Comparative Analysis of Sequential and Parallel Algorithms for N-gram Extraction using C++ and Python

Dylan Fouepe
E-mail address
dylan.fouepe@edu.unifi.it

Abstract

This project examines the performance differences between sequential and parallel algorithms for n-gram extraction and counting. Implementations are done in C++ using the OpenMP library and in python using the asyncio concurrent library. By processing a corpus of text, the study highlights the efficiency and scalability of parallel computing techniques compared to their sequential counterparts.

1. Introduction

The rapid growth of data has necessitated the development of efficient algorithms for text processing. One common task in natural language processing is the extraction and counting of n-grams, which are contiguous sequences of n items from a given text.Traditional sequential algorithms, while straightforward, often fail to scale effectively with large datasets. This project investigates the potential of parallel computing to enhance the performance of n-gram extraction algorithms. By leveraging the OpenMP library in C++ ad the asyncio library in python, we aim to compare the execution times and efficiency of sequential versus parallel implementations. This comparison will provide insights into the benefits and challenges of parallelizing text processing tasks.

1.1. N-grams

N-grams are contiguous sequences of n items from a given text, where the items can be characters, words, or other linguistic units. They are widely used in natural language processing (NLP) and text mining tasks for capturing patterns and relationships within textual data. N-grams of different lengths (unigrams, bigrams, trigrams, etc.) provide insights into language structure, syntax, and semantics. For example, bigrams (two-word sequences) can help identify commonly occuring phrases or expressions, while trigrams (three-word sequences) can capture more complex linguistic patterns. In this project, the focus is on extracting and counting n-grams from a corpus using both sequential and parallel algorithms. Efficient n-gram extraction is crucial for tasks such as language modeling, sentiment analysis, and information retrieval, making it essential to evaluate the performance of different computational approaches, including their scalability and speed in handling large-scale datasets.

1.2. Optimizing N-gram Extraction: Harnessing Parallel Computing in C++ with OpenMP and python with Joblib and asyncio

In this project, both C++ with OpenMP and Python with Joblib and asyncio libraries are employed to explore efficient algorithms for n-gram extraction from textual data. C++ is chosen for its high-performance capabilities and robustness in handling intensive computations. OpenMP, a widely used API for parallel programming in C++, facilitates the concurrent execution of

tasks accross multiple threads, making it suitable for optimizing the performance of algorithms on multi-core processors. The parallelization offered by OpenMP allows tasks to be divided into smaller units that can be executed simultaneously, thereby enhancing overall computational efficiency. On the other hand, Python is selected for its versatility and ease of use in rapid prototyping and development. Joblib, a popular library in the Python ecosystem, provides tools for lightweight parallel computing, enabling efficient execution of parallel tasks such as n-gram extraction. Asyncio, another Python library, supports asynchronous programming, which is particularly beneficial for handling concurrent I/O-bound operations efficiently. By leveraging asyncio, tasks that involve reading, processing, and analyzing textual data can overlap and run concurrently, reducing idle time and improving throughput. The choice of both languages and libraries allows for a comprehensive comparison of sequential versus parallel approaches in n-gram extraction. This comparison not only evaluates the performance gains achieved through parallel computing but also assesses factors such as scalability, resource utilization, and ease of implementation. Ultimately, the project aims to provide insights into the optimal strategies for leveraging parallelism in text processing tasks, thereby contributing to advancements in natural language processing and computational linguistics.