



Cracow University of Technology
Department of Computer Sciences



Natural Language Processing – Erasmus
Ac. Year 2023/2024

Text Summarizer Application

Authors:

Karyna Ouahrani

Yuliia Boiko

1. Abstract

This project involves the development of a Text Summarizer Application using Python's tkinter library for the graphical user interface (GUI) and Natural Language Processing (NLP) techniques from the nltk library for text summarization. The application allows users to input a block of text and generate a concise summary, enhancing readability and understanding. This report provides a comprehensive overview of the project's objectives, scope, methodology, theoretical background, practical implementation, results, and conclusions.

2. Introduction

2.1 Aim

There has been a massive increase in text data from numerous sources in recent years. This text is an invaluable source of knowledge and information that must be carefully summarized in order to be put to good use. The aim of this project is to develop a user-friendly software application that can automatically generate a summary of a given text. This tool is designed to help users quickly understand the main points of lengthy documents, articles, or any substantial text, thus improving their productivity and comprehension.

2.2 Scope

The scope of this project encompasses the following aspects:

- **Development of a Graphical User Interface (GUI):** Using tkinter, the application includes interactive components like Start, Exit, and Restart buttons to enhance user interaction.
- **Text Summarization Functionality:** Leveraging the nltk library, the application processes and summarizes input text efficiently.
- **User Experience Design:** Incorporating styling elements to make the application visually appealing and easy to navigate.
- **Instructional Feedback:** Providing clear instructions and error messages to guide users through the summarization process.

2.3 Methodology

Tools and Libraries Used:

- **Python:** The primary programming language for development the application
- **tkinter:** A standard Python library used for creating the GUI. [4]
- **nltk (Natural Language Toolkit):** Utilized for various NLP tasks including tokenization, stopwords removal, and sentence scoring. [5]
- **Heapq:** A Python library used for selecting the most important sentences based on their scores.[6]

Development Process:

1. **Setting Up the Environment:** Installed necessary libraries and downloaded required NLTK datasets.
2. **Designing the GUI:** Created the main window with interactive buttons and a text input area.
3. **Implementing the Text Summarization Function:** Developed a function that summarizes text using NLP techniques.
4. **Integrating Components:** Ensured seamless interaction between the GUI elements and the summarization function.
5. **Adding Styling:** Enhanced the visual appeal of the application through custom fonts, colors, and layout adjustments.

Theoretical Part

3. Text Summarization Theory

The text summarization is a process to create an easily understood and readable summary while maintaining the main ideas and overall meaning of the text. Many techniques for automatic text summarization have been developed recently and are being used effectively across a range of domains. As a preview of the documents, search engines provide snippets. Additional examples include information extraction techniques or news websites that provide summarized summaries of news items to make browsing easier. [2] [3]

Types of Text Summarization:

In general, automatic summarization can be achieved by two distinct methods: abstraction and extraction. Extractive summarization techniques rely only on sentence extraction from the source text; they locate key passages in the text and produce them verbatim. In contrast, the goal of abstractive summarization techniques is to generate significant information in an original way. To put it another way, they analyze and understand the text using sophisticated natural language processing algorithms to produce a new, condensed version that only includes the most important details from the original. The majority of current summarizing research has focused on extractive summarization, even though human-generated summaries are typically not exhaustive. [2]

When compared to automatic abstractive summaries, purely extractive summaries frequently produce superior outcomes. This is due to the fact that data-

driven procedures, such phrase extraction, are often easier to handle than abstractive summarizing methods, which deal with issues like semantic representation, inference, and natural language production. Currently, there is not a summarization system that is entirely abstractive. To create the text's abstract, existing abstractive summarizers frequently depend on an extractive preprocessing step.[3]

Techniques Used:

For this project, we focused on extractive summarization, which includes the following steps:

- **Tokenization:** The process of breaking a string into distinguishable linguistic units that make up a piece of language data. [1]
- **Stopword Removal:** Common words that add little meaning (e.g., 'and', 'the') are removed to focus on significant terms. Stopwords are words that appear frequently but do not contribute to the overall meaning of the text.[1]
- **Word Frequency Calculation:** Counting the occurrences of each word to determine its importance within the text. This helps in identifying key terms that are central to the text's theme.[1]
- **Sentence Scoring:** Assigning scores to sentences based on the frequency of important words they contain. Sentences that contain more high-frequency words are considered more important.[1]
- **Summary Generation:** Selecting and concatenating the top-scoring sentences to form the summary. This step involves choosing a subset of sentences that best represent the overall content of the original text.[1]

Practical Part

4. Solution and Implementation

4.1 Libraries and Data Setup:

- The primary programming language of text summarizer application is Python.
- Importing essential libraries: nltk for natural language processing, heapq for sentence scoring, and tkinter for the GUI.

4.2 Graphical User Interface (GUI)

The GUI was designed using tkinter and includes the following components:

- **Main Frame:** The initial interface that contains an introductory message explaining the purpose of the application and two buttons: Start and Exit.
- **Text Input Frame:** Appears when the Start button is pressed, containing a text box for input, a button to summarize the text, and a Restart button to clear the input and start over.

4.3 Text Summarization Function

The text summarization function processes the input text as follows:

- **Tokenization:** The input text is split into sentences and words.
- **Stopword Removal:** Stopwords and punctuation are removed from the word list.
- **Word Frequency Calculation:** The frequencies of the remaining words are calculated.
- **Sentence Scoring:** Sentences are scored based on the frequencies of the words they contain.

- **Summary Generation:** The top-scoring sentences are selected to form the summary.

```
def text_summarizer(text, num_sentences=3):
    # Tokenize the text into sentences
    sentences = sent_tokenize(text)

    # Tokenize the text into words and convert to Lower case
    words = word_tokenize(text.lower())

    # Remove stopwords and punctuation from the word list
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words and word not in string.punctuation]

    # Calculate word frequencies
    word_freq = {}
    for word in words:
        if word not in word_freq:
            word_freq[word] = 1
        else:
            word_freq[word] += 1

    # Calculate sentence scores based on word frequencies
    sentence_scores = {}
    for sent in sentences:
        for word in word_tokenize(sent.lower()):
            if word in word_freq:
                if sent not in sentence_scores:
                    sentence_scores[sent] = word_freq[word]
                else:
                    sentence_scores[sent] += word_freq[word]

    # Select the top sentences based on their scores
    summary_sentences = heapq.nlargest(num_sentences, sentence_scores, key=sentence_scores.get)

    # Join the top sentences to form the summary
    summary = ' '.join(summary_sentences)
    return summary
```

Fig1. Part of the code of text summarizer function

4.4 Integration and Styling

The components were integrated to ensure seamless interaction:

- **Start Button:** Switches from the mainframe to the text input frame.
- **Summarize Button:** Calls the summarization function and displays the result below the input text box.
- **Restart Button:** Clears the input text and summary, allowing the user to start the process again.

Styling was applied to enhance the user experience, including:

- Custom fonts for better readability.
- Background colors to distinguish different sections.
- Button styling to make interactive elements more intuitive.

4.5 Results

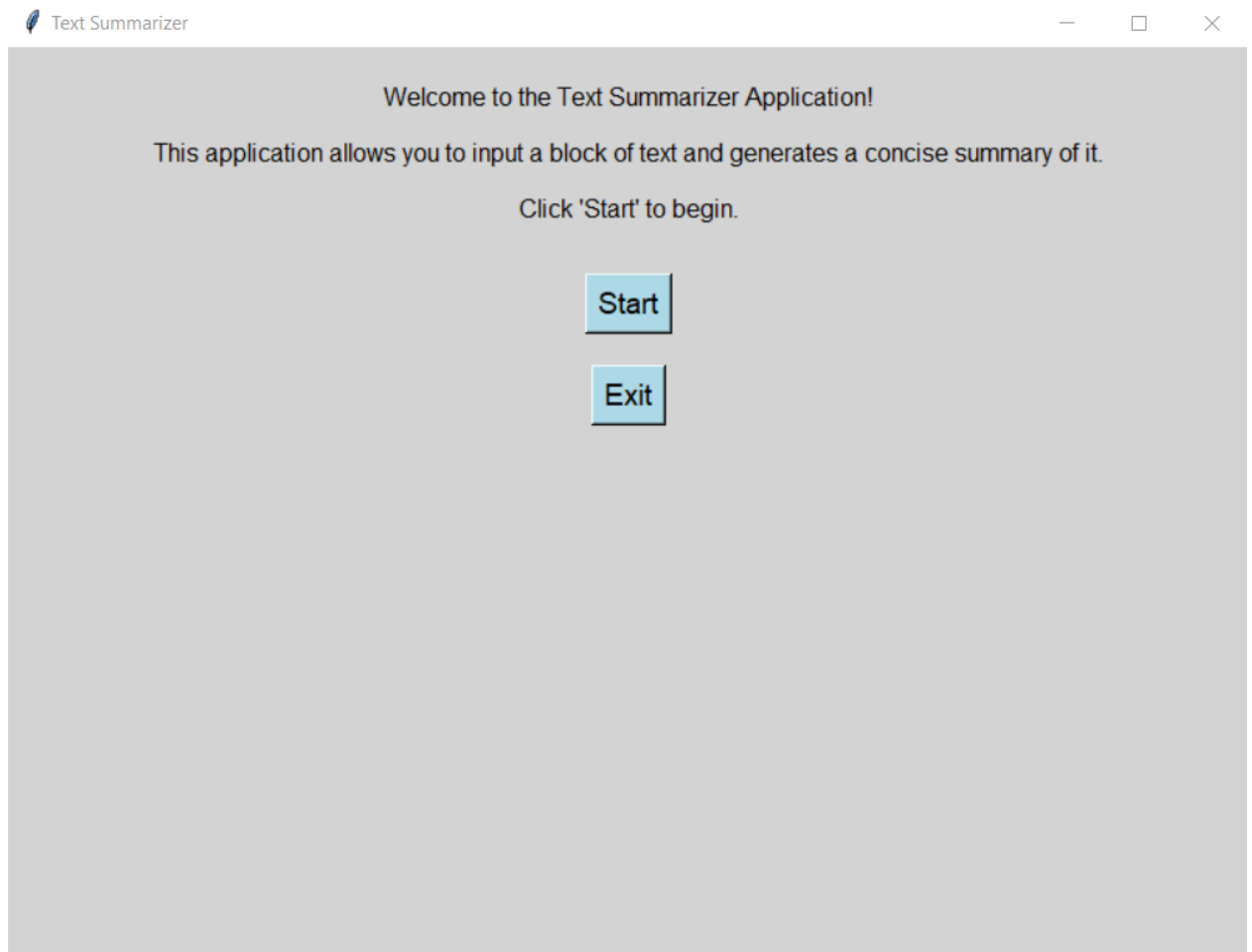


Fig2. Main Frame of the application

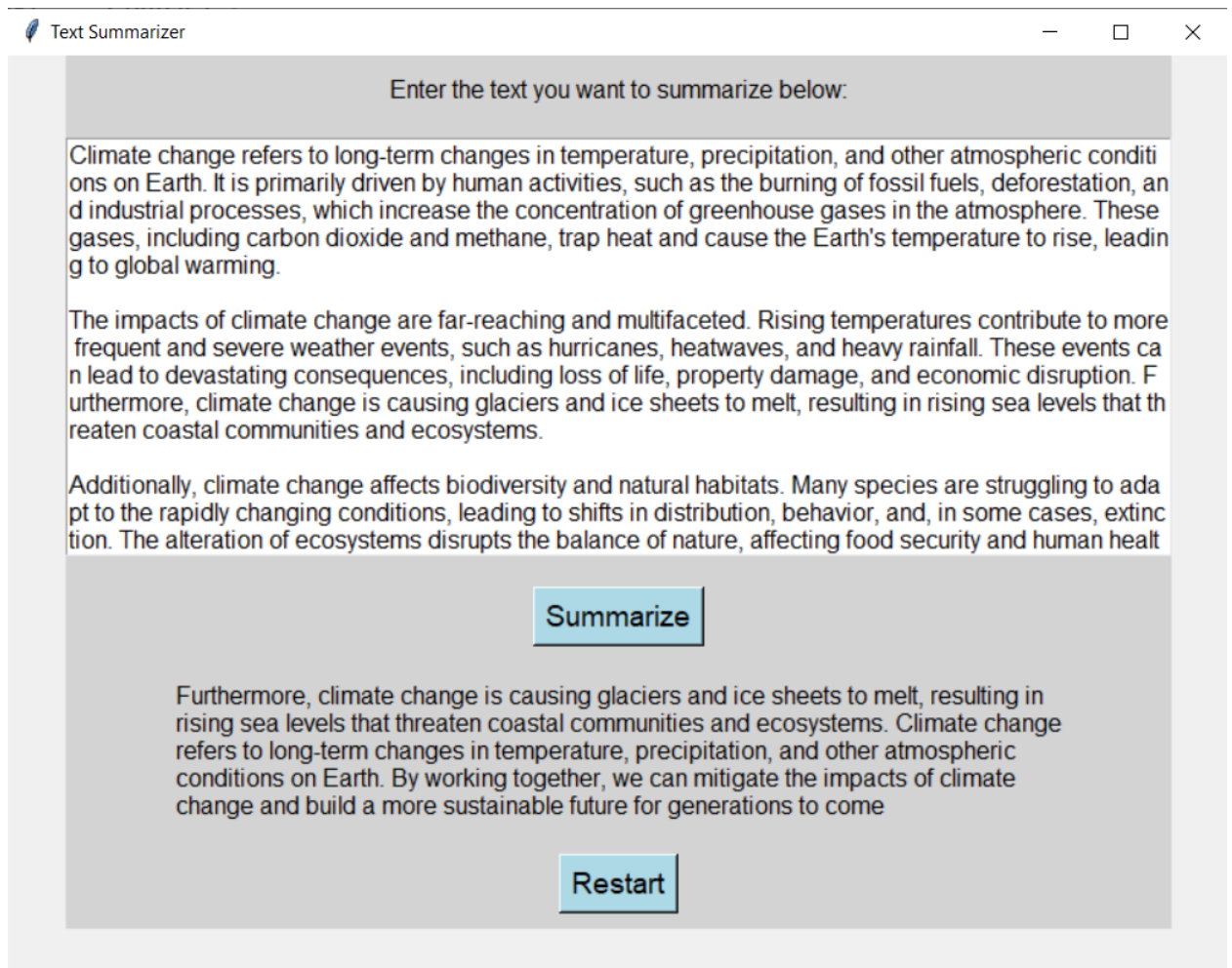


Fig3. The application for text summarizer in use

The application successfully summarizes input text into concise versions, as demonstrated by various test inputs. The functionality satisfies the project goals, and the GUI offers a user-friendly interface. It is practical and effective since users can provide material, create summaries, and resume the process with efficiency.

Conclusion

The project's objectives have all been successfully accomplished. It was effective in creating an application that is convenient to use for text summarization, using nltk for text processing and tkinter for the graphical user interface (GUI). The resulting program gives users brief overviews of the input texts and has excellent summarizing functionality. The project's requirements and expectations have all been met by the seamless and user-friendly text summarizing tool that has been produced via the integration of libraries and technologies.

This application serves as a valuable tool for people and professionals across various domains. For people in a variety of fields, both professionals as well as people can benefit greatly from this application. It can be used by students to quickly summarize extensive academic books for homework or research. It can be used by experts in disciplines like journalism, research, and content development to highlight important ideas or points from papers, reports, or articles. The tool can also help people with busy schedules by effectively processing information from emails, blogs, and news items. All things considered, the tool provides a practical and time-saving option for anyone who needs to reduce lengthy texts into readable summaries.

Bibliography

1. Bird S., Klein E., Loper E. Natural language processing with Python: analyzing text with the natural language toolkit. – " O'Reilly Media, Inc.", 2009.
2. Allahyari M. et al. Text summarization techniques: a brief survey //arXiv preprint arXiv:1707.02268. – 2017.
3. Gambhir M., Gupta V. Recent automatic text summarization techniques: a survey //Artificial Intelligence Review. – 2017. – T. 47. – №. 1. – C. 1-66.
4. Lundh F. An introduction to tkinter. – 1999.
5. Bird S. NLTK: the natural language toolkit //Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions. – 2006. – C. 69-72.
6. Brodal G. S. Priority queues with decreasing keys //Theoretical Computer Science. – 2024. – T. 1000. – C. 114563.