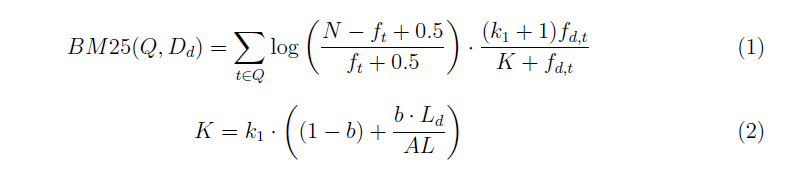
**Ranked Retrieval**

For implementation of BM25 function, firstly we modify our index part in order to calculate the document length and store them in map file, which printing “Document ID + Document Name + Document Length” for each document in map file. Then in the “loadMap” method we collect these information and store them in ArrayList<Document> documents.

After that, in BM25 method, the factor N (the number of documents in the collection) could be calculated by getting how many documents we collected from map file by calling “documents.size()”, the factor Ld (the document length) could be calculated by calling “documents.getLength()” and also the factor AL (the average document length) could be calculated by summing all the document length divided by the number of documents.

In the searching process, we created a String list called “searchTerms”which include all search terms, and for each of it we check all documents for the term frequency within the document in “bm25” method (we set term frequency for each word in each document during the “loadLexicons” and “search” method which completed in assignment1). If this term occurs we put this document into a new “ArrayList<Document> occurList” within this method, which can exclude all other irrelevant documents for the scoring. Hence we can get the factor Ft (the number of documents containing term t) by calling “occurList.size()”. Then, in each occur document, we get the current document’s Fdt (number of occurrences of t in d) and Ld, using the function provided by assignment specification to calculate the score and give this score to the document by calling “occurDocument.setScore()”.



We run it for every occur document with every query term by for loop. For example,

For (every search term){

For (every document) {

If (term occur){

Add to occurList;

}

For (every document in occurList){

Get Fdt and Ld;

Calculate BM25 score;

Add score to this document;

}

}

}

We are using the similar heapify process which provided by “Heapify” on page 69 in week 3 lecture note (IR-03-Ranking1). We pass the ArrayList “replace” to “heapifyList” method and use the “document.getscore()” function to heapify, which means we are actually changing the document position in replace by sorting, therefore we have all the information we need for result printing such as “Document ID”, “Document Num” and etc..

During the sorting of score by min-heap data structure in “heapifyList” method, we pick up the position 0 in “replace” after each min-heap sorting process, and add to the “ArrayList<Document> sortedDocument” which is the final ArrayList<Document> contains all the sorted scored documents.