Word Autocomplete Proposal: Prefix-Complete

Group: Kevin Durand (kpd216), Jonathan Tavarez (jt3481), Max Hahn (meh2280)

Parallelized Algorithm Description

The algorithm being parallelized is BFS over a trie of word frequencies from a given input dataset.

Getting Input Data (Link to dataset)

We will use a kaggle dataset containing a common list of english words and their associated frequencies to rank all potential autocomplete suggestions by constructing a trie. The dataset is already cleaned so no preprocessing is necessary.

Sequential Algorithm:

In order to autocomplete a word we must know possible endings to a given word prefix. This can be determined using a trie data structure, which allows us to find possible completions by traversing the tree from beginning of our prefix to all possible endings/leaves of our trie. All possible endings/leaves will be ranked according to their frequency, and the most frequent one will be served back to the user.

Parallelization Plan (e.g., strategies, Par monad, REPA, Accelerate, etc.):

Our sequential algorithm requires us to traverse many branches of the tree from our starting point. We can parallelize this by sparking for each traversal of a given node's descendant up to a given depth. Different threads can build a thread-specific list of possible words and recombine at the end to find the word with the highest frequency count. Traversing the tree does not modify it which allows us to parallelize the traversal and each spark will then capture the word with the highest frequency associated. We can then merge the sparks at the end together and take the word with highest frequency associated.

Evaluation:

We will use threadscope with rtsopts to evaluate the creation and usage of sparks in the program to find the optimal depth at which we want to keep making new sparks for our parallel traversal. Since at a low enough depth too many sparks would be created and "fizzle" and not do productive work.

Resources

- Trie: https://www.geeksforgeeks.org/trie-insert-and-search/
- BFS: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/