DATA INTERN ASSIGNMENT

News Article Classification Project

Objective

The objective of this project is to build an application that collects news articles from various RSS feeds, stores them in a database, and categorizes them into predefined categories:

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

RSS Feeds Used

- http://rss.cnn.com/rss/cnn_topstories.rss
- http://qz.com/feed
- http://feeds.foxnews.com/foxnews/politics
- http://feeds.reuters.com/reuters/businessNews
- http://feeds.feedburner.com/NewshourWorld
- https://feeds.bbci.co.uk/news/world/asia/india/rss.xml

Programming Language

- **Python**

Libraries Used

- **feedparser**: For parsing RSS feeds.
- **SQLAlchemy**: For database interaction.
- **Celery**: For task queue management.
- **NLTK/spaCy**: For natural language processing and text classification.

Step-by-Step Implementation

1. Setting Up the Environment

Installing Required Libraries:

pip install feedparser sqlalchemy psycopg2 celery nltk spacy

2. Data Extraction Using Feedparser

```
import feedparser
rss_feeds = [
    "http://rss.cnn.com/rss/cnn_topstories.rss",
    "http://qz.com/feed",
    "http://feeds.foxnews.com/foxnews/politics",
    "http://feeds.reuters.com/reuters/businessNews",
    "http://feeds.feedburner.com/NewshourWorld",
    "https://feeds.bbci.co.uk/news/world/asia/india/rss.xml"
]
def fetch_articles(feed_url):
    articles = []
    feed = feedparser.parse(feed_url)
    for entry in feed.entries:
        article = {
            'title': entry.title,
            'link': entry.link,
            'published': entry.published,
            'summary': entry.summary
        }
        articles.append(article)
    return articles
all_articles = []
for feed in rss_feeds:
    all_articles.extend(fetch_articles(feed))
print("Fetched Articles: ", all_articles)
```

3. Database Design and Storage

Using **SQLAlchemy** to define the database schema and manage article storage.

```
from sqlalchemy import create_engine, Column, String, Integer, DateTime
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

Base = declarative_base()
class NewsArticle(Base):
    __tablename__ = 'news_articles'
    id = Column(Integer, primary_key=True)
    title = Column(String)
    link = Column(String, unique=True)
    published = Column(DateTime)
    summary = Column(String)
# Database setup
engine = create_engine('postgresql://username:password@localhost/news_db')
Base.metadata.create_all(engine)
Session = sessionmaker(bind=engine)
session = Session()
def store_articles(articles):
    for article in articles:
        if not session.query(NewsArticle).filter_by(link=article['link']).first():
            new_article = NewsArticle(**article)
            session.add(new_article)
    session.commit()
store_articles(all_articles)
```

4. Text Classification Using NLTK or spaCy

Setting up the Natural Language Processing (NLP) pipeline for classification.

NLTK Setup:

```
import nltk
```

```
nltk.download('punkt')
nltk.download('stopwords')
```

Using spaCy for text classification:

```
import spacy
nlp = spacy.load("en_core_web_sm")
def classify_article(text):
    doc = nlp(text)
    # Dummy classification logic, to be replaced with a trained model
    if "protest" in text or "riot" in text:
        return "Terrorism / protest / political unrest / riot"
    elif "happy" in text or "success" in text:
        return "Positive/Uplifting"
    elif "earthquake" in text or "flood" in text:
        return "Natural Disasters"
    else:
        return "Others"
```

5. Task Queue Management Using Celery

Setting up Celery for asynchronous task processing.

```
from celery import Celery
app = Celery('news_tasks', broker='redis://localhost:6379/0')
@app.task
def process_new_articles():
    articles = fetch_articles()
    for article in articles:
        category = classify_article(article['summary'])
        article['category'] = category
        store_articles([article])
```

6. Error Handling and Logging

Adding logging for better traceability.

```
import logging
logging.basicConfig(level=logging.INFO)
def fetch_articles_with_logging(feed_url):
    try:
        return fetch_articles(feed_url)
    except Exception as e:
        logging.error(f"Failed to fetch articles from {feed_url}: {e}")
```

Deliverables

- **Python Code**: The full code for each component (RSS parsing, database storage, NLP classification, task queue setup).
- **Documentation**: Detailed explanation of each step, the libraries used, and the design choices made.
- **Data Dump**: Resulting data in SQL, CSV, or JSON format (based on your preference).

Conclusion

This project showcases the use of data pipelines, asynchronous task management, and machine learning for categorizing news articles. It provides a solid base for further enhancements like using advanced NLP models and integrating new data sources.