IR Assignment-1

Group Members –

Ankit Talreja Sahitya(MT22012)
Ashutosh Choubey(MT22020)
Arohi Shrivastava(MT22017)

Libraries used in Assignment :

- **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.
- **re :** Provides operations of detecting patterns with the help of regular expressions.
- **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.

UNDERSTANDING OF DATA:
1. Read data using path from drive.
2. Checked the total count of files in the folder. The number of files in the folder is 1400.
3. A list is created of all the files .
4. The key and value of the dictionary are taken and a dataframe is created for storing the filenames.

PRE-PROCESSING 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. remove the punctuation marks from the text files.
5. 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

AND FUNCTION IMPLEMENTATION :

1. The document names in which terms are matched are stored in a list called **document_sets.** Initially the list does not contain any document names i.e it's empty.
2. Intersect operation is performed between the two lists taken from the document_sets list and comparisons needed is also calculated.
3. The function returns the count of the comparisons and the list **results** storing the document matched.

OR FUNCTION IMPLEMENTATION :

1. The document names in which terms are matched are stored in a list called **document_sets.** Initially the list does not contain any document names i.e it's empty
2. Union operation is performed between the two lists taken from the document_sets list and comparisons needed is also calculated.
3. The function returns the count of the comparisons and the list **results** storing the document matched.

ANDNOT FUNCTION IMPLEMENTATION :

1. AND NOT operation is the combination of AND and NOT operation.
2. We start with implementing the NOT operation first which is implemented as complement() method.
3. Now the AND operation is done and is applied on list1 and complemented list obtained from complement function and the “result” list contains the result of the combination.
4.Here also the comparisons are calculated simultaneously using comparison_and() function.

ORNOT FUNCTION IMPLEMENTATION :

1. OR NOT operation is the combination of OR and NOT operation.
2. We start with implementing the NOT operation first which is implemented as complement() method.
3. Now the OR operation is done and is applied on list1 and complemented list obtained from complement function and the “result” list contains the result of the combination.
4.Here also the comparisons are calculated simultaneously using comparison_and() function.

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. Then remove the punctuation marks from the text files.
5. Then remove the blank spaces from all the files in the folder

Here is the sample output from question 2 :

```
2
simple shear is a flow
AND,OR
['simple', 'shear', 'flow']
Query 1
Number of documents retrieved for query 1 : 673
Names of the documents retrieved for query 1 : ['cranfield0098', 'cranfield0124', 'cranfield0132', 'cranfield0269', 'cranfield0057', 'cranfield1325', 'cranfield0270', 'cranfield0299',
Number of comparisons required for query 1 : 919

transient heat conduction
OR,AND NOT
['transient', 'heat', 'conduction']
Query 2
Number of documents retrieved for query 2 : 209
Names of the documents retrieved for query 2 : ['cranfield0262', 'cranfield0098', 'cranfield1217', 'cranfield0269', 'cranfield1281', 'cranfield1012', 'cranfield0270', 'cranfield1381',
Number of comparisons required for query 2 : 1604
```

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. Then remove the punctuation marks from the text files.
5. Then 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 list which stores all the bigrams using create_bigrams() function
3.Then created an inverted_index dictionary to store bigram inverted index and saved it into a bigram_index_file.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 documents[]
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

```
1
Enter querysimple is shear
Name of documents retrieved from query 1 using bigram inverted index: cranfield0002,cranfield0003,cranfield0389,
Number of documents retrieved from query 1 using bigram inverted index: 3
Name of documents retrieved from query 1 using positional index: cranfield0002,cranfield0003,cranfield0024,cranfield0029,cranfi
eld0033,cranfield0034,cranfield0039,cranfield0044,cranfield0045,cranfield0049,cranfield0054,cranfield0055,cranfield0060,cranfie
ld0064,cranfield0085,cranfield0089,cranfield0091,cranfield0092,cranfield0101,cranfield0106,cranfield0115,cranfield0120,cranfiel
d0128,cranfield0134,cranfield0140,cranfield0151,cranfield0158,cranfield0176,cranfield0187,cranfield0193,cranfield0196,cranfield
0205,cranfield0225,cranfield0227,cranfield0229,cranfield0244,cranfield0245,cranfield0280,cranfield0281,cranfield0283,cranfield0
294,cranfield0317,cranfield0319,cranfield0334,cranfield0336,cranfield0343,cranfield0352,cranfield0359,cranfield0369,cranfield03
70,cranfield0375,cranfield0377,cranfield0383,cranfield0388,cranfield0389,cranfield0395,cranfield0397,cranfield0404,cranfield041
7,cranfield0447,cranfield0456,cranfield0459,cranfield0467,cranfield0469,cranfield0480,cranfield0482,cranfield0495,cranfield049
7,cranfield0499,cranfield0503,cranfield0513,cranfield0514,cranfield0515,cranfield0520,cranfield0549,cranfield0564,cranfield056
7,cranfield0575,cranfield0580,cranfield0585,cranfield0606,cranfield0630,cranfield0648,cranfield0650,cranfield0652,cranfield065
6,cranfield0660,cranfield0663,cranfield0676,cranfield0677,cranfield0685,cranfield0686,cranfield0687,cranfield0727,cranfield073
0,cranfield0753,cranfield0757,cranfield0777,cranfield0798,cranfield0799,cranfield0811,cranfield0823,cranfield0824,cranfield082
5,cranfield0829,cranfield0831,cranfield0833,cranfield0848,cranfield0852,cranfield0869,cranfield0873,cranfield0881,cranfield088
5,cranfield0888,cranfield0901,cranfield0903,cranfield0908,cranfield0910,cranfield0916,cranfield0927,cranfield0928,cranfield094
0,cranfield0952,cranfield0961,cranfield0974,cranfield1003,cranfield1020,cranfield1024,cranfield1037,cranfield1041,cranfield104
6,cranfield1068,cranfield1070,cranfield1074,cranfield1087,cranfield1123,cranfield1125,cranfield1127,cranfield1129,cranfield113
3,cranfield1140,cranfield1187,cranfield1193,cranfield1197,cranfield1202,cranfield1204,cranfield1211,cranfield1219,cranfield122
9,cranfield1231,cranfield1238,cranfield1248,cranfield1304,cranfield1305,cranfield1311,cranfield1313,cranfield1333,cranfield134
0,cranfield1347,cranfield1356,cranfield1360,cranfield1386,cranfield1387,cranfield1388,cranfield1396,cranfield0004,cranfield000
9,cranfield0016,cranfield0065,cranfield0099,cranfield0109,cranfield0116,cranfield0121,cranfield0126,cranfield0165,cranfield017
1,cranfield0191,cranfield0192,cranfield0255,cranfield0306,cranfield0324,cranfield0329,cranfield0365,cranfield0366,cranfield039
3,cranfield0398,cranfield0400,cranfield0412,cranfield0418,cranfield0419,cranfield0452,cranfield0453,cranfield0484,cranfield049
1,cranfield0538,cranfield0550,cranfield0629,cranfield0659,cranfield0664,cranfield0720,cranfield0820,cranfield0826,cranfield085
4,cranfield0889,cranfield0929,cranfield0943,cranfield0953,cranfield0976,cranfield1034,cranfield1038,cranfield1048,cranfield105
0,cranfield1106,cranfield1117,cranfield1119,cranfield1121,cranfield1128,cranfield1130,cranfield1173,cranfield1215,cranfield122
7,cranfield1233,cranfield1237,cranfield1244,cranfield1250,cranfield1251,cranfield1275,cranfield1302,cranfield1366,cranfield139
2,cranfield1397,cranfield1398,cranfield1399,cranfield1400,
Number of documents retrieved from query 1 using positional index: 247
```