News Article Categorization System

Objective

The objective of this project is to build an application that fetches news articles from a set of predefined RSS feeds, stores them in a relational database, and classifies them into predefined categories based on the content. This application will allow the user to easily monitor news updates and categorise articles into four categories:

- **Terrorism / Protest / Political Unrest / Riot**
- **Positive / Uplifting**
- **Natural Disasters**
- **Others**

System Architecture Overview

The application is designed as a modular system consisting of three main components:

1. RSS Feed Parser and Data Extraction: Collects and parses data from multiple RSS feeds.
2. Database Storage: Stores the fetched news articles in a relational database while avoiding duplicate entries.
3. Asynchronous Task Queue: Manages the processing of news articles for categorization via a task queue (using Celery).
4. Categorization Engine: Utilises machine learning model and Natural Language Processing (NLP) to classify the articles into one of the four categories.

Tech Stack

- Programming Language: Python
- Libraries:
    - Feedparser for reading and parsing RSS feeds.
    - SQLAlchemy for database interaction with MySQL.
    - Celery for managing asynchronous tasks.
    - NLTK (Natural Language Toolkit) or spaCy for natural language processing and classification.
    - Scikit-learn for training the ml model
    - Pandas for data manipulation
    - Joblib for dumping the model
    - Re (regular expression) for extracting the relevant text
- Database: MySQL
- Task Queue: Celery with Redis as the broker.

Detailed Components

1. **RSS Feed Parser and Data Extraction**

The RSS Feed parser collects articles from multiple RSS feeds, processes them, and extracts relevant details such as title, content, publication date, and source URL. The Feedparser library is used for parsing the RSS feed data.

Data Extraction Logic:

- The RSS feed URLs are provided in a list. For each feed, the system parses the feed and extracts relevant fields like title, summary, published, and link.
- It extracts the content from the summary using the regular expression for removing the image url and image text.
- Proper logging have been added
- The system ensures no duplicate articles are fetched by checking for duplicate titles or URLs before saving them to the database.

Example of RSS feed URLs used:

http://qz.com/feed
http://feeds.foxnews.com/foxnews/politics

2. **Database Storage**

The extracted news articles are stored in a relational database - MySQL. Each article's details, including title, content, publication date, and source, are stored in the database. SQLAlchemy is used as the ORM (Object Relational Mapper) to handle database operations efficiently.

Database Schema:

Table: news_articles

- **id**: Auto incremented primary key
- **title**: The title of the news article (unique constraint to avoid duplicates)
- **content**: The content or summary of the article
- **published:** The publication date of the article
- **source**: The URL source of the article
- **category**: The category assigned to the article based on its content (NULL until processed)

Database Duplication Handling:
Before saving an article, the system checks for duplicate entries based on the title and URL. If a duplicate is found, the system skips inserting the article into the database.

3. **Asynchronous Task Queue and Processing**

A Celery task queue is implemented to handle the processing of articles asynchronously. Once new articles are fetched and stored in the database, they are sent to the task queue for further processing. This decouples the feed parsing from the computationally intensive categorization process.

Celery Worker:
A Celery worker listens for new tasks that involve processing uncategorized news articles. Each article is processed individually and categorised based on its content.

Task Queue Logic:
Articles that have not yet been categorised are pushed into a queue. The worker then fetches these articles from the queue, processes them, and assigns them a category based on NLP classification.

4. Categorization Engine Using NLP

The categorization of articles is done using Natural Language Processing (NLP) techniques. The application uses NLTK or spaCy to classify each article into one of the four predefined categories based on specific keywords or patterns present in the text.

Machine Learning Approach :
For more complex classification, a machine learning based model could be implemented using training data and supervised learning techniques (e.g., using TF IDF with Naive Bayes or Logistic Regression).

Logging and Error Handling

Logging is implemented throughout the application to track events and errors during processing. This includes logging for:

- Successful RSS feed parsing.
- Successful database insertions and categorizations.
- Errors during feed parsing, database connections, and NLP classification.

Additionally, Celery's error handling mechanism is used to retry tasks in case of network errors or database connectivity issues. Logging is enabled for every Celery task execution to ensure traceability.

How to Run the Application

Prerequisites:

- Python 3.8+
- MySQL database setup
- Redis for Celery task queue
- Required Python libraries (feedparser, SQLAlchemy, Celery, nltk, spaCy, re, pandas, scikit-learn, joblib)

Installation:

1. **Clone the repository:**
   bash
   git clone https://github.com/yourrepo/newsaggregator.git
   cd news aggregator

2. **Install the required Python packages:**
   bash
   pip install r requirements.txt

3. **Set up the database:**
Create a MySQL database named
Update the database connection string in the configuration file (models.py) to point to your database.

4. **Start Redis:**
   bash
   Redis server

5. **Run the Celery worker:**
   Start the Celery worker to process tasks:
   celery -A tasks worker --loglevel=info -P solo

6. **Run the feed parser**:
   Execute the feed parser to fetch and process RSS feeds:

   python feed_parser.py

Future Enhancements

Machine Learning Classifier: Improve the categorization by fine tuning the given machine learning based classifier to categorise articles with higher accuracy.

Web Interface: Create a simple dashboard to view categorised articles and control the feed parsing process.

Automatic Scheduling: Implement automatic scheduling to periodically fetch new articles from RSS feeds at regular intervals using Celery beat or a cron job.

Conclusion

This application provides a comprehensive solution to automatically collect, store, and categorise news articles from RSS feeds. By combining RSS feed parsing with asynchronous task processing and natural language processing, the system efficiently processes and categorises large amounts of data while minimising the load on the system. The modular design ensures that it can be easily extended and enhanced with additional functionality, such as advanced machine learning models or a web based user interface.

Code Flow

There are many python files which are used for implementing the logic given in architecture. Let's take a look one by one:-

1. Feed_parser.py:
   It loads the feed from the url, parses then ,extracts the relevant content text and other details from the feed using regex and then saves the data into the sql database

2. models.py:
   It contains a basic configuration for the connection to the sql database.

3. task.py:
   It listens to the asynchronized task coming to the celery which involves processing uncategorized news articles based on the given categories using the ml model and saves the categories again to the sql.

4. ml_model.py:
   This was used to train the model which can categorise the data into given categories. The model was saved which was used in task.py. The model has been trained by extracting the sample data from the feed and labelling them based on their categories.

5. logger.py:
   It is used for properly logging the error messages and the events.