Readme File

Initially, follow the below sequence for executing the code

- Upload all the dataset files on google colab.
- Change the directory to the drive folder where all the files are

```
[3] from google.colab import drive
    drive.mount('/content/drive/')

    Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).

[4] import os
    path ='/content/drive/MyDrive/Assignment-1-IR-F'
    os.chdir(path)
```

present as displayed below:

Question 1:

This question is in the code file **'IR_Assignement1_Q1.ipynb'**

Pre-processing:

- Import libraries then Read the dataset
- We first tokenize the terms and then take each term of the sentence and convert it into lower case

```
def lower_case(text):
    lower_case_text = text.lower()
    return lower_case_text


def tokenize(text):
  tokenizer = TweetTokenizer()
  token_list = tokenizer.tokenize(text)
  return token_list
```

- Then we remove the special characters and extra blank space
- After that, we remove the stop words

```
def stop_word(text):
  sentence = []
  stop_words = set(stopwords.words("english"))
  for w in text:
        if w not in stop_words:
            sentence.append(w)
        else:
          continue
  return sentence

def remove_punc(text):
  punc_tokenizer = nltk.RegexpTokenizer(r"\w+")
  text = punc_tokenizer.tokenize(text)
  return text

def tokenizeReg(text):
    tok=RegexpTokenizer('[A-Za-z0-9]?\w+')
    return tok.tokenize(text)
```

- After that, we lemmatize the term.

```
[12] def lemmatization(file):
       lemmatizer = WordNetLemmatizer()
       for i in range(0,len(file)):
         lemma = lemmatizer.lemmatize(file[i])
         file[i] = lemma
       return file
```

For each part of this question that needs to be implemented, the question part is mentioned using a text cell at the beginning of the question part in the notebook file. These subparts can be executed after performing the pre-processing steps.

Assumptions:

- The input of operations is taken in the form of a string separated by a comma in part D.
- We're not using any stemming here instead lemmatization is used

Methodology:

- Firstly, the pre-processing is done on all the files and an inverted index is created with the help of a dictionary named '`finalDict`'
- For part C, different operation methods are created as shown for the 'OR' operation, and before these operations pre-processing is performed on the input queries

```
def OR(input1, input2):
    if type(input1) == str:
      input1 = finalDict.get(input1)
      input1 = set(input1[1])
    elif type(input1) == list:
      input1 = set(input1)

    if type(input2) == str:
      input2 = finalDict.get(input2)
      input2 = set(input2[1])
    elif type(input2) == list:
      input2 = set(input2)

    print('\nDocuments retrieved are:',input1.union(input2))
    print('\nNo. of documents retrieved after performing OR operation:',len(input1.union(input2)))
```

- For part D, a method named '`comparizon`' is created which returns the number of comparisons made, size of the result list, and result list after performing the corresponding operations.

Question 2:

This question is in the code file **'IR_A!_Q2_G86.ipynb'**
Follow the cell sequence for the execution.

Pre-processing:

- Import libraries then Read the dataset
- We first tokenize the terms
- Then we remove the special characters and extra blank space
- After that, we remove the stop words

- Then we take each term of the sentence and convert it into lower case
- After that, we lemmatize the term.

So these are the preprocessing, which we are doing in this question.

Methodology:

All the preprocessing are done in a separate function.
We are having two main functions the first one is positionalIndex and the second is executeQuery. The position index function is returning the dictionary having word position with its corresponding term and the next one is executeQuery, which helps us to execute the query or a phrase query.

Code snippets of positionalIndex

```
def positionalIndex():
 terms = {}
 files = getFilePath()

 for i in range(len(files)):
    # File Path
     path = files[i]

   # Word Position
     wordPos = 0

   # Document ID
     docId = files[i]
```

To execute the query see this snippet.

```
query = str(input('Enter Query:'))
finalResult = executeQuery(query)
print("Number of Documents Retrived are as Follows:" ,len(finalResult))
finalResult
```

Assumptions:

- We are not doing stemming here.
- Also, we are not removing the numbers if it is present in the sentence, as a user may want to search for a phone number. So keeping that in mind we are only removing the numbers which are present between the sentences.