IR ASSIGNMENT 1

Ques 1

- **PRE-PROCESSING:**

The text preprocessing has the following steps →

1. Convert the text to lower case.
2. Remove special characters.
3. Tokenize the text into tokens.
4. For each token, if it contains any special characters, remove them from the token.
5. Put the tokens in a set only to keep their single occurrence.
6. Keep tokens that are alphabetic and not present in the English stopwords.
7. Convert the tokens to a list.
8. For each token, if it is present in the inverted index data structure, append the doc to its posting. Else, add the token to the inverted index data structure.

- **METHODOLOGY:**

The steps in the implementation of the Inverted Index →

1. Read the text of each file in the folder.
2. Create a data frame where each row contains the filename, its raw text, and its processed text.
3. Sort the inverted index data structure lexicographically by tokens.
4. For each token, sort the posting list lexicographically by doc names.

The steps in the implementation of the Boolean Queries →

1. Implement the boolean queries
   For an operation between two tokens, the result will give →
   a. OR - Docs containing either of the tokens.
   b. AND - Docs containing both the tokens.
   c. AND NOT - Docs containing the first token but not the second. Find the common docs for both the tokens. Remove them from the documents containing the first token.
   d. OR NOT - Docs either containing the first token or not containing the second token. Find all the docs not containing the second token. Union them with the docs containing the first token.
   The number of comparisons in each operation →
   a. OR - For each pair of posting, a comparison is made between them.
   b. AND - For each pair of posting, a comparison is made between them.
   c. AND NOT - For each pair of postings, a comparison is made between them to find the common postings. Compare the common postings with that of the first token.

d. OR NOT - Check for all docs if they contain the second token. For each pair of posting, a comparison is made between the docs containing first and not containing second.

- **ASSUMPTIONS:**

Following assumptions have been made →

1. Since lemmatization is done, the tokens will convert to their root word.
2. The valid query → number of tokens will be one greater than the number of operators given.
3. Operators will be given as a string separated by commas.

Ques 2

- **PRE-PROCESSING:**

The text preprocessing has the following steps →

1. After reading the file, we first make the file lowercase.
2. Then we did convert text streams into tokens using TweetTokenizer().
3. After that, we remove all the stopwords from the stopwords library.
4. Then, we removed punctuations using regular expressions.
5. Finally, remove blank tokens if any in the token list got till now.

- **METHODOLOGY:**

The steps in the implementation of the Positional Index →

1. We initially took the token list after pre-processing for the particular file and enumerate each token by its name and position in the file.
2. Also created two data structures **PosInd** and **mapping** for storing all positional indexes of tokens and mapping of file number with file name respectively.
3. Then checks if token already exists then first increment the total document frequency and further checks if docNo do exist:
   3.1 If it does exist then simply append its position to the positional list.
   3.2 Else add the new list for the document.

4. If a token does not exist then first create a list and add frequency to be 1 and then add dictionary and add the docNo and position in the new positional list.

For Query Processing:

1. First, take the query from the user and preprocess it.

2. Then for each token, We convert the dictionary to the list and then check its positional list.
3. For each docNo in the list, if it matches and compares their position and if the position of the left token is lesser than the right then we added the right position into the new list, else we move the right pointer ahead.
4. Then we added that docNo along with a new list for the output list.
5. The final output list will contain all the required docNos and at last, we get doc names using the mapping data structure.

- **ASSUMPTIONS:**

  Following assumptions have been made →

1. We performed TweetTokenizer which keeps hashtags intact.
2. We do not handle any special characters if any in the file.
3. While query processing, We are assuming the token distance to be 5 as the max query size can be 5.

- **REFERENCES:**

1. https://www.geeksforgeeks.org/python-positional-index/
2. https://nlp.stanford.edu/IR-book/html/htmledition/positional-indexes-1.html
3. https://nlp.stanford.edu/IR-book/html/htmledition/a-first-take-at-building-an-inverted-index-1.html
4. https://www.geeksforgeeks.org/create-inverted-index-for-file-using-python/