README

NLP and Knowledge Graph-Based Analysis of SWECRIS and ERC Database

Capstone- Odd Semester 2023

Guide: Dr. N Arul Murugan

Group 5: Abhishek Acharya (MT22004)

Note:-

1. To run the Notebook in Google Colab, we need to connect the computer to the internet and simply Run the Notebook. As an alternative approach, the CSV files can be read directly and the rest of the code remains the same.
2. All the outputs and graphs will be generated one by one. As there are a lot of operations, it takes a few minutes to get all the outputs.

1. SweCris_Database.ipynb

Key steps in the code include:

a. Data Retrieval: Utilizing the SweCris API to fetch project data related to diseases, viruses, and infections.
b. Data Processing: Cleaning and organizing the retrieved data into a structured format.
c. Text Analysis: Employing natural language processing techniques to analyze project abstracts. This includes the extraction of relationships between keywords (e.g., disease, virus, infection) and other relevant words occurring within a specified range.
d. Knowledge Graph Generation: Creating knowledge graphs to visually represent relationships between keywords and associated terms in project abstracts.
e. Frequency Word Graph: Calculating the frequencies of words associated with keywords, assigning weights based on these frequencies, and plotting the top-weighted words for each keyword.
f. Graph Filtering: Filtering the knowledge graph to focus on nodes with a degree greater than one, reducing graph density and emphasizing more significant relationships.
g. Graph Visualization: Generating visual representations of the knowledge graph using networkx and matplotlib.

The code serves as a comprehensive tool for exploring and understanding the prevalent themes, associations, and trends within the SweCris Database related to diseases, viruses, and infections.

2. ***ERC_Database.ipynb***
   This code works similar to the SweCris Database, and does all the key operations stated above.

3. ***Analysis_and_Comparison_SWECRIS_and_ERC.ipynb***
   This code performs an analysis on two datasets, SWECRIS and the ERC Database, focusing on project titles, abstracts, and disease mentions. The major steps are:-

   A. SWECRIS Data Retrieval:
      a. Fetches project data from the SWECRIS API.
      b. Extracts relevant columns and filters out unnecessary information.
      c. Tokenizes and preprocesses project titles for each funding year range.
      d. Identifies the top 10 words in project titles for each funding year range.

   B. ERC Database Data Retrieval:
      a. Reads project data from an Excel file (ERC Database).
      b. Extracts relevant columns and preprocesses project titles similar to SWECRIS.
      c. Identifies the top 10 words in project titles for each funding year range.

   C. Word Frequency Visualization:
      a. Plots histograms comparing the top 10 words in project titles between SWECRIS and ERC for each funding year range.

   D. Project Abstract Analysis:
      a. Analyzes project abstracts from both SWECRIS and ERC datasets.
      b. Tokenizes, lemmatizes, and filters stop words for abstracts.
      c. Identifies the top 100 most common words in project abstracts across all years and selects the relevant top 10 words.
      d. Defines additional stopwords to be removed from the analysis.
      e. Visualizes the top 10 words in project abstracts for each funding year range.

E. Disease Mention Analysis:
    a. Defines a list of diseases of interest.
    b. Determines the top 10 diseases mentioned in project titles and abstracts for both SWECRIS and ERC.
    c. Plots year-wise trends for the occurrence of the top diseases, combining data from both datasets.
    d. In summary, the code fetches project data from two sources, analyzes and visualizes word frequencies in project titles and abstracts, and explores trends in disease mentions over the years in both datasets.

In summary, the code fetches project data from two sources, analyzes and visualizes word frequencies in project titles and abstracts, and explores trends in disease mentions over the years in both datasets.

4. ***SERB_PRISM_Database.ipynb***
   [This code will give an error at the end.]

   The Code attempts to use Selenium and BeautifulSoup to scrape data from different websites. Here's a concise description of the main functionalities:

   A. Web Scraping with Selenium and BeautifulSoup for Cereals and Oilseeds Markets:
      a. Selenium is used to automate a web browser (Chrome) in a headless mode (without a visible GUI).
      b. BeautifulSoup is employed to parse the HTML content of the web page.
      c. The target site is "https://ahdb.org.uk/cereals-oilseeds-markets."
      d. Multiple tables from the site are extracted using pd.read_html(html), and the content of the first two tables is displayed.

   B. Web Scraping with Selenium for Census Data:
      a. Requests and BeautifulSoup are used to extract values of social groups from the site "https://agcensus.dacnet.nic.in/DistCharacteristic.aspx."

b. Selenium is then used to interact with dropdown lists and submit buttons to scrape data for different social groups and districts. The data is extracted and stored in a list.

C. Web Scraping with Selenium for Project Details:
   a. The code attempts to extract project details from "https://prism.serbonline.in/FundingDetails."
   b. It iterates through different years and programs, selects dropdown values, and attempts to click on a submit button to fetch the result.
   c. Similar to the Census Data scraping, this code faces challenges due to dynamic elements, absence of IDs, and potential issues with available data for different years.

D. Issues and Comments:
   a. The code for scraping Census Data is commented out due to multiple issues:
   b. The site is dynamic, making it challenging to interact with dropdowns and submit buttons.
   c. The absence of IDs for some elements, like the submit button, complicates automation.
   d. Handling multiple pages of tables is not addressed.
   e. Not all program options are available for all years, leading to potential issues reading blank tables.

In summary, the code attempts to automate web scraping tasks using Selenium and BeautifulSoup, but it faces challenges due to the dynamic nature of the websites and the absence of certain elements needed for automation.