RSS Feed Processor and Classifier

Comprehensive Documentation

Table of Contents

---

1. Introduction

The RSS Feed Processor and Classifier is a Python-based application designed to collect news articles from various RSS feeds, store them in a MySQL database, and categorize them into predefined categories using machine learning techniques. This document provides a comprehensive guide to understanding, setting up, and running the application.

1.1 Project Objectives

- Collect news articles from multiple RSS feeds
- Store article data in a structured database
- Categorize articles into predefined categories
- Process articles asynchronously for improved performance
- Provide a scalable and maintainable solution

1.2 Technology Stack

- Python 3.8+
- MySQL
- SQLAlchemy (ORM)
- Celery (Task Queue)
- Redis (Message Broker)
- Scikit-learn (Machine Learning)
- NLTK (Natural Language Processing)

---

2. System Architecture

The application follows a modular architecture with the following key components:

1. **RSS Feed Parser**: Fetches and parses RSS feeds
2. **Database Layer**: Manages data storage and retrieval
3. **Task Queue**: Handles asynchronous processing of articles
4. **Classification Engine**: Categorizes articles using machine learning

[RSS Feeds] -> [Feed Parser] -> [Database] <- [Task Queue] <- [Classification Engine]

---

3. Installation and Setup

3.1 Prerequisites

- Python 3.8 or higher
- MySQL Server
- Redis Server

3.2 Environment Setup

1. Create a virtual environment:
2. python -m venv rss_env

source rss_env/bin/activate # On Windows: rss_env\Scripts\activate

3. Install required packages:

pip install feedparser sqlalchemy mysql-connector-python celery redis scikit-learn nltk

3.3 Database Setup

1. Create a MySQL database:
2. **CREATE DATABASE** rss_processor;
3. Update the DB_URL in main.py with your database credentials:
4. DB_URL = "mysql+mysqlconnector://username:password@localhost/rss_processor"

3.4 Redis Setup

Ensure Redis is installed and running on the default port (6379).

---

4. Code Structure

The application consists of a single Python script (main.py) with the following structure:

- Imports and configuration

- Database model definition
- Celery task queue setup
- RSS feed parsing function
- Article processing task
- Classification logic
- Main execution function

---

5. Key Components

5.1 Database Model

The Article class defines the schema for storing articles:

```python
class Article(Base):
    __tablename__ = 'articles'
    id = Column(Integer, primary_key=True)
    title = Column(String(255))
    content = Column(Text)
    pub_date = Column(DateTime)
    source_url = Column(String(255), unique=True)
    category = Column(String(50))
```

5.2 RSS Feed Parsing

The parse_feed function fetches and processes RSS feeds:

```python
def parse_feed(feed_url):
    # Fetch and parse feed
    # Store new articles in database
    # Queue articles for processing
```

5.3 Asynchronous Processing

Celery is used for asynchronous article processing:

```python
@app.task(bind=True, default_retry_delay=300, max_retries=5)
def process_article(self, article_id):
    # Retrieve article from database
    # Classify article
    # Update database with classification
```

5.4 Classification Engine

A scikit-learn pipeline is used for article classification:

```
pipeline = Pipeline([
   ('tfidf', TfidfVectorizer(stop_words=stopwords.words('english'))),
   ('clf', MultinomialNB()),
])
```

---

6. Running the Application

1. Start the Celery worker:

```
celery -A main worker --loglevel=info
```

2. In a separate terminal, run the main script:

```
python main.py
```

The application will start processing RSS feeds, storing articles, and classifying them asynchronously.

---

7. Maintenance and Troubleshooting

7.1 Logging

The application uses Python's logging module. Logs are written to app.log:

```
logging.basicConfig(filename='app.log', level=logging.INFO,
                    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')
```

7.2 Common Issues

1. **Database Connection Errors**: Ensure MySQL is running and credentials are correct.
2. **Celery Worker Not Starting**: Check if Redis is running and accessible.
3. **Classification Errors**: Verify that the NLTK data is downloaded correctly.

---

8. Future Enhancements

1. Implement a web interface for monitoring and management
2. Enhance the classification model with a larger, domain-specific dataset
3. Add support for full-text article extraction
4. Implement periodic scheduling for RSS feed checks
5. Develop a comprehensive test suite for improved reliability

---

9. Conclusion

The RSS Feed Processor and Classifier provides a robust solution for automating the collection and categorization of news articles. By leveraging asynchronous processing and machine learning, it offers a scalable approach to handling large volumes of data from multiple sources.

In [ ]: