Information Retrieval Assignment: 01

Group Members:
Atul (2017032)
Prashant Jain (2018253)
Akanksha Pandey (MT20048)
Shivani Mishra (MT20062)

Methodology:

1.Read Files:

To Read files present in stories folder

*files_list = glob.glob("drive/My Drive/stories/" + '**/*', recursive=True)*

Open a particular file

rfile = open(file_path,'r',errors = 'ignore')

Read content of that file

read_file = rfile.read()

2.Perform Preprocessing steps given below

1. Tokenize the content of the file into sentences using ***sent_tokenize()***.
2. Split each sentence into words on the basis of space character using ***split()***.
3. Convert all the words to lowercase using ***lower().***
4. Expand contractions using ***contractions library.***
5. Remove punctuations and elongated words using ***re library and regular expressions.***
6. Remove Extra spaces using ***re library and regular expressions.***
7. Remove stopwords.
8. Perform lemmatization on the words to find root words using ***WordNetLemmatizer.***

3.To Create Unigram Inverted Index:

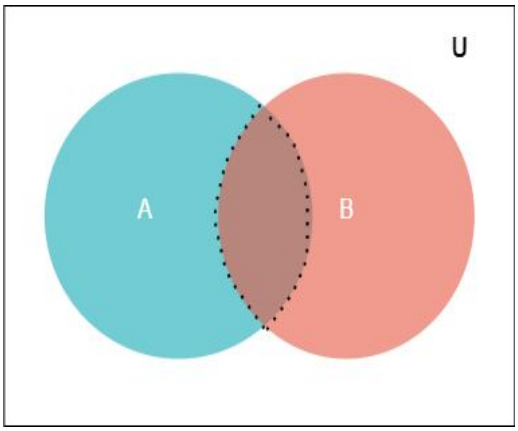
1. The unique_words_final is the list of all of the unique words found in all the documents.
2. A dictionary is used to map all doc id to their file names.
3. The dictionary linked_list_data contains all the words as the keys as the words and their corresponding values is a sorted linked list of all the documents in which the word exists.
4. The pickle package is used to write the above dictionary in a file.

4.Query Processing:

1. Read query and operation sequence.
2. Apply the same preprocessing on the input query that was applied on files.
3. Convert Operations to lower-case.

4. **If operation is AND:**

We are finding the intersection of both the documents.For this operation, compare the first element of both the list then if not equal, skip small, if equal print one and skip both, and finally if one list is over then stop the operation.



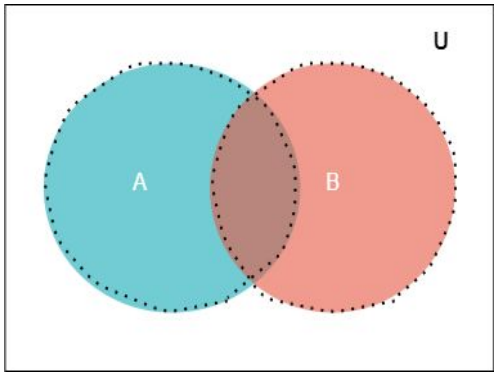

```
Enter Query divide conquer
Enter Operations and
['divide', 'conquer']
aquith.txt
outcast.dos
aesop11.txt
sre-dark.txt

4
22
```

Figure1 : Sample Output : AND

5. **If operation is OR:**

We are finding the concatenation of both the documents. For this operation, compare the first element of both the list then if not equal, print small, if equal print one and skip both, and finally if one list is over then print all elements of the other list.

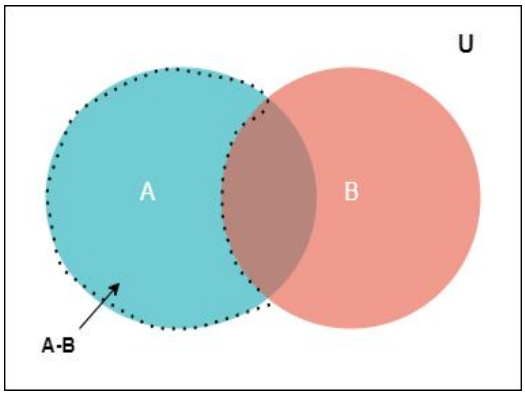



```
Enter Query pipe river
Enter Operations or
['pipe', 'river']
consumdr.hum
timem.hac
candle.hum
rocket.sf
enchdup.hum
vday.hum
elite.app
cooldark.sto
archive
wolf7kid.txt
gold3ber.txt
```

Figure2: Sample Output : OR

6. **If operation is And Not:**

AND NOT operation contains the 1st list but not the 2nd one. A AND NOT B = A-B



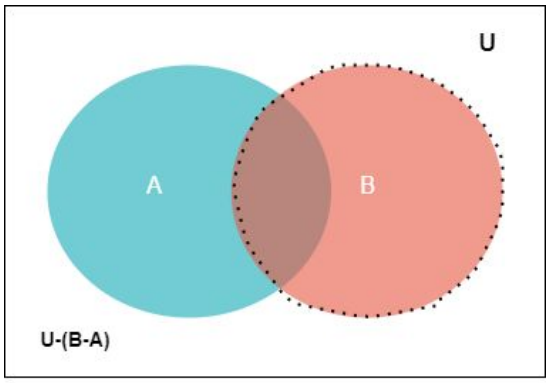

```
Enter Query divide conquer
Enter Operations and-not
['divide', 'conquer']
rocket.sf
goldbug.poe
gulliver.txt
yukon.txt
quest
beggars.txt
batlslau.txt
radar_ra.txt
dicegame.txt
history5.txt

10
22
```

Figure 3 : Sample Output : AND-NOT

7. **If operation is Or Not:**

We have used the following formula to perform this operation: A OR NOT B = U - (B-A)




```
Enter Query pipe river
Enter Operations or-not
['pipe', 'river']
nitepeek.sto
blind.txt
partya.txt
life.txt
cameloto.hum
sight.txt
snow.txt
beyond.hum
```

Figure 4: Sample Output : OR-NOT

Assumptions:

1. We have removed all the unnecessary files while reading. Total File count after reading was : 467.
2. It is assumed that while giving input for operations OR NOT and AND NOT there will be a '-' in between. For example *OR-NOT, And-Not.*
3. Queries will be spaced separated.
4. Operators will be spaced separated.

Outputs:

Expected Outputs :

1)Input query: lion stood thoughtfully for a moment
Input operation sequence: [OR, OR , OR]

Expected query after preprocessing: lion OR stood OR thoughtfully OR moment
Output-
Number of documents matched: 270
No. of comparisons required: 671

Output of our function :

```
Number of documents matched:  270
No. of comparisons required:  677
```

```
Enter Query lion stood thoughtfully for a moment
Enter Operations OR OR OR
Input Query ['lion', 'stood', 'thoughtfully', 'moment']
Input Operation Sequence [ OR OR OR ]
nitepeek.sto
blind.txt
sight.txt
snow.txt
beyond.hum
consumdr.hum
tree.txt
aluminum.hum
timem.hac
spiders.txt
corcor.hum
rocket.sf
game.txt
enchdup.hum
ladylust.hum
immorti.hum
vday.hum
elite.app
eyeargon.hum
cooldark.sto
archive
imagin.hum
testpilo.hum
adv_alad.txt
emperor3.txt
empnclot.txt
```

Expected Outputs :

2)Input query: telephone,paved, roads
Input operation sequence: [OR NOT, AND NOT]

Expected query after preprocessing: telephone OR NOT paved AND NOT roads
Output-
Number of documents matched: 466
No. of comparisons required: 739

Output of our function :

```
Number of documents matched:  347
No. of comparisons required:  871
```

```
Enter Query telephone, paved, roads
Enter Operations OR-NOT AND-NOT
Input Query ['telephone', 'paved', 'road']
Input Operation Sequence [ OR-NOT AND-NOT ]
life.txt
cameloto.hum
beyond.hum
timem.hac
contrad1.hum
corcor.hum
rocket.sf
game.txt
excerpt.txt
ladylust.hum
immorti.hum
elite.app
eyeargon.hum
fantas.hum
imagin.hum
confilct.fun
testpilo.hum
advtthum.txt
elveshoe.txt
wolfcran.txt
adv_alad.txt
narciss.txt
emperor3.txt
aircon.txt
empnclot.txt
wolf7kid.txt
wolflamb.txt
```