Data collection

A computer screen shot of text

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

A black screen with text on it

Description automatically generated

A black background with yellow and green lights

Description automatically generated

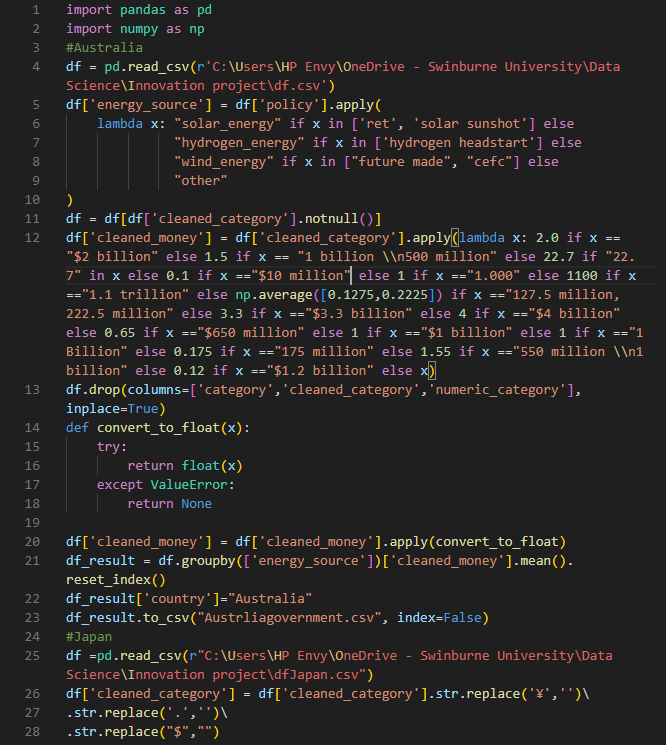
This code demonstrates a process for:

1. Extracting text from multiple PDFs in a directory.
2. Preprocessing the extracted text (cleaning and normalization).
3. Using OpenAI's API to categorize the text based on subsidies offered.

There are 7 steps:

|  |  |
| --- | --- |
| Step | Code demonstration |
| Pdf extraction and dataframe creation | Imports necessary libraries: extract\_text from pdfminer.high\_level, pandas for data manipulation, and os for file system access.  Defines the directory containing your PDFs.  Uses os.listdir to get a list of filenames in the directory.  Creates an empty Pandas DataFrame with columns file\_name and text.  Iterates through each filename:   * Constructs the full path to the PDF using the directory and filename. * Extracts text from the PDF using extract\_text and stores it in the text variable. * Creates a temporary DataFrame df1 with file\_name and text data. * Concatenates df1 with the main DataFrame df to accumulate data for all PDFs. |
| Data preprocessing | Converts all filenames to lowercase for case-insensitive processing.  Creates a new column policy to categorize the processed text |
| Text preprocessing functions | Installs necessary libraries for text processing: nltk, textblob.  Downloads required resources from NLTK (stopwords, wordnet).  Defines functions for various text preprocessing steps:   * tokenization: Splits text into individual words. * to\_lowecase: Converts all words to lowercase. * remove\_stopwords: Removes common stop words like "the", "a", "an". * stem\_tokens (optional): Applies stemming to reduce words to their base form (e.g., "running" becomes "run"). * lemmatize\_tokens (optional): Applies lemmatization to reduce words to their dictionary form (e.g., "running" becomes "run"). * remove\_special\_characters: Removes special characters like punctuation and symbols. * handle\_emojis (optional): Converts emojis to text representation using demojize. * correct\_spelling (optional): Attempts to correct potential spelling errors using TextBlob.   Defines a function preprocess\_text that combines these steps for comprehensive cleaning. |
| Applying preprocessing | Applies the preprocess\_text function to each row's text in the text column.  Stores the cleaned text in a new column named cleaned\_text. |
| Openai integration for category extraction | Initializes an OpenAI client assuming a local instance running on port 1234 and provides your API key.  Defines a function get\_category that takes a keyword (potentially the cleaned text) as input.  Uses OpenAI's chat completion API ("mlx-community/Llama-3.2-3B-Instruct-4bit" model) to interact with the model in a question-answer format.  The prompt asks the model to find the subsidy number based on the keyword (potentially the content of the document).  Extracts the predicted response from the OpenAI response and returns it. |
| Extracting category information | Creates a new column category initially filled with NaN (Not a Number).  Defines a regular expression pattern to identify the content within single quotes from the OpenAI response (assuming the predicted subsidy number is within quotes).  Iterates through each row in the DataFrame:   * Extracts the category information using the regular expression. * If a match is found, updates the category column with the extracted subsidy number. |
| Cleaning and assigning final category | Creates a new column cleaned\_category initially filled with NaN.  Uses the same regular expression pattern as before.  Iterates through each row using .iterrows(): |

Cleaning data



A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

A black background with numbers and symbols

Description automatically generated

The main purpose of this code is to clean the raw data after crawling and after that the code will concat all these three types of data and exchange the money currency to the same scale.

In this code we will do:

Extract and clean data from CSV files containing information on government subsidies.

Assign energy sources to each data point based on relevant keywords or patterns.

Calculate monetary values associated with the subsidies, considering different currency formats and units.

Group and analyze the data to compare subsidy levels across different energy sources and countries.

Output the results in a structured format (CSV files) for further analysis or visualization.

There are 8 steps:

|  |  |
| --- | --- |
| Step | Code demonstration |
| Import libraries | Imports pandas and numpy for data manipulation and numerical operations. |
| Load data | Reads a CSV file named "df.csv" into a Pandas DataFrame named df. |
| Assign energy source | Creates a new column energy\_source based on the policy column.  Uses lambda expressions to assign energy sources based on specific keywords within the policy values. |
| Filter data | Filters the DataFrame to keep only rows where the cleaned\_category is not null, ensuring only rows with valid categories are included. |
| Clean and convert money amounts | Creates a new column cleaned\_money to store cleaned and converted monetary values.  Applies a lambda function to clean the cleaned\_category column by removing currency symbols and periods.  Converts the cleaned values to numeric format using a conver\_number function that handles potential conversion issues.  Assigns specific numeric values based on the cleaned categories, considering various patterns and keywords. |
| Data cleaning and conversion | Converts the cleaned\_money column to float for numerical calculations.  Drops unnecessary columns category, cleaned\_category, and numeric\_category. |
| Group and analyze data | Groups the DataFrame by energy\_source and calculates the mean of cleaned\_money for each group.  Resets the index to create a new DataFrame with columns energy\_source and cleaned\_money  Adds a country column with the value "Australia" to indicate the data's origin. |
| Concatenate and convert currency | Concatenates the results from Australia, Japan, and China into a single DataFrame df\_total.  Converts the cleaned\_money values for Japan and China to USD using conversion rates. |