DIGITAL ASSIGNMENT #1 – ELA

Name: Aman Anand

Reg. No.: 19BCE0521

Course: Web Mining Lab

Faculty: Shashank Mouli Satapathy

Slot: L39+L40

Questions:

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

1. Write a python program to
   a. Extract the contents (excluding any tags) from two websites (https://en.wikipedia.org/wiki/Web_mining&https://en.wikipedia.org/wiki/Data_mining).
   b. Remove stopwords [using Spacy Module] (including the special characters/symbols) from the contents retrieved from those two URLs and save the contents in two separate .txt file.
      - [List of additional Stop words to be considered = [dot, comma, single-quote, double quote, question mark, brackets [square, parentheses, curly, angle], exclamation mark]]
   c. Display the POS tag (sentence-wise) for all the stopwords (excluding the special character/symbols), which are removed from the content, using pandas dataframe as per the format given below:

Original Sentence	List of Stopwords	POS-Tags
Web mining is the application of data mining techniques to discover patterns from the World Wide Web.	is	VBZ
	the	DT
	of	IN
	to	TO
	from	IN
	the	DT

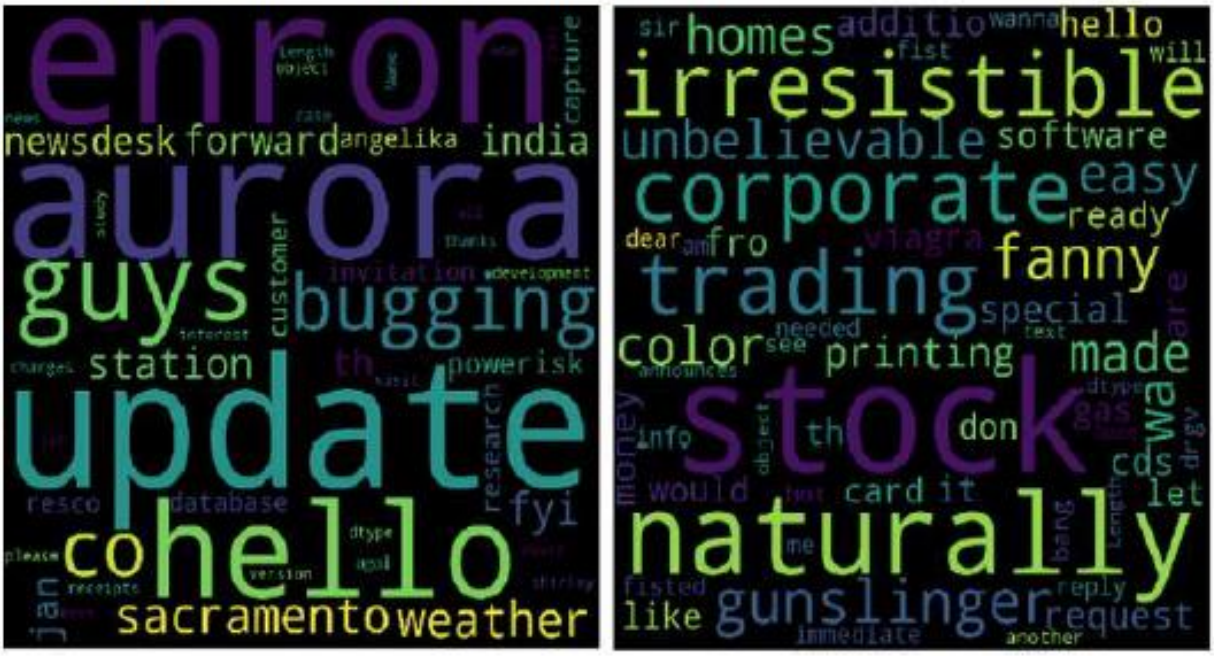
Assessment - 1

   d. Display the Term-Document incidence matrix using Boolean, Bag-of-words and Complete representation (Use pandas dataframe). Prepare three separate table, one for each type of representation as per the format given below:

Terms	DOC1	DOC2
Web	5	0
Data	0	1

   e. Input a search a query (preferably a sentence) and compare the contents of the both pages with the processed query. Display the similarity result based on highest frequency matching count of the term.

2. Write a python program to prepare the **Word Clouds** representation based on the content present in the two document files prepared in Q.No. 1. A sample Word Clouds representation is provided below for reference.



3. Write a python program to show the implementation of sentence paraphrasing through synonyms (retaining semantic meaning) for the following four sentences. Display at least three other paraphrased sentences for each sentence mentioned below.

   a. The quick brown fox jumps over the lazy dog
   b. We can rewrite history as much as we like.
   c. Once you know all the elements, it's not difficult to pull together a sentence.
   d. The incessant ticking and chiming echoed off the weathered walls of the clock repair shop.

P.T.O

Answer:

Code:

```
"""
Created on Sat Feb 20 11:38:32 2021
@author: Aman Anand 19BCE0521
"""
#imports
import spacy
import nltk
import urllib
import bs4 as bs
from nltk.tokenize import word_tokenize as w_t
from nltk.tag import pos_tag
from nltk.corpus import wordnet as wn
import random
from spacy.lang.en.stop_words import STOP_WORDS as stw
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud


#scraping wikipedia
source1 =
urllib.request.urlopen('https://en.wikipedia.org/wiki/Web_mining').read().decod
e('utf8')
source2 = urllib.request.urlopen('https://en.wikipedia.org/wiki/Data_mining')
text1 = bs.BeautifulSoup(source1,'html.parser').get_text()
text2 = bs.BeautifulSoup(source2,'html.parser').get_text()


#tokenizing scraped text
token_wm = w_t(text1)
token_dm = w_t(text2)

#adding and removing stop words from the scarped text
nlplib = spacy.load('en_core_web_sm')
stop_words = list(stw)
```

```
new_stpwrd = [";",":","-
",".",",","\"","'","?","[","]","(",")","{","}","<",">","!"]
for _ in new_stpwrd:
    if _ not in stop_words:
        stop_words.append(_)
for word in stop_words:
    lex = nlplib.vocab[word]
    lex.is_stop = True


wm_filtex=[];dm_filtex=[]
for _ in token_wm:
    lex = nlplib.vocab[_]
    if lex.is_stop == False:
        wm_filtex.append(_)
for _ in token_dm:
    lex = nlplib.vocab[_]
    if lex.is_stop == False:
        dm_filtex.append(_)


#saving to external files:
doc1 = open(r"D:\\VIT\\1. WINTER SEMESTER (20-21)\\Web
Mining\\LAB\\WM.txt",'w', encoding=('utf-8'))
for _ in wm_filtex:
    doc1.write(_ + '\n')
doc1.close()
doc2 = open(r"D:\\VIT\\1. WINTER SEMESTER (20-21)\\Web
Mining\\LAB\\DM.txt",'w', encoding=('utf-8'))
for _ in dm_filtex:
    doc2.write(_ + '\n')
doc2.close()


#Question 1(c)
wm_sen = nltk.sent_tokenize(text1)
dm_sen = nltk.sent_tokenize(text2)
wm_sen = wm_sen[1:]
dm_sen = dm_sen[1:]
```

```
sentence1 = []; stpWrds1=[]; posTag1=[];sentStpWrds1=[];sentPosTag1=[]
for _ in wm_sen:
    wrdtkn1 = w_t(_)
    wrdtkn1 = [_ for _ in wrdtkn1 if _ not in new_stpwrd]
    sentence1.append(' '.join(map(str, wrdtkn1)))
    tag = nltk.pos_tag(wrdtkn1)
    for x in tag:
        lexx = nlplib.vocab[x[0]]
        if lexx.is_stop == True:
            stpWrds1.append(x[0])
            posTag1.append(x[1])
    sentStpWrds1.append(stpWrds1)
    sentPosTag1.append(posTag1)

sentence2 = []; stpWrds2=[]; posTag2=[];sentStpWrds2=[];sentPosTag2=[]
for _ in dm_sen:
    wrdtkn2 = w_t(_)
    wrdtkn2 = [_ for _ in wrdtkn2 if _ not in new_stpwrd]
    sentence2.append(' '.join(map(str, wrdtkn2)))
    tag = nltk.pos_tag(wrdtkn2)
    for x in tag:
        lexx = nlplib.vocab[x[0]]
        if lexx.is_stop == True:
            stpWrds2.append(x[0])
            posTag2.append(x[1])
    sentStpWrds2.append(stpWrds2)
    sentPosTag2.append(posTag2)

sent1=[];c=0;
for _ in sentStpWrds1:
    leng=len(_)
    while(leng>0):
        if (leng==len(_)):
            sent1.append(sentence1[c])
        else:
```

```
            sent1.append(" ")
        leng -=1
    c+=1

sent2=[];d=0;
for _ in sentStpWrds2:
    lengg=len(_)
    while(lengg>0):
        if (lengg==len(_)):
            sent2.append(sentence2[c])
        else:
            sent2.append(" ")
        lengg -=1
    d+=1

stpword1 =[]
for _ in sentStpWrds1:
    for j in _:
        stpword1.append(j)

stpword2 =[]
for _ in sentStpWrds2:
    for j in _:
        stpword2.append(j)

postagg1=[]
for _ in sentPosTag1:
    for j in _:
        postagg1.append(j)

postagg2=[]
for _ in sentPosTag2:
    for j in _:
        postagg2.append(j)
```

```
df1=pd.DataFrame(data={
    'Sentence': sent1,
    'Stop Words': stpword1,
    'PosTags': postagg1})

df2=pd.DataFrame(data={
    'Sentence': sent2,
    'Stop Words': stpword2,
    'PosTags': postagg2})
print("For the Web Mining Wikipedia Page -->\n")
print(df1.head(20))
print("For the Data Mining Wikipedia Page-->\n")
print(df2.head(20))

#Question 1(d)

#Boolean Representation
WM_Boolean=[];DM_Boolean=[];WM_BagOfWords=[];DM_BagOfWords=[];WM_Complete=[];DM
_Complete=[]
totalTermList = list(set(wm_filtex + dm_filtex))
for _ in totalTermList:
    if _ in wm_filtex:
        WM_Boolean.append(1)
    else:
        WM_Boolean.append(0)
    if _ in dm_filtex:
        DM_Boolean.append(1)
    else:
        DM_Boolean.append(0)

dfBool = pd.DataFrame({
    'Terms': totalTermList,
    'DOC1': WM_Boolean,
    'DOC2': DM_Boolean})
print(dfBool.head(20))
```

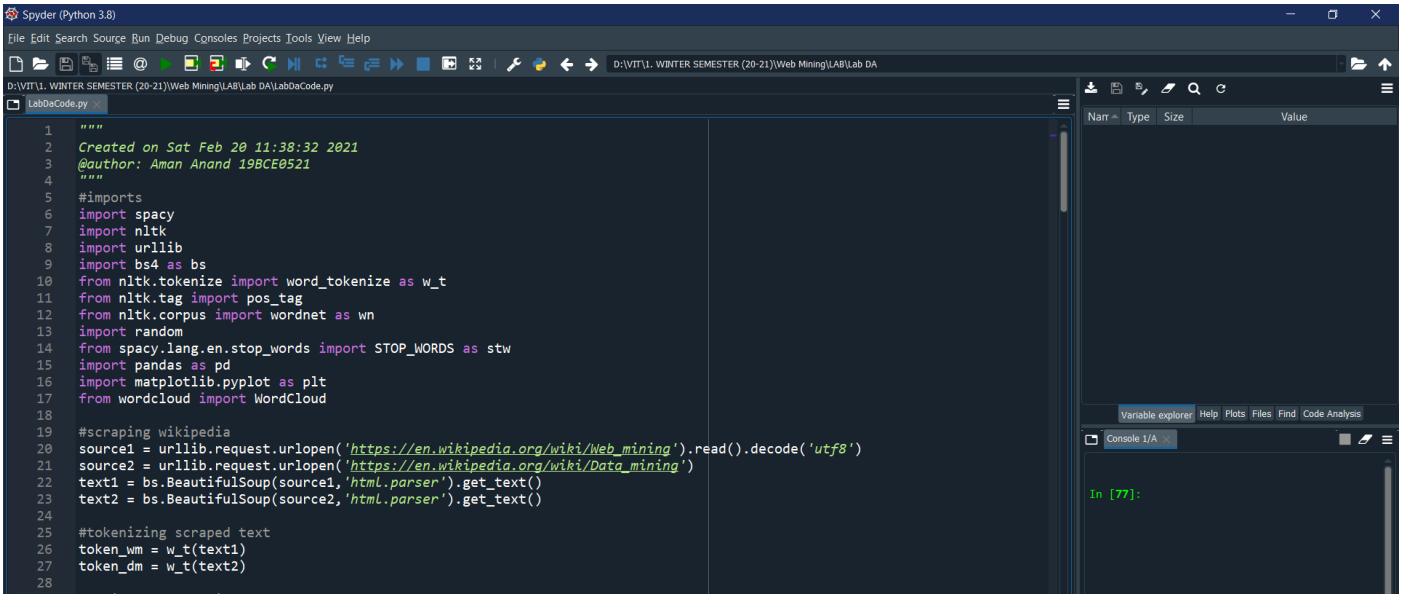
```
#Bag of Words Representation
for _ in totalTermList:
    WM_BagOfWords.append(wm_filtex.count(_))
    DM_BagOfWords.append(dm_filtex.count(_))

dfBoW = pd.DataFrame({
    'Terms': totalTermList,
    'DOC1': WM_BagOfWords,
    'DOC2': DM_BagOfWords})
print(dfBoW.head(20))

#Complete Representation
wordTotalList = wm_filtex + dm_filtex
for p in wordTotalList:
    WM_Complete.append([k for k,x in enumerate(wm_filtex) if x==p])
    DM_Complete.append([k for k,x in enumerate(dm_filtex) if x==p])

dfComplete = pd.DataFrame({
    'Terms': wordTotalList,
    'DOC1': WM_Complete,
    'DOC2': DM_Complete})

print(dfComplete.head(10))

#Question 1(e)
def queryRespond():
    query = input()
    tokens = w_t(query)
    query_filter=[]
    for _ in tokens:
        lex = nlplib.vocab[_]
        if lex.is_stop == False:
            query_filter.append(_)
    count1=[];count2=[];
```

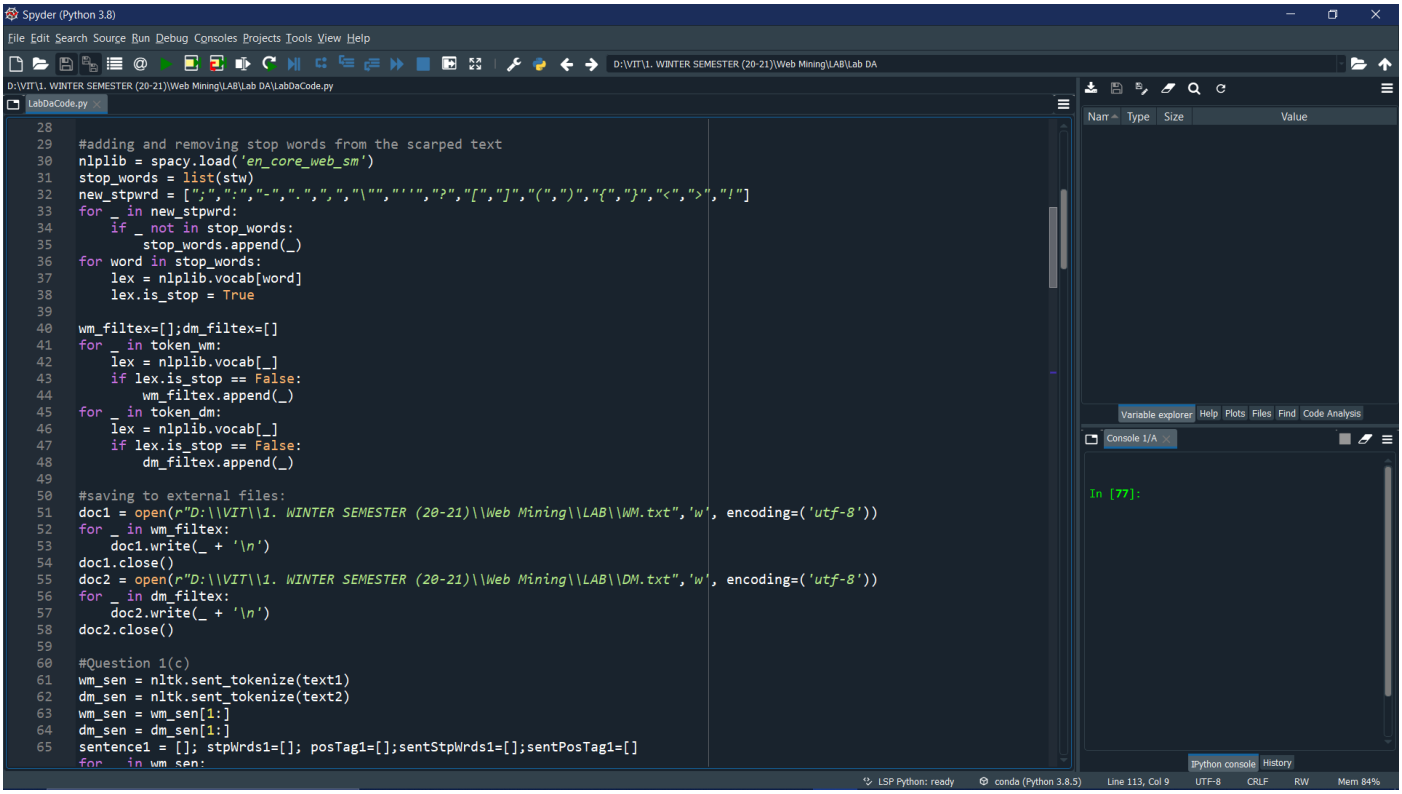
```
    for _ in query_filter:
        if _ in wm_filtex:
            count1.append(_)
        if _ in dm_filtex:
            count2.append(_)
    if len(count1)>len(count2):
        print("Document 1 is more similar w.r.t the query")
        print("DOC 1 occurence frequency:",len(count1))
        print("DOC 2 occurence frequency:",len(count2))
    elif len(count1)<len(count2):
        print("Document 2 is more similar w.r.t the query")
        print("DOC 2 occurence frequency:",len(count2))
        print("DOC 1 occurence frequency:",len(count1))
    else:
        print("Both documents are equally similar")
        print("DOC 1 occurence frequency:",len(count1))
        print("DOC 2 occurence frequency:",len(count2))

queryRespond()

#Question 2
wmTXT = " ";dmTXT = " "
wmTXT = wmTXT.join(wm_filtex)
dmTXT = dmTXT.join(dm_filtex)
WC1 = WordCloud(width=480, height=480, margin=0).generate(wmTXT)
plt.imshow(WC1)
plt.axis("off")
plt.show()
WC2 = WordCloud(width=480, height=480, margin=0).generate(dmTXT)
plt.imshow(WC2)
plt.axis("off")
plt.show()

#Question 3
def tag(sentence):
```

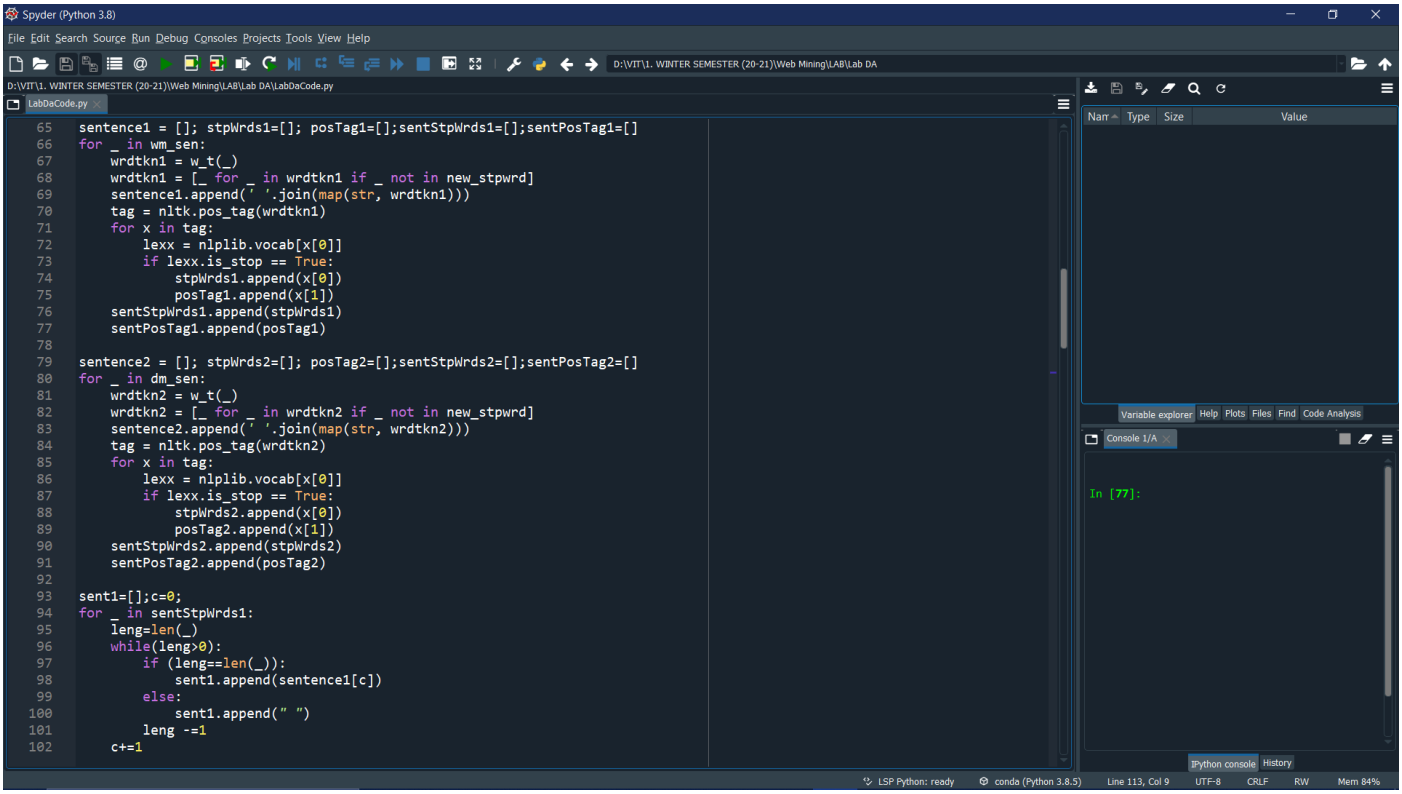
```
    words = w_t(sentence)
    words = pos_tag(words)
    return words

def paraphraseable(tag):
    return tag.startswith('NN') or tag == 'VB' or tag.startswith('JJ')

def pos(tag):
    if tag.startswith('NN'):
        return wn.NOUN
    elif tag.startswith('V'):
        return wn.VERB

def synonyms(word, tag):
    lemma_lists = [ss.lemmas() for ss in wn.synsets(word, pos(tag))]
    lemmas = [lemma.name() for lemma in sum(lemma_lists, [])]
    return set(lemmas)

def synonymIfExists(sentence):
    for (word, t) in tag(sentence):
        if paraphraseable(t):
            syns = synonyms(word, t)
            if syns:
                if len(syns) > 1:
                    yield [word, list(syns)]
                    continue
        yield [word, []]

def paraphrase(sentence):
    return [x for x in synonymIfExists(sentence)]

listOfSentences = ["The quick brown fox jumps over the lazy dog",
                   "We can rewrite history as much as we like.",
                   "Once you know all the elements, it’s not difficult to pull
together a sentence.",
```

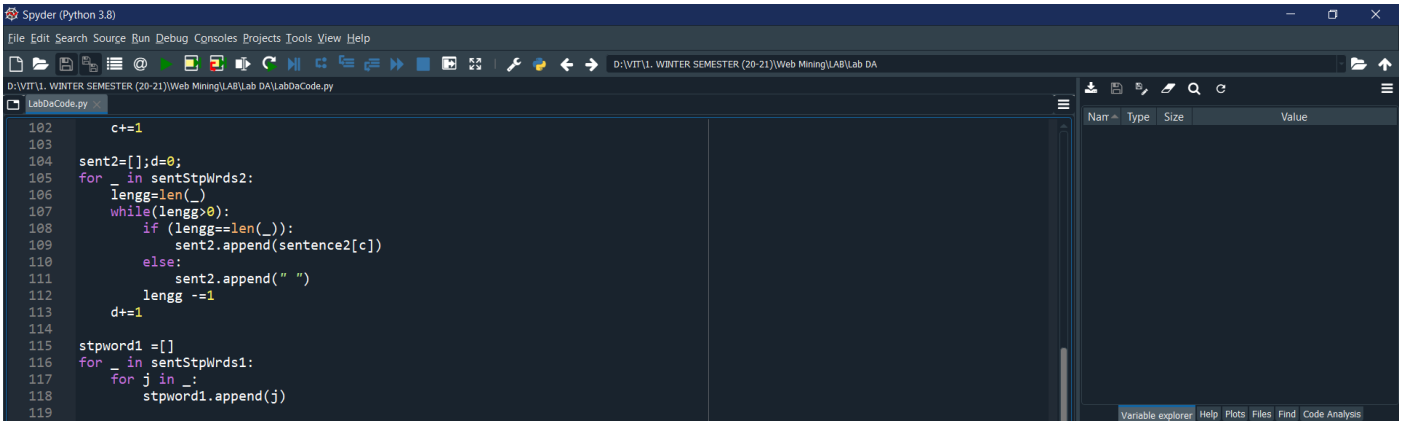
```
                    "The incessant ticking and chiming echoed off the weathered
walls of the clock repair shop."]


for sentence in listOfSentences:
    print(sentence)
    print("----------------\n")
    paraSent = paraphrase(sentence)
    phrase = []
    x = []
    for i in range(0,3):
        x=[]
        for i in paraSent:
            if(len(i[1])==0):
                t = i[0]
            else:
                t = random.choice(i[1])
            x.append(t)
        phrase.append(" ".join(map(str,x)))
    for i in phrase:
        print(i)
    print("\n\n")
```

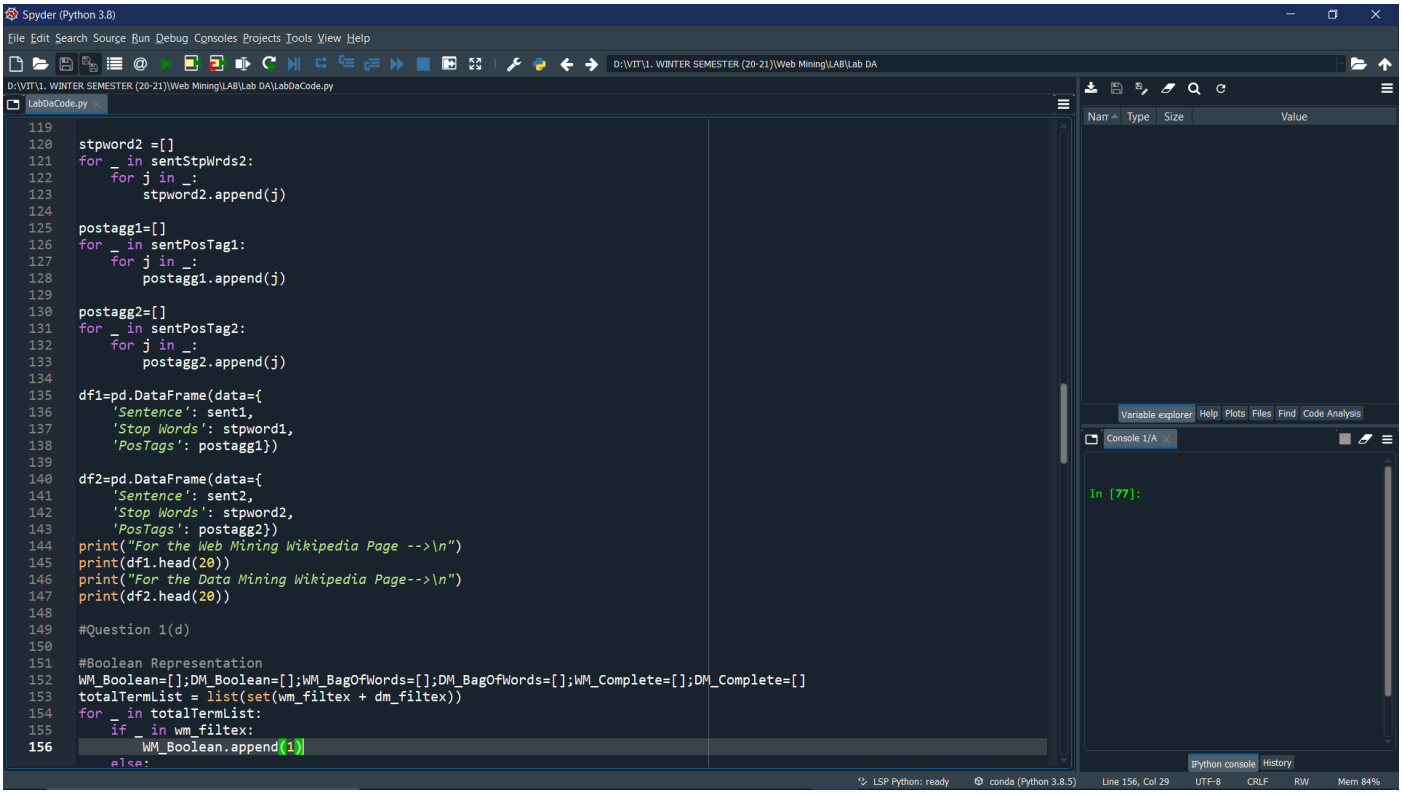
Code Editor Screenshots:



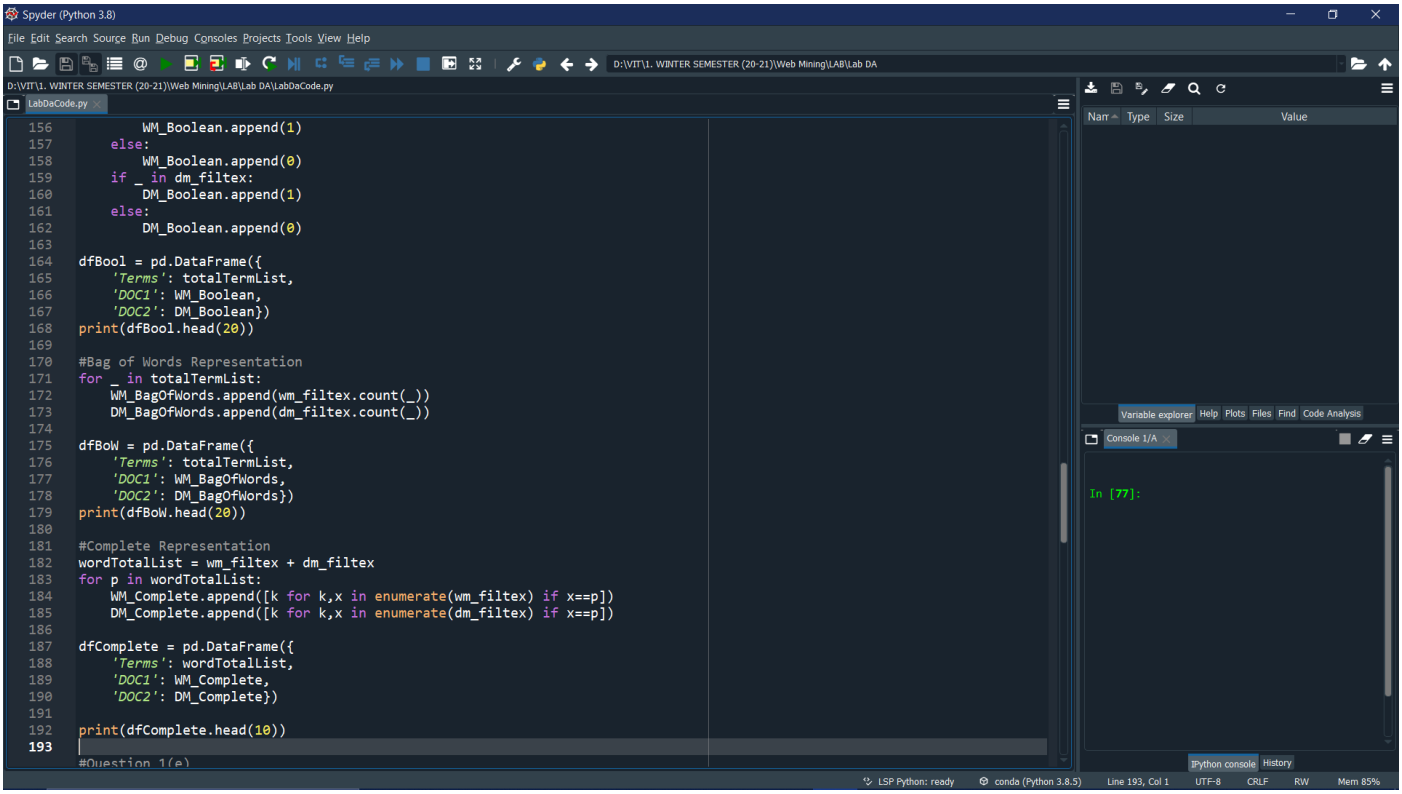

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA\LabDaCode.py
LabDaCode.py
#adding and removing stop words from the scarped text
nlplib = spacy.load('en_core_web_sm')
stop_words = list(stw)
new_stpwrd = [",","?",":","-",".","'","\"","'","?","[","]","(",")","{","}","<",">","!"]
for _ in new_stpwrd:
if _ not in stop_words:
stop_words.append(_)
for word in stop_words:
lex = nlplib.vocab[word]
lex.is_stop = True
wm_filtex=[];dm_filtex=[]
for _ in token_wm:
lex = nlplib.vocab[_]
if lex.is_stop == False:
wm_filtex.append(_)
for _ in token_dm:
lex = nlplib.vocab[_]
if lex.is_stop == False:
dm_filtex.append(_)
#saving to external files:
doc1 = open(r"D:\\VIT\\1. WINTER SEMESTER (20-21)\\Web Mining\\LAB\\WM.txt",'w', encoding='utf-8')
for _ in wm_filtex:
doc1.write(_ + '\n')
doc1.close()
doc2 = open(r"D:\\VIT\\1. WINTER SEMESTER (20-21)\\Web Mining\\LAB\\DM.txt",'w', encoding='utf-8')
for _ in dm_filtex:
doc2.write(_ + '\n')
doc2.close()
#Question 1(c)
wm_sen = nltk.sent_tokenize(text1)
dm_sen = nltk.sent_tokenize(text2)
wm_sen = wm_sen[1:]
dm_sen = dm_sen[1:]
sentence1 = []; stpWrds1=[]; posTag1=[];sentStpWrds1=[];sentPosTag1=[]
for _ in wm_sen:
Variable explorer Help Plots Files Find Code Analysis
Console 1/A
In [77]:
IPython console History
LSP Python: ready
conda (Python 3.8.5)
Line 113, Col 9
UTF-8
CRLF
RW
Mem 84%


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA\LabDaCode.py
LabDaCode.py
sentence1 = []; stpWrds1=[]; posTag1=[];sentStpWrds1=[];sentPosTag1=[]
for _ in wm_sen:
wrdtkn1 = w_t(_)
wrdtkn1 = [_ for _ in wrdtkn1 if _ not in new_stpwrd]
sentence1.append(' '.join(map(str, wrdtkn1)))
tag = nltk.pos_tag(wrdtkn1)
for x in tag:
lexx = nlplib.vocab[x[0]]
if lexx.is_stop == True:
stpWrds1.append(x[0])
posTag1.append(x[1])
sentStpWrds1.append(stpWrds1)
sentPosTag1.append(posTag1)
sentence2 = []; stpWrds2=[]; posTag2=[];sentStpWrds2=[];sentPosTag2=[]
for _ in dm_sen:
wrdtkn2 = w_t(_)
wrdtkn2 = [_ for _ in wrdtkn2 if _ not in new_stpwrd]
sentence2.append(' '.join(map(str, wrdtkn2)))
tag = nltk.pos_tag(wrdtkn2)
for x in tag:
lexx = nlplib.vocab[x[0]]
if lexx.is_stop == True:
stpWrds2.append(x[0])
posTag2.append(x[1])
sentStpWrds2.append(stpWrds2)
sentPosTag2.append(posTag2)
sent1=[];c=0;
for _ in sentStpWrds1:
leng=len(_)
while(leng>0):
if (leng==len(_)):
sent1.append(sentence1[c])
else:
sent1.append(" ")
leng -=1
c+=1
Variable explorer Help Plots Files Find Code Analysis
Console 1/A
In [77]:
IPython console History
LSP Python: ready
conda (Python 3.8.5)
Line 113, Col 9
UTF-8
CRLF
RW
Mem 84%

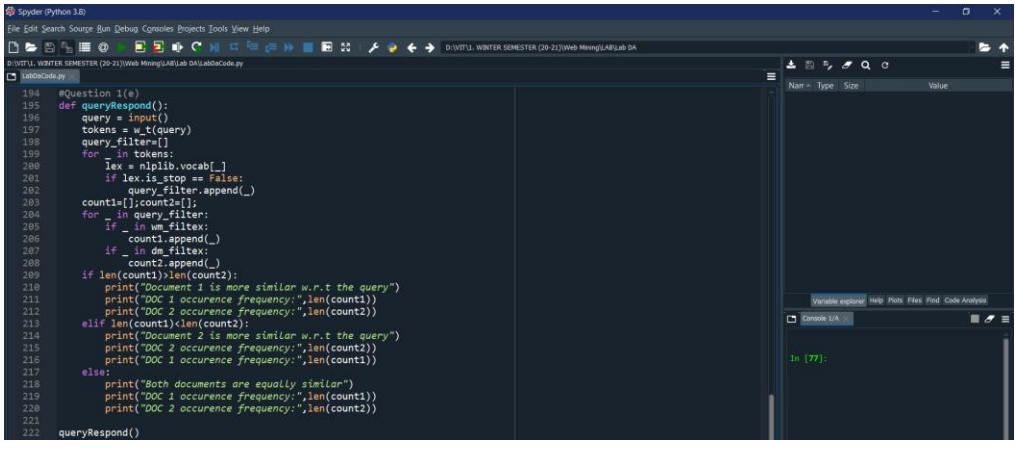

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA\LabDaCode.py
LabDaCode.py
c+=1
sent2=[];d=0;
for _ in sentStpWrds2:
lengg=len(_)
while(lengg>0):
if (lengg==len(_)):
sent2.append(sentence2[c])
else:
sent2.append(" ")
lengg -=1
d+=1
stpword1 =[]
for _ in sentStpWrds1:
for j in _:
stpword1.append(j)
Variable explorer Help Plots Files Find Code Analysis



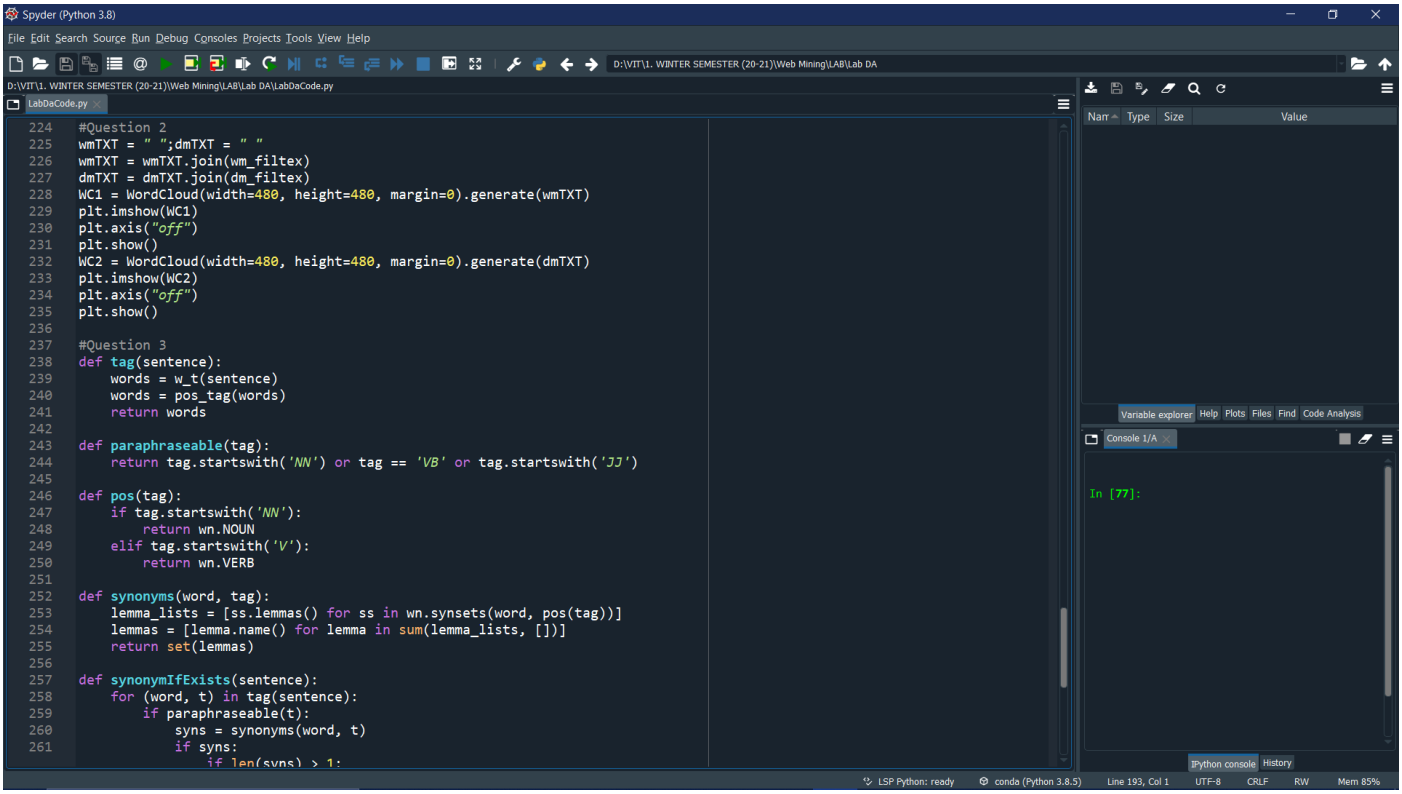
```
stpword2 =[]
for _ in sentStpWrds2:
    for j in _:
        stpword2.append(j)

postagg1=[]
for _ in sentPosTag1:
    for j in _:
        postagg1.append(j)

postagg2=[]
for _ in sentPosTag2:
    for j in _:
        postagg2.append(j)

df1=pd.DataFrame(data={
    'Sentence': sent1,
    'Stop Words': stpword1,
    'PosTags': postagg1})

df2=pd.DataFrame(data={
    'Sentence': sent2,
    'Stop Words': stpword2,
    'PosTags': postagg2})
print("For the Web Mining Wikipedia Page -->\n")
print(df1.head(20))
print("For the Data Mining Wikipedia Page-->\n")
print(df2.head(20))

#Question 1(d)

#Boolean Representation
WM_Boolean=[];DM_Boolean=[];WM_BagOfWords=[];DM_BagOfWords=[];WM_Complete=[];DM_Complete=[]
totalTermList = list(set(wm_filtex + dm_filtex))
for _ in totalTermList:
    if _ in wm_filtex:
        WM_Boolean.append(1)
    else:
```

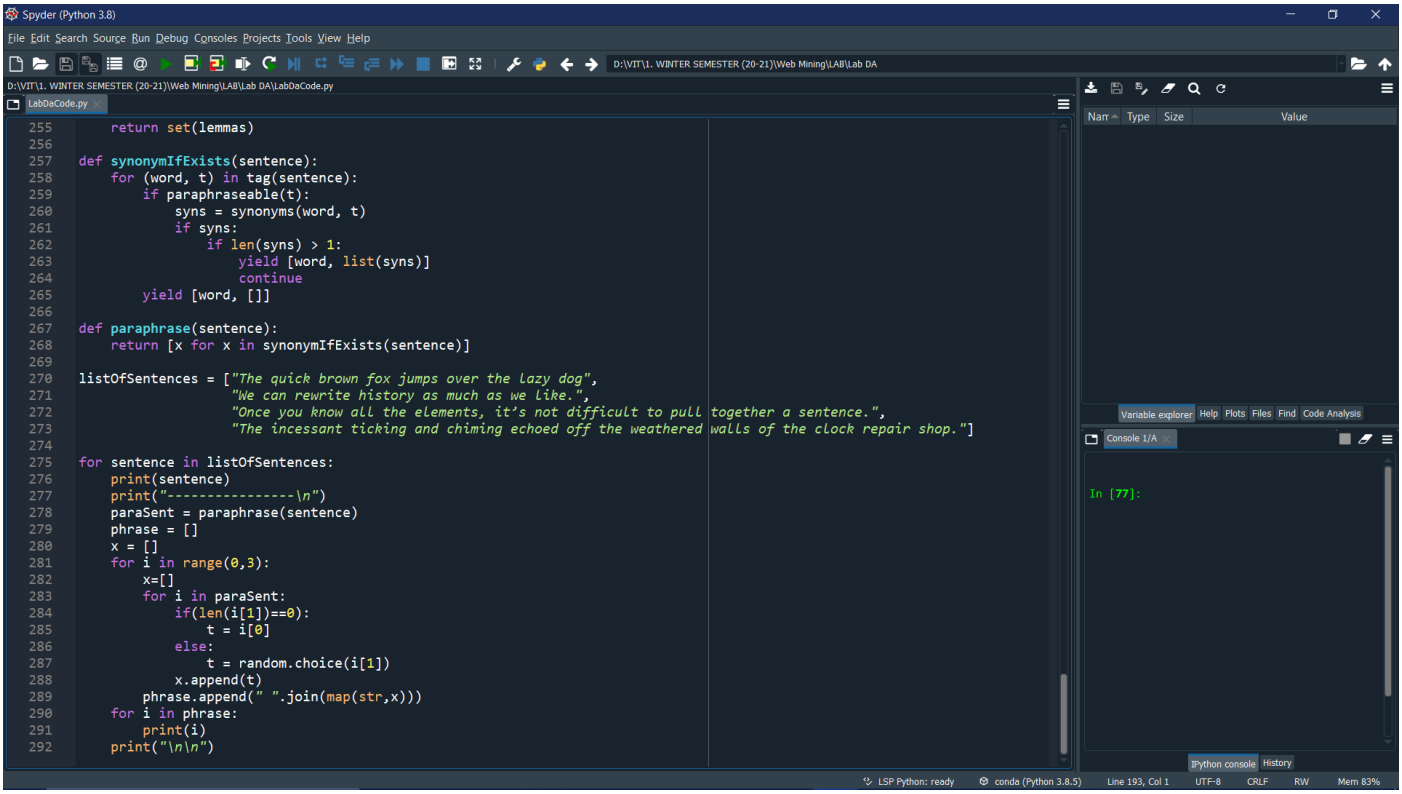


```
        WM_Boolean.append(1)
    else:
        WM_Boolean.append(0)
    if _ in dm_filtex:
        DM_Boolean.append(1)
    else:
        DM_Boolean.append(0)

dfBool = pd.DataFrame({
    'Terms': totalTermList,
    'DOC1': WM_Boolean,
    'DOC2': DM_Boolean})
print(dfBool.head(20))

#Bag of Words Representation
for _ in totalTermList:
    WM_BagOfWords.append(wm_filtex.count(_))
    DM_BagOfWords.append(dm_filtex.count(_))

dfBoW = pd.DataFrame({
    'Terms': totalTermList,
    'DOC1': WM_BagOfWords,
    'DOC2': DM_BagOfWords})
print(dfBoW.head(20))

#Complete Representation
wordTotalList = wm_filtex + dm_filtex
for p in wordTotalList:
    WM_Complete.append([k for k,x in enumerate(wm_filtex) if x==p])
    DM_Complete.append([k for k,x in enumerate(dm_filtex) if x==p])

dfComplete = pd.DataFrame({
    'Terms': wordTotalList,
    'DOC1': WM_Complete,
    'DOC2': DM_Complete})

print(dfComplete.head(10))

#Question 1(e)
```

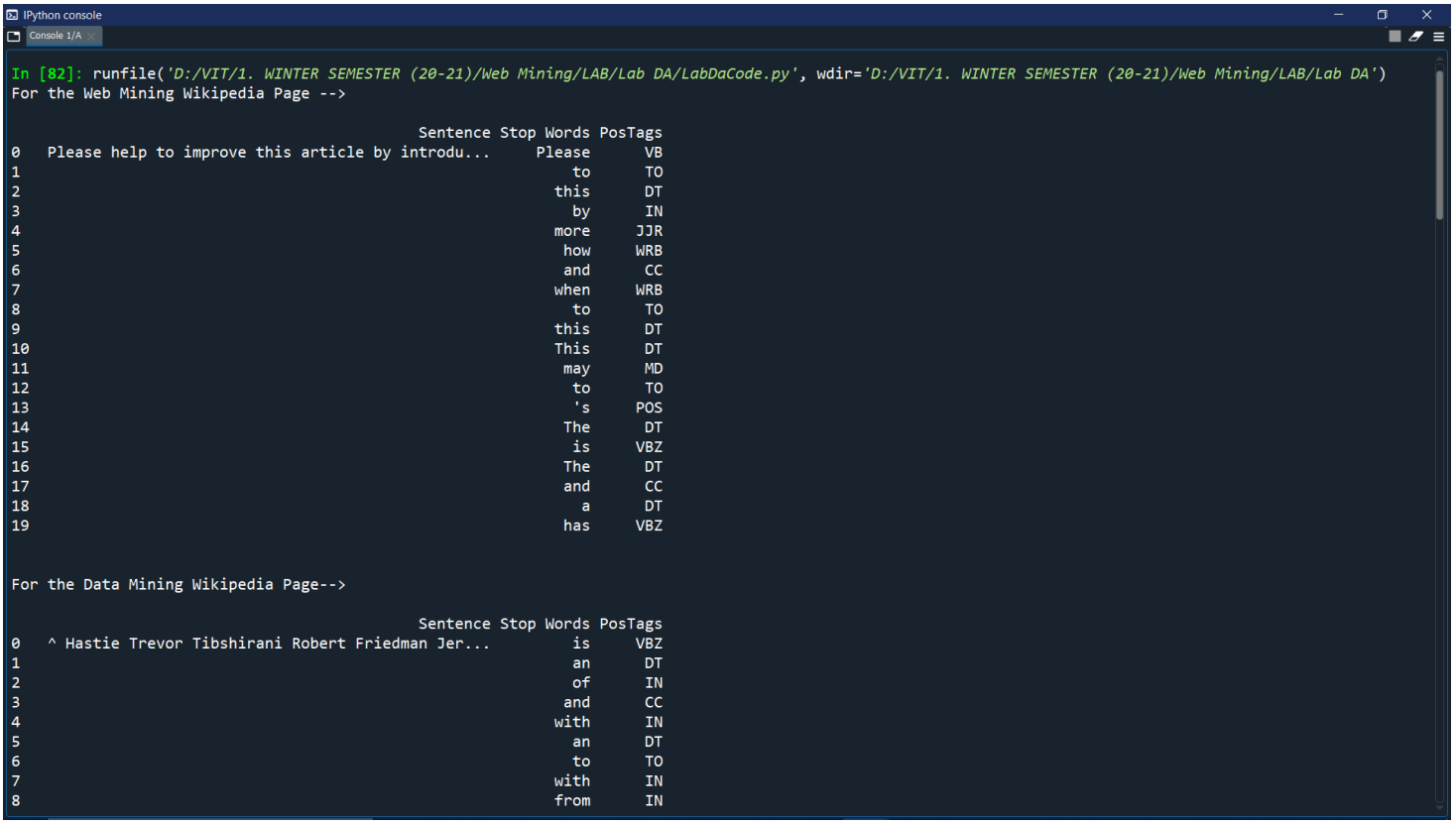


```
#Question 1(e)
def queryRespond():
    query = input()
    tokens = w_t(query)
    query_filter=[]
    for _ in tokens:
        lex = nlplib.vocab[_]
        if lex.is_stop == False:
            query_filter.append(_)
    count1=[];count2=[];
    for _ in query_filter:
        if _ in wm_filtex:
            count1.append(_)
        if _ in dm_filtex:
            count2.append(_)
    if len(count1)>len(count2):
        print("Document 1 is more similar w.r.t the query")
        print("DOC 1 occurence frequency:",len(count1))
        print("DOC 2 occurence frequency:",len(count2))
    elif len(count1)<len(count2):
        print("Document 2 is more similar w.r.t the query")
        print("DOC 2 occurence frequency:",len(count2))
        print("DOC 1 occurence frequency:",len(count1))
    else:
        print("Both documents are equally similar")
        print("DOC 1 occurence frequency:",len(count1))
        print("DOC 2 occurence frequency:",len(count2))

queryRespond()
```

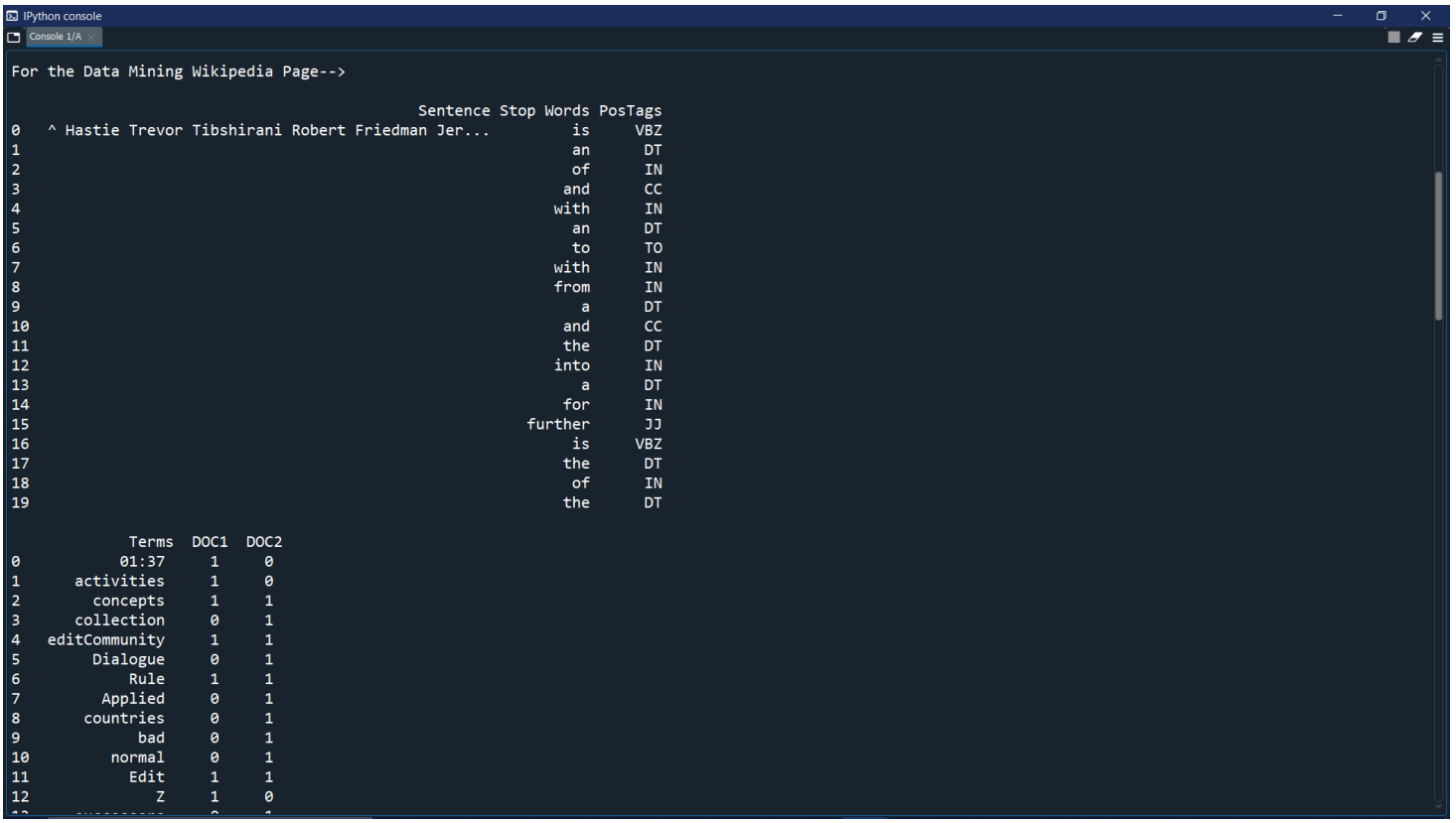

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA\LabDaCode.py
LabDaCode.py
#Question 2
wmTXT = " ";dmTXT = " "
wmTXT = wmTXT.join(wm_filtex)
dmTXT = dmTXT.join(dm_filtex)
WC1 = WordCloud(width=480, height=480, margin=0).generate(wmTXT)
plt.imshow(WC1)
plt.axis("off")
plt.show()
WC2 = WordCloud(width=480, height=480, margin=0).generate(dmTXT)
plt.imshow(WC2)
plt.axis("off")
plt.show()
#Question 3
def tag(sentence):
words = w_t(sentence)
words = pos_tag(words)
return words
def paraphraseable(tag):
return tag.startswith('NN') or tag == 'VB' or tag.startswith('JJ')
def pos(tag):
if tag.startswith('NN'):
return wn.NOUN
elif tag.startswith('V'):
return wn.VERB
def synonyms(word, tag):
lemma_lists = [ss.lemmas() for ss in wn.synsets(word, pos(tag))]
lemmas = [lemma.name() for lemma in sum(lemma_lists, [])]
return set(lemmas)
def synonymIfExists(sentence):
for (word, t) in tag(sentence):
if paraphraseable(t):
syns = synonyms(word, t)
if syns:
if len(syns) > 1:
Name Type Size Value
Variable explorer Help Plots Files Find Code Analysis
Console 1/A
In [77]:
IPython console History
LSP Python: ready
conda (Python 3.8.5)
Line 193, Col 1
UTF-8
CRLF
RW
Mem 85%


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\VIT\1. WINTER SEMESTER (20-21)\Web Mining\LAB\Lab DA\LabDaCode.py
LabDaCode.py
return set(lemmas)
def synonymIfExists(sentence):
for (word, t) in tag(sentence):
if paraphraseable(t):
syns = synonyms(word, t)
if syns:
if len(syns) > 1:
yield [word, list(syns)]
continue
yield [word, []]
def paraphrase(sentence):
return [x for x in synonymIfExists(sentence)]
listOfSentences = ["The quick brown fox jumps over the lazy dog",
"We can rewrite history as much as we like.",
"Once you know all the elements, it's not difficult to pull together a sentence.",
"The incessant ticking and chiming echoed off the weathered walls of the clock repair shop."]
for sentence in listOfSentences:
print(sentence)
print("----------------\n")
paraSent = paraphrase(sentence)
phrase = []
x = []
for i in range(0,3):
x=[]
for i in paraSent:
if(len(i[1])==0):
t = i[0]
else:
t = random.choice(i[1])
x.append(t)
phrase.append(" ".join(map(str,x)))
for i in phrase:
print(i)
print("\n\n")
Name Type Size Value
Variable explorer Help Plots Files Find Code Analysis
Console 1/A
In [77]:
IPython console History
LSP Python: ready
conda (Python 3.8.5)
Line 193, Col 1
UTF-8
CRLF
RW
Mem 83%

Output Screenshots:



```
IPython console
Console 1/A

In [82]: runfile('D:/VIT/1. WINTER SEMESTER (20-21)/Web Mining/LAB/Lab DA/LabDaCode.py', wdir='D:/VIT/1. WINTER SEMESTER (20-21)/Web Mining/LAB/Lab DA')
For the Web Mining Wikipedia Page -->

                                          Sentence Stop Words PosTags
0   Please help to improve this article by introdu...     Please      VB
1                                                         to      TO
2                                                       this      DT
3                                                         by      IN
4                                                       more     JJR
5                                                        how     WRB
6                                                        and      CC
7                                                       when     WRB
8                                                         to      TO
9                                                       this      DT
10                                                      This      DT
11                                                       may      MD
12                                                        to      TO
13                                                        's     POS
14                                                       The      DT
15                                                        is     VBZ
16                                                       The      DT
17                                                       and      CC
18                                                         a      DT
19                                                       has     VBZ


For the Data Mining Wikipedia Page-->

                                          Sentence Stop Words PosTags
0   ^ Hastie Trevor Tibshirani Robert Friedman Jer...         is     VBZ
1                                                         an      DT
2                                                         of      IN
3                                                        and      CC
4                                                       with      IN
5                                                         an      DT
6                                                         to      TO
7                                                       with      IN
8                                                       from      IN
```



```
IPython console
Console 1/A

For the Data Mining Wikipedia Page-->

                                          Sentence Stop Words PosTags
0   ^ Hastie Trevor Tibshirani Robert Friedman Jer...         is     VBZ
1                                                         an      DT
2                                                         of      IN
3                                                        and      CC
4                                                       with      IN
5                                                         an      DT
6                                                         to      TO
7                                                       with      IN
8                                                       from      IN
9                                                          a      DT
10                                                       and      CC
11                                                       the      DT
12                                                      into      IN
13                                                         a      DT
14                                                       for      IN
15                                                   further      JJ
16                                                        is     VBZ
17                                                       the      DT
18                                                        of      IN
19                                                       the      DT

            Terms  DOC1  DOC2
0           01:37     1     0
1      activities     1     0
2        concepts     1     1
3      collection     0     1
4   editCommunity     1     1
5        Dialogue     0     1
6            Rule     1     1
7         Applied     0     1
8       countries     0     1
9             bad     0     1
10         normal     0     1
11           Edit     1     1
12              Z     1     0
```

Boolean Representation: (Top 20 results in table only)

	Terms	DOC1	DOC2
0	01:37	1	0
1	activities	1	0
2	concepts	1	1
3	collection	0	1
4	editCommunity	1	1
5	Dialogue	0	1
6	Rule	1	1
7	Applied	0	1
8	countries	0	1
9	bad	0	1
10	normal	0	1
11	Edit	1	1
12	Z	1	0
13	successors	0	1
14	document	1	0
15	analyzing	1	1
16	demonstrate	1	0
17	Methodology	0	1
18	Ting	1	0
19	ability	1	1

Bag of Words Representation: (Top 20 results in table only)

	Terms	DOC1	DOC2
0	01:37	1	0
1	activities	2	0
2	concepts	1	2
3	collection	0	5
4	editCommunity	1	1
5	Dialogue	0	1
6	Rule	1	1
7	Applied	0	2
8	countries	0	1
9	bad	0	1
10	normal	0	1
11	Edit	1	1
12	Z	1	0
13	successors	0	1
14	document	5	0
15	analyzing	1	4
16	demonstrate	1	0
17	Methodology	0	4
18	Ting	3	0
19	ability	1	1

Complete Representation: (Top 20 results in table only)

dfComplete - DataFrame

Index	Terms	DOC1	DOC2
0	Web	[0, 3, 64, 74, 107, 113, 115, 118, 125, 138, 1…	[2593, 2660, 3602, 4089]
1	mining	[1, 4, 65, 68, 78, 109, 120, 127, 140, 153, 16…	[1, 4, 23, 201, 218, 242, 280, 290, 322, 341, …
2	Wikipedia	[2, 5, 42, 1342, 1345, 2223, 2226]	[2, 5, 4364, 4367]
3	Web	[0, 3, 64, 74, 107, 113, 115, 118, 125, 138, 1…	[2593, 2660, 3602, 4089]
4	mining	[1, 4, 65, 68, 78, 109, 120, 127, 140, 153, 16…	[1, 4, 23, 201, 218, 242, 280, 290, 322, 341, …
5	Wikipedia	[2, 5, 42, 1342, 1345, 2223, 2226]	[2, 5, 4364, 4367]
6	free	[6]	[6]
7	encyclopedia	[7]	[7]
8	Jump	[8, 10]	[8, 10]
9	navigation	[9, 169]	[9]
10	Jump	[8, 10]	[8, 10]
11	search	[11, 970, 1014]	[11, 1742, 2170, 2179, 3304]
12	article	[12, 28, 38, 47, 57]	[603, 1545, 1902, 2598]
13	includes	[13, 409]	[]
14	list	[14, 872]	[2737]
15	references	[15, 51, 226, 1674]	[]
16	related	[16, 464]	[475, 3740]
17	reading	[17]	[570, 3495]
18	external	[18]	[]
19	links	[19, 951, 957, 977, 2174, 2190]	[573, 3735, 4302, 4331]
20	sources	[20, 105, 1004]	[1643]

Document Similarity Result

```
Web Mining is important for Search engine Optimization
Both documents are equally similar
DOC 1 occurence frequency: 4
DOC 2 occurence frequency: 4
```

WordCloud for Doc1:

WordCloud for Doc2:

Synonym Rephrasing of Sentences:

```
The quick brown fox jumps over the lazy dog
---------------

The agile Robert_Brown dodger jumps over the otiose frankfurter
The fast John_Brown Charles_James_Fox jumps over the faineant cad
The speedy Brown Fox jumps over the indolent dog



We can rewrite history as much as we like.
---------------

We can rewrite history as much as we like .
We can rewrite chronicle as much as we like .
We can rewrite account as much as we like .



Once you know all the elements, it’s not difficult to pull together a sentence.
---------------

Once you know all the ingredient , it ’ s not difficult to pull_up together a conviction .
Once you know all the elements , it ’ s not unmanageable to attract together a sentence .
Once you know all the chemical_element , it ’ s not hard to draw together a condemnation .



The incessant ticking and chiming echoed off the weathered walls of the clock repair shop.
---------------

The unremitting ticking and chiming echoed off the brave paries of the clock mend shop_class .
The unremitting tick and chiming echoed off the endure bulwark of the clock repair shop_class .
The perpetual ticking and chiming echoed off the endure paries of the clock haunt store .
```

X—X—X—X—X—X—X—X—X—X—X—X—X—X—X