Exercise 2 - Lucene, BM25, and evaluation

Group09

Samira Mahdi Zadeh- 1329639

Thiago Carneiro-1028762

**The prototype**

Our prototype for indexing and searching using Lucene 4.7.1 consists of 3 classes:

* IndexLucene.java

In this class we created the Lucene index for 20\_newsgroups\_subset and stored it in a directory called “indexLucene”.

To do so we used the “Standard analyzer” of Lucene to build up the tokens as bag of word model. The analyzer is also responsible for the vocabulary normalization ( e.g. case folding and removing stopwords). Then we traversed the subset recursively to find the files to be indexed. For each file we created a Lucene Document which is the basic unit for indexing and searching and contains a list of fields. The fields that we stored in this document are:

* “path”: the path of file > indexed, the original field value stored in the index
* “modified”: the last modified date of the file > indexed
* “contents”: the contents(text) of the file > indexed and tokenized

In which the path value and contents were used in searching and printing results for SearchLucene.java

* SearchLucene.java

In this class we used Lucene functionality to search for top 100 results from already indexed subset which was stored in “indexLucene” directory from previous run of IndexLucene.java.

In order to make query from “topic” files, we read each file and excluded the especial characters (+ - && || ! ( ) { } [ ] ^ " ~ \* ? : \ /) from it; because otherwise the queryParser of Lucene would manipulate them as a part of query syntax. We then parsed this query using the same analyzer (StandardAnalyzer) that we used for indexing and then called search method of Lucene to perform search and return top 100 matching results and finally we printed the results according to the specified format:

topic1 Q0 misc.forsale\76057 1 5.6695824 group09\_experiment2

topic1 Q0 misc.forsale\76442 2 4.0179787 group09\_experiment2

…

* BM25LSimilarity.java

Considering the assignment requirement for providing BM25L score calculator, we modified the Lucenes’ BM25Similarity class which implements the following formula:

To avoid overly penalizing of very long documents we introduced the δ parameter with default value of 0.5 as the new class member variable of our BM25L implementation and modified the normalization formula to:

**How to run the prototype**

**?????????????????????????????????????**

**How the score is calculated for two documents adjacent in the ranked list**

In order to do this we used a simple query of “small case” on our indexed subset and retrieved 100 most relevant results. We used Lucene’s searcher.explain() method and print out the returned explanation objects as follows:

0.47844946 = (MATCH) sum of:

0.28310627 = (MATCH) weight(contents:small in 1327) [DefaultSimilarity], result of:

0.28310627 = score(doc=1327,freq=1.0 = termFreq=1.0

), product of:

0.76923084 = queryWeight, product of:

3.92574 = idf(docFreq=428, maxDocs=8000)

0.19594544 = queryNorm

0.36803812 = fieldWeight in 1327, product of:

1.0 = tf(freq=1.0), with freq of:

1.0 = termFreq=1.0

3.92574 = idf(docFreq=428, maxDocs=8000)

0.09375 = fieldNorm(doc=1327)

0.1953432 = (MATCH) weight(contents:case in 1327) [DefaultSimilarity], result of:

0.1953432 = score(doc=1327,freq=1.0 = termFreq=1.0

), product of:

0.6389709 = queryWeight, product of:

3.2609634 = idf(docFreq=833, maxDocs=8000)

0.19594544 = queryNorm

0.30571532 = fieldWeight in 1327, product of:

1.0 = tf(freq=1.0), with freq of:

1.0 = termFreq=1.0

3.2609634 = idf(docFreq=833, maxDocs=8000)

0.09375 = fieldNorm(doc=1327)

topic1 Q0 comp.sys.ibm.pc.hardware\60400 1 0.4784495 group09\_experiment2

0.3987079 = (MATCH) sum of:

0.2359219 = (MATCH) weight(contents:small in 1332) [DefaultSimilarity], result of:

0.2359219 = score(doc=1332,freq=1.0 = termFreq=1.0

), product of:

0.76923084 = queryWeight, product of:

3.92574 = idf(docFreq=428, maxDocs=8000)

0.19594544 = queryNorm

0.30669844 = fieldWeight in 1332, product of:

1.0 = tf(freq=1.0), with freq of:

1.0 = termFreq=1.0

3.92574 = idf(docFreq=428, maxDocs=8000)

0.078125 = fieldNorm(doc=1332)

0.16278599 = (MATCH) weight(contents:case in 1332) [DefaultSimilarity], result of:

0.16278599 = score(doc=1332,freq=1.0 = termFreq=1.0

), product of:

0.6389709 = queryWeight, product of:

3.2609634 = idf(docFreq=833, maxDocs=8000)

0.19594544 = queryNorm

0.25476277 = fieldWeight in 1332, product of:

1.0 = tf(freq=1.0), with freq of:

1.0 = termFreq=1.0

3.2609634 = idf(docFreq=833, maxDocs=8000)

0.078125 = fieldNorm(doc=1332)

topic1 Q0 comp.sys.ibm.pc.hardware\60415 5 0.3987079 group09\_experiment2

According to the output of explanation, the total score of a query in a documents is equal to sum of the weight of each term in query in that document. So:

score = weight (“small”) + weight (“case”)

where: weight(term) = queryWeight x fieldWeight

where: queryWeight = idf x queryNorm ; fieldWeight = tf x idf x fieldNorm

putting them all together we come up with:

Which is actually a form of tf.idf similarity function used by Lucene as the Default similarity function.

**Evaluation Results**

???????????????????????????????????????????????