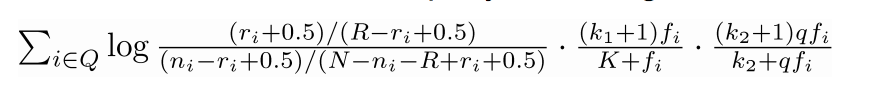
Below are the materials I reffered to for building BM25 model:

* Introduction to Information Retrieval. Manning, Raghavan, and Schütze
* [NNaji\_IR\_7](https://blackboard.neu.edu/bbcswebdav/pid-13607227-dt-content-rid-21937887_1/xid-21937887_1) Slide
* <http://www.cs.cornell.edu/courses/cs4300/2013fa/lectures/retrieval-models-2-4pp.pdf>

Formula used is :



where-

ri is the # of relevant documents containing term i (set to 0 )

ni is the # of docs containing term i

N is the total # of docs in the collection

R is the number of relevant documents for this query ! (set to 0)

fi is the frequency of term i in the doc under consideration

qfi is the frequency of term i in the query

k1 1.2

k2 100

/Users/sasankauppu/Desktop/Screen Shot 2017-03-28 at 11.28.45 PM.png

b 0.75

So for each query term the score is aggregated added to the document\_id. And the document\_id with the overall bm25 score is ranked first.

For Task-1:

Java and Lucene are used for implementation. I am reading query from a file. Below should be the format of the query file. [query.txt]

1:global warming potential

2:green power renewable energy

3:solar energy california

4:light bulb bulbs alternative alternatives

For Task-2:

Python is used for implementation of BM25. My source code is divided into to two files, uni-indexer.py will calculate the inverted index and bm25.py will give the top-100 documents for query from a file. Below should be the format of query file. [query.txt]

global warming potential

green power renewable energy

solar energy california

light bulb bulbs alternative alternatives