IR Report Assignment - 1

Mohammad Aflah Khan, 2020082
Neemesh Yadav, 2020529

Ans 1) For the first part, we first read each file's data and searched for the RegEx patterns `"<TITLE>(.*)</TITLE>"` and `"<TEXT>(.*)</TEXT>"`, to extract all the text written between the TITLE and TEXT tags as is described in the problem statement.
For the second part, we followed the same preprocessing pipeline as is described in the problem statement. We used the NLTK library for preprocessing, wherein we majorly used the NLTK Tokenizer for tokenizing the data, and the NLTK corpus for getting the stopwords in English language. We have also used the NLTK TreebankWordDetokenizer to convert the tokenized data back to its free text format.
One assumption that we have followed while removing the punctuations is that, we have not changed those tokens which contain the hyphen punctuation mark, and we have also followed the same preprocessing pipeline for processing the queries in the proceeding tasks. We hypothesize that by doing so we're keeping the tokens like boundary layer, and boundary-layer independent.

Ans 2)
Input -

Input Sequence: 'reynolds number and potential shear'
Operations: 'and, or not, and'

Output -

Query 1: reynolds AND number OR NOT potential AND shear
Number of documents retrieved for query 1: 88
Names of the documents retrieved for query 1: ['cranfield0002', 'cranfield0003', 'cranfield0004', 'cranfield0009', 'cranfield0016', 'cranfield0045', 'cranfield0050', 'cranfield0065', 'cranfield0088', 'cranfield0089', 'cranfield0099', 'cranfield0109', 'cranfield0116', 'cranfield0121', 'cranfield0126', 'cranfield0165', 'cranfield0171', 'cranfield0180', 'cranfield0187', 'cranfield0191', 'cranfield0192', 'cranfield0255', 'cranfield0268', 'cranfield0306', 'cranfield0324', 'cranfield0329', 'cranfield0365', 'cranfield0366', 'cranfield0388', 'cranfield0389', 'cranfield0393', 'cranfield0397', 'cranfield0398', 'cranfield0400', 'cranfield0412', 'cranfield0418', 'cranfield0419', 'cranfield0452', 'cranfield0453', 'cranfield0484', 'cranfield0491', 'cranfield0517', 'cranfield0538', 'cranfield0550', 'cranfield0629', 'cranfield0659', 'cranfield0660', 'cranfield0664', 'cranfield0720', 'cranfield0820', 'cranfield0826', 'cranfield0854', 'cranfield0889', 'cranfield0929', 'cranfield0940', 'cranfield0943', 'cranfield0953', 'cranfield0956', 'cranfield0960', 'cranfield0976', 'cranfield1034', 'cranfield1038', 'cranfield1048', 'cranfield1050', 'cranfield1106', 'cranfield1107', 'cranfield1117', 'cranfield1119', 'cranfield1121', 'cranfield1128', 'cranfield1130', 'cranfield1173', 'cranfield1215', 'cranfield1227', 'cranfield1233', 'cranfield1237', 'cranfield1244', 'cranfield1250', 'cranfield1275', 'cranfield1302', 'cranfield1366', 'cranfield1387', 'cranfield1392', 'cranfield1396', 'cranfield1397', 'cranfield1398', 'cranfield1399', 'cranfield1400']

```
Number of comparisons required for query 1: 3637
```

Ans 3)

Input -

```
median section outer
```

Output-

```
Number of documents retrieved for query 1 using bigram inverted index: 1
Names of documents retrieved for query 1 using bigram inverted index: ['cranfield0922']
Number of documents retrieved for query 1 using positional index: 0
Names of documents retrieved for query 1 using positional index: []
```

We can see the bigram system retrieves 'cranfield0922' but the positional system does not. This is because 'cranfield0922' contains 2 bigrams median section and section outer but not the entire phrase.