Checkpoint 2
March 22 2017

Task 1. Matplotlib

In this task we will be writing "wrappers" for the matplotlib package in python. You were exposed to matplotlib in CSCI 203. It is a package that allows us to make really nice plotting with simple function calls similar to Matlab. A wrapper function is a function in a computer program whose main purpose is solely calling a different function. The wrapper may also do some converting from one format to a different format that is needed. The real main goal of the wrapper function is to provide a uniform interface for interacting with the program. Here we will be doing this for the plot.py file. I have already added the needed start of the interface and import for these packages. Note the matplotlib.use("TkAgg") in the import. This is to make sure it will work correctly for students using their own packages on a Mac, as the two different graphic interfaces can have a bad effect on each other. This class is a wrapper too! As you can see, the only real functions/methods we need to implement are static! This is because we want a single place to go to call needed functions. This is commonly done in object-oriented programming and design. (One reason many seasoned programmers like procedural coding better.) You will have to write the details for the twoDScatter and twoDBar. The twoDScatter plot will make a 2D scatter plot of points (x,y) by calling a function from Matplotlib. The x-values will be in a python list from xList and the y-values will be in a python list from the yList. I leave options related to color and size up to you. Note that we may want to add an optional x labels or y labels (we can forget about these for the time, and come back to these later when we try to make the program look better). The twoDBar is similar to the twoDScatter. We can think of the xList as the location on the x-axis we would want to put the bar, and yList is a python list of the heights of these bars using a function in Matplotlib. Again, we can forget about the xLabels and yLabels for this checkpoint.

Make sure to test your wrapper interface with the testMyPlot that just plots some made up data.

Task 2. Getting Some Statistics.

Next we turn to getting some simple statistics from our email dataset. We will be doing this by using the methods in the stats.py file. Again, the Stats class here is a type of procedural wrapper. In general, these methods don't require us to think of objects or worry about a shared state-variable-attribute. However, this is a traditional method to group related methods in object oriented programming/analysis. We will be modifying this file a lot in the project as we keep adding new algorithms that we may learn to speed-up our program! Right now we are just adding the basic function calls (findFreqDic, topNSort, bottomNSort). We will not have to worry about the Heap type functions right now. In findFreqDic, we take in a python list of words. These words may not be unique. We will count the frequency of the words and place it in a python dictionary with the keys being unique words and the value being the frequency count. This is very similar to the word analysis done in your CSCI 203 project. In topNSort, we will take in a dictionary (similar to what you made in findFreqDic) and use the built-in python sort to sort based on frequency. We will return a dictionary of the length n that contains the n most frequently used words as keys and frequency as data. In bottomNSort, we will do a similar

process as in topNSort, but now return a dictionary with the least common keys in the given dictionary.

Task 3. Interface

In main.py, I have started the interface for the program. This part of the program follows more procedural programming style in place of object oriented design. This could have been just as easy with objects/classes, but I wanted to provide experience with procedural programming style for a learning experience. There are two real parts to this interface. Part 1 is the UserInput class in util.py. This is a class that is used to store the shared states of what the user provides to us (i.e., the values for variables). I have added some of the variables we may need. However, you are free to add additional ones as needed for your project. Part 2 is the set of methods that produce the input/output interface to interact with the user. For this checkpoint, you will finish the code for user_interface. This will require you to make a list of Documents to store from the document reader done in Checkpoint 1. Part of the checkpoint is to see how you figure out completing this step (i.e., seeing you connect and interact with multiple objects…. Something that you have had little experience doing before now, so it will take time to think about it). After, you can now complete the interface for the topicAnalysisTrain and topicAnalyisEval. These will use the Documents you made, get common words from them, use the functions in stats.py and plot using plot.py. For now, the analysis is simple. Just ask the user for how many words and find the most common and least common.