CIS 350-001—Homework 1

Jonathan Levine
jonlevi@sas.upenn.edu

January 24, 2018

Question 1

I designed my application in two files: one that deals with user interactions and one that deals with the data extraction and analysis. I chose to separate these two processes in order to modularize my code and make it adaptable to later changes.
In the Data class I created two separate classes for the State object and a helpful tuple object for locations. I chose to make these classes in order to make the code more concise and neat, as it is easy to store data structures containing the State object, each containing a location object. It also helped with sorting and storing, as the state object could implement the comparable interface as well as a unique hash Code. I specifically chose not to create a Tweet object as I think that given the other data structures (the maps I used), it would have been redundant. I also made a careful decision that the Main class never had to know about any of the classes made in Data.java. Those classes were only for internal use, and any information accessible through the Data objects public methods did not return any references to the internal classes in the Data class.

Question 2

Since I separated the Data class and the Main class, it would be very easy to reuse the Data class that I made in a new program that does not use the command line or scanner class, but uses some sort of GUI to allow users to make decisions. Since the Main class and the Data class only interact through the Data object that is returned, the program could easily be modified to take in user interactions in a different way and to still use the same Data object without having to change anything about it.

The Data class was built with its fields as general data structures (maps and sets) to contain the tweet-/state information, and not in any file specific formats. Therefore if in future renditions the data comes in in a different format, it will just require an overloaded constructor that builds the data structures in a different way, but since the structures themselves will be unchanged the rest of the code that does the analysis can be reused as is.

Furthermore, since the Main class reads in user input and then passes the command onto separate helper functions for each option, it would be easy to implement more options in this program. One would just have to add some feature to the Data class along with a public method returning some information, and the Main class could create a new method to display that information. The format and interaction of the two classes allow for easy dynamic adding and/or removing of the features of the menu.
I also would like to note that I assume that a hash tag is a # sign followed by alphanumeric characters of the underscore character. I used the regex term $\#(w+)$ to represent this.

Question 3

The code was designed to be easy to read. The user input and the data analysis are completely separate and are therefore much easier to reason about. The data analysis tools are accessed through a Data object that is initialized with a data set, and with public methods that access necessary information. This compartmentalization of function allows for easy reasoning when dealing with initializing a data set and extracting pieces of information in an organized fashion. Furthermore, all of the more involved methods are separated into separate functions for readability and testing. For example, the main method of the Main class calls a helper method to get user input, which then passes the answer to another method which chooses their option, which then passes it onto its appropriate action method etc... This logical flow from method to method allows for easier reasoning and debugging. Also, the creation of the State class in Data.java allows reasoning about some of the data within the Data class to be easier. The state class contains all of the information about the state, including the number of tweets from that state, and a map can easily be made mapping state to other information about it. This allows for simple data analysis methods that are readable and debug-able.

Question 4

In terms of algorithmic efficiency, I followed the principles of ”lazy approach”, and rather than pre-process all of the data for every query, I only do work when necessary. All of the input/reading the data happens during the construction of the data object, but the heavier workload of the sorting and parsing only happens in a lazy way, i.e. only when necessary. I have chosen this approach because preprocessing would be less efficient, assuming that a user isn’t going to want to, say, look at all the states sorted by number of tweets multiple times. Therefore it saves the work for then. Furthermore, I used regular expressions for string parsing which is optimized over iterated over characters and matching against characters. Using regexs and pattern matches allow for some of the more computationally expensive operations (such as searching for hash tags) to be optimized. I also made sure that I never have to read the data more than once. Once I read the data the first time, I initialize all of the data structures I will need for the rest of the life of that data object. In addition for this process being more flexible, it is also optimal in that reading in the data can be an expensive operation to do multiple times, whereas access in a hash Map or hash Set is fast and inexpensive. That being said it did require more memory in order to store these data in these data structures, but not too much memory as to be an issue of memory usage.

Question 5

I thoroughly tested the program as a user: I played with the user interface and made sure that the program could accept proper as well as bad inputs, bad files, incorrect spellings, variations of capitalization, etc. For the actual data analysis part, I made a few JUnit tests to test the functionality of some of the specific functions, such as the regex matching and the sorting operations. I also tested the IO methods with sample text files to make sure that the reading and writing was happening properly. Throughout the course of the code I had a few assert statements to make sure that at specific checkpoints some of the data structures were initialized or filled etc. however those were deleted in the final version.