CS 202 Group Project
Alaskan Text Surveyor

Bryan Beus
Sarah Carter

February 23, 2020

Source Code Link:
`https://github.com/siddhartha-crypto/cs202_group_project.git`
Git Commit Messages
`https://github.com/siddhartha-crypto/cs202_group_project/commits`
This project took approximately 2.5 hours to complete.

1 Instructions

The goal of this group project is to combine design, pair programming, and iterative software development practices to create a useful application for end users.

2 Pitch

The Alaskan Text Surveyor scans text files and creates a search and classification structure that extends beyond normal "Search" functionality.

The results of the scan display various output that are useful for readers, such as library-science classifications for texts and passages within the texts.

Furthermore, the reader is able to enter search functions that allow the reader to discover associated meanings of large quantities of text.

For example, the project may allow the reader to search for passages that are ”negative” or ”positive” in tone, or other related tasks.

The limitations of the Alaskan Text Surveyor will become more clear as the project develops. Natural language processing is a broad field, and the scope of this project and class only allow for a limited amount of functionality.

3 Project Iteration 1: Design

3.1 Overall Design

The Text Analyzer software performs two main actions.

The software first scans into its memory a few dozen preset text documents that establish a baseline analysis. These documents derive from disparate categories, such as biography, science, geography, and more, to create a broad overview of the field of literary analysis.

The software performs several types of analysis on the texts, including section breakdowns, word count analysis, content type, and more, and provides a final report of the software's findings.

Having completed the preset baseline documents, the Text Analyzer software allows the user to choose another text (or texts) on which the software is to perform a similar analysis.

This analysis performs all the same actions as before, and also creates a comparison between the user's chosen text and the preset texts.

These elements are displayed in a graphic user interface (GUI), allowing the user to visual the content of their indicated document.

3.2 Prior Art

Three main types of inspiration include search engines, machine-learning software for language processing, and the common grammar improvement features found in major word-processing software.

The databases and searching functions of popular search engines, such as Google.com and Amazon.com, feature text analyzers that perform analysis on all documents available to the user through these portals.

Machine-learning software that attempts to auto-complete what a user is currently typing into a search bar is another type of software that performs detailed analysis on text documents. Many types of machine-learning software perform these functions; a popular example is TensorFlow.

The simplest example is the grammar-completion software that exists in common word processors, such as the Microsoft Office Suite's Word processor.

3.3 Technical Design

The Text Analyzer is built in C++. For the text analysis portion of functionality, the software uses the or the GUI, the software uses the ibrary.

The software needs several classes, including `Analysis`, `Display` , and report `Report`. These classes perform analysis on provided text, display analysis findings to the user, and store report data. Each class has its own header and source file. A `Miscellaneous` class can hold any additional functionality. The overall structure of the software is defined in the `main()` function.

Sarah is interested in focusing on the GUI aspect of the software, while Bryan is interested in focusing on the text analysis aspect.

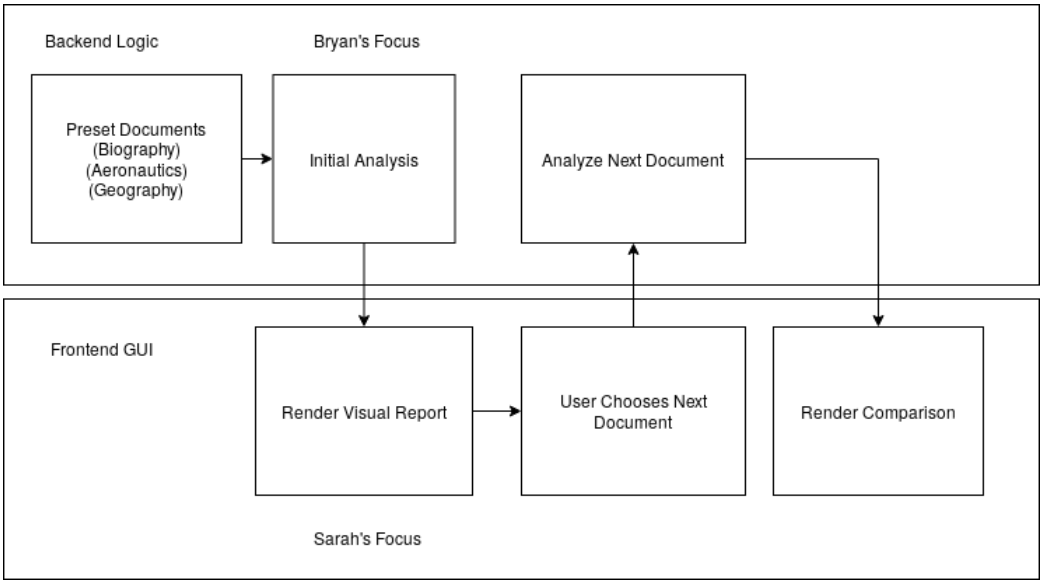

Figure 1: Text Analyzer Flow Chart

3.4 Required Libraries

The GUI aspect of the software relies on the GTK+ library. This library is ideal for the software's purposes as the functionality appears to be user friendly from a developer's point of view. All functions necessary to perform the software are provided in the library.

The text-analysis aspect relies on the MeTA toolkit. This library includes many of the analysis features desired in the software, including topic models, classification algorithms, and more.