Book Recommendation System

INTRODUCTION

Our recommendation system is a **content-based model** designed specifically for **book recommendations**.

By utilizing a content-based approach, our system proposes books to users based on their **past engagement with similar items.**

The system analyzes **the content and attributes of the books,** such as genre, author, plot, and writing style, to identify similarities and provide personalized recommendations.

By considering the user's reading **history and preferences**, the system aims to suggest books that align with their individual tastes and interests.

Through the utilization of content-based filtering, our recommendation system strives to **enhance the reading experience** by offering relevant and engaging book suggestions tailored to each user's unique preferences.

OUR DATASET

SOURCE : GOOGLE BOOKS	Book's Name, ID, Author, Language, Category, Average Rate, Maturity, Publisher, Published Date and Page Count.

The dataset used for our book recommendation system is sourced from **Google Books**. Google Books provides a comprehensive collection of book information, including various attributes and metadata that are valuable for our recommendation system.

The dataset consists of a wide range of features related to books, allowing for detailed analysis and recommendation generation. The features included in the dataset are as follows:

1. **Book Name:** The title of the book.

2. **ID**: A unique identifier for each book in the dataset.
3. **Author**: The name of the book's author or authors.
4. **Language**: The language in which the book is written.
5. **Category**: The genre or category to which the book belongs, such as fiction, non-fiction, romance, mystery, etc.
6. **Average Rate**: The average rating or score given to the book by readers.
7. **Maturity**: An indication of the book's intended audience, such as children, young adults, or mature readers.
8. **Publisher**: The name of the publishing company responsible for producing the book.
9. **Published Date**: The date when the book was published.
10. **Page Count**: The total number of pages in the book.

These features provide valuable information about the books, enabling our recommendation system to understand the content, genre, authorship, and other important aspects.

The dataset is represented in a **structured format**, typically using **columns and rows**, with each row representing a unique book and each column representing a specific feature or attribute.

This **comprehensive dataset** allows our system to make informed recommendations based on the users' preferences and engagement with similar books.

DATA STRUCTURES DESIGN

OUR CLASSES

1) User Class:

- __init__(self, user_id): Initializes a user object with a unique user ID and an empty list to store the books the user has read. This function sets up the initial state of the User class.
- add_book(self, book): Adds a book object to the list of books the user has read. This function allows the user to update their list of read books.
- get_read_books(self): Returns the list of books the user has read. This function provides access to the user's read book list.

2) Book Class:

- __init__(self, book_id, title, authors, language, categories, average_rating, maturity, publisher, year, num_pages): Initializes a book object with the specified att
- ributes. This function sets up the initial state of the Book class.
- get_attribute(self, attribute_name): Returns the value of the specified attribute for the book. This function allows accessing the values of different attributes of a book.

Interactions between User and Book Classes:

When a user reads a book, the add_book() method of the User class is called to add the book object to the user's list of read books. This interaction helps maintain the user's read book list.

The get_read_books() method of the User class is used to retrieve the list of books the user has read. This interaction allows accessing the user's read book list.

For content-based filtering, the get_attribute() method of the Book class is used to retrieve the value of a specific attribute for a book. This interaction is crucial for comparing attributes across different books.

The other principal features we will use after collecting and cleaning the data are :

- **Item-Item Similarity:** Calculate similarity between items based on their attributes
- **Item Rankings:** based on the users preferences and the books he read before we could make a **linked list** where the first elements are the ones that are too similar to his prefereces
- **Querying and Recommendation Generation:** When a user requests recommendations, you can use their query (e.g., an author they're interested in) to retrieve the most similar items from the item-item similarity data structure. You can then provide the top-ranked items from the retrieved set as recommendations.

The design of our program:

After we give the user 100 random books to read, we create the user profile based on the books the user has read, we capture information about their preferences for attributes like genre, language, author, page count, and more, the we assign weights to each attribute based on the user's preferences.

Then, we determine the similarity between the user's preferences and the attributes of each book.

- In categorical attributes like genre and language: we calculate the similarity based on whether the book matches the user's preferred categories and use a **hash-map** to store the weights calculated.
- In numerical attributes like page count or publication year: we measure the similarity based on the numerical distance between the user's preferred value and the value of the book.

After , we calculate an overall recommendation score for each book by combining the weighted similarity scores across all attributes.

Finally, we sort the books based on their recommendation scores and present the top-ranked books as recommendations to the user.

FINAL OUTCOME

The final output of our content-based book recommendation system will be a **personalized list of book recommendations for each user**. The recommendations will be based on the user's past engagement with similar books and the content analysis of the books themselves. The system will generate a **ranked list of recommended books, with the most relevant and fitting options appearing at the top.**

A **minimum** successful result for our recommendation system would involve **providing a list of recommended books that align with the user's preferences and exhibit some degree of**

similarity to their past engagements. This basic outcome should demonstrate that the system is functioning and capable of offering relevant book suggestions based on content analysis.

An **ideal or maximum** result for our recommendation system would involve generating h**ighly accurate and diverse recommendations that closely match the user's preferences and introduce them to new and interesting books**. The system would take into account not only the content-based similarities but also factors like genre, author, language, and other relevant attributes to curate a list of highly personalized book recommendations.