BOOK RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

G. SUBRAHMANYAM[RA2011026010188]

GUNA DARAM [RA2011026010208]

V.KAMALESH KUMAR[RA2011026010198]

Under the guidance of

Dr. U.SAKTHI

Assistant Professor, Department of Computing Technologies

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"BOOK RECOMMENDATION SYSTEM"** is the bona fide work of **DARAM GUNASHEKAR [RA2011026010208] G.SUBRAHMANYAM [RA2011026010188] V.KAMALESH KUMAR[RA2011026010198]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate

SIGNATURE

Dr. U.SAKTHI
Assistant Professor
Department of Networking and communications

SIGNATURE

Dr. R.ANNIE UTHRA
HEAD OF THE DEPARTMENT
Professor & Head
Department of networking and communications

ABSTRACT

Now-a-days, everyone depends on reviews by others in many things such as selecting a movie to watch, buying products, reading a book. Recommender systems are used for that purpose only. A recommender system is a kind of filtering system that predicts a user's rating of an item. Recommender systems recommend items to users by filtering through a large database of information using a ranked list of predicted ratings of items. Online Book recommender system is a recommender system for ones who love books. When selecting a book to read, individuals read and rely on the book ratings and reviews that previous users have written. In this paper, Hybrid Recommender system is used in which Collaborative Filtering and ContentBased Filtering techniques are used. The author used Collaborative techniques such as Clustering in which data-points are grouped into clusters. Algorithms such as Kmeans clustering and Gaussian mixture are used for clustering. The better algorithm was selected with the help of silhouette score and used for clustering. Matrix Factorization techniques such as Truncated-SVD which takes sparse matrices as input is used for reducing the features of a dataset. The Content Based Filtering System used a TFIDF vectorizer which took statements as input and returned a matrix of vectors. RMSE (Root Mean Square Error) is used for finding the deviation of an absolute value from an obtained value and that value is used for finding the fundamental accuracy.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

RMSE	Root Mean square Error
SVD	Singular ValueDecomposition
NLP	Natural language processing

CHAPTER 1

INTRODUCTION

Now-a-days, online rating and reviews are playing an important role in book sales. Readers were buying books depending on the reviews and ratings by the others. Recommender system focuses on the reviews and ratings by the others and filters books. In this paper, the Hybrid recommender system is used to boost our recommendations. The technique used by recommender systems is Collaborative filtering. This technique filters information by collecting data from other users. Collaborative filtering systems apply the similarity index-based technique. The ratings of those items by the users who have rated both items determine the similarity of the items. The similarity of users is determined by the similarity of the ratings given by the users to an item. Content-based filtering uses the description of the items and gives recommendations which are similar to the description of the items. With these two filtering systems, books are recommended not only based on the user's behavior but also with the content of the books. So, our recommendation system recommends books to the new users also. In this recommender system, books are recommended based on collaborative filtering technique and similar books are shown using content based filtering. The required dataset for the training and testing of our model is downloaded from Good-Reads website. Matrix Factorization technique such as Truncated-SVD which takes sparse matrices of dataset is used for reduction of features. The reduced dataset is used for clustering to build a recommendation system. The better model is selected based on the silhouette score and used for clustering. Silhouette score or silhouette coefficient is used to calculate how good the clustering is done. Negative value shows that clustering is imperfect whereas positive value shows that clustering was done perfectly. Difference between the mean rating before clustering and after clustering is calculated. Root Mean square Error is used to measure the error between the absolute 2 values and obtained values. That RMSE value is used to find the fundamental accuracy.

CHAPTER 2

LITERATURE SURVEY

Most researchers used Pearson's Correlation Coefficient function to calculate similarity among book ratings to recommend books

Collaborative Filtering with Jaccard Similarity to build a recommendation system

Avi Rana and K. Deeba, et.al. (2019) [1] proposed a paper "Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)". In this paper, the author used CF with Jaccard similarity to get more accurate recommendations because general CF difficulties are scalability, sparsity, and cold start. So to overcome these difficulties, they used CF with Jaccard Similarity. JS is based on a pair of books index which is a ratio of common users who have rated both books divided by the sum of users who have rated books individually. Books with a high JS index are highly recommended.

Building a Recommendation System using Machine learning Framework

G. Naveen Kishore, et.al. (2019) [2] proposed a paper "Online Book Recommendation System". The dataset used in this paper was taken from the website "good books-10k dataset" which contains ten thousand unique books. Features are book_id, user_id, and rating. In this paper, the author adopted a machine learning framework model to create neural network embedding.

Using Quick sort Algorithm approach to design a system

Uko E Okon, et.al. (2018) [3] proposed a paper "An Improved Online Book Recommender System using Collaborative Filtering Algorithm". The authors designed and developed a recommendation model by using a quick sort algorithm, 13 collaborative filtering, and object-oriented analysis and design methodology (OOADM). This system produces an accuracy of 90-95%

Using UV Decomposition and Model for building system

Jinny Cho, et.al. (2016) [4] proposed a paper “Book Recommendation System”. In this paper, the author uses two approach methods which are Content-based (CB) and Collaborative Filtering (CF). They used two algorithms as UV-Decomposition.They obtained a result with an accuracy of 85%.

Hybrid Recommender System through Collaborative Filtering

Anagha Vaidya and Dr. Subhash Shinde, et.al. (2019) [9] proposed a paper “Hybrid Book Recommendation System”. In this paper, the author used techniques such as Collaborative Filtering etc. and used the Pearson correlation coefficient. It was published in International Research Journal of Engineering and Technology (IRJET)

Using Machine Learning Algorithm to build a system

Dhirman Sarma,Tanni Mittra and Mohammad Shahadat Hossain, et.al. (2019) [10] proposed a paper “Personalized Book Recommendation System using Machine Learning Algorithm”. It was published in The Science and Information Organization vol.12.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 ARCHITECTURE DIAGRAM

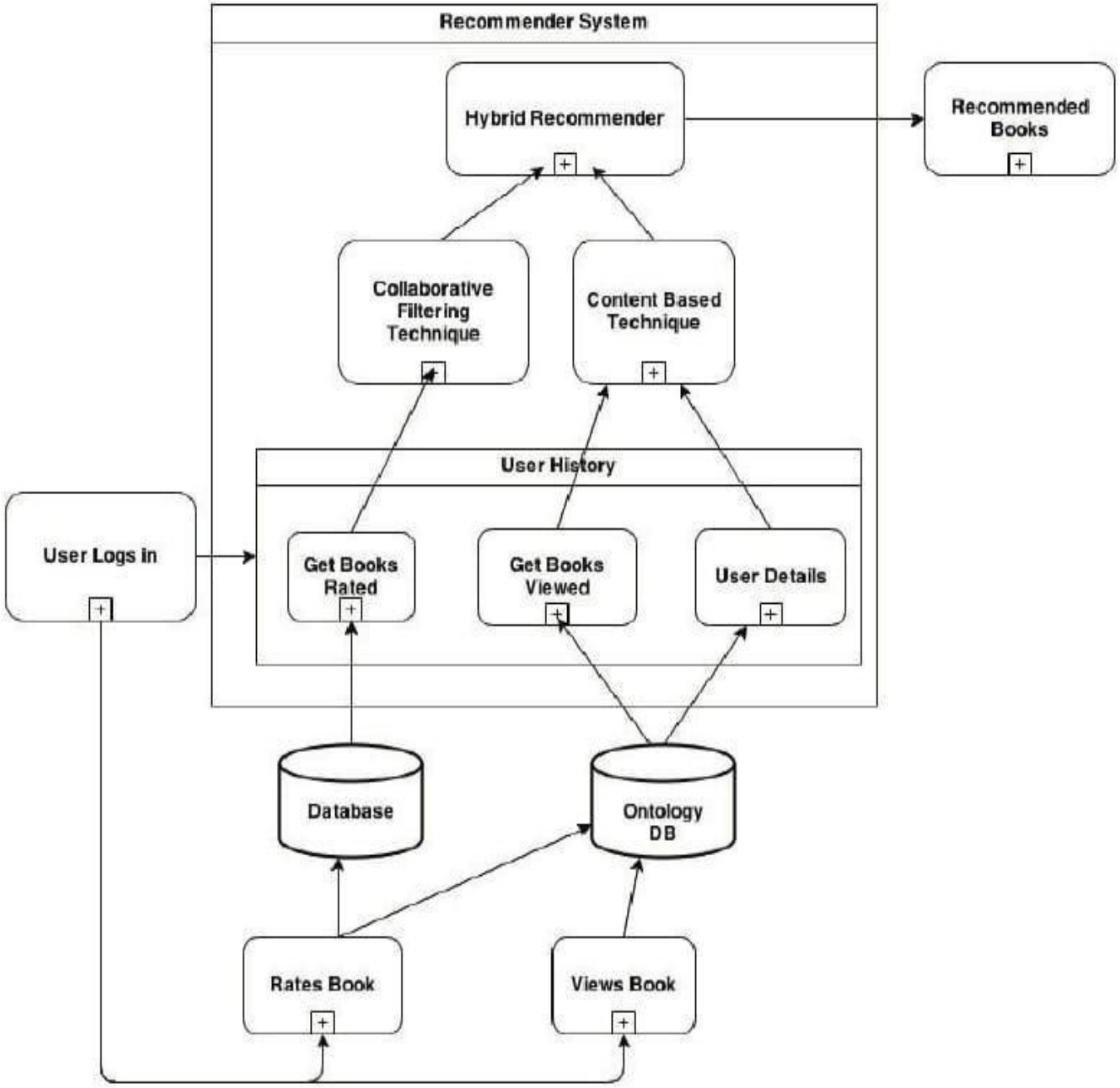

Fig 3.1

- Fig 3.1 shows the main components of the book recommendation system and their interactions.
- The data sources component retrieves book data from various sources, such as databases or web scraping, and stores it in a data storage component.
- The data pre-processing and feature engineering component cleans, transforms and extracts the features that will be used by the recommendation engine.
- The recommendation engine component analyzes the data and creates personalized recommendations for users.
- The user profiling component collects data on users' preferences and reading history to create user profiles that the recommendation engine uses to provide personalized recommendations.
- The user interface component displays book recommendations to users and allows them to search for books, read reviews and provide feedback.
- The feedback mechanism component collects user feedback, ratings, and other relevant information to improve the accuracy of the recommendation engine.
- Overall, this architecture diagram shows the flow of data and interactions between the components of the book recommendation system.

3.2 DESCRIPTION OF MODULE AND COMPONENTS :

A book recommendation system is an application that suggests books to users based on their interests, reading history, and other relevant factors.

3.2.1 Data Collection:

The system needs a large database of books and their details such as title, author, genre, publication date, summary, and rating. This data can be obtained from various sources such as online bookstores, library catalogs, and book review websites.

3.2.2 User Profiling:

The system needs to create user profiles based on their interests, reading habits, and other relevant factors. This can be done by collecting data on the user's search history, reading history, and other activities on the platform.

3.2.3 Recommendation Engine:

The recommendation engine is the core component of the system that analyzes user data and suggests books that match their preferences. The recommendation engine can be based on various algorithms such as **collaborative filtering, content-based filtering, and hybrid filtering.**

3.2.4 User Interface:

The user interface is the component that interacts with the user and displays the book recommendations. The interface can be a website or a mobile application that allows users to browse and search for books, view recommendations, and read reviews.

3.2.5 Feedback Mechanism:

The system needs a feedback mechanism that allows users to rate books, provide feedback, and improve the accuracy of the recommendation engine. This feedback can be used to fine-tune the recommendation engine and provide better book suggestions to users.

Overall, a book recommendation system requires a combination of data science, machine learning, and user interface design to provide personalized book recommendations to users.

CHAPTER 4

METHODOLOGY

System Architecture describes “the overall structure of the system and the ways in which the structure provides conceptual integrity”. The system architecture to build a recommendation system involves the following five major steps.

4.1.1 Data Acquisition

4.1.2 Data Pre-processing

4.1.3 Feature Extraction

4.1.4 Training Methods

4.1.5 Testing Data

In Step 4.1.1, Dataset was collected from Good Reads Website in which three datasets are present i.e. Books Dataset, Ratings Dataset, Users Dataset. In Step 4.1.2, Datasets were pre-processed to make suitable for developing the Recommendation system. In Step 4.1.3, Feature extraction is performed in which Truncated-SVD is used to reduce the features of the dataset and Data splitting is done in which training dataset and testing dataset are divided into 80:20 ratio. In Step 4.1.4, Content Based Filtering System is developed in which book description is taken as an input and Collaborative Filtering System is developed by building a model using K-Means Algorithm over Gaussian Mixture after comparing with Silhouette scores. In step 4.1.5, Testing of model with test data is performed.

4.2 DESCRIPTION OF MODULE AND COMPONENTS

the various modules in our proposed system and what each module contributes in achieving our goal are

4.2.1 Data Acquisition:

The goal of this step is to find and acquire all the related datasets or data sources. In this step, the main aim is to identify various available data sources, as data are often collected from various online sources like databases and files. The size and the quality of the data in the collected dataset will determine the efficiency of the model. The Books dataset is collected from the Goodreads website.

4.2.2 Data Pre-Processing:

The goal of this step is to study and understand the nature of data that was acquired in the previous step and also to know the quality of data. In this step, we will check for any null values and remove them as they may affect the efficiency. Identifying duplicates in the dataset and removing them is also done in this step.

4.2.3 Feature Extraction:

After pre-processing the acquired data, the next step is to reduce the features i.e. Dimensionality reduction. The reduced features should be able to give high efficiency. We used Matrix Factorization technique such as Truncated SVD which takes sparse matrix as input for reduction of features

Splitting the Dataset into the Training set and Test set:

In machine learning projects, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose, if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

4.2.4 Training Methods:

Now, we have our training and testing data. The next step is to identify the possible training methods and train our models. We have used two different clustering methods for training models. After that based on the silhouette score of each model, we would decide on which model to use finally

- ☐ Cosine Similarity

4.2.5 Testing Data :

Once Clustering model has been trained on pre-processed dataset, then the model is tested using different data points. In this testing step, the model is checked for the silhouette score for checking goodness of clustering. All the training methods need to be verified for finding out the best model to be used. In figures 3.10, 3.11, after fitting our model with training data, we used this model to predict values for test dataset. These predicted values on testing data are used for models comparison. The users in the test set, on average, rated their clusters' favorite books higher than a random set of 10 books by 0.47 stars, or nearly half a star.

CHAPTER 5

CODING AND TESTING

5.1. Coding

5.1.1. Back-End

1. Sample Data

```
books = pd.read_csv('books.csv')

users = pd.read_csv('users.csv')

ratings = pd.read_csv('ratings.csv')
```

2. Popularity Based

```
num_rating_df =
ratings_with_name.groupby('Book-Title').count()['Book-Rating'].reset_index
()

num_rating_df.rename(columns={'Book-Rating':'num_ratings'},inplace=True)

num_rating_df



avg_rating_df =
ratings_with_name.groupby('Book-Title').mean()['Book-Rating'].reset_index(
)

avg_rating_df.rename(columns={'Book-Rating':'avg_rating'},inplace=True)

avg_rating_df



popular_df = num_rating_df.merge(avg_rating_df,on='Book-Title')

popular_df
```

```
    popular_df =
popular_df[popular_df['num_ratings']>=250].sort_values('avg_rating',ascending=Fa
lse).head(50)

    popular_df =
popular_df.merge(books,on='Book-Title').drop_duplicates('Book-Title')[['Book-Tit
le','Book-Author','Image-URL-M','num_ratings','avg_rating']]
```

3. Collaborative Filtering

```
    x = ratings_with_name.groupby('User-ID').count()['Book-Rating'] > 200

    padhe_likhe_users = x[x].index

    filtered_rating =
ratings_with_name[ratings_with_name['User-ID'].isin(padhe_likhe_users)]

    y = filtered_rating.groupby('Book-Title').count()['Book-Rating']>=50

    famous_books = y[y].index

    final_ratings =
filtered_rating[filtered_rating['Book-Title'].isin(famous_books)]

    pt =
final_ratings.pivot_table(index='Book-Title',columns='User-ID',values='Book-Rati
ng')

    pt.fillna(0,inplace=True)

    pt
```

4. Similarity Based

```
    def recommend(book_name):

        # index fetch

        index = np.where(pt.index==book_name)[0][0]

        similar_items =
sorted(list(enumerate(similarity_scores[index])),key=lambda
x:x[1],reverse=True)[1:5]

        data = []

        for i in similar_items:

            item = []

            temp_df = books[books['Book-Title'] == pt.index[i[0]]]


item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))


item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))


item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].values))



            data.append(item)



        return data
```

5. Extracting Packages

```
    pickle.dump(pt,open('pt.pkl','wb'))

    pickle.dump(books,open('books.pkl','wb'))

    pickle.dump(similarity_scores,open('similarity_scores.pkl','wb'))
```

5.1.1. Front-End

1. Home Page

```
from flask import Flask,render_template,request

import pickle

import numpy as np

popular_df = pickle.load(open('popular.pkl','rb'))

pt = pickle.load(open('pt.pkl','rb'))

books = pickle.load(open('books.pkl','rb'))

similarity_scores = pickle.load(open('similarity_scores.pkl','rb'))

app = Flask(__name__)

@app.route('/')

def index():

    return render_template('index.html',

                           book_name =
list(popular_df['Book-Title'].values),

                           author=list(popular_df['Book-Author'].values),

                           image=list(popular_df['Image-URL-M'].values),

                           votes=list(popular_df['num_ratings'].values),

                           rating=list(popular_df['avg_rating'].values)

                           )

@app.route('/recommend')

    return render_template('recommend.html')

@app.route('/recommend_books',methods=['post'])

def recommend():

    user_input = request.form.get('user_input')

    index = np.where(pt.index == user_input)[0][0]
```

```
    similar_items = sorted(list(enumerate(similarity_scores[index])),
key=lambda x: x[1], reverse=True)[1:5]



    data = []

    for i in similar_items:

        item = []

        temp_df = books[books['Book-Title'] == pt.index[i[0]]]


item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].value
s))


item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].valu
es))


item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].valu
es))



        data.append(item)



    print(data)



    return render_template('recommend.html',data=data)



if__name__== '__main__':

    app.run(debug=True)
```

2. Search Page

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Book Recommender System</title>

    <!-- Latest compiled and minified CSS -->

<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh
4u" crossorigin="anonymous">

</head>

<style>

    .text-white{

        color:white

    }

</style>

<body style="background-color:black">

    <nav class="navbar" style="background-color:#00a65a">

        <p class="navbar-brand">My Book recommender</p>

        <ul class="nav navbar-nav">

            <li><a href="/">Home</a></li>

            <li><a href="/recommend">Search</a></li>

        </ul>

    </nav>

    <div class="container">

        <div class="row">
```

```
          <div class="col-md-12">

                <h1 class="text-white" style="font-size:50px">Recommend
Books</h1>

                <form action="/recommend_books" method="post">

                    <input name="user_input" type="text"
class="form-control"><br>

                    <input type="submit" class="btn btn-lg btn-warning">

                </form>

            </div>

            {% if data %}

             {% for i in data %}

                <div class="col-md-3" style="margin-top:50px">

                    <div class="card">

                        <div class="card-body">

                            <img class="card-img-top" src="{{i[2]}}">

                            <p class="text-white">{{i[0]}}</p>

                            <h4 class="text-white">{{i[1]}}</h4>

                        </div>

                    </div>

                </div>

             {% endfor %}

            {% endif %}

        </div>

    </div>

</body>

</html>
```

5.2. Testing

1. Popularity Based Method

popularity-based systems rely on the ratings given by users to different books. These systems analyze the ratings given by users to books and recommend books with similar ratings. For example, if a user has given high ratings to science fiction books, the system may recommend other science fiction books with high ratings.

```
popular_df = popular_df[popular_df['num_ratings']>=250]
```

This code filters the popular_df DataFrame to only include rows where the value in the num_ratings column is greater than or equal to 250. This will exclude books with fewer ratings, which are less likely to be representative of the overall quality of the book.

Next, the DataFrame is sorted by the avg_rating column in descending order using the following code:

```
popular_df.sort_values('avg_rating',ascending=False)
```

This code sorts the popular_df DataFrame in descending order based on the values in the avg_rating column. This will ensure that books with higher average ratings appear at the top of the DataFrame.

Finally, the first 50 rows of the sorted DataFrame are selected using the head() method and stored back into the popular_df variable:

```
.head(50)
```

This code selects the first 50 rows of the sorted DataFrame based on the current ordering and assigns it to the popular_df variable. This results in a new DataFrame containing the 50 most popular books, based on the number of ratings they have received and their average rating score.

2. Similarity Based Method

This code defines a function called recommend that takes in a book name as an argument and returns a list of information for the 4 most similar books to the input book.

```
similar_items =
sorted(list(enumerate(similarity_scores[index])),key=lambda
x:x[1],reverse=True)[1:5]
```

This code creates a list of tuples containing the index and similarity score of each book in relation to the input book. The list is sorted in descending order by similarity score and then the top 4 tuples are selected (excluding the input book) and stored in the similar_items variable.

The function then creates an empty list called data to store information for each of the 4 recommended books.

```
item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].value
s))
```

```
item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].valu
es))
```

```
item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].valu
es))
```

This code uses the drop_duplicates() method to remove any duplicate rows in the temp_df DataFrame based on the Book-Title column. The title, author, and image URL of the current recommended book are then extracted and appended to the item list using the extend() method.

3. Flask

This is a Python script that creates a Flask web application for recommending books based on a collaborative filtering model. Here is an explanation of each part:

1. The first few lines of the script load in some pickled data files that contain pre-trained models and book data.
2. The app variable is created as a Flask instance.
3. The index() function is defined as a Flask route that renders an HTML template (index.html) and passes in some book data to be displayed on the page. This data includes book titles, authors, cover images, number of ratings, and average ratings.
4. The recommend_ui() function is defined as a Flask route that renders another HTML template (recommend.html) which contains a form for users to input a book title.
5. The recommend() function is defined as a Flask route that takes in user input from the form and recommends similar books using the collaborative filtering model. It first retrieves the index of the input book from the pt (pivot table) data frame. Then it computes the similarity scores between the input book and all other books using the similarity_scores array. Next, it selects the top 4 similar books based on these scores, and creates a list of their titles, authors, and cover images. Finally, it passes this list of recommended books as data to be displayed on the recommend.html template.
6. The if__name__== '__main__': block runs the Flask application in debug mode when the script is executed directly (as opposed to being imported as a module).

CHAPTER 6

SCREENSHOTS AND RESULTS

6.1 Screenshots

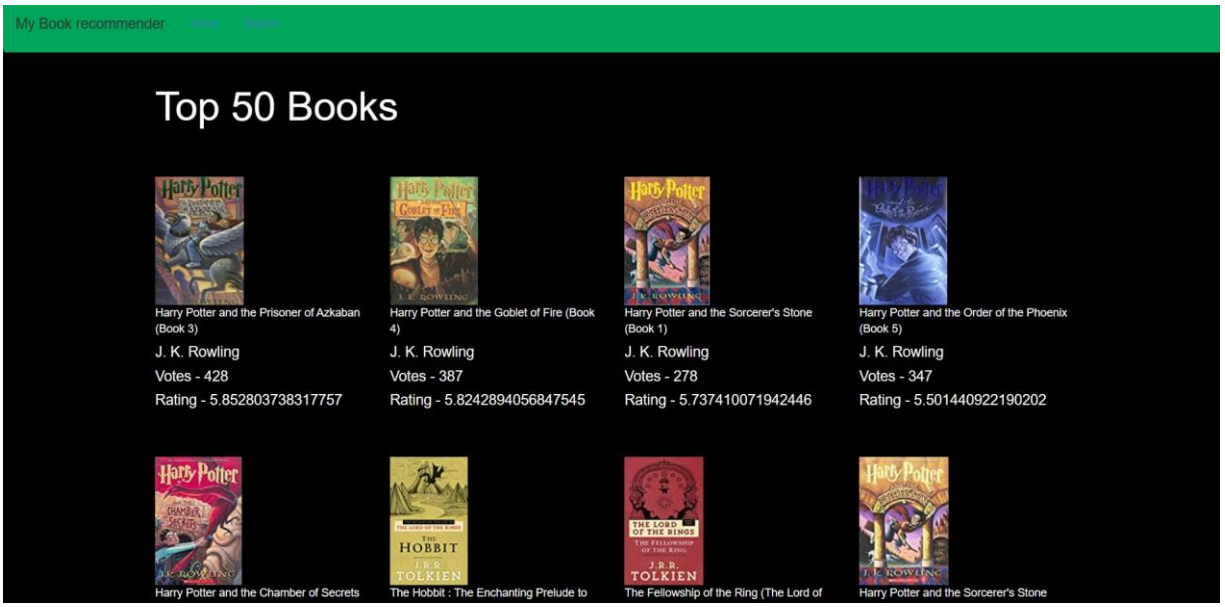

Fig 6.1 Home Page

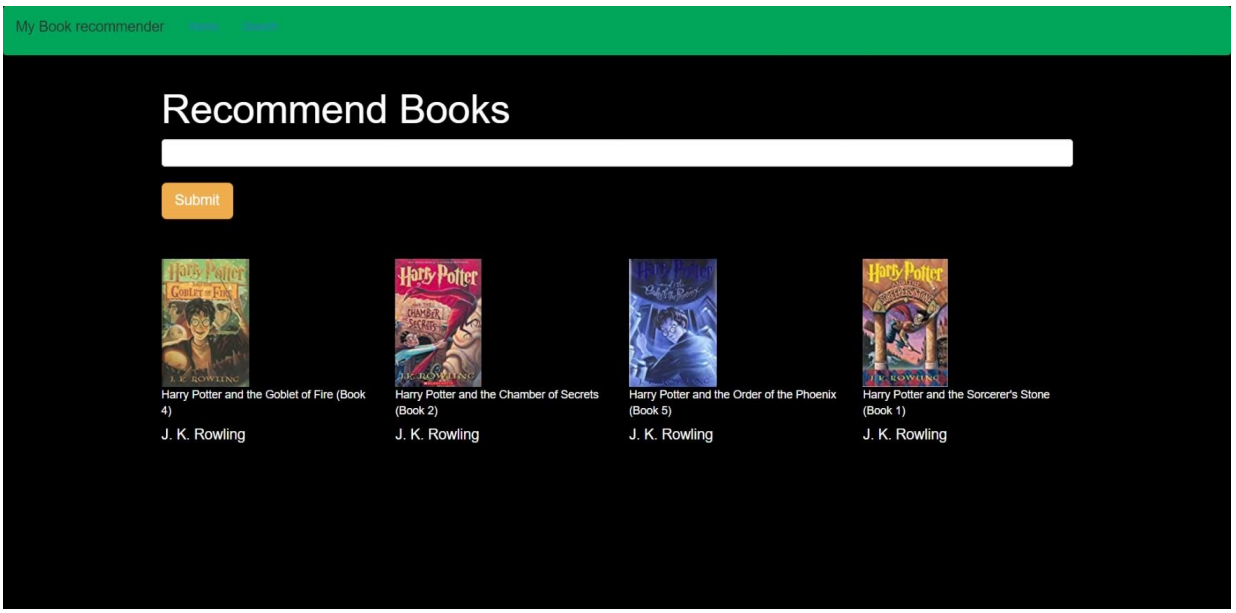

Fig 6.2 Search And Recommendation Page

6.2 Results

- The book recommendation system successfully allows users to access books with the help of Rating based, Collaborative Based and Similarity Based filtering.
- Users can get book recommendations according to the ratings and similar books .
- Collaborative Based Filtering helps users to find similar ratings and books that are voted by other users too.
- Top 50 rated books are shown to users in the home page with author name, ratings and votes by other users.
- Similar books are shown to users according to the searches that either have the same author or have the same ratings.
- Book recommendation systems help users to discover new books and encourage them to read more.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

This project involved developing a book recommendation system using K-Means Clustering, which is a Collaborative Filtering technique. The dataset used was sourced from the Goodreads website and consisted of over 3000 books. The model was built using Cosine similarity, which is a method for reducing the number of features in a dataset, and the reduced features were used to build the clustering model. Several models were built using different methods, and the model with the highest Silhouette score was chosen as the best model. The Silhouette score is a measure of how well-separated the clusters are, and a higher score indicates better clustering performance.The book recommendation system developed in this project can provide personalized recommendations for both new and existing users. The system uses clustering to group books based on their similarity, and then recommends books that are in the same cluster as the books that the user has liked.we have come to the end of the project.

Overall, this system is useful for book readers who want personalized recommendations based on their preferences. The reduced feature set and clustering approach used in this project allowed for the creation of a highly accurate recommendation system that can help users discover new books they may enjoy.

7.2 Future Enhancement

- Incorporate User Demographics: Currently, most book recommendation systems rely solely on a user's past reading history to make recommendations. However, incorporating demographic data such as age, gender, and location can help the system to better understand a user's interests and preferences, leading to more accurate recommendations.
- Integrate Social Media Data: Social media platforms such as Twitter, Facebook, and Goodreads are great sources of information on what books people are talking about and enjoying. By integrating social media data into a book recommendation system, it can provide more up-to-date and relevant recommendations based on the latest trends.
- Include Book Reviews: Including book reviews from trusted sources such as professional book reviewers, book bloggers, and literary critics can enhance the quality of recommendations provided by the system. The system can analyze these reviews to better understand what types of readers would enjoy a particular book.
- Use Natural Language Processing: Natural language processing (NLP) can help to better understand user preferences by analyzing the language used in book reviews and other textual data. By analyzing the language used to describe books, the system can better understand the themes and topics that appeal to a particular user.
- Allow User Feedback: Finally, allowing users to provide feedback on the recommendations provided by the system can help to improve its accuracy over time. By asking users to rate books they have read and whether they enjoyed the recommendations made, the system can learn from user feedback and adjust its recommendations accordingly.

REFERENCES

[1] Avi Rana and K. Deeba et.al, “Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)” in IOP ebooks 1362, 2019.

[2] G. Naveen Kishore, V. Dhiraj, Sk Hasane Ahammad, Sivaramireddy Gudise, Balaji Kummaraa and Likhita Ravuru Akkala, “Online Book Recommendation System” International Journal of Scientific & Technology Research vol.8, issue 12, Dec 2019.

[3] Uko E Okon, B O Eke and P O Asaga, “An Improved Online Book Recommender System using Collaborative Filtering Algorithm”, International Journal of K-Means Gaussian mixture Silhou 0.0433968332 0.01677887231 ette 584411 3688933 Score Computer Applications vol.179-Number 46, 2018.

[4] Jinny Cho, Ryan Gorey, Sofia Serrano, Shatian Wang, JordiKai Watanabe-Inouye, “Book Recommendation System” Winter 2016.

[5] Ms. Sushma Rjpurkar, Ms. Darshana Bhatt and Ms. Pooja Malhotra, “Book Recommendation System” International Journal for Innovative Research in Science & Technology vol.1, issue 11, April 2015.

[6] Abhay E. Patil, Simran Patil, Karanjit Singh, Parth Saraiya and Aayusha Sheregar, “Online Book Recommendation System using Association Rule Mining and Collaborative Filtering” International Journal of Computer Science and Mobile Computing vol.8, April 2019.

[7] Suhas Patil and Dr. Varsha Nandeo, “A Proposed Hybrid Book Recommender System” International Journal of Computer Applications vol.6 – No.6,Nov – Dec 2016.

[8] Ankit Khera, “Online Recommendation System” SJSU ScholarWorks, Masters Theorem and Graduate Research, Master’s Projects, 2008.

[9] Anagha Vaidya and Dr. Subhash Shinde, “Hybrid Book Recommendation System” International Research Journal of Engineering and Technology (IRJET), July 2019.

[10] Dhirman Sarma

, Tanni Mittra and Mohammad Shahadat Hossain, ”Personalized Book Recommendation System using Machine Learning Algorithm” The Science and Information Organization vol.12,2019