Activity based

Project Report on

System Programming

Submitted to Vishwakarma University, Pune

Under the Initiative of

Contemporary Curriculum, Pedagogy, and Practice (C2P2)

By

Yashashree Sagar Bedmutha

SRN No : 202100177

Roll No : 02

Div : E

Third Year Engineering

Department of Computer Engineering

Faculty of Science and Technology

Academic Year

2023-2024

1.1. Abstract

The main goal of the project is to develop a flexible lexical analyzer for the English language. Tokenization, classification, design and implementation, and its wide range of applications in language processing are all included in this comprehensive project. English text is effectively broken up into structured tokens by the analyzer, providing an essential basis for more complicated natural language processing tasks. A crucial aspect is its capacity to recognize and classify tokens for a range of applications, including text categorization, sentiment analysis, and information retrieval. The goal of this project is to produce a reliable and versatile tool that can be used in a variety of language processing contexts. The method simplifies the process of obtaining insightful information from unstructured English text data by use of rigorous design and execution procedures.

1.2.Introduction

Within the field of programming and natural language processing, the main objective of this project is to develop a basic, yet flexible, lexical analyzer that is specifically designed for the English language. A thorough examination of the analyzer's lifetime, including crucial stages like design and implementation, tokenization, classification, and its extensive application in the field of language processing, is required for this project.

Careful design considerations are prioritized from the beginning. The project begins with a strategic blueprint that carefully shapes the data structures, algorithms, and architectural elements of the lexical analyzer. This design step converts abstract design principles into tangible code, laying the groundwork for the implementation that follows. The goal is to give the analyzer the capacity to parse input text, recognize and classify tokens quickly, and follow pre-established lexical rules.

To improve analyzability, tokenization—the division of English text into meaningful units like words, numbers, punctuation, and operators—is a crucial component. Classification and recognition play equally important roles in allowing the analyzer to recognize and classify different kinds of tokens according to pre-established guidelines. Due to its adaptability, the tool can be easily integrated into a wide range of language processing applications, such as information retrieval, text categorization, and sentiment analysis.

The analyzer serves as the cornerstone for more complex natural language processing activities as well. It supports tasks like named entity recognition, sentiment analysis, and part-of-

speech tagging, which enable the extraction of insightful information from unstructured English text data.

1.3. Literature Summary

The "Lexical Analyzer for English Language" program's foundation is made up of logical elements and libraries that make text analysis and classification easier. The main libraries utilized are Swing for graphical user interface (GUI) creation and the Stanford NLP library for linguistic analysis. An outline of the GUI's libraries, logic, and alignment is provided below:

- **Stanford NLP Library:** The software uses the Stanford NLP library, which provides necessary tools for part-of-speech tagging, sentence splitting, and tokenization, for natural language processing. This library is essential to the input text's lexical analysis.

- **Swing Library:** Swing is used to create a graphical user interface. It is a component of the Java standard library. It enables the development of text fields for input and output, execute and clear buttons, and an interactive user interface in general.
- **GUI alignment:** To guarantee usability, the GUI's components are carefully positioned. Text input and result display are supported in the input and output text boxes. The user can interact with the analysis by using the "Run" and "Clear" buttons to execute the analysis and clear the text boxes.

1.4. Methodology

Design Components of the GUI

- **Text Input and Output Boxes**: The program's functionality depends on the integration of text input and output boxes. The user can submit text for examination through the input text section. It provides a recognizable and easy-to-use text entry interface that can effectively handle big text volumes. Text areas are used because they are flexible and can accommodate text with different lengths and levels of complexity. It guarantees that text input is simple for users, which makes the program useful and approachable.
- **Run and Clear Buttons:** Having "Run" and "Clear" buttons allows the user to engage with the program's logic without any interruptions. The analysing process is sparked by the "Run" button. After users enter text and select "Run," the application reads the text, tokenizes it, and groups tokens according to different categories. Users can acquire results in an understandable and practical manner thanks to the interaction between the GUI component and the software logic. Users can quickly reset the input and output sections with the "Clear" button, which is a useful tool that makes it easy to start over with fresh material or additional analysis. The user control and usability of the program are improved by this connection.
- **Font Selections:** A purposeful design decision was made to use a monospaced font in both the input and output sections. Because they retain regular character width and consistent alignment, monospaced fonts are particularly useful for displaying text and code. It guarantees that results and tokens are displayed in an orderly and understandable manner within the program. The selection of fonts improves the GUI's aesthetic appeal and makes it seem more polished and user-friendly.

Pseudo-Code For Gui

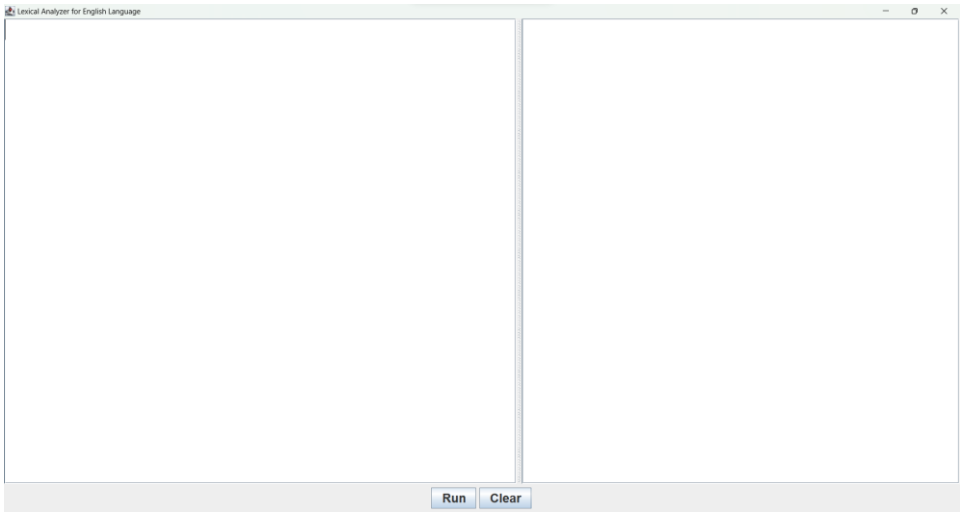
1. Initialize the GUI window
2. Set the window title to "Lexical Analyzer for English Language"
3. Set the window size to 1550 pixels in width and 820 pixels in height
4. Specify that the window should close the program on exit
5. Create an input text area
    - Set its number of rows to 20
    - Set its number of columns to 40
    - Enable line wrapping for text
    - Enable word wrapping for text
    - Set the font to a monospaced font with a bold style (e.g., "Monospaced", Font.BOLD, 24)
6. Create an output text area
    - Set its number of rows to 20
    - Set its number of columns to 40
    - Make it non-editable
    - Set the font to a monospaced font with a bold style (e.g., "Monospaced", Font.BOLD, 18)
7. Create a "Run" button
    - Set the button label to "Run"
    - Set the font for the button text (e.g., "Arial", Font.BOLD, 20)
    - Define an action listener to handle the "Run" button click event
8. Create a "Clear" button
    - Set the button label to "Clear"
    - Set the font for the button text (e.g., "Arial", Font.BOLD, 20)
    - Define an action listener to handle the "Clear" button click event
9. Create a layout for the GUI (e.g., BorderLayout)
    - Add the input text area to the layout's center
    - Add the output text area to the layout's center
    - Add the "Run" and "Clear" buttons to the layout's south
10. Configure the GUI window to display the layout
11. Make the GUI window visible

1.5 Result

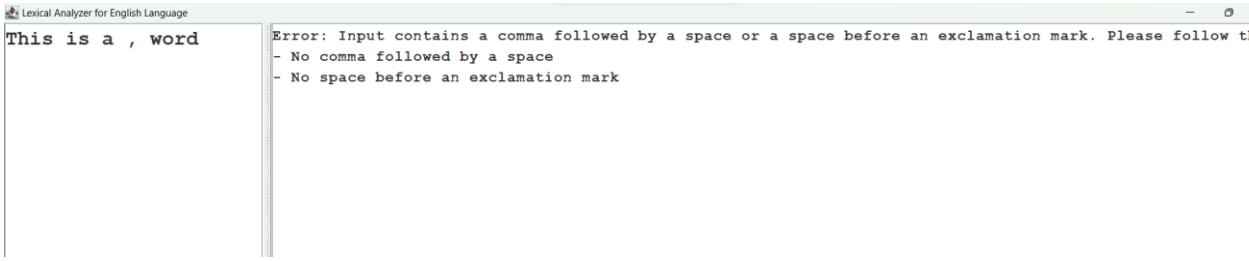
A powerful and intuitive tool for analyzing and classifying English text, "Lexical Analyzer for English Language" gives users comprehensive insights into the structure of the text they enter. The outcome includes multiple important elements:

- **Text Analysis and Categorization:** The application analyzes the text in great detail after receiving input from the user. Each token is methodically categorized according to its part-of-speech characteristics, which can include operators, special characters, numbers, punctuation, adjectives, adverbs, verbs, and adjectives. The analytical powers of the application include comprehending the subtleties of English language use in the given text.
- **User Interaction:** The application has a graphical user interface (GUI) that makes interacting with it easy. The "Run" and "Clear" buttons allow users to interact with the software by simply entering text into the corresponding text input field. The "Run" button is the entry point that allows you to start the analysis process, giving you control and accessibility over it. Users can reset the input and output text fields using the "Clear" button, making room for new text input or additional analysis.
- **Output Presentation:** The output text box is where the application displays the results of the analysis. For every category, the analysis comprises a count and matching lists of tokens. The distribution of linguistic elements inside the text is immediately apparent to users. Users will be able to understand the relevance of different token categories quickly because to the presentation's clarity and informativeness.
- **Token categories and counts** are displayed in the result, each with a list of tokens and a count. Nouns, verbs, adjectives, adverbs, numerals, punctuation, operators, and special characters are some of these categories. Whether the text is a code, story, or any other type of English writing, this thorough analysis gives users a profound grasp of the language elements present.
- **Analytical Utility:** The application functions as a useful analytical tool in addition to offering a list of tokens. It helps researchers, developers, linguists, and everyone else with an interest in text analysis. Users can evaluate language usage, derive insights, and make judgments based on the structure of the text by utilizing the application.
- **User-Centered Design:** To ensure that users can work with the application comfortably and effectively, text input and output sections have been thoughtfully built within the user interface. The monospaced font selection improves reading and alignment, adding to the GUI's polished appearance.

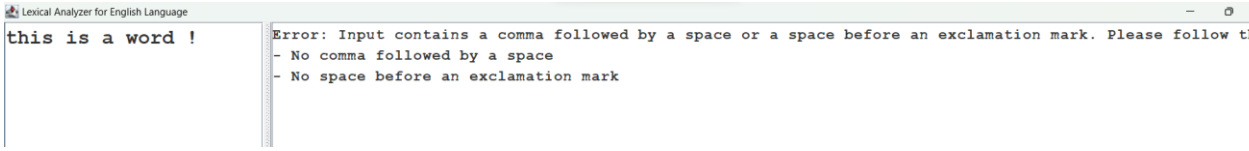
Output:



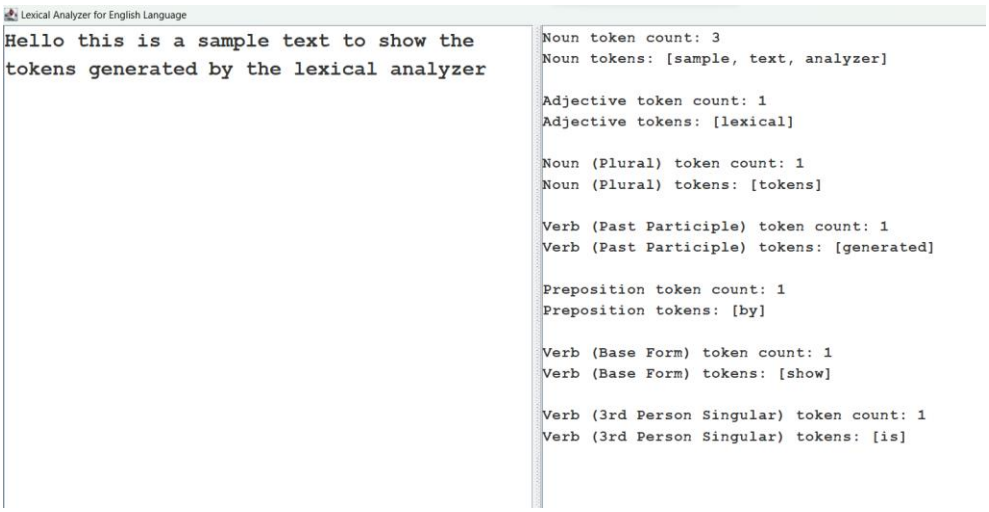
- Error Handling :
  - Comma



  - Exclamation Mark



- Proper Output :



1.6. References

- Stanford NLP library's official documentation
- Hersh, W. R., Campbell, E. M., & Malveau, S. E. (Year). *Assessing the feasibility of large-scale natural language processing in a corpus of ordinary medical records: a lexical analysis.*
- Lee, S., Binkley, D., Gold, N., Islam, S., Krinke, J., & Yoo, S. (Year). *Evaluating lexical approximation of program dependence.*
- Tjabaka, L. H. (Year). *The rise of lexical subjects in English covert infinitives.*
- Swing in Java
    - https://www.javatpoint.com/java-swing
    - https://www.geeksforgeeks.org/introduction-to-java-swing/
    - https://docs.oracle.com/javase/tutorial/uiswing/start/index.html