Department of Computer Science

Cleveland State University

Cleveland, Ohio.

Fall 2020

Lab # 4	
Students Name & ID	Maaz Muhammad Khan 2788572
Due date	3/31/2020
Teacher	Dr. Sunnie S Chung
Subject / Course	Big Data

Contents

Abstract:

This report deals with the text analysis to find the Top N most related documents in a collection per a given user query (topics) in a Question Answering (QA) System, each document can be transformed to be represented as a vector of weights on the topic terms (topic words/keywords/phrases in bi-gram or a tri-gram) in TF-IDF..

Tool:

Python is used as primary language for coding on Spyder IDE.

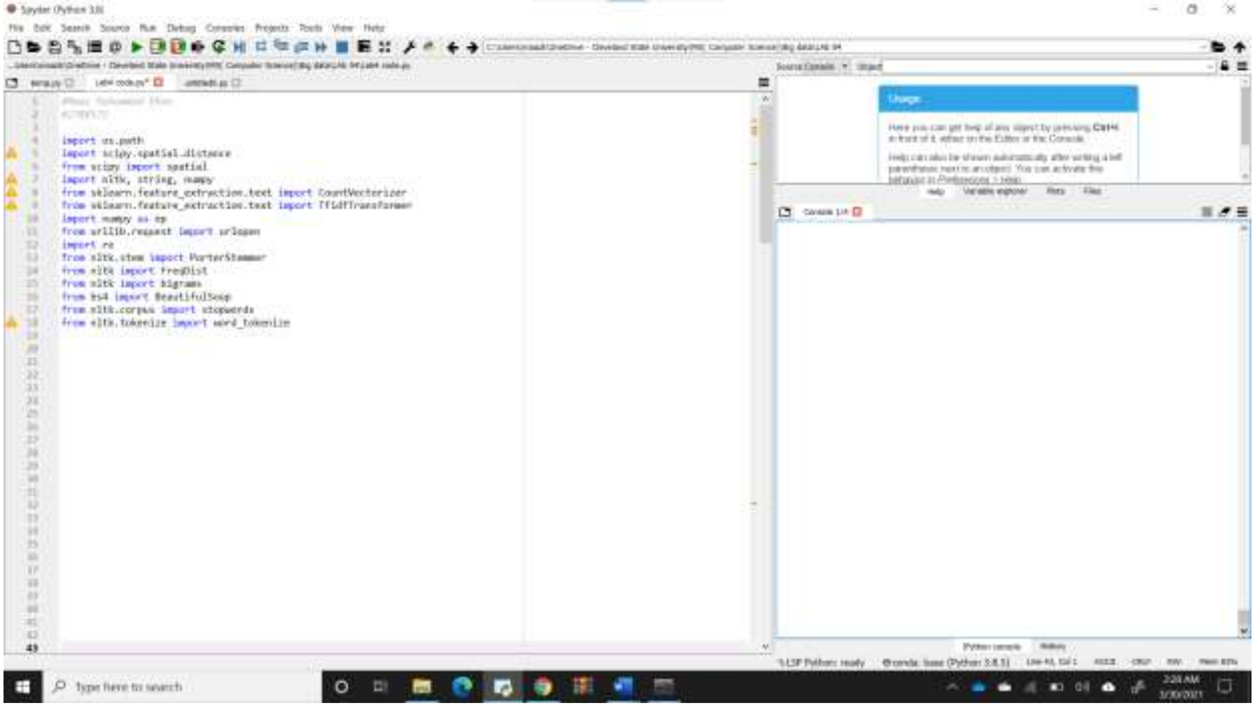
Additional Information:

I was facing error in scrapping text for edx website . I got a permission refusal from edx website that's why I only considered 5 documents to implement my lab 4.

Document 1= "https://en.wikipedia.org/wiki/Engineering",
Document 2= "http://my.clevelandclinic.org/research",
Document 3= "https://en.wikipedia.org/wiki/Data_mining",
Document 4= "https://en.wikipedia.org/wiki/Data_mining#Data_mining",
Document 5= "http://cis.csuohio.edu/~sschung/"

Steps:

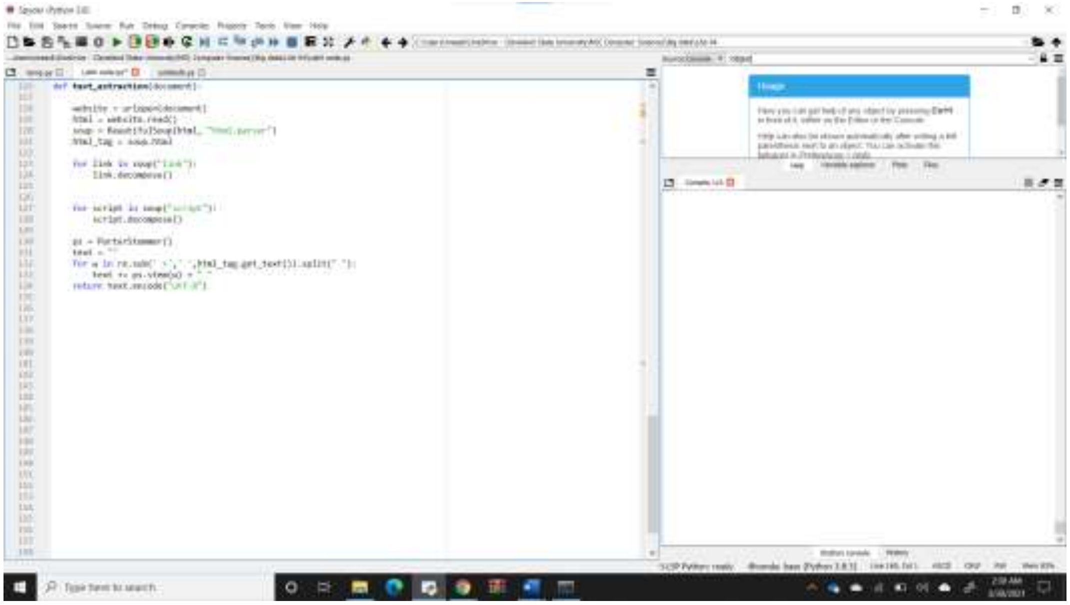
1) Setting up all the libraries and packages for use in project



```
[illegible]
[illegible]

import os.path
import scipy.spatial.distance
from scipy import spatial
import nltk, string, numpy
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import numpy as np
from urllib.request import urlopen
import re
from nltk.stem import PorterStemmer
from nltk import FreqDist
from nltk import bigrams
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

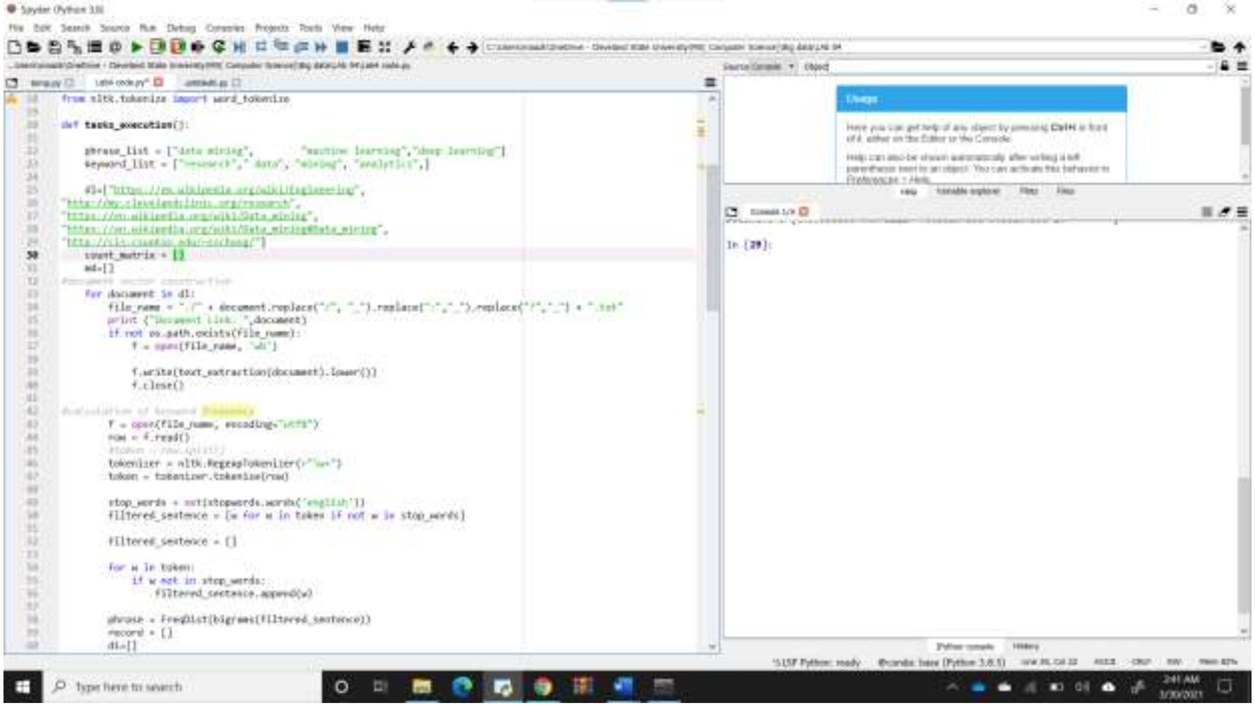
2) Part 1: Preprocessing to Build Document Vectors for Web Page Content Analysis

Then wrote a script for text extraction from website and then applied porter stemming on the text



```python
def text_extraction(document):

    website = urlopen(document)
    html = website.read()
    soup = BeautifulSoup(html, "html.parser")
    html_tag = soup.html

    for link in soup("link"):
        link.decompose()


    for script in soup("script"):
        script.decompose()

    ps = PorterStemmer()
    text = ""
    for w in re.sub(' +',' ',html_tag.get_text()).split(" "):
        text += ps.stem(w) + " "
    return text.encode("utf-8")
```

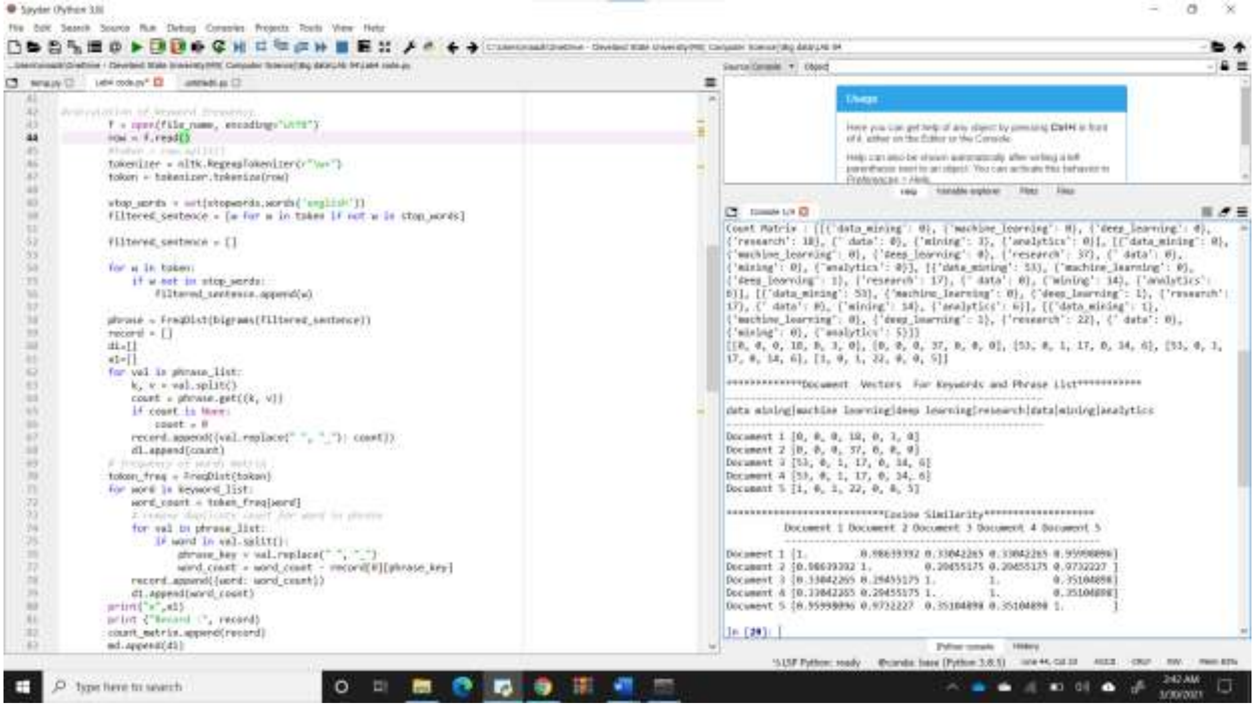
3) Converted text into lower while saving in a text file.
   Removed stop words using NLTK
   Applied RegexpTokenizer.tokenize(text) with RegexpTokenizer , it returns text as a list of words with punctuation's removed.



```python
from nltk.tokenize import word_tokenize

def tasks_execution():

    phrase_list = ["data mining",        "machine learning","deep learning"]
    keyword_list = ["research"," data", "mining", "analytics",]

    dl=["https://en.wikipedia.org/wiki/Engineering",
"http://my.clevelandclinic.org/research",
"https://en.wikipedia.org/wiki/Data_mining",
"https://en.wikipedia.org/wiki/Data_mining#Data_mining",
"http://cis.csuohio.edu/~sschung/"]
    count_matrix = []
    md=[]
#document vector construction
    for document in dl:
        file_name = "./" + document.replace("/", "_").replace(":","_").replace("?","_") + ".txt"
        print ("Document Link: ",document)
        if not os.path.exists(file_name):
            f = open(file_name, 'w')

            f.write(text_extraction(document).lower())
            f.close()

#calculation of keyword frequency
        f = open(file_name, encoding="utf8")
        row = f.read()
        #token = row.split()
        tokenizer = nltk.RegexpTokenizer(r"\w+")
        token = tokenizer.tokenize(row)

        stop_words = set(stopwords.words('english'))
        filtered_sentence = [w for w in token if not w in stop_words]

        filtered_sentence = []

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

        phrase = FreqDist(bigrams(filtered_sentence))
        record = []
        dl=[]
```

```
In [29]:
```

4) Created Data vector of keyword_list and phrase_List for every document.



```
#calculation of Keyword Frequency
        f = open(file_name, encoding="utf8")
        row = f.read()
        #token = row.split()
        tokenizer = nltk.RegexpTokenizer(r"\w+")
        token = tokenizer.tokenize(row)

        stop_words = set(stopwords.words('english'))
        filtered_sentence = [w for w in token if not w in stop_words]

        filtered_sentence = []

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

        phrase = FreqDist(bigrams(filtered_sentence))
        record = []
        d1=[]
        x1=[]
        for val in phrase_list:
            k, v = val.split()
            count = phrase.get((k, v))
            if count is None:
                count = 0
            record.append({val.replace(" ", "_"): count})
            d1.append(count)
        # frequency of word matrix
        token_freq = FreqDist(token)
        for word in keyword_list:
            word_count = token_freq[word]
            # remove duplicate count for word in phrase
            for val in phrase_list:
                if word in val.split():
                    phrase_key = val.replace(" ", "_")
                    word_count = word_count - record[0][phrase_key]
            record.append({word: word_count})
            d1.append(word_count)
        print("x",x1)
        print ("Record :", record)
        count_matrix.append(record)
        md.append(d1)
```

```
Count Matrix : [[{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0},
{'research': 18}, {' data': 0}, {'mining': 3}, {'analytics': 0}], [{'data_mining': 0},
{'machine_learning': 0}, {'deep_learning': 0}, {'research': 37}, {' data': 0},
{'mining': 0}, {'analytics': 0}], [{'data_mining': 53}, {'machine_learning': 0},
{'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics':
6}], [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research':
17}, {' data': 0}, {'mining': 14}, {'analytics': 6}], [{'data_mining': 1},
{'machine_learning': 0}, {'deep_learning': 1}, {'research': 22}, {' data': 0},
{'mining': 0}, {'analytics': 5}]]
[[0, 0, 0, 18, 0, 3, 0], [0, 0, 0, 37, 0, 0, 0], [53, 0, 1, 17, 0, 14, 6], [53, 0, 1,
17, 0, 14, 6], [1, 0, 1, 22, 0, 0, 5]]

*************Document  Vectors  For Keywords and Phrase List***********
--------------------------------------------------------------------
data mining|machine learning|deep learning|research|data|mining|analytics
--------------------------------------------------------------------
Document 1 [0, 0, 0, 18, 0, 3, 0]
Document 2 [0, 0, 0, 37, 0, 0, 0]
Document 3 [53, 0, 1, 17, 0, 14, 6]
Document 4 [53, 0, 1, 17, 0, 14, 6]
Document 5 [1, 0, 1, 22, 0, 0, 5]

***************************Cosine Similarity*******************
           Document 1 Document 2 Document 3 Document 4 Document 5
           ----------------------------------------------------------
Document 1 [1.         0.98639392 0.33042265 0.33042265 0.95998896]
Document 2 [0.98639392 1.         0.29455175 0.29455175 0.9732227 ]
Document 3 [0.33042265 0.29455175 1.         1.         0.35104898]
Document 4 [0.33042265 0.29455175 1.         1.         0.35104898]
Document 5 [0.95998896 0.9732227  0.35104898 0.35104898 1.        ]

In [28]: 
```

Document Link: https://en.wikipedia.org/wiki/Engineering

x []

Record : [{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0}, {'research': 18}, {' data': 0}, {'mining': 3}, {'analytics': 0}]

Document Link: http://my.clevelandclinic.org/research

x []

Record : [{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0}, {'research': 37}, {' data': 0}, {'mining': 0}, {'analytics': 0}]

Document Link: https://en.wikipedia.org/wiki/Data_mining

x []

Record : [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics': 6}]

Document Link: https://en.wikipedia.org/wiki/Data_mining#Data_mining

x []

Record : [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics': 6}]

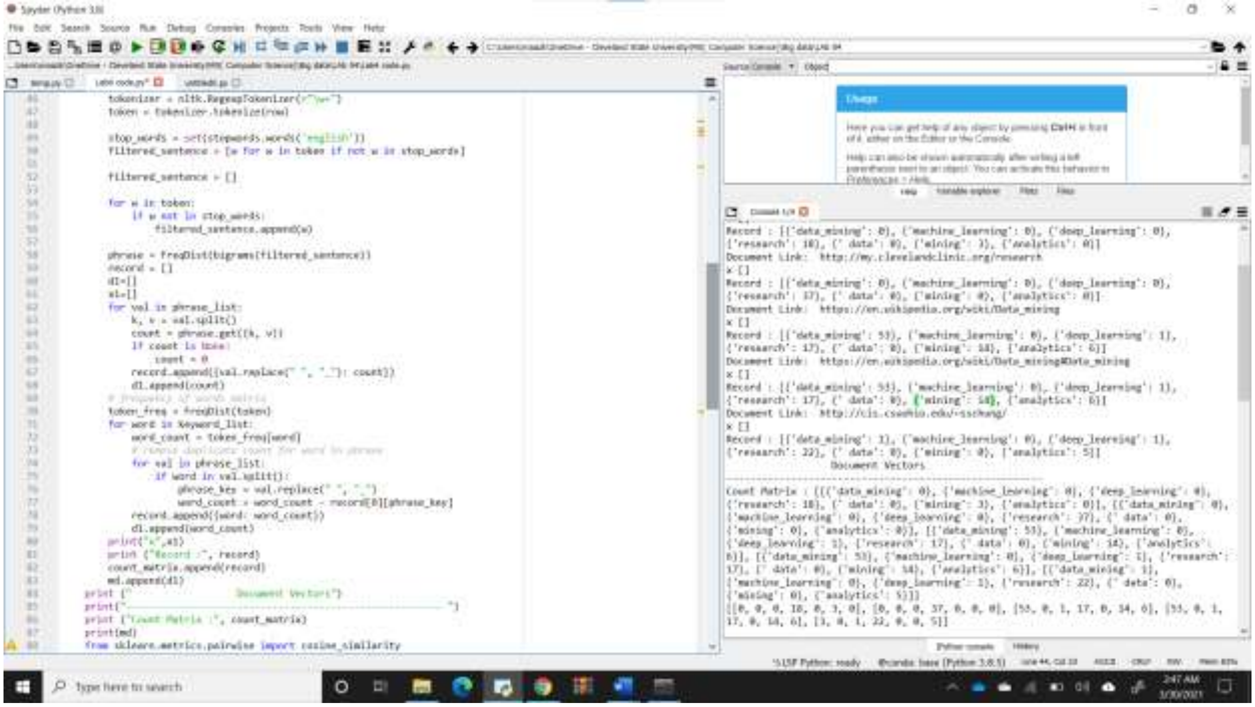
Document Link: http://cis.csuohio.edu/~sschung/

x []

Record : [{'data_mining': 1}, {'machine_learning': 0}, {'deep_learning': 1}, {'research': 22}, {' data': 0}, {'mining': 0}, {'analytics': 5}]

Document Vectors

5) Then creating count matrix



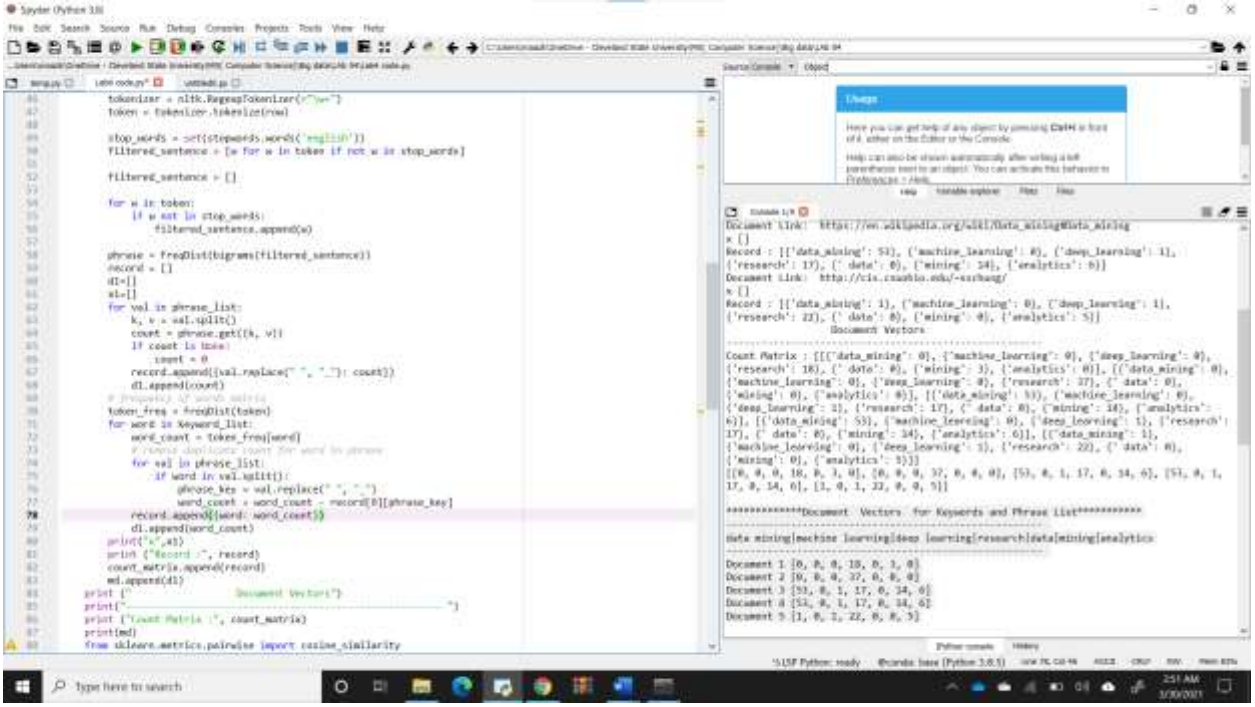
```
        tokenizer = nltk.RegexpTokenizer(r"\w+")
        token = tokenizer.tokenize(row)

        stop_words = set(stopwords.words('english'))
        filtered_sentence = [w for w in token if not w in stop_words]

        filtered_sentence = []

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

        phrase = FreqDist(bigrams(filtered_sentence))
        record = []
        d1=[]
        a1=[]
        for val in phrase_list:
            k, v = val.split()
            count = phrase.get((k, v))
            if count is None:
                count = 0
            record.append({val.replace(" ", "_"): count})
            d1.append(count)
        # frequency of words matrix
        token_freq = FreqDist(token)
        for word in Keyword_list:
            word_count = token_freq[word]
            # remove duplicate count for word in phrase
            for val in phrase_list:
                if word in val.split():
                    phrase_key = val.replace(" ", "_")
                    word_count = word_count - record[0][phrase_key]
            record.append({word: word_count})
            d1.append(word_count)
        print("x",a1)
        print ("Record :", record)
        count_matrix.append(record)
        md.append(d1)
    print ("                    Document Vectors")
    print("------------------------------------------------------------")
    print ("Count Matrix :", count_matrix)
    print(md)
    from sklearn.metrics.pairwise import cosine_similarity
```

```
Record : [{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0},
{'research': 18}, {' data': 0}, {'mining': 3}, {'analytics': 0}]
Document Link:  http://my.clevelandclinic.org/research
x []
Record : [{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0},
{'research': 37}, {' data': 0}, {'mining': 0}, {'analytics': 0}]
Document Link:  https://en.wikipedia.org/wiki/Data_mining
x []
Record : [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1},
{'research': 17}, {' data': 0}, {'mining': 14}, {'analytics': 6}]
Document Link:  https://en.wikipedia.org/wiki/Data_mining#Data_mining
x []
Record : [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1},
{'research': 17}, {' data': 0}, {'mining': 14}, {'analytics': 6}]
Document Link:  http://cis.csuohio.edu/~sschung/
x []
Record : [{'data_mining': 1}, {'machine_learning': 0}, {'deep_learning': 1},
{'research': 22}, {' data': 0}, {'mining': 0}, {'analytics': 5}]
                Document Vectors
------------------------------------------------------------
Count Matrix : [[{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0},
{'research': 18}, {' data': 0}, {'mining': 3}, {'analytics': 0}], [{'data_mining': 0},
{'machine_learning': 0}, {'deep_learning': 0}, {'research': 37}, {' data': 0},
{'mining': 0}, {'analytics': 0}], [{'data_mining': 53}, {'machine_learning': 0},
{'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics':
6}], [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research':
17}, {' data': 0}, {'mining': 14}, {'analytics': 6}], [{'data_mining': 1},
{'machine_learning': 0}, {'deep_learning': 1}, {'research': 22}, {' data': 0},
{'mining': 0}, {'analytics': 5}]]
[[0, 0, 0, 18, 0, 3, 0], [0, 0, 0, 37, 0, 0, 0], [53, 0, 1, 17, 0, 14, 6], [53, 0, 1,
17, 0, 14, 6], [1, 0, 1, 22, 0, 0, 5]]
```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
temp.py
Lab4 code.py*
webtext.py
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
Help
Variable explorer
Plots
Files
Console 1/A
Python console
History
LSP Python: ready
conda: base (Python 3.8.5)
Type here to search
2:47 AM
3/30/2021

Count Matrix : [[{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0}, {'research': 18}, {' data': 0}, {'mining': 3}, {'analytics': 0}], [{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0}, {'research': 37}, {' data': 0}, {'mining': 0}, {'analytics': 0}], [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics': 6}], [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics': 6}], [{'data_mining': 1}, {'machine_learning': 0}, {'deep_learning': 1}, {'research': 22}, {' data': 0}, {'mining': 0}, {'analytics': 5}]

6) Calculation of Data vector of keyword_list and phrase_List and created a matrix .

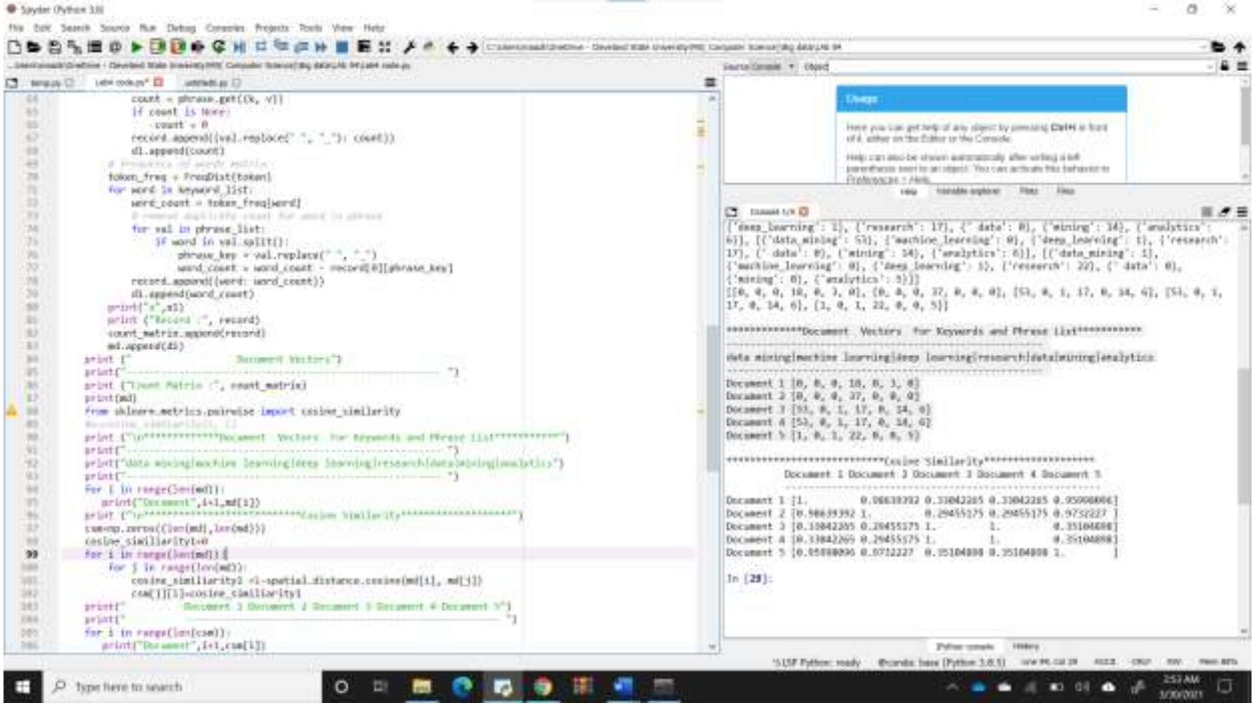

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
temp.py
Lab4 code.py*
untitled0.py
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
Help
Variable explorer
Plots
Files
Console 1/A
Python console
History
LSP Python: ready
conda: base (Python 3.8.5)
Type here to search
2:51 AM
1/30/2021

```
        tokenizer = nltk.RegexpTokenizer(r"\w+")
        token = tokenizer.tokenize(row)

        stop_words = set(stopwords.words('english'))
        filtered_sentence = [w for w in token if not w in stop_words]

        filtered_sentence = []

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

        phrase = FreqDist(bigrams(filtered_sentence))
        record = []
        dl=[]
        xl=[]
        for val in phrase_list:
            k, v = val.split()
            count = phrase.get((k, v))
            if count is None:
                count = 0
            record.append({val.replace(" ", "_"): count})
            dl.append(count)
        # frequency of words matrix
        token_freq = FreqDist(token)
        for word in Keyword_list:
            word_count = token_freq[word]
            # remove duplicate count for word in phrase
            for val in phrase_list:
                if word in val.split():
                    phrase_key = val.replace(" ", "_")
                    word_count = word_count - record[0][phrase_key]
            record.append({word: word_count})
            dl.append(word_count)
        print("x",xl)
        print ("Record :", record)
        count_matrix.append(record)
        md.append(dl)
    print ("                    Document Vectors")
    print("-------------------------------------------------------------")
    print ("Count Matrix :", count_matrix)
    print(md)
    from sklearn.metrics.pairwise import cosine_similarity
```

```
Document Link:  https://en.wikipedia.org/wiki/Data_mining#Data_mining
x []
Record : [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1},
{'research': 17}, {' data': 0}, {'mining': 14}, {'analytics': 6}]
Document Link:  http://cis.csuohio.edu/~sschung/
x []
Record : [{'data_mining': 1}, {'machine_learning': 0}, {'deep_learning': 1},
{'research': 22}, {' data': 0}, {'mining': 0}, {'analytics': 5}]
                    Document Vectors
-------------------------------------------------------------
Count Matrix : [[{'data_mining': 0}, {'machine_learning': 0}, {'deep_learning': 0},
{'research': 18}, {' data': 0}, {'mining': 3}, {'analytics': 0}], [{'data_mining': 0},
{'machine_learning': 0}, {'deep_learning': 0}, {'research': 37}, {' data': 0},
{'mining': 0}, {'analytics': 0}], [{'data_mining': 53}, {'machine_learning': 0},
{'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics':
6}], [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research':
17}, {' data': 0}, {'mining': 14}, {'analytics': 6}], [{'data_mining': 1},
{'machine_learning': 0}, {'deep_learning': 1}, {'research': 22}, {' data': 0},
{'mining': 0}, {'analytics': 5}]]
[[0, 0, 0, 18, 0, 3, 0], [0, 0, 0, 37, 0, 0, 0], [53, 0, 1, 17, 0, 14, 6], [53, 0, 1,
17, 0, 14, 6], [1, 0, 1, 22, 0, 0, 5]]

*************Document  Vectors  For Keywords and Phrase List***********
-------------------------------------------------------------
data mining|machine learning|deep learning|research|data|mining|analytics
-------------------------------------------------------------
Document 1 [0, 0, 0, 18, 0, 3, 0]
Document 2 [0, 0, 0, 37, 0, 0, 0]
Document 3 [53, 0, 1, 17, 0, 14, 6]
Document 4 [53, 0, 1, 17, 0, 14, 6]
Document 5 [1, 0, 1, 22, 0, 0, 5]
```

```
*************Document  Vectors  For Keywords and Phrase List***********
-----------------------------------------------------
data mining|machine learning|deep learning|research|data|mining|analytics
-----------------------------------------------------
Document 1 [0, 0, 0, 18, 0, 3, 0]
Document 2 [0, 0, 0, 37, 0, 0, 0]
Document 3 [53, 0, 1, 17, 0, 14, 6]
Document 4 [53, 0, 1, 17, 0, 14, 6]
Document 5 [1, 0, 1, 22, 0, 0, 5]
```

7) Then calculated cosine similarity using scipy built-in method.



```
            count = phrase.get((k, v))
            if count is None:
                count = 0
            record.append({val.replace(" ", "_"): count})
            d1.append(count)
        # frequency of words matrix
        token_freq = FreqDist(token)
        for word in keyword_list:
            word_count = token_freq[word]
            # remove duplicate count for word in phrase
            for val in phrase_list:
                if word in val.split():
                    phrase_key = val.replace(" ", "_")
                    word_count = word_count - record[0][phrase_key]
            record.append({word: word_count})
            d1.append(word_count)
        print("x",x1)
        print ("Record :", record)
        count_matrix.append(record)
        md.append(d1)
    print ("                    Document Vectors")
    print("--------------------------------------------------")
    print ("Count Matrix :", count_matrix)
    print(md)
    from sklearn.metrics.pairwise import cosine_similarity
    #cosine_similarity(d1, [])
    print ("\n************Document  Vectors  for Keywords and Phrase List***********")
    print("--------------------------------------------------")
    print("data mining|machine learning|deep learning|research|data|mining|analytics")
    print("--------------------------------------------------")
    for i in range(len(md)):
      print("Document",i+1,md[i])
    print ("\n**************************Cosine Similarity******************")
    csm=np.zeros((len(md),len(md)))
    cosine_similiarity1=0
    for i in range(len(md)):
        for j in range(len(md)):
            cosine_similiarity1 =1-spatial.distance.cosine(md[i], md[j])
            csm[i][j]=cosine_similiarity1
    print("          Document 1 Document 2 Document 3 Document 4 Document 5")
    print("          --------------------------------------------------")
    for i in range(len(csm)):
      print("Document",i+1,csm[i])
```

```
{'deep_learning': 1}, {'research': 17}, {' data': 0}, {'mining': 14}, {'analytics':
6}], [{'data_mining': 53}, {'machine_learning': 0}, {'deep_learning': 1}, {'research':
17}, {' data': 0}, {'mining': 14}, {'analytics': 6}], [{'data_mining': 1},
{'machine_learning': 0}, {'deep_learning': 1}, {'research': 22}, {' data': 0},
{'mining': 0}, {'analytics': 5}]]
[[0, 0, 0, 18, 0, 3, 0], [0, 0, 0, 37, 0, 0, 0], [53, 0, 1, 17, 0, 14, 6], [53, 0, 1,
17, 0, 14, 6], [1, 0, 1, 22, 0, 0, 5]]

************Document  Vectors  for Keywords and Phrase List***********
-----------------------------------------------------------
data mining|machine learning|deep learning|research|data|mining|analytics
-----------------------------------------------------------
Document 1 [0, 0, 0, 18, 0, 3, 0]
Document 2 [0, 0, 0, 37, 0, 0, 0]
Document 3 [53, 0, 1, 17, 0, 14, 6]
Document 4 [53, 0, 1, 17, 0, 14, 6]
Document 5 [1, 0, 1, 22, 0, 0, 5]

**************************Cosine Similarity******************
          Document 1 Document 2 Document 3 Document 4 Document 5
          -----------------------------------------------------
Document 1 [1.         0.98639392 0.33042265 0.33042265 0.95998096]
Document 2 [0.98639392 1.         0.29455175 0.29455175 0.9732227 ]
Document 3 [0.33042265 0.29455175 1.         1.         0.35104898]
Document 4 [0.33042265 0.29455175 1.         1.         0.35104898]
Document 5 [0.95998096 0.9732227  0.35104898 0.35104898 1.        ]

In [29]:
```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
temp.py
Lab4 code.py*
untitled0.py
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
Help
Variable explorer
Plots
Files
Console 1/A
Python console
History
LSP Python: ready
conda: base (Python 3.8.5)
Type here to search
2:53 AM

```
***************************Cosine Similarity*******************

    ***************************Cosine Similarity*******************

    Document 1 Document 2 Document 3 Document 4 Document 5

    -----------------------------------------------------

Document 1 [1.         0.98639392 0.33042265 0.33042265 0.95998096]

Document 2 [0.98639392 1.         0.29455175 0.29455175 0.9732227 ]

Document 3 [0.33042265 0.29455175 1.         1.         0.35104898]

Document 4 [0.33042265 0.29455175 1.         1.         0.35104898]

Document 5 [0.95998096 0.9732227  0.35104898 0.35104898 1.        ]
```

8) Part3 : Analysis and discussion.

Discussion briefly about your topic analysis with your cosine similarity matrix focusing on that:
Whether each value (in Cosine Sim) of each pair of any two docs indicate the similarity correctly?

Cosine similarity comes out 1 for a documen when calculated with itself.

Which 2 docs are most similar in terms of 7 given topics?
Document 4 and 5 are most similar.

The Topics of Doc6 is similar to the Topics of Doc 4 and 5?
Explain Why or Why Not in terms of 7 TFs? If not, what are the reasons?

Because their content is all about data mining and have similar words.

Their word count and phrases count of topic words is same.

References:

Wikipedia

Microsoft.com

http://eecs.csuohio.edu/~sschung/CIS660/CIS660F20.html#Lab