Xi'an Jiaotong-Liverpool University

西交利物浦大學

Lab 6: Feature Generation

Author **Jeongyeong Park**

Module **INT104**

Teacher Xiaobo Jin

Date **27**th/April/202

Introduction

Feature generation is the method of generating new features from one or more existing features. In this lab, the features are generated from the text collections. All files in subdirectories of the dataset should be read and texts in these files should be split into words. In order to deal with files more efficiently in this lab, the name of them would better save into 2-dimensional list. Since files might contain some stopwords which appear frequently but do not convey much meaning, they should be removed. For TF-IDF, unique words which appear even once in the files should be obtained. Finally, as going through all files in the dataset, words in each file are compared to obtained unique words and TF-IDF will be computed. When words are dealt in this lab, they should be the key of dictionary for each file so they can be tracked easily.

Design & Implementation

1. Design

1) Design for reading the text files and split the document text into words

First, names of subdirectories and the files in each subdirectories should be recorded in lists(subdirec_List and textfile_List respectively) so they can be used for reading files. Subdirec_List will be a linear list in order to save the name of each subdirectory, and textfile_List will be a 2-dimensional list in order to save all files in each subdirectory in each list. In order to read the files and save splitted words from the document text, each list for each subdirectory is created. Then, as we go through each file, the document texts are read and splitted into words and the splitting separator is non-alphabet letters. Therefore, when this process is done, non-alphabet letters are replaced into ‘ ‘(one space), all words in the document text are changed into lower-case. Finally, all words are separated with the default separator which is any whitespace since all non-alphabet letters changed into ‘ ‘. The following line will be used :

re.sub(r’[^a-zA-Z]’,’ ‘,word.lower()).split()

2) Design for removing stopwords from the text collection

In order to remove stopwords that are appearing frequently but having no information, a text file containing stopwords should be read first. After reading this text file, each stopword will be saved in a list. Since words in every file are splitted with a splitting separator which is non-alphabet, stopwords also should be splitted with the same splitting separator. The following line is used :

re.sub(r'[^a-zA-z]','',stopword)

Then, the stopword list will be converted to set in order to remove duplicate words. Words in each word list of each file will be compared to these stopwords in the stopword list and if a word is the same as the stopword in the stopwords list, it will be removed. In python list, if there are two identical elements in the list, and that element should be removed, only the first one will be removed from the list. Thus, following line is used :

```
list1=[word for word in list1 if word not in list2]
```

to make sure all stopwords in the words lists of files are removed.

3) Design for performing word stemming to remove the word suffix

For stemming, following line is used :

```
list=[stemmer.stem(plural) for plural in list]
```

4) Design for computing the frequency of each word in each document

For this step, unique words appearing in files located in the dataset should be obtained first. By using set, unique words appearing in files in each subdirectory are obtained first. Set will automatically remove duplicate words. Then, these sets are combined together in order to get final unique words set with the following line :

```
final_set=set1.union(set2.union(set3.union(set4.union(set5))))
```

After obtaining unique words, this set is converted to list to have index. A dictionary is created for saving the frequency of each word in each document. In order to keep tracking filename and the frequency of each word together, nested dictionary is created such as following example :

dictionary={'file_name1':{'word1' : 2, 'word2':0…….}, 'file_name2':{'word1' : 0, 'word2' : 2…..},....}

For counting words, first, the inner dictionaries are created for all files in the dataset. Then, unique words will be set as keys for inner dictionaries and value will be set for 0. Example of key line :

```
inner_dictionary=dict.fromkeys(unique_word_list,0)
```

Then, as going through the list of words of the file, every time the word in the list of unique words is detected in the list of words of the file, one is added to the value of having a key as that word. Example of key lines :

```
for word in word_list_of_file1:

    dictionary[file1][word]+=1
```

This process is done for each subdirectory.

5) Design for computing document frequency

In order to obtain document frequency, a dictionary for it is needed and set unique words in list as keys and set value for it as 0. The following line will be used :

```
dictionary=dict.fromkeys(unique_words_list,0)
```

Then, as going through a dictionary of frequency of a word, if a value of a particular word is not 0 in that file, add one to the value having key as that word. Example of key lines :

```
for word in unique_word_list:
    if (dictionary_of_frequency[file1][word]!=0):
        dictionary_of_document_fre[word]+=1
```

6) Design for computing document-by-word matrix

In order to obtain document-by-word matirx, a_{ik} which is the weight of word k in document i. For a_{ik}, $a_{ik} = f_{ik} * \log(\frac{N}{n_k})$ formula is used. In order to calculate, N which is the number of documents in the dataset should be found. Then, an array in order to save a_{ik} is created. Additionally, in order to save each a_{ik} for each word, N numbers of dictionaries will be created inside this array and starting from the first file of first subdirectory dictionary will be indexed 0 from N-1. By using dictionary for the frequency of word in document and dictionary for the document frequency, computing a_{ik}. Example of key lines:

```
for word in unique_word_list:
    array_for_weight[index][word]=dict_frequency[file][word]*math.log(N/dict_n[word])
```

After that, list is created to save value for final matrix. Inside this linear list, N number of dictionaries will be created to get value A_{ik} for each word in each file. A_{ik} is computed with formula :

$$A_{ik} = \frac{a_{ik}}{\sqrt{\sum_{j=1}^{D} a_{ij}^2}}$$ D=the number of the unique words in the document

Words in same document share same denominator in this formula. Therefore, for each file, calculating denominator first, and then use it to calculate to calculate each A_{ik} value. After calculating every A_{ik} in the same file, denominator should be set 0 in order to calculate new denominator for words in next file. Example of key lines :

```
denominator=0
for word in unique_word_list:
```

denominator=math.sqrt(float(array_weight[index][word]))+denominator

for word in unique_word_list:

document_by_word[index][word]=array_weight[index][word]/denominator

denominator=0

After computing A_{ik} for all files in the dataset, convert this list within dictionaries into DataFrame. Since the A_{N*D} matrix should be computed, convert this DataFrame into numpy array. Finally, save this numpy array into npz file.

2. Implementation

1) Code for reading the text files and split the document text into words

```
import os
import numpy as np
from nltk.stem.porter import *
import pandas as pd
import math


#Listing name of subdirectories in dataset
subdirec_List=os.listdir('/dataset')

#Listing files in each subdirectories(using jagged list)
textfile_List=[]
for i in range (len(subdirec_List)):
        textfile_List.append([])
        textfile_List[i]=os.listdir('/dataset/'+subdirec_List[i])
```

Figure 1. Code for reading the name of files

```
wd_alt =[]
wd_comp=[]
wd_rec =[]
wd_soc=[]
wd_talk=[]
#Open and read files in each subdirectories
for i in range (len(subdirec_List)):
        for j in textfile_List[i]:
                if(i==0):
                        f=open('/dataset/'+subdirec_List[i]+'/'+j,'r',encoding='Latin1')
                        r = f.read()
                        wd_alt.append(re.sub(r'[^a-zA-Z]',' ',r.lower()).split())
                elif(i==1):
                        f = open('/dataset/' + subdirec_List[i] + '/' + j, 'r', encoding='Latin1')
                        r = f.read()
                        wd_comp.append(re.sub(r'[^a-zA-Z]', ' ', r.lower()).split())
                elif(i==2):
                        f = open('/dataset/' + subdirec_List[i] + '/' + j, 'r', encoding='Latin1')
                        r = f.read()
                        wd_rec.append(re.sub(r'[^a-zA-Z]', ' ', r.lower()).split())
                elif(i==3):
                        f = open('/dataset/' + subdirec_List[i] + '/' + j, 'r', encoding='Latin1')
                        r = f.read()
                        wd_soc.append(re.sub(r'[^a-zA-Z]', ' ', r.lower()).split())
                else:
                        f = open('/dataset/' + subdirec_List[i] + '/' + j, 'r', encoding='Latin1')
                        r = f.read()
                        wd_talk.append(re.sub(r'[^a-zA-Z]', ' ', r.lower()).split())
```

Figure 2. Code for splitting the document text into words

2) Code for removing stopwords from the text collection

```
#save stopwords in the list
sw = open('/stopwords.txt','r',encoding='Latin1')
stopword=sw.read().splitlines()
for j in range (len(stopword)):
    stopword[j]=re.sub(r'[^a-zA-z]','',stopword[j])

#removing stopwords
#for alt.atheism
for i in range (len(wd_alt)):
    wd_alt[i]=[word for word in wd_alt[i] if word not in stopword]
#for comp.graphics
for i in range (len(wd_comp)):
    wd_comp[i]=[word for word in wd_comp[i] if word not in stopword]
#for rec.motorcycles
for i in range (len(wd_rec)):
    wd_rec[i]=[word for word in wd_rec[i] if word not in stopword]
#for soc.religion.christian
for i in range (len(wd_soc)):
    wd_soc[i]=[word for word in wd_soc[i] if word not in stopword]
#for talk.politics.misc
for i in range (len(wd_talk)):
    wd_talk[i]=[word for word in wd_talk[i] if word not in stopword]
```

Figure 3. Code for removing stopwords

3) Code for performing word stemming to remove the word suffix

```
#stemming
stemmer=PorterStemmer()
#for alt.atheism
for i in range (len(wd_alt)):
        wd_alt[i]=[stemmer.stem(plural) for plural in wd_alt[i]]
#for comp.graphics
for i in range (len(wd_comp)):
        wd_comp[i]=[stemmer.stem(plural) for plural in wd_comp[i]]
#for rec.motorcycles
for i in range (len(wd_rec)):
        wd_rec[i] = [stemmer.stem(plural) for plural in wd_rec[i]]
#for soc.religion.christian
for i in range (len(wd_soc)):
        wd_soc[i] = [stemmer.stem(plural) for plural in wd_soc[i]]
#for talk.politics.misc
for i in range (len(wd_talk)):
        wd_talk[i] = [stemmer.stem(plural) for plural in wd_talk[i]]

```

Figure 4. Code for stemming

4) Code for computing the frequency of each word in each document

```
#creating set for unique word appear even once in one of files in dataset
# for alt.atheism
uniwd_alt=set()
for i in range (len(wd_alt)):
    uniwd_alt.update(wd_alt[i])
# for comp.graphics
uniwd_comp=set()
for i in range (len(wd_comp)):
    uniwd_comp.update(wd_comp[i])
# for rec.motorcycles
uniwd_rec=set()
for i in range (len(wd_rec)):
    uniwd_rec.update(wd_rec[i])
# for soc.religion.christian
uniwd_soc=set()
for i in range (len(wd_soc)):
    uniwd_soc.update(wd_soc[i])
# for talk.politics.misc
uniwd_talk=set()
for i in range (len(wd_talk)):
    uniwd_talk.update(wd_talk[i])

# final unique word
uniwd=set()
uniwd=uniwd_alt.union(uniwd_comp.union(uniwd_rec.union(uniwd_soc.union(uniwd_talk))))
uniwd=list(uniwd)

```

Figure 5. Code for obtaining set of unique word and converting it to list

```
#creating dictionary for storing the frequency of word k in document i
dict_f={}
for i in range (len(textfile_List)):
    for j in range (len(textfile_List[i])):
        dict_f[textfile_List[i][j]]=dict.fromkeys(uniwd,0) #Adding unique words as a key of inner dictionary and set 0 for value

# for files in alt.atheism
for i in range (len(textfile_List[0])):
    for word in wd_alt[i]:
        dict_f[textfile_List[0][i]][word]+=1
#for files in comp.graphics
for i in range (len(textfile_List[1])):
    for word in wd_comp[i]:
        dict_f[textfile_List[1][i]][word]+=1
#for files in rec.motorcycles
for i in range (len(textfile_List[2])):
    for word in wd_rec[i]:
        dict_f[textfile_List[2][i]][word]+=1
#for files in soc.religion.christian
for i in range (len(textfile_List[3])):
    for word in wd_soc[i]:
        dict_f[textfile_List[3][i]][word]+=1
#for files in talk.politics.misc
for i in range (len(textfile_List[4])):
    for word in wd_talk[i]:
        dict_f[textfile_List[4][i]][word]+=1
```

Figure 6. Code for frequency of word

5) Design for computing document frequency

```
#creating dictionary for storing the document frequency
dict_n=dict.fromkeys(uniwd,0)
for i in range (len(textfile_List)):
    for j in range (len(textfile_List[i])):
        for word in uniwd:
            if (dict_f[textfile_List[i][j]][word]!=0):
                dict_n[word]+=1
```

Figure 7. Code for document frequency

6) Design for computing document-by-word matrix

```
# finding the number of documents(N) in the dataset
N=0
for i in range (len(textfile_List)):
    for j in range (len(textfile_List[i])):
        N=N+1
```

Figure 8. Code for N which is the number of documents in the dataset

```
#computing the weight of word k in document i : aik
array_weight=[]
for i in range (len(textfile_List)):
    for j in range (len(textfile_List[i])):
        array_weight.append(dict.fromkeys(uniwd,0))

index=-1
for i in range (len(textfile_List)):
    for j in range (len(textfile_List[i])):
        index=index+1
        for word in uniwd:
            array_weight[index][word]=dict_f[textfile_List[i][j]][word]*math.log(N/dict_n[word])
```

Figure 9. Code for the weight of word

```
#computing the document-by-word matrix
document_by_word=[]
for i in range (len(textfile_List)):
    for j in range (len(textfile_List[i])):
        document_by_word.append(dict.fromkeys(uniwd,0))

index2=-1
denominator=0
for i in range (len(textfile_List)):
    for j in range (len(textfile_List[i])):
        index2=index2+1
        for word in uniwd:
            denominator=math.sqrt(float(array_weight[index2][word]))+denominator
        for word in uniwd:
            document_by_word[index2][word]=array_weight[index2][word]/denominator
        denominator=0
```

Figure 10. Code for the document-by-word list within dictionaries

```
#Creating final matirx and save into npz file
df_matrix=pd.DataFrame(document_by_word)
final_matrix=df_matrix.to_numpy()
np.savez('train-20ng.npz',X=final_matrix)
```

Figure 11. Code for the final matrix

Test

Following figures are examples of testing to see if the code is working properly.

1) Test for reading the text files and split the document text into words

```
Print subdirect_List
['alt.atheism', 'comp.graphics', 'rec.motorcycles', 'soc.religion.christian', 'talk.politics.misc']
```

Figure 12. Result of subdirect_List

```
Print textfile_List
[['49960', '51060', '51119', '51120', '51121', '51122', '51123', '51124', '51125', '51126', '51127', '51128', '51130', '51131', '51132', '51133', '51134', '51135', '51136',
 '51139', '51140', '51141', '51142', '51143', '51144', '51145', '51146', '51147', '51148', '51149', '51150', '51151', '51152', '51153', '51154', '51155', '51156', '51157',
 '51158', '51159', '51160', '51161', '51162', '51163', '51164', '51165', '51169', '51170', '51171', '51172', '51173', '51174', '51175', '51176', '51177', '51178', '51179',
 '51180', '51181', '51182', '51183', '51184', '51185', '51186', '51187', '51188', '51189', '51190', '51191', '51192', '51193', '51194', '51195', '51196', '51197', '51198',
 '51199', '51200', '51201', '51202', '51203', '51204', '51205', '51206', '51208', '51209', '51210', '51211', '51212', '51213', '51214', '51215', '51216', '51217', '51218',
 '51219', '51220', '51221', '51222', '51223', '51224', '51225', '51226', '51227', '51228', '51229', '51230', '51231', '51232', '51233', '51234', '51235', '51236', '51237',
 '51238', '51239', '51240', '51241', '51242', '51243', '51244', '51245', '51246', '51247', '51249', '51250', '51251', '51252', '51253', '51254', '51255', '51256', '51258',
 '51259', '51260', '51261', '51262', '51265', '51266', '51267', '51268', '51269', '51270', '51271', '51272', '51273', '51274', '51275', '51276', '51277', '51278', '51279',
 '51280', '51281', '51282', '51283', '51284', '51285', '51286', '51287', '51288', '51290', '51291', '51292', '51293', '51294', '51295', '51296', '51297', '51298', '51299',
 '51300', '51301', '51302', '51303', '51304', '51305', '51306', '51307', '51308', '51309', '51310', '51311', '51312', '51313', '51314', '51315', '51316', '51317', '51318',
 '51319', '51320', '52499', '52909', '52910', '53055', '53056', '53057', '53058', '53059', '53062', '53064', '53065', '53066', '53067', '53069', '53070', '53071', '53072',
 '53073', '53075', '53078', '53081', '53082', '53083', '53085', '53086', '53087', '53090', '53093', '53094', '53095', '53096', '53097', '53098', '53099', '53106', '53108',
 '53110', '53111', '53112', '53113', '53114', '53117', '53118', '53120', '53121', '53122', '53123', '53124', '53125', '53126', '53127', '53130', '53131', '53132', '53133',
 '53134', '53135', '53136', '53137', '53139', '53140', '53141', '53142', '53143', '53144', '53145', '53149', '53150', '53151', '53153', '53154', '53157', '53158', '53159',
 '53160', '53161', '53162', '53163', '53164', '53165', '53166', '53167', '53168', '53170', '53171', '53172', '53173', '53174', '53175', '53176', '53177', '53178', '53179',
 '53180', '53181', '53182', '53183', '53184', '53185', '53186', '53187', '53188', '53190', '53191', '53192', '53193', '53194', '53195', '53196', '53197', '53198', '53199',
 '53201', '53203', '53208', '53209', '53210', '53211', '53212', '53213', '53214', '53215', '53216', '53217', '53218', '53219', '53220', '53221', '53222', '53223', '53224',
 '53225', '53226', '53228', '53229', '53230', '53231', '53232', '53235', '53237', '53238', '53239', '53240', '53243', '53248', '53249', '53250', '53251', '53252', '53256',
 '53258', '53266', '53267', '53269', '53271', '53274', '53275', '53281', '53282', '53283', '53284', '53285', '53286', '53287', '53288', '53289', '53290', '53292', '53298',
 '53303', '53304', '53305', '53306', '53307', '53308', '53309', '53311', '53312', '53314', '53323', '53334', '53347', '53351', '53366', '53370', '53371', '53373', '53374',
 '53375', '53376', '53377', '53380', '53381', '53382', '53383', '53387', '53389', '53390', '53391', '53434', '53435', '53436', '53437', '53438', '53439', '53440', '53441',
 '53442', '53443', '53445', '53449', '53459', '53460', '53465', '53466', '53467', '53468', '53471', '53477', '53478', '53483', '53509', '53510', '53512', '53515', '53518',
 '53519', '53521', '53522', '53523', '53524', '53525', '53526', '53527', '53528', '53529', '53531', '53532', '53533', '53534', '53535', '53571', '53572', '53573', '53574',
 '53654', '53655', '53656', '53660', '53661', '53753', '53754', '53755', '53756', '53757', '53758', '53759', '53760', '53761', '53762', '53763', '53764', '53765', '53766',
 '53780', '53785', '54165', '54166', '54167', '54168', '54178', '54179', '54180', '54181', '54182', '54198', '54199', '54200', '54201', '54202', '54203', '54204', '54227',
```

Figure 13. Omitted result of textfile_List

```
['from', 'mathew', 'mathew', 'mantis', 'co', 'uk', 'subject', 'alt', 'atheism', 'faq', 'atheist', 'resources', 'summary', 'books', 'addresses', 'music', 'anything',
 'related', 'to', 'atheism', 'keywords', 'faq', 'atheism', 'books', 'music', 'fiction', 'addresses', 'contacts', 'expires', 'thu', 'apr', 'gmt', 'distribution', 'world',
 'organization', 'mantis', 'consultants', 'cambridge', 'uk', 'supersedes', 'mantis', 'co', 'uk', 'lines', 'archive', 'name', 'atheism', 'resources', 'alt', 'atheism',
 'archive', 'name', 'resources', 'last', 'modified', 'december', 'version', 'atheist', 'resources', 'addresses', 'of', 'atheist', 'organizations', 'usa', 'freedom', 'from',
 'religion', 'foundation', 'darwin', 'fish', 'bumper', 'stickers', 'and', 'assorted', 'other', 'atheist', 'paraphernalia', 'are', 'available', 'from', 'the', 'freedom',
 'from', 'religion', 'foundation', 'in', 'the', 'us', 'write', 'to', 'ffrf', 'p', 'o', 'box', 'madison', 'wi', 'telephone', 'evolution', 'designs', 'evolution', 'designs',
 'sell', 'the', 'darwin', 'fish', 'it', 's', 'a', 'fish', 'symbol', 'like', 'the', 'ones', 'christians', 'stick', 'on', 'their', 'cars', 'but', 'with', 'feet', 'and', 'the',
 'word', 'darwin', 'written', 'inside', 'the', 'deluxe', 'moulded', 'd', 'plastic', 'fish', 'is', 'postpaid', 'in', 'the', 'us', 'write', 'to', 'evolution', 'designs',
 'laurel', 'canyon', 'north', 'hollywood', 'ca', 'people', 'in', 'the', 'san', 'francisco', 'bay', 'area', 'can', 'get', 'darwin', 'fish', 'from', 'lynn', 'gold', 'try',
 'mailing', 'figmo', 'netcom', 'com', 'for', 'net', 'people', 'who', 'go', 'to', 'lynn', 'directly', 'the', 'price', 'is', 'per', 'fish', 'american', 'atheist', 'press',
 'aap', 'publish', 'various', 'atheist', 'books', 'critiques', 'of', 'the', 'bible', 'lists', 'of', 'biblical', 'contradictions', 'and', 'so', 'on', 'one', 'such', 'book',
 'is', 'the', 'bible', 'handbook', 'by', 'w', 'p', 'ball', 'and', 'g', 'w', 'foote', 'american', 'atheist', 'press', 'pp', 'isbn', 'nd', 'edition', 'bible', 'contradictions',
 'absurdities', 'atrocities', 'immoralities', 'contains', 'ball', 'foote', 'the', 'bible', 'contradicts', 'itself', 'aap', 'based', 'on', 'the', 'king', 'james', 'version',
 'of', 'the', 'bible', 'write', 'to', 'american', 'atheist', 'press', 'p', 'o', 'box', 'austin', 'tx', 'or', 'cameron', 'road', 'austin', 'tx', 'telephone', 'fax',
 'prometheus', 'books', 'sell', 'books', 'including', 'haught', 's', 'holy', 'horrors', 'see', 'below', 'write', 'to', 'east', 'amherst', 'street', 'buffalo', 'new', 'york',
 'telephone', 'an', 'alternate', 'address', 'which', 'may', 'be', 'newer', 'or', 'older', 'is', 'prometheus', 'books', 'glenn', 'drive', 'buffalo', 'ny', 'african',
 'americans', 'for', 'humanism', 'an', 'organization', 'promoting', 'black', 'secular', 'humanism', 'and', 'uncovering', 'the', 'history', 'of', 'black', 'freethought',
 'they', 'publish', 'a', 'quarterly', 'newsletter', 'aah', 'examiner', 'write', 'to', 'norm', 'r', 'allen', 'jr', 'african', 'americans', 'for', 'humanism', 'p', 'o', 'box',
 'buffalo', 'ny', 'united', 'kingdom', 'rationalist', 'press', 'association', 'national', 'secular', 'society', 'islington', 'high', 'street', 'holloway', 'road', 'london',
 'n', 'ew', 'london', 'n', 'nl', 'british', 'humanist', 'association', 'south', 'place', 'ethical', 'society', 'lamb', 's', 'conduit', 'passage', 'conway', 'hall', 'london',
 'wc', 'r', 'rh', 'red', 'lion', 'square', 'london', 'wc', 'r', 'rl', 'fax', 'the', 'national', 'secular', 'society', 'publish', 'the', 'freethinker', 'a', 'monthly',
 'magazine', 'founded', 'in', 'germany', 'ibka', 'e', 'v', 'internationaler', 'bund', 'der', 'konfessionslosen', 'und', 'atheisten', 'postfach', 'd', 'berlin', 'germany',
 'ibka', 'publish', 'a', 'journal', 'miz', 'materialien', 'und', 'informationen', 'zur', 'zeit', 'politisches', 'journal', 'der', 'konfessionslosesn', 'und', 'atheisten',
 'hrsg', 'ibka', 'e', 'v', 'miz', 'vertrieb', 'postfach', 'd', 'berlin', 'germany', 'for', 'atheist', 'books', 'write', 'to', 'ibdk', 'internationaler', 'b', 'ucherdienst',
 'der', 'konfessionslosen', 'postfach', 'd', 'hannover', 'germany', 'telephone', 'books', 'fiction', 'thomas', 'm', 'disch', 'the', 'santa', 'claus', 'compromise', 'short',
 'story', 'the', 'ultimate', 'proof', 'that', 'santa', 'exists', 'all', 'characters', 'and', 'events', 'are', 'fictitious', 'any', 'similarity', 'to', 'living', 'or', 'dead',
```

Figure 14. One of the omitted result of reading file

2) Test for removing stopwords from the text collection

```
#testing stopwords
for word in stopword:
    if word in wd_alt[0]:
        print("There is still a stopword in the list")
print("Test is finished")

for i in range (len(wd_talk))
```

```
Run:  main
C:\Users\raonm\anaconda3\envs\pythonProject\python.exe C:/Users/raonm/PycharmProjects/pythonProject/main.py
Test is finished
```

Figure 15. Example of testing to see if stopwords are removed from the file

3) Code for performing word stemming to remove the word suffix

```
#Testing stemming
print(wd_alt[0])

#creating set for unique word appear even once in one of files in dataset
# for alt.atheism
```

```
Run:  main
C:\Users\raonm\anaconda3\envs\pythonProject\python.exe C:/Users/raonm/PycharmProjects/pythonProject/main.py
['mathew', 'mathew', 'manti', 'uk', 'subject', 'alt', 'atheism', 'faq', 'atheist', 'resourc', 'summari', 'book', 'address', 'music', 'atheism', 'keyword', 'faq', 'atheism',
 'book', 'music', 'fiction', 'address', 'contact', 'expir', 'thu', 'apr', 'gmt', 'distribut', 'organ', 'manti', 'consult', 'cambridg', 'uk', 'supersed', 'manti', 'uk',
 'line', 'archiv', 'atheism', 'resourc', 'alt', 'atheism', 'archiv', 'resourc', 'modifi', 'decemb', 'version', 'atheist', 'resourc', 'address', 'atheist', 'organ', 'usa',
 'freedom', 'religion', 'foundat', 'darwin', 'fish', 'bumper', 'sticker', 'assort', 'atheist', 'paraphernalia', 'freedom', 'religion', 'foundat', 'write', 'ffrf', 'box',
 'madison', 'wi', 'telephon', 'evolut', 'design', 'evolut', 'design', 'sell', 'darwin', 'fish', 'fish', 'symbol', 'christian', 'stick', 'car', 'feet', 'word', 'darwin',
 'written', 'delux', 'mould', 'plastic', 'fish', 'postpaid', 'write', 'evolut', 'design', 'laurel', 'canyon', 'north', 'hollywood', 'peopl', 'san', 'francisco', 'bay',
 'area', 'darwin', 'fish', 'lynn', 'gold', 'mail', 'figmo', 'netcom', 'net', 'peopl', 'lynn', 'price', 'fish', 'american', 'atheist', 'press', 'aap', 'publish', 'atheist',
 'book', 'critiqu', 'bibl', 'list', 'biblic', 'contradict', 'book', 'bibl', 'handbook', 'ball', 'foot', 'american', 'atheist', 'press', 'isbn', 'edit', 'bibl', 'contradict',
 'absurd', 'atroc', 'immor', 'ball', 'foot', 'bibl', 'contradict', 'aap', 'base', 'king', 'jame', 'version', 'bibl', 'write', 'american', 'atheist', 'press', 'box', 'austin'
  'tx', 'cameron', 'road', 'austin', 'tx', 'telephon', 'fax', 'prometheu', 'book', 'sell', 'book', 'includ', 'haught', 'holi', 'horror', 'write', 'east', 'amherst', 'street'
  'buffalo', 'york', 'telephon', 'altern', 'address', 'newer', 'older', 'prometheu', 'book', 'glenn', 'drive', 'buffalo', 'ny', 'african', 'american', 'human', 'organ',
 'promot', 'black', 'secular', 'human', 'uncov', 'histori', 'black', 'freethought', 'publish', 'quarterli', 'newslett', 'aah', 'examin', 'write', 'norm', 'allen', 'jr',
 'african', 'american', 'human', 'box', 'buffalo', 'ny', 'unit', 'kingdom', 'rationalist', 'press', 'associ', 'nation', 'secular', 'societi', 'islington', 'high', 'street',
 'holloway', 'road', 'london', 'ew', 'london', 'nl', 'british', 'humanist', 'associ', 'south', 'place', 'ethic', 'societi', 'lamb', 'conduit', 'passag', 'conway', 'hall',
 'london', 'wc', 'rh', 'red', 'lion', 'squar', 'london', 'wc', 'rl', 'fax', 'nation', 'secular', 'societi', 'publish', 'freethink', 'monthli', 'magazin', 'found', 'germani',
 'ibka', 'international', 'bund', 'der', 'konfessionslosen', 'und', 'atheisten', 'postfach', 'berlin', 'germani', 'ibka', 'publish', 'journal', 'miz', 'materialien', 'und',
 'informationen', 'zur', 'zeit', 'politisch', 'journal', 'der', 'konfessionslosesn', 'und', 'atheisten', 'hrsg', 'ibka', 'miz', 'vertrieb', 'postfach', 'berlin', 'germani',
```

Figure 16. Example of testing to see the result of stemming

4) Test for computing the frequency of each word in each document

Figure 17. One of the omitted result of the frequency of words in file

5) Test for computing document frequency

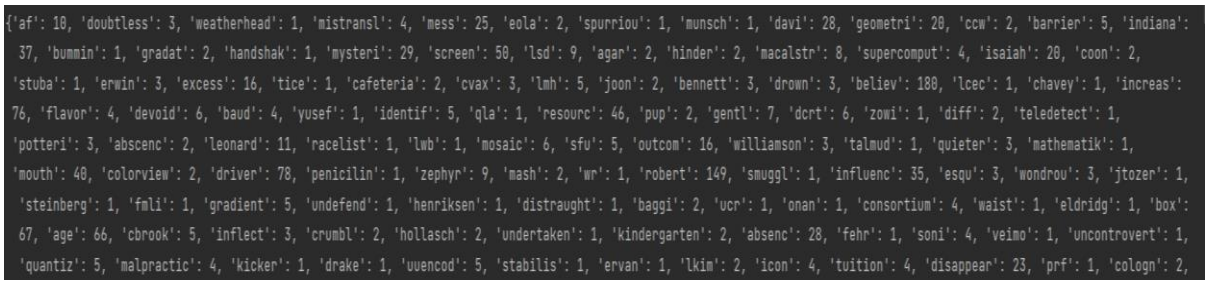
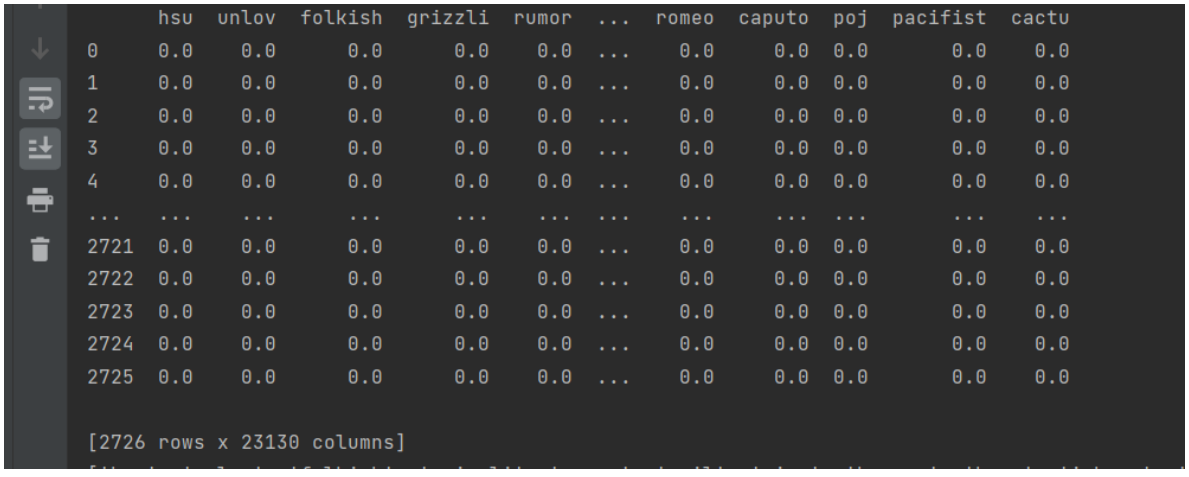


Figure 18. The omitted result of the document frequency

6) Design for computing document-by-word matrix



	hsu	unlov	folkish	grizzli	rumor	...	romeo	caputo	poj	pacifist	cactu
0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
2721	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2722	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2723	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2724	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2725	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

[2726 rows x 23130 columns]

Figure 19. The result of document-by-word matrix as DataFrame for precise view

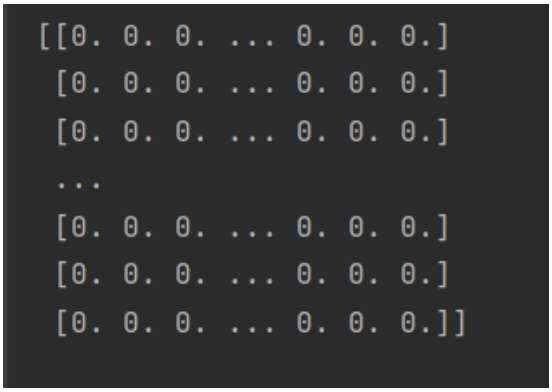


Figure 20. Corresponding matrix(numpy array) of DataFrame above

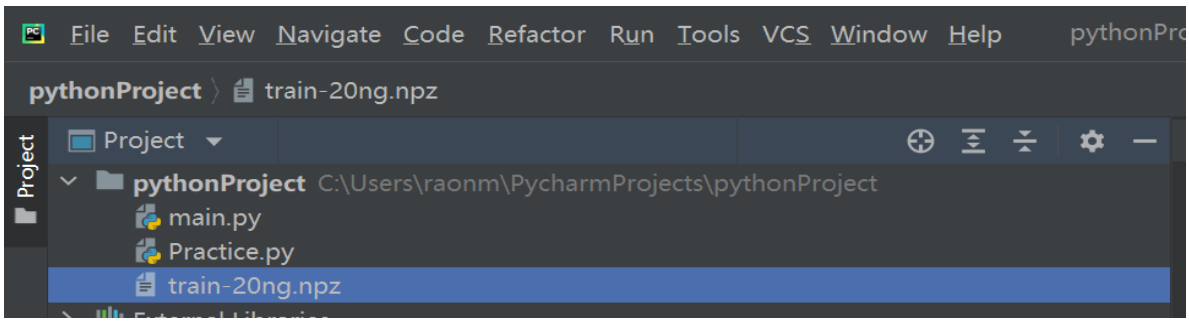


Figure 21. The result of saving final document-by-word matrix

Conclusion

The feature generation done in this lab, which is generating the features from the text collections is important i in text analysis. For example, it can be used for extracting important words in particular document, or comparin comparing similarity between documents. In the process of feature generation, various lists are used to save na names of subdirectories, and files, and words that are split from each document. In this lab, when stopwords ar are removed from the words in list, 'List Comprehension' should be used. Since using any type of loops to work for list in python takes risk because the result not expected might be got. For example, if a list contained two 'e' only first 'e' will be removed. When unique words are obtaining, 'set' should be used because it automatically d deletes duplicate words. Then, this set will be convert into list to keep index of words. For steps of computing T TF-IDF, every file should be gone through by using proper lists and indexing of it. After all, for document-by- word matrix, list containing inner dictionaries for each file of A_{ik} will be converted into DataFrame and then t that DataFrame will be converted into numpy array. Therefore, A_{ik} values can be stored into 2- array. Moreover, if the list of unique word is printed above the final 2- word can be seen more easily.