**Document Retrieval System**

**Submitted By**

**Praneeth Reddy Vellaboyana**

**Under Guidance Of**

**Professor Kal Bugrara**

Contents

[1. Steps to run the Project: 2](#_Toc435399248)

[2. Information Retrieval: 2](#_Toc435399249)

[3. Inverted Index: 3](#_Toc435399250)

[4. Query Processing and Ranking (Using BM25): 3](#_Toc435399251)

[5. References: 7](#_Toc435399252)

# Steps to run the Project:

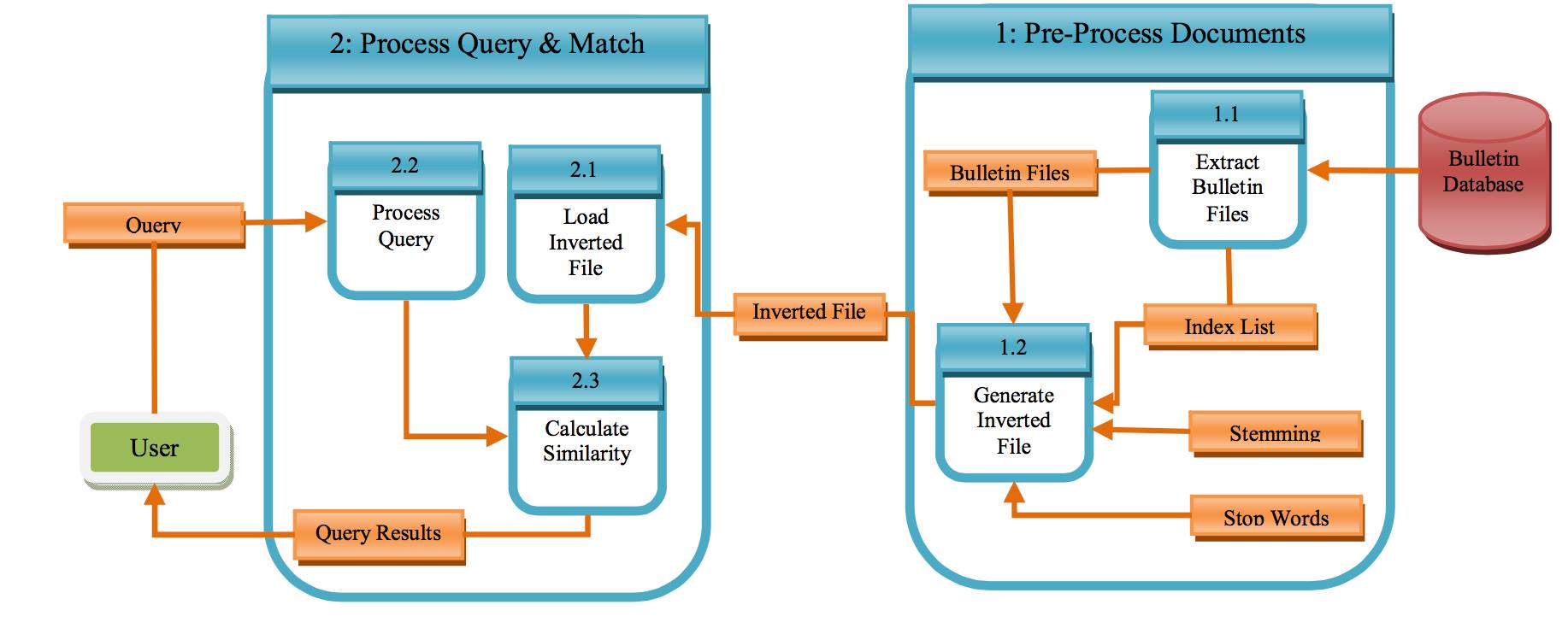
* You need to have an IDE and tomcat to deploy and run your project
* Import the project to your workspace.
* Go to **IntegerConstants.java** and change the path from where the application needs to pick the necessary files i.e., **cur\_dir** should be System.getProperty("user.dir"); OR if your current directory is not in the project provide the **project path ending with the \\**
* Now, run the project on the server and enter few search queries which would scan the list of files in the files folder of the project.
* Example search queries: Commonwealth, Blackhole etc.,
* Designed the Web interface using Bootstrap, JSPs and Servlets.

# Information Retrieval:

Information retrieval is a process of mining the huge volume of data/text available. There are many ways to retrieve information such as Boolean, space-vector, probabilistic and natural language processing.

The process involves two part

1. Forming the inverted index by processing the input files.
2. Taking the search query and matching it with inverted files.

****

The inverted index should be optimum for quick and easy retrieval and the data structures used should be relevant.

# Inverted Index:

The inverted index has been calculated by tokenizing the input text documents using StringTokenizer from Java. Each token will be separated using the following list of delimiters “~!@#øü$-=ä|ß°± %^&\\\"\*()?\_+{}:[]/';”.

Later on, the token will be check for stop words and when found that stop word is found that particular token will be eliminated from the list of the 313 standard stopwords. This will help the BM25 (Best Matching) Ranking algorithm to calculate the score based on the relevance.

The stop words removal will cut down the word list then, we need to put these token into a Bag-of-words format which are ordered alphabetically using a TreeMap data structure.

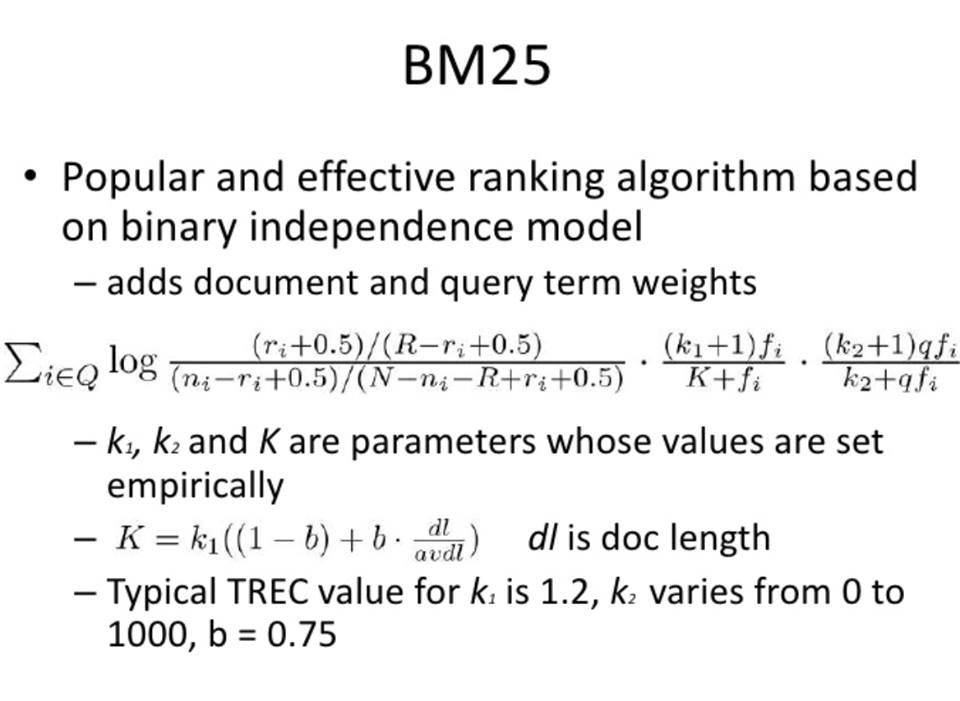
Before putting into the TreeMap, we need to stem the words to get the root word so that we can reduce the chuck of the words to a good amount which will then be put in a TreeMap.

Stemming is a linguistic process for which we have used Porter stemmer from the open nlp api using the private class which implements the stemmer interface.

As, I have used BM25 algorithm I have also calculated the token count in the sense the number of tokens per each documents.

# Query Processing and Ranking (Using BM25):

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of the inter-relationship between the query terms within a document (e.g., their relative proximity).



In BM25 derivation the IDF component is derived from the Binary Independence Model. The Binary Independence Model is a probabilistic information retrieval technique that makes some simple assumptions to make the estimation of document/query similarity probability feasible.

The BM25 rank algorithm ranks using the relevance something different from the vector space model which uses cosine function to calculate similarity.

Now the BM25 algorithm then retrieves the document and the frequency count of the word which matches the query string. So, each of the query word and the relevant information are placed in the query index TreeMap.

Now, iterating each of the Entry in the TreeMap we calculate the query frequency index for each word in the query which considers the weight of the word and also helps to calculate the frequency.

The set of constant used in the formula to improve the precision of relevance, to advance the binary independence model.

Here are the List of the constants used are referred from [1]:

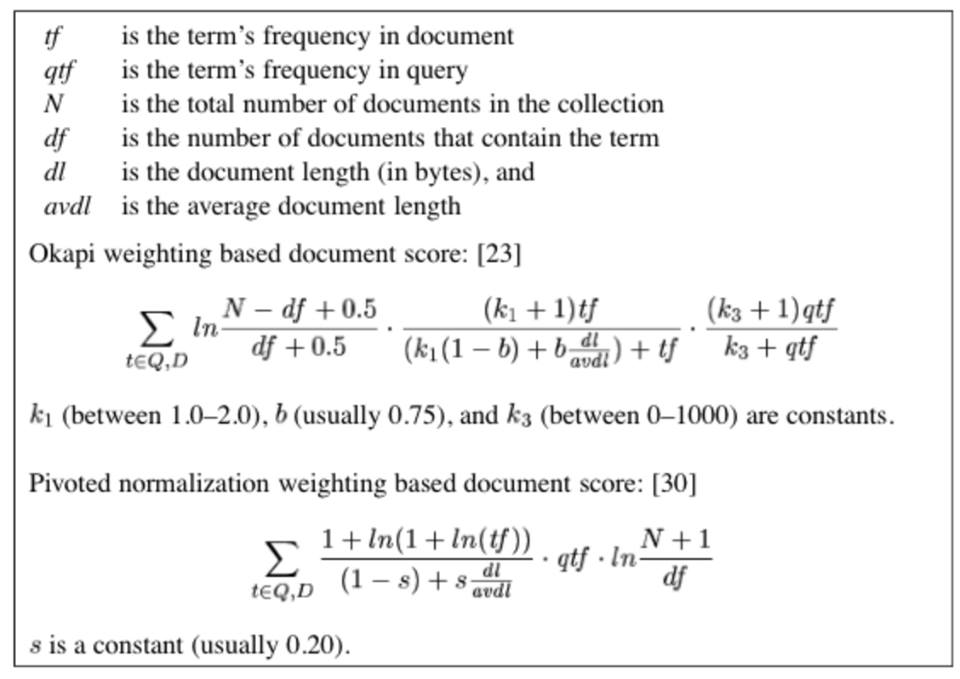
***k1*** = 1.2;

***k2*** = 100.0;

***b*** = 0.50; // scaling factor

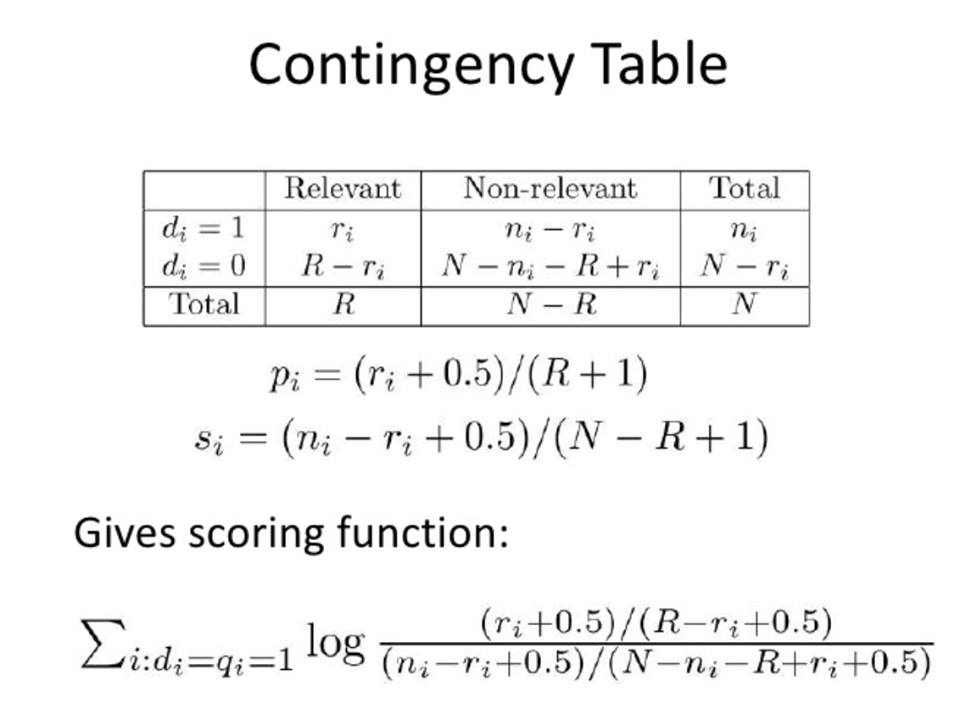
***ri*** = 0.30;//Relevant document for each term i

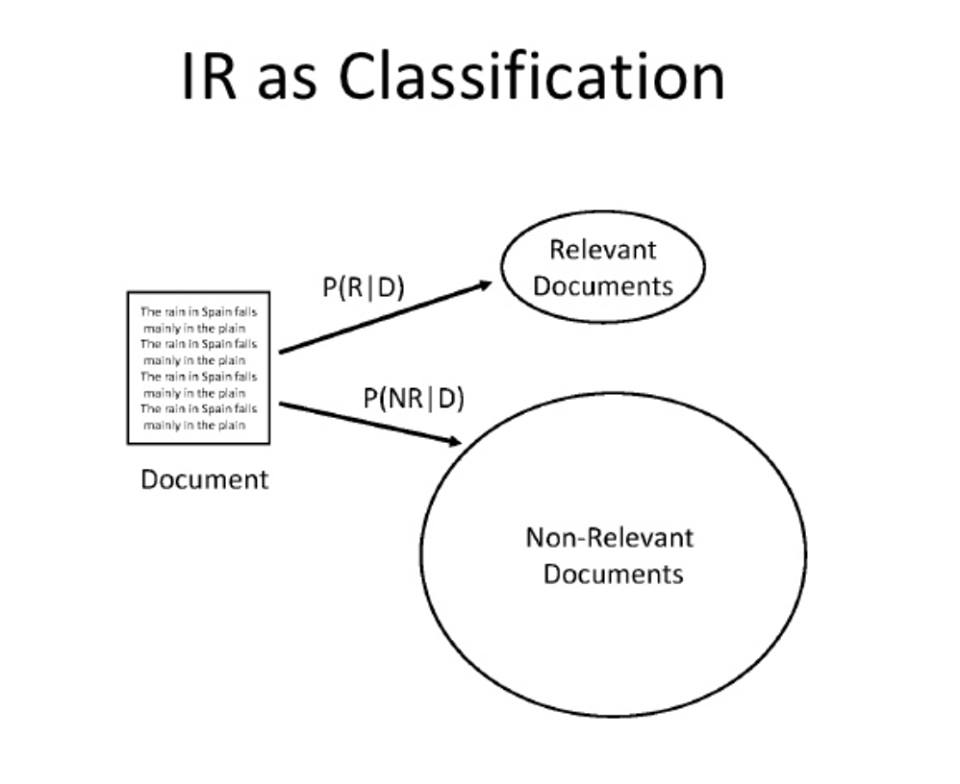
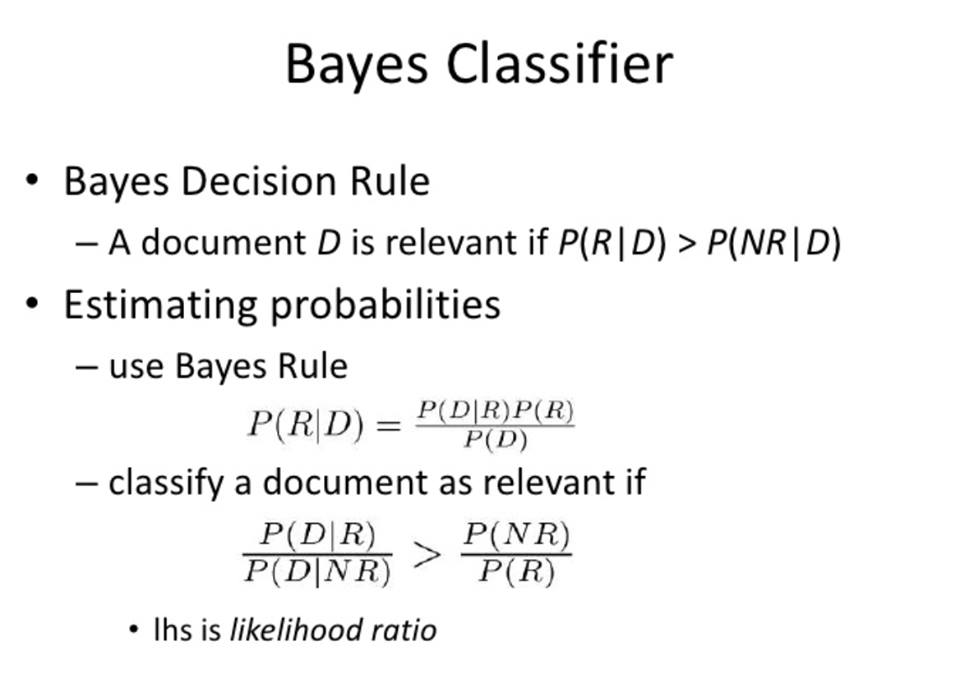
***R*** = 0.0; // total relevant documents for query

****

N is the total number of documents:

The contingency table represents the probabilistic approach of the query matching. This helps to calculate the relevance of the terms. The relevant documents from the total documents will be calculated by the below table,



Using the Bayes classifier the relevant and non-relevant documents are divided the formula is as, 

Based on the contingency Table the score is calculated and the documents retrieved are put in a TreeMap in reverse order of Rank to rank the most relevant documents based on the query String.

I have used Servlets and JSPs to implement the Information Retrieval model as the init() method of the servlet contains the inverted index process which runs only once for the whole cycle and each time we enter a query it(BM25) matches with the inverted index.

# References:

**[1]http://xapian.org/docs/bm25.html**

**[2]** **http://en.wikipedia.org/wiki/Binary\_Independence\_Model**

**[3]** **http://en.wikipedia.org/wiki/Okapi\_BM25**