**LAB 1 – Reddit Recursion**

For our initial lab concentrating on recursion, I chose the Reddit Sentiment Analysis option. Essentially, the goal of the lab was to be able to traverse the “commentforest” of comments and replies in a given Reddit thread using recursion. With the PRAW library, each comment was appended to a Python list, and each comment’s respective replies was appended to a list on the individual comment’s node, which was where recursion came in.

At first, it was difficult to understand that Python’s lists were indexed, but also possessed features of Nodes (such as in JAVA). In other words, while each comment had a particular index, the “.replies” feature of PRAW allowed access to the replies of the comments, which were also indexed. The way I figured this problem could be tackled was by using a for-loop to iterate through the initial, or top-level, comments, and then using a recursive case within the for-loop to check for the existence of replies to each comment (not all comments had replies). If the “comments[i].replies != None” comparison returned as true, it would then recursively call the analysis method to iterate through the replies of the comments and not just the comments. **This method was initially tested by simply having it print out all of the comments and replies of a given thread and comparing them visually on my terminal to the actual URL.** It was successful, which then allowed me to begin the analysis with the libraries given by Dr. Aguirre.

Once the traversal of the “commentforest” was achieved, it was rather simple to use the provided library to analyze the comments. Three separate lists were created as global variables, one for positive comments, one for negative comments, and one for neutral comments. In the same **process\_comments** method, with each iteration of the for-loop, the returned value for **get\_text\_neutral\_proba**, **get\_text\_positive\_proba**, and **get\_text\_negative\_proba**, were stored in variables called **negScore**, **neuScore**, and **posScore**. Once these values were obtained, they were compared to see which one had the highest value, in which case, the actual comment itself was “deposited” into its respective list (negative, positive, or neutral). Since these value generations and comparisons took place with every iteration, the values were renewed with each respective comment and reply, which I believe assisted in minimizing the **memory tax** of the method.

In addition, the program then simply printed the negative comments list, positive comments list, and neutral comments list. However, since an attempt at the extra-credit was made, I opted to make a user-menu using a makeshift switch case in order to allow the user to choose what they wished to display. In addition, the option of providing the desired URL for analysis was given to the user. I had hoped to implement a “change URL” feature in my main menu, but this was not possible due to time constraints.

Since the extra-credit required that the oldest comment, oldest positive, and oldest negative comment be printed, I chose to use a PRAW function that pre-sorted the comments before they were imported as “comments” in main (see line #38 in Main.py). Since the comments were already sorted from oldest to newest, it was easy to provide the oldest comment in each list by simply holding a counter and updating the counter with every request to see the oldest comment. This prevents duplicates and allows the oldest, second oldest, third oldest, etc. to be printed in any requested list. Perhaps this was not the way that these methods were intended to be implemented, however, the specifics were not specified and I feel that my program performed the requested tasks.

Once this method was completed, it was tested by using an assortment of Reddit URLs, some of which were expected to possess more positive, more neutral, or more negative comments. I simply updated the URL with each program run and observed the results. The test URLs used were:

**Politically Charged URL** (expected negative comments)**- 303 comments**

https://www.reddit.com/r/politics/comments/9056d9/reddit\_users\_are\_manipulating\_google\_images\_to/

**Gaming URL** (expected negative/positive comments)**- 270 comments**

https://www.reddit.com/r/FortNiteBR/comments/9er3po/troll\_people\_with\_this\_please/

**Original Recursion URL in Lab description** (expected neutral comments) **– 200 comments**

<https://www.reddit.com/r/learnprogramming/comments/5w50g5/eli5_what_is_recursion/>

**Larger, Politically Charged URL** (expected negative comments) **6,000 comments**

https://www.reddit.com/r/politics/comments/9fhmra/trump\_claims\_3000\_people\_did\_not\_die\_in\_puerto/

While some of the negative comments were categorized as neutral or positive, and some positive were categorized as negative or neutral, the program provided a rather accurate output. As Dr. Aguirre had explained, the sentiment analysis software compares comments to keywords and phrases, which are deemed positive, negative, or neutral by the developers. Since certain things like sarcasm are hard to detect, the program may categorize comments with limited accuracy.

**Time Complexity**

The speed of the program was considerably quick and exhibited signs of running at a **O(n)** speed when there were limited replies to comments and **O(n2)** when the program had heavy comments and replies to comments. Even on the larger, politically charged URL, the comments were analyzed at a considerably quick rate. This is due to the fact that the initial for-loop for top-level comments runs in **O(n),** and, if the comment at index “i” has replies, the recursive-call calls the same for-loop one more time, hence **O(n2)** in the “worst-case.”

**Conclusion**

This was definitely a very rewarding and beneficial lab for me, as it allowed me to learn a lot more about Python and also about recursion. I was under the erroneous impression that recursion implied absolutely no iterative functions (for-loops, or any loops), however, my method exhibited the best of both worlds. Python is proving to be very efficient and to the point, which is actually something new to me having only really worked with JAVA. I look forward to learning this program in more detail as I progress in my computer science career and exploration.

**Appendix**

* positiveCommentsList = [] # global list created for positive comments
* negativeCommentsList = [] # global list created for negative comments
* neutralCommentsList = [] # global list created for neutral comments
* negCounter = 0 #the following counters are for the access of the oldest, oldest positive, oldest negative, and oldest neutral comments

posCounter = 0

neutralCounter = 0

oldestCounter = 0

* submission.comment\_sort = 'old' ## this pre-sorts the data from oldest to newest before importing them as submission, as read on PRAW docs
* def process\_comments(comms): # method implemented for lab
* def display\_list(list): # simply prints linear lists created after analysis
* def display\_oldest\_negative(): # extra credit method
* def display\_oldest\_positive(): # extra credit method
* def display\_oldest\_neutral(): # extra credit method

**Academic Dishonesty Certification:**

“I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any students in the class.”

Angel Villalpando

09/17/2018