**Documentation**

**Text Summarization App**

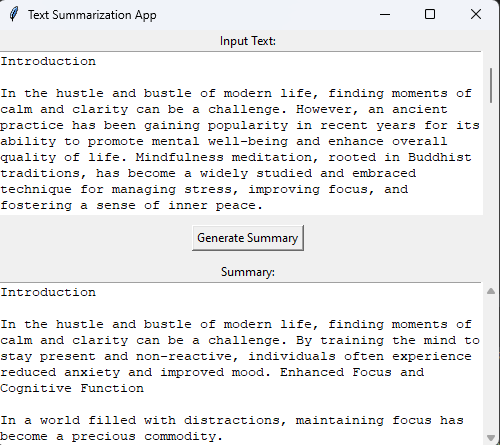


**Submitted by:**

Carl Michael C. De Taza

Sue C. Sinangote

David Frank Edison Regodos

****



**Prompt:** Introduction

The Text Summarization App is a user-friendly desktop application developed using Python and the Tkinter GUI library. Designed to simplify the process of summarizing textual content, the application leverages the power of the Hugging Face `transformers` library, specifically the `Summarizer` module. This documentation provides an overview of the application's functionalities, the underlying summarization process, and its practical applications.

**Processing Workflow**

The core of the Text Summarization App lies in its ability to generate abstractive summaries of input text. The summarization process is facilitated by the `Summarizer` module, which employs a pre-trained BERT-based model. The steps involved include:

* **Input Text Extraction:** The user inputs text into the application using the provided text area.
* **Summarization with BERT:** The `Summarizer` module takes the input text and utilizes the BERT-based model to generate an abstractive summary.
* **Displaying the Summary:** The generated summary is then displayed in the output text area, providing users with a concise and coherent representation of the input content.

**Usage of the Application**

The Text Summarization App serves as a valuable tool in various scenarios, providing users with an efficient means of extracting key information from lengthy texts.

* **Document Summarization:** Professionals dealing with extensive documents can quickly obtain summarized versions for efficient review and decision-making.
* **Content Consumption:** Readers can use the app to generate summaries of articles or blog posts, facilitating a quicker understanding of the content.



**Prompt:** Advantage and Disadvantage

**Advantages of the Text Summarization App:**

* **Efficient Text Summarization:** The Text Summarization App leverages a pre-trained BERT-based model to efficiently generate abstractive summaries. This enables users to distill essential information from large texts quickly, enhancing productivity in tasks such as document review or content consumption.

**Disadvantages of the Text Summarization App:**

* **Limited Customization Options:** The application currently provides a predefined summarization model with limited customization options. Users seeking fine-tuning or specific model adjustments may find the app restrictive in terms of configurability.



**Prompt:** Recommendations

**User-Defined Summarization Parameters:** Enhance the application by introducing options for users to customize summarization parameters. This could include adjusting summarization length, controlling the level of abstraction, or selecting different pre-trained models based on specific use cases.

**Include Summarization Tips:** Integrate a tooltip or information section within the app to provide users with tips on optimizing summarization results. This could include guidance on input text formatting, handling specific types of content, or maximizing the effectiveness of the summarization process.

**Integration with External Tools:** Explore possibilities for integrating the application with external tools or APIs that offer specialized summarization capabilities. This could broaden the range of summarization techniques available to users.



**Prompt:** System Requirements

**Hardware Requirements:**

* No specific hardware requirements beyond a standard computer with a display.

**Software Requirements:**

|  |  |
| --- | --- |
| **Name** | **Descriptions** |
| **Python 3.x** | Python 3.x, the latest version, brings key improvements like Unicode support, enhanced syntax, and asyncio. It's actively maintained, secure, and backward incompatible with Python 2.x. Widely used in web development, AI, and more, Python 3.x is the recommended choice for new projects. |
| **Tkinter** | Tkinter is the standard GUI (Graphical User Interface) toolkit that comes with Python. It provides a set of tools and widgets for creating graphical user interfaces and is based on the Tk GUI toolkit. Tkinter allows developers to create windows, dialogs, buttons, textboxes, and other GUI elements for their Python applications. |
| `**summarizer` library** | is a Python library that provides a simple interface for text summarization using pre-trained BERT (Bidirectional Encoder Representations from Transformers) models. The library is built on top of the Hugging Face transformers library, which is a popular open-source library for natural language processing (NLP) tasks. |



**Prompt:** Installation

**Python**

Ensure that Python 3.x is installed on your system. You can download it from the official Python website: [Python Downloads](<https://www.python.org/downloads/> )

**Create a virtual environment(optional)**

* **Open Command Prompt:**

Open the Command Prompt on your Windows desktop.

* **Navigate to the Desired Directory:**

Use the **cd** command to navigate to the directory where you want to create the virtual environment. For example:



* **Create Virtual Environment:**

Run the following command to create a virtual environment named **env**:

****

* **Activate Virtual Environment:**

To activate the virtual environment, run the appropriate activation script. In the Command Prompt:****

**Clone Repository from Github**

* **Get the Repository URL:**

On the GitHub repository page, click on the "Code" button. Make sure to select the "HTTPS" option. Copy the repository URL provided: <https://github.com/Chael07/drill3_Text_Summarization-.git>

* **Open Terminal Command Prompt (Windows):**

Open a terminal or command prompt on your local machine.

* **Navigate to the Directory Where You Want to Clone the Repository:**

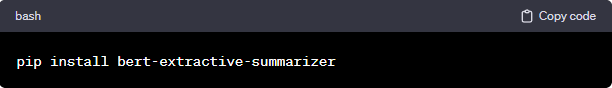
Use the cd command to navigate to the directory where you want to store the cloned repository. For example:****

* **Clone the Repository:**

Use the git clone command followed by the repository URL. Replace <https://github.com/Chael07/drill3_Text_Summarization-.git> with the URL you copied in step 1.

**Required Python Packages**

Install the required Python packages in same directory using the following command:

****



**Prompt:** Application Overview

The Text Summarization App provides a user-friendly interface for generating abstractive summaries of input text using a pre-trained BERT-based summarization model.

**Features**

* **Input Text Area:** Allows users to input text for summarization.
* **Generate Summary Button:** Triggers the summarization process.
* **Output Text Area:** Displays the generated summary.

**Code Structure**

The code is organized into a single Python script (`nlp.py`). Here's an overview of the code structure:

**Imported Libraries:**

* `tkinter`: Tkinter library for GUI.
* `scrolledtext` from Tkinter: Provides a scrolled text widget.
* `Summarizer` from the `summarizer` library: Implements the BERT-based summarization model.

**Functions:**

* `generate\_summary` Function:
* Extracts input text from the input area.
* Utilizes the `Summarizer` model to generate a summary.
* Displays the generated summary in the output area.

**Main Application Window:**

* Sets up the Tkinter main application window with the title "Text Summarization App."

**GUI Components:**

* Input Text Area (`text\_input`):
* Accepts user input for summarization.

**Generate Button (`generate\_button`):**

* Triggers the summarization process.

**Output Text Area (`text\_output`):**

* Displays the generated summary.



**Prompt:** Conclusion

The Text Summarization App offers a streamlined approach to text summarization, enabling users to obtain concise and informative summaries with ease. While it excels in efficiency, introducing user-defined parameters, enhancing user guidance, and providing options for model selection would further elevate its usability and cater to a broader user base. By addressing these recommendations, the application can evolve into a more versatile and user-centric tool for various summarization needs.



**Prompt:** Source Code Used in the Project

import tkinter as tk

from tkinter import scrolledtext

from summarizer import Summarizer

def generate\_summary():

    input\_text = text\_input.get("1.0", "end-1c")

    if input\_text:

        model = Summarizer()

        summary = model(input\_text)

        text\_output.config(state="normal")

        text\_output.delete("1.0", tk.END)

        text\_output.insert(tk.END, summary)

        text\_output.config(state="disabled")

# Create the main application window

root = tk.Tk()

root.title("Text Summarization App")

# Create and configure the input text area

text\_input\_label = tk.Label(root, text="Input Text:")

text\_input\_label.pack()

text\_input = scrolledtext.ScrolledText(root, wrap=tk.WORD, width=60, height=10)

text\_input.pack()

# Create and configure the generate button

generate\_button = tk.Button(root, text="Generate Summary", command=generate\_summary)

generate\_button.pack(pady=10)

# Create and configure the output text area

text\_output\_label = tk.Label(root, text="Summary:")

text\_output\_label.pack()

text\_output = scrolledtext.ScrolledText(root, wrap=tk.WORD, width=60, height=10, state="disabled")

text\_output.pack()

# Run the application

root.mainloop()

**END**