Gisma University of Applied Sciences

Gisma University of Applied Sciences

Department of Computer and Data Sciences

B198c7 AI Applications for Digital Business

Book Recommendation System

Contents

1. Introduction

In today's digitized world we have tons of books accessible online, digital libraries and virtual bookstores, which find it very difficult for the avid readers to find the book they wish to read. Millions of books, across different genres, authors and topics to search for the perfect read, all manually is too much work and too much overwhelm. To tackle this, a Book Recommendation System offers to recommend the books based on the user's interest, reading history and preferences.

1.1 Overview

Book Recommendation System is a web application written with the machine learning powering the web app to provide a book suggestion based on user preferences. Using Collaborative Filtering, Content Based Filtering on the platform to generate the personalized book suggestions. It analyzes book features for the user and returns seven matching books to the user from system processes for user input.

This system is implemented with the Flask framework, which means it is lightweight, efficient and interactive. It gives an intuitive interface for users to find the book recommendation effortlessly. For this project, we make use of a dataset that came from Kaggle having over 13 million book entries which makes our recommendations robust to real world data.

Thus, the Collaborative Filtering technique of the system will suggest books by figuring out at what point the user behavior pattern is and then recommending the titles that other users who have similar preferences had enjoyed. Content Based Filtering on the other hand instead tries to suggest similar items to the user based on characteristics of a book (such as genre, author, key words), and focuses on books themselves. Through the combination of these two approaches, the system provides more accurate and varied recommendations.

This system has a web interface where the users can look up book titles or pick from popular categories. Real time dynamic recommendations are provided to users which makes their browsing experience better. The system is designed with scalable architecture upon which it can be scaled indefinitely without the system going back to the weeds for performance.

1.2 Objective

In this project, the Objective section is about improving book discovery through book recommendation based on users' interest and reading habits. To do this, the system analyzes different book attributes such as title, author, genre and user rating to produce intelligent book suggestions.

Additionally, this system aims to:

1.Personalized Recommendations

The idea is to increase user engagement through personalized book suggestion that is tailored to an individual preference.

To predict books, it uses past reading behavior and ratings, but providing books that match the user's taste.

2.Machine Learning for Accuracy

Recommendations are optimized based on advanced machine learning techniques.

The quality of system recommendations continues to improve, the system adaptively learns from the user interactions and gets better and better.

3. User-Friendly Interface

I added the web application to be Flask-based which makes it intuitive as well as seamless.

The interface is simple yet easy to use to find a book.

4. Exploratory Data Analysis (EDA)

The system follows data analysis for understanding the trends in the popularity of book.

EDA helps refine the recommendations and improve the user experience.

The project seeks to integrate these components to develop a smart book recommendation system in smart book recommendation system that improves the reading habit and makes the book discovery more enjoyable.

2. Features and Functionality

2.1 Key Features

Popular Books Section:

- This section shows a selection of fifty most read books depending on the general rating and reviews submitted by users.
- This feature makes it possible for users to know what is popular within the reading circles and high ranking books from a larger pool of readers.
- They are assessing different parameters of its popularity such as the number of ratings, mean rating score, engagement level of the users.

- Personalized Recommendations:

- Supplies seven books, which are similar to the book selected by the user.
- It incorporates CF algorithm and CB algorithm to offer solutions that have been seen to contain similar components of the client's taste in books.
- The recommendations are based on user intermittently and changes in aspects such as preferences.

- Hybrid Recommendation Approach:
- The operation of the system utilizes a dual-recommendation approach that is as follows:
- Recommendations – Collaborative Filtering Works on the basis of user-item interactions and prescribes book liking or suggesting books using the likeness of other users.
- Content-based filtering – looks at book characteristics like the type of the book, author of the book and the description to find books with similar characteristics.
- Thus, increasing the accuracy of the recommendation since each of the used methods has its own drawbacks and this way it is possible to avoid them.

Flask Web Application:

- Recommendation system for the selected store is created using Flask which is a micro web framework.
- This makes it possible to provide a speedy, innovative, and easy system for the browsing of book recommendations.
- The layout of the web application is developed to be as friendly as possible so that users in mobile devices can also access the application.

Efficient Data Handling:

- Thus, before analyzing the dataset, it goes through the data cleansing and pre-processing stages to eliminate errors, duplications, etc.

Steps include:

- Dealing with missing values of book information
- Standardizing the ratings will ensure that the quality of the recommendations given is not distorted in any way
- This tutorial discusses how to tune Oracle in order for it to perform faster in terms of its storage and retrieval system.

2.2 How the System Works

A structured approach is used to recommend book suggestions in the book recommendation system that will result in a relevant and trustworthy set of suggestions to the publisher. The process begins with:

This involves Data Collection, when a dataset made up of 13 million book entries is taken from Kaggle. The essential metadata that make up a basis for recommendation logic formation includes title, author, genre and user ratings, all present in this dataset.

Once collected, the data undergoes

It includes cleaning, structuring and formatting the data in such a way that the data can be easily processed. This step helps to ignore unnecessary values, separate duplicate records, and eliminates any inconsistency to make the data more valuable for analysis and model training.

Following this,

It does the Exploratory Data Analysis (EDA) to extract important insight and trend in user preference. EDA will help to understand pattern like top popular genres, how frequently rated books, user engagement etc. all this will be used to improve the recommendation model.

Recommendation Engine is the central part of the system: it uses two main methods for recommendation: Collaborative Filtering and Content Based Filtering.

Collaborative Filtering predicts books rating and preference according to the similar user and recommend books similar user enjoyed. On the other hand,

In Content Based Filtering, it uses Cosine Similarity to compare book attributes in order to suggest books to a reader who has read another book.

Using book attributes to compare between books and recommend the books similar to a user’s appraised book. The system combines both methods to have more accurate and personalized book suggestions.

Finally, a Flask-Based Web Application is presented with the recommendations to users. For the sake of its user friendliness and dynamism, this web application encourages an easy exploration of book suggestions as well as easy viewing of detailed information along with engaging with the recommendation system. The system merges data driven insights and machine learning techniques to provide an interactive and smart book discovery experience.

3. Tech Stack and Tools Used

3.1 Machine Learning Technologies

The book recommendation system incorporates the use of machine learning that aid in the generation of relevant and best recommendations. Another important method of recommendation is collaborative filtering which comes to a conclusion regarding which books should be suggested by evaluating efficiency of previous choices made by users. It makes predictions by finding similarities between peoples’ behaviour or items in question using user-based or item-based similarities. In this kind of collaborative filtering, the system selects users with similar reading preferences as the target user and provides the target user with the list of books that the users liked to read. However, one of the most common recommendation strategies is based on items, that is, books which are usually rated by users together. Another method include content-based where books are recommended based on factors such as genre, author or specific keywords related to specific book titles. Unlike the previous techniques, this does not involve the user to strongly interact with it as it focuses on attributes of books. Furthermore, cosine similarity is employed in order to examine how similar two books are since text-based features are being compared. Through the calculation of the angle between their features there is the determination of their similarity where a smaller angle generally means that they are almost the same. These techniques are used to improve the recommendation system to provide better recommendation to the users regarding the books to read.

- 3.2 Development Tools

Python:

- The language that was used as the primary language in writing the entire code for the recommendation system.

- Popular due to its vast ecosystem of libraries for data science, machine learning, and web development.

Pandas & NumPy:

- Pandas: Offers Data Structures For all Data types and specifically provides a Data frame by which one can manage the data structure easily.
- NumPy : Provides support for arithmetic processing of Data Matrix, to accomplish high speed mathematical computation of Big Data.
- Both are applicable where they are employed in cleaning and pre-processing data before feeding it into the data analysis models.
- Scikit-learn:

- Machine learning contains many ready-made models for classification, regression or clustering built in the python language.
- In used for implementing recommendation algorithms like Collaborative filtering as well as Cosine similarity.

1. **Matplotlib & Seaborn**:

- Plotting library for creating static, animated and interactive visualizations, Matplotlib.
- It is oratory on Matplotlib, supplies high-level functions to help with statistical visualization.
- Used for Exploratory Data Analysis (EDA) to visualize the trends, correlations and insights from the book ratings and metadata.

Flask:

- It is a lightweight web framework for Python as it helps in developing a web interface for recommendation system.
- It was used to create APIs and integrate the machine learning model with a user friendly.

HTML & CSS:

- HyperText Markup Language (HTML): A standard markup language used for creating web pages.
- Cascading Style Sheets (CSS): Is used to enhance the visual style, layout and responsiveness of the user interface.
- They contribute to the front-end of the book recommendation system to guarantee a good user experience.

4. Installation and Setup

Running the Book Recommendation System

Prerequisites

Do the following checklists before you run the project:

Python 3.x

Node.js and npm

Steps to Set Up and Run the Project

1.Install Python dependencies:

Open a terminal or a command prompt and write the following command:

```
 pip install -r requirements.txt
```

2. Install Node.js dependencies:

```
 npm install
```

3. Build the CSS using Tailwind:

```
 npm run build:css
```

4. Start the Flask application:

```
 python app.py
```

5. Access the application:

Firstly, start the server after which you can access it through a web browser by entering the following URL:

  http://127.0.0.1:5000

Application Features

Home Page: http://127.0.0.1:5000/

Top 50 Books: http://127.0.0.1:5000/top50

Book Recommendations: http://127.0.0.1:5000/recommend

About Page: http://127.0.0.1:5000/about

To find the information regarding books, one has to type in the name of the books like the following ones:

Harry Potter and the Prisoner of Azkaban

The Da Vinci Code

1984

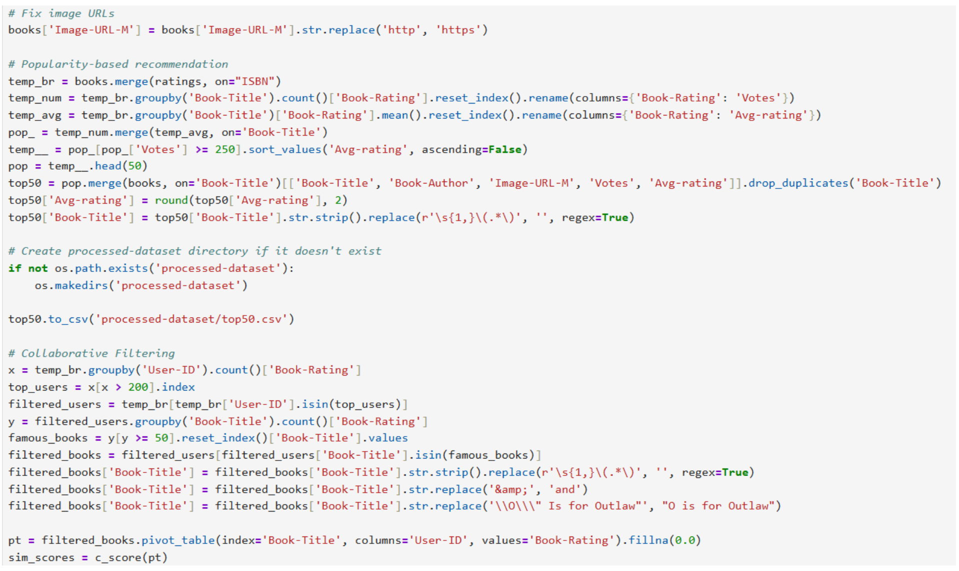
This system enables one to have an easy time in searching for books by their own desired genre or one that is recommended.

5. Model and Methodology

5.1 Data Processing and Analysis

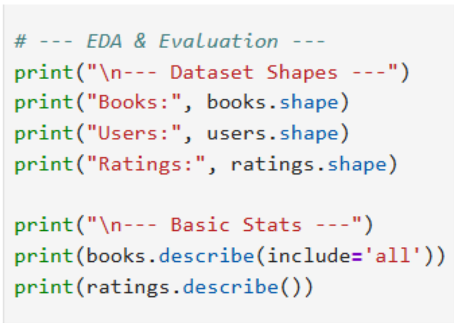
The data processing and analysis step is very important in the process of getting the data ready for application in the book recommendation system. The first step is :

data cleaning process in which all the necessary books with incomplete or wrong data are removed in order to provide accurate and reliable data. They include the process of detecting and filling missing values, elimination of duplication records and errors present in the records.



```python
# Fix image URLs
books['Image-URL-M'] = books['Image-URL-M'].str.replace('http', 'https')

# Popularity-based recommendation
temp_br = books.merge(ratings, on="ISBN")
temp_num = temp_br.groupby('Book-Title').count()['Book-Rating'].reset_index().rename(columns={'Book-Rating': 'Votes'})
temp_avg = temp_br.groupby('Book-Title')['Book-Rating'].mean().reset_index().rename(columns={'Book-Rating': 'Avg-rating'})
pop_ = temp_num.merge(temp_avg, on='Book-Title')
temp__ = pop_[pop_['Votes'] >= 250].sort_values('Avg-rating', ascending=False)
pop = temp__.head(50)
top50 = pop.merge(books, on='Book-Title')[['Book-Title', 'Book-Author', 'Image-URL-M', 'Votes', 'Avg-rating']].drop_duplicates('Book-Title')
top50['Avg-rating'] = round(top50['Avg-rating'], 2)
top50['Book-Title'] = top50['Book-Title'].str.strip().replace(r'\s{1,}\(.*\)', '', regex=True)

# Create processed-dataset directory if it doesn't exist
if not os.path.exists('processed-dataset'):
    os.makedirs('processed-dataset')

top50.to_csv('processed-dataset/top50.csv')

# Collaborative Filtering
x = temp_br.groupby('User-ID').count()['Book-Rating']
top_users = x[x > 200].index
filtered_users = temp_br[temp_br['User-ID'].isin(top_users)]
y = filtered_users.groupby('Book-Title').count()['Book-Rating']
famous_books = y[y >= 50].reset_index()['Book-Title'].values
filtered_books = filtered_users[filtered_users['Book-Title'].isin(famous_books)]
filtered_books['Book-Title'] = filtered_books['Book-Title'].str.strip().replace(r'\s{1,}\(.*\)', '', regex=True)
filtered_books['Book-Title'] = filtered_books['Book-Title'].str.replace('&amp;', 'and')
filtered_books['Book-Title'] = filtered_books['Book-Title'].str.replace('\\O\\\" Is for Outlaw"', "O is for Outlaw")

pt = filtered_books.pivot_table(index='Book-Title', columns='User-ID', values='Book-Rating').fillna(0.0)
sim_scores = c_score(pt)
```

After cleaning the data, EDA and data visualization are conducted to analyze and present the data and to draw conclusion based on that. Graphics and statistical diagrams are created with the goal of detecting tendencies, tendencies and patterns among the users, regarding their readings and the books, within the data. These visualizations have features such as the top genre, top rated books, and the trend of the users' engagement levels.



```
# --- EDA & Evaluation ---
print("\n--- Dataset Shapes ---")
print("Books:", books.shape)
print("Users:", users.shape)
print("Ratings:", ratings.shape)

print("\n--- Basic Stats ---")
print(books.describe(include='all'))
print(ratings.describe())
```

Another key step is: data preparation where essential attributes of books such as the title, author, and genre, the great features that can be used to train the model are engineered. It entails encoding categorical data into numerical format, feature extraction from texts and cleaning data in order to enhance machine learning. Through systemizing the data and analyzing the data, it will be easy for the recommendation system to come up with more appropriate book recommendations to the users.

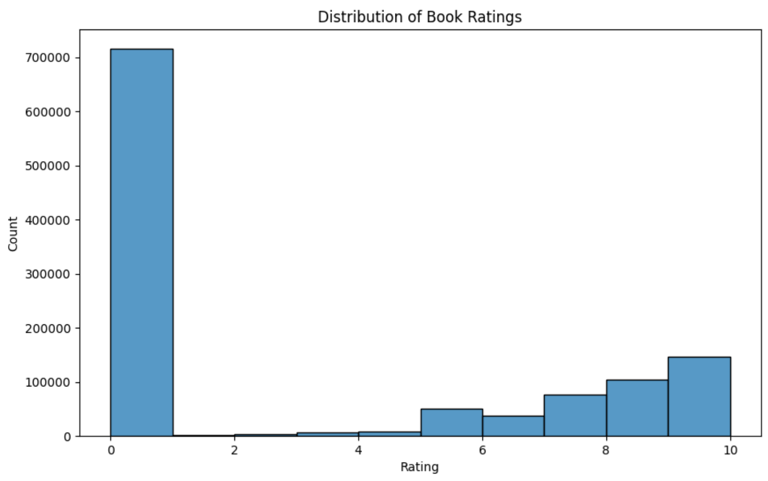


Fig: Distribution of Book Ratings

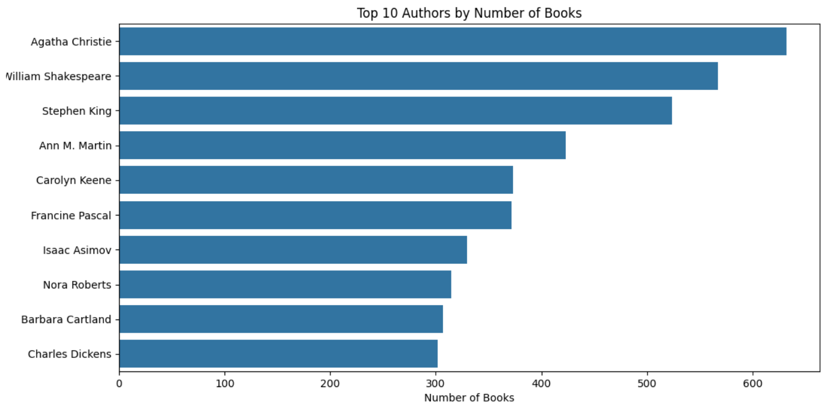


Fig: Top Authors

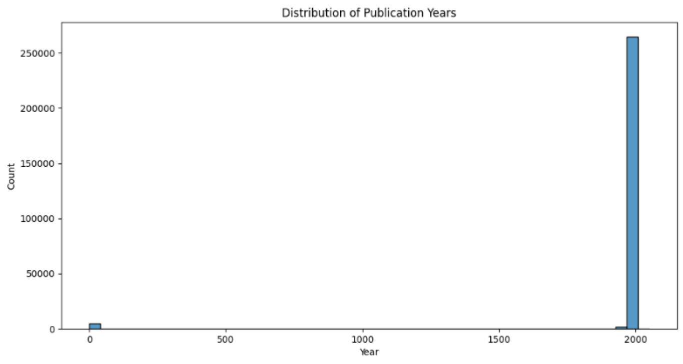


Fig: Distribution of Publication Years

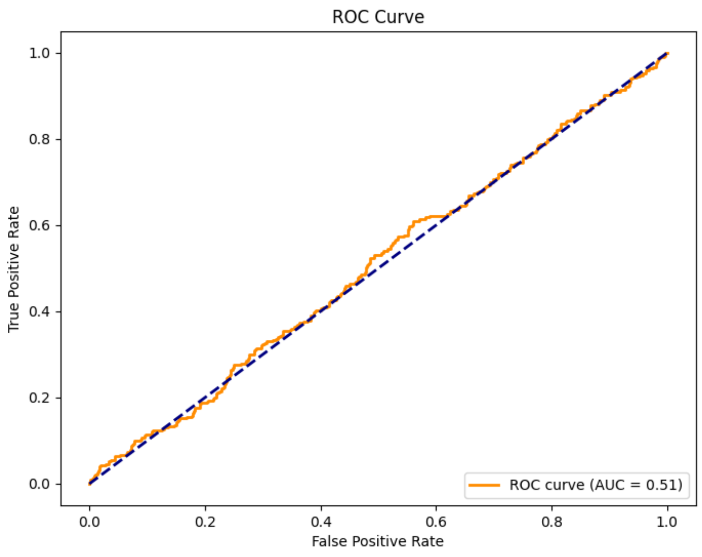
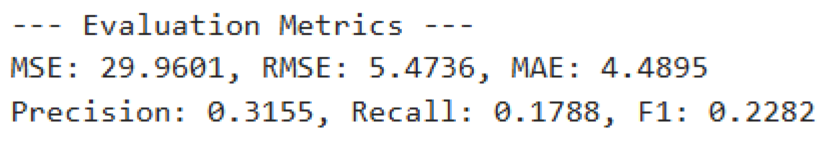


Fig: ROC Curve



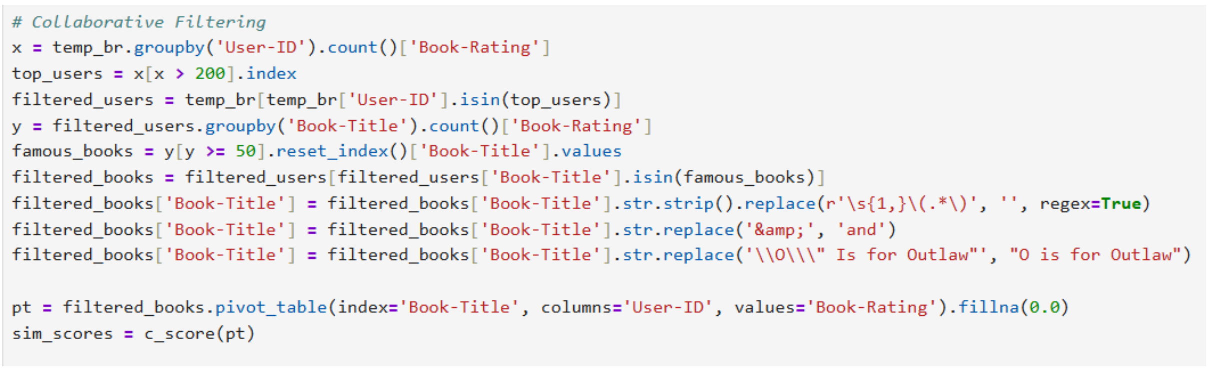
```
--- Evaluation Metrics ---
MSE: 29.9601, RMSE: 5.4736, MAE: 4.4895
Precision: 0.3155, Recall: 0.1788, F1: 0.2282
```

Fig: Evaluation Metrics

5.2 Recommendation Techniques

In generating book recommendations for the users, two methods are mostly employed in the recommendation system: Collaborative Filtering and Content Based Filtering.

Collaborative Filtering involves analyzing users’ ratings to come up with books that are similar in terms of the ratings given. It determines books that are most frequently read by users, who have read the book also and provides ratings for books that are similar to the books that the user absorbs. To ensure that results are as accurate as possible and to enhance efficiency, the system utilizes matrix factorisation that results in big datasets being decomposed into more manageable matrices through a variety of matrix factors even in cases where a few ratings are available. This way users are provided with recommendation more as a result of group reading frequency.



```python
# Collaborative Filtering
x = temp_br.groupby('User-ID').count()['Book-Rating']
top_users = x[x > 200].index
filtered_users = temp_br[temp_br['User-ID'].isin(top_users)]
y = filtered_users.groupby('Book-Title').count()['Book-Rating']
famous_books = y[y >= 50].reset_index()['Book-Title'].values
filtered_books = filtered_users[filtered_users['Book-Title'].isin(famous_books)]
filtered_books['Book-Title'] = filtered_books['Book-Title'].str.strip().replace(r'\s{1,}\(.*\)', '', regex=True)
filtered_books['Book-Title'] = filtered_books['Book-Title'].str.replace('&amp;', 'and')
filtered_books['Book-Title'] = filtered_books['Book-Title'].str.replace('\\O\\\" Is for Outlaw"', "O is for Outlaw")

pt = filtered_books.pivot_table(index='Book-Title', columns='User-ID', values='Book-Rating').fillna(0.0)
sim_scores = c_score(pt)
```

Content-Based Filtering, on the other hand, focuses on the characteristics of books rather than user behavior. It uses Cosine Similarity to compare books by calculating similarities of how closely two books are related to each other on features such as author, genre, reviews, etc. It analyzes books descriptions and metadata for recommendation, giving personalized suggestions, even for books that the user has limited ratings over.

```
# Popularity-based recommendation
temp_br = books.merge(ratings, on="ISBN")
temp_num = temp_br.groupby('Book-Title').count()['Book-Rating'].reset_index().rename(columns={'Book-Rating': 'Votes'})
temp_avg = temp_br.groupby('Book-Title')['Book-Rating'].mean().reset_index().rename(columns={'Book-Rating': 'Avg-rating'})
pop_ = temp_num.merge(temp_avg, on='Book-Title')
temp__ = pop_[pop_['Votes'] >= 250].sort_values('Avg-rating', ascending=False)
pop = temp__.head(50)
top50 = pop.merge(books, on='Book-Title')[['Book-Title', 'Book-Author', 'Image-URL-M', 'Votes', 'Avg-rating']].drop_duplicates('Book-Title')
top50['Avg-rating'] = round(top50['Avg-rating'], 2)
top50['Book-Title'] = top50['Book-Title'].str.strip().replace(r'\s{1,}\(.*\)', '', regex=True)
```

The combination of the both techniques provides well rounded recommendations, using user behavior and book attributes to improve reading experience.

6. Application Endpoints and Functionality

6.1 Popular Books Page

Endpoint: /top50

Shows a list of 50 popular book ranked by average user rating.

6.2 Book Recommendation Page

Endpoint: /recommend

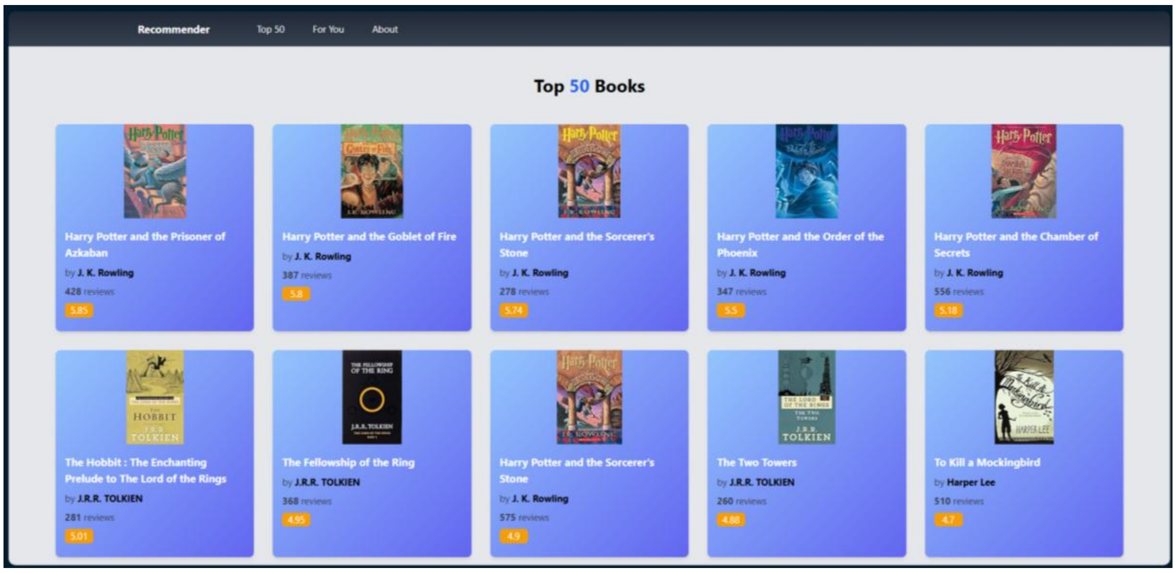
It is a function that accepts user input(book title) and returns seven book recommendations.

It ensures accurate recommendations with the help of Cosine Similarity Scores.

7. Application Preview

7.1 Top 50 Books Page

Mainly, the Application Preview contains 2 main parts: Top 50 Books Page and Book Recommendation Page. The Top 50 Books is a page which shows a limited amount of the highest rated books out of the dataset which help users find the popular as well as highly reviewed books. It is a feature that helps users look through top rated books from its community, or predefined criteria.



Top 50 Books

Harry Potter and the Prisoner of Azkaban
by J. K. Rowling
428 reviews
5.85

Harry Potter and the Goblet of Fire
by J. K. Rowling
387 reviews
5.8

Harry Potter and the Sorcerer's Stone
by J. K. Rowling
278 reviews
5.74

Harry Potter and the Order of the Phoenix
by J. K. Rowling
347 reviews
5.5

Harry Potter and the Chamber of Secrets
by J. K. Rowling
556 reviews
5.18

The Hobbit : The Enchanting Prelude to The Lord of the Rings
by J.R.R. TOLKIEN
281 reviews
5.01

The Fellowship of the Ring
by J.R.R. TOLKIEN
368 reviews
4.95

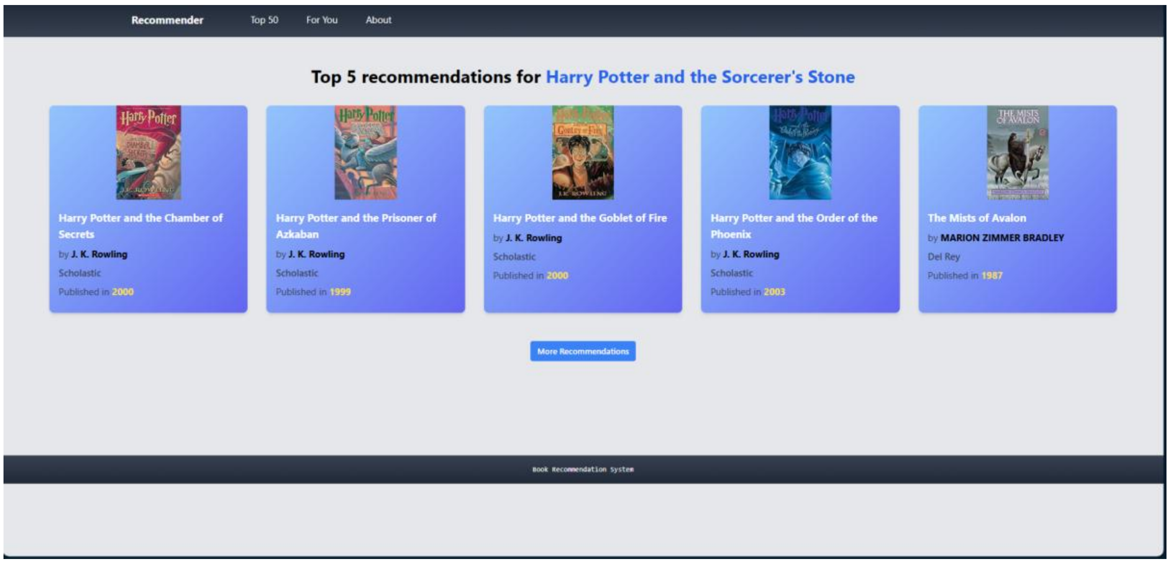
Harry Potter and the Sorcerer's Stone
by J. K. Rowling
575 reviews
4.9

The Two Towers
by J.R.R. TOLKIEN
260 reviews
4.88

To Kill a Mockingbird
by Harper Lee
510 reviews
4.7

7.2 Book Recommendation Page

The Book Recommendation Page allows the users to type the name of the book they like and get seven recommended books based on similarity. Using these factors to analyze, these recommendations are designed to help users find books that fit that flavor of reading, genre, themes or reader reviews. This feature helps in giving personalized suggestions according to the users' interests.



Top 5 recommendations for Harry Potter and the Sorcerer's Stone

Harry Potter and the Chamber of Secrets
by J. K. Rowling
Scholastic
Published in 2000

Harry Potter and the Prisoner of Azkaban
by J. K. Rowling
Scholastic
Published in 1999

Harry Potter and the Goblet of Fire
by J. K. Rowling
Scholastic
Published in 2000

Harry Potter and the Order of the Phoenix
by J. K. Rowling
Scholastic
Published in 2003

The Mists of Avalon
by MARION ZIMMER BRADLEY
Del Rey
Published in 1987

Book Recommendation System

8. Future Enhancements

8.1 Planned Improvements

The other thing that the application will do is to integrate deep learning using Neural Networks to improve its ability to analyze user preferences and thus enhance the accuracy of recommendations.

User based recommendations will also be introduced by using reading history to recommend user to related book.

Because I want to provide a better and a more responsive and an enjoyable user interface, I will be using modern frameworks like React.js to enhance entire user experience.

On top of that, users should be able to filter their searches by genre, author or rating to make the search process more efficient and tailored to the user's specific tastes.

9. Conclusion

Book Recommendation System is a smart application that can discover book by using the machine learning algorithms. It offers the personalized book suggestions by combining Collaborative Filtering and Content Based Filtering based on the user behavior and book attributes.

The system is built using a Flask based interface, which allows a smooth, easy, and efficient UX. It standardizes book discovery, while at the same time maintaining accuracy and relevance in recommendations.

In future, deep learning techniques will be used to make the score more accurate, the personalization are better and the user experience better through advanced filtering options. The design of function 2 is to make book discovery even easier and more tailored to personal preferences.

Github Link

https://github.com/MuhammadAhmadJamil18/B198c7-Ai-application-for-Digital-Systems-.git