**Book Recommendation System**

**Submitted for**

**Statistical Machine Learning CSET211**

Submitted by:

**(E23CSEU1263) HARSHABAD SINGH**

Submitted to

**SANCHALI DAS**

**July-Dec 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Content** | **Page No.** |
|  | **INTRODUCTION** | 1 |
|  | **LITERATURE REVIEW** | 1 |
|  | **DATA DISCRIPTION** | 2 |
|  | **METHODOLOGY** | 2 |
|  | **FLASK API IMPLEMENTATION** | 3 |
|  | **RESULTS** | 3 |
|  | **CHALLENGES AND LIMITATION** | 4 |
|  | **CONCLUSION** | 5 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |

**Introduction**

**Overview of the Project:**

* This project aims to build a book recommendation system that suggests books to users based on their preferences, past ratings, and the ratings of similar users. The system utilizes collaborative filtering, a popular technique in recommendation systems.
* The project is implemented using Python libraries like Pandas, NumPy, Matplotlib, Seaborn, and Flask for the API.

**Objective:**

* The goal is to create a system where users can receive personalized book recommendations. By analyzing the book ratings and user interaction data, the system will suggest books that a user might like based on the preferences of users with similar reading tastes.
* The system is designed to scale with increasing data and can be integrated into applications for book-related services or platforms.

**Literature Review**

**Recommendation Systems**:

Recommendation systems are widely used in e-commerce, entertainment, and media platforms. These systems help users discover new content or products that they might not have found otherwise.

**Collaborative Filtering**: This is one of the most common techniques used in recommendation systems. Collaborative filtering can be divided into:

**User-based collaborative filtering**: This approach suggests items by finding similar users. If users A and B have similar ratings for most items, the system will recommend books that user B has rated highly to user A.

**Item-based collaborative filtering**: In this approach, the system suggests items based on their similarity to items that a user has already rated highly.

**Content-Based Filtering**: This method recommends items that are like those the user has liked before, based on item attributes.

**Tools and Libraries Used**:

**Pandas**: For data manipulation and analysis, including merging datasets, cleaning, and aggregating data.

**NumPy**: For numerical operations, especially during the computation of correlations and aggregations.

**Matplotlib/Seaborn**: For data visualization to present insights and the distribution of ratings.

**Flask**: To build the API that delivers book recommendations to users in real-time.

**Scikit-learn**: For additional data manipulation and model evaluation (e.g., calculating similarity).

**StatsModels**: Used for statistical testing, including testing for normality and hypothesis testing.

**SciPy**: Used for statistical tests like t-tests and Mann-Whitney tests.

**Data Description**

**Dataset Overview:**

Books Dataset: Contains information about the books, including title, author, genre, and ISBN.

Ratings Dataset: Contains user ratings for the books, including User-ID, ISBN, and the rating given by the user.

Users Dataset: Contains user information such as User-ID, location, and age.

**Data Preprocessing:**

Missing Values: The datasets were checked for missing values, and appropriate steps were taken to handle them (e.g., dropping rows with missing ratings or filling them with mean values).

Merging Datasets: The Books, Ratings, and Users datasets were merged to create a consolidated data frame that contains the relevant information for analysis.

Filtering Books with Low Ratings: Books with fewer than 200 ratings were excluded from the recommendation process, as they may not provide enough data for meaningful recommendations.

**Exploratory Data Analysis (EDA):**

Rating Distribution: Analyzed the distribution of ratings to understand how users rate books and identify any biases in rating patterns (e.g., most ratings being 8 or 9).

Top Books and Users: Identified the most popular books and the users who rated the most books. This helps to understand the most engaged users and books in the dataset.

**Methodology**

**Data Merging:**

The Books, Ratings, and Users datasets were merged using the ISBN and User-ID columns to create a comprehensive data frame that contains information on both books and users.

**Collaborative Filtering Approach:**

A user-item matrix was created where the rows represent users and the columns represent books. The values in this matrix represent the ratings that users have given to books.

Finding Similar Users: We calculated the similarity between users using Pearson correlation or cosine similarity. The idea was that users who have similar tastes in books would be likely to rate books similarly.

**Book Recommendation Logic:**

Once similar users are identified, the system recommends books that have been rated highly by these similar users but not yet rated by the target user.

The recommendations are weighted by how similar the user is to the others. Users with higher similarity scores contribute more to the weighted rating for a book.

**Flask API Implementation**

**API Structure:**

The Flask application was structured into various routes, with a primary route for generating recommendations for a user.

The application listens for requests with a user ID, processes the data, and returns the recommended books.

**API Functionality:**

**GET /recommendations**: This endpoint takes a user ID as input and returns the top 5 book recommendations based on the collaborative filtering algorithm.

**GET /similar\_users**: This endpoint provides a list of similar users to a given user.

**Integration with the Data Model:**

The recommendation system, implemented using Pandas, was integrated into the Flask routes. When a request is made, the corresponding function queries the user-item matrix, calculates similarities, and returns book recommendations.

**Testing:**

The Flask API was tested with various users to ensure the recommendations were accurate. Unit tests were written for each route to check for correct behavior and edge cases (e.g., what happens when a user has not rated enough books).

**Results**

**Evaluation of Recommendations:**

The system successfully recommended books for users based on the ratings of similar users. Example outputs were presented where the top recommended books were shown for a specific user.

**Visualizations:**

**Rating Distribution**: A histogram was plotted to show how ratings were distributed across books. Most books had ratings between 7 and 10.

**Top Rated Books**: A bar plot showing the average ratings per book helped visualize the highest-rated books.

**Challenges and Limitations**

**Data Issues:**

Missing data in user ratings, especially for books with low engagement, was a major challenge. To handle this, books with fewer than 200 ratings were excluded.

**Model Limitations:**

**Cold Start Problem**: New books or users with limited ratings make it difficult to recommend accurately. A content-based filtering model could help in such cases.

**Scalability**: As the dataset grows, calculating user similarities for all pairs becomes computationally expensive.

**API Performance:**

Handling large numbers of simultaneous requests and ensuring fast response times was challenging, and further optimization of the API might be necessary.

**Conclusion**

**Summary of Findings:**

The book recommendation system based on collaborative filtering performed well in recommending books based on user similarities. The Flask API provides real-time book recommendations for users based on their past ratings.

**Future Work:**

Exploring hybrid recommendation systems (combining collaborative and content-based filtering).

Implementing matrix factorization methods such as Singular Value Decomposition (SVD) to improve recommendation accuracy.

Scaling the system to handle larger datasets and providing more personalized recommendations.