**Project Report**

**Extracting contents from topics:** After finding inverted index and forward index. I am reading the contents from the file topics and extracting <num> tag, <title> tag, <narr> tag, <desc> tag. Creating three dictionary: querytitledic, querytitledescdic, querytitlenarrdic

querytitle contains query number as key and the its respective contents in title tag as value, similarly querytitledesdic has query number as key, content in title and description tags as value, querytitlenarrdic has query number as key and contents in title and narration tag as value.

**Query function**

**Term weighting**

I have written a query function, this function takes in one query dictionary and other empty dictionary as input arguments. In this function the words in query dictionary are split and are converted into wordids using inversedict, and then storing them in other dictionary called querytitle1, then we are finding the frequency of that querytitle1 using Counter and storing it in counter1 dictionary, Later we are finding inverted index of wordid and counting the number of documents in that inverted index (by counting number of “:” symbols), now we will calculate **idf**=logarithm (total number of documents)/(count of documents in inverted index of the term). Later we calculate Wtq =idf \* worid frequency in that query. Now iterating through the inverted index we will calculate Wtd=idf \* frequency of wordid in inverted index and W = Wtd\*Wtq,

**Normalization:**

Now, we will find the forward index of that document and then iterating through that forward index we will find inverted index of all the wordids and then count number of docids in it (by counting number of “:” symbols) and calculating idf1=log(total number of documents)/(total number of docids) , now calculating x=(wordid frequency)2 \* (idf1)2

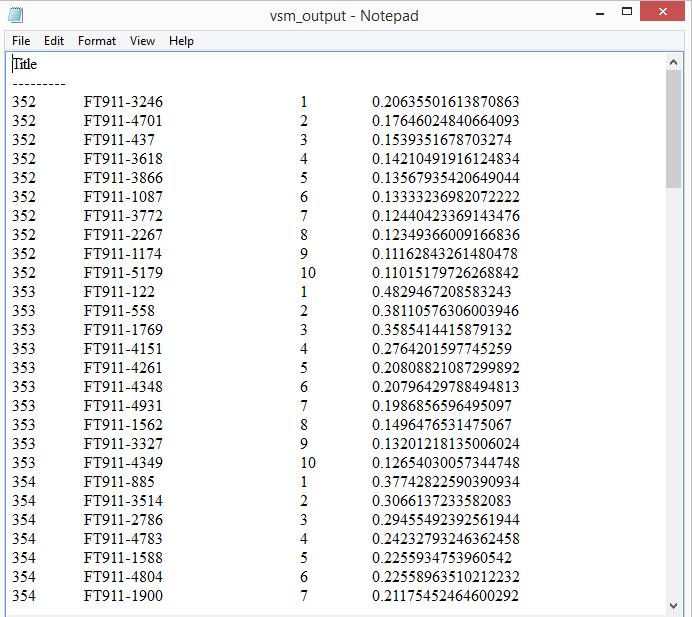
Now iterating through counter1 which has frequency of wordids of query , finding the inverted index of the wordid and counting number of docids (by counting number of “:” symbols) now calculating idf1=log(total number of documents)/(total number of docids), now calculating y=(wordid frequency)2 \* (idf1)2.

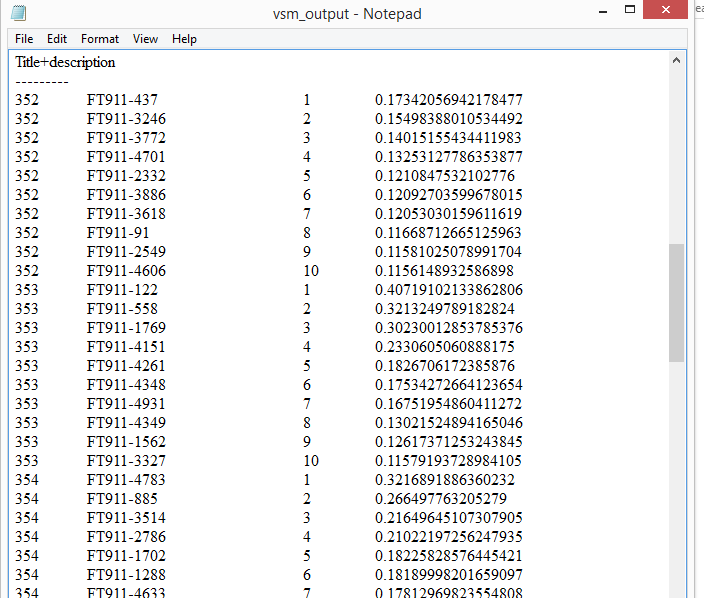
Finally we calculate final score by calculating as shown below

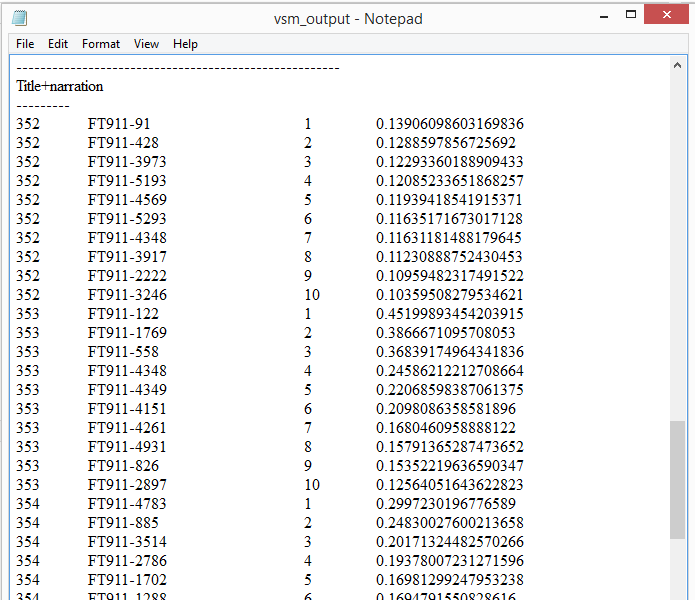
s =W/square root (x)\*square root(y)

Now updating the score dictionary, later sorting it in descending order we store in other dictionary called newscore with key as query number and value as the score.

Calling query functions for three combinations of query (querytitledic, querytitledescdic, querytitlenarrdic) and writing the first 10 in each query into a file called vsm\_output



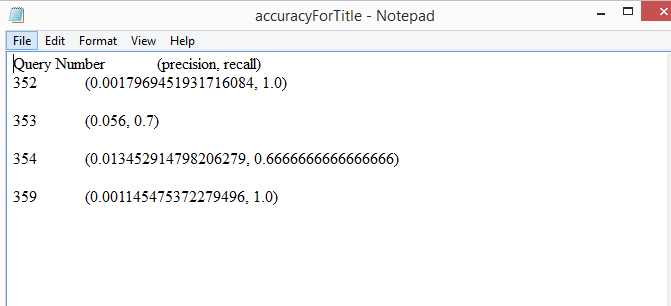


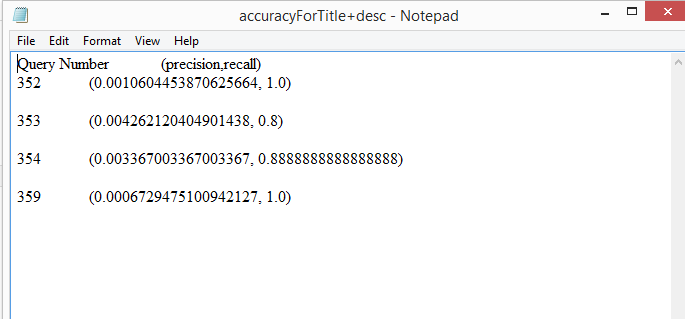


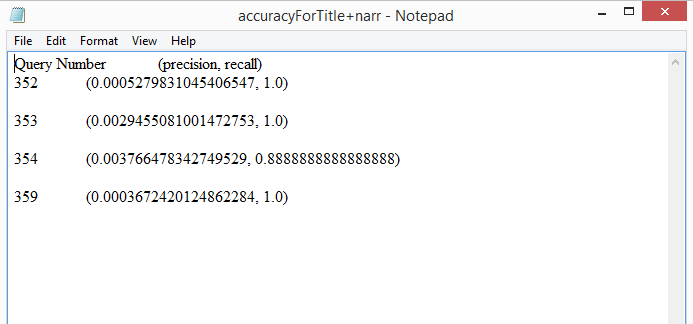
**(Precision and recall) Accuracy function**: I have also written accuracy function which takes newscore and other dictionary p (having doc numbers as value and query number as key). Iterating through newscore I am finding common documents count in p and newscore, which is called commondoc, count=total number of documents in newscore, to find precision pre=commondoc/ count, to find recall r=commondoc/ number of documents in p. Now storing pre and r in a dictionary called accuracy with key as query number.

Finally printing accuracy dictionary in a file

Calling accuracy functions for three combinations of query (querytitledic, querytitledescdic, querytitlenarrdic) and writing them in three files called (accurcyForTitle, accurcyForTitle+desc, accurcyForTitle+narr.)







**Observation:**

From the Final results of precision and recall, we can observe that as the number of terms in query increases the precision decreases as the words increases and the documents retrieved are also more so the precision is decreasing.

Recall is increased as the number of retrieved documents are more and the relevant documents will also increase.