**SHARAD DIXIT**

**CMSC 676 INFORMATION RETRIEVAL**

1. **OBJECTIVE**

The objective of this assignment was to build a command line retrieval engine on the top of the inverted index. The retrieval engine takes in queries which are lists of words, run them through preprocessing that is lowercasing, remove stop words and then match the query words against the inverted file to come up with document weights (the sum of the term weights in the document). Display the ten top-ranking document identifiers to the user.

**2.) ARCHITECTURE**

The command line retrieval engine works on top of inverted index that is in which the inverted index is saved to a text file on a server/disk, and then further query is matched with the inverted index in order to obtain the documents which match the query and further tf\*idf score is calculated for each score and top 10 documents with highest score are retrieved according the score, that is more score results in more ranking. Therefore this project is divided in two parts that is:-

1. Building of Customized Inverted Index
2. Query matching with inverted index for result retrieval

**# BUILDING OF CUSTOMIZED INVERTED INDEX**

* The inverted index developed has the following architecture in which each token is present in the inverted index and with that the documents that token belong to are present with the frequency count of the token in each document. This kind of inverted index helps us to calculate score only for the query and the matched elements of the inverted index, that is the inverse document frequency is calculated by seeing in how many documents the token is present and the term frequency of each document is also present in the inverted index with each document.
* After creating the customized inverted index, it is stored to server/disk which can be used further for query matching

**Sample Inverted Index:**

**Cat(TOKEN): <FILE1 ,10(TERM FREQUENCY); FILE2, 22 ; FILE 3, 33 ; ……………; FILE N, N >**

**# QUERY MATCHING WITH INVERTED INDEX FOR RESULT RETRIEVAL**

* The developed inverted index is matched with the query to retrieve documents with highest score. This is done by first matching the token with the inverted index and then the corresponding index elements are obtained for further processing. After the retrieval of corresponding elements of the token, the term frequency for each document is taken and inverse document frequency is calculated and hence tf\*idf is calculated for each documents and then top 10 documents are retrieved according to score calculated.

1. **METHODOLOGY**

The project methodology is divided in two following steps:

**# BUILDING OF CUSTOMIZED INVERTED INDEX**

* To build inverted index, I used HashMap with Key as the tokens and Value as an ArrayList of ArrayList. The ArrayList can be described as horizontal list which is outer ArrayList and vertical list which is inner ArrayList, that is horizontal list consist of different vertical list and vertical list consist of two elements that is first as document identifier and second as term frequency of token.
* Therefore, the above methodology is used for creating an inverted index and this index is then stored to disk as a text file for further use for retrieval of documents with score calculated.

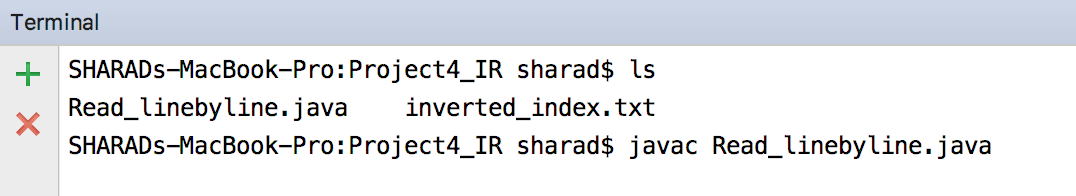
**# QUERY MATCHING WITH INVERTED INDEX FOR RESULT RETRIEVAL**

* The query is taken from user through command line where after receiving of query the query is matched with the inverted index. The matched elements of inverted index are obtained and then for each query by the user, a HashMap is created. The matched elements contain document identifier with the term frequency of the documents. Therefore, the Key of the HashMap becomes the document identifier and the Value becomes the corresponding term frequency of the token in that document. This process is done for each token and corresponding Document and Term Frequency HashMap is created for each token. Each HashMap is stored in an ArrayList, so that the HashMap can be accessed for calculation of tf\*idf weights and values updated of the HashMap as the weights value.
* After the Hash Maps are created for each token, then the Key of each map is compared with that of the other map to find documents which have all the tokens of the query. The common Keys are then further stored in another HashMap with the documents weight and Value of the HashMap as the cumulative sum of the common documents weights.
* The final HashMap created is then sorted according to values that is sorted according to weights with weights more value at upper position and the result is then displayed to user.
* Hence, the above methodology provides the user with the top 10 Documents with their corresponding weights calculated

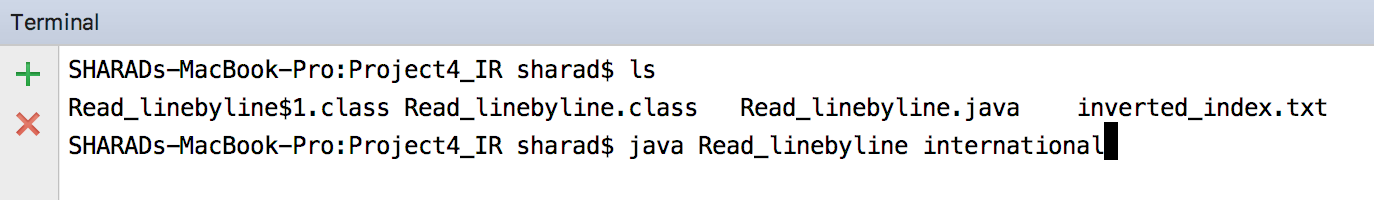
1. **EVALUATION OF PROGRAM**

* The efficiency of the retrieval is very optimized because of the use of the HashMap instead of ArrayList, but the use of HashMap results in more consumption of memory.
* The retrieval time is near about a second which is good in terms of a search engine and which can be further improved if instead of saving the inverted index as a text file it can be directly serialized as an object to the disk and deserialized when needed for query.

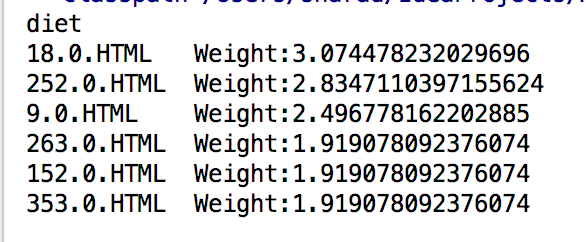
1. **OUTPUT**
2. **Compilation of Java source code:**

****

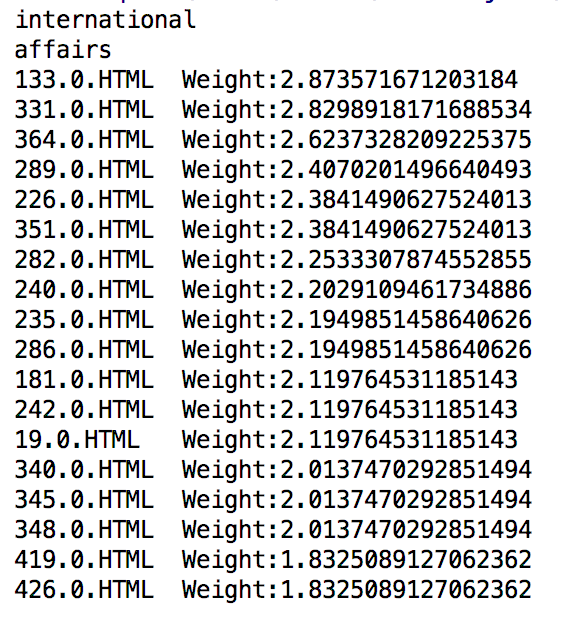
1. **Running of the generated class file against the user query**

****

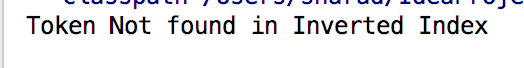
1. **Output for query**
2. **Diet**

****

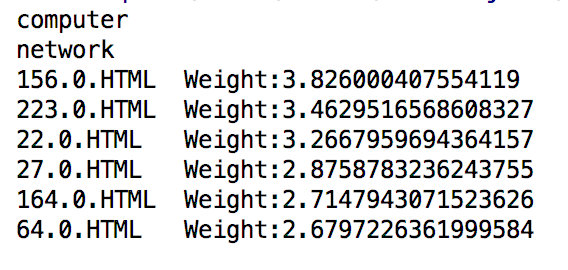
1. **International affairs**

****

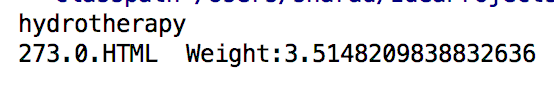
1. **Zimbabwe**

****

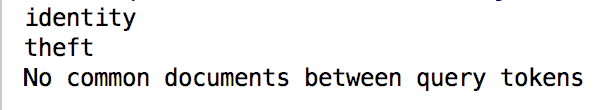
1. **computer network**

****

1. **hydrotherapy**

****

1. **identity theft**

****

1. **Working of Program**

**A) Pre-Requisite**

# Inverted index has been created already and imported in the package so that the source code picks it on its own. This inverted index created helps in fast retrieval of the documents against the query.

**B) Developing Environment**

# Mac OS with 16GB RAM

**C) Submission Package**

# Report

# Source Code for retrieval of documents (search engine) with Java Class (Trial.java & Trial1.class) inside (src directory)

# Source Code for inverted index development with developed inverted index text file(Read\_linebyline.java) inside Project4\_IR (directory)