.corpus.stopwords.words(fileids='english')
nostop=[]
for i in range(len(tokenized)):
    nostop.append([word for word in tokenized[i] if word not in stopword_list])
# print(nostop)
```

[32]:
```
from nltk import PorterStemmer
ps=PorterStemmer()
final=[]
for i in range(len(nostop)):
    final.append(' '.join([ps.stem(word) for word in nostop[i]]))
# print(final)
```

q3.

[33]:
```
from sklearn.feature_extraction.text import CountVectorizer
cntvec = CountVectorizer()
X = cntvec.fit_transform(final)
# print(X.toarray())
# print(cntvec.get_feature_names())
```

[34]:
```
import pandas as pd
df_countvect = pd.DataFrame(data = X.toarray(),columns=cntvec.
 ↪get_feature_names(),index=filenames)
# print(df_countvect)
```

[35]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
vc = tfidf.fit_transform(final).toarray()
```

```
# print(vc)
```

[36]:
```
import pandas as pd
df_tfidf = pd.DataFrame(data = vc,columns=tfidf.
↪get_feature_names(),index=filenames)
# print(df_tfidf)
```

[37]:
```
# Without using in-built functions

# performing countvectorize
doc_frequency_count={}
for i in range(len(final)):
    frequency_count={}
    for word in final[i].split():
        if word not in frequency_count.keys():
            frequency_count[word]=0
        frequency_count[word]+=1
    doc_frequency_count[filenames[i]]=frequency_count
# print(doc_frequency_count)

df=pd.DataFrame(doc_frequency_count)
df.fillna(0,inplace=True)
# df
```

[38]:
```
# performing tf-idf
import math

def IDF(corpus, unique_words):
   idf_dict={}
   N=len(corpus)
   for i in unique_words:
     count=0
     for sen in corpus:
       if i in sen.split():
         count=count+1
       idf_dict[i]=(math.log((1+N)/(count+1)))+1
   return idf_dict

def fit(whole_data):
    unique_words = set()
    if isinstance(whole_data, (list,)):
      for x in whole_data:
        for y in x.split():
          if len(y)<2:
            continue
          unique_words.add(y)
      unique_words = sorted(list(unique_words))
```

```
      vocab = {j:i for i,j in enumerate(unique_words)}
      Idf_values_of_all_unique_words=IDF(whole_data,unique_words)
    return vocab, Idf_values_of_all_unique_words

Vocabulary, idf_of_vocabulary=fit(final)
```

[39]:
```
# print(list(Vocabulary.keys()))
# print(list(idf_of_vocabulary.values()))
```

[42]:
```
import numpy as np
from scipy.sparse import csr_matrix
from collections import Counter
from sklearn.preprocessing import normalize

def transform(dataset,vocabulary,idf_values):
     sparse_matrix= csr_matrix( (len(dataset), len(vocabulary)), dtype=np.
 ↪float64)
     for row  in range(0,len(dataset)):
       number_of_words_in_sentence=Counter(dataset[row].split())
       for word in dataset[row].split():
           if word in  list(vocabulary.keys()):
               tf_idf_value=(number_of_words_in_sentence[word]/len(dataset[row].
 ↪split()))*(idf_values[word])
               sparse_matrix[row,vocabulary[word]]=tf_idf_value
     output =normalize(sparse_matrix, norm='l2', axis=1, copy=True,␣
 ↪return_norm=False)
     return output

final_output=transform(final,Vocabulary,idf_of_vocabulary)
# print(final_output.shape)
```

```
C:\Python\Python38\lib\site-packages\scipy\sparse\_index.py:82:
SparseEfficiencyWarning: Changing the sparsity structure of a csr_matrix is
expensive. lil_matrix is more efficient.
  self._set_intXint(row, col, x.flat[0])
```

[50]:
```
# print(final_output.toarray())
df_tfidf_vec = pd.DataFrame(data = final_output.toarray(),columns=tfidf.
 ↪get_feature_names(),index=filenames)
# print(df_tfidf_vec)
```

0.3 Q4:

[51]:
```
import requests
from bs4 import BeautifulSoup
import csv
```

```
URL = "http://www.values.com/inspirational-quotes"
r = requests.get(URL)

soup = BeautifulSoup(r.content, 'html5lib')

quotes_list = []
quotes_file = []

table = soup.find('div', attrs = {'id':'all_quotes'})

i=1
for row in table.findAll('div',
                         attrs = {'class':'col-6 col-lg-3 text-center␣
 ↪margin-30px-bottom sm-margin-30px-top'}):

    quotes_file.append('quote'+str(i))
    quotes_list.append(row.img['alt'].split(" #")[0])
    i=i+1
```

[52]:
```
# print(quotes_list) # all the quotes
```

[54]:
```
quotes_lower=[]
quotes_alpha=[]
for i in range(len(quotes_list)):
    quotes_lower.append(' '.join([word.lower() for word in quotes_list[i].
 ↪split()]))
# print(lower)
for i in range(len(quotes_lower)):
    quotes_alpha.append(' '.join([word for word in quotes_lower[i].split() if␣
 ↪word.isalpha()]))
# print(quotes_alpha)

quotes_tokenized=[]
for i in range(len(quotes_alpha)):
    quotes_tokenized.append(nltk.word_tokenize(quotes_alpha[i]))
# print(quotes_tokenized)

stopword_list=nltk.corpus.stopwords.words(fileids='english')
quotes_nostop=[]
for i in range(len(quotes_tokenized)):
    quotes_nostop.append([word for word in quotes_tokenized[i] if word not in␣
 ↪stopword_list])
# print(quotes_nostop)

ps=PorterStemmer()
quotes_final=[]
```

```
for i in range(len(quotes_nostop)):
    quotes_final.append(' '.join([ps.stem(word) for word in quotes_nostop[i]]))
# print(quotes_final)

cntvec = CountVectorizer()
quotes_fvec = cntvec.fit_transform(quotes_final)
# print(quotes_fvec.toarray()) # feature vector

import pandas as pd
df_quotesvect = pd.DataFrame(data = quotes_fvec.toarray(),columns=cntvec.
 ↪get_feature_names(), index=quotes_file)
# print(df_quotesvect)
```

[ ]:

[ ]: