Created by Fransandreimmanuel Colendres in compliance with the requirements of CoE 135.

This project is the implementation of a Markov Chain text generator, with an emphasis on the dictionary building process. It aims to explore the processing of large amounts of data into useful information by creating a Markov Chain text generator. By applying different methods in the dictionary building process, the project attempts to find a way to efficiently scale the processing time of data with the amount of data input. This is accomplished by applying the CoE 135 concepts of threads and concurrency-- by utilizing multiple threads to process data, the project aims to create a program that is faster at processing large amounts of data than its single threaded counterpart.

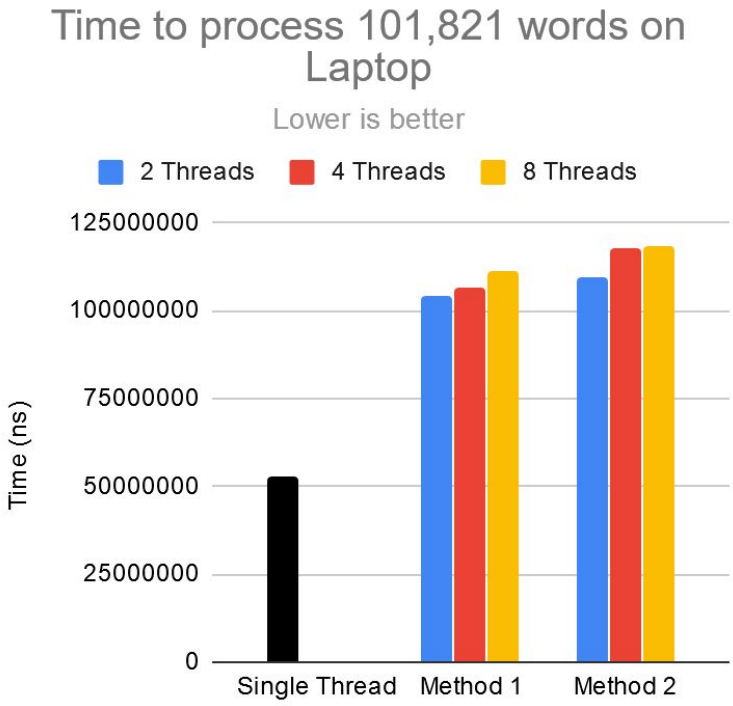
MarkovTextGen_ST contains all of the files needed for the text generator's single threaded version.
MarkovTextGen_MT contains all of the files needed for the first method of the text generator's multi threaded version, wherein the program assigns an entire .txt file to each thread to process.
MarkovTextGen_MT_2 contains all of the files needed for the first method of the text generator's multi threaded version, wherein each thread processes a part of every .txt file.
All of these folders contain their own numbered .txt files. These are the files that will be processed by the program. masterlist.txt contains the relevant information on these files, like word count, and name and author of the literary work. The bounds indicated by the word count at the bottom indicates what files add up to the word count (eg. 101821 words in 00000.txt-00032.txt)

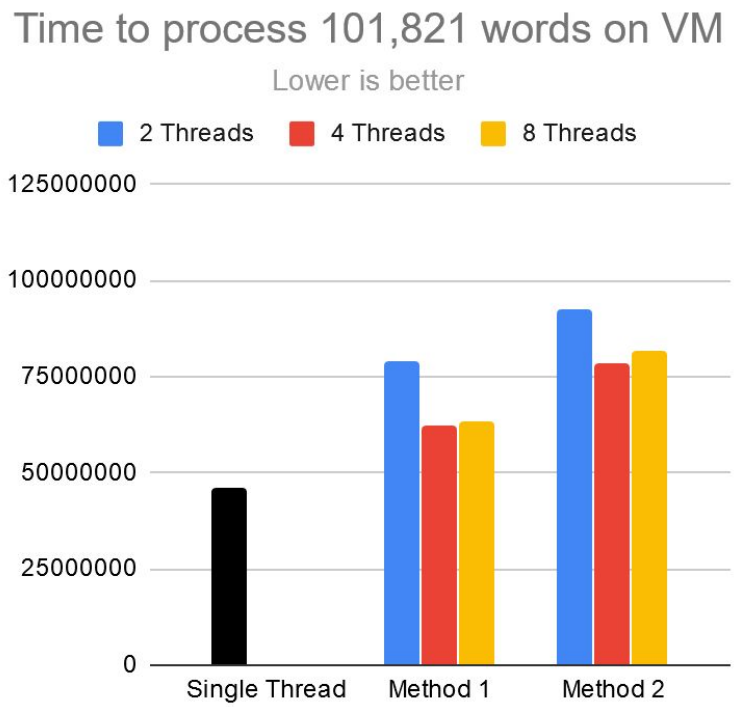
For all of these methods, just run make while in their respective directories to compile the program, and run Colendres_MP to run the program.
For the multi threaded version, edit threadCount in order to change how many threads are employed
For the multi threaded version, edit fileTot to define which .txt files to process. (eg. fileTot = 200 will make the program process 00000.txt - 00200.txt). For the single threaded version, this can be done by editing upper bound of the 'for' loop incrementing the variable 'filenum'

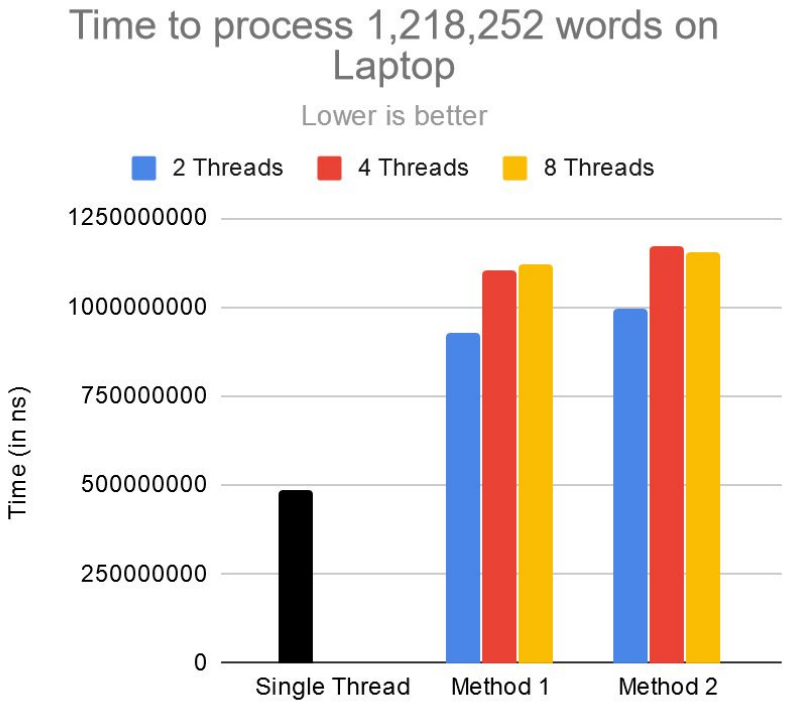
The project uses the murmurhash library by Peter Scott, obtained from https://github.com/PeterScott/murmur3.

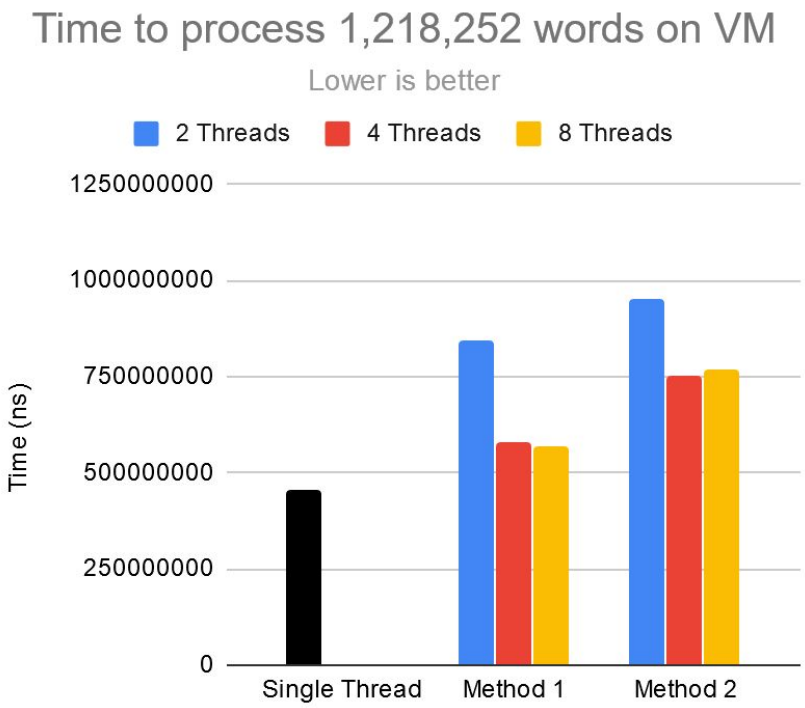
This project resulted in the multi threaded versions running slower than the single threaded versions. Results may vary depending on the hardware used. Multi threaded methods may change significantly depending on the hardware specifications of the machine running it. Single threaded methods may change significantly, but not as drastic as the multi threaded methods.

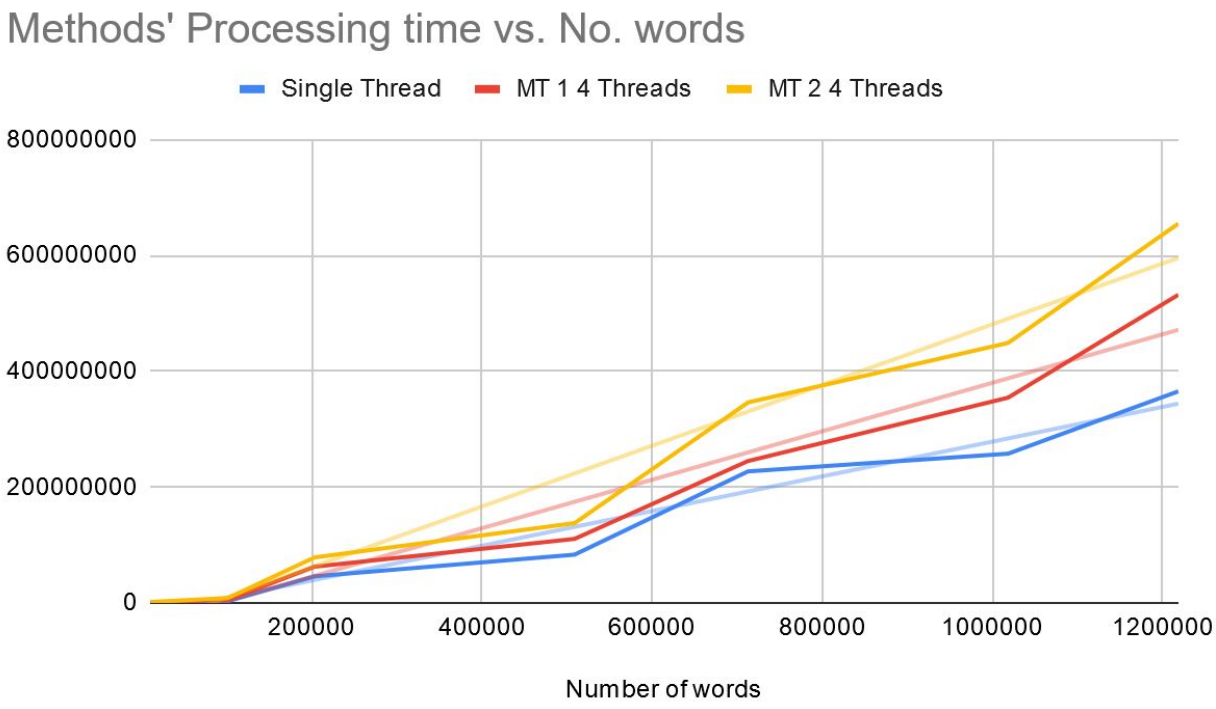
The graphs below illustrate the performance of the programs. The laptop ran an i5 7200U @ 2.5 GHz, the VM ran an i5 4670k @ 3.2 GHz. For the graphs illustrating scale, the measurements were taken on the VM.

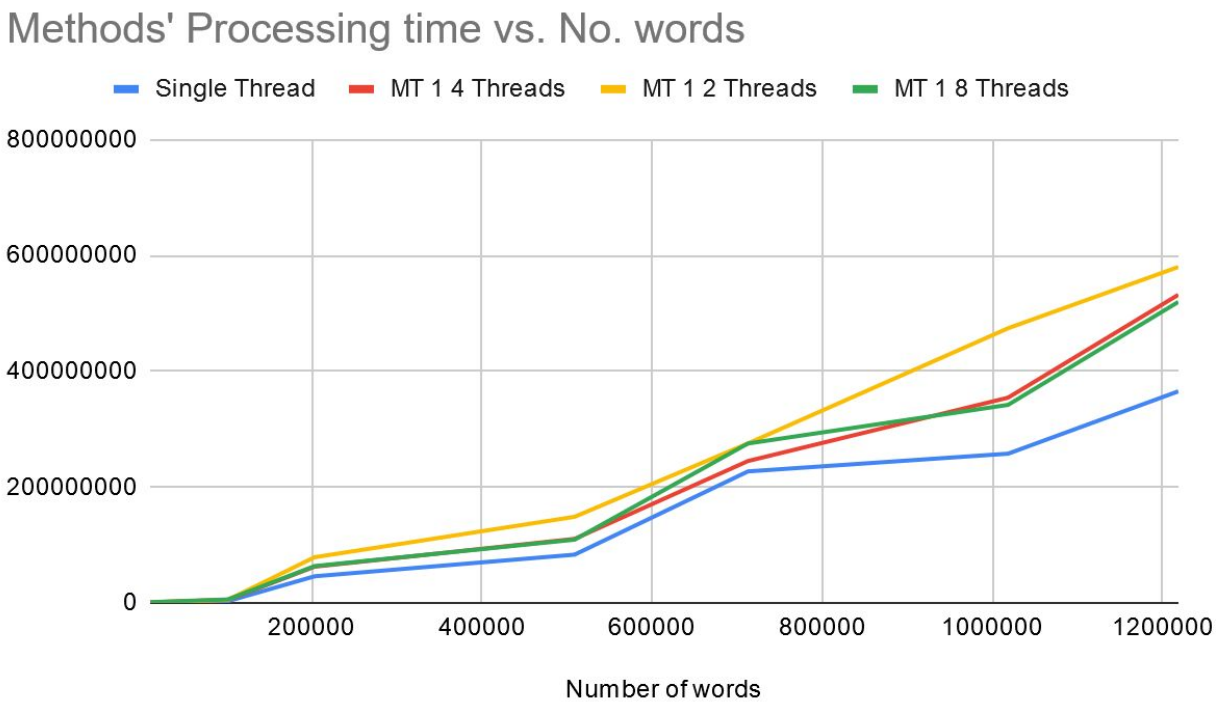

Time to process 101,821 words on Laptop
Lower is better
2 Threads
4 Threads
8 Threads
125000000
100000000
75000000
50000000
25000000
0
Time (ns)
Single Thread
Method 1
Method 2

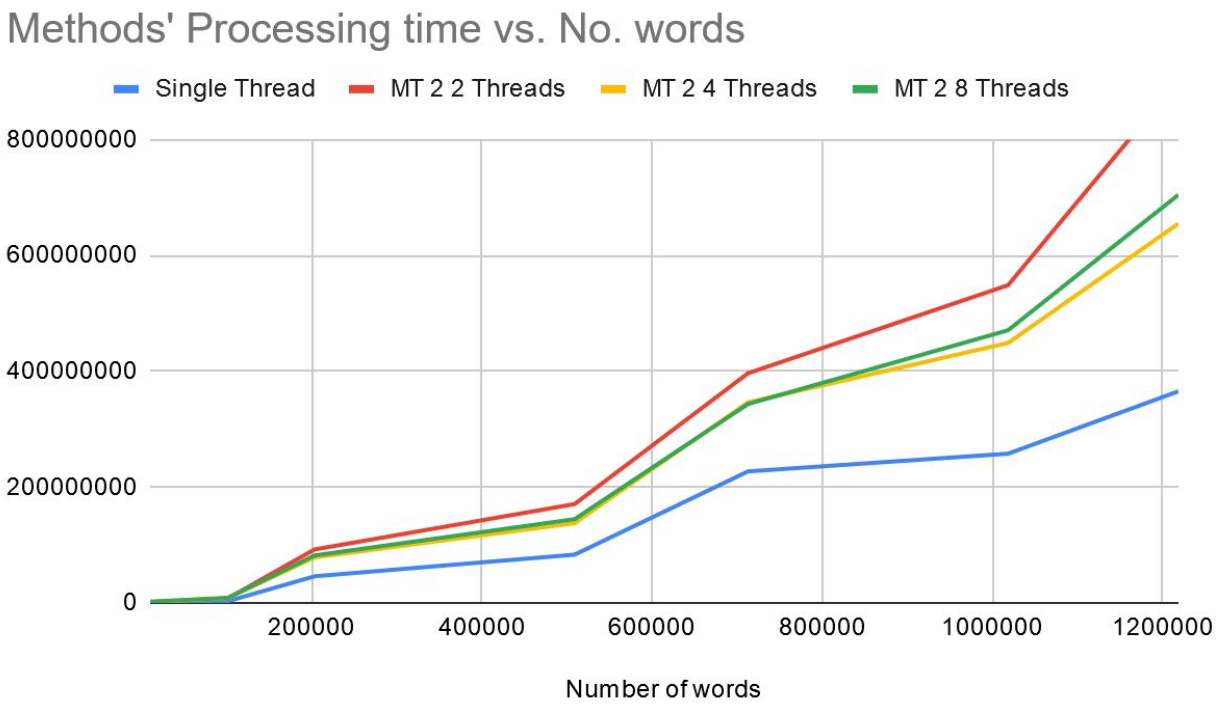

Time to process 101,821 words on VM
Lower is better
2 Threads
4 Threads
8 Threads
125000000
100000000
75000000
50000000
25000000
0
Single Thread
Method 1
Method 2


Time to process 1,218,252 words on Laptop
Lower is better
2 Threads
4 Threads
8 Threads
1250000000
1000000000
750000000
500000000
250000000
0
Time (in ns)
Single Thread
Method 1
Method 2


Time to process 1,218,252 words on VM
Lower is better
2 Threads
4 Threads
8 Threads
1250000000
1000000000
750000000
500000000
250000000
0
Time (ns)
Single Thread
Method 1
Method 2


Methods' Processing time vs. No. words
Single Thread
MT 1 4 Threads
MT 2 4 Threads
800000000
600000000
400000000
200000000
0
200000
400000
600000
800000
1000000
1200000
Number of words


Methods' Processing time vs. No. words
Single Thread
MT 1 4 Threads
MT 1 2 Threads
MT 1 8 Threads
800000000
600000000
400000000
200000000
0
200000
400000
600000
800000
1000000
1200000
Number of words


Methods' Processing time vs. No. words
Single Thread
MT 2 2 Threads
MT 2 4 Threads
MT 2 8 Threads
800000000
600000000
400000000
200000000
0
200000
400000
600000
800000
1000000
1200000
Number of words