Introduction to Information Retrieval

Assignment 01: Web Crawler

Web Crawler for Extracting Data from Tuoitre.vn

Objective: Develop a Python-based web crawler that retrieves at least 100 posts from three different categories on `tuoitre.vn`. For each post, the crawler should extract detailed information, including the main content, comments, and media files (audio and images), and save them locally in a structured format.

Requirements:

1. **Programming Language:** Use Python.
2. **Libraries:** Using Python libs (e.g. BeautifulSoup, Scrapy, Selenium...) to crawl for direct web scraping.
3. **Functionality:**
    - The program should accept the following inputs:
        - Category URLs (three different categories on `tuoitre.vn`).
        - Number of posts to retrieve from each category (at least **100** posts total across all categories).
    - For each category, scrape the specified number of posts, collecting comprehensive details about each post.
    - Ensure that at least one post has more than **20** comments.
    - Ensure the program handles errors gracefully (e.g., invalid URLs, missing data, network issues).
4. **Data Extraction:**
    - **For each post, extract and save:**
        - **Post Details:**
            1. `postId`: A unique identifier for the post.
            2. `title`: The title of the post.
            3. `content`: The main content/body of the post.
            4. `author`: The name of the author or publisher.
            5. `date`: The publication date of the post.
            6. `category`: The category from which the post was scraped.
            7. `audio podcast`: The URL of any audio content associated with the post (if available). Save the audio file locally as `./audio/<postId>.<audio_extension>` (e.g., `./audio/12345.mp3`).

        - **Media Files:** Save all images from the post in a local folder named after the `postId` (e.g., `./images/<postId>/`). Each image should retain its original filename and extension (e.g., `./images/12345/image1.jpg`).
        - **Vote Reactions:** Extract and list all available vote reactions for the post (e.g., like, love, angry, etc.).
    - **For each comment, extract and save:**
        - `commentId`: A unique identifier for the comment.
        - `author`: The name of the comment author.
        - `text`: The comment text.
        - `date`: The publication date of the comment.
        - `vote react list`: List of vote reactions for the comment (e.g., like, love, angry, etc.).
        - `comment replies:` Extract replies to each comment (if available), including:
            a. `commentId`: A unique identifier for the reply.
            b. `author`: The name of the reply author.
            c. `text`: The reply text.
            d. `date`: The publication date of the reply.
            e. `vote react list`: List of vote reactions for the reply (e.g., like, love, angry, etc.).

5. **Data Storage:**
    - Save the collected data for each post in a structured format (JSON or YAML). The file should be named `<postId>.<json/yaml>` (e.g., `12345.json` or `12345.yaml`).
    - Store each post's audio file in the `./audio/` directory and images in `./images/<postId>/`.
6. **Report:**
    - Write a brief report (500-700 words) explaining your approach, the challenges you encountered, and how you addressed them.
    - Include sample output (e.g., snippets of JSON/YAML files, images and audio saved locally).

Submission:

1. **Python Code**: Submit your Python script(s).
2. **Data Files**:
    - Submit the generated JSON/YAML files in a folder named `data/`.
3. **Media Files**:
    - If the images and audio files are large:
        - Zip the `audio/` folder and upload it to Google Drive.
        - Zip the `images/` folder and upload it to Google Drive.

    Share the Google Drive links in a `media_links.txt` file with the following format:
    ```
    Audio files: <Google Drive link to zipped audio folder>
    Image files: <Google Drive link to zipped images folder>
    ```
4. **Report**: Submit your report as a PDF.
5. **Zipping Instructions**:
    - Ensure that the `data/` folder and the `media_links.txt` file are included in the final zip file for submission.

Additional Notes:

- Be mindful of `tuoitre.vn`'s `robots.txt` file. Ensure your crawler respects the rules and does not violate the website's terms of service.
- Consider using tools such as `time.sleep()` to add delays between requests, or `fake-useragent` to mimic different user agents.
- Multi-threading or asynchronous programming can be used to speed up data collection if needed.
- Handle scenarios such as pagination, missing images/audio, or posts without comments gracefully.

Evaluation Criteria:

- Correctness: Does the program retrieve and store the correct data, including saving audio and images, and handling nested comments/replies?
- Code Quality: Is the code well-organized, documented, and efficient?
- Error Handling: Can the program handle common errors and edge cases gracefully?
- Report Quality: Is the report clear, concise, and does it explain your solution well?

Resources:

- **requests** Documentation: https://requests.readthedocs.io/en/latest/
- Implementing Web Scraping in Python with **BeautifulSoup**: https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/
- **Selenium** Documentation: https://www.selenium.dev/documentation/webdriver/getting_started/

Plagiarism Policy:

- **Plagiarism is strictly prohibited.** Any form of copying or using unauthorized code will result in a score of **0** for this assignment. Please ensure all code is your own and properly cited if you use any external resources.