*Readme File*

**Group Members: Sargon Hasso, Kenan Matawie**

**Instructions**

NOTE: I uploaded this material to GitHub at this location: <https://github.com/shasso/ICHI-2015-DataAnalyticsChallenge>. If you clone this project, then all what you need is there. You can still follow the instructions in this document, but any refernce to ‘zipped’ material and steps involved in unzipping files should be ignored. See screen shot of this repository on GitHub at the end of this document.

The Application was developed in Java using Netbeans IDE with JDK 1.8. There are two Netbeans projects associated with application corresponding to the two executable components of the application that need to be run in this order:

1. The Indexer component: hcDAChallengeBuildIndex.jar
2. The Searcher component: hcDAChallengeSearcher.jar

**Deliverables**

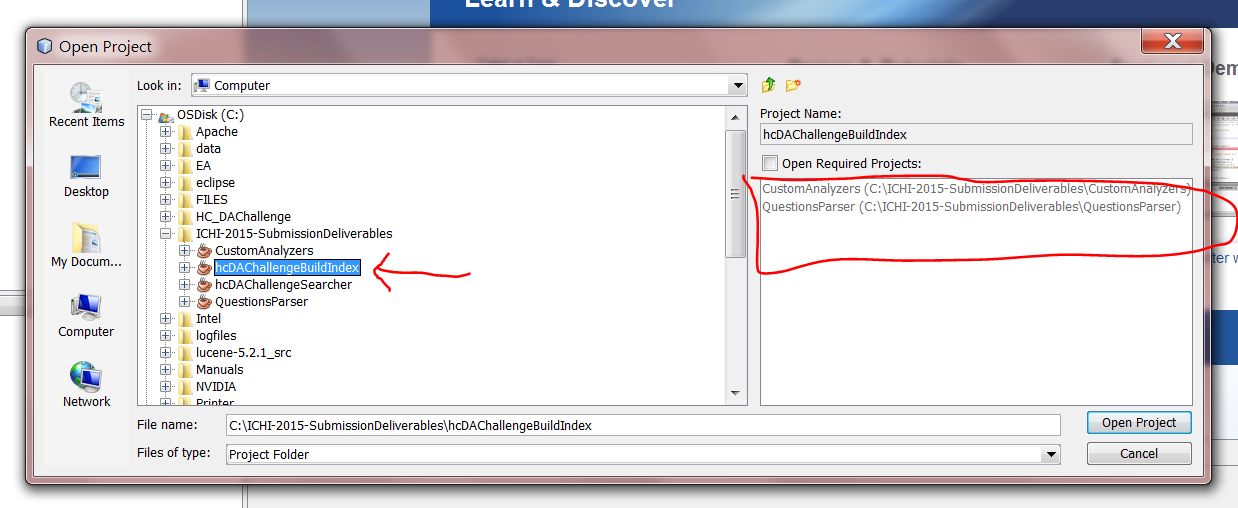
The entire Netbeans project directory is zipped for the two the folders corresponding to the two components mentioned above. Also, there are other dependencies and run time files that are needed and are delivered as zip files (in 7z format).

1. hcDAChallengeBuildIndex.7z—for indexer component
2. hcDAChallengeSearcher.7z—for search component
3. CustomAnalyzers.7z—first dependency project needed by both components
4. QuestionsParser.7z—second dependency project needed by both components
5. ‘supplementary.7z’: when unzipped, it contains additional files that are needed at runtime:
   1. ‘app.properties’: runtime properties file for the two components (the same file is used for both)
   2. ‘wn\_s.pl’: this a WordNet thesaurus database that is used by both executables
   3. ‘ICHI-Challenge-Sample-Question-Corpus.txt’ file from ICHI-2015
   4. ‘ICHI-Challenge-Sample-Test-Questions.txt’ from ICHI-2015

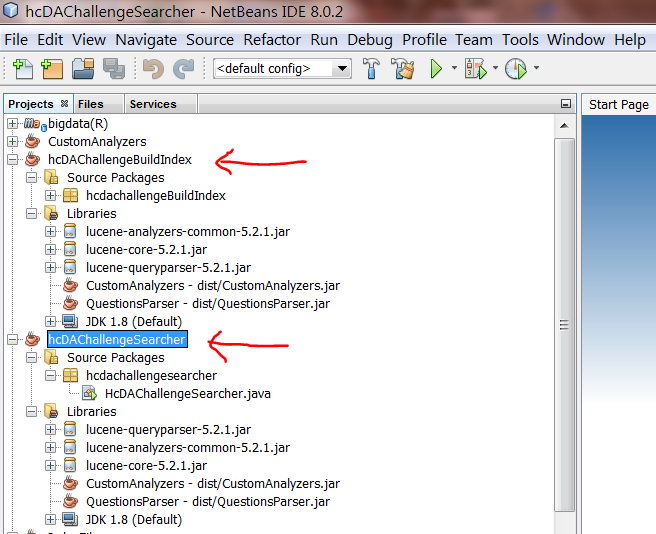
These folders have everything that is required to build the two executables including the dependencies.

**Instructions to build the executables jar files with Netbeans IDE 8.0.2 on Windows 7 system**

1. Copy the zip files anywhere in the file system, for illustration purposes I will assume ‘c:\ICHI-2015-SubmissionDeliverables’ folder.
2. Unzip ‘hcDAChallengeBuildIndex.7z’ , ‘hcDAChallengeSearcher.7z’,’ CustomAnalyzers.7z’, and ‘QuestionsParser.7z’ files and add them to the folder created in step 1.
3. You should preserve the file structure as is. To do so, just open the zip files, then select the project folders and drag them into ‘c:\ ICHI-2015-SubmissionDeliverables’ folder.
4. Now you should have 4 folders
   1. hcDAChallengeBuildIndex
   2. hcDAChallengeSearcher
   3. CustomAnalyzers
   4. QuestionsParser
5. In Netbeans IDE, from its menu select File🡪Open Project…
6. In the Open Project dialog box, select hcDAChallengeBuildIndex project. Notice Netbeans also selected its dependency projects: CustomAnalyzers and QuestionsParser



1. Repeat step 6 for hcDAChallengeSearcher project.
2. Now, your two projects should be listed in the Nebeans projects. If you expand them, you should see something like this:



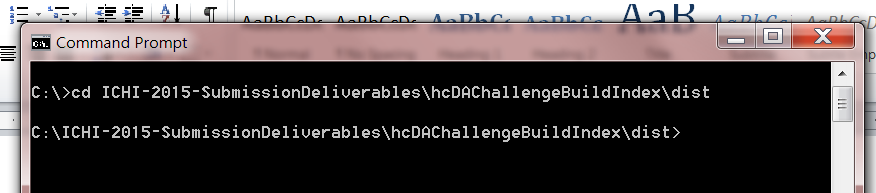
1. Right click on any project, and select ‘Clean and Build’ from the pop-up menu
2. The executable jar files will be generated in the ‘dist’ folder created by Netbeans build process.

**Pre-requisites to run the two software components**

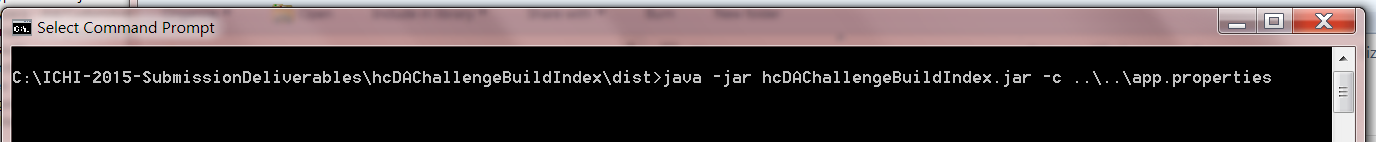
1. Locations of executables: Netbeans packages and builds everything in the ‘dist’ folder. Therefore, as we will show below, nothing else is needed to setup in the environment if the jar files are run from ‘dist’ folder.
2. Unzip the ‘supplementary.7z’ file, and extract the ‘app.properties’ and place it, continuing from our sample session above, in the ‘c:\ ICHI-2015-SubmissionDeliverables’ folder.
3. In the same ‘c:\ ICHI-2015-SubmissionDeliverables’ folder, create these folders:
   1. HC\_Index—a folder that the hcDAChallengeBuildIndex.jar writes the index to
   2. HC\_Data—a folder that is used by both components for input questions and questions corpus files, e.g. ‘ICHI-Challenge-Sample-Question-Corpus.txt’ and ‘ICHI-Challenge-Sample-Test-Questions.txt’.
   3. HC\_Test—all output results are written to this folder.
   4. HC\_Log—all log activities are written to this folder.
4. Properties file ‘app.properties’: this is an important file that is required by both components in order to run. Not all properties are required by the two components. Some properties are shared by both. You should not need to change anything in the provided ‘app.properties’ file if you follow the illustrative example here. However, it is important that you create the folders in step 3 above. Each line is a simple ‘key=value’ pair that sets some of the dynamic properties that the program needs each time it runs. Some of the properties, like folder locations, are needed so the components know where to place its generated artifacts like index database, log file, test results, etc.
5. Repeat step 2, except now extract ‘wn\_s.pl’, ‘ICHI-Challenge-Sample-Question-Corpus.txt’, and ‘ICHI-Challenge-Sample-Test-Questions.txt’ files in the HC\_Data folder created from step 3 above.
6. In ‘Instructions to run…’ below, we will point out any properties that you may need to change.

**Instructions to run the Indexing component applications**

1. Summary—this component takes an input questions corpus file, e.g. ‘ICHI-Challenge-Sample-Question-Corpus.txt’, and generates an index database and stores it in ‘.\HC\_Index’ folder.
2. From DOS command window, change the current working directory to the ‘dist’ folder that Netbeans generated:



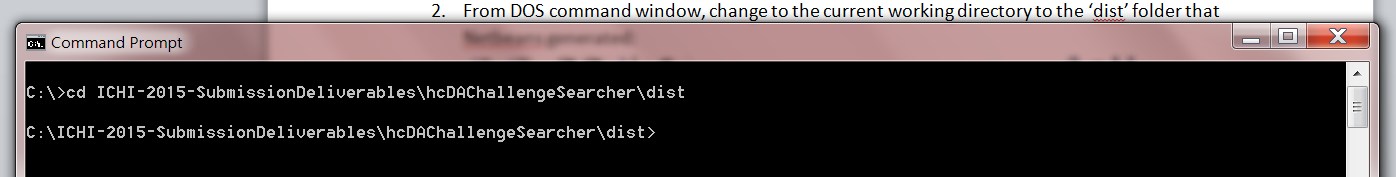
1. At the command prompt, type in: java -jar hcDAChallengeBuildIndex.jar -c ..\..\app.properties   
   The –c is command line option tells the program the location of the properties files.



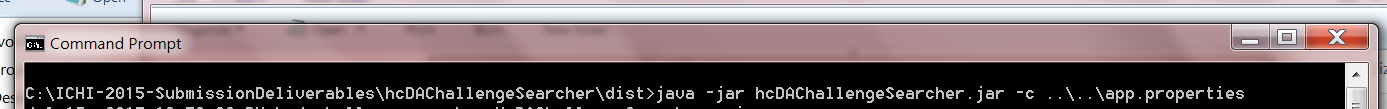
1. If everything goes well, the indexing component should generate a bunch of informational output which can be ignored unless of course you see the dreadful java runtime errors. Also, all screen output is logged in the HC\_Log folder. Each instance of program run generates a new log file. Log file names have this format: 07-17-2015\_11-20-22.log corresponding to ‘Date\_Time.log’.

**Instructions to run the Search applications**

1. Summary—this is the component that returns the solution to the Data Analytics Challenge sample questions. It takes as its input a questions file like ‘ICHI-Challenge-Sample-Test-Questions.txt’ and generates an output as we will describe here in this section. It also assumes that the indexing component was run first.
2. From DOS command window, change the current working directory to the ‘dist’ folder that Netbeans generated:



1. At the command prompt, type in: java -jar hcDAChallengeSearcher.jar -c ..\..\app.properties  
     
   The –c is command line option tells the program the location of the properties files.



1. Again, if all goes well, the search component generates some diagnostic information and an actual output in this format repeated for all the 10 questions in the questions file:

Found 95 document(s) (in 0 milliseconds) that matched query 'ID: 20

Topic: Cough medicine for a diabetic

Body: After having a cold last week, I have developed a dry, tickling cough. Can anyone recommend a cough medicine for a diabetic?

':

(1) [197] Can anyone recommend a good cough suppressant that does not raise bg?

(2) [169] I have been a diabetic for 25 years; I have been experiencing pretty severe tingling and needles in my left foot (they co

me and go). Also, I was diagnosed with Cubital Tunnel Syndrome in my left elbow at the end of 2010. I read a lot of threads talking about the supplements which help with DPN. How long can I take these for? I'm only in my 30s. Can I take those for the rest of my life, if they work? Does efficiency of the supplements wear off after a while?

(3) [188] Is there a beer we Type II can drink? Can you suggest one?

1. The output represents two sections: the input part and an output part.
2. The input part is repeated so you know what was the input question. You can ignore everything except the ‘Body: “ section (in Red text) and this is actually what the search component consumes. The rest is just for informational purposes only. The question ID is used for reference only.
3. The output section represents the system response with the required 3 output questions that match the input question/query. The question id from the corpus is also generated for reference only as was the input question ID.
4. This component also writes the output in a succinct compact manner in a file in the HC\_Test directory, e.g. ‘S0\_100.txt’ file. It has the following content:

11,193,137,152

12,127,155,183

13,118,138,113

14,113,104,103

15,113,104,122

16,113,104,192

17,169,116,183

18,155,112,145

19,197,188,193

20,197,169,188

Each line corresponds to the input question from questions file. There are 10 questions in this case. All of these are questions from the input sample questions and output from the questions corpus file. The first number is the ID of the input and the next three the output from the corpus file. For example, this line ‘11,193,137,152’ corresponds to question 11, with response question IDs 193, 137, 152.

1. The output file name has a convention like: {S|F}[0,1,2,4]\_TestID.txt. That is it begins with either ‘F\_’ of ‘S\_’, followed by a digit in this list: 0, 1, 2, 4, followed by ‘\_’, followed by some number we designate as test run id. This number can be controlled from the properties file: ‘Test\_Id=100’. The list 0,1,2,4 is a number that is controlled by ‘Sim\_Idx=0’ line from properties file and corresponds to the ranking function used to rank documents. Changing this value actually affects the results output, but we use it as a convenient bookkeeping mechanism in the output file name so we know which algorithm was used to generate what output. The initial letter ‘S’ or ‘F’ also has meaning and this value designates whether a synonym file was used, for ‘S’, or not ‘F’ when doing the searching. It is controlled by the presence of this line in the properties file: ‘Wordnet\_File=C:/ICHI-2015-SubmissionDeliverables/HC\_Data/wn\_s.pl’. If it is not present, the program will search without synonymic query expansion and the letter ‘F’ is used as part of the generated output file name for bookkeeping purposes.
2. If you also want to capture the human readable output from the command line you can redirect the output to a file as in:

java -jar hcDAChallengeSearcher.jar -c ..\..\app.properties > c:\ICH

-2015-SubmissionDeliverables\HC\_Test\output.txt

**Data Input Assumptions**

1. Both executable components expects the input questions corpus and the questions sample files to be in this format as specified in the sample data given for testing:

[12]Exercise Advice

What kind of exercise is most beneficial to diabetics? Should I do cardio or weights or swimming or a combination?

…Repeat for the rest of the questions

1. In other words, the schema should be as follows:

[id] topic NEW\_LINE

‘Question content’ NEW\_LINE (can be repeated as many times as required]

1. The parser that we developed is specific to this input structure and if it deviates, it will break.

**Netbeans Projects on GitHub**