F21SC Industrial Programming
Assessed Coursework 2

Team members:

Yan Min Chan (H00321508)
Xavier Mitault (H00478948)

Contribution:

Yan Min has contributed to the design of the GUI (Graphical User Interface) and graph plotting using *matplotlib* package in the program, the introduction, requirements' checklist, user guide, developer guide, reflection, conclusion and reference in the report. Yan Min also resentenced the report.

Xavier has contributed to the command-line part, the improvement of the file loading, design consideration, the user guide, developer guide, reflection and reference in the report.

The team has a one hour meeting every week and most part of the program is done by pair programming.

Introduction

A program to analyse and display document tracking data from a major web site is developed in this coursework. The program is developed in Python 3 and the data provided is written in JSON format (each line is a new JSON item). This report is intended for the program user to aid on navigating the GUI (Graphical User Interface) and the CLI (Command Line Interface) usage and for developers to have a clear understanding on the program for further development.

Requirements' checklist

The functionalities implemented in the program includes:

1. **Loads file:** The program loads the JSON file containing document tracking data provided by the user. It is available in both GUI usage and a CLI usage. A JSON file with 3 million lines takes around 6 seconds to process in an average computer (with cache enabled).
2. **Document view by country/continent:** The program is given a document id "env_doc_id" and the countries and continents where the document has been viewed will be displayed as a bar plot. It is assumed that "viewed" is the "impression" event type.
3. **Views by browser:** The program identify the popularity of browsers from the data set and the occurrence of each browser will be displayed as a bar plot. There are two options to view by browser, first is all browser identifiers ("View by Browser") and second is browsers classified into main browser ("View by Main Browser"). Both options allow the program user to filter browser views by various "event_type" or including all of them, providing the functionality to identify browser popularity on different event type.
4. **Reader profiles:** The program identify the top 10 readers based on their total time spent reading documents ("event_type" is "pagereadtime"). The result can be displayed as text in the GUI or a bar plot with user id and their total time spent reading displayed.
5. **"Also likes" functionality:** The program takes a document id and a user id, returning a list of "also like" documents with the number of other users that had also read them. If there are more than 10 documents, only the top 10 will be displayed. The results can be displayed as text in the GUI or a graph highlighting the input document and user in green.
6. **GUI usage:** The program launches a GUI with all the functionalities mentioned above. A walkthrough of the GUI is provided in the User Guide section.
7. **CLI usage:** The program can be called to get directly the features leveraging a command line interface. It helps for usage in scripts or headless environments (no graphical interface available).

Design Considerations

The application (GUI) will be launched if no parameters are given when running the program in the terminal. This helps user to launch the program without having to know the parameters and options.

The GUI has a page per feature. This helps clean the design and let the user know exactly where he is and what parameters need to be given.

To select a feature, in the Tools Bar, the menu “Feature” offers a dropdown with the choices available.

The program includes a help option to give explanation about each parameter and options. This guides the user when running the program in the terminal and helps developers to understand the program when considering further development.

Each option from the CLI has a way to ask for the plot version to be shown instead of the text format.

User Guide

1. Download the program file.
2. Launch the program by double clicking on it or running it inside a terminal.

Graphical User Interface

The home page is the page with “Also Likes” feature.

The screenshots of all pages for each feature and the description are as below:

1. Features: A dropdown menu of the functionalities. (example below on the right)
2. Load File: Opens a window prompting user to load .json file containing the data.

Also likes functionality:

3. Input boxes for entering the *document id* and *user id*. The user with *user id* should had read the document with *document id*.
4. Render: Generate the result in basic text format.
5. Generate graph: Generate the result in bar plot to be more visual friendly.
6. Zone where the text result will be.

Document view by country functionality (similar for view by continent):

7. Input box for entering the *document id*. Clicking the ok button will display a bar plot summarising the countries where the viewers are from (summarise continent the viewers are from for View by continent).

View by browser (similar for view by main browser):

8. A drop-down listing event types to analyse browser popularity (main browser popularity for View by main browser).

Reader profiles:

9. Render: Generate the result in basic text format.
10. Generate graph: Generate the result in bar plot to be more visual friendly.
11. Zone where the text result will be.

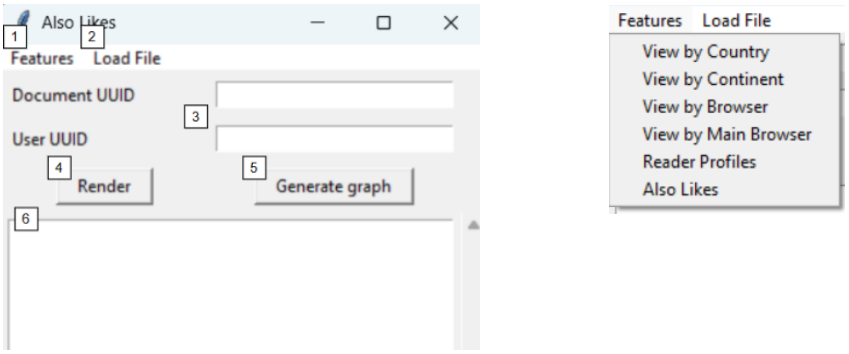

Figure 1: The "Also Likes" page

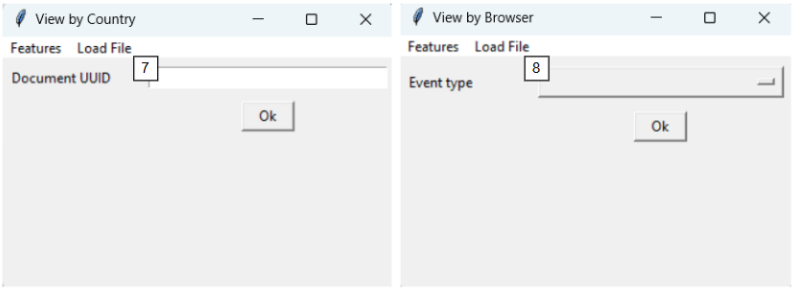

Figure 2: The "View by Country" and "View by Browser" page

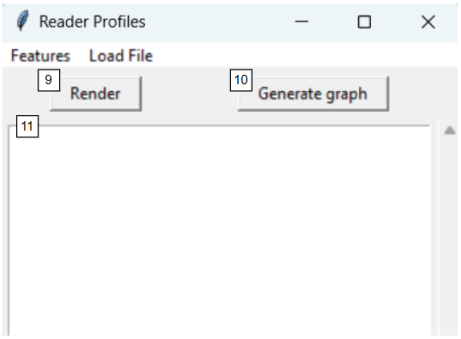

Figure 3: The "Reader Profiles" page

Command Line Interface

To see more options run it with the “--help” option. Example of output:

```
usage: f21sccw2.py [-h] [-u user_uuid] [-d doc_uuid] [-t task_id] [-f file] [--disable-cache] [--gui]

options:
  -h, --help       show this help message and exit
  -u user_uuid     user uuid
  -d doc_uuid      document uuid
  -t task_id       task id
  -f file          json file
  --disable-cache  disable cache for file (create another file with the same name .pkl by default)
  --gui            use matplotlib render or basic text render
```

Example of usage:

f21sccw2 -d “110411015935-6b1fc1a5af3540338aaf984038b74c23” -u “450e62176225c0d1” -t 4 -f “./samples/sample_3m_lines.json”

Example of results for the task_id 4:

visitor_uuid	event_readtime
000bfe531c92c6d6	504778.0
000c91a806745d7a	359872.0
000971745615596c	340188.0
0008dae6efe5b5a7	308375.0
0003a860c29e6ad2	130909.0
000a5e67d8e9643b	62808.0
0006991ef3d72343	43775.0
00052ce80e7f6bc9	23535.0
0003350a9255754f	3000.0
0003e601dcbc9dea	1514.0

Developer Guide

Some additional libraries used in the program are *orjson*, *pycountry_convert*, *graphviz*.

The *orjson* library is a more efficient library comparing to the *json* internal library to parse JSON formatted data. The *pycountry_convert* package is used to convert country code from ISO3166 alpha-2 format into its continent name. The *graphviz* library is an interface over the *.dot* program from *graphviz*.

To install them, run the following commands:

pip install orjson
pip install pycountry_convert
pip install graphviz

Alternatively, one can use the *pyproject.toml* file and a python package manager like *uv* or *pdm* to install all the dependencies. Any dependencies that a developer want to add should be listed in the *pyproject.toml* file.

A model-view-controller (MVC) design is used in the design of the GUI in the program. A brief description of the Model, View and Controller class is included below:

Class	**Description**
Model	Code specific to the JSON and pandas manipulation. Processes the JSON data in a pandas dataframe.
View	Code specific to the design of the GUI, it only uses the controller to interact with the models.
Controller	Interface to the Model, parse the result of the Model and export it in a way that the View can understand.

Cache is also implemented in loading JSON file in the program. It is a basic caching mechanism using the *pickle* python internal library that dump a class data on a file under a binary format. It helps reduce file reading, JSON parsing and pandas dataframe creation time cost.

To test the program for regression or unexpected crash, please execute the tests before submitting the modification:

python src/test_model.py

A *README.md* is provided too.

Testing

A Python internal library *unittest* is used to test the basic functionality of the program.

The test cases are:

1. Performance of *load_data* with the cache mechanism set as True and False.
2. Expected value from each functionality.
3. Catching ValueError Exception when a functionality of the program is run without loading any JSON data.

The tests are in the *src/test_model.py* file.

When the code is bundled to get an executable, the graphviz dot program can't access the configuration file needed to generate a pdf. It works fine when we execute our program source with the python executable.

Reflections

Python is great language for prototyping purpose. It is both easy to understand (English like) and easy to do a lot of things with it. There are also a lot of packages readily available for different functionality. For example, the *graphviz* package supports .dot language of the Graphviz drawing software, making graph drawing in Python way easier.

We used the MVC design pattern in classes to separately handle the display and backend code in our program. Next, we added type hint, where this new capability of Python provided a lot of help in pair programming, and make understanding code written by another developer easier. Then, Exception is used to raise errors inside the program, preventing program crash.

The main limitation of the application is the time needed to parse and load the json files. We used *mmap*, *orjson* and cache methods but the time usage is always a little too much. We tried to add threading but the way we tried it did not decrease the time. We could try to use pool processing of chunk of text and not line by line. This should improve time usage.

Python is a great language for statisticians because of the amount of library available in this field. The library *pandas* is a goldmine for Python. A new "*pandas*-like" library called *polars* is being developed that is in most of the case more performant, so python will stay the main language for this kind of usage.

Conclusion

We are proud of the performance improvement from loading the file using *pandas.from_csv(lines=True)* to using *mmap*, *orjson* and cache methods.

The usage of *tkinter* was new for us so we learned how to make simple and quick interface. We are also proud of the simple, yet nice interface created, using *grid()* from the package to arrange widgets.

The usage of pandas reminded us how to use dataframe for large action on formated data.

References

Graphviz. Python Package Index, version 0.20.3. Available at: https://pypi.org/project/graphviz/

Ijl, Orjson. GitHub repository, version 3.10.12. Available at: https://github.com/ijl/orjson

J.D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007. Available at: https://matplotlib.org/

Lundh, F., 1999. An introduction to tkinter. Available at: https://docs.python.org/3/library/tkinter.html

McKinney, W. & others, 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*. pp. 51–56. Available at: https://pandas.pydata.org/

Pycountry-convert. Python Package Index, version 0.7.2. Available at: https://pypi.org/project/pycountry-convert/

GeeksforGeeks, 2019. Python | Menu widget in tkinter, seen on November. Available at: https://www.geeksforgeeks.org/python-menu-widget-in-tkinter/

GeeksforGeeks, 2022. Tkinter Application to Switch Between Different Page Frames. Available at: https://www.geeksforgeeks.org/tkinter-application-to-switch-between-different-page-frames/