CSI 4107 Assignment 1

Martin Moncion 6146324

Jeremy Sivaneswaran

Division of work:

|  |  |
| --- | --- |
| Building The Parser | Martin |
| Creating the Index | Jeremy |
| Search Index functionality | Martin & Jeremy |
| Creating Results File | Martin & Jeremy |
| Implementing the Evaluation Script | Martin |
| Documentation & Report | Martin & Jeremy |

Program Details:

The parser is the function getParsedList(). The Parser uses a buffer reader to read every line from a text file that’s stored in the projects data folder. Within the Data folder is a file called SampleData.txt which contains the collection of twitter messages. Buffer reader reads the file line by line and stores it in an array list called parsedlist. The array list is then parsed to remove stop words, websites, numbers and non-characters and saved in the Data folder as ParsedData.txt. Parsing is done within a for loop that that takes each word from a twitter message and removes the stop words from it, then it puts the string back together. And uses the replaceall function and a couple of regular expressions to remove websites, numbers and non-characters.

Stemmer.java is the sample code provided for the assignment. Two functions were added to to stem either an entire file or a single string. Update index is used to call the stemmer class so that it can stemm the file and then be added into our inverted index.

Once the list is Stemmed it gets added to our inverted index. In this program we use Lucene to create our inverted index. With Lucene creating the index is simple you call

Directory index = FSDirectory.*open*(**new** File("Data\\index"));

We also need an analyzer to build tokenstreams, which analyzes the text. The English analyzer was used because it stems the input to ease searching.

EnglishAnalyzer analyzer = **new** EnglishAnalyzer (Version.~~LUCENE\_44~~);

The updateindex function is where the index is written into. The indexwriter class is used to add, modify, delete and maintain documents in the index. In order to place words in the index the index writer is configured with the analyzer. It then calls the addDoc function to add the document with the provided nanlyzer to tokenize the the parsed data. We create a new textfield to be added into the index using doc.add

doc.add(**new** TextField("title", title, Field.Store.***YES***));

To search the index we use several classes from lucene. We use the Query class to make Lucene query from a string, an index reader to read from the index we created, an indexsearcher to perform the search, TopscoreDocCollector is used to store the results and ScoreDoc to iterate through our results. Every query gets stemmed and all non-characters are removed and each letter is set to lowercase.

// Text to search

String querystr = *TestQueryList*.get(l).toLowerCase();

querystr = querystr.replaceAll("[^a-zA-Z0-9]", " ");

querystr = *stemwords*.getStringStem(querystr);

Query q = **new** ~~QueryParser~~(Version.~~LUCENE\_44~~, "title", analyzer).parse(querystr);

// Searching code

**int** hitsPerPage = 1000;

reader = DirectoryReader.*open*(index);

IndexSearcher searcher = **new** IndexSearcher(reader);

TopScoreDocCollector collector = TopScoreDocCollector.*create*(hitsPerPage, **true**);

searcher.search(q, collector);

ScoreDoc[] hits = collector.topDocs().scoreDocs;

Then to create our results file we use the following for loop

**for**(**int** i=0;i<hits.length;++i){

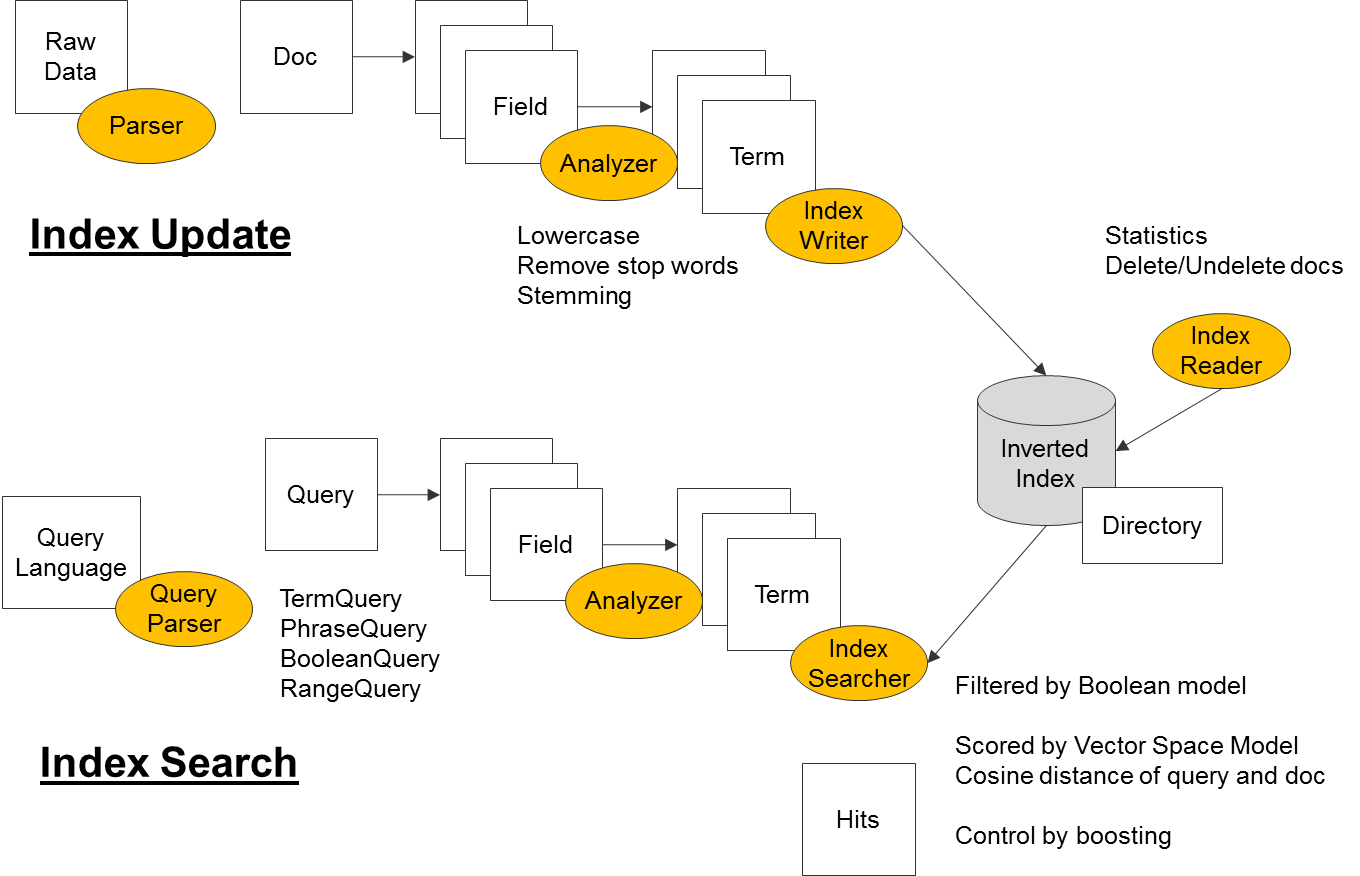
**int** docId = hits[i].doc;

Document d = searcher.doc(docId);

writer.write(*TestList*.get(l) + " Q0 " + d.get("isbn") + " " + (i + 1) +" " + hits[i].score +"\t" + "myrun" + "\n");

}

Searcher.doc ranks the most relevant documents at number one. The list is iterated into the SearchResults.txt file and repeated for each query. Lucene uses a default scoring weights results using TF—IDF.



Source: http://3.bp.blogspot.com/-sk7BbSn3t\_E/USeevUEf\_BI/AAAAAAAAA-k/q2hKWv3rl5U/s1600/p1.png

Evaluation:

This step was done separately by using the TREC\_eval script. I ran the script using a linux virtual machine and transferred the files to the virtual machine to run. I used the following command to create the results/

./trec\_eval -q –c –M1000 Feedback.test SearchResults.test > Results.txt

-q: In addition to summary evaluation, give evaluation for each query

-c: Average over the complete set of queries in the relevance judgements instead of the queries in the intersection of relevance judgements and results. Missing queries will contribute a value of 0 to all evaluation measures.

-M<num>: Max number of docs per topic to use in evaluation (discard rest).

The results are then stored in the Results.txt file in Data/output\_results folder

How to run:

Lucene.java is able to run in eclipse it does not take any arguments. (Note the updateIndex function is commented out because the index is already created. It duplicates the terms if the index is not wiped. The easiest way to clear the index is to delete the all the files in Data/index folder.)

The results are saved in the projects data folder as SearchResults.txt

To run the trec\_eval script you need a linux machine and I used the following command:

./trec\_eval -q –c –M1000 assignmentfiles/Feedback.test assignmentfiles/SearchResults.test > assignmentfiles/Results.txt

The data structures used were arraylist since they are dynamic in size and we don’t know how many twitter messages there are. To optimize our results retrieved we used the porter stemmer to receive better results. We are also collecting the data locally from text files instead of creating a webcrawler to retrieve the data from the site. We used the latest version of Lucene to get the fastest searches possible. As for storing the index we use FSDirectory.open() to select the best type of index based on the systems limitations.

Our vocabulary ended up being 49283 unique terms. You can run the function gettotal() to receive the result

Vocabulary Sample:

a in 1 documents

aa in 12 documents

aaa in 7 documents

aaaaaaaaaaaaaa in 1 documents

aaaaaaaaaaaaaaaaaaaaaaa in 1 documents

aaaaaaaaaaaaaah in 1 documents

aaaaaaaaaargh in 1 documents

aaaaaaawwwwww in 1 documents

aaaaaawwwwwwww in 1 documents

aaaaah in 1 documents

aaaaaiiiii in 1 documents

aaaaao in 1 documents

aaaahh in 1 documents

aaaain in 1 documents

aaaaronnnn in 1 documents

aaaggghhh in 2 documents

aaah in 2 documents

aaahh in 1 documents

aaand in 1 documents

aaanniek in 1 documents

aaannnnddd in 1 documents

aaargh in 1 documents

aaawwww in 1 documents

aaawwwwhhhh in 1 documents

aab in 1 documents

aacon in 1 documents

aacrao in 1 documents

aadithama in 1 documents

aadukalam in 1 documents

aafaqu in 2 documents

aafia in 1 documents

aag in 1 documents

aaha in 1 documents

aahaha in 1 documents

aahahah in 1 documents

aahhahahaha in 1 documents

aai in 1 documents

aaiisss in 1 documents

aaj in 3 documents

aalbamcfli in 1 documents

aambc in 1 documents

aamco in 1 documents

aamcocarcar in 1 documents

aameen in 1 documents

aamer in 4 documents

aan in 31 documents

aanholt in 1 documents

aanhoudingen in 1 documents

aankleden in 1 documents

aankoopproc in 3 documents

aannnnyywhher in 1 documents

aanstekelijk in 1 documents

aanval in 1 documents

aanzettten in 1 documents

aapko in 1 documents

aapl in 2 documents

aargh in 2 documents

aarmaanta in 1 documents

aaron in 37 documents

aaronbertrand in 1 documents

aaronfresh in 2 documents

aaronrodg in 2 documents

aarp in 1 documents

aarptx in 1 documents

aart in 1 documents

aartipaarti in 1 documents

aashiyana in 1 documents

aaup in 1 documents

aawayi in 1 documents

ab in 28 documents

aba in 1 documents

ababa in 2 documents

abadi in 1 documents

abai in 1 documents

abajo in 2 documents

aban in 1 documents

abandon in 17 documents

abandonando in 1 documents

abandono in 1 documents

abang in 1 documents

abank in 1 documents

abarth in 1 documents

abat in 1 documents

abba in 1 documents

abbevil in 1 documents

abbeyniezgoda in 1 documents

abbi in 5 documents

abbigliamento in 1 documents

abbotsford in 2 documents

abbott in 4 documents

abbrevi in 1 documents

abc in 64 documents

abcdefg in 1 documents

abcenviron in 1 documents

abcnew in 5 documents

abcnewsradio in 1 documents

abcpolit in 1 documents

abcscienc in 1 documents

abcthedrum in 1 documents

abcworldnew in 1 documents

Note: the query resultsare the stemmed results

first 10 answers to query 1:

MB001

30260724248870912 bbc world servic outlin cut to staff

30198105513140224 bbc new bbc world servic cut to be outlin to staff

30236884051435520 bbc confirm world servic cut

30275282464153600 bbc world servic to cut a quarter of it staff after lose million in fund from the foreign offic

34952194402811904 save bbc world servic from savag cut

33823403328671744 world servic cut why we need the bbc

32504175552102401 save bbc world servic from savag cut

32158658863304705 save bbc world servic from savag cut

30303184207478784 bbc blame govern for world servic cut

29990282447552512 bbc world servic to cut up to job guardian

first 10 answers to query 25:

MB025

31550836899323904 tsa shut door on privat airport screen program utter bs cnn

31286354960715777 tsa shut door on privat airport screen program cnn

32609015158542336 tf travel rt bitter american tsa shut door on privat airport screen program cnn

31823291815567361 tsa shut down privat airport screen program is headlin now on www fedsmith com

31738694356434944 tsa shut door on privat airport screen program patriot updat via addthi

31320463862931456 tsa halt privat screen program

32528974961713152 atl busi chronicl tsa to test new screen at hartsfield jackson

33174905985957888 ditch tsa airport no longer can opt out

32955753920733184 tsa tri less nuditi at airport scan

32541161675558912 tsa to test new screen at hartsfield jackson the tsa in come dai at hartsfield jackson atlanta internatio

Final results discussion:

The final results are stored in Data/outut\_results/Results.txt