Scraper and Video Summarizer

Prepared by

Joseph Ashraf Anwar Nasr	20p4328
Ali Amer Ibrahim Mahmoud	1900881

Project Report

Table of Contents

Overview

This report documents the development of two Python applications and their associated performance evaluation scripts:

1. **Scraper Application**
    - Automates the extraction of movie data from IMDb.
    - Stores the data in a MySQL database.
2. **Video Summarizer Application**
    - Analyzes movie trailers to detect engaging scenes.
    - Generates summary videos with captions based on detected scenes.

Repository Details

A GitHub repository hosts all related code, scripts, and instructions. The repository structure includes:

- **Scripts**:
    - `Scraper.py`: Scrapes movie data and populates the database.
    - `video_summarizer.py`: Summarizes trailers by detecting key scenes and adding captions.
    - `performance_evaluation.py`: Evaluates the performance of the scraper.
    - `video_summarizer_evaluation.py`: Evaluates the performance of the video summarizer.

- **Documentation**:
  - `README.md`: Provides setup instructions and details for replicating the work.
- **Dependencies**:
  - `requirements.txt`: Lists required Python packages.

Instructions for Replication

1. **Clone the Repository**:

   ```
   git clone <repository_url>
   cd <repository_directory>
   ```

2. **Install Dependencies**:

   ```
   pip install -r requirements.txt
   ```

3. **Set Up Database**:
   - Create a MySQL database.
   - Update database credentials in `Scraper.py` and `video_summarizer.py`.
   - Initialize the `movies` table using the schema in `Scraper.py`.
4. **Run Applications**:
   - Execute `Scraper.py` to populate the database.
   - Run `video_summarizer.py` to generate trailer summaries.
5. **Evaluate Performance**:
   - Execute `performance_evaluation.py` and `video_summarizer_evaluation.py` to log performance metrics.

Key Metrics

Scraper Performance

- Execution time for various scraping tasks.
- Memory and CPU usage during data extraction.

Video Summarizer Performance

- Time taken to detect scenes and generate summaries.
- Resource usage during video processing.

Results and Discussion

Achievements

- Efficiently scraped and stored movie data, including titles, release dates, ratings, and trailers.
- Generated concise and engaging video summaries with captions.

Challenges

- Optimizing resource usage for large-scale video processing.
- Handling diverse video formats and metadata extraction.

Repository Link

Access the GitHub repository here: https://github.com/aliamer0/SPE

Conclusion

This project demonstrates the integration of web scraping and video summarization using Python. Future improvements could include advanced error handling and support for additional data sources