Elementary Systems Programming Project

01286120 Elementary Systems Programming

Software Engineering Program,

Department of Computer Engineering,

School of Engineering, KMITL

By

66011533 Eaint Kay Khaing Kyaw

Text Data Statics Collector

Introduction

Text Data Statics Collector is a command line utility that allows users to analyze and rank text documents found in a specified folder. It provides various insights about the documents such as word count, character count, line count, avg word length and most common words. Users can choose to sort the documents by different criteria, including word count, line count and character count. Additionally, the tool can search for a specified word within the documents and provide the locations where the word is found.

Motivation

This project was motivated by the need for a simple and flexible tool to quickly access and compare text files in a directory. Sometimes, we may want to find some words easily and may sort the files by different criteria. Moreover, this project mat be beneficial in various contexts such as content management, research or data exploration.

Features:

Document Statistics: Collect and display the following statistics for each text file:

- Word Count
- Character Count
- Line Count
- Average Word length
- Most Common words

Folder Statistics:

- Provide overall statistics for the entire folder, including most common words and their frequencies across all files.

Document Ranking:

- Ranking by Word Count: Ranks documents based on their word count in descending order.
- Ranking by Line Count: Ranks documents based on their line count in descending order.
- Ranking by Character Count: Ranks documents based on their cha count in descending order.

Word Search:

- Searches for a specified word within all documents in a folder and provides locations where the word is found.
- Generates a csv file including those locations

HTML Report:

- Generates an HTML report summarizing the analysis and rankings of the documents. The report includes a table with document statistics and a histogram of the most common words.

Usage

Command Line Interface

User can generate a html report and histogram including word count, line count, character count, average word length, most common words in each file and in specific folder.

- To sort by line count
  cargo run <input folder> line

  Example Usage
  cargo run "D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project" line

- To sort by word count
  cargo run <input folder> word

- To sort by character count
  cargo run <input folder> cha

- To find specific word
  cargo run <input folder> find_word <word>

  Example Usage
  cargo run "D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project" find_word HAHA

Implementation

This project utilizes several libraries, including:

- std::env: For command-line argument handling
- std::path: For working with file paths
- std::collections::HashMap: For storing and analyzing word frequency
- std::fs For file I/O operations
- csv: For writing search results to a csv file
- std::error::Error: For error handling

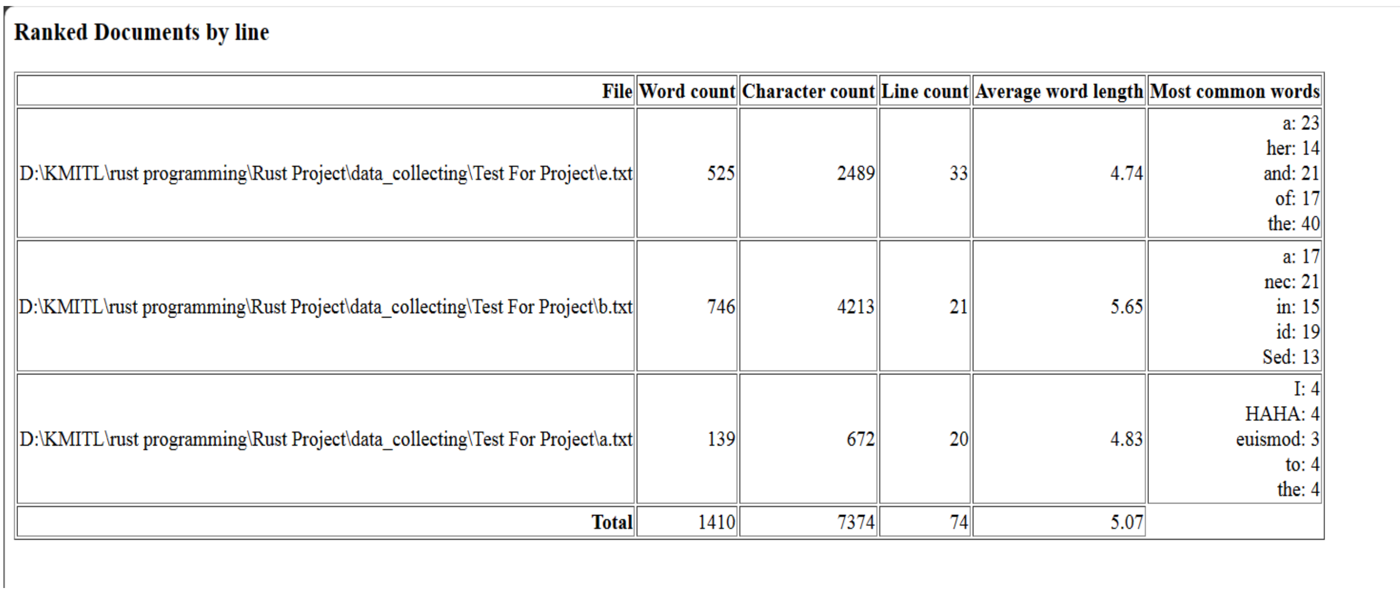
The core function of the Text Statistics Collector is encapsulated in the “Document” struct , which represents individual documents and provides methods for document analysis, ranking and HTML report generation.

Conclusion

The Text Statistics Collector is a versatile and efficient tool for analyzing and searching within text documents. It serves as a valuable resource for various use cases, including content management, data analysis, and research. Users can easily tailor the tool to their specific needs and preferences by choosing the sorting methods and searching for specified words within documents.

The project can be extended with additional features, such as more advanced search options, support for additional file formats.

Text Statistics Collector represents a powerful tool for text document analysis demonstrating the capabilities of the Rust Programming language and its ecosystem for building efficient and versatile command-line utilities.



Ranked Documents by line

File	Word count	Character count	Line count	Average word length	Most common words
D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	525	2489	33	4.74	a: 23 her: 14 and: 21 of: 17 the: 40
D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\b.txt	746	4213	21	5.65	a: 17 nec: 21 in: 15 id: 19 Sed: 13
D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\a.txt	139	672	20	4.83	I: 4 HAHA: 4 euismod: 3 to: 4 the: 4
Total	1410	7374	74	5.07	

Figure 1. HTML report for each file and folder showing word, character, line count and others

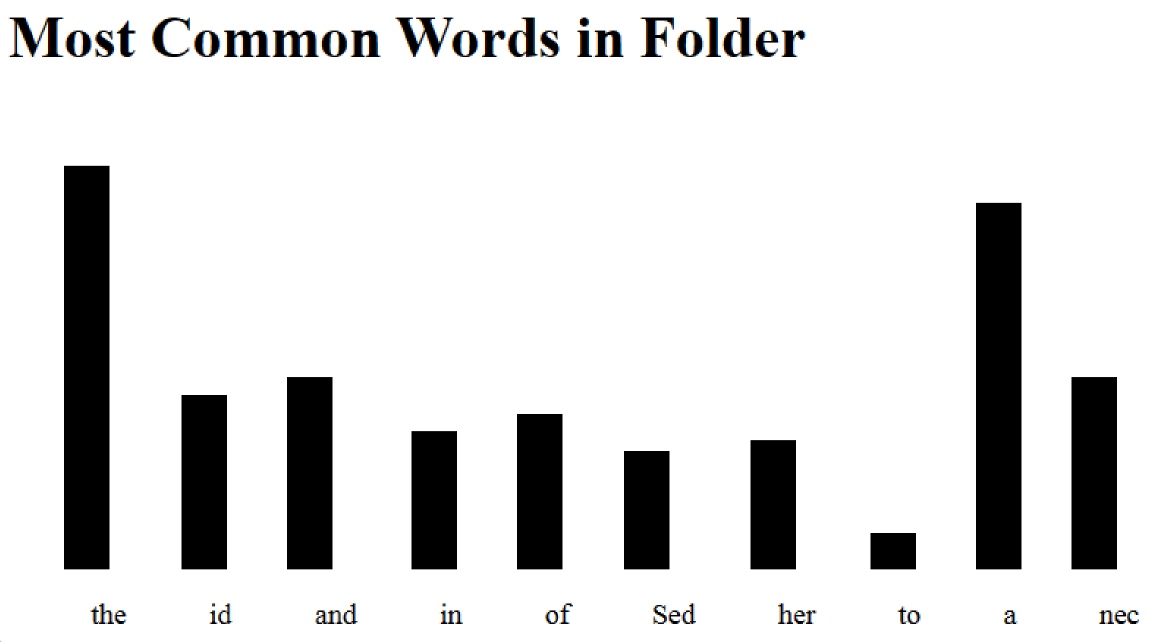
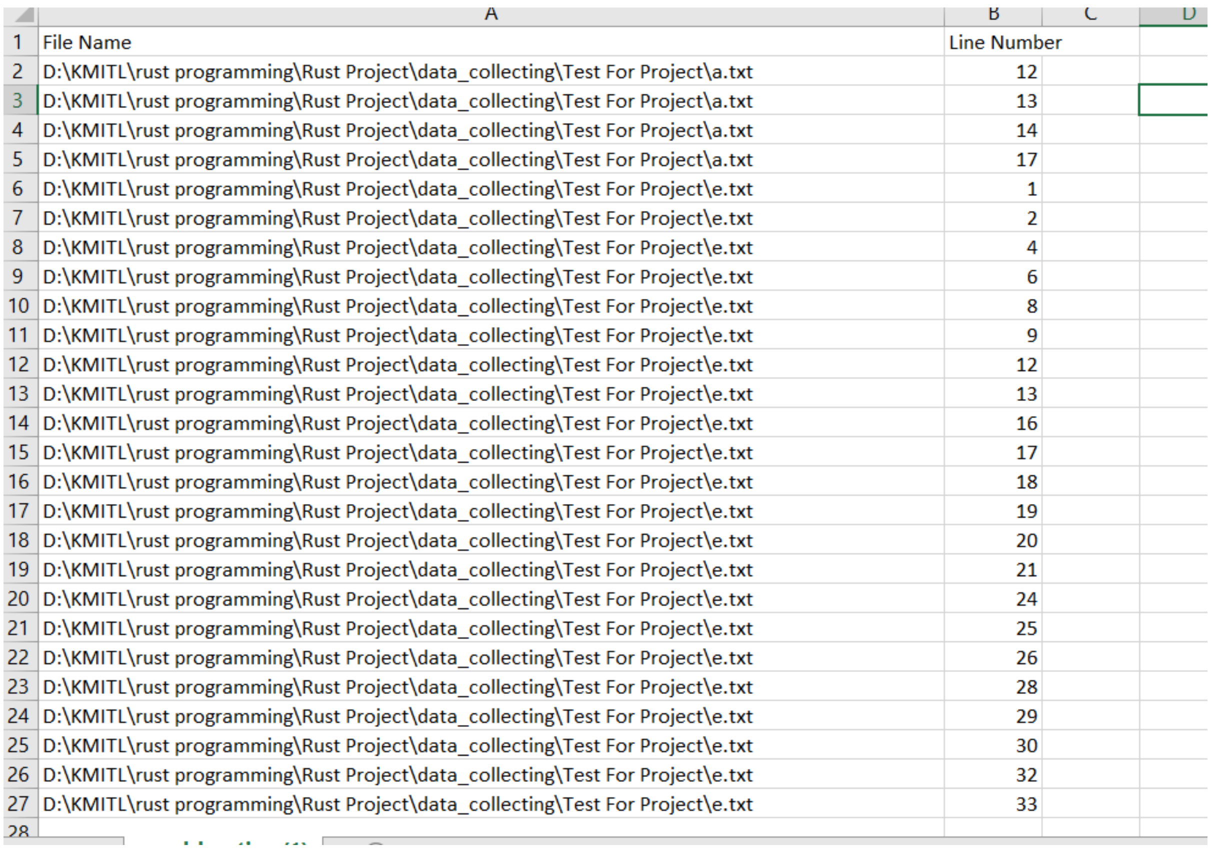
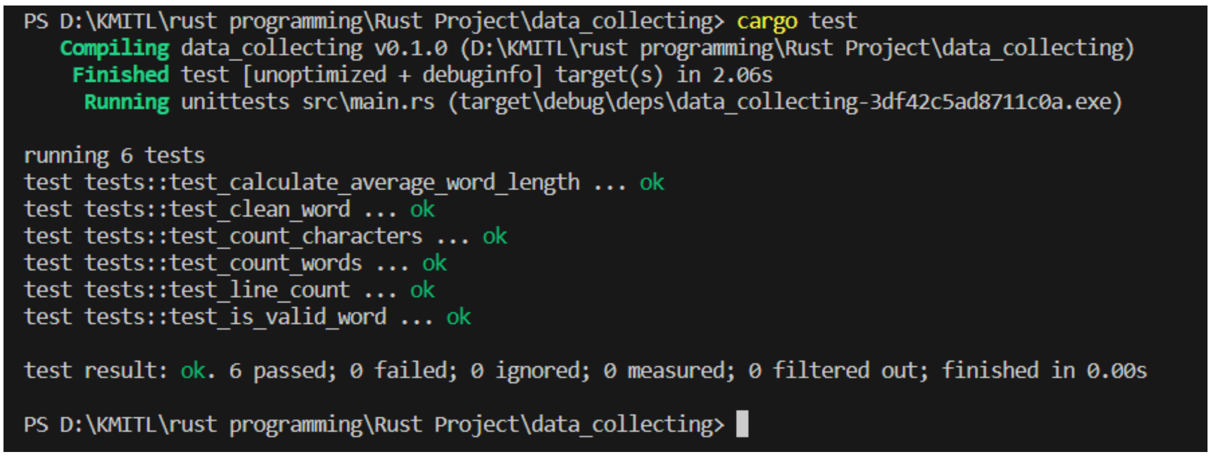


Figure2: Histogram in HTML showing Most Common Words in Folder



	A	B
1	File Name	Line Number
2	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\a.txt	12
3	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\a.txt	13
4	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\a.txt	14
5	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\a.txt	17
6	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	1
7	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	2
8	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	4
9	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	6
10	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	8
11	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	9
12	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	12
13	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	13
14	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	16
15	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	17
16	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	18
17	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	19
18	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	20
19	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	21
20	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	24
21	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	25
22	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	26
23	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	28
24	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	29
25	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	30
26	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	32
27	D:\KMITL\rust programming\Rust Project\data_collecting\Test For Project\e.txt	33
28		

Figure 4. Showing Word Location in csv file



```
PS D:\KMITL\rust programming\Rust Project\data_collecting> cargo test
   Compiling data_collecting v0.1.0 (D:\KMITL\rust programming\Rust Project\data_collecting)
    Finished test [unoptimized + debuginfo] target(s) in 2.06s
     Running unittests src\main.rs (target\debug\deps\data_collecting-3df42c5ad8711c0a.exe)

running 6 tests
test tests::test_calculate_average_word_length ... ok
test tests::test_clean_word ... ok
test tests::test_count_characters ... ok
test tests::test_count_words ... ok
test tests::test_line_count ... ok
test tests::test_is_valid_word ... ok

test result: ok. 6 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

PS D:\KMITL\rust programming\Rust Project\data_collecting>
```

Figure 4. Test Files

▪