CS512 Book Recommendation System - Fall 2018

Akshay Sovani
Rutgers University
Piscataway, NJ, USA
Email: as3041@rutgers.edu

Sharvani Pratinidhi
Rutgers University
Piscataway, NJ, USA
Email: sharvani.pratinidhi@rutgers.edu

Ajinkya Patankar
Rutgers University
Piscataway, NJ, USA
Email: ajinkya.patankar@rutgers.edu

***Abstract*— Recommendation Systems in general are used to give suitable recommendations to users depending upon their liking, age, qualities and other conditional factors. Recommendation systems use a large dataset to recommend entities similar to the input entities. The objective of our project is to provide a list of books similar to the ones that the users have inputted.**

I. Project Description

The Book Recommendation System recommends a list of books to the end user. The recommendation is based on the ratings of the books already present in the system. The system inputs ratings for books from the user, finds the most similar books and creates a list of them and provides the list back to the user. This is achieved through matrix factorization using alternating least squares (ALS) algorithm for collaborative filtering. ALS algorithm has been proven to achieve good performance.

The specifications for the project are as follows:

Project Type: Implementation of algorithm not covered in the class.
Project usefulness: It enables end users to quickly find the books of their liking. If they enter their favourite book, they will get back a list of new books which they are highly likely to find interesting.
Project feasibility: The project can be implemented in one semester.
Stumbling block: Finding appropriate dataset.

A. Stage1 - The Requirement Gathering Stage.

- Types of users are as follows:

  a. **Book Seller**
  Real world scenario:
  Consider the following scenario :
  1. Jack is a book seller who wants to increase his sales by recommending books to his customers. He takes a survey for his customers and receives rating from them. Then he passes those ratings to the system.
  2. The system will identify customer who has given similar ratings as of the customers of Jack.
  3. Then, the system will output similar books for individual customers of Jack.
  4. Jack will sell those recommended books to his customers and increase his sales.

  System data input: Rating for books by the customers of Jack.
  System input data type: bookID: String, userID: String, Rating: String
  System data output: Books similar to the books accessed by the customers.
  System output data type: Input bookID: String ,List of recommended similar booksID: List[]

  b. **A Book enthusiast(reader/critique)**
  Real world scenario:
  Consider the following scenario:
  1. Jenny is a member of a book club and wants to read more books. She inputs the ratings of books she liked in the system.
  2. The system will identify other users who have given similar ratings to books as Jenny.
  3. Then, the system will output the list of similar books to Jenny.

  System data input: Rating for books by Jenny.
  System input data type: bookID: String, userID: String, Rating: String
  System data output: List of books similar to the books entered by Jenny.
  System output data type: Input bookID: String ,List of recommended similar booksID: List[]

  c. **Jim wants to gift a book to Joe**
  Real world scenario:
  Consider the following scenario:
  1. Joe is an avid book reader to whom Jim wants to gift a book for his birthday.
  2. Jim identifies the book that Joe liked the most and entered that into system. The system will identify other users who have given similar ratings to books as Jim.
  3. Then, the system will output the list of similar books to Jim and Jim will gift one of them to Joe.

  System data input: Rating for books by Jim.
  System input data type: bookID: String, userID: String, Rating: String
  System data output: List of books similar to the books

entered by Jim.
System output data type: Input bookID: String ,List of recommended similar booksID: List[]

- Time line for completion of the major implementation stages:
  Stage 1: Requirement Gathering - by Oct 27
  Stage 2: Design - by Nov 10
  Stage 3: Infrastructure Implementation - by Nov 24
  Stage 4: User Interface - by Dec 8

  Labour Division:
  Akshay Sovani : Formualting the problem statement, Implementation of ALS algorithm, Documentation, Analysis of algorithm

  Sharvani Pratinidhi : Formualting the problem statement, Implementation of ALS algorithm, Documentation, Design of user interface

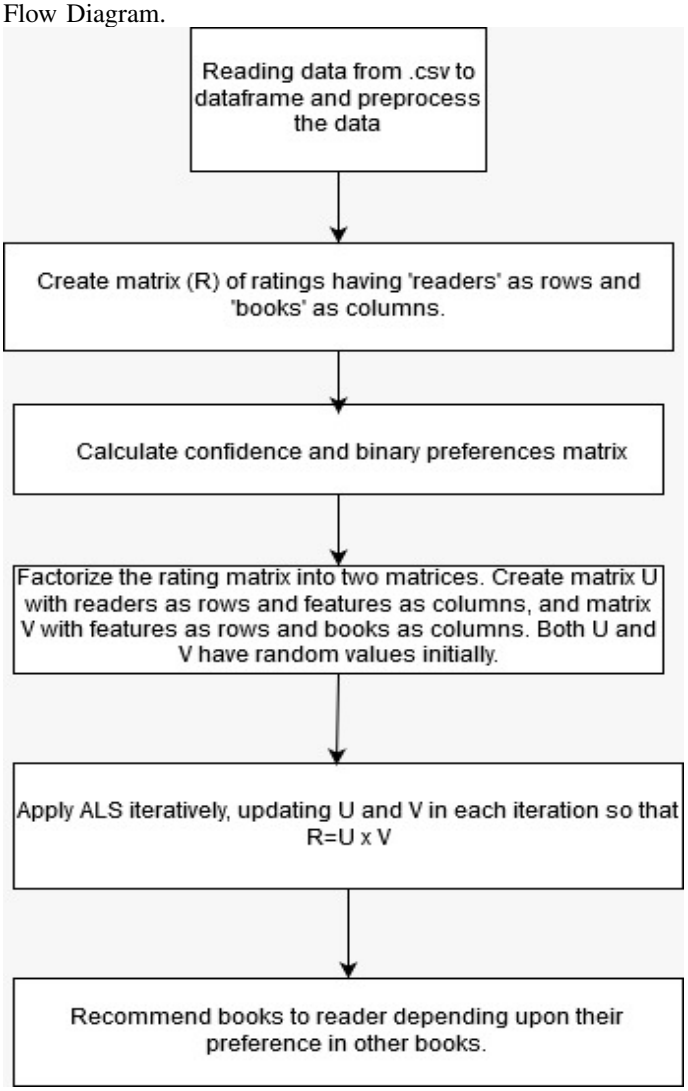
  Ajinkya Patankar : Formualting the problem statement, Data acquisition and preprocessing, Implementation of ALS algorithm, Documentation

B. Stage2 - The Design Stage.

- Short Textual Project Description.
  The Book Recommendation System recommends a list of books to the user depending upon their preferences. The recommendation is done through collaborative filtering which predicts the choice of a user based on many others users who have made similar choices in the past. The data used for training the recommendation system consists of users and the ratings given by them to a plethora of books. Additionally, other details of the book such as its Title, ISBN number, Author,etc and those of users such as their age and location are also available. These details are used as a lookup table while providing the list of Titles of books.

  The initial step is to read the data from its csv file into the program as a dataframe, preprocess and clean the data, and then transform it to a sparse matrix. This sparse matrix (R) contains the ratings given by users(rows) to the books(columns). The preprocessing stage includes removing the rows with all null values, deleting duplicate entries of books,etc.

  The next step is to calculate the confidence and preference of each user for each book. Preference is expressesd as a binary and describes whether the user has read and rated the book, or whether there is no information about the users rating of the book(i.e null values). Preference is important as the null values do not indicate that the user dislikes the book. The confidence is a scaled up form of the rating given by the user. The rating needs to be scaled up so that the difference between two adjacent ratings is significant.
  Then, the matrix of ratings(R) is factorized into two matrices (U and V). U and V have random values initially. The Alternating Least Squares algorithm works on these matrices iteratively to bring their product as close as possible to R matrix. The list of books is recommended by calculating the dot product of the user vector with the transpose of the book matrix (V transpose).
- Flow Diagram.



- High Level Pseudo Code System Description.

  Pseudo code is as follows:
  1. Import all the necessary libraries for the algorithm. These are: Pandas, numpy, scipy.sparse, sklearn.preprocessing

  2. Read the .csv file into Ratings dataframe.
  cols = [reader_id,book_id,ratings]
  Ratings = pd.read_table(ratings.csv,columns=cols)

3. Drop the rows with null values.
Ratings = Ratings.dropna()

4. Generation of sparse matrix to hold the ratings given by readers to books:
rows = Ratings.reader_id.astype(int)
columns = Ratings.book_id.astype(int)
Rating_Matrix = sparse.csr_matrix((ratings, (rows, columns)),
shape=(len(reader_id), len(book_id)))

5. Define a function ALS.
Input parameters: The matrix Ratings and an int number_of_iterations i.e. Number of times the vectors U and V will be updated.
Output: U and V matrices where U and V are vectors for users and books respectively.

ALS (Rating_Matrix, number_of_iterations, alpha_val){

1. Create a Confidence matrix by multiplying all the values in the Rating_Matrix by an integer value.
Confidence_matrix = Rating_Matrix * alpha_val
2. Create two matrices:
(size: users by features) and V(size: books by features)
3. For i in xrange(number_of_iterations):

yTy = Y.T.dot(Y)
xTx = X.T.dot(X)

reader_size, book_size = Rating_Matrix.shape()
for u in xrange(reader_size):

Get the user row from Confidence_matrix[u,:]
Calculate the binary preference p(u)
Calculate Cu and Cu - I
Put all the values in the spsolve() and compute X[u].

for u in xrange(book_size):
Get the book column from Confidence_matrix[:,i].T.toarray()
Calculate the binary preference p(i)
Calculate Ci and Ci - I
Put all the values in the spsolve() and compute Y[u].

Return the vectors X and Y.
Return X,Y
}

6. Using the vectors X and Y returned by ALS() function, recommend a vector of books to user i.

- Time and space complexity Space and Time Complexity:
U: No of users/readers
B: No of Books
1) Import data from source CSV file into python dataframe.
Time Complexity:- constant time (0), Space Complexity:- O(UxB)

2) Create Matrix of order UxB
Time Complexity:- O(UxB), Space Complexity:- O(UxB)

3) Calculate Preference Matrix of order same as UxB since we calculate it by the form:
P(i,j) = 1 if R(i,j) 0
$0ifR(i,j)0$

Therefore, Time Complexity:- O(UxB), Space Complexity:- O(UxB)

Calculate Confidence Matrix of same order by the formula.
C(i,j) = 1 + alpha*(R(i,j))
where alpha is the scaling factor. Therefore, Time Complexity: O(UxB), Space Complexity:- O(UxB)

4) Create matrices U,V of order Uand Fwhere U=users, B=books and F=features by using functions from python's random library.

Time Complexity:- O(U) + O(F)

5) Calculate the minimum loss function by alternating least square(ALS) method.

Time Complexity: O(b$\hat{3}$), Space Complexity: O(U)

6) Recommendation of vector of similar books

Time Complexity: O(U), Space Complexity: O(1)

Total Time Complexity:- O(B$\hat{3}$)
Total Space Complexity:- O(UxB)

- Algorithms and Data Structures. The dataset is stored and manipulated in the matrix which is essentially a 2D array in python. The matrix is factorized and the loss function is minimized using Alternating Least Squares Algorithm to compute two matrices.

These two matrices are updated with every iteration by updating the individual user and item vectors(Xu and Yi respectively) in them. These two vectors are as follows:

$$x_u = (Y^T Y + Y^T (C^u - I)Y + \lambda I)^{-1} Y^T C^u p(u)$$

$$y_i = (X^T X + X^T (C^i - I)X + \lambda I)^{-1} X^T C^i p(i)$$

Xu : It denotes a vector for user u.

Yi: It denotes vector for book i.

Vector Ui has been multiplied with transpose of the vector V to get the recommendation score which is used to recommend the vector of similar books to the book of a particular reader. Finally, using this score, a vector of similar books is calculated and returned.

Thus, a reader gets number of books similar to the book he/she reads.

- Flow Diagram Major Constraints.
  The matrix R must be created properly in order to correctly recommend a similar book to the reader. Else, the algorithm might recommend a book that the reader may not like to read.

  – Integrity Constraint.
    While applying ALS(), number of iterations should be decided depending on the data. Either too less iterations or too many iterations could affect the resultant recommended book vector.

C. Stage3 - The Implementation Stage.

Language used : Python and programming environment : Jupyter notebook

- Sample small data snippet.

	A	B	C
1	User-ID	ISBN	Book-Rating
2	276725	034545104	0
3	276726	1.55E+08	5
4	276727	4.47E+08	0
5	276729	052165615	3
6	276729	5.22E+08	6
7	276733	2.08E+09	0
8	276736	3.26E+09	8
9	276737	6.01E+08	6
10	276744	038550120	7
11	276745	3.42E+08	10
12	276746	4.25E+08	0
13	276746	4.49E+08	0
14	276746	5.54E+08	0
15	276746	055356451	0
16	276746	7.86E+08	0
17	276746	7.86E+08	0
18	276747	60517794	9
19	276747	4.51E+08	0
20	276747	6.1E+08	0
21	276747	6.72E+08	9
22	276747	6.8E+08	8

-

	A	B	C	D	E	F	G	H	I	J	K
1	ISBN;"Book-Title";"Book-Author";"Year-Of-Publication";"Publisher";"Image-URL-S";"Image-URL-M";"Image-URL-L"										
2	0195153448;"Classical Mythology";"Mark P. O. Morford";"2002";"Oxford University Press";"http://images.amazon.com/										
3	0002005018;"Clara Callan";"Richard Bruce Wright";"2001";"HarperFlamingo Canada";"http://images.amazon.com/imag										
4	0060973129;"Decision in Normandy";"Carlo D'Este";"1991";"HarperPerennial";"http://images.amazon.com/images/P/0(										
5	0374157065;"Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the Virus That Caused It";"Gina E										
6	0393045218;"The Mummies of Urumchi";"E. J. W. Barber";"1999";"W. W. Norton & Company";"http://images.ama										
7	0399135782;"The Kitchen God's Wife";"Amy Tan";"1991";"Putnam Pub Group";"http://images.amazon.com/images/P/0:										
8	0425176428;"What If?: The World's Foremost Military Historians Imagine What Might Have Been";"Robert Cowley";"20(										
9	0671870432;"PLEADING GUILTY";"Scott Turow";"1993";"Audioworks";"http://images.amazon.com/images/P/06718704										
10	0679425608;"Under the Black Flag: The Romance and the Reality of Life Among the Pirates";"David Cordingly";"1996";"R										
11	074322678X;"Where You'll Find Me: And Other Stories";"Ann Beattie";"2002";"Scribner";"http://images.amazon.com/im										
12	0771074670;"Nights Below Station Street";"David Adams Richards";"1988";"Emblem Editions";"http://images.amazon.cc										
13	080652121X;"Hitler's Secret Bankers: The Myth of Swiss Neutrality During the Holocaust";"Adam Lebor";"2000";"Citadel										
14	0887841740;"The Middle Stories";"Sheila Heti";"2004";"House of Anansi Press";"http://images.amazon.com/images/P/0										
15	1552041778;"Jane Doe";"R. J. Kaiser";"1999";"Mira Books";"http://images.amazon.com/images/P/1552041778.01.THUI										
16	1558746218;"A Second Chicken Soup for the Woman's Soul (Chicken Soup for the Soul Series)";"Jack Canfield";"1998";"H										
17	1567407781;"The Witchfinder (Amos Walker Mystery Series)";"Loren D. Estleman";"1998";"Brilliance Audio - Trade";"htt										
18	1575663937;"More Cunning Than Man: A Social History of Rats and Man";"Robert Hendrickson";"1999";"Kensington Pub										
19	1881320189;"Goodbye to the Buttermilk Sky";"Julia Oliver";"1994";"River City Pub";"http://images.amazon.com/images/										
20	0440234743;"The Testament";"John Grisham";"1999";"Dell";"http://images.amazon.com/images/P/0440234743.01.THU										
21	0452264464;"Beloved (Plume Contemporary Fiction)";"Toni Morrison";"1994";"Plume";"http://images.amazon.com/ima										

Sample small output

```
give_similar(45911,X,Y)
```

	Book
0	"Harry Potter and the Order of the Phoenix (Book 5)"
1	"Chicken Soup for the Woman's Soul (Chicken Soup for the Soul Series (Paper))"
2	"Face the Fire (Three Sisters Island Trilogy)"
3	"Girl in Hyacinth Blue"
4	"Daughter of Fortune : A Novel (Oprah's Book Club (Hardcover))"
5	"The Tale of the Body Thief (Vampire Chronicles (Paperback))"
6	"The Celestine Prophecy (Celestine Prophecy)"
7	"Purity in Death"
8	"Life's Little Instruction Book (Life's Little Instruction Books (Paperback))"
9	"Strange Fits of Passion: A Novel"

-

```
recommend(1156,ratings_matrix,X,Y,10)
```

	Book	Score
0	"A Painted House"	1.000000
1	"The No. 1 Ladies' Detective Agency (Today Show Book Club #8)"	0.838920
2	"Snow Falling on Cedars"	0.829849
3	"The Bridges of Madison County"	0.795668
4	"The Hundred Secret Senses"	0.740169
5	"All I Really Need to Know"	0.703483
6	"Lonesome Dove"	0.700446
7	"Harry Potter and the Sorcerer's Stone (Book 1)"	0.690732
8	"Tears of the Giraffe (No.1 Ladies Detective Agency)"	0.676075
9	"A Child Called \"It\": One Child's Courage to Survive"	0.672392

- Working code

```
#Importing required python libraries

import pandas as pd
import numpy as np
import scipy.sparse as sparse
from scipy.sparse.linalg import spsolve
from sklearn.preprocessing import MinMaxScaler

books_df = pd.read_csv('/home/sovaniakshay/Documents/Books.csv',engine='python',sep=',')
books_df = books_df.rename(columns = {'ISBN;"Book-Title";"Book-Author";"Year-Of-Publication";"
books_df['book_title'] = books_df['ISBN'].str.split(';').str[1]
books_df['ISBN'] = books_df['ISBN'].str.split(';').str[0]
new_books_df = books_df[['book_title','ISBN']].drop_duplicates()
ratings_df = pd.read_csv("/home/sovaniakshay/Documents/CS512/Book-Ratings.csv", error_bad_line

merged_books_df = pd.merge(ratings_df,new_books_df,on='ISBN')
merged_books_df['ISBN_id'] = merged_books_df['ISBN'].astype("category").cat.codes
merged_books_df['USER_id'] = merged_books_df['User-ID'].astype("category").cat.codes

books_lookup = merged_books_df[['ISBN_id','book_title']].drop_duplicates()
books_lookup['ISBN_id'] = books_lookup.ISBN_id.astype(str)

merged_books_df.columns = ['User_ID','ISBN','Book_Rating','Book_Title','ISBN_ID','USER_id']

#full_data_books is a dataframe having complete information about book name, ID, userID.
full_data_books = merged_books_df.copy()


merged_books_df = merged_books_df.drop(['User_ID','ISBN','Book_Title'],axis=1)
input_data_for_matrix = merged_books_df.loc[merged_books_df.Book_Rating != 0]

users = list(np.sort(input_data_for_matrix['USER_id'].unique()))
ISBN = list(np.sort(input_data_for_matrix['ISBN_ID'].unique()))
rating = list(input_data_for_matrix['Book_Rating'])
rows = input_data_for_matrix['USER_id'].astype(int)
cols = input_data_for_matrix['ISBN_ID'].astype(int)

ratings_matrix=sparse.csr_matrix((rating,(rows,cols)),shape = (len(rows),len(cols)))
```

```
# ALS function generates 2 vectors X and Y

def ALS(ratings_matrix,alpha,iterations,lambda_val,features):
    confidence_matrix = ratings_matrix * alpha

    user_size,book_size = ratings_matrix.shape
    print(user_size)
    print(book_size)
    X = sparse.csr_matrix(np.random.normal(size = (user_size,features)))
    Y = sparse.csr_matrix(np.random.normal(size = (book_size,features)))

    X_I = sparse.eye(user_size)
    Y_I = sparse.eye(book_size)

    I = sparse.eye(features)
    lI = lambda_val * I
    u_iteration = 0
    i_iteration = 0
    for i in range(iterations):
        #Update the two vectors X and Y
        print ('itearation %d of total %d iterations' % (i+1,iterations))

        yTy = Y.T.dot(Y)
        xTx = X.T.dot(X)

        for u in range(user_size):
            u_iteration += 1
            print('In U for the %d time' % (u_iteration))
            u_row = confidence_matrix[u,:].toarray()

            p_u = u_row.copy()
            p_u[p_u != 0] = 1.0

            CuI = sparse.diags(u_row, [0])
            Cu = CuI + Y_I

            yT_CuI_y = Y.T.dot(CuI).dot(Y)
            yT_Cu_pu = Y.T.dot(Cu).dot(p_u.T)
            X[u] = spsolve(yTy + yT_CuI_y + lI, yT_Cu_pu)
            #This is one row updation of of the user vectors


        for i in range(book_size):
            i_iteration += 1
            print('In I for the %d time' % (i_iteration))
            i_row = confidence_matrix[:,i].T.toarray()

            p_i = i_row.copy()
            p_i[p_i != 0] = 1.0

            CiI = sparse.diags(i_row, [0])
            Ci = CiI + X_I

            xT_CiI_x = X.T.dot(CiI).dot(X)
            xT_Ci_pi = X.T.dot(Ci).dot(p_i.T)
            Y[i] = spsolve(xTx + xT_CiI_x + lI, xT_Ci_pi)
            #This is one column updation of of the book vectors
    print('Vectores X and Y have been created')
    return X, Y
    #in the end, we return the two vectors X and Y


#Function to find similar books

# Let's find similar books.
def give_similar(book_id,X,Y):

    # Get the book row for the book of which similar books are to be displayed
    book_vec = Y[book_id].T

    # Calculate the similarity score between this book and other books
    # and select the top 10 most similar.
    scores = Y.dot(book_vec).toarray().reshape(1,-1)[0]
    top_10 = np.argsort(scores)[::-1][:10]

    books = []
    books_scores = []

    # Get and print the actual book names and scores
    for idx in top_10:
        books_scores.append(scores[idx])

    similar = pd.DataFrame({'Book': books})
    pd.set_option('display.max_colwidth', -1)
    print (similar)



def recommend(USER_id, ratings_matrix, X, Y, no_of_books):
    """Recommend items for a given user given a trained model

        user_id (int): The id of the user we want to create recommendations for.
        ratings_matrix (csr_matrix): Our original training data.
        X (csr_matrix): The trained user x features vectors
        Y (csr_matrix): The trained item x features vectors
        no_of_books (int): How many recommendations we want to return.

    Returns:
        recommendations (pandas.DataFrame): DataFrame with recommended books
    """

    # Get all interactions by the user
    user_interactions = ratings_matrix[USER_id,:].toarray()

    user_interactions = user_interactions.reshape(-1) + 1
    user_interactions[user_interactions > 1] = 0

    #Now, calculate the recommendation
    #dot-product of the user vectors with the item vectors.
    rec_vector = X[USER_id,:].dot(Y.T).toarray()

    # Let's scale our scores between 0 and 1 to make it all easier to interpret.
    min_max = MinMaxScaler()
    rec_vector_scaled = min_max.fit_transform(rec_vector.reshape(-1,1))[:,0]
    recommend_vector = user_interactions*rec_vector_scaled

    # Get all the books indices in order of recommendations (descending) and
    # select only the top "num_items" items.
    book_index = np.argsort(recommend_vector)[::-1][:no_of_books]

    books = []
    scores = []

    # Loop through our recommended artist indicies and look up the actual book name

    for book in book_index:
        scores.append(recommend_vector[book])

    # Create a new dataframe with recommended artist names and scores
    recommendations = pd.DataFrame({'Book': books, 'Score': scores})

    print(recommendations)
```

Demo and sample findings

- Data size: In terms of Disk Resident:
  books.csv(87 MB) + books-rating.csv(19 MB) = 106 MB
- This project successfully recommends books for a given user. It suggests similar books to input books as well. These featurs of the project allows the user to quickly find the book he/she might like the most depending on the aleady given ratings by him/her. The functionality used has been implemented in such a way that it wont recommend already rated book to the user.
  One of the unique features of this project is that it not only recommends top 10 books but also tells the user the scores depending upon which the books have been recommended. So, the user can find pattern in the scores and can think about probable books depending upon the pattern he/she has found.

D. Stage4 - User Interface.

Describe a User Interface (UI) to your application along with the related information that will be shown on each interface view (How users will query or navigate the data and view the query or navigation results). The emphasis should be placed on the process a user needs to follow in order to meet a particular information need in a user-friendly manner. The deliverables for this stage include the following items :

- The modes of user interaction with the data (text queries, mouse hovering, and/or mouse clicks ?).
- The error messages that will pop-up when users access and/or updates are denied
- The information messages or results that wil pop-up in response to user interface events.
- The error messages in response to data range constraints violations.
- The interface mechanisms that activate different views in order to facilitate data accesses, according to users' needs.
- Each view created must be justified. Any triggers built upon those views should be explained and justified as well. At least one project view should be created with a justification for its use.

Please insert your deliverables for Stage4 as follows:

- The user interface for this project consists of one main user window giving the UI user the functionality to choose between two choices as follows:

  1. Find similar item to the book he/she likes.
  2. Find system recommended book for a particular user.

  User will be able to view 2 buttons one for similar book and another for recommendation respectively. Once the button has been clicked, respective functions

will get called. The first one will call to find_similar() function already implemented. The second one will call to recommend() function.

- The user has Two different navigation paths as follows: 1. Path for similarity() function: User will be directed to the new window which will give him/her the results for similarity. (i.e. similar books). 2. Path for recommendation() function: User will be directed to the new window which will give him/her the results for recommendation. (i.e. 10 recommended books to the user ID he/she selects).

  The error messages popping-up when users access and/or updates are denied (along with explanations and examples):

  - The error message: "Please enter correct book-title".
  - The error message explanation (upon which violation it takes place):
    The message will appear if the book-title is not present in the dataset. Please insert the error message explanation in here.
  - The error message example according to user(s) scenario(s):
    If the book title is not available in the dataset, then user will get the above error message and will be asked to enter a new book title. As book is not present in data, system wont be able to find the book and wont be able to recommend books similar to that book.

- The information messages or results that pop-up in response to user interface events.

  - The information message explanation and the corresponding event trigger
  - The error message example in response to data range constraints and the coresponding user's scenario A new window will open for each function. i.e. one for recommendation and another for similarity().

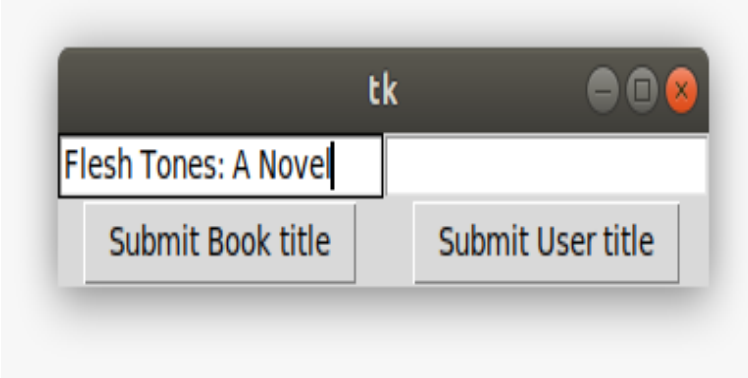
- The interface mechanisms that activate different views.

  - The interface mechanism: There are 2 buttons as interface mechanisms between user and system. Both have their respective functionalities (These interface mechanisms are similarity and recommendation buttons trigerring relevent functions).

- This below function gets called internally and outputs the similar books.

```
give_similar(45911,X,Y)
```

	Book
0	"Harry Potter and the Order of the Phoenix (Book 5)"
1	"Chicken Soup for the Woman's Soul (Chicken Soup for the Soul Series (Paper))"
2	"Face the Fire (Three Sisters Island Trilogy)"
3	"Girl in Hyacinth Blue"
4	"Daughter of Fortune : A Novel (Oprah's Book Club (Hardcover))"
5	"The Tale of the Body Thief (Vampire Chronicles (Paperback))"
6	"The Celestine Prophecy (Celestine Prophecy)"
7	"Purity in Death"
8	"Life's Little Instruction Book (Life's Little Instruction Books (Paperback))"
9	"Strange Fits of Passion: A Novel"



- This below function gets called internally and outputs the recommended books.

```
recommend(1156,ratings_matrix,X,Y,10)
```

	Book	Score
0	"A Painted House"	1.000000
1	"The No. 1 Ladies' Detective Agency (Today Show Book Club #8)"	0.838920
2	"Snow Falling on Cedars"	0.829849
3	"The Bridges of Madison County"	0.795668
4	"The Hundred Secret Senses"	0.740169
5	"All I Really Need to Know"	0.703483
6	"Lonesome Dove"	0.700446
7	"Harry Potter and the Sorcerer's Stone (Book 1)"	0.690732
8	"Tears of the Giraffe (No.1 Ladies Detective Agency)"	0.676075
9	"A Child Called \"It\": One Child's Courage to Survive"	0.672392