uOttawa

SYS 5160 Systems Integration

Final Report: Recommendation System

Professor: Dr. Tet Yeap
Yupeng Guo 300152481

Table of Contents

Executive Summary

A recommendation system is an information filtering system used to predict the users' rating and preference of items. There are mainly two recommendation systems: content-based filtering and collaborative filtering. The project utilizes data mining and machine learning to develop the recommendation system. In content-based filtering, the project implements two classification models to label the category of books and make predictions. In collaborative filtering, the project uses memory-based and model-based algorithms to calculate the predicted rating. In addition, this project implements python-Django framework to simulate a book recommender web application using Book-Crossing dataset.

Index items: recommendation system, content-based filtering, collaborative filtering, machine learning, python, Django framework

1. Introduction

This project used the Book-Crossing dataset that was collected by Cai-Nicolas Ziegle and can be retrieved from http://www2.informatik.uni-freiburg.de/~cziegler/BX/. The original dataset contains 981757 ratings, 10001 unique books, and 278859 unique users. Considering the sparsity of the dataset that some users have less ratings or some books were rarely reviewed by users, I filtered out some books that the number of ratings is less than 50. This left us with 24135 ratings, 811 unique books and 2134 unique users into training, development and evaluation. In addition, I retrieved book descriptions based on the corresponding ISBN from the Google Books API: https://www.googleapis.com/books/v1/volumes?q=isbn:yourISBN. These descriptions, along with other book information, were used in the content-based filtering. The recommendation system mainly relied on the information about user_id, book_id, ratings and descriptions.

2. Data preprocessing: text cleaning and feature extraction (BOW)

By definition, data preprocessing is a data mining technique that involves transforming raw texting data into an understandable format for mathematical modeling, which includes data cleaning and feature extraction [6]. For the data cleaning, the system first converted all the letters into lower case. Then it removed punctuations from each chunk and tokenized the documents to remove stopwords. Stopwords refers to those in any language that are meaningless to a sentence. In other words, removing stopwords has no influence on the meaning of the sentence. The system relied on the 'nltk' library to identify stopwords. Lastly, the system did the lemmatization to transform words into their base form according to the intended meaning and context surrounding that sentence.

Aftering the data cleaning, the system used Bag of Word (BOW) to extract the features from the content description to identify the similar books in the dataset. By the definition, the BOW model is a simplifying representation used in natural language processing in which a text is represented as the bag of its words, only considering the frequency of words and disregarding grammar and work order. The system assumes that the text of the content description contains keywords that can be used to find out similar books. Firstly, it built up a vocabulary based on the dataset. Each word in the vocabulary occurs once and represents one feature of the book. Then it counts the frequency of words in the content description to generate vectors for different books within the dataset. Finally, the system can get feature matrix shape, which is denoted as *Number of book * Length of vocabulary*. However, the BOW model exists the dimensionality problem that the more books the system includes the larger the vocabularies are. As a result, the feature matrix becomes huge and sparse, which may affect the efficiency of feature extraction. Therefore, the system minimized the length of vocabulary by cleaning the text, removing stopwords and doing lemmatization in previous steps to mitigate the dimensionality issue with the BOW model.

3. The recommendation System

There are two recommendation models: content-based filtering and collaborative filtering. Content-based filtering tries to make recommendations on the basis of similar items according to the attributes instead of users' behaviors or preferences. But collaborative filtering tries to group similar users together based on users' preferences and behaviors and make predictions. As shown in figure 2, user A and user B are defined as similar users based on historical data. Then the model will recommend items that are favored by user A to user B.

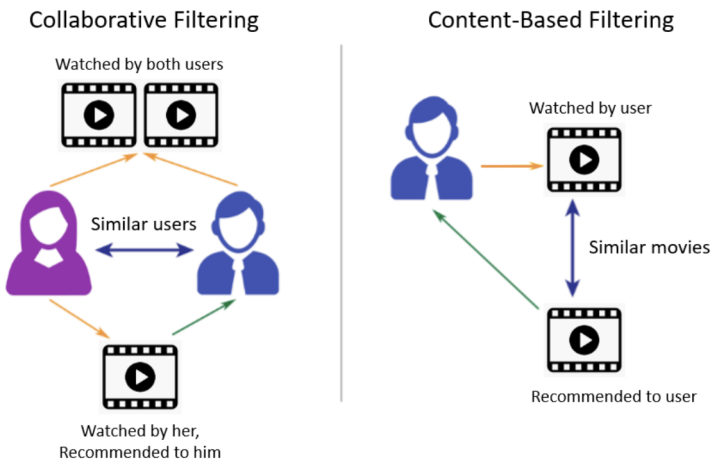


Figure 1: comparison between collaborative and content-based filtering

3.1 Content-based filtering

The sklearn python library provides functionality for supervised neighbors-based learning methods [2]. It is possible to do the classification based on the book content description and its corresponding categories. We can employ the classification models to label different categories based on the extracted feature from the BOW model. The system implements two classification models to predict the categories of the books: SVM and KNN.

Support Vector Machines (SVM): SVM is a supervised machine learning model that is used for classification and regression analysis. SVM takes text features into account and constructs a hyperplane (i.e., decision boundary) which can best separate different categories.

K-Nearest Neighbors (KNN): the assumption of KNN is that similar items are close to each other. For instance, there are two existing categories: A and B. A new data point is incorporated. We can decide the value of K, which means how many closest neighbors we consider in this process. If K = 5, then the algorithm calculates the *Euclidean distance* between new data points and other data points. After that, it selects 5 nearest neighbors. 3 out of 5 data points belong to category A compared to 2 in category B. Hence, we can assign this new data point to category A.

We can divide books into groups based on the categories: fiction, biography and social science. The confusion matrix of the two classification models are show below (figure 2):

		Confusion Matrix
SVM	Accuracy	1 5 4 | a = biography
	0.62	5 62 42 | b = social sicence
		1 35 86 | c = fiction
KNN	Accuracy	1 1 8 | a = biography
	0.57	0 33 76 | b = social sicence
		0 18 104 | c = fiction

Figure 2: confusion matrix of SVM and KNN model

After classifying the books into different categories, we can calculate the *Cosine Similarity* between different books in the same categories. If we input a book ID, we can get a list of similar books from the same category. For instance, if we want to find some similar books of *"Harry Potter and the Goblet of Fire"*, the system can return a list of similar books based on the book ID (figure 3).

```
def get_id_from_index(index):
    book_id = book_info_bow[book_info_bow.index == index]["book_id"].values[0]
    title = book_info_bow[book_info_bow.index == index]["title"].values[0][0]

    return (f'{book_id}: {title}')
i=0
for product in sorted_similar_products:
    print(get_id_from_index(product[0]))
    i=i+1
    if i>15:
        break
```

```
6: Harry Potter and the Goblet of Fire (Harry Potter, #4)
5: Harry Potter and the Prisoner of Azkaban (Harry Potter, #3)
2: Harry Potter and the Order of the Phoenix (Harry Potter, #5)
1: Harry Potter and the Half-Blood Prince (Harry Potter, #6)
6689: James and the Giant Peach
3590: The Adventures of Sherlock Holmes
3: Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
2677: A Modest Proposal and Other Satirical Works
5094: The Drawing of the Three (The Dark Tower, #2)
6749: Oblivion
7036: The Kalahari Typing School for Men (No. 1 Ladies' Detective Agency, #4)
8698: So Long, and Thanks for All the Fish (Hitchhiker's Guide to the Galaxy, #4)
7996: Redwall (Redwall, #1)
8155: A Woman of Substance (Emma Harte Saga #1)
3985: The Amazing Adventures of Kavalier & Clay
2183: Sentimental Education
```

Figure 3: outputs of content-based filtering

3.2 Collaborative filtering

Collaborative filtering using memory-based algorithm

For collaborative filtering, it calculates the similarities of the users to make predictions based on either memory-based or model-based algorithms. Before doing prediction, we need to create a matrix according to the book ratings. The rows in the matrix represent the ratings of books for each user and the columns refer to the ratings by users for each book. Since some books haven't been rated by some users, the default rating is set to be 0. Assuming the columns in the matrix as the vectors, the distance between the vectors refer to the similarities between the users. To determine the similarities of the vectors, we can implement *Cosine Similarity* which is a common n-dimensional distance metric to calculate the distance between two vectors. But there is one circumstance that needs to be considered. One user may give all books quite high ratings, such as 4 and 5 (range from 1 to 5). Another user might give all ratings between 3 and 4 for the same books. These two users might have similar preference. In order to alleviate the potential prediction error and improve the performance, we can normalize the ratings by 'average rating'.

Specifically, each rating value subtracts the average rating value of that user to get a new rating value.

In order to evaluate the performance of the prediction, one of the methods is to calculate the *root mean square error (RMSE)* of the predicted rating values in the testing set. The system uses a *sklearn* python library to generate training and testing subset from the original dataset. 80% of the dataset is the training and 20% is the testing. By the definition of RMSE, it is always non-negative with a value of 0 to infinite based on the data. The value of 0 indicates a perfect fit to the data. In general, a lower RMSE is better than a higher one. The RMSE of this model are shown below:

User-based CF RMSE	3.953640025
Item-based CF RMSE	3.953667845

The system gets RMSM = 3.9 in both user-based and item-based methods. The *Mean Square Error* (MSE) can be calculated as 0.79 (3.95 / 5 = 0.79 > 0.6). This is pretty bad. Although memory-based collaborative filtering is easy to implement, the predictions didn't reflect the preference of the users.

Collaborative filtering using model-based algorithms

Compared with memory-based collaborative filtering in which it uses all the data in the database to generate a prediction, the model-based collaborative filtering uses the data in the dataset to create a model that can be used to generate predictions [7]. Also model-based methods are inherently more scalable and can deal with higher sparsity levels than memory-based method [3]. We use *Matrix Factorization-based* algorithms that are built in the *surprise* python library [5]. The library contains several model-based algorithms, such as *Singular Value Decomposition* (SVD) and *Non-negative Matrix Factorization* (NMF). In addition, we use 5 folds cross-validation to create the models.

Evaluating RMSE of NMF model							
	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Mean**	**Std**
RMSE	1.0639	1.0551	1.0621	1.0779	1.0737	1.0665	0.0082
Fit time	5.21	5.07	5.14	5.16	5.63	5.24	0.2
Test time	0.09	0.15	0.08	0.09	0.08	0.1	0.03
Evaluating RMSE of SVD model							
	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Mean**	**Std**
RMSE	0.9265	0.9161	0.9196	0.9236	0.9233	0.9218	0.0036
Fit time	2.97	2.96	2.96	2.96	2.94	2.96	0.01
Test time	0.12	0.1	0.1	0.21	0.1	0.13	0.04

Figure 4: RMSE on five splits in NMF and SVD model

The figure 4 above illustrates the RMSE in SVD and NMF models. It indicates that using a 80% train-test split, the SVM and NMF model get the RMSE = 0.9218 and 1.0665. And the MSE in SVD and NMF models are 0.1843 (0.9218 / 5) and 0.2133 (1.0665 / 5) respectively. It shows that using model-based algorithms can get better performance than using memory-based algorithms in prediction, 0.1843 and 0.2133 versus 0.79 in MSE. Also it shows that the SVD algorithm may be the better option for this dataset (0.1843 versus 0.2133 in MSE).

According to the results of performance evaluation, the system utilizes the SVD algorithm to estimate the rating of books for each user. Then the system can recommend books to the users based on the user ID. For instance, if one user ID is '18313', the system can return a list of recommended books with the predicted ratings as following (figure 5):

```
example_reading_list = get_reading_list(userid=18313)
```

```
Heretics of Dune (Dune Chronicles #5): 4.659935588249317
Harry Potter Collection (Harry Potter, #1-6): 4.46253404261773
The Lord of the Rings (The Lord of the Rings, #1-3): 4.389361092470132
Heidi: 4.343556342318165
The Lord of the Rings: The Art of The Fellowship of the Ring: 4.263801002150561
Moon Palace: 3.7137673187820885
```

Figure 5: outputs of model-based collaborative filtering in google colab

4. System stimulation

In this section, We will discuss the system architecture and the system web design. The book recommendation web application using Django framework. Django is an open source web framework which is employed in many commercial environments. Compared with Node.js in Javascript, Django is easy to use, more stable and flexible because it takes advantage of multiple libraries available in python. Other web technologies in this project are used including HTML5, css Javascript and SQlites (database).

4.1 System Architecture

The figure below displays the client/server architecture applied to the recommendation web application. The end users can use the browser to send the http GET/POST requests to the server. The browser displays the recommended results after the server receiving the request and sending the data back. In the server side, it contains the Django server, recommendation system with content-based and collaborative filterings, and SQlite database. The Django server can

handle the http requests from the browser, retrieve data from the database and connect to the recommendation system. The database contains the Book-Crossing dataset. For instance, while the user sends the http Get request with the user_id = 18313 to the server, the server connects to the recommendation system, the recommendation system obtains the dataset from the database, does calculations and returns the list of recommended books to the server. The server sends the data back to the browser.

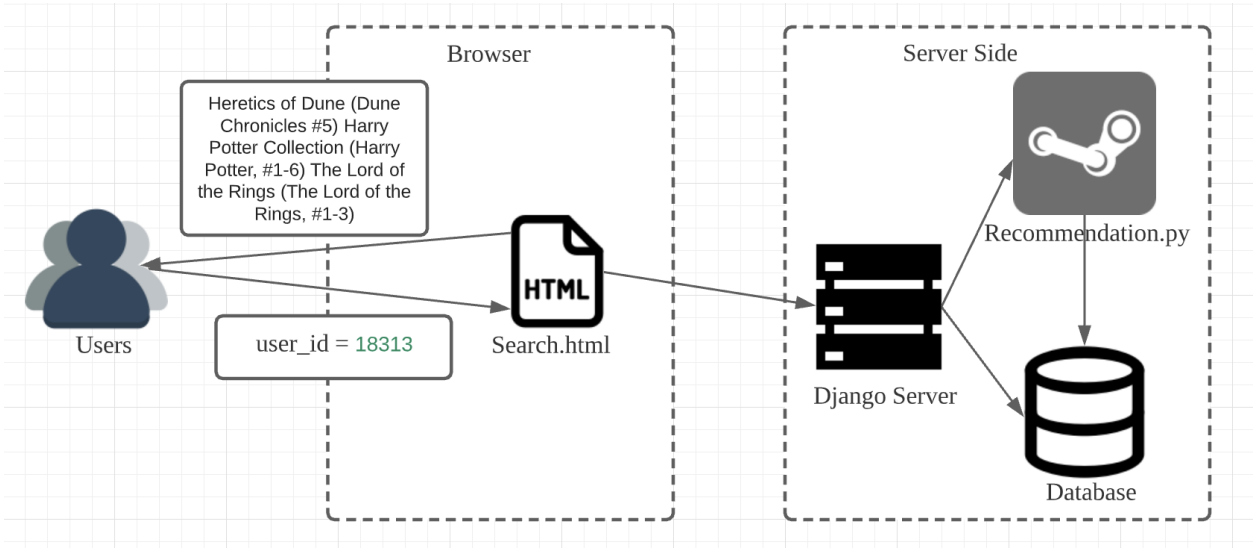
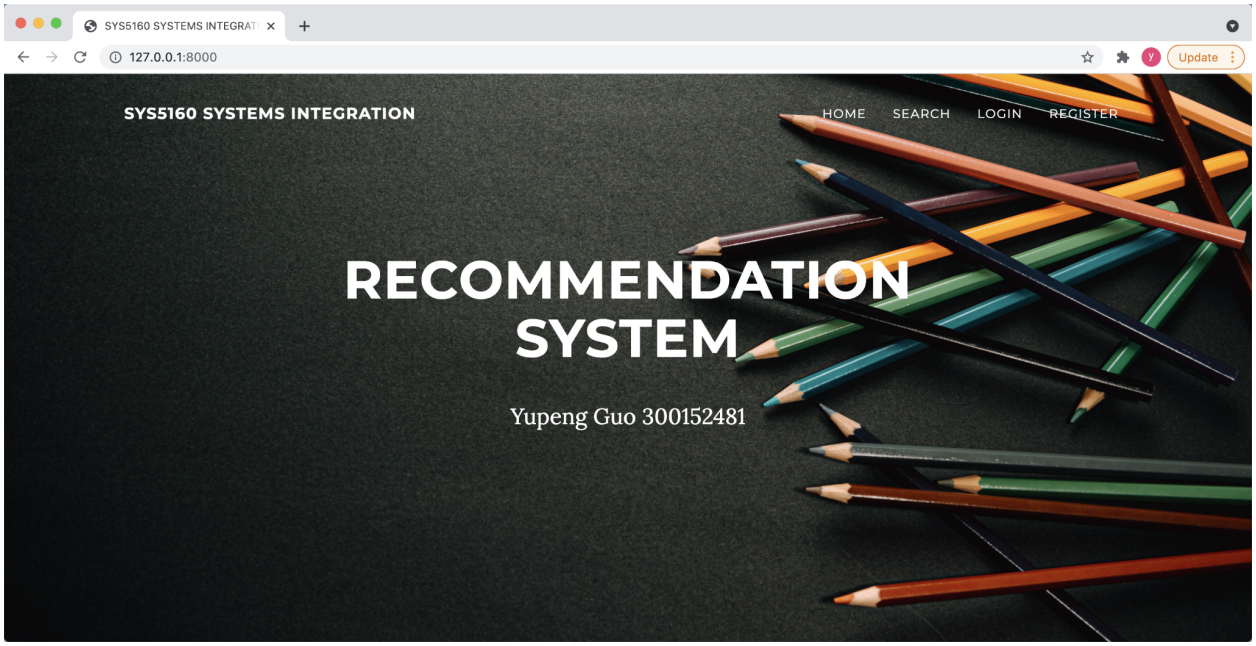


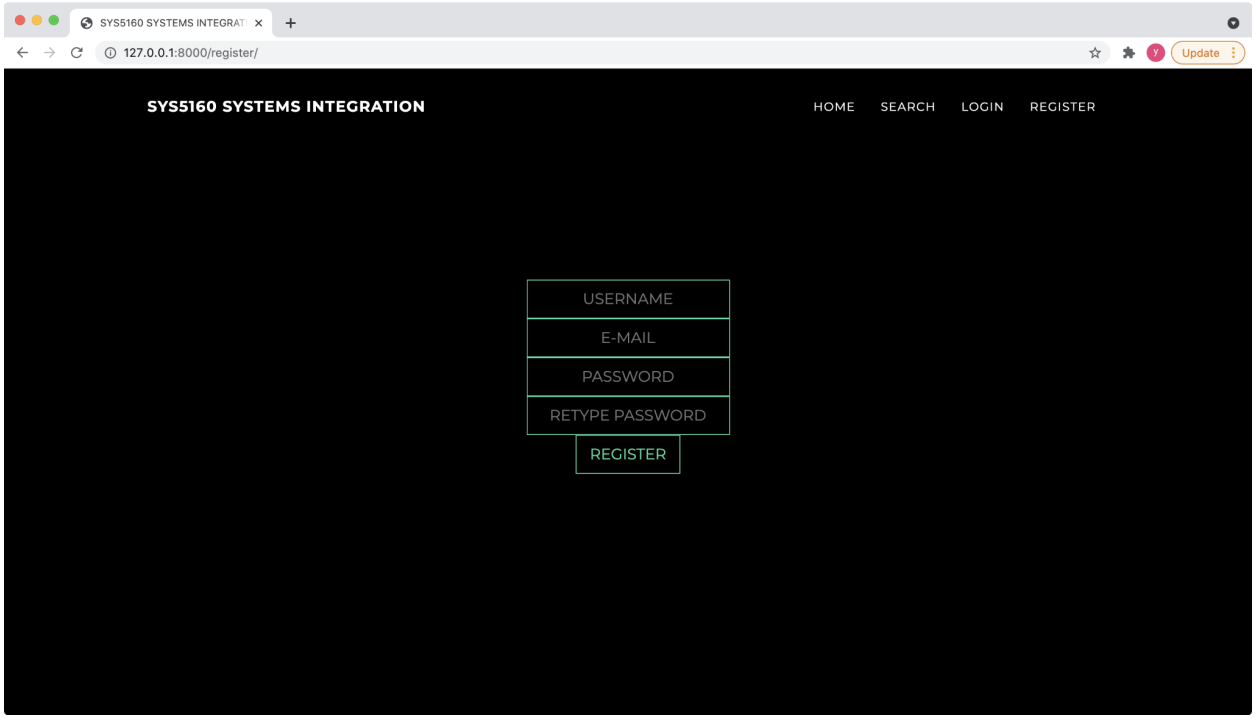
Figure 6: the client-server architecture of basic recommendation web application

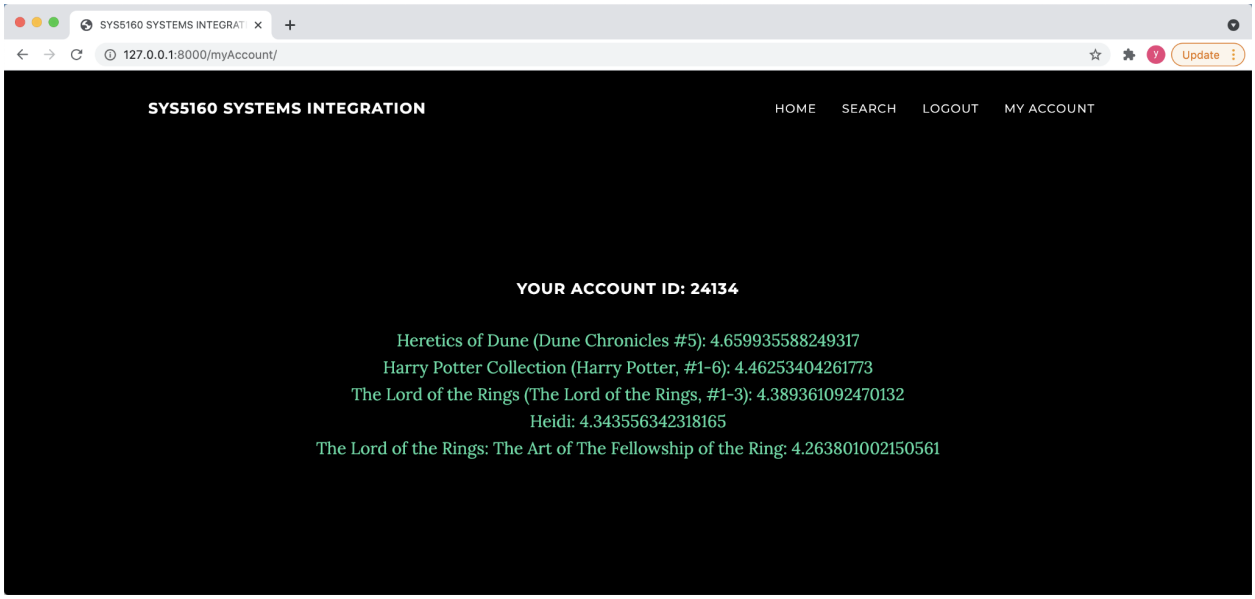
4.2 System Web Design

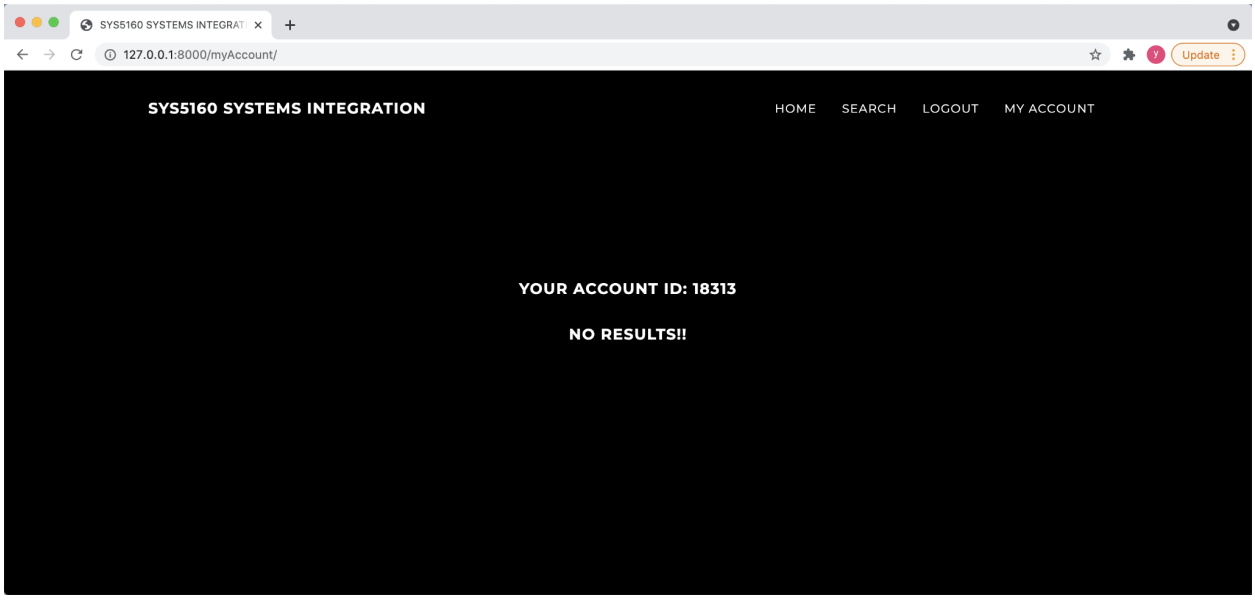
The following figure shows the home page of the web application of the recommendation system. After deploying the project according to the README.txt file, you can run the Django server in the console, open the browser and go to http://127.0.0.1:8000/.



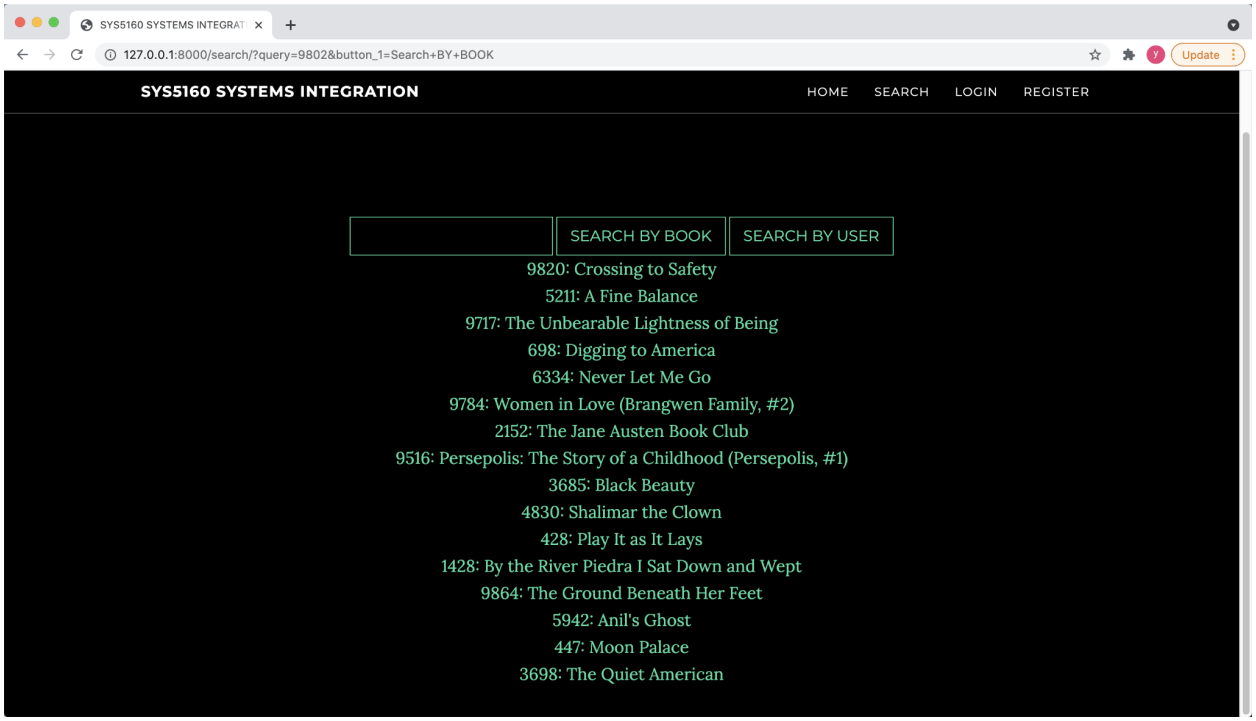
The web application allows the user to register and login into the system. You can go to the register page to register your information. After you logged in, if the dataset contains your information, the myAccount page can show the recommended books based on your user ID. If the dataset didn't contain your information, the myAccount page will show 'No result'.




SYS5160 SYSTEMS INTEGRAT
127.0.0.1:8000/myAccount/
Update
SYS5160 SYSTEMS INTEGRATION
HOME
SEARCH
LOGOUT
MY ACCOUNT
YOUR ACCOUNT ID: 24134
Heretics of Dune (Dune Chronicles #5): 4.659935588249317
Harry Potter Collection (Harry Potter, #1-6): 4.46253404261773
The Lord of the Rings (The Lord of the Rings, #1-3): 4.389361092470132
Heidi: 4.343556342318165
The Lord of the Rings: The Art of The Fellowship of the Ring: 4.263801002150561


SYS5160 SYSTEMS INTEGRAT
127.0.0.1:8000/myAccount/
Update
SYS5160 SYSTEMS INTEGRATION
HOME
SEARCH
LOGOUT
MY ACCOUNT
YOUR ACCOUNT ID: 18313
NO RESULTS!!

The recommendation system contains the content-based filtering and collaborative filtering. In the search page, you can search similar books based on content-based filtering. Also you can search relevant books that you may interest in using the user ID.



5. Conclusion

This project developed a web application in recommendation system using Book-Crossing dataset. In the recommendation system, we implement content-based and collaborative filtering to make predictions. I found that the performance and accuracy of prediction depends on the quantity and quality of the original dataset. Training a machine learning model on an imbalance dataset can affect the accuracy of classification and prediction. In future, I will try to use different databases to further improve the algorithms of this recommendation system.

Reference

[1]. Z. Cai-Nicolas, “Book-Crossing Dataset”. Retrieved from: http://www2.informatik.uni-freiburg.de/~cziegler/BX/

[2]. Scikit-learn developers, Nearest Neighbors. Retrieved from: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors

[3]. M. Tom, “Book-Crossing Recommender System”. Retrieved from https://github.com/tttgm/fellowshipai

[4]. Google Developer, “Google Book APIs”. Retrieved from: https://developers.google.com/books/docs/v1/using

[5]. Surprise, Matrix Factorization-based algorithms. Retrieved from https://surprise.readthedocs.io/en/stable/matrix_factorization.html

[6]. S. Mohit, “Data Preprocessing: 6 Necessary Steps for Data Scientists”. Retrieved from https://hackernoon.com/what-steps-should-one-take-while-doing-data-preprocessing-502c993e1caa

[7]. P. H. Aditya, I. Budi, Q. Munajat, “A Comparative Analysis of Memory-based and Model-based Collaborative Filtering on the Implementation of Recommender System for Ecommerce in Indonesia : A Case Study PT X”. *IEEE*. 2016

[8]. C.Jinny, G. Ryan, S. Sofia, W. Shatian, W. JordiKai, “Book Recommendation System”.