:[17] In

```
import nltk #need for dealing with text
import os # need for looping through folders
import string
import numpy as np
import copy
import pandas as pd
import pickle
import re
```

:[18] In

```
pip install num2words
```

```
Requirement already satisfied: num2words in c:\users\acer\anaconda3\lib\site
(-packages (0.5.10
Requirement already satisfied: docopt>=0.6.2 in c:\users\acer\anaconda3\lib
(\site-packages (from num2words) (0.6.2
.Note: you may need to restart the kernel to use updated packages
```

:[19] In

```
title = "used files"
os.chdir(r'C:\mini_newsgroups\comp.graphics')
paths = []
for (dirpath, dirnames, filenames) in os.walk(str(os.getcwd())+'/'+title+'/'):
    for i in filenames:
        paths.append(str(dirpath)+str("\\")+i)

paths
```

Out[19]:

```
,'C:\\mini_newsgroups\\comp.graphics/used files/\\37916']
,'C:\\mini_newsgroups\\comp.graphics/used files/\\37921'
,'C:\\mini_newsgroups\\comp.graphics/used files/\\37930'
['C:\\mini_newsgroups\\comp.graphics/used files/\\37936'
```

:[20] In

```
for i in paths:
    file = open(paths[paths.index(i)])
    print(file.read(),'\n\n-------------------------\n\n')
```

```
Path: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs7.ece.cmu.edu!e
uropa.eng.gtefsd.com!gatech!asuvax!cs.utexas.edu!zaphod.mps.ohio-state.ed
u!saimiri.primate.wisc.edu!usenet.coe.montana.edu!news.u.washington.edu!uw
-beaver!cs.ubc.ca!unixg.ubc.ca!kakwa.ucs.ualberta.ca!ersys!joth
(From: joth@ersys.edmonton.ab.ca (Joe Tham
Newsgroups: comp.graphics
?Subject: Where can I find SIPP
<Message-ID: <yFXJ2B2w165w@ersys.edmonton.ab.ca
Date: Mon, 05 Apr 93 14:58:21 MDT
Organization: Edmonton Remote Systems #2, Edmonton, AB, Canada
Lines: 11

 I recently got a file describing a library of rendering routines
 called SIPP (SImple Polygon Processor).  Could anyone tell me where I can
?FTP the source code and which is the newest version around
 Also, I've never used Renderman so I was wondering if Renderman
 is like SIPP?  ie. a library of rendering routines which one uses to make
...a program that creates the image

Th
```

:[21] In

```
#Removing stop words
def remove_stop_words(data):
    stop_words = stopwords.words('english')
    words = word_tokenize(str(data))
    new_text = ""
    for w in words:
        if w not in stop_words:
            new_text = new_text + " " + w
    return np.char.strip(new_text)

#Removing punctuation
def remove_punctuation(data):
    symbols = "!\"#$%&()*+-./:;<=>?@[\]^_`{|}~\n"
    for i in range(len(symbols)):
        data = np.char.replace(data, symbols[i], ' ')
        data = np.char.replace(data, "  ", " ")
    data = np.char.replace(data, ',', '')
    return data

#Convert to lowercase
def convert_lower_case(data):
    return np.char.lower(data)

#Stemming
def stemming(data):
    stemmer= PorterStemmer()

    tokens = word_tokenize(str(data))
    new_text = ""
    for w in tokens:
        new_text = new_text + " " + stemmer.stem(w)
    return np.char.strip(new_text)

#Converting numbers to its equivalent words
def convert_numbers(data):
    data = np.char.replace(data, "0", " zero ")
    data = np.char.replace(data, "1", " one ")
    data = np.char.replace(data, "2", " two ")
    data = np.char.replace(data, "3", " three ")
    data = np.char.replace(data, "4", " four ")
    data = np.char.replace(data, "5", " five ")
    data = np.char.replace(data, "6", " six ")
    data = np.char.replace(data, "7", " seven ")
    data = np.char.replace(data, "8", " eight ")
    data = np.char.replace(data, "9", " nine ")
    return data

#Removing header
def remove_header(data):
    try:
        ind = data.index('\n\n')
        data = data[ind:]
    except:
        print("No Header")
    return data

#Removing apostrophe
def remove_apostrophe(data):
    return np.char.replace(data, "'", "")
```

```
#Removing single characters
def remove_single_characters(data):
    words = word_tokenize(str(data))
    new_text = ""
    for w in words:
        if len(w) > 1:
            new_text = new_text + " " + w
    return np.char.strip(new_text)

#to use all the previous
def preprocess(data):
    data = remove_header(data)
    data = convert_lower_case(data)
    data = convert_numbers(data)
    data = remove_punctuation(data)
    data = remove_stop_words(data)
    data = remove_apostrophe(data)
    data = remove_single_characters(data)
    data = stemming(data)
    return data
```

:[22] In

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from collections import Counter
from num2words import num2words
```

:[23] In

```
processed_text = []
for i in range(len(filenames)):
    file = open(dirpath+'/'+ filenames[i], 'r', encoding='cp1250', errors='ignore')
    text = file.read().strip()
    file.close()
    processed_text.append(word_tokenize(str(preprocess(text))))
    print(processed_text)
```

```
 recent', 'got', 'file', 'describ', 'librari', 'render', 'routin', 'call',']]
'sipp', 'simpl', 'polygon', 'processor', 'could', 'anyon', 'tell', 'ftp', 's
 ourc', 'code', 'newest', 'version', 'around', 'also', 've', 'never', 'use',
'renderman', 'wonder', 'renderman', 'like', 'sipp', 'ie', 'librari', 'rende
r', 'routin', 'one', 'use', 'make', 'program', 'creat', 'imag', 'thank', 'jo
[['e', 'tham', 'joe', 'tham', 'joth', 'ersi', 'edmonton', 'ab', 'ca
 recent', 'got', 'file', 'describ', 'librari', 'render', 'routin', 'call',']]
'sipp', 'simpl', 'polygon', 'processor', 'could', 'anyon', 'tell', 'ftp', 's
 ourc', 'code', 'newest', 'version', 'around', 'also', 've', 'never', 'use',
'renderman', 'wonder', 'renderman', 'like', 'sipp', 'ie', 'librari', 'rende
r', 'routin', 'one', 'use', 'make', 'program', 'creat', 'imag', 'thank', 'jo
e', 'tham', 'joe', 'tham', 'joth', 'ersi', 'edmonton', 'ab', 'ca'], ['hell
 o', 'everybodi', 'use', 'pixar', 'renderman', 'three', 'scene', 'descript',
 'languag', 'creat', 'three', 'world', 'pleas', 'help', 'use', 'renderman',
 'librari', 'next', 'document', 'nextstep', 'version', 'renderman', 'avail',
 'creat', 'complic', 'scene', 'render', 'use', 'surfac', 'shader', 'bring',
'life', 'appli', 'shadow', 'reflect', 'far', 'understand', 'defin', 'environ
ment', 'shadow', 'map', 'produc', 'reflect', 'shadow', 'know', 'use', 'advi
s', 'simpl', 'rib', 'exampl', 'appreci', 'thank', 'advanc', 'alex', 'koleso
 v', 'moscow', 'russia', 'talu', 'imag', 'commun', 'corpor', 'mail', 'alex',
[['talu', 'msk', 'su', 'next', 'mail', 'accept
 recent', 'got', 'file', 'describ', 'librari', 'render', 'routin', 'call',']]
'sipp', 'simpl', 'polygon', 'processor', 'could', 'anyon', 'tell', 'ftp', 's
 ourc', 'code', 'newest', 'version', 'around', 'also', 've', 'never', 'use',
'renderman', 'wonder', 'renderman', 'like', 'sipp', 'ie', 'librari', 'rende
r', 'routin', 'one', 'use', 'make', 'program', 'creat', 'imag', 'thank', 'jo
e', 'tham', 'joe', 'tham', 'joth', 'ersi', 'edmonton', 'ab', 'ca'], ['hell
 o', 'everybodi', 'use', 'pixar', 'renderman', 'three', 'scene', 'descript',
 'languag', 'creat', 'three', 'world', 'pleas', 'help', 'use', 'renderman',
 'librari', 'next', 'document', 'nextstep', 'version', 'renderman', 'avail',
 'creat', 'complic', 'scene', 'render', 'use', 'surfac', 'shader', 'bring',
'life', 'appli', 'shadow', 'reflect', 'far', 'understand', 'defin', 'environ
ment', 'shadow', 'map', 'produc', 'reflect', 'shadow', 'know', 'use', 'advi
s', 'simpl', 'rib', 'exampl', 'appreci', 'thank', 'advanc', 'alex', 'koleso
 v', 'moscow', 'russia', 'talu', 'imag', 'commun', 'corpor', 'mail', 'alex',
 'talu', 'msk', 'su', 'next', 'mail', 'accept'], ['anybodi', 'know', 'good',
'two', 'graphic', 'packag', 'avail', 'ibm', 'rs', 'six', 'zero', 'zero', 'ze
ro', 'aix', 'look', 'someth', 'like', 'dec', 'gk', 'hewlett', 'packard', 'st
arbas', 'reason', 'good', 'support', 'differ', 'output', 'devic', 'like', 'p
 lotter', 'termin', 'etc', 'tri', 'also', 'xgk', 'one', 'one', 'distribut',
 'ibm', 'implement', 'phig', 'work', 'requir', 'output', 'devic', 'window',
 'salesman', 'ibm', 'familiar', 'graphic', 'expect', 'good', 'solut', 'ari',
 'ari', 'suutari', 'ari', 'carel', 'fi', 'carelcomp', 'oy', 'lappeenranta',
[['finland
 recent', 'got', 'file', 'describ', 'librari', 'render', 'routin', 'call',']]
'sipp', 'simpl', 'polygon', 'processor', 'could', 'anyon', 'tell', 'ftp', 's
 ourc', 'code', 'newest', 'version', 'around', 'also', 've', 'never', 'use',
'renderman', 'wonder', 'renderman', 'like', 'sipp', 'ie', 'librari', 'rende
r', 'routin', 'one', 'use', 'make', 'program', 'creat', 'imag', 'thank', 'jo
e', 'tham', 'joe', 'tham', 'joth', 'ersi', 'edmonton', 'ab', 'ca'], ['hell
 o', 'everybodi', 'use', 'pixar', 'renderman', 'three', 'scene', 'descript',
```

```
 'languag', 'creat', 'three', 'world', 'pleas', 'help', 'use', 'renderman',
 'librari', 'next', 'document', 'nextstep', 'version', 'renderman', 'avail',
 'creat', 'complic', 'scene', 'render', 'use', 'surfac', 'shader', 'bring',
'life', 'appli', 'shadow', 'reflect', 'far', 'understand', 'defin', 'environ
ment', 'shadow', 'map', 'produc', 'reflect', 'shadow', 'know', 'use', 'advi
s', 'simpl', 'rib', 'exampl', 'appreci', 'thank', 'advanc', 'alex', 'koleso
 v', 'moscow', 'russia', 'talu', 'imag', 'commun', 'corpor', 'mail', 'alex',
 'talu', 'msk', 'su', 'next', 'mail', 'accept'], ['anybodi', 'know', 'good',
'two', 'graphic', 'packag', 'avail', 'ibm', 'rs', 'six', 'zero', 'zero', 'ze
ro', 'aix', 'look', 'someth', 'like', 'dec', 'gk', 'hewlett', 'packard', 'st
arbas', 'reason', 'good', 'support', 'differ', 'output', 'devic', 'like', 'p
 lotter', 'termin', 'etc', 'tri', 'also', 'xgk', 'one', 'one', 'distribut',
 'ibm', 'implement', 'phig', 'work', 'requir', 'output', 'devic', 'window',
 'salesman', 'ibm', 'familiar', 'graphic', 'expect', 'good', 'solut', 'ari',
 'ari', 'suutari', 'ari', 'carel', 'fi', 'carelcomp', 'oy', 'lappeenranta',
 'finland'], ['requir', 'bgi', 'driver', 'super', 'vga', 'display', 'super',
'xvga', 'display', 'anyon', 'know', 'could', 'obtain', 'relev', 'driver', 'f
[['tp', 'site', 'regard', 'simon', 'crow
```

:[24] In

```
DF = {}
N = len(processed_text)

for i in range(N):
    tokens = processed_text[i]
    for w in tokens:
        try:
            DF[w].add(i)
        except:
            DF[w] = {i}

for i in DF:
    DF[i] = len(DF[i])

DF
```

Out[24]:

```
            ,recent': 1'}
            ,got': 1'
            ,file': 1'
            ,describ': 1'
            ,librari': 2'
            ,render': 2'
            ,routin': 1'
            ,call': 1'
            ,sipp': 1'
            ,simpl': 2'
            ,polygon': 1'
            ,processor': 1'
            ,could': 2'
            ,anyon': 2'
            ,tell': 1'
            ,ftp': 2'
            ,sourc': 1'
            ,code': 1'
```

:[25] In

```
total_vocab_size = [x for x in DF]
print(total_vocab_size[:5])
```

```
['recent', 'got', 'file', 'describ', 'librari']
```

:[26] In

```
print(len(DF))
```

```
144
```

:[27] In

```
def doc_freq(word):
    c = 0
    try:
        c = DF[word]
    except:
        pass
    return c
```

:[28] In

```
word = 'one'
print('word = ',word,'--> frequency = ', doc_freq(word))
```

```
word =  one --> frequency =  2
```

:[29] In

```
doc = 0
tf_idf = {}

for i in range(N):

    tokens = processed_text[i]
    counter = Counter(tokens + processed_text[i])
    words_count = len(tokens + processed_text[i])

    for token in np.unique(tokens):

        tf = counter[token]/words_count
        df = doc_freq(token)
        idf = np.log((N+1)/(df+1))

        tf_idf[doc, token] = tf*idf

    doc += 1

tf_idf
```

Out[29]:

```
             ,ab'): 0.018325814637483104' ,0)}
             ,also'): 0.010216512475319815' ,0)
             ,anyon'): 0.010216512475319815' ,0)
             ,around'): 0.018325814637483104' ,0)
             ,ca'): 0.018325814637483104' ,0)
             ,call'): 0.018325814637483104' ,0)
             ,code'): 0.018325814637483104' ,0)
             ,could'): 0.010216512475319815' ,0)
             ,creat'): 0.010216512475319815' ,0)
             ,describ'): 0.018325814637483104' ,0)
             ,edmonton'): 0.018325814637483104' ,0)
             ,ersi'): 0.018325814637483104' ,0)
             ,file'): 0.018325814637483104' ,0)
             ,ftp'): 0.010216512475319815' ,0)
             ,got'): 0.018325814637483104' ,0)
             ,ie'): 0.018325814637483104' ,0)
             ,imag'): 0.010216512475319815' ,0)
             ,joe'): 0.03665162927496621' ,0)
```

:[30] In

```
dict_items = tf_idf.items()

print(list(dict_items)[:1])
```

```
            [(ab'), 0.018325814637483104' ,0))]
```

:[31] In

```
def matching_score(k, query):
    preprocessed_query = preprocess(query)
    tokens = word_tokenize(str(preprocessed_query))
    print("Matching Score")
    print("\nQuery:", query)
    print("")
    print(tokens)

    query_weights = {}
    for key in tf_idf:
        if key[1] in tokens:
            try:
                query_weights[key[0]] += tf_idf[key]
            except:
                query_weights[key[0]] = tf_idf[key]

    query_weights = sorted(query_weights.items(), key=lambda x: x[1], reverse=True)
    print("")

    l = []

    for i in query_weights[:k]:
        l.append(i[0])

    print(l)

matching_score(2, "I recently got a file describing a library")
```

```
No Header
Matching Score

Query: I recently got a file describing a library

['recent', 'got', 'file', 'describ', 'librari']

[1 ,0]
```

:[ ] In

```

```