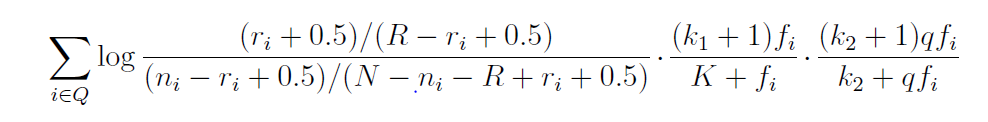
|  |  |  |
| --- | --- | --- |
|  |  |  |

BM25 ImplemtatIon

# BM25 FoRMULA :



Where :

* ri : Number of Relevant documents containing the word “i” (Note: Set to 0 since no relevance info is known)
* ni : Number of documents in the corpus that contains the word “i”
* N : Total number of documents in the collection
* R : Number of relevant documents for this query (Set to 0 since no relevance info is known)
* fi : Frequency of term i in the document under consideration
* qfi : Frequency of term i in the query
* k1,k2, K : are constants (details below)

The constant *k*1 determines how the *tf* component of the term weight changes as *fi* increases.

If *k*1 = 0, the term frequency component would be ignored and only term presence or absence would matter.

If *k*1 is large, the term weight component would increase nearly linearly with *fi*.

In TREC experiments, a typical value for *k*1 is 1.2, which causes the effect of *fi* to be very non-linear

The constant *k*2 has a similar role in the query term weight. Typical values for this parameter

are in the range 0 to 1,000, meaning that performance is less sensitive to *k*2 than it is to *k*1. This is because query term frequencies are much lower and less variable than document term frequencies.

Constant K : Normalizes the *tf* component by document length.

*K* = *k*1((1 *− b*) + *b · dl/avdl*)

Where *b* is a parameter, *dl* is the length of the document, and *avdl* is the average length of a document in the collection. The constant *b* regulates the impact of the length normalization, where *b* = 0 corresponds to no length normalization, *b* = 1 is full normalization. In TREC experiments, a value of *b* = 0*.*75 was found to be effective

***PROGRAM WISE IMPLEMNTATION :***

Take each query and break them down to individual terms (Binary Independence is assumed)

Now, for each of those terms we get the number of documents that have that query word mentioned.

For each of those file we calculate the BM25 value.

If all the query terms are present and they appear often the document will have a higher ranking.

Code for BM25 :

v = (((float(ri) + 0.5) / (float(R) - float(ri) + 0.5)) / (  
 (float(ni) - float(ri) + 0.5) / (float(N) - float(ni) - float(R) + float(ri) + 0.5)))  
v1 = math.log(v, 2)  
v2 = (((float(k1) + 1) \* float(fi)) / (float(K) + float(fi)))  
v3 = (((float(k2) + 1) \* float(qfi)) / (float(k2) + float(qfi)))  
BMval = (v1 \* v2 \* v3)  
BM25val += BMval

v corresponds to the first part of the equation

given we know ri,R,ni,N,fi and qfi

v2 corresponds to the second part of the equation

v3 corresponds to the third part of the equation.

This score is calculated for each query term in a query for each document that containd the query term.

Term Source :

* ni : Obtained from Unigram-DF dictionary created for Task 3

This dictionary has the term and number of documents they appear in

* fi : Obtained from Unigram\_inverted-index created for Task 3

This dictionary has the term and frequency of how many times they appear in each file in which they are present

* dl : is obtained from Unigram-File-Token-Count

This now has the name of each file and the length of the file

* avdl is obtained by using Unigram-TF I sum all the terms in the corpus divide by the total number of files this gives the average document length of each file in the corpus