UNIVERSITY OF ST. GALLEN

School of Management, Economics,
Law, Social Sciences, International Affairs and Computer Science

Group Project:

Poem Analysis Tool

Submitted by

Stanislaw Bucior	22-605-844
Guillaume Bongard	18-606-772
Elisabeth Schneider	17-123-654
Timothy Bucher	19-612-464

Course
Skills: Programming with Advanced Computer Languages

Professors
Dr. Mario Silic

Date of Submission
22 December 2023

Descriptive Abstract

This program uses the OpenAI GPT API to analyze poems from PDF files. It aims to automate the analysis through the API calling a trained assistant and facilitates the results into a structured format. To do so, the program prompts the user to input the path to a PDF file and then extracts the text from the file. The text is then sent to the OpenAI API as a message in a thread created for an analysis by the AI. The program uses functions to manage the interaction with the OpenAI assistant and retrieves its responses. Once the answers are obtained, the program parses the analysis results encoded in JSON format and structures this data into a tabular format using the Pandas library and creating a DataFrame for further analysis and manipulations.

1. User Instructions

- For security and functionality reasons, it is highly recommended to run this code within Visual Studio Code. The secret key is programmed to deactivate if it is running in an online service to safeguard against unauthorized access by ChatGPT. Typically, secret keys remain hidden to maintain security. However, for the convenience and educational purpose of this university project, we've made an exception and provided the key in this document. Please handle the key with care. You can copy our code using the following Github Link: https://github.com/stanislawbucior/PoemAnalysisTool/blob/main/Poem%20Analysis%20Tool%20Final%20version.py
- The program asks for an openai API key. Enter the following key: **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**
- Make sure that the libraries mentioned in chapter 2. Libraries are installed.
- Run the code.
- Now you can choose how you want to provide the poem you like to analyze (PDF or provide Text). Now, provide the text of a poem you want to analyse or choose the PDF option and upload a file in which there is a poem you want to be analyzed. Please make sure to upload PDF files only as PDF/A files are not going to work.
- The program outputs a table with the poem analysis.

2. Libraries

This section details the different libraries used in the program and their functionalities. Please make sure that these libraries are installed on your device using the Open Python Terminal.

OpenAI Python Package (openai):
The openai package facilitates interaction with the OpenAI GPT-3 API. It allows the program to create threads, send messages for analysis, run AI assistants, and retrieve responses. Install the package using the command “pip install openai”

Operating System Module (os):
The os module provides functions for interacting with the operating system. In this program, it is used to check the existence of the specified PDF file using os.path.isfile().

Time Module (time):
The time module provides time-related functions. Within this program, it's used for managing delays between checking the status of a run within an OpenAI thread.

PDF Handling Module (PyPDF2):

The PyPDF2 library is employed for reading PDF files. More specifically, it's used to extract text content from the PDF file provided by the user. You can install this module using the command: “pip install PyPDF2”

JSON Module (json):
The json module allows the program to encode and decode JSON data. We used it to parse JSON-encoded responses received from the OpenAI API.

Pandas Data Analysis Library (pandas):
The pandas library is used for data manipulation and analysis with tabular data. In this program, it's utilized to convert the JSON-based analysis output into a structured DataFrame, facilitating data organization and potential analysis.

Tk Interface Package (tkinter):
Tkinter is used for building the user interface of Python applications, providing tools to create interactive elements like buttons, text fields, and windows. We used the Tkinter package for a graphical user interface for a poem analysis tool, allowing users to choose files, input text, and display results through a simple and interactive windowed application.

3. Code Execution

This section presents the core functionalities of the program:

1. **Input PDF File Path:**
   - Invokes the *get_pdf_path()* function to prompt the user for the full path to the PDF file containing poems for analysis.
   - Checks if the specified file exists using *os.path.isfile(pdf_file_path).*
   - If the file doesn’t exist, it notifies the user that the file was not found.
   - If the file exists, it proceeds with further analysis.

2. **OpenAI Thread Creation:**
   - Generates a new thread using *client.beta.threads.create()* from the OpenAI client.
   - Collects the extracted text content from the PDF file using *extract_text_from_pdf(pdf_file_path).*

3. **Message Sending and Assistant Interaction:**
   - Sends the extracted text content for analysis by the OpenAI assistant via *send_message(thread.id, pdf_text)*.
   - Executes the run*_assistant_and_get_response()* function to run the OpenAI assistant and retrieve its responses.
   - If an error occurs during the assistant's execution, it catches the exception and prints an error message.

4. **Conversion to DataFrame:**
   - Checks if there are any responses obtained from the assistant (if answers:).
   - If responses exist, attempts to parse the JSON-encoded analysis results retrieved from the assistant.
   - The program uses Pandas *(pd.json_normalize())* to convert the analysis output into a DataFrame *(df),* enabling structured data representation for potential further analysis.

4. Functions

This section describes the functions used in the python code to create the different functionalities of the program:

get_pdf_path():

- Prompts the user to input the full path of the PDF file to be analyzed.
- Returns the entered PDF file path after removing unnecessary white spaces.

extract_text_from_pdf(pdf_file_path):

- Opens and reads the specified PDF file *(pdf_file_path).*
- Utilizes the *PdfReader* from *PyPDF2* to extract text content from each page of the PDF.
- Returns the concatenated text content of all the pages.

send_message(thread_id, user_message):

- Sends a message to an OpenAI API thread.
- Creates a message within the specified thread *(thread_id)* with the content provided *(user_message).*

wait_on_run(run, thread):

- Keeps checking the status of a specific run within a thread until it's completed.
- Retrieves the status of the run and waits until it's either completed, failed, or cancelled.
- Utilizes *time.sleep(1)* to wait for a short duration between status checks

run_assistant_and_get_response(assistant_id, thread_id, last_response_id=None):

- Initiates the OpenAI assistant within a thread and retrieves responses.
- Creates a run for the specified assistant within the given thread.
- Waits for the run to complete and retrieves the messages from the thread.
- Extracts and collects the assistant's responses since the last received response (if any).
- Returns the ID of the latest response and a list of answers received from the assistant.

5. Assignment Notes

I This code serves as an interface for a poem analysis tool, leveraging OpenAI's capabilities to interpret and provide insights into the text. It's designed with the flexibility to adapt to various tasks, such as summarizing texts, analyzing data or other functions depending on the user's needs. Each person using our written code can create their own assistant with a unique secret key by visiting the websites https://platform.openai.com/api-keys or https://platform.openai.com/assistants and changing the code accordingly.

Our assistant has been trained with the following instructions:

You have to analyze parts of poems. The user enters a poem in pdf or text form, just analyze what you've been given.

- *name (if known)*
- *author (if known)*
- *year (if known)*

- *summary (max 30 words)*
- *topic (1 word)*
- *poetic form*

Always package the output in the following JSON format. Only output the JSON format, no other responses.

If any answers are not known, give the output "Not known"

```
{
  "poemAnalysisOutput": {
    "name": "Name of the poem (if entered)",
    "author": "Author of the poem (if entered)",
    "year": "Year of publication (if entered)",
    "summary": "Short summary (if entered)",
    "topic": "Main theme of the poem (if entered)",
    "poeticForm": "Form of the poem (if entered)",
    "analysis": {
      "themes": "Identified themes and motifs",
      "style": "Style and language analysis",
      "structure": "Analysis of the structure and composition",
      "historicalContext": "Historical context of the poem",
      "literaryDevices": "Literary devices used",
      "interpretation": "Possible interpretations and meanings"
    }
  }
}
```

It's important to note that using this service and code is not free. The pricing details, which depend on factors like the size of the files processed, and the output provided by ChatGPT, can be reviewed at

OpenAI's Pricing page. For this project, $5 have been provided to the secret key as during our tests, we figured out that each poem analysis only costs a few cents.

If there are any questions concerning the API secret key or the provided budget, please contact Elisabeth Schneider at elisabeth.schneider@unisg.ch