WEB MINING

[ASSIGNMENT-1]

Roll no.-2081001204

M.Tech. 2nd Semester

(Statistical Computing-Data Science)

Submitted by:Anjali Gautam

Ques 1. Given a web document such as email/contact details.Write a program to extract the following using regular expression:

(Write Regular Expressions in documentation)

a. Extract Phone number(10 digit number) and other variants.
b. Extract Email id.
c. Extract Date.

Sol.:

#Import required packages such as *re* is used for functionality of regular expression and *requests* is used for the functionality related to url here.

```
import re

import datetime

from datetime import date

import requests

the_url = 'http://www.jnu.ac.in/iha_admin'
```

#Create function for obtaining the whole document by providing the url:

```
def get_doc(url):

     r= requests.get(url).text

     return r
```

#After obtaining the whole document,create function for extracting *the E-mail id,Date* and *Phone Number* by using their respective regular expressions:

```
def extract(d):

     email_id= re.findall(r"[\w.]+@[\w.]+",d)

      k=[]

      for i in email_id:

                k.append(i)

     print(k)

     date =   re.findall(r"[\d]{1,2}\.[\d]{1,2}\.[\d]{4}",d)

     l=[]

     for i in date:
```

```
                l.append(i)
     print(l)
     contact_num= re.findall(r"\d\d\d\d\d\d\d\d",d)
     n=[]
      for i in contact_num:
               n.append(i)
      print(n)
```

#Function calling for obtaining the text and extracting the *E-mail id,Date* and *Phone Number.*

```
document= get_doc(the_url)
extracted_doc=extract(document)
```

Ques 2. Find the phrase/collocations from a text document such as news article/any article of social interest,any article on current topic,a book chapter using following approaches:

Take a text document of your choice,show the results on atleast three documents.

Discuss the logic/formulae used and sample results in documentation along with analytical comments,highlight the actual keyphrases based on your general knowledge.

a) Most Frequent bigrams.
b) Ranked on the basis of mutual information.Use standard formula for mutual information(Hint:Find probability of combined occurrence of bigram and divide by their individual occurrence probability)
c) Noun phrases based on the following POS patterns.

Tag Pattern	**Example**
AN	Linear Function
NN	Regression Coefficient
AAN	Gaussian Random Variable
ANN	Cumulative Distribution Function
NAN	Mean Squared Error
NPN	Degree of Freedom
NNN	Class Probability Function

Sol.:

#Importing important libraries such as nltk,CountVectorizer, TfidfVectorizer etc:

```
import nltk
import re
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import pandas as pd
```

#Now take the text

```
text1 =["""The most common symptoms of COVID-
19 are fever, tiredness, and dry cough.
Some patients may have aches and pains, nasal congestion, runn
y nose,  sore throat or diarrhea.
These symptoms are usually mild and begin gradually.
Some people become infected but don’t develop any symptoms and
don't   feel unwell.Most people (about 80%) recover from the d
isease without   needing special treatment.Around 1 out of eve
ry 6 people who gets COVID-
19 becomes seriously ill and develops difficulty breathing.
Older people, and those with underlying medical problems like
high blood pressure, heart problems or diabetes, are more like
ly to develop serious illness.People with fever, cough and dif
ficulty breathing should seek medical attention."""]

# Using CountVectorizer for getting bigrams.
vector1= CountVectorizer(ngram_range=(2, 2))
X1 = vector1.fit_transform(text1) #to count frequeny of each
bigram.
f= (vector1.get_feature_names())    #for obtaining bigrams.
print("\n\nFeatures : \n", f)
print("\n\nX1 : \n", X1.toarray())

# Ranked on the basis of mutual information using
TfidfVectorizer.
vector2= TfidfVectorizer(ngram_range=(2, 2))
X2 = vector2.fit_transform(text1)
s= (X2.toarray())
print("\n\nScores : \n",s)
sums = X2.sum(axis=0)
data= []
for col, term in enumerate(f):
    data.append((term, sums[0, col]))
ranking= pd.DataFrame(data, columns=['term', 'rank'])
words = (ranking.sort_values('rank', ascending=False))
print("\n\nWords head : \n", words.head(10))


# Generate noun phrases based on the Part of Speech Pattern:
import nltk
text2= """Coronaviruses are a large family of viruses which ma
y cause illness in animals or humans.In humans, several corona
viruses are known to cause respiratory infections ranging from
 the common cold to more severe diseases such as Middle East R
espiratory Syndrome (MERS)and SevereAcute Respiratory Syndrome
 (SARS).The most recently discovered coronavirus causes corona
virus disease COVID-19"""

tokens = nltk.word_tokenize(text2)    #convert the text into
tokens.

print(tokens)
tag = nltk.pos_tag(tokens)   #assign the tag to each token.
print(tag)
```

<u>Ques 3</u>: Consider some results from tutorial sites by giving a query,say 'tutorials on data structures' or result from any other repository.Store the results.The files are expected to be of type Doc,PDF,HTML(ignore other files),use document parser,pdf parser and html parser to parse the content in the form of text,tables and images.
Divide the text into sentences and words.
Remove Stop words.
<u>Sol.</u> :

- ***<u>To obtain results by giving some query as'tutorials on web mining</u>***:

```
# we have to import required package 'googlesearch'
to perform the functionality of Google Search

try:
    from googlesearch import search
except ImportError:
    print("No module named 'google' found")

# to search
query = "tutorials on web mining"

for j in search(query,stop=10):
    print(j)
```

- ***<u>To parse the HTML File</u>:***

```
#Firstly we have to import the required packages such as
Beautifulsoup,get_text etc.

import urllib.request
from urllib.request import urlopen
from bs4 import BeautifulSoup
from inscriptis import get_text
import re
url='https://en.wikipedia.org/wiki/Mark_Zuckerberg'

#To open the url
html = urlopen(url)
#Use html.parser to parse the html file
bs = BeautifulSoup(html, 'html.parser')
#Store all images in the 'images' variable by using find_all
function
images = bs.find_all('img', {'src':re.compile('.jpg')})
```

```
html1 = urllib.request.urlopen(url).read().decode('utf-8')

#For getting the text from the Html file

text = get_text(html1)
print(text)

#For print all the image file in the html document
for image in images:
    print(image['src']+'\n')
```

- ***To parse the pdf file*** :

```
# creating a pdf file object
pdfFileObj = open('machine learning.pdf', 'rb')

# creating a pdf reader object
pdfReader = PyPDF2.PdfFileReader(pdfFileObj)

# printing number of pages in pdf file
print(pdfReader.numPages)

# creating a page object
pageObj = pdfReader.getPage(0)

# extracting text from page
print(pageObj.extractText())
import re
s = pageObj.extractText()
print(s.split())
m = re.split(r'(?<=[^A-Z].[.?]) +(?=[A-Z])', s)
for i in m:
  print(i)

# closing the pdf file object
pdfFileObj.close()
```

- ***To parse the Doc file*** :

```
from docx import Document
d= Document('IRJET-V3I12333.docx')
d.save('IRJET-V3I12333.docx')
#Extracting paragraph from Doc file
for para in d.paragraphs:
    print(para.text)
```

```
#Extracting tables from Doc file

for table in d.tables:
    for row in table.rows:
      for cell in row.cells:
        for para in cell.paragraphs:
          print (para.text)
#Extracting images from Doc file

for image in d.inline_shapes:
    print (image.width, image.height)
```

- ***Divide the the text into sentences and words***

```
# importing required packages

import nltk
nltk.download()
nltk.download('punkt')
text="""web mining is the process of data mining techniques to
automatically discover and extract information from web
documents and services.The main purpose of web mining is
discovering useful information from the world wide web and its
usage patterns.It is used to predict user behaviour."""

#Split text into sentences

sentence=nltk.sent_tokenize(text)

#Split text into words

word=nltk.words_tokenize(text)
```

- ***Removing Stopwords***

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example= "This is a sample sentence, showing off the stop
words filtration."

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example)

filtered_sentence =[w for w in word_tokens if not w in
stop_words]

filtered_sentence = []
```

```
for w in word_tokens:
     if w not in stop_words:
          filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```