INFORMATION RETRIEVAL

Assignment 1

Group Members :
Rohit Kesarwani (MT22059)
Sushil Kumar(MT22077)
Shambhavi Sharma (MT22066)

LIBRARIES USED

- **os :** os is used for importing dataset folders from directory.
- **NLTK :** The Natural Language Toolkit (nltk) is used for performing tasks such as: Tokenization, Part-of-speech tagging, Sentiment Analysis,Stemming and Lemmatization etc.
- **BeautifulSoup :** It is used for web scraping and parsing HTML and XML documents, providing a convenient and flexible way to extract and manipulate data from web pages.
- **re :** Provides operations of detecting patterns with the help of regular expressions.
- **json :** Using json we can easily convert between Python objects and JSON data, making it easier to exchange data between applications written in different programming languages. The json module provides a method for encoding Python objects into JSON format: json.dumps() used to return a string representation of a JSON-encoded object.
- **pickle :** The pickle module in Python is used for serialising and deserializing Python objects.
- **string :** For working with text data present in dataset files.

PREPROCESSING

DATA UPLOADING :
Import files from the device.

DATA UNDERSTANDING:
1. Read data using os.listdir("file location").
2. Checked the total count of files in the folder. The number of files in the folder is 1400.
3. A dictionary is created with **docId** name. The key contains the document number and the value contains the file name.
4. The key and value of the dictionary are taken and a dataframe is created for storing the filenames.

PRE-PROCESSING STEPS ON THE DATASET:
1. Converted the complete text in all the files to lowercase using the .lower() function.
2. Then performed tokenization using the .word_tokenize() method..
3. Removing stopwords from the tokenized text file using stopwords module.
4. Used re.sub() to remove the punctuation marks from the text files.
5. Then used re.sub() to remove the blank spaces from all the files in the folder

PRE-PROCESSING on the INPUT query :
1. Convert the query to lower case.
2. Perform tokenization.
3. Remove stopwords
4. Remove punctuation marks.
5. Strip all the blank spaces from the query.

QUESTION 2

FUNCTION DEFINITIONS FOR OR, AND, NOT, AND NOT and OR NOT

AND FUNCTION IMPLEMENTATION :

1. The document names in which terms are matched are stored in a list called **answer.** Initially the list does not contain any document names i.e it's empty.
2. The documents of the two posting lists term wise are compared with each other and the iteration is performed until one of the lists completes.
3. The iterator puts the document number in the output list and moves forward if the document numbers are matched with each other.
4. The number of comparisons are counted as the iterator moves and comparisons are done.
5. The function returns the count of the comparisons and the list **answer** storing the document matched.

OR FUNCTION IMPLEMENTATION :

1. The document names in which terms are matched are stored in a list called **answer.** Initially the list does not contain any document names i.e it's empty.
2. The document numbers matching is performed and the document is added to the list called **answer** if the document is matched and the iterator is moved forward for both the lists.
3. If the document numbers don't match the smaller document number is added to the **answer** list and the iterator skips that document number in that particular posting list.
4. The number of comparisons are counted as the iterator moves and comparisons are done.
5. The function returns the count of the comparisons and the list **answer** storing the document matched.

NOT FUNCTION IMPLEMENTATION :

1. Documents which do not contain any terms or the document which are not present in the posting list are the outputs returned by the NOT function.
2. The document numbers present are matched with the document numbers present in the posting lists and the count for comparisons is increased.
3. The document counts are added to the **answer** list if it is not present in the posting list.

4. It will return zero in the case when all the document numbers are present in the posting list i.e **answer** list will not contain any document.

ANDNOT FUNCTION IMPLEMENTATION :

1. AND NOT function is the combination of AND and NOT functions.
2. We start with implementing the NOT function first which is implemented in one of the posting lists(list2).
3. Now the AND function implementation is done and is applied on posting list1 and not posting list obtained from not function and the "answer" list contains the result of the combination.
4. Here the comparisons are the total count of NOT function implementation and AND function implementation.

ORNOT FUNCTION IMPLEMENTATION :

1. OR NOT function is the combination of OR and NOT function together.
2. We start with implementing the NOT function first which is implemented in one of the posting lists(list2).
3. Now the OR function implementation is done and is applied on posting list1 and not posting list obtained from not function and the output list contains the result of the combination.
4. Here the comparisons are the total count of NOT function implementation and OR function implementation.

Some points to remember about other functions:

- There is a helper function defined which takes which operation is needed to perform on the posting lists.
- The output list returns the document obtained and the total comparison returns the number of comparisons after applying the functions respectively.
- The preprocessing of the input query is also done which is passed by the user.

DATA UPLOADING :
Import files from the device.

PRE-PROCESSING STEPS :
1. Converted the complete text in all the files to lowercase using the .lower() function.
2. Then performed tokenization using the .word_tokenize() method..
3. Removing stopwords from the tokenized text file using stopwords module .
4. Used re.sub() to remove the punctuation marks from the text files.

5. Then used re.sub() to remove the blank spaces from all the files in the folder

```
In [48]: driver_code()
```

```
Enter No. of Queries
1
Enter your query:
simple shear
Enter  1  operations needed to be performed
OR
Query  1
simple OR shear

Number of documents retrieved for query  1 :  284


Names of the documents retrieved for query  1 :  ['cranfield0002', 'cranfield0003', 'cranfield0024', 'cranfield0029', 'cranfiel
d0033', 'cranfield0034', 'cranfield0039', 'cranfield0044', 'cranfield0045', 'cranfield0049', 'cranfield0054', 'cranfield0055',
'cranfield0060', 'cranfield0064', 'cranfield0068', 'cranfield0085', 'cranfield0089', 'cranfield0091', 'cranfield0092', 'cranfie
ld0093', 'cranfield0094', 'cranfield0101', 'cranfield0106', 'cranfield0115', 'cranfield0120', 'cranfield0122', 'cranfield0128',
'cranfield0134', 'cranfield0140', 'cranfield0147', 'cranfield0151', 'cranfield0158', 'cranfield0176', 'cranfield0182', 'cranfie
ld0187', 'cranfield0191', 'cranfield0193', 'cranfield0196', 'cranfield0205', 'cranfield0223', 'cranfield0225', 'cranfield0227',
'cranfield0229', 'cranfield0244', 'cranfield0245', 'cranfield0264', 'cranfield0280', 'cranfield0281', 'cranfield0283', 'cranfie
ld0291', 'cranfield0294', 'cranfield0317', 'cranfield0319', 'cranfield0329', 'cranfield0334', 'cranfield0336', 'cranfield0343',
'cranfield0352', 'cranfield0359', 'cranfield0369', 'cranfield0370', 'cranfield0372', 'cranfield0375', 'cranfield0377', 'cranfie
ld0379', 'cranfield0383', 'cranfield0388', 'cranfield0389', 'cranfield0395', 'cranfield0397', 'cranfield0404', 'cranfield0406',
'cranfield0417', 'cranfield0447', 'cranfield0456', 'cranfield0459', 'cranfield0461', 'cranfield0467', 'cranfield0469', 'cranfie
ld0472', 'cranfield0474', 'cranfield0480', 'cranfield0482', 'cranfield0495', 'cranfield0497', 'cranfield0499', 'cranfield0503',
'cranfield0513', 'cranfield0514', 'cranfield0515', 'cranfield0520', 'cranfield0549', 'cranfield0561', 'cranfield0564', 'cranfie
ld0567', 'cranfield0575', 'cranfield0580', 'cranfield0585', 'cranfield0606', 'cranfield0607', 'cranfield0630', 'cranfield0648',
'cranfield0650', 'cranfield0652', 'cranfield0656', 'cranfield0660', 'cranfield0663', 'cranfield0676', 'cranfield0677', 'cranfie
ld0685', 'cranfield0686', 'cranfield0687', 'cranfield0703', 'cranfield0719', 'cranfield0727', 'cranfield0730', 'cranfield0753',
'cranfield0757', 'cranfield0777', 'cranfield0798', 'cranfield0799', 'cranfield0811', 'cranfield0823', 'cranfield0824', 'cranfie
ld0825', 'cranfield0829', 'cranfield0831', 'cranfield0833', 'cranfield0844', 'cranfield0846', 'cranfield0848', 'cranfield0852',
'cranfield0867', 'cranfield0869', 'cranfield0873', 'cranfield0881', 'cranfield0885', 'cranfield0888', 'cranfield0901', 'cranfie
ld0903', 'cranfield0908', 'cranfield0910', 'cranfield0914', 'cranfield0916', 'cranfield0927', 'cranfield0928', 'cranfield0940',
'cranfield0952', 'cranfield0961', 'cranfield0974', 'cranfield1003', 'cranfield1020', 'cranfield1024', 'cranfield1037', 'cranfie
ld1041', 'cranfield1044', 'cranfield1046', 'cranfield1063', 'cranfield1068', 'cranfield1070', 'cranfield1074', 'cranfield1079',
```

QUESTION 3

DATA UPLOADING :
Import files from the device.

PRE-PROCESSING STEPS :
1. Converted the complete text in all the files to lowercase using the .lower() function.
2. Then performed tokenization using the .word_tokenize() method..
3. Removing stopwords from the tokenized text file using stopwords module .
4. Used re.sub() to remove the punctuation marks from the text files.
5. Then used re.sub() to remove the blank spaces from all the files in the folder

Bigram Inverted Index :

1. Creating a dictionary docId for storing document number and file name.
2. Creating a bigram_dict dictionary which stores bigrams and then we convert them into lists .
3.Then created an inverted_index dictionary to store bigram inverted index and saved it into a bigram_inverted_index.pkl.

Implementing the Bigram Inverted indexing :

1. If the number of bigrams generated is 0 then the output will show 0 documents as the number of documents retrieved.
2. If the number of bigrams generated is 1 then all documents will be searched for 1 word phrase or token generated and the document names will be displayed along with their Ids.
3. If the number of bigrams generated is 2 then all documents will be searched for 2 word phrases or token generated and the document names will be displayed along with their Ids.
4. Similar steps will be followed for 3, 4, 5 bigrams generated respectively.
5. If the number of tokens generated are more than 5, then the function will not be executed because the query length exceeds 5.
6. If there are random phrases provided by the user which are not present in the document, then invalid phrase message will be displayed.

Positional index:

1. Created dictionary for storing file names along with index numbers in nameDictionary.

2. Created a dictionary positional_index_dict for storing the positional index of terms and then stored it into a positional_index.pkl file.
3. The input query is taken from the user and preprocessing is also done on the input query.

Implementing the positional indexing:

1. If the number of tokens generated is 0 then the output will show 0 documents as the number of documents retrieved.
2. If the number of tokens generated is 1 then all documents will be searched for 1 word phrase or token generated and the document names will be displayed along with their Ids.
3. If the number of tokens generated is 2 then all documents will be searched for 2 word phrases or token generated and the document names will be displayed along with their Ids.
4. Similar steps will be followed for 3, 4, 5 tokens generated respectively.
5. If the number of tokens generated are more than 5, then the function will not be executed because the query length exceeds 5.
6. If there are random phrases provided by the user which are not present in the document, then invalid phrase message will be displayed.

PRE-PROCESSING on the INPUT query :
1. Convert the query to lower case.
2. Strip all the white spaces from the query.
3. Remove punctuation marks.
4. Perform tokenization.
5. Remove stopwords