IR Assignment-1

Samyak Jain (2019098) and Sarthak Johari (2019099)

Q1

Preprocessing :

After extracting the text from the files we performed the following preprocessing steps on the text to convert them into valid tokens which would be later used in making the unigram inverted index data structure.

1. First we converted the text to lowercase.
2. Text tends to have the shortened form of a combination of two words For Example “I will” is written as “I’ll” to fix this issue we have fixed the contractions from the text.
3. We then performed tokenization on the text
4. Only alphanumeric tokens were kept and the rest were discarded.
5. We then removed all the stopwords from our tokens.
6. Punctuations and blank spaces were removed from the tokens and finally we had our valid sets of tokens.

Methodology :

Creating the inverted index data structure

After creating the valid tokens for each file we began to implement the unigram inverted index data structure.

For this we maintained two dictionaries, one for postings and other for maintaining the number of files where that token can be found. We iterated over the tokens of each file.

First we checked if the token was present in the index,

- If absent we added the token to the index and initialized it with the document Id of the file we are in.
- If present we checked if the document Id was present in the posting of that token.
    - If absent we added the document Id in the posting of the token and incremented the frequency count by 1.

After this we first sorted the entire data structures by the term or the token and converted it to alphabetical order

Then, we sorted the each individual posting list in ascending order of the document ids and finally our inverted index was ready

Creating the logical operators query

- AND

  We wanted to perform the logical AND operator to perform the query between the postings of two tokens. We made a function which took the posting list of both the tokens. Then we performed the intersection of the two by performing the merge algorithm. The fact that the posting lists are already sorted makes the merge operation much more efficient and simpler.

  The first step was to check if any of the postings were empty. i.e the token itself did not exist in our inverted index. If any of the two token's postings were empty we returned an empty array. Only if both were non empty we went ahead with merging.

For the merge operation we kept two pointers one for each posting list and then traversed through the posting lists and compared the document ids where the pointers are during their traversal.

1. If the document ids are the same that means the tokens are present in both the files we append this document id in our answer array that we maintain. Then the pointers are incremented by 1.
2. If the value at first pointer is smaller than the value at second pointer we increment the first pointer by 1 and vice versa.

- OR

To perform the logical OR operator. We checked the posting of both the tokens.

1. If both the postings were empty we returned an empty array as the answer.
2. If any one of the postings were empty, we returned the other token's posting list as the answer to the OR operation.

3. For the merge operation we kept two pointers one for each posting list and then traversed through the posting lists and compared the document ids where the pointers are during their traversal.

    - If the document ids are the same that means the tokens are present in both the files we append this document id in our answer array that we maintain. Then the pointers are incremented by 1.
    - If the value at first pointer is smaller than the value at second pointer we increment the first pointer by 1 and add the document ID at the pointer in our answer array and vice versa.

- NOT
  Here we simply returned all the document IDs that are not present in the posting list that was given as the input. If the posting list is empty then all the document ids would be returned.

- AND_NOT

  Here we first check if the first posting list is empty or not. If yes we return an empty array as an answer. If no, we perform the NOT operator on the second posting list and check if the answer is empty or not. If yes, we return an empty array as an answer, if no, we perform the AND operator between the first posting list and output of the NOT operation on the second posting list.

- OR_ NOT

  Here we first perform the NOT operator on the second posting list. We then perform the OR operation on the first posting list and output obtained after the NOT operator of the second posting list.

Before performing the logical operations. The input query was preprocessed in the similar way we pre-processed the text that was present in the files. The query was converted into a valid token and then used to perform the operations.