CS172 Project 2: Twitter Tweet Tweet Crawler
Collaboration Report

Collaboration Details:

Team Members:

Michael Villanueva	mvill025	861074235
Kevin Hsieh	khsie003	861054367

All the Work was done together meaning, we collaborated on all parts of the project. Because we only had one file to work on, we took turns coding on the Eclipse IDE. While one person was working on the code, the other person did research to figure out the other parts of structure of the project and the implementation of it.

Overview of System:

Architecture:

We build the project using Java EE IDE along side with tomcat to build the server. We imported a java class that we wrote into the project hierarchy as a package so that we can use the java class in our index.jsp file. The index.jsp file is our main file which consist of the html formatted tags with java incorporated codes, which we used to initialize our LuceneSearch class and use its search function. the LuceneSearch class is the java class we wrote to index the tweets and search through the index.

Tomcat 6.0 was set up within the Java EE IDE to make a server at localhost for the application. We are using the default runtime configuration. The server starts up whenever we run the application and stops whenever the application is terminated.

Index Structure

We use lucene to index all the tweets from the crawler into one directory. This allows a fast and simple search in contrast to trying to index to multiple directories. Since lucene only needs to return results from one directory, the search for the best results in the index takes

shorter time than returning results from multiple directories and finding the best among those results. There is a trade-off though, if we had implemented
the program to index to multiple directories, then we may have been able to multi-thread indexing (thread per directory). This could allow a much faster indexing time than the current program.

Inside each index, we have five different fields stored: user,tweet,time,url,title. We use user,tweet,title,time to evaluate relevancy and score to the query during search. url is used when results are returned and will hyperlink any url associated with the tweet.

Search Algorithm

We use lucene to implement the search algorithm. For each field we tell lucene to match the query, we give it a weight. Depending on how relevant the field is with the query, lucene will give it a value, the value is scaled by the weight of the field. Lucene adds up all the field values of the index/document and compares it to the scores of the the rest of the indexes in the directory. We then return the highest scored documents as results.

Limitations:

Lucene Lock, Single Thread Indexing

When indexing, Lucene implements a write lock on the directory that it is currently writing its indexes to. Meaning only the thread that wrote the lock there can write to it after the lock is implement. As one thread would attempt to add an index to the directory it would be blocked and time out. This prevented us from implementing a multi-thread indexer. If we were able to change lucene so that it could write to the same directory, we would have made the indexer multithreaded. Since the indexer is single thread, it can only add tweets at a much slower pace.

Lucene does most of the ranking

Lucene indexes and ranks the indexes for the Web crawler. Since everything is done through lucene, we have to abide mostly to lucene format. It isn't that much of a limitation since lucene is well implemented to maximize indexing and searching, but It would have prevented us from using some very custom and unique ranking/indexing algorithms.

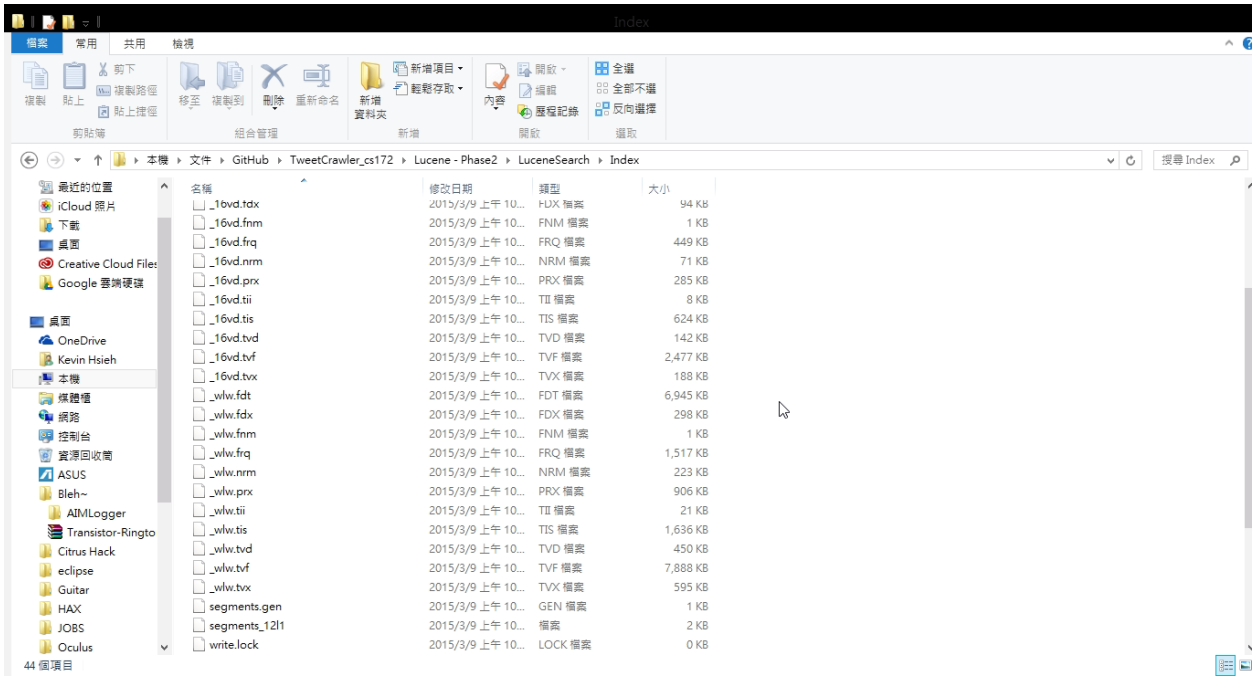
Instruction on how to deploy the system:

To get the application output from Eclipse, first, open up the project in Eclipse. Make the appropriate edit, in LuceneSearch.java, to the path of the INDEX_DIR, where the indexer will make the directory at. Also edit the path to the *file* variable in getFile() method, this is the folder with all the tweets in simple txt files with the format starting file1.txt to file"*n"*.txt.

After these changes are made, use a Tomcat Server with Java JRE 1.8 to run the application. Make sure to uncomment line 44, the getline() method call within main, whenever you want to create indexes from file directory. Try not to call getFile() too many times for that it will re-index the same tweets making multiple entries in the index directory.

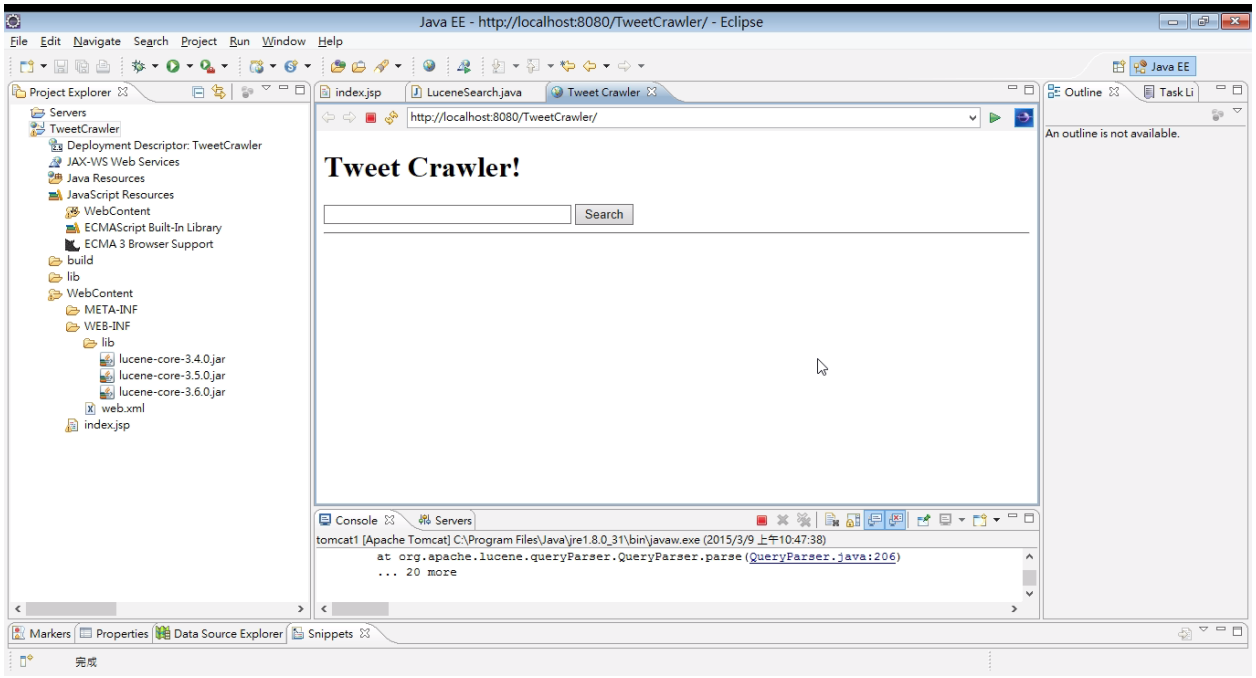
Screenshots showing the system in action:

folder with of all Lucene Indexes:



LuceneSearcher.java indexing tweets (each is a tweet):

HomePage of TweetCrawler:



TweetCrawler with “hello” as the query: