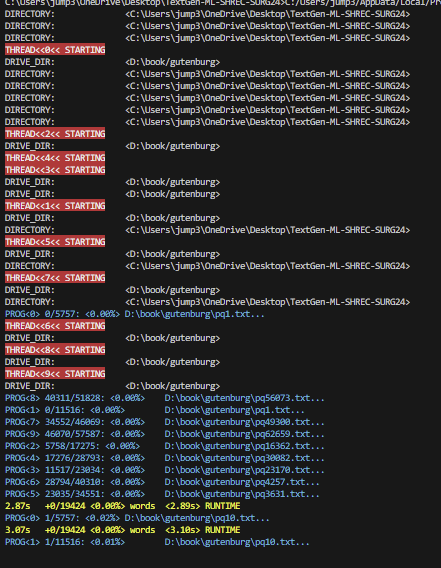
before (serial)

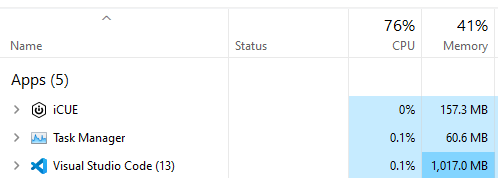




would’ve taken 50 days

**after (parallel)**10 threads (41 days)





Had the idea of making a Red-black tree with the dictionary but was afraid of RAM usage. Solution was to keep data in a text file and just load each line at a time into RAM. This was extremely slow, even with parallelization. Serial: 50 days for Gutenberg. Para: 42 days.

Also was going to originally make a separate ML model to detect if a word was “proper” but was able to sort out words so well that this wasn’t needed. If there was a small hiccup, the neuron connection between that word would be so low that it practically wouldn’t exist in training anyway.

Dictionary types

* RAMmin txt file: word per line, load a line (word) into ram one at a time
  + Too slow
* RAMmax Red-black Tree: key by string
* RAMmax Red-black Tree: key by int encoded string
  + Tiktoken:Gpt2
    - If a string is too long it has multiple tokens and is longer then just the string
* RAMmax enumerate sorted list: proper coding ‘dictionary’ term
  + Would have to recreate object for every addition, doesn’t really work for this approach
  + DOES absolutely make sense for **saving dictionary** (our term) as a nonRAM file
    - Convert into sorted list, enumerate, save as ‘.bin’
    - Load back into a new file
  + Any dictionary would have to be converted to this anyway for the model to use

Dictionary Conclusion:

* Load whole data into RAM, convert into pure words
* RAM Red-Black Tree (key by string), save from RAM as enumerated sorted list ‘.bin’ file
* Load into a model from this file

After RedBlack Tree with maxRAM

A computer screen shot of a program

Description automatically generated

unfathomably faster