CP423 Text Retrieval

Assignment 1 Report

Group Members:

Obi Ihejirika - 190970850

Jeetindra Dhori - 180726000

Griffin Krunic - 180432940

Group Member Contribution

Every group member completed a question and we each completed the explanations for each:

- Obi Ihejirika completed question 1
- Jeetindra Dhori completed question 2
- Griffin Krunic completed question 3

Question 1 Explanation

Webcrawler1 allows a user to crawl through a website and download its contents. It takes the URL of the website to start the crawl as an argument, along with several optional arguments such as the maximum depth to crawl, whether to rewrite the contents if they already exist, and whether to display verbose information while crawling.

The script uses the following functions =:

1) get_hash: calculates the hash value for a given URL.
2) download_page: downloads the content of a URL.
3) extract_links: extracts all hyperlinks from the downloaded content.
4) write_file: writes the downloaded content to a file.
5) log_crawl: appends the crawling information to a log file.
6) crawl: the main function that performs the crawl. It takes a URL, current depth, maximum depth, rewrite flag, verbose flag, and log file name as arguments.
7) file_exists: checks if a file exists.
8) main: the main function of the script that parses the arguments and calls the crawl function.

The script can be run from the terminal using the command "python webcrawler1.py <url> --maxdepth <depth> --rewrite <True/False> --verbose <True/False>".

Question 2 Explanation

Webcrawler2 uses the requests library to download the webpages, and then uses beautifulsoup4 to parse the data as html. I separated the code into multiple functions for better readability and manageability:

1) get_hash(url)

    This function creates a hash to be used as the filename based on imputed url

2) create_hash_file(url, data)

    This function creates the H.text file of the get_hash of the url and the downloaded page data

3) create_json_file(url, data)

    This function creates the file json file using get_hash of the url, and the json data

4) download_page(url)

    This function uses the request library to download the webpage, and if it cannot be downloaded properly throw an error

5) extract_data(soup, data)

    This function uses the webpage data that was parsed into html by bs4 and searches for the specific combination of tags, classes, and ids that identify the data points we want. We manually search for the combination to find the data we want and then add it to a dictionary called 'data' with the key as the requested name for the data point, such as 'researcher_title' or 'co_author_link'. Some data points such as the citations, are grouped together in the html, so we get the text containing all of them, and then split it. Since they are in a specific order, we know which splice they are in and we assign the corresponding values to the data points and add that to the dict.

6) extract_paper_data(papers, paper)

    Extracting the data for papers are different as it requires us to open a new webpage and extract data from it. So I separated this function from the extract_data function for better readability and then I extract the data in the same fashion as extract_data. The only difference is since all the fields have the same tag, I had to manually search all the fields for the correct data point and then extract the data of their corresponding value instead.

7) __name__ == '__main__'

The main function, just takes the url parameter from argos and then continues to run all the functions in the proper order

Question 3 Explanation

Webcrawler3.py consists of one main function called webcrawler3 which takes one argument being ‘url’.The following steps are taken within this function:

1) The content of the initial url that the user provides gets downloaded
2) The hash value of the url is then calculated
3) All of the html tags are replaced with 1
4) All of the words are replaced with 0
5) These values are then split into a list of strings and assigned to the ‘final_content’ variable
6) All combinations of i and j are then calculated to find the best i and j values with the
7) This content is then written to a file and downloaded by opening a file for writing using the ‘with’ statement and the ‘open’ function.the file name is then created by adding ‘.txt’ to the end of the ‘hash_value’. The write method is then called writing the list ‘final_content[iBest:jBest]’ to the specified file
8) The data is then plotted using matplotlib onto a 3d surface where x and y axes are grids of indices and the z axis is the count of the number of times the character ‘0’ in the substrings of the final_content variable.