**Information Retrieval Assignment-1**

**-SUBHANI SHAIK (MT18117)**

**Dataset: ‘**20newsgroup’ it had 19,997 instances. Some documents have same name as others. So, used document id as folder name and document name.

**Pre-Processing the dataset:**

For pre-processing the text I used NLTK library and done-

* Contraction:
* changed to lowercase
* tokenization: used RegexpTokenizer and used “[A-Za-z0-9]?\w+” regular expression to remove unnecessary literals and symbols.
* Lemmatization
* Stemming: to convert into its root word.

**1. Inverted Index**

* After the pre-processing, taken the distinct words and created the Inverted Index of structure:

<term: document frequency, list of document Ids>

Where document ids are in sorted order.

* Taken the terms along with document id and first sorted them according to term and then by document id.
* Created inverted index by using above created sorted terms using dictionary data-structure and saved that dictionary (Inverted index) and sorted terms list using joblib library.
* Got **184606** dictionary keys.
* Ex: see outputs in code file.

**2. Search queries**

After creating inverted index, to retrieve documents for Boolean queries- done the text-processing on query terms and retrieved the document ids for those terms and converted them to ‘sets’ to perform query operations.

Get the posting lists for terms x and y from the dictionary in which these terms are present and done the following Boolean queries-

* **x OR y**

Performed the “Set Union **(x U y)”** function between x and y document ids.

It has given the documents where either one of x or y or both terms are present.

* **x AND y**

Performed the “Set Intersection **(x intersection y)””** function between x and y document ids.

It retrieved the documents where both ‘x’ and ‘y’ terms are present.

* **x AND NOT y**

Performed the “Set Difference **(x - y)**”” function between x and y document ids.

It retrieved the documents where only ‘x’ terms are present.

* **x OR NOT y**

Performed the Set operation like “**Universal-(y-x)”** function between x and y document ids.

It retrieved the documents where either ‘x’ term or not ‘y’ term present.

**3. Positional Index and Phrase Queries**

* Done the pre-processing on specified documents and created the positional index of structure:

<term: docfreq, < docId: [term positions in document]> >

Where positions are in sorted order.

* For Phrase Queries: implemented as taken the common documents where all tokens of phrase query are present.
* And for each document- created the positions list for each token in query.
* Next, Subtract the positions according their index in query and taken the intersection.
* If intersection is empty list then that document don’t have that phrase.
* If intersection returns some value (not empty list) then that document have that phrase.
* Later saved the positional index dictionary using joblib and loaded when necessary and search the phrase query.
* By using positional index we can search for proximity queries also.