Intelligent Book Management & Recommendation System

---

1. Overview

The Intelligent Book Management System is a Python-based application designed to manage a PostgreSQL database of books, generate summaries for books, and recommend books based on user preferences. The application has been developed to run locally and deploy on AWS, ensuring scalability and accessibility.

Key Components:

- **BART Model Integration**: For generating summaries.
- **Collaborative Filtering and Content-Based Filtering**: For book recommendations.
- **Summarization Technique**: Summarization of summarization to handle large documents.
- **RESTful API**: For managing books, reviews, and generating recommendations.

2. System Components

1. **Database Setup**:
    - **PostgreSQL Database**: Stores book information, including titles, authors, genres, publication dates, and summaries.
    - **Tables**:
        - `books`: Contains fields `id`, `title`, `author`, `genre`, `year_published`, and `summary`.
        - `reviews`: Contains fields `id`, `book_id` (foreign key referencing `books`), `user_id`, `review_text`, and `rating`.
2. **BART Model for Summarization**:
    - **Model**: The BART model replaces Llama3 for generating book summaries due to system resources.
    - **Technique**: Implemented a "summarization of summarization" approach where large documents are chunked into smaller parts. Each part is summarized individually, and these summaries are then summarized further to generate a final concise summary. This technique ensures efficient processing of large texts within model constraints.

3. Recommendation Engine: Detailed Explanation

The recommendation engine in this system combines **Collaborative Filtering** and **Content-Based Filtering** to suggest books tailored to a user's preferences. Here's how the engine works, followed by the provided code that implements these techniques.

Collaborative Filtering

- **Objective**: Recommend books based on the collective user behaviour, particularly how users with similar tastes rate books.
- **User-Item Interaction Matrix**: The engine constructs a matrix where rows represent users, columns represent books, and the cells contain the ratings given by users to books. This matrix allows the engine to understand the relationship between users and their preferences.
- **Cosine Similarity**: The similarity between users is calculated using cosine similarity, which measures how close the user's ratings are to others in terms of the angle between their rating vectors. A smaller angle (closer to 1) means users have similar preferences.

Content-Based Filtering

- **Objective**: Recommend books based on the characteristics (e.g., genre) of books the user has previously liked.
- **TF-IDF Vectorization**: The genre of each book is converted into a numerical representation using Term Frequency-Inverse Document Frequency (TF-IDF), which helps in comparing the content (genre) across different books.
- **Cosine Similarity**: Similar to user-based filtering, cosine similarity is used to calculate how similar the genres of different books are.

Integration of Both Methods

- **Weighted Ratings**: The ratings are adjusted by taking into account both the user similarity and the genre similarity. The weighted ratings are then used to rank books.
- **Filtering**: Books that the user has already bought or reviewed are filtered out to ensure only new recommendations are presented.

4. Instructions to Run the Application

1. **Setup and Installation**:
   - **Instal Database**:
     i. `Install Postgresql database`
   - **Install python Dependencies**:
     i. `pip install -r requirements.txt`
2. **Database Configuration**:
   a. Configure PostgreSQL database with the following details:
      - Database Name: `books_db`
      - `Keep all the details in .env file like username, password, host, port and database name`

3. **Running the FastAPI Server**:
    a. Ensure the PostgreSQL server is running.
    - Open the main folder where you can see requirements.txt, main.py and other files available. open the terminal and follow the below instructions
    - **pip install -r requirements.txt**
    - **uvicorn main:app –reload**
4. **Accessing the API**:
    a. There is a separate document which consists of api signature and fields to run the client code.
    b. You can find the client code in **test_cases_and_data**
    c. Open the folder & follow the below instructions
    d. python / python3 user_reg_client.py – **it will register the users**.
    e. python / python3 client_books_summary.py - **it will push the summarised books data into DB**
    f. python / python3 client_review.py – **It will push the summarised reviews data into the DB**
    g. python / python3 client_recommendation.py – **Based on User, it will recommend the other books.**

5.Conclusion

This document provides a comprehensive guide to the Intelligent Book Management System, focusing on the techniques used and detailed instructions for setting up the environment and running the API. The system leverages advanced AI models for summarization and recommendation, ensuring efficient and accurate results. The provided instructions will enable you to successfully implement and use the system in your local environment.