PFP Project Proposal: N-Gram Counter

Grayson Newell (gln2109)

November 2025

1 Introduction

In Natural Language Processing, an n-gram is a subset of n consecutive tokens (words) taken from a longer sequence, padded with START and END tokens. For example, the sequence [START, 'The', 'dog', 'ran', '.', END] yields the following trigrams ($n = 3$):

[START, 'The', 'dog']; ['The', 'dog', 'ran']; ['dog', 'ran', '.']; ['ran', '.', END].

n-gram frequency maps (for unigrams, bigrams, and trigrams) form the basis of many primitive NLP models. This project will generate these maps from large text corpora.

2 Sequential Overview

I will create a corpus in the form of a large .txt file, at least 20MB, by concatenating books from Project Gutenberg or Wikipedia articles. The program will:

1. Process the file line by line, treating each as a sequence of tokens.
2. Pad each sequence with START and END tokens.
3. Count unigrams, bigrams, and trigrams, storing them in separate frequency maps.
4. Output the top ten n-grams for each n.

I plan to optimize the sequential implementation to use efficient data structures before parallelization, and experiment with datasets exhibiting differing levels of sequence-length variance.

3 Parallel Approach

My plan for parallel counting of n-grams is to process the corpus in chunks. This will require the following considerations:

- Finding an optimal chunk size to avoid load balancing and garbage collection issues.
- Observing how values of n affect performance.
- Efficiently combining counts from each chunk into frequency maps.
- Handling conflicts at chunk edges.
- Experimenting with dynamic chunk sizes to further improve performance.

Performance will be measured on 1-8 cores, using Amdahl's law to analyze speedup. This metric will then be compared for different implementation parameters (chunk size, data structures, etc).