:[1] In

```
import pandas as pd
import numpy as np
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
import os
import string
import copy
import pickle
```

```
nltk_data] Downloading package stopwords to]
...nltk_data]     C:\Users\D7me_\AppData\Roaming\nltk_data]
!nltk_data]   Package stopwords is already up-to-date]
nltk_data] Downloading package punkt to]
...nltk_data]     C:\Users\D7me_\AppData\Roaming\nltk_data]
!nltk_data]   Package punkt is already up-to-date]
```

:[2] In

```
title = "20_newsgroups"
os.chdir("C:\\20_newsgroups\\20_newsgroups\\")
```

:[3] In

```
paths = []
for (dirpath, dirnames, filenames) in os.walk(str(os.getcwd())+'/'+title+'/'):
    for i in filenames:
        paths.append(str(dirpath)+str("\\")+i)
```

:[4] In

```
print(dirpath)
```

```
C:\20_newsgroups\20_newsgroups/20_newsgroups/alt.atheism
```

:[5] In

```python
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)
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)
```

:[6] In

```

def preprocess(data, query):
    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
```

:[7] In

```
doc = 0
postings = pd.DataFrame()

for path in paths:
    file = open(path, 'r', encoding='cp1250')
    text = file.read().strip()
    file.close()
    preprocessed_text = preprocess(text, False)

    #Genrate matrex posting list
    if doc%100 == 0:
        print(doc)
    tokens = word_tokenize(str(preprocessed_text))
    for token in tokens:
        if token in postings:
            p = postings[token][0]
            p.add(doc)
            postings[token][0] = p
        else:
            postings.insert(value=[{doc}], loc=0, column=token)
    doc += 1

#Save the output:
postings.to_pickle(title + "_unigram_postings")
```

```
0
```

:[8] In

```
postings
```

Out[8]:

fi	one	decemb	nine	...	mcdowel	dj	ftp	rutger	pub	soc	uneven	side	compil	exam	
}	,0} ,1 ,2 ,3 ,4 ,5 ,7 ,8 ,13 ,14 ,16 ,17 ,18 ...1	{0}	,0} ,2 ,1 ,5 ,4 ,17 ,18 {19	...	{21}	{21}	{21}	{21}	{21}	{21}	{21}	{21}	{21}	{21}	0

rows × 1949 columns 1

:[9] In

```
postings = pd.read_pickle(title + "_unigram_postings")
```

:[10] In

```
s1 = postings['one'][0]
s2 = postings['nine'][0]
s3 = postings['exam'][0]
print(s1)
print(s2)
print(s3)

print('one AND nine AND exam = ', s1 & s2 & s3)
```

```
{21 ,20 ,19 ,18 ,17 ,16 ,14 ,13 ,8 ,7 ,5 ,4 ,3 ,2 ,1 ,0}
{19 ,18 ,17 ,5 ,4 ,2 ,1 ,0}
{21}
()one AND nine AND exam =  set
```

:[11] In

```
def get_not(word):
    a = postings[word][0]
    b = set(range(len(paths)))
    return b.difference(a)

s1 = postings['one'][0]
s2 = postings['nine'][0]
s3 = get_not('exam')

print(s1)
print(s2)
print(s3)

print('one AND nine NOT exam = ', s1 & s2 & s3)
```

```
{21 ,20 ,19 ,18 ,17 ,16 ,14 ,13 ,8 ,7 ,5 ,4 ,3 ,2 ,1 ,0}
{19 ,18 ,17 ,5 ,4 ,2 ,1 ,0}
{20 ,19 ,18 ,17 ,16 ,15 ,14 ,13 ,12 ,11 ,10 ,9 ,8 ,7 ,6 ,5 ,4 ,3 ,2 ,1 ,0}
{one AND nine NOT exam =  {0, 1, 2, 4, 5, 17, 18, 19
```

:[12] In

```
def generate_command_tokens(query):
    query = query.lower()
    tokens = word_tokenize(query)

    commands = []
    query_words = []

    for t in tokens:
        if t not in ['and', 'or', 'not']:
            processed_word = preprocess([t], True)
            print(str(processed_word))
            query_words.append(str(processed_word))
        else:
            commands.append(t)

    return commands, query_words
```

:[13] In

```
def gen_not_tuple(query_words, commands):
    tup = []
    while 'not' in commands:
        i= commands.index('not')
        word = query_words[i]
        word_postings = get_not(word)
        tup.append(word_postings)
        commands.pop(i)
        query_words[i] = i
        print("\nAfter Not Processing: ",commands, query_words)
    return tup
```

:[14] In

```
def binary_operations(query_words, commands, tup):
    a = postings[query_words[0]][0]
    query_words.pop(0)

    for i in range(len(commands)):
        if type(query_words[i]) == int:
            b = tup.pop(0)
        else:
            b = postings[query_words[i]][0]

        if commands[i] == 'and':
            a = a.intersection(b)
        elif commands[i] == 'or':
            q= a.union(b)
        else:
            print('Invaled Command')

    return a
```

:[15] In

```
def execute_query(query):
    commands, query_words = generate_command_tokens(query)
    tup = gen_not_tuple(query_words, commands)
    print('\nCommands: ', commands)
    print('\nQuery Words: ', query_words)
    print('\nTup: ', tup)

    final_set = binary_operations(query_words, commands, tup)
    print('\nFinal Set: ', final_set)

    return final_set
```

:[16] In

```
def print_file(file):
    out_file = open(path[file], 'r', encoding='cp1250')
    out_text = out_file.read()
    print(out_test)
```

:[18] In

```
query = 'exam and not resourc'
lists = execute_query(query)
```

```
No Header
exam
No Header
resourc

[After Not Processing:  ['and'] ['exam', 1

['Commands:  ['and

[Query Words:  ['exam', 1

Tup:  [{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2
[{0, 21

{Final Set:  {21
```

:[ ] In

```
```