Google News Scraper Project Report (MLOPS ASSIGNMENT 01)

1. Overview

This project involves building a Google News scraping pipeline using Python. The implementation consists of six modules that scrape the Google News homepage, extract top stories, retrieve headlines and thumbnails, store the extracted data in a database, perform de-duplication checks, and orchestrate the entire process efficiently.

---

2. Modules

Module 1: Scrape Google News Homepage (`google_news_scraper.py`)

- Uses `requests` and `BeautifulSoup` to fetch and parse the Google News homepage.
- Ensures dynamic URL handling through a configuration file.
- Implements logging for success or failure events.

Module 2: Extract 'Top Stories' Link (`top_stories_scraper.py`)

- Parses the homepage HTML to find the ‘Top Stories’ section dynamically.
- Uses `BeautifulSoup` to extract the relevant link.
- Avoids hardcoded keywords by making it configurable.
- Logs warnings if the link is not found.

Module 3: Scrape Story Details (`story_scraper.py`)

- Extracts headlines, thumbnails, and URLs from the ‘Top Stories’ page.
- Handles lazy loading of images to retrieve complete data.
- Uses `requests` for downloading images and `BeautifulSoup` for parsing HTML.
- Stores extracted metadata with timestamps.

Module 4: Store Data in Database (`database_storage.py`)

- Supports PostgreSQL as the database backend.
- Creates tables for headlines and images if they do not exist.
- Stores extracted stories and their associated images in the database.
- Uses SQL constraints to prevent duplicate records.

Module 5: De-duplication Check (`deduplication.py`)

- Prevents duplicate entries in the database.
- Uses NLP-based similarity checks instead of strict headline matching.
- Implements `sentence-transformers` to compute semantic similarity between headlines.
- Ensures only unique and relevant stories are stored.

Module 6: Orchestration Script (`orchestrator.py`)

- Coordinates the execution of all modules sequentially.
- Implements logging to track execution timestamps, errors, and debugging information.
- Supports easy integration with CronJobs for scheduled execution.

---

3. Configuration File

The configuration file (`config.py`) defines key parameters such as:

- Base URLs and headers for HTTP requests.
- Class names for extracting headlines and images.
- **Database connection settings (Users must configure their own PostgreSQL credentials such as `host`, `port`, and `password` in `config.py` before running the script).**
- SQL statements for creating necessary tables.
- Log file location for tracking execution history.

---

4. How to Run the Scripts

Prerequisites

- Install Python 3.x
- Install required dependencies using:
- `pip install -r requirements.txt`
- Configure PostgreSQL and update `config.py` with correct database credentials.
- Install `sentence-transformers` for NLP-based similarity checks:
- `pip install sentence-transformers`

Execution Steps

1. **Run the Orchestrator Script**
2. `python orchestrator.py`

   This will execute all modules sequentially and store the extracted news data in the database.

3. **Set Up a Cron Job (Optional)**
   - Open the crontab editor:
   - `crontab -e`
   - Add the following line to run the scraper every hour:
   - `0 * * * * /usr/bin/python3 /path/to/orchestrator.py`

---

5. Database Schema

The project uses a PostgreSQL database with the following schema:

- `headlines` table:
  - `id` (Primary Key, Auto Increment)
  - `headline` (Unique Text)
  - `url` (Unique Text)
  - `scrape_timestamp` (Timestamp)
- `images` table:
  - `id` (Primary Key, Auto Increment)
  - `headline_id` (Foreign Key referencing `headlines.id` with cascading delete)
  - `thumbnail` (Binary Image Data)

---

6. Logging and Debugging

- Logs are stored in `scraper.log`.
- Each module logs critical steps and errors to facilitate debugging.
- Error handling is implemented to prevent crashes and ensure robustness.

---

7. Conclusion

This project successfully implements a robust and modular Google News scraper. It efficiently extracts, stores, and manages news data while ensuring flexibility and scalability. The pipeline can be enhanced further with additional features and optimizations for improved performance.

Submitted By: Sankalp Shrivastava DA24S021