README/REPORT
ASSIGNMENT-1

❑ Assumptions

- The given contains 467 files but in our assignment, we have taken 472 files. The additional files are the content of other folders of the same given link.

- We have used the NLTK library for data pre-processing steps.

❑ Data Pre-processing steps

- Imported the required NLTK library for data preprocessing.

- The sentence in the file may have stop-words, punctuation symbols, and special characters. For the query processing, they need to be removed from the sentence.

- The special characters such as “ ‘ “ etc. are removed and joined the result as string using the code:

```
remove_digits = str.maketrans('', '', digits)
    read = read.translate(remove_digits)

    read = read.replace("""","")

    res = re.findall(r'\w+', read)
    res = " ".join(res)
```

- In the continuation, tokenization takes place. Each word is considered a token. Code used:

```
read = []
    words = word_tokenize(res)
```

- NLTK already contains stop words which are used in the code to identify what are the stopwords in the sentences of the file and need to be removed.

  Code for Stopword removal:

```
for w in words:
        if not w in stopwords.words():
        read.append(ps.stem(w))
```

❑ Methodology

After the completion of data pre-processing for all the files, Now the next step is to create the inverted index data structure.

❖ Construction of Inverted index data structure:

- For a word in the data structure, it will have a list of documents ID's that contains the word.

- Therefore it will iterate for each word in each sentence of each file, and check if the word exists in data structure (INDEX) or not.

- For a particular file, we have a list of words after data processing. Iterating over that list to check if a word is already in inverted index data structure (INDEX) or not. If the word exists in INDEX already, then a particular current/file ID is appended in the list for the word in INDEX.

- The INDEX (inverted index data structure) is a dictionary with words as key and list of documents ID's containing the word as value.

- Thus iterating over the words list for each file and referring to INDEX, appending the documents ID's, the Inverted index data structure is prepared.

- We have also created another dictionary known as MAP that contains the ID of the document as key and their corresponding names as values.

Note: After construction of INDEX and MAP dictionaries, we have saved it for computational purposes using pickle.

❖ Query processing

- The user is asked to define the query.
- The input query is taken for pre-processing that we have discussed in the above sections.
- After the query pre-processing, we have a list of words.
- The user is then asked to input a number of queries to execute.
- Thus the system will produce the results for input number of queries.
- The user is then asked to enter the input operation sequence.
- The next task is to insert the defined operations in between the words in words list for further processing.
- The results will have a list that contains the number of words and the operations in between them. Each operation will have both left and right operands (word).
- Furthermore, the precedence of the operators/operations plays an important role. Precedence : Not > And > OR.
- The higher priority operation is operated first and its results are then operated with the next lower priority operation and so on till we get a single list as result.
- While performing operation between the words, the minimum number of comparisons are also calculated. Code for the minimum number of comparisons is present in the code file.
- The final operated list contains the number of documents retrieved against the input query.
- The boundary case such as when there exists only a single word in word list after preprocessing of input query, then no operations will be performed. For no words in the input query, similar action is being taken.

- In the result, with the help of a MAP dictionary that stores the name of documents against the document ID, the document names are also printed in the output for the query result.

- The output contains the number of documents retrieved, name of the documents retrieved and number of minimum comparisons against the input query.

❖ Inbuilt/User defined functions used

- **min_no_of_comparisons :** It takes two lists as arguments. The list corresponds to the words for which are performing operation. The output function is the minimum number of comparisons required.

- **Word_tokenize:** It takes a string as input and results a list of tokens/words.

- **Pickle.dump:** It is used to save the results into a file for later purpose.

- **Pickle.load:** It is used to load the results from the saved file to a file for use.

- **Lambda:** It is a general function available in python.

- **Symmetric_difference**

- Basic functions of list such as append, insert.

❖ **Observed Results**

Sample 1.

```
Enter no of queries
3
Query 1

Sentence : Hello everyone shall we go to movie tonight.
Please enter 5 operations

Operation : and,or not,and not,and,and
No of documents : 380
Documents list :
fantas.hum
lgoldbrd.txt
bestwish
encamp01.txt
buldream.txt
ladylust.hum
assorted.txt
```

```
nigel.2
weeprncs.txt
quarter.c16
myeyes
non4
sleprncs.txt
perf
rock
excerpt.txt
blackrdr
hitch2.txt
immortal
floobs.txt
radar_ra.txt
prince.art
fea3
Minimum no of comparisons required are 580
```

Sample 2.

```
Sentence : Kapil is going for ISRO interview on 4th March.
Please enter 5 operations

Operation : and,or,and not,or not,and
No of documents : 454
Documents list :
fantas.hum
crabhern.txt
lgoldbrd.txt
bestwish
encamp01.txt
buldream.txt
ladylust.hum
assorted.txt
rocket.sf
non2
game.txt
perf
rock
excerpt.txt
timem.hac
blackrdr
hitch2.txt
6napolen.txt
hop-frog.poe
immortal
floobs.txt
radar_ra.txt
keepmodu.txt
prince.art
psf.txt
sre09.txt
fea3
Minimum no of comparisons required are 65
```

Sample 3.

```
Query 3

Sentence : "If you can't explain it simply, you don't understand it well
enough."
Please enter 7 operations

Operation : and not,or,or not,or not,and,and not,or not
No of documents : 471
Documents list :
fantas.hum
crabhern.txt
lgoldbrd.txt
bestwish
encamp01.txt
buldream.txt
ladylust.hum
assorted.txt
```

```
rock
excerpt.txt
timem.hac
blackrdr
hitch2.txt
6napolen.txt
hop-frog.poe
immortal
floobs.txt
radar_ra.txt
keepmodu.txt
prince.art
psf.txt
5orange.txt
sre09.txt
fea3
Minimum no of comparisons required are 702
```