Book Recommendation System Documentation

Overview:

This project is a **Book Recommendation System** that uses collaborative filtering to suggest books to users based on their reading preferences. The system is built using the **Nearest Neighbors** algorithm, which identifies books that are similar to the ones a user has already read or liked. The dataset used in this project contains information about books, users, and their ratings.

The goal of this project is to provide personalized book recommendations by analyzing user ratings and identifying patterns in the data. The system recommends books that are similar to a given book, based on the ratings provided by users who have read both books.

Dataset:

The project uses three main datasets:

1. **Books.csv**: Contains information about books, including:
   - ISBN: Unique identifier for each book.
   - Book-Title: Title of the book.
   - Book-Author: Author of the book.
   - Year-Of-Publication: Year the book was published.
   - Publisher: Publisher of the book.
   - Image-URL-S, Image-URL-M, Image-URL-L: URLs for small, medium, and large images of the book (not used in the analysis).
2. **Ratings.csv**: Contains user ratings for books, including:
   - User-ID: Unique identifier for each user.
   - ISBN: Unique identifier for each book.
   - Book-Rating: Rating given by the user (on a scale of 0 to 10).
3. **Users.csv**: Contains information about users, including:
   - User-ID: Unique identifier for each user.
   - Location: Location of the user.
   - Age: Age of the user (some values are missing).

Data Preprocessing:

1. Loading the Dataset

- The datasets are loaded using pandas.read_csv().
- The Books.csv dataset contains 271,360 entries, while the Ratings.csv dataset contains 1,149,780 entries, and the Users.csv dataset contains 278,858 entries.

2. Data Cleaning

- Unnecessary columns (Image-URL-S, Image-URL-M, Image-URL-L) are dropped from the Books.csv dataset.
- The Ratings.csv dataset is filtered to include only users who have rated at least 200 books. This reduces the dataset to 526,356 entries.
- The Ratings.csv dataset is merged with the Books.csv dataset based on the ISBN column to combine book information with user ratings.

3. Filtering Books with High Ratings

- Books with fewer than 50 ratings are removed to ensure that only popular books are considered for recommendations. This reduces the dataset to 61,853 entries.
- Duplicate entries (where the same user has rated the same book multiple times) are removed, resulting in a final dataset of 59,850 entries.

4. Creating a Pivot Table

- A pivot table is created where rows represent books, columns represent users, and values represent the ratings given by users to books. This table is used to calculate similarities between books.
- Null values in the pivot table are replaced with 0 to handle missing ratings.

5. Using Sparse Matrix

- To reduce memory usage and improve performance, the pivot table is converted into a **sparse matrix** using scipy.sparse.csr_matrix.

Model Building:

1. Nearest Neighbors Algorithm

- The **Nearest Neighbors** algorithm is used to find books that are similar to a given book. The algorithm calculates the distance between books based on user ratings.
- The model is trained using the sparse matrix representation of the pivot table.

2. Model Training

- The model is trained using the NearestNeighbors class from sklearn.neighbors with the algorithm='brute' parameter, which calculates the distance between all pairs of books.

Recommendation System:

1. Input Data

- The system takes a book title as input and finds the index of the book in the pivot table.
- The input data is reshaped into a format suitable for the model.

2. Finding Similar Books

- The model uses the kneighbors() method to find the 6 closest books to the input book based on user ratings.
- The distances and indices of the closest books are returned.

3. Output Recommendations

- The system outputs the titles of the 5 most similar books (excluding the input book itself).

Recommend Books for a Given Title

- The function recommend_books(bookname) is used to get recommendations for a specific book.

Key Features:

1. **Collaborative Filtering**: The system uses collaborative filtering to recommend books based on user ratings.
2. **Popularity Filtering**: Only books with at least 50 ratings are considered for recommendations, ensuring that the recommendations are based on popular books.
3. **Sparse Matrix**: The use of a sparse matrix reduces memory usage and improves the efficiency of the model.
4. **Personalized Recommendations**: The system provides personalized recommendations based on the similarity between books.

Dependencies:

- **Python Libraries**:
  - pandas
  - numpy
  - scipy
  - scikit-learn

How to Run the Code:

1. **Install Dependencies**:
   Ensure that all the required libraries are installed. You can install them using pip:

   ```
   pip install pandas numpy scipy scikit-learn
   ```

2. **Load the Dataset**:
   Place the Books.csv, Ratings.csv, and Users.csv files in the same directory as the notebook or script.
3. **Run the Code**:
   Execute the code cells in the provided Jupyter Notebook (main.ipynb) to preprocess the data, train the model, and generate recommendations.
4. **Generate Recommendations**:
   Use the recommend_books(bookname) function to get book recommendations for a specific book title.

Conclusion:

This Book Recommendation System is a simple yet effective tool for suggesting books to users based on their reading preferences. By analyzing user ratings and identifying patterns in the data, the system can provide personalized recommendations that are likely to be of interest to the user. The system can be further improved by incorporating additional features, such as user-based recommendations and content-based filtering, to enhance the accuracy and relevance of the recommendations.

References:

- **Scikit-learn Documentation**: https://scikit-learn.org/stable/
- **Pandas Documentation**: https://pandas.pydata.org/pandas-docs/stable/
- **NumPy Documentation**: https://numpy.org/doc/
- **SciPy Documentation**: https://docs.scipy.org/doc/scipy/