Austin Ross

CS441 Text Analysis Report

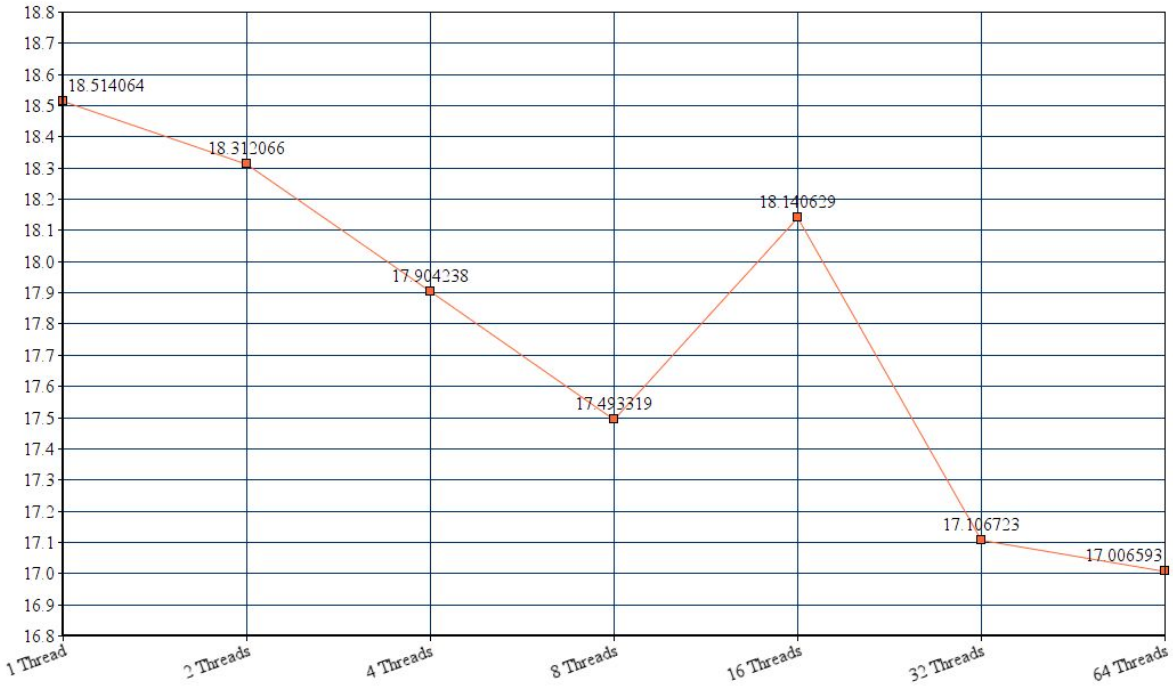
The initial part of this report will cover my findings from this program, including average execution time as a function of the number of threads, as well as the information summation of the text analysis of n-char substrings. Following that will be a short summary of my experience while putting together this program.

My Findings

Number of Threads	Average Execution Time (s)
1	18.514064102000003
2	18.31206623666667
4	17.904238362666668
8	17.493319461666665
16	18.140629453666666
32	17.106723469000002
64	17.006592855666668

Below is a graph created using the values from the table above, as you can see from both the table as well as the graph, the execution time of the text analysis (in most instances) decreases as the number of threads being utilized increases (with the exception of the 16 thread result.

Austin Ross



My program also confirmed that as we evaluated more characters at a time, the total information in the file decreases. This can be seen from the values reported below.

Total Information of n-char Evaluation	
n	**Information Summation**
1	514483.42205134436
2	109448.52659697284
3	31658.460220207045

The probability distribution for the program can be seen in the source code file submission within the ProbabilityDistribution.txt file. I ran the program once in order to output the information to this file, then commented out the lines that are responsible for the file output as to not continue running them without reason (the probability distribution is the same regardless of # of threads being used.) The program also outputs the results to the console, but since there is so much data, most of it cannot be seen, so I decided to create this file as a means of viewing the distribution, but if you would like to confirm that the data in the file is in fact coming from the program execution simply uncomment these lines and re-run the program.

Austin Ross

Summary

While piecing together this program I ran into quite a few roadblocks given this was my first time truly coding in a functional language. Prior to this class I had some experience with Clojure as I had to use it for some basic evaluations in an internship last summer, but my experience there was nothing compared to the requirements of this program. I spent a lot of the time simply trying to figure out what would be basic string manipulation in another language and how it is done using clojure. After I determined the proper way to loop through a string I was able to pretty easily create the key values for my map utilizing substrings of length 1/2/3. When I realized how to gather my keys I then had to figure out an entirely new problem, how to count how many times that particular substring occurs within the string. I eventually determined how to check if the substring I am currently at in the loop has already been seen, and if so I could increment the value +1. If it had not been seen to that point, I added the substring to the map with the initial value of 1. From there it was pretty easy to find the probability distributions (# of times a particular n-char substring occurs / total # of n-char substrings). Finally I used the values printed in the probability distribution as part of the information summation equation, and summed up all of the results.