IR Assignment 1

Methodology Followed for Solving Question 1:

Methodology followed for solving the first question is written as follows:

1. Pre-processed all the files while iterating over each line of all the files, steps for which are given in the next section.
2. Created Dictionary for storing inverted indexing for the files where each word is key in it and doc Id's of all the documents which have the respective word stored in it.
3. After generating the inverted indexing, we wrote the four methods for solving the given queries in part (C) for the question as follows:

    - X or Y – Iterated over the doc ID list for both the words and merged them together while using the merge algorithm for it.
      Complexity – O(m+n), where m is number of documents containing 'X' in it and n is number of documents containing 'Y' in it, as we have to iterate over both the lists to perform the merge algorithm.

    - X and Y – Iterated over both the doc ID lists and whenever the doc ID is less for either of them then, just skipped that iteration and increased the pointer for it and whenever, they are equal just added the respective doc ID to the list.
      Complexity – O(m+n), where m is number of documents containing 'X' in it and n is number of documents containing 'Y' in it, as we have to iterate over both the lists to check whether they are present in the same doc.

    - X or not Y – Followed similar approach just like X or Y but before applying it, just replaced the list of doc ID's with wherever, Y is not present.
      Complexity – O(number of documents), because we have to iterate over each doc to find whether document has Y or not, and then have to simply do merge algorithm which takes O(m+n) time.

    - X and not Y - Followed similar approach just like X and Y but before applying it, just replaced the list of doc ID's with wherever, Y is not present.
      Complexity – O(number of documents), because we have to iterate over each doc to find whether document has Y or not, and then have to simply check whether they are present in the same doc or not which takes O(m+n) time.

4. For obtaining the doc-docID mapping created another dictionary while parsing over each document, and stores document ID as the key and document name as value for it.
5. For producing the required output, obtained the resultant document ID's in a resultant list and mapped the document name from previously generated doc-docID dictionary.
6. For processing queries for part (D) for the question followed the below approach:

    - Took query as input from the user.

- Generated token for the same and on basis of the token inputted the number of operation (or/and/or not/and not)
- For finding the resultant documents with the inputted query, took first and second word and taking the first operation into account generated the resultant set of documents in a list, then performed next operation taking previously generated resultant set and next word's doc ID's and applying the next operator to it, storing the resultant set of docID's again on the same result set.
- After completion of all the operations, is the resultant list contains the docID's then, printed the document name for the same and if size is zero then, no document matched for the given query.
- Repeated the same steps for all the queries.

Methodology Followed for Solving Question 2:

Methodology followed for solving the second question is written as follows:

1. Pre-processed all the files while iterating over each line of all the files, steps for which are given in the next section.
2. Created Dictionary for storing positional indexing for the files where each word is key in it and then, in it's value stored dictionary further, where key is the document ID and position ID as value in it.
3. For processing the phrase query in the positional indexing, followed the approach as:

    - Processed each query, steps for which are given in the following section.
    - Then, for each word in the list brought the positions.
    - According to position of each word, added the same index to the positions in the list.
    - And for the value of the position which matched for all the words in the phrase query, we returned the document ID and then mapped document name for the same, from the earlier generated docID -docName dictionary.

Preprocessing Followed

Steps followed for pre-processing is given as follows:

1. Lowered all the words present in the sentence.
2. Removed all the words with only digits.
3. Further removed white spaces from the given strings.
4. Stripped extra spaces from the line from beginning and end of the line.
5. Finally, removed all the stop words from the line which are present in the NLTK library.

Assumption Taken into Account

1. We assumed that no query would contain digits or special characters, hence we removed the same in the pre-processing step.
2. Assumed that user would not enter any query containing all the stop words, hence removed all the stop words from the query.