Greek Philosphy Definition Search (GPDS) Program

Members: Nguyen Hoa Quynh Nhung, Tran Nguyen Anh Thu

Introduction to Computer Programming (Python), Sep 2023
University of Trento, Italy

1 Project Description

Greek Philosophy Definition Search (GPDS) is a small program aimed at beginners and enthusiasts of Ancient Greek Philosophy, retrieving text from relevant English works between 1882 and 1920. The program may be used for research purposes by suggesting relevant keywords and definitions, or simply for creating graphic art inspired by philosophical concepts. The program offers two main functions: Definition Search and Word Cloud Builder, as well as a save option for text output.

1.0.1 General Concept

The initial concept was a chatbot program who could respond to the user with a 5-sentence (maximum) paragraph on a certain topic, made by appending sentences from relevant books. Due to the resulting paragraph lacking in cohesion, the concept was later changed into providing 5 different definitive sentences for the requested keyword, plus the corresponding book information. Another feature, the word cloud, was introduced to help the user visualize relevant concepts, while adding aesthetic value to the program.

The project's theme is Ancient Greek Philosophy, since this is a topic where Project Gutenberg's XIX-XX century books would not be outdated. The 5 default books in this program, all written by popular researchers of their time, range from summarization (by John Marshall) to critique and commentary (A.W. Benn, Friedrich Nietzsche).

We also decided to simplify user interaction to input and output textboxes, along with adding the word cloud feature. The current user interface now consists of one main window, a tkinter save-file window for the Save feature, and the terminal mainly for text preprocessing.

1.1 Components

The program requires the following folders and files to function properly:

Folder	Files within	Description
Main folder	main.py, main_class.py program_module.py, text_process_module.py, README.txt, requirements.txt	The root file is main.py. The 3 other .py files are its modules. The 'README' and 'requirements' files are used for installation.
output	.csv - preprocessed text data; .png - word cloud images	The .csv files and cloud images are in the same **output** folder for convenience, since the .csv files are also the output of text preprocessing.
original_txt	.txt - unprocessed text data	The .txt files are sources for the .csv files. Its contents are not compulsory for running main.py, but are needed for text preprocessing.
images	.png - for buttons; .jpeg - for background (bg) assets	These files are replaceable, as long as the new file shares the same name, size and type.

1.2 Development

1.2.1 Data Structure

Initially, the team planned to organize the data as a .json file. However, to facilitate later .csv files down the pipeline, the team decided to use only .csv files for the entire data structure. The option to export all sentences as a .json file is still available as a function in program_module.py, if the user wishes to utilize it for another purpose. This is why the hierarchical data structure is still retained for flexibility, instead of duplicating information for each dataframe item.

The current data include both mandatory and optional data. The optional data are mostly not called during the main.py process, though they may aid users during citation.

The processed .csv files include the all_books.csv for the word cloud feature, and the other three files: keyword.csv, sent_def.csv, and sent_other.csv. To generate five definitions (maximum), the program selects at most two random sentences from the higher-priority sent_def.csv file, then selects three or more sentences from sent_other.csv until there are exactly five sentences. The higher-priority file contains sentences with the desired structures that characterize a definition, ranked as follows (the smaller, the higher): "defined":1, "definition":1, "refers":2, "refer":2, "means":2, "mean":2, "imply":3, "im-

plies":3, "implied": 3, "can be":4.

The sent_other.csv contain all sentences that do not contain these structures, but could potentially contain definitive (e.g. "X + is/are") or informative sentences. For sent_other.csv, the program will select items containing the most occurrences of the requested keyword. The last file, keyword.csv is originally meant for suggesting and/or limiting the keywords that the user could use as input. However, due to time limitations, this auto-suggestion (text prediction) feature has not been implemented yet.

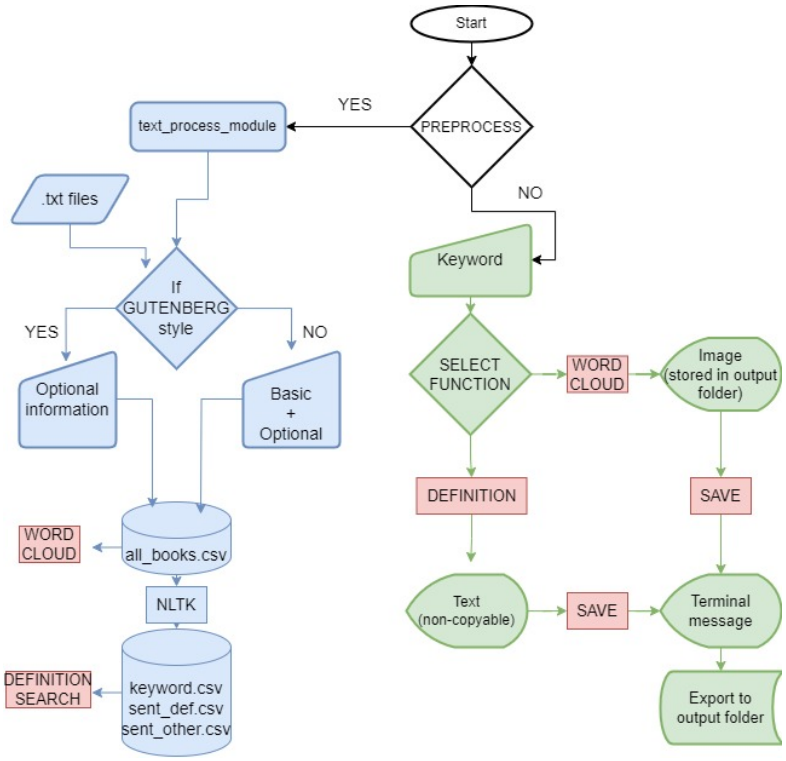

Figure 1: Flowchart from preprocessing to export.

1.2.2 Main Libraries

- pandas: flexible data structure and easy export;
- nltk: natural language processing to separate sentences instead of relying on punctuation/spelling rules;
- wordcloud: generates a word cloud from a given string, fully customizable with image templates;
- pygame: an intuitive GUI option for simple games and programs, also very OOP-friendly.
  Downside: likely to cause crashes and slowdowns if best practices are not followed.

1.3 Future Suggestions

There are many possible ways to enrich the functionality and optimize this app, given more time and coding experience:

- Use the keyword.csv file to make keyword suggestions while the user is typing;
- Increase the word cloud's image quality. The current image output is a 370x370 pixels image (excluding the borders which pad the image to 640x480);
- Allow for copying to user clipboard and saving directly by right-clicking;
- Experiment with more flexible GUI options such as Pyqt5, wxpython or GTK.

1.4 Contributions

Member Nguyen Hoa Quynh Nhung contributed to: the definition feature (nltk-to-csv component **(3 csv files)** + the definition search function), as well as code optimization throughout the project (extensive code cleanup and OOP changes for **program_module + text_process**; small error fix to main.py).

Member Tran Nguyen Anh Thu is responsible for: the word cloud feature (the text preprocess component until **all_books.csv** + the word cloud function), the pygame GUI (**main.py + main_class.py** with OOP), the PDF report and README file. All members are responsible for the project's general idea and feature concepts.

2 Links

- Github repository at: https://github.com/mayleyc/gpds
- All books retrieved from: https://www.gutenberg.org/
  NB: The .txt files used for processing are also kept in the txt_original folder.