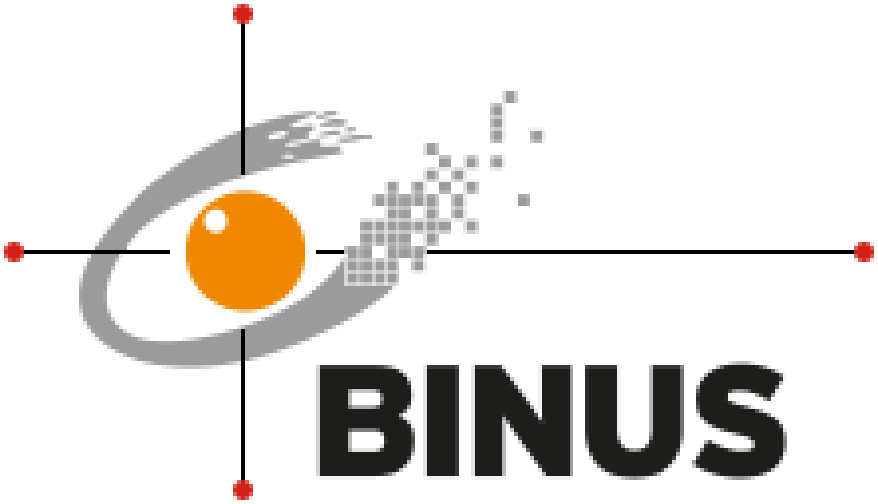


BINUS
UNIVERSITY
INTERNATIONAL

News Article Summarizer

Troy Poetra Prajoga|ID: 2702291910 | Algorithm and Programming

Table of Contents:

Project Specifications

Project Overview

The Program created is a news article summarizer. This program is designed to allow users to quickly retrieve information on a news article. The user will be able to obtain the title, author, publication date, summary and sentiment. The program also allows users to save the information on the news article.

Objectives

The main objectives of the projects are as follows:

Design a user-friendly graphical user interface(GUI) for the article summarizer.

Retrieve the title, author, publication date, summary, and sentiment of news articles.

Save the data so that it can be retrieved again later.

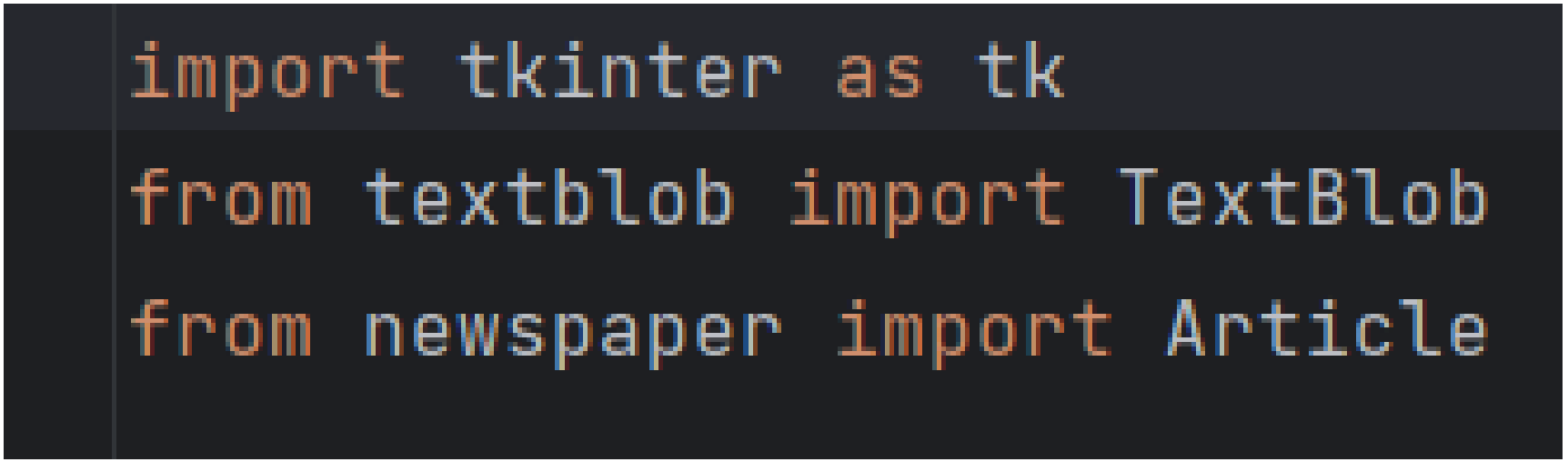
Check the saved data of news articles.

Solution Design

In this program, there are two files which are main.py and file.txt. The main.py file contains three classes which are SummarizeGUI, SavedGUI, and HomeGUI. These GUIs were made using tkinter. The text file that is used is file.txt.

Imports

I used three imports. The first one is tkinter which is used for creating the GUIs. The second import I used is textblob which is used for the sentiment analysis portion of the summarizer. The third import I used is the article import which is used for extracting the information about the articles.



widgets function

Here is the function for creating the general widgets of the GUI that will be used in the SummarizeGUI and SavedGUI classes. In this function I create a label, as well as a display label for displaying information for the title, author, publication date, summary, and sentiment of the news article that will be summarized. I also made a home button with the command to destroy the current GUI and reopen the home GUI.

```python
# Create the common widgets for SummarizeGUI and SavedGUI
2 usages
def widgets(self):
    # make title label
    title_label = tk.Label(self.root, text="Title")
    title_label.pack()

    # make title box and user cant change
    self.title = tk.Text(self.root, height=1, width=140)
    self.title.config(state='disabled', bg='#dddddd')
    self.title.pack()

    # make author label
    author_label = tk.Label(self.root, text="Author")
    author_label.pack()

    # make author box and user cant change
    self.author = tk.Text(self.root, height=1, width=140)
    self.author.config(state='disabled', bg='#dddddd')
    self.author.pack()

    # make publication label
    publication_label = tk.Label(self.root, text="Publishing Date")
    publication_label.pack()

    # make publication box and user cant change
    self.publication = tk.Text(self.root, height=1, width=140)
    self.publication.config(state='disabled', bg='#dddddd')
    self.publication.pack()
```

```python
# make summary label
summary_label = tk.Label(self.root, text="Summary")
summary_label.pack()

# make summary box and user cant change
self.summary = tk.Text(self.root, height=20, width=140)
self.summary.config(state='disabled', bg='#dddddd')
self.summary.pack()

# make sentiment label
sentiment_label = tk.Label(self.root, text="Sentiment Analysis")
sentiment_label.pack()

# make sentiment box and user cant change
self.sentiment = tk.Text(self.root, height=1, width=140)
self.sentiment.config(state='disabled', bg='#dddddd')
self.sentiment.pack()

# make home button
home_button = tk.Button(self.root, text="Go to Home", command=self.destroy)
home_button.pack(side=tk.BOTTOM, pady=5)
```

change_contents function

This function is used to change the contents of the display labels when the AI does the summarization. It first changes the states of the labels to be normal which allows the contents to be changed. Then, after the state is changed, it deletes the current content and inserts the new summarized content. At the end, it disables the states of the display labels so that the content cannot be changed.

```
# Function for changing the contents of SummarizeGUI and SavedGUI
2 usages
def change_contents(self, title, author, publication, summary, sentiment):
    # Allows content to be changed
    self.title.config(state='normal')
    self.author.config(state='normal')
    self.publication.config(state='normal')
    self.summary.config(state='normal')
    self.sentiment.config(state='normal')

    # Changing the contents
    self.title.delete('1.0', 'end')
    self.title.insert('1.0', title)

    self.author.delete('1.0', 'end')
    self.author.insert('1.0', author)

    self.publication.delete('1.0', 'end')
    self.publication.insert('1.0', publication)

    self.summary.delete('1.0', 'end')
    self.summary.insert('1.0', summary)

    self.sentiment.delete('1.0', 'end')
    self.sentiment.insert('1.0', sentiment)
```

```
# Stopping the contents from being changed
self.title.config(state='disabled')
self.author.config(state='disabled')
self.publication.config(state='disabled')
self.summary.config(state='disabled')
self.sentiment.config(state='disabled')
```

SummarizeGUI Class

This is the SummarizeGUI class and it is used for creating the GUI for summarizing news articles. First I initialized the window and the variables of the summary which include title, author, publication date, summary, sentiment, and url text. I also initialize a dictionary that will be used for storing information in a text file. Finally, I also call the create_widgets function for creating all the widgets necessary for the GUI.

```python
# Class for creating a GUI for summarizing articles
1 usage
class SummarizeGUI:
    # initializing variables
    def __init__(self, home_instance):
        # create window, title, and instance of HomeGUI(used for switching between windows).
        self.home_instance = home_instance
        self.root = tk.Tk()
        self.root.title("News Summarizer")

        # initializing all variables in the summary
        self.title = None
        self.author = None
        self.publication = None
        self.summary = None
        self.sentiment = None
        self.url_text = None

        # initializing dictionary for text file
        self.article_info = {
            "Title": "",
            "Author": "",
            "Publish Date": "",
            "Summary": "",
            "Sentiment": ""
        }

        # calling the create_widgets() function to build the GUI.
        self.create_widgets()
```

display_article function

This function analyzes the inputted article text and checks its sentiment using the TextBlob function. Then it uses the change_contents function to change the contents of what is displayed in the GUI. Finally it updates the dictionary with the new summarized content in case the user wants to save the summarized information.

```
# This function is for displaying the article summary
1 usage
def display_article(self, article):
    # Analyze the article then change the contents
    analysis = TextBlob(article.text)
    change_contents(self, article.title, article.authors, article.publish_date, article.summary,
                    sentiment: f'Polarity: {analysis.polarity} Sentiment: {"positive" if analysis.polarity > 0 else "negative" if analysis.polarity < 0 else "neutral"}')

    # This updates the dictionary with the summary
    self.article_info["Title"] = article.title
    self.article_info["Author"] = article.authors
    self.article_info["Publish Date"] = str(article.publish_date)
    self.article_info["Summary"] = article.summary
    self.article_info["Sentiment"] = (f'Polarity: '
                                      f'{analysis.polarity} Sentiment: {"positive" if analysis.polarity > 0 else "negative" if analysis.polarity < 0 else "neutral"}')
```

summarize function

I created a summarize function which does the actual summarizing part of the article. This function takes in the url that the user inputs, downloads the information of the article, then parses the data (separating the data into the parts that it needs like title, author, etc.), and finally uses natural language processing for the summarization. At the end of this function, it calls the display_article function to display the contents of the article.

```
# This summarizes the article
1 usage
def summarize(self):
    url = self.url_text.get('1.0', "end").strip()
    article = Article(url)
    article.download()
    article.parse()
    article.nlp()

    # Call the function to display the article summary
    self.display_article(article)
```

save_summary function

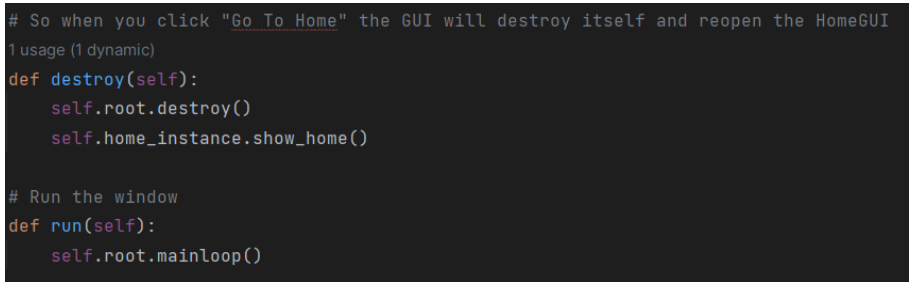
This function creates a dictionary that will be easy to write into the text file. This is done by having a key that states what the data is, and a value that is retrieved from the article_info dictionary. Then, the keys and values are written into a text file called file.txt so that the data can be retrieved later.

```
# This saves the summary into a text file using the dictionary
1 usage
def save_summary(self):
    saved_data = {
        "Title": self.article_info["Title"],
        "Author": self.article_info["Author"],
        "Publish Date": self.article_info["Publish Date"],
        "Summary": self.article_info["Summary"],
        "Sentiment": self.article_info["Sentiment"]
    }

    # Writes content into the text file
    with open("file.txt", "w") as file:
        for key, value in saved_data.items():
            file.write(f"{key}: {value}\n")
```

create_widgets, destroy and run functions

This function creates all the widgets for the SummarizeGUI. It first uses the widgets function to create the general widgets. Then I created a label and a textbox for the user to input the news article url. I also added a summarize button with the command to call the summarize function so that the article information would be displayed. Finally, I also added a save button with the command to call the save_summary function so that the article information can be saved into a text file. The destroy function is used for deleting the current instance of the GUI that is running and re-opening the already created instance of the HomeGUI. The run function runs the window.

```
# Creates the GUI by calling the widgets function and also  adding a little bit of it's own unique attributes
1 usage
def create_widgets(self):
    widgets(self)

    # Create URL Label
    url_label = tk.Label(self.root, text_="URL")
    url_label.pack()

    # Make state enabled so users can enter URL
    self.url_text = tk.Text(self.root, height=1, width=140)
    self.url_text.pack()

    # Make summarize button
    summarize_button = tk.Button(self.root, text="Summmarize", command=self.summarize)
    summarize_button.pack(pady=5)

    # Make save button
    save_button = tk.Button(self.root, text="Save Summary", command=self.save_summary)
    save_button.pack(pady=5)
```



```
# So when you click "Go To Home" the GUI will destroy itself and reopen the HomeGUI
1 usage (1 dynamic)
def destroy(self):
    self.root.destroy()
    self.home_instance.show_home()

# Run the window
def run(self):
    self.root.mainloop()
```

SavedGUI Class

This is the SavedGUI class and it is used for creating the GUI for displaying saved summaries. First I initialized the window and the variables of the summary which include title, author, publication date, summary, and sentiment. I also call the widgets function to create the GUI and also call the display_saved_data function which will display the saved data.

```
# GUI for saved summary
1 usage
class SavedGUI:
    # initializing variables
    def __init__(self, home_instance, saved_data):
        # create window, title, and instance of HomeGUI(used for switching between windows).
        self.home_instance = home_instance
        self.root = tk.Tk()
        self.root.title("Saved Summary")

        # initializing variables
        self.title = None
        self.author = None
        self.publication = None
        self.summary = None
        self.sentiment = None

        # create widgets and call the display_saved_data function
        widgets(self)
        self.display_saved_data(saved_data)
```

display_saved_data, destroy and run functions

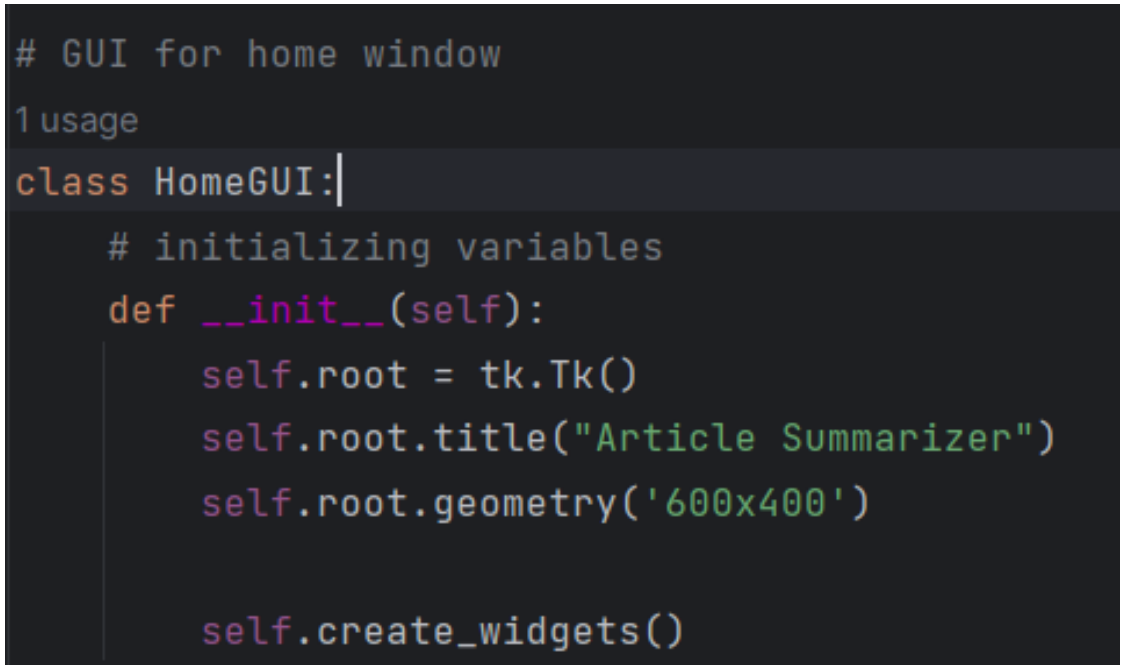
The display_saved_data changes the contents of the text labels in the GUI by getting the data from the text file and displaying it using the change_contents function. The destroy function is used for deleting the current instance of the GUI that is running and re-opening the already created instance of the HomeGUI. The run function runs the window.

```
# display the saved summary without allowing the user to change anything
1 usage
def display_saved_data(self, saved_data):
    # Change the contents of saved summary
    change_contents(self, saved_data.get("Title", ""), saved_data.get("Author", ""), saved_data.get("Publish Date", ""),
                    saved_data.get("Summary", ""), saved_data.get("Sentiment", ""))
```

```
# Function to destroy window and reopen home window
1 usage (1 dynamic)
def destroy(self):
    self.root.destroy()
    self.home_instance.show_home()

# Run the window
1 usage
def run(self):
    self.root.mainloop()
```

HomeGUI Class

This is the HomeGUI class and it is used for creating the GUI for displaying the home window. I initialized the window and also called the create_widgets function to create the widgets of the window.



```
# GUI for home window
1 usage
class HomeGUI:
    # initializing variables
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Article Summarizer")
        self.root.geometry('600x400')

        self.create_widgets()
```

show_saved_summary function

This is the show_saved_summary function and it is used to show the saved summary. It first hides the home window, then it opens the text file and creates a dictionary called “saved_data”. Then it will read the text file line by line and store the information in the dictionary. Finally, the function will create a new instance of the SavedGUI using the saved_data dictionary as a parameter so that the data in the text file can be displayed.

```
1 usage
def show_saved_summary(self):
    self.root.withdraw()
    # Read the file and put it in a dictionary called saved_data
    file_path = 'file.txt'
    with open(file_path, 'r') as file:
        saved_data = {}
        current_key = None

        for line in file:
            if line.strip():
                key_value = line.split( sep: ':',  maxsplit: 1)
                if len(key_value) == 2:
                    current_key = key_value[0].strip()
                    saved_data[current_key] = key_value[1].strip()
                elif current_key:
                    saved_data[current_key] += f'\n{line.strip()}'

        # Open an instance of SavedGUI with the retrieved saved data
        SavedGUI(self, saved_data).run()
```

create_new_summary and create_widgets functions

The create_new_summary function closes the home window and creates a new instance of the SummarizeGUI class so that a SummarizeGUI window will pop up. The create_widgets function creates widgets for the home window. There is a header and two buttons. One button is called "Check Saved Summary" with the command to execute the show_saved_summary function. The other button is called "Create New Summary" with the command to execute the create_new_summary function.

```
# Open the SummarizeGUI and hide the home window.
1 usage
def create_new_summary(self):
    self.root.withdraw()
    SummarizeGUI(self)
```

```
# Create the HomeGUI
1 usage
def create_widgets(self):
    # Header
    header_label = tk.Label(self.root, text="Article Summarizer", font=("Helvetica", 24), pady=20)
    header_label.pack()

    # Button to check saved summaries
    check_saved_button = tk.Button(self.root, text="Check Saved Summary", command=self.show_saved_summary)
    check_saved_button.pack(pady=10)

    # Button to create a new summary
    create_summary_button = tk.Button(self.root, text="Create New Summary", command=self.create_new_summary)
    create_summary_button.pack(pady=10)
```

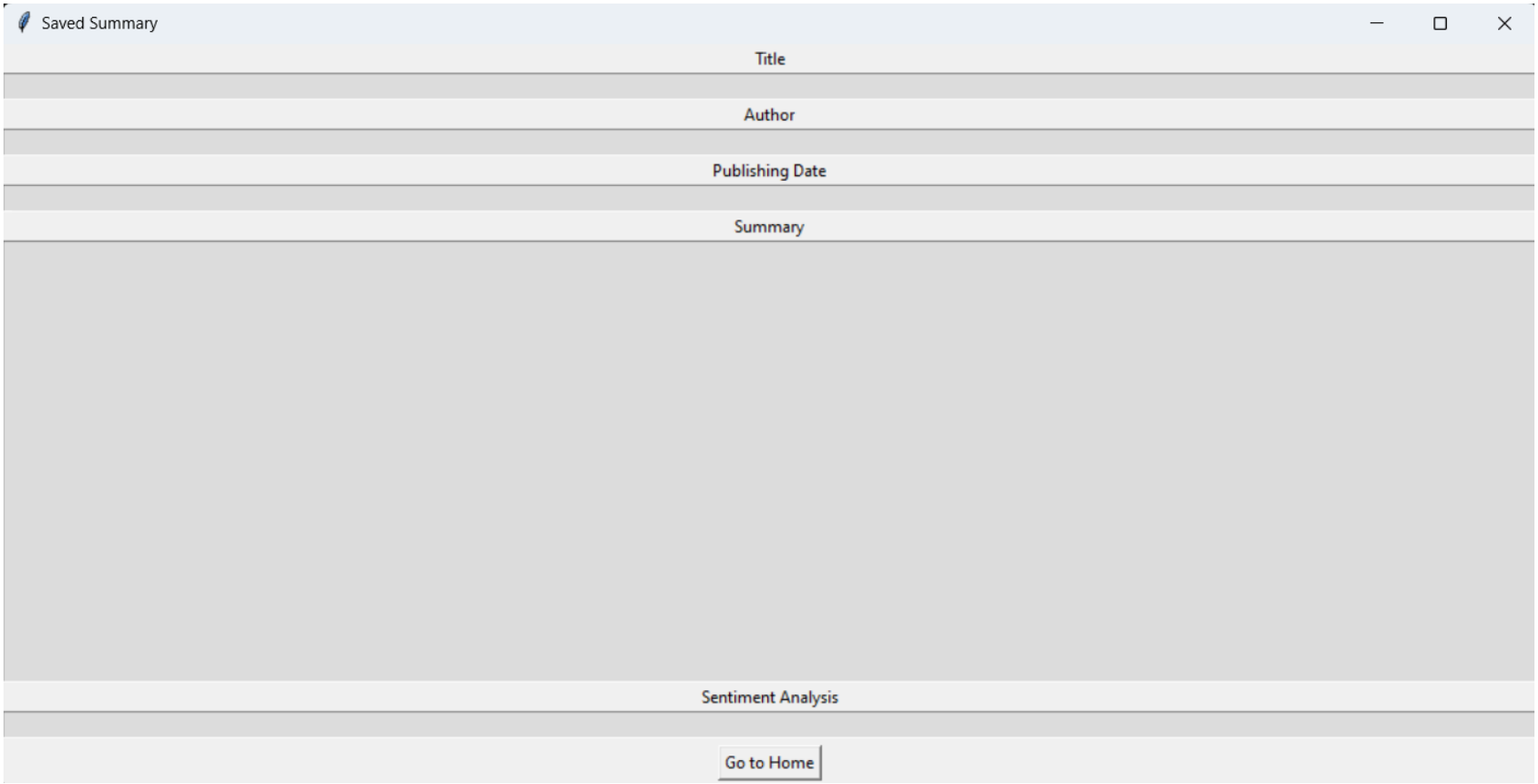
show_home, run, and main functions

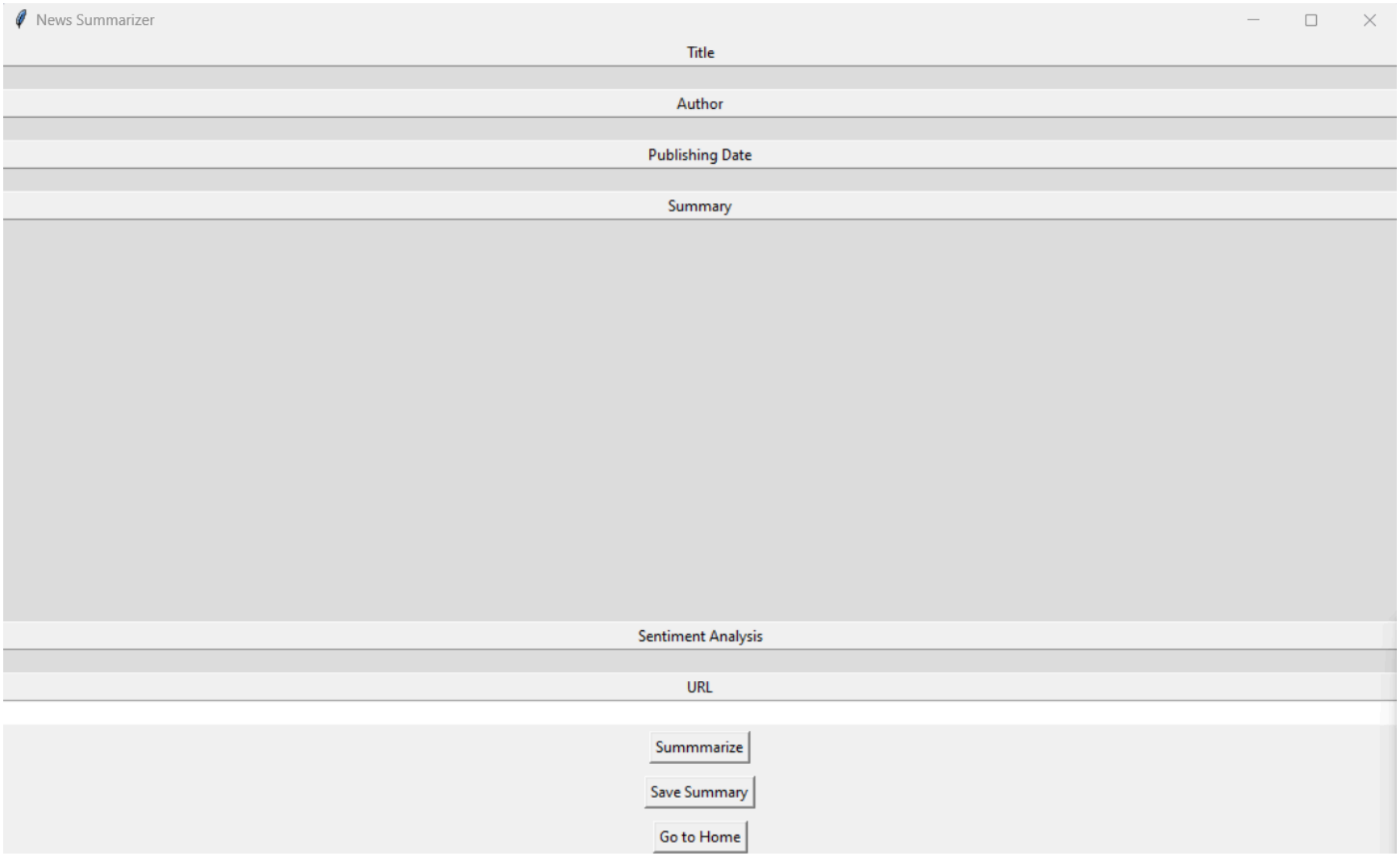
The show_home function shows the home window, and the run function keeps the home window running. The main function creates an instance of the HomeGUI and calls the run function. The whole program runs when the main function is called.

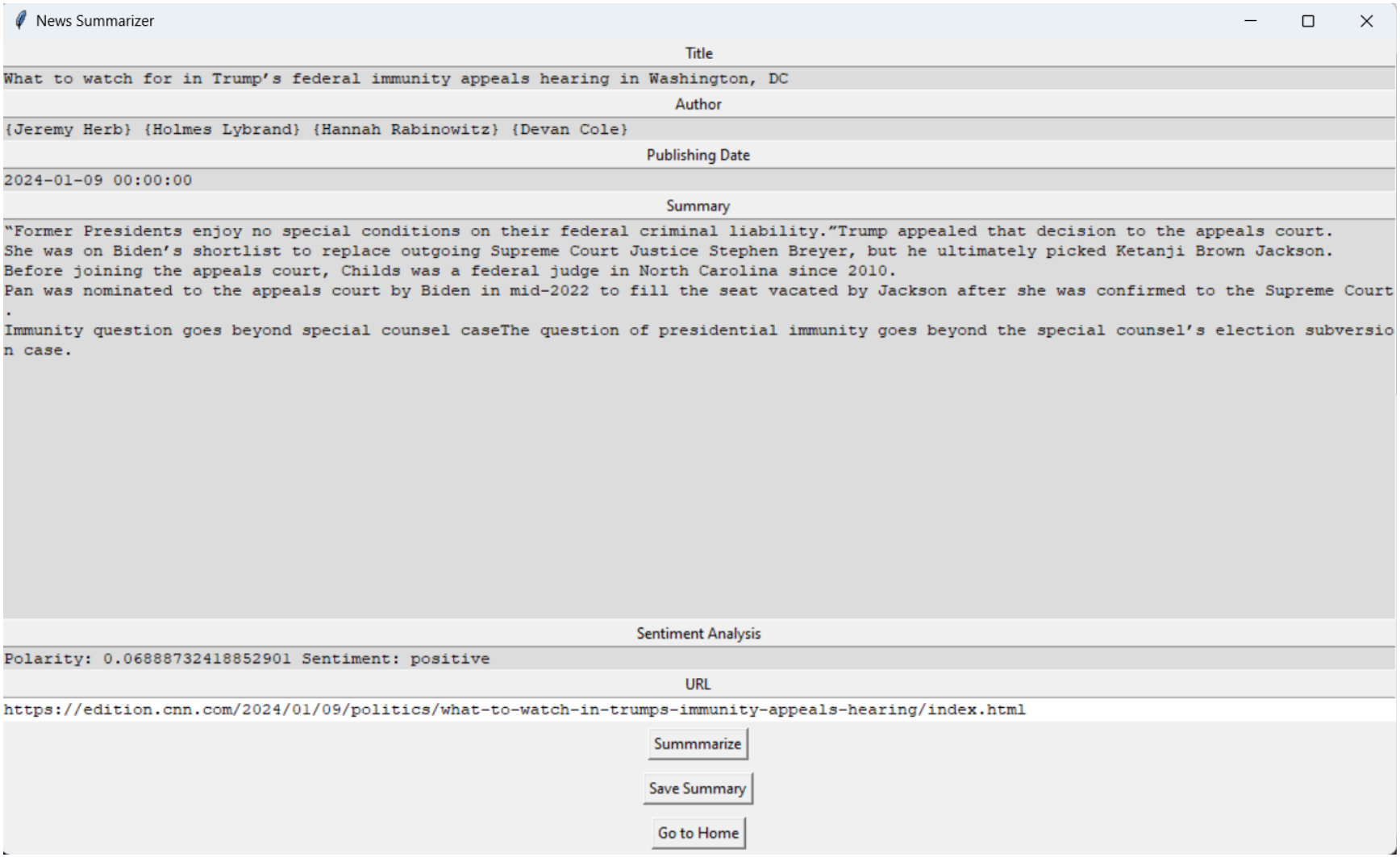
```
# Show home window
2 usages (2 dynamic)
def show_home(self):
    self.root.update()
    self.root.deiconify()

# Run the window
1 usage
def run(self):
    self.root.mainloop()
```

```
# Create an instance of HomeGUI and run it
1 usage
def main():
    home_instance = HomeGUI()
    home_instance.run()


main()
```

Evidence of Working Program


Saved Summary
Title
Author
Publishing Date
Summary
Sentiment Analysis
Go to Home


News Summarizer
Title
Author
Publishing Date
Summary
Sentiment Analysis
URL
Summmarize
Save Summary
Go to Home


News Summarizer
Title
What to watch for in Trump’s federal immunity appeals hearing in Washington, DC
Author
{Jeremy Herb} {Holmes Lybrand} {Hannah Rabinowitz} {Devan Cole}
Publishing Date
2024-01-09 00:00:00
Summary
“Former Presidents enjoy no special conditions on their federal criminal liability.”Trump appealed that decision to the appeals court.
She was on Biden’s shortlist to replace outgoing Supreme Court Justice Stephen Breyer, but he ultimately picked Ketanji Brown Jackson.
Before joining the appeals court, Childs was a federal judge in North Carolina since 2010.
Pan was nominated to the appeals court by Biden in mid-2022 to fill the seat vacated by Jackson after she was confirmed to the Supreme Court
.
Immunity question goes beyond special counsel caseThe question of presidential immunity goes beyond the special counsel’s election subversio
n case.
Sentiment Analysis
Polarity: 0.06888732418852901 Sentiment: positive
URL
https://edition.cnn.com/2024/01/09/politics/what-to-watch-in-trumps-immunity-appeals-hearing/index.html
Summmarize
Save Summary
Go to Home

Saved Summary

Title

What to watch for in Trump’s federal immunity appeals hearing in Washington, DC

Author

['Jeremy Herb', 'Holmes Lybrand', 'Hannah Rabinowitz', 'Devan Cole']

Publishing Date

2024-01-09 00:00:00

Summary

“Former Presidents enjoy no special conditions on their federal criminal liability.”Trump appealed that decision to the appeals court.
She was on Biden’s shortlist to replace outgoing Supreme Court Justice Stephen Breyer, but he ultimately picked Ketanji Brown Jackson.
Before joining the appeals court, Childs was a federal judge in North Carolina since 2010.
Pan was nominated to the appeals court by Biden in mid-2022 to fill the seat vacated by Jackson after she was confirmed to the Supreme Court
.
Immunity question goes beyond special counsel caseThe question of presidential immunity goes beyond the special counsel’s election subversio
n case.

Sentiment Analysis

Polarity: 0.06888732418852901 Sentiment: positive

Go to Home