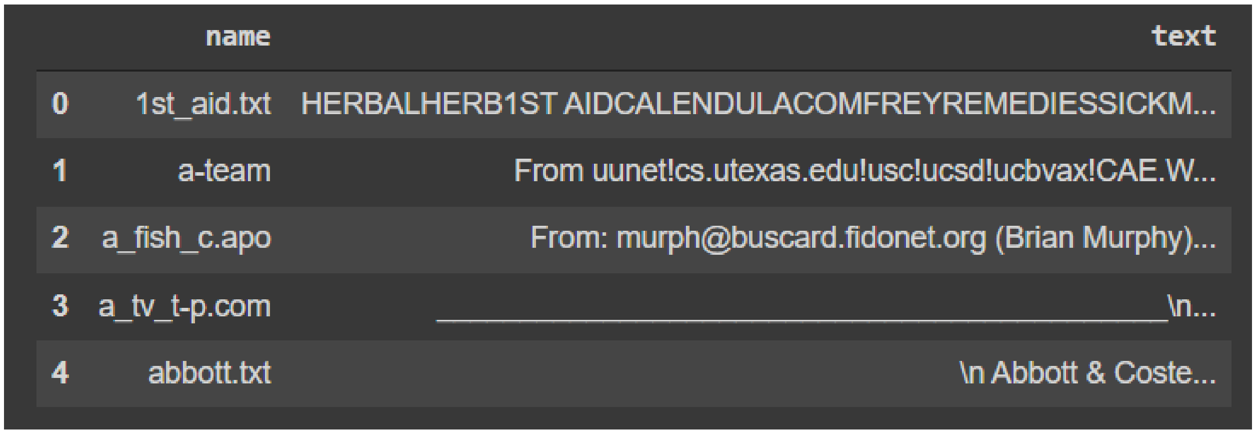
CCSE508: Information Retrieval
Assignment 1

Group- 76
Kshitij Mohan (2019054) - (Github: Kshitij-M)
Shreya Tomar (2019110) - (Github: Shreya0229)

TA assigned- Deepi Garg - (Github: deepigarg)

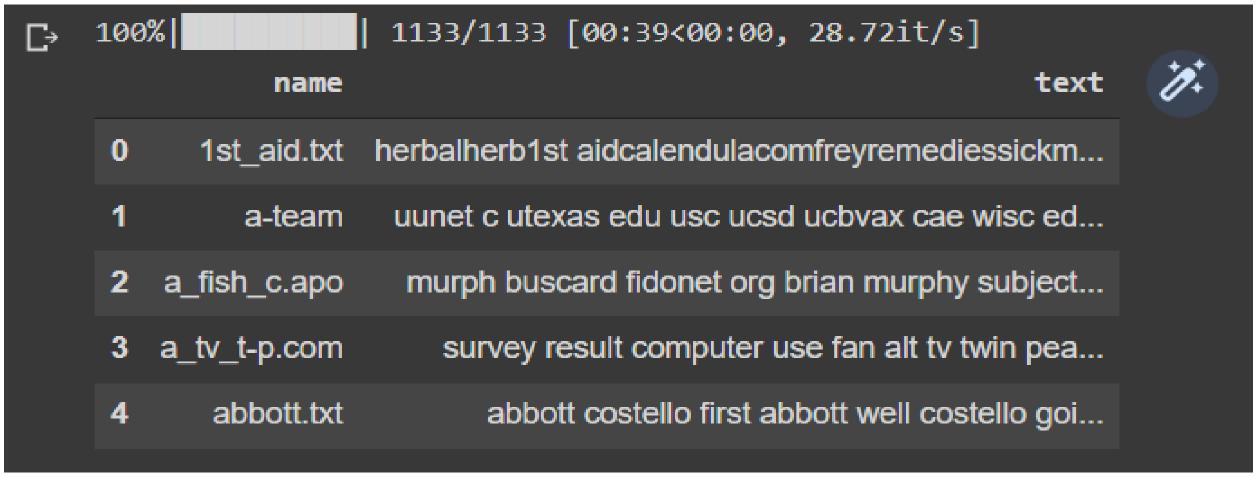
Q1.
Preprocessing: We have followed the following steps for preprocessing:
- We have read all the text files using utf-8 encoding and ignored all the errors in reading the same.
- First step of processing starts by converting all the text to lower case.
- This is followed by removing all the punctuations -
  """˛şË›ÃºÅŸ§ż±ŕőíä°üß!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c"""
- This is followed by removing English stop words.
- The final step of preprocessing is lemmatization.

Before Preprocessing:



	name	text
0	1st_aid.txt	HERBALHERB1ST AIDCALENDULACOMFREYREMEDIESSICKM...
1	a-team	From uunet!cs.utexas.edu!usc!ucsd!ucbvax!CAE.W...
2	a_fish_c.apo	From: murph@buscard.fidonet.org (Brian Murphy)...
3	a_tv_t-p.com	____\n...
4	abbott.txt	\n Abbott & Coste...

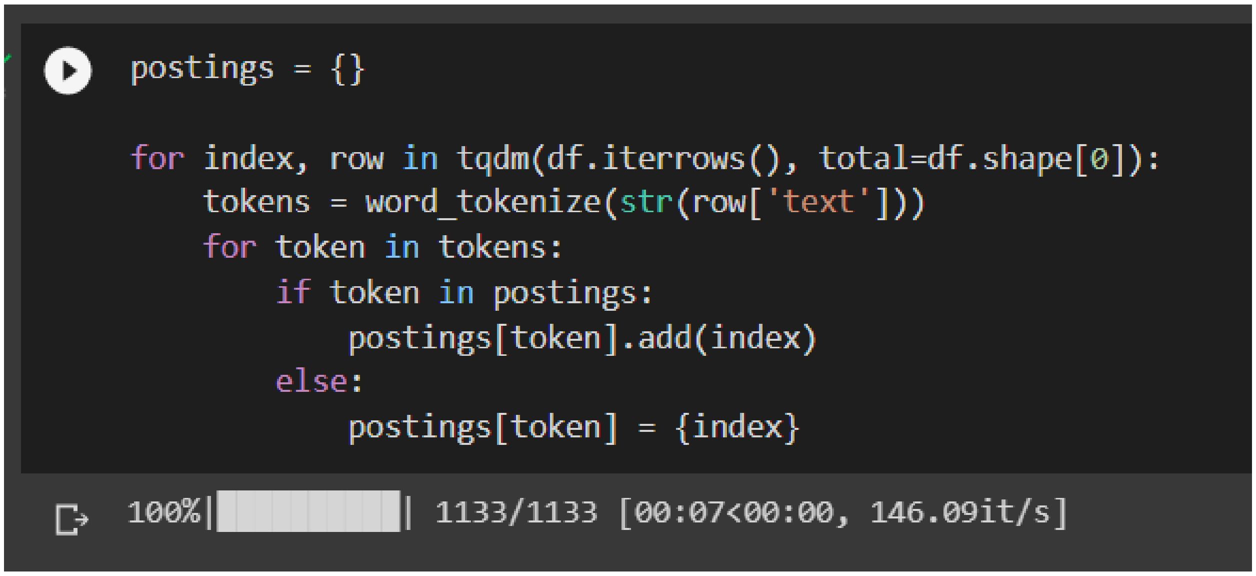
After Preprocessing:



```
100%|██████████| 1133/1133 [00:39<00:00, 28.72it/s]
```

	name	text
0	1st_aid.txt	herbalherb1st aidcalendulacomfreyremediessickm...
1	a-team	uunet c utexas edu usc ucsd ucbvax cae wisc ed...
2	a_fish_c.apo	murph buscard fidonet org brian murphy subject...
3	a_tv_t-p.com	survey result computer use fan alt tv twin pea...
4	abbott.txt	abbott costello first abbott well costello goi...

Unigram Inverted Index Data Structure:
We have implemented this data structure using a dictionary. We simply iterate thorugh the documents (which are already sorted in ascending order) and store the indexes of documents for every word in the corpus in a set. As the order of documents is increasing, the resulting documents are also sorted in ascending order which makes if more efficient while performing union or intersection operations.

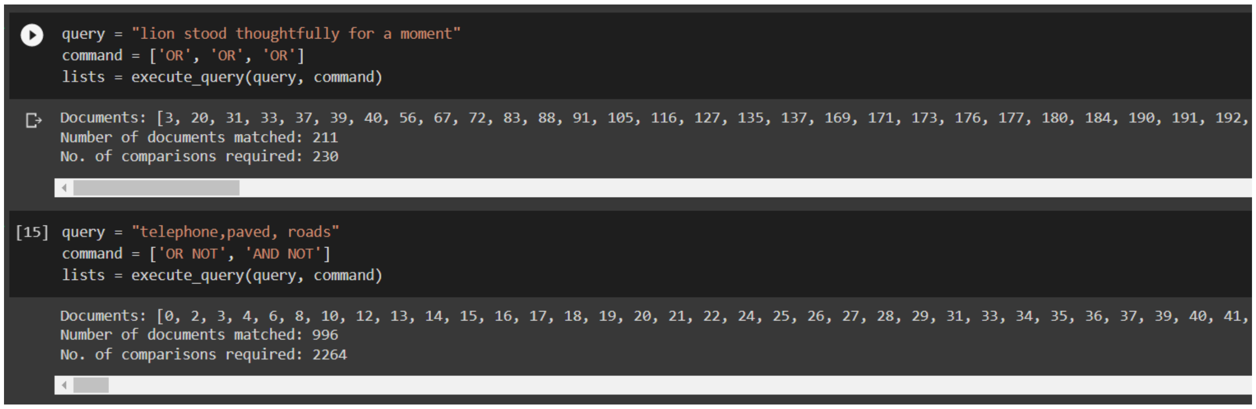


```python
postings = {}

for index, row in tqdm(df.iterrows(), total=df.shape[0]):
    tokens = word_tokenize(str(row['text']))
    for token in tokens:
        if token in postings:
            postings[token].add(index)
        else:
            postings[token] = {index}
```

```
100%|██████████| 1133/1133 [00:07<00:00, 146.09it/s]
```

Finally, We merge the sets according to the input queries and operations. ‘NOT’ operations are done first followed by ‘AND’ and ‘OR’ operations.
The results of the given inputs are as follows:



```
query = "lion stood thoughtfully for a moment"
command = ['OR', 'OR', 'OR']
lists = execute_query(query, command)
```

```
Documents: [3, 20, 31, 33, 37, 39, 40, 56, 67, 72, 83, 88, 91, 105, 116, 127, 135, 137, 169, 171, 173, 176, 177, 180, 184, 190, 191, 192,
Number of documents matched: 211
No. of comparisons required: 230
```

```
[15] query = "telephone,paved, roads"
     command = ['OR NOT', 'AND NOT']
     lists = execute_query(query, command)
```

```
Documents: [0, 2, 3, 4, 6, 8, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 28, 29, 31, 33, 34, 35, 36, 37, 39, 40, 41,
Number of documents matched: 996
No. of comparisons required: 2264
```

Duplicate Runtime:

- Upload zip file of the dataset.
- Run all the cells.

Q2.

I have read all the files given in the dataset. For preprocessing, firstly all the text is converted to lower case text. It is then tokenized using Tweektokenizer and the stopwords are removed form it. The punctuations like bracket, comma, semicolon, colon, etc. are removed. Then it is stripped and any black space token available are removed.

The positional index data structure is created using a dictionary and lists by enumerating the tokens which we have got after preprocessing.

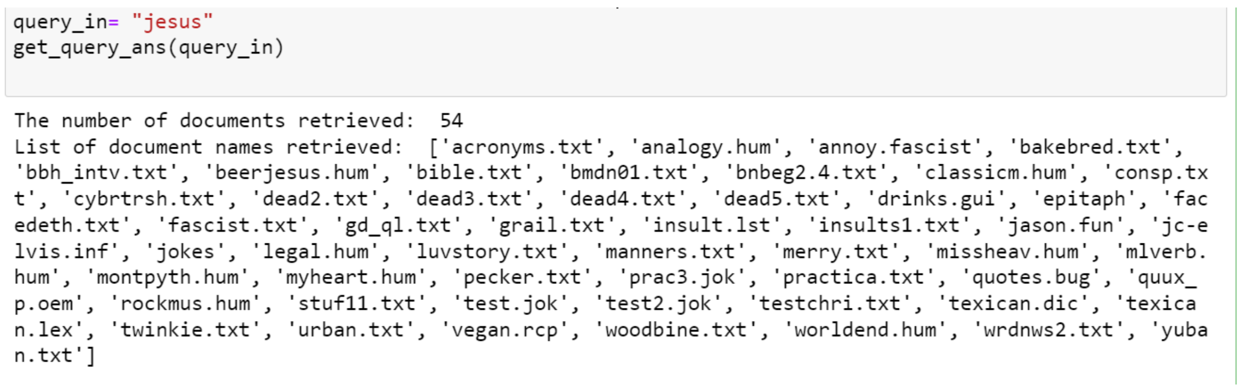
```
tok= token
for count,name in enumerate(tok):
    if name in tokens:
        tokens[name][0]=tokens[name][0]+1
        if doc_id in tokens[name][1]:
            tokens[name][1][doc_id].append(count)
            continue
        tokens[name][1][doc_id]=[count]


    else:
        tokens[name]=[]
        en= 1
        tokens[name].append(en)
        tokens[name].append({})
        input_n+=1
        tokens[name][1][doc_id]=[count]
```

The query is taken as input from the user. The support for the query which is a phrase query is done. If the length of the query word after preprocessing the query word is 1 then only the single word is used to retrieve the documents.

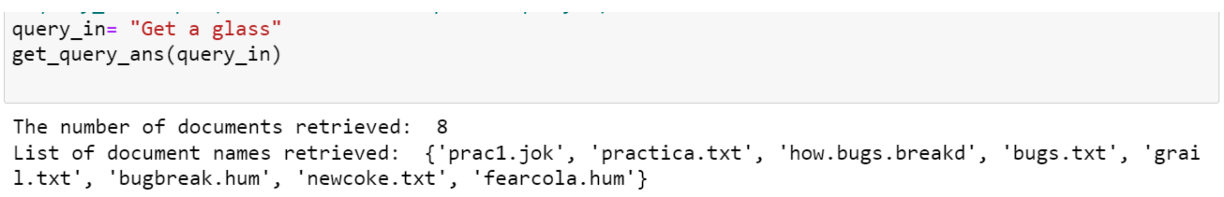
On the other hand, if we have more than one word in the query after preprocessing the query word, then for each of these consecutive words the documents are retrieved as shown in the code.

The output is the number of documents retrieved and the list of document names retrieved.
For example,



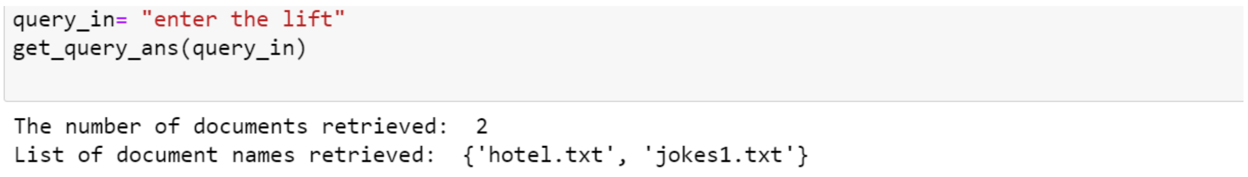
```
query_in= "jesus"
get_query_ans(query_in)
```

```
The number of documents retrieved:  54
List of document names retrieved:  ['acronyms.txt', 'analogy.hum', 'annoy.fascist', 'bakebred.txt',
'bbh_intv.txt', 'beerjesus.hum', 'bible.txt', 'bmdn01.txt', 'bnbeg2.4.txt', 'classicm.hum', 'consp.tx
t', 'cybrtrsh.txt', 'dead2.txt', 'dead3.txt', 'dead4.txt', 'dead5.txt', 'drinks.gui', 'epitaph', 'fac
edeth.txt', 'fascist.txt', 'gd_ql.txt', 'grail.txt', 'insult.lst', 'insults1.txt', 'jason.fun', 'jc-e
lvis.inf', 'jokes', 'legal.hum', 'luvstory.txt', 'manners.txt', 'merry.txt', 'missheav.hum', 'mlverb.
hum', 'montpyth.hum', 'myheart.hum', 'pecker.txt', 'prac3.jok', 'practica.txt', 'quotes.bug', 'quux_
p.oem', 'rockmus.hum', 'stuf11.txt', 'test.jok', 'test2.jok', 'testchri.txt', 'texican.dic', 'texica
n.lex', 'twinkie.txt', 'urban.txt', 'vegan.rcp', 'woodbine.txt', 'worldend.hum', 'wrdnws2.txt', 'yuba
n.txt']
```



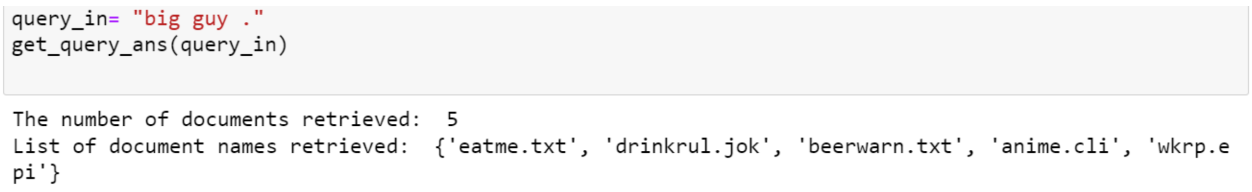
```
query_in= "Get a glass"
get_query_ans(query_in)
```

```
The number of documents retrieved:  8
List of document names retrieved:  {'prac1.jok', 'practica.txt', 'how.bugs.breakd', 'bugs.txt', 'grai
l.txt', 'bugbreak.hum', 'newcoke.txt', 'fearcola.hum'}
```



```
query_in= "enter the lift"
get_query_ans(query_in)
```

```
The number of documents retrieved:  2
List of document names retrieved:  {'hotel.txt', 'jokes1.txt'}
```



```
query_in= "big guy ."
get_query_ans(query_in)
```

```
The number of documents retrieved:  5
List of document names retrieved:  {'eatme.txt', 'drinkrul.jok', 'beerwarn.txt', 'anime.cli', 'wkrp.e
pi'}
```

Referrences:
https://www.geeksforgeeks.org/python-positional-index/
https://github.com/anandsharma26/Information-Retrieval/blob/master/A1_MT19059/Question2_IR.ipynb