In [1]:

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

In [12]:

```
title = "20_newsgroups"
os.chdir("C:/Users/user/Desktop/20_newsgroups")
```

In [13]:

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

In [14]:

```
print(dirpath)
```

```
C:\Users\user\Desktop\20_newsgroups/20_newsgroups/alt.atheism
```

:[15] In

```
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)


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

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


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)

#make number 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

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


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

```
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)
```

:[16] In

```
#Exercise 1
def preprocess(data, query):
    if not 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
```

:[17] 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


postings.to_pickle(title + "_unigram_postings")
```

0

In [18]:

```
postings
```

Out[18]:

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

In [19]:

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

In [20]:

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

print('intersection = ', 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}
()intersection =  set
```

In [57]:

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

In [58]:

```
def get_word_postings(word):
    preprocessed_word=str(preprocess(word,True))
    print(preprocessed_word)
    print("Frequency :",len(postings[preprocessed_word][0]))
    print("postings list :",(postings[preprocessed_word][0]))
get_word_postings("nine")
```

```
nine
Frequency : 8
{postings list : {0, 1, 2, 4, 5, 17, 18, 19
```

In [59]:

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

Out[59]:

```
{21 ,20 ,16 ,15 ,14 ,13 ,12 ,11 ,10 ,9 ,8 ,7 ,6 ,3}
```

In [69]:

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

    commands=[]
    query_word=[]

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

In [73]:

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

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

    for i in range(len(commands)):
        if type(query_word[i])==int:
            b=tup.pop(0)
        else:
            b=postings[query_word[i]][0]
        if commands[i]=='and':
            a=a.intersection(b)
        elif commands[i]=='or':
                a=a.union(b)
        else:
            print("Invaild Command")

    return a

def execute_query(query):
    commands,query_word=generate_command_tokens(query)
    tup=gen_not_tuple(query_word,commands)

    print("\nCommands ", commands)
    print("\nquery word",query_word)
    print("\ntup",len(tup))
    final_set=binary_operations(query_word,commands , tup)
    print("\nFinal set",final_set)
    return final_set

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

In [78]:

```
query="nine and exam"
```

In [79]:

```
lists=execute_query(query)
```

```
['nine']
['exam']

['Commands  ['and

["['query word ["['nine']", "['exam

tup 0
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent cal
(l last
 G:\anacondaProgram\lib\site-packages\pandas\core\indexes\base.py in
(get_loc(self, key, method, tolerance
:try              2645
(return self._engine.get_loc(key                  2646 <-
:except KeyError              2647

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Py
()ObjectHashTable.get_item

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Py
()ObjectHashTable.get_item

"['KeyError: "['nine

:During handling of the above exception, another exception occurred

KeyError                                  Traceback (most recent cal
(l last
<ipython-input-79-b1ef25bd3a45> in <module>
(lists=execute_query(query 1 <----

(ipython-input-73-4fee4e460ea2> in execute_query(query>
(print("\nquery word",query_word     36
((print("\ntup",len(tup     37
(final_set=binary_operations(query_word,commands , tup     38 <---
(print("\nFinal set",final_set     39
return final_set     40

ipython-input-73-4fee4e460ea2> in binary_operations(query_word, com>
(mands, tup
 12
:(def binary_operations(query_word,commands , tup 13
[a=postings[query_word[0]][0     14 <---
(query_word.pop(0     15
 16

G:\anacondaProgram\lib\site-packages\pandas\core\frame.py in __getit
(em__(self, key
:if self.columns.nlevels > 1             2798
(return self._getitem_multilevel(key                  2799
```

```
(indexer = self.columns.get_loc(key            2800 <-
:(if is_integer(indexer             2801
[indexer = [indexer                 2802

 G:\anacondaProgram\lib\site-packages\pandas\core\indexes\base.py in
(get_loc(self, key, method, tolerance
(return self._engine.get_loc(key                2646
:except KeyError             2647
return self._engine.get_loc(self._maybe_cast                2648 <-
((_indexer(key
indexer = self.get_indexer([key], method=method, tol         2649
(erance=tolerance
:if indexer.ndim > 1 or indexer.size > 1         2650

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

()pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Py
()ObjectHashTable.get_item

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.Py
()ObjectHashTable.get_item

"['KeyError: "['nine
```

:[ ] In