Reflection Report - Text-Analyser

Introduction

This project involved developing a console-based Python application designed to perform comprehensive text analysis and document processing. Working independently, I was responsible for planning, designing, and implementing every component of the system. My main goal was to create a tool that could read different file formats, analyze text in multiple ways, and present results in a clear and usable format, all while ensuring that the application remained fully configurable and user-friendly.

Project Overview

The application supports reading both `.txt` and `.py` files, and the list of supported formats is customizable through a `config.json` file. This configuration-based approach allows users to manage formats without modifying the source code, making the application adaptable to different use cases. I also integrated an automatic dependency installation mechanism so that the application can set up its required packages with user consent, eliminating manual installation steps and ensuring smoother execution. To increase usability, I ensured the program runs consistently in both traditional terminal environments and Jupyter Notebook.

Key Features Implemented

1. Configurable File Reading

The application can load and process `.txt` and `.py` files using Python's built-in file handling functionalities. Users can adjust file format support in the `config.json` file, giving control over how the program behaves without modifying the code.

2. Automatic Dependency Installation

A dependency checker (check_dependency function in the util.py file) runs when the application starts. It detects missing Python packages and asks the user for permission to install them. This reduces setup time and ensures that the program can run on systems where required libraries may not be present. But at least python should have been installed.

3. Text Analysis Features

I implemented multiple layers of analysis for input text

- **Basic statistics:** counts of lines, paragraphs, sentences, words, and characters.
- **Word frequency analysis:** identifying repeated words and ranking them by occurrence and also identifying the shortest word and longest word.
- **Sentence analysis:** examining sentence length and structure.
- **Character analysis:** evaluating the distribution of characters, including letters, digits, spaces, and punctuation and also identifying the most common letters.

4. Exporting Results

After performing all analyses, the program can export the results into a document. This feature allows users to save or share their analysis output in a structured format. This is human readable and easy to understand.

5. Spelling Mistake Detection

The application includes a spelling checker that scans the text for potential errors. This adds extra value by helping users identify writing issues beyond simple statistics. The users can see the line number and the incorrect words in the output file. Then they can manually check whether it needs to be corrected or not.

Development Process and Challenges

Developing the entire application alone required careful planning and incremental implementation. One major challenge was deciding how to pass the analyzed values between multiple files in the project. Since different analysis tasks like line analysis, word analysis, and character analysis were handled in separate modules, I had to think carefully about how to share the calculated values across these files.

There were situations where I needed to display results from multiple analyses together, but the related data was not produced in the same module. To solve this, I decided to keep instance references inside the relevant files. By storing the analysis results in instance variables within the class instances, I could access the required statistics from any part of the program without duplicating work or creating tight coupling between modules. This approach made the code more organized and reliable, while still allowing flexibility in where and how results are displayed.

Another significant challenge was implementing the automatic dependency installation feature in a safe and user-friendly way. I had to ensure the program does not interrupt the user unexpectedly and that installations occur only with clear permission. First, I implemented the program in Jupyter Notebook, and it worked perfectly because Jupyter Notebook has the Matplotlib module pre-installed in the environment. However, when I ran the same program in my terminal, I noticed that it failed due to missing dependencies. So I had to think about how to solve this problem, and that's why I developed an automatic dependency resolution mechanism inside the program itself. This way, the user doesn't have to manually install the required dependencies.

Another difficulty was designing the analysis logic to handle various text patterns. Sentence splitting, for example, can be complicated due to punctuation differences, abbreviations, or inconsistent formatting. I tested multiple approaches to ensure accuracy across many text samples.

Learning Outcomes

Completing this project alone helped strengthen my understanding of Python, especially in areas such as file handling, text processing, dynamic imports and configuration management. I gained experience in designing modular code, creating configurable applications, and handling errors gracefully.

I also learned how to create a clear and structured README file so others can understand how to download the project, set it up, and run it. This documentation is especially useful for someone who is starting from zero, as they can simply refer to the README to get everything configured and begin using the application without confusion.

I also improved my ability to think through user workflows. Implementing features like automatic dependency installation and export functionality taught me how to design software that simplifies user experience rather than complicating it.

Working independently gave me confidence in managing all aspects of a project, from architecture planning to debugging and final refinement and testing it. It required discipline, problem-solving, and attention to detail, all of which contributed to my growth as a developer.

Conclusion

In summary, this project allowed me to build a robust and flexible console-based Python application capable of reading documents, analyzing text, identifying spelling mistakes, and exporting results. Developing the project alone deepened my technical knowledge and improved my ability to design practical, user-oriented software solutions. The experience strengthened both my programming skills and my ability to manage a complete software project from start to finish.

Developed by: Kariyawasam Pathiranage Tharanga Mahavila
Project link : https://github.com/TharangaMahavila/Text-Analyser