News Collector Application: Documentation

Overview

The News Collector Application is designed to:

1. Collect news articles from various RSS feeds.
2. Parse and extract essential information from each article.
3. Store the articles in a database, ensuring no duplicates.
4. Classify the articles into predefined categories using machine learning techniques (NLTK/spaCy).
5. Process articles asynchronously using a Celery task queue.
6. Implement logging and error handling to manage potential issues.

This document outlines the logic and design choices made while implementing the application, followed by a brief assessment of the ETL pipeline, task scheduling infrastructure, and data handling abilities.

Application Flow

1. Feed Parsing and Data Extraction

- **Logic**:
  - The application reads a list of RSS feeds using the `feedparser` library.
  - Each feed is parsed to extract the `title`, `content`, `published date`, and `source URL`.
  - If an article's content is not provided directly in the feed, the `newspaper3k` library is used to download and parse the article content.
- **Design Choice**:
  - `feedparser` was chosen for its simplicity in handling RSS feeds.

- `newspaper3k` was used for scraping full article content when not provided in the feed, ensuring completeness in the extracted data.

2. Database Storage

- **Logic**:
  - An SQLite database is used to store the articles in a table with columns for title, content, publication date, source, URL (unique), and category.
  - The uniqueness of each article is ensured via the URL field. An `IntegrityError` is caught to handle duplicate articles gracefully.
- **Design Choice**:
  - SQLite was selected for ease of setup, as this is a lightweight database suitable for local storage needs. In a production environment, this could easily be swapped out for a more scalable database such as PostgreSQL or MySQL.

3. Task Queue and News Processing

- **Logic**:
  - Celery is used to set up a task queue for processing articles asynchronously.
  - When new articles are parsed, they are queued using the Celery `.delay()` method, allowing processing (categorization and storage) to happen in the background.
  - A Celery worker continuously listens for new tasks, consumes them, categorizes the articles, and stores them in the database.
- **Design Choice**:
  - Celery, combined with Redis as the message broker, was selected to handle task queuing and asynchronous processing. This ensures that article processing does not block feed parsing and supports scaling the system for larger workloads.

4. Article Categorization

- **Logic**:
  - Each article is categorized based on its content using predefined categories (e.g., "Politics," "Technology," "Health"). Keywords associated with each category are checked within the article content.
  - Alternatively, machine learning tools such as NLTK or spaCy could be used for advanced text processing.
- **Design Choice**:
  - NLTK/spaCy was chosen for their ability to perform natural language processing, allowing the application to classify articles based on context and keywords, and to provide a simple classification solution for this project.

5. Logging and Error Handling

- **Logic**:
  - Logs are generated throughout the application to track events, such as successful feed parsing, task queuing, and database updates.
  - Error handling is implemented for network issues, feed parsing errors, database integrity errors (e.g., duplicate articles), and article scraping issues.
- **Design Choice**:
  - Logging helps with debugging and monitoring the application in real-time. Errors are handled gracefully to ensure the application continues running even when issues occur.

Assessment of Abilities

Building and Managing Complex ETL Pipelines

- **ETL Overview**: The application essentially builds an ETL (Extract, Transform, Load) pipeline:
  - **Extract**: Articles are extracted from RSS feeds.

  - **Transform**: The articles are processed, categorized, and enriched with additional metadata.
  - **Load**: Articles are stored in a database.
- **Complexity**: The pipeline can handle a variety of sources (RSS feeds) and includes data cleaning (duplicate handling), text processing (categorization), and error handling. Adding additional feeds or using other data sources like APIs would require minimal modification to the existing pipeline.

Building Scheduling and Distributed Task Management Infrastructure

- **Scheduling**: The application uses Celery to schedule and manage distributed tasks. Article processing tasks are queued and processed asynchronously, allowing for high scalability and parallelism.
- **Task Management**: By leveraging Celery with Redis as the message broker, the application can easily handle distributed workloads. Celery's task management system makes it easy to monitor and retry failed tasks, improving fault tolerance.

Working with Data Sources (RSS, APIs, etc.)

- **Data Sources**: The application demonstrates proficiency in working with multiple data sources:
  - RSS feeds are handled via `feedparser`, which efficiently parses the XML data format.
  - Full article extraction is handled using the `newspaper3k` library, which scrapes and processes article content from web pages.
- **Extendability**: The structure allows for the addition of more data sources (APIs, other feeds) with minimal adjustments.

Implementing Existing Machine Learning

- **Categorization**: Basic text classification is implemented using keywords, but the application is designed to support integration with more advanced machine learning libraries such as NLTK and spaCy.

    - For more sophisticated use cases, text classification models can be trained to categorize articles into more complex or granular categories.
- **ML Readiness**: The current implementation is well-suited for deploying more advanced machine learning models, such as topic modeling (Latent Dirichlet Allocation) or text classification using supervised learning methods.

Conclusion

The News Collector Application efficiently processes data from multiple sources, categorizes it using both keyword-based logic and machine learning tools, and manages tasks asynchronously using Celery. It demonstrates solid capabilities in managing complex ETL pipelines, distributed task scheduling, and working with various data sources.