Books Recommendation System

CMPE-256 Individual Project

Purva Deekshit

SID: 013749723

I. Abstract

This report contains the analysis and implementation of books recommendation system, using multiple approaches of recommendation, such as content-based filtering, user-based collaborative filtering, item-based collaborative filtering, deep learning recommendation, trending books recommendation and hybrid recommendation. I have also implemented all the approaches and compared their results at the end to get top N recommendations from each of the approach. For evaluation, I have shown the top 10 books recommended from each of the recommender systems, for an entered user ID.

II. Motivation

The goal of this project is to recommend useful books, based on students/readers interests and requirements. It will be helpful in saving the time spent on searching for relevant books from a huge books database. I have tried to incorporate all the aspects of recommendation, to apply theoretical concepts in real-world application. So, instead of using only user preferences or knowledge from community, I have combined them to implement hybrid recommender system. Another approach is to recommend current trending books related to categories which user have previously referenced. I have also applied machine learning so the model can learn from historical data and make predictions based on the hidden features.

III. Data Collection

Since we refer multiple sources/websites before purchasing a new book, I have also collected books and user metadata from two websites. I have combined data from Google Books API and Goodreads API to get books information, user information and ratings.

1. Books Data:

a. From Google API:

I have collected the data from Google Books API using Google API Client Library for Python. Google Books API's documentation: https://developers.google.com/books/docs/v1/getting_started

To use the API, we need to register on Google books website using Gmail account. It generates a developer key that needs to be passed with every request. Below is the code snippet for fetching books data:

```
url = 'https://www.googleapis.com/books/v1/volumes?q=' + str(fetch_item) + ' &key=' + str(DEVELOPER_KEY)
    url_data = requests.get(url,params=params).json()
```

Google API metadata includes various categories of books, such as: Business, Economics, Children, Computer Science, Mathematics, Literature, Education, etc.

I have selected 14 categories, as shown:

```
keywords = ['Photography+books',
            'Business+entrepreneurship+books',
            'programming+language+books',
            'technology+books'
            'Comics+Novels',
            'Children+story+books',
            'Parenting+books',
            'Health+and+Fitness+books',
            'Science+Fiction+books',
            'Mystery+novels',
            'Spirituality+books',
            'Mathematics+books',
            'Biographies',
            'Travel+adeventure+books']
```

For each category, I have fetched 20 pages, having 40 records on each page. So, total 800 records for each category are fetched. Total number of books fetched is 7456.

Google API also includes information about books, such as: Title, Author, Category, Description, Language, Industry Identifier (ISBN), Website Link, Selling Info, etc.

Below is the screenshot of all the fields extracted for each record:

```
keys = ['title','subtitle','authors','publisher','publishedDate','description',
        'industryIdentifiers','readingModes','pageCount','printType','categories',
        'averageRating','ratingsCount','maturityRating','language','infoLink']
```

b. From Goodreads API:

I have fetched user reviews, data per ISBN from goodreads.com using Goodreads API: https://www.goodreads.com/api. For each record, I have fetched the ISBN. ISBN can be 10 or 13 digits. I have considered 13 digits only.

Libraries used:
from bs4 import BeautifulSoup
from goodreads import client

For each ISBN from Goodreads API, I have extracted the below information:
Goodreads_rating, goodreads_ratings_count, goodreads_rev_link.

Below is the screenshot of code snippet for extracting this information:

```
#Fetch review metadata
if info['isbn']:
    gd_info = fetch_goodreads_data(isbn)
    if 'goodreads_rating' in gd_info.keys():
        info['goodreads_rating'] = gd_info['goodreads_rating']
    if 'goodreads_ratings_count' in gd_info.keys():
        info['goodreads_ratings_count'] = gd_info['goodreads_ratings_count']
    if 'goodreads_rev_link' in gd_info.keys():
        info['goodreads_rev_link'] = gd_info['goodreads_rev_link']
```

After collecting all the data, below is the screenshot of books information dataframe:

```
1 books_df.shape
```

```
(7456, 20)
```

```
1 books_df.head(20)
```

averageRating	categories	description	goodreads_rating	goodreads_ratings_count	goodreads_rev_link	industryIdentifiers
4.0	[Computers]	Furnishes an overview of digital photography, ...	4.13	2477	https://www.goodreads.com/book/show/17938945-t...	[{'type': 'ISBN_13', 'identifier': '9780321948...
NaN	[Photography]	Scott Kelby, author of the top-selling digital...	4.34	164	https://www.goodreads.com/book/show/22826307-t...	[{'type': 'ISBN_13', 'identifier': '9780133856...
4.0	[Computers]	Offers advice for shooting different types of ...	4.34	164	https://www.goodreads.com/book/show/20839346-t...	[{'type': 'ISBN_13', 'identifier': '9780133856...
5.0	[Photography]	Winner of the National Book Critics' Circle A...	3.81	33529	https://www.goodreads.com/book/show/12023566-o...	[{'type': 'ISBN_10', 'identifier': '1429957115...

2. User Data:

I have sent a query to Goodreads with the ISBN received from Google API to get a reviews link for book with that ISBN. This link provides the information of users who have rated the book. After fetching the data from this link, that is, goodreads_rev_link, we get the following information for each user review:

ISBN, User ID, Username, User Rating and User Text review for the book.

I have converted this user information and user reviews information into a pandas dataframe. Below is the screenshot of reviews dataframe:

```
1 reviews_df.shape
```

```
(103930, 5)
```

```
1 reviews_df.head(20)
```

	isbn	user_id	user_name	user_ratings	user_review_text
0	9780321948540	779608	Erin	4.0	There are a lot of great tips in this volume, ...
1	9780321948540	753824	Rolf Häsänen	4.0	Not as many useful tips as volume1 unless you ...
2	9780321948540	35798331	Valery	4.0	Not as helpful as the first, but does go into ...
3	9780321948540	2100772	Bruce	1.0	OMG. Surely the absolute worst, amature photo ...
4	9780321948540	1794100	Icepick	5.0	The format may not be for everybody, but I rea...
5	9780321948540	3425034	Deb	4.0	Lots of great tips! This writer is engaging an...

I have saved this data into two pickle files, books.pkl for books and reviews.pkl for reviews, so we don't have to run the code each time for generating the data.

IV. Data Analysis, Cleaning and Preprocessing

For categories which have missing values, Google API does not include the specific field in API response. As a part of data cleaning and preprocessing, I have identified such missing fields/values and filtered them. I have also removed non-informative features.

1. Books:

a. Data before cleaning and pre-processing:

```
books = pd.read_pickle('books.pkl')
```

```
books.shape
```

```
(7456, 20)
```

b. Checked and dropped null values:

```
books.dtypes
```

```
authors                      object
averageRating               float64
categories                   object
description                  object
goodreads_rating             object
```

```
#Display how many nan values per column
books.isnull().sum()
```

```
authors                       788
averageRating                5306
categories                    970
description                  1821
goodreads_rating             2784
```

c. Removed duplicate rows:

```
#Get count of all books
count_books = books.isbn.count()
print("Total book count : ", count_books)
```

```
Total book count :  5741
```

```
#Get unique books
unique_books = books.isbn.nunique()
print("No of unique books : ", unique_books)
```

```
No of unique books :  5204
```

d. Dropped non-informative features:

```
#Keep only feature columns
columns_to_drop = ['averageRating','goodreads_rating','goodreads_ratings_count',
                   'industryIdentifiers','ratingsCount','google_ratings',
                   'gr_rating_float','gr_rating_count_float','goodreads_ratings']
books.drop(columns=columns_to_drop,inplace=True)
```

e. Standardized data by converting float values in page counts to integers and published dates in uniform format of year:

```
#Consider number of pages as integer
books['pageCount'] = books['pageCount'].astype(int)
```

```
#Consider published date as year
books['publishedDate'] =  pd.to_datetime(books['publishedDate'])
books['publishedDate'] = pd.DatetimeIndex(books['publishedDate']).year
```

f. Replaced unavailable values with 0:

```
#Replace not available ratings with 0
books['averageRating'].fillna(0, inplace=True)
books['ratingsCount'].fillna(0, inplace=True)
books['goodreads_rating'].fillna(0, inplace=True)
books['goodreads_ratings_count'].fillna(0, inplace=True)
```

g. Converted authors and categories from list to comma separated values:

```
#Convert Authors from list to comma separated list
books['authors'] = books['authors'].apply(', '.join)
```

```
#Convert Categories from list to comma separated values
books['categories'] = books['categories'].apply(', '.join)
```

h. Calculated weighted average for ratings from Google and Goodreads websites to get a single ratings data.

```
#Calculate weighted average from google ratings and good reads for each book
books['gr_rating_float'] = books['goodreads_rating'].astype(float)
books['gr_rating_count_float'] = books['goodreads_ratings_count'].astype(float)
```

```
books.shape
```

```
(7456, 19)
```

i. Separated reading modes into two columns: readingModes_text and readingModes_img:

```
#Split readingmodes json to multiple columns
reading_modes = pd.DataFrame(books['readingModes'].values.tolist())
reading_modes.columns = 'readingModes_'+ reading_modes.columns
reading_modes.columns
```

```
Index(['readingModes_image', 'readingModes_text'], dtype='object')
```

```
books['readingModes_image'] = reading_modes[['readingModes_image']]
```

```
books['readingModes_text'] = reading_modes[['readingModes_text']]
```

j. Data after cleaning and pre-processing:

```
books.shape
```

```
(3396, 12)
```

k. Saved this pre-processed pandas dataframe in processed_books.pkl file.

2. Reviews:

a. Reviews dataframe before cleaning and preprocessing:

```
reviews.shape
```

```
(103930, 5)
```

b. Checked and dropped null values:

```
#Check null
reviews.isnull().sum()
```

```
isbn                   0
user_id            55729
user_name          55729
user_ratings       55729
user_review_text   55729
dtype: int64
```

```
#drop na
reviews.dropna(inplace=True)
```

```
reviews.shape
```

```
(48201, 5)
```

c. Grouped by top 20 users who have rated the maximum time:

```
#Get top 20 users
top_users = reviews.groupby('user_id').isbn.count().sort_values(ascending=False)
print("Users who have rated most : ")
print(top_users[:10])
```

```
Users who have rated most :
user_id
4213258     74
4622890     66
597461      62
269235      61
5253785     53
16731747    53
614778      51
416390      51
45618       48
279256      44
Name: isbn, dtype: int64
```

d. Saved this preprocessed pandas dataframe in processed_reviews.pkl file.

3. Books-To-Reviews:

a. Books_reviews dataframe before cleaning and pre-processing:

```
full_metadata.shape
```

```
(56660, 17)
```

b. Merged books and reviews data frames according to the ISBN.

```
#Merge books and reviews
full_metadata = reviews.join(books.set_index('isbn'), on='isbn')
```

c. Checked and dropped null values:

```
full_metadata.drop(columns=['readingModes_image','readingModes_text'],inplace=True)
```

d. Books_reviews dataframe after cleaning and pre-processing:

```
full_metadata.shape
```

```
(51487, 15)
```

e. Sample of books_reviews dataframe:

```
full_metadata.head(10)
```

ew_text	authors	categories	description	language	pageCount	printType	publishedYear	publisher	title	weighted_avg
e a lot of s in this lume, ...	Scott Kelby	Computers	Furnishes an overview of digital photography, ...	en	236.0	BOOK	2013.0	Pearson Education	The Digital Photography Book	4.128036
as many l tips as 1 unless you ...	Scott Kelby	Computers	Furnishes an overview of digital photography, ...	en	236.0	BOOK	2013.0	Pearson Education	The Digital Photography Book	4.128036

f. Saved this merged dataframe as full_metadata.pkl file.

V. Approaches

I have implemented the below 5 approaches for recommending top N books:

1. Content-based filtering
2. Collaborative filtering
   a. User-based collaborative filtering
   b. Item-based collaborative filtering
3. Deep learning model
4. Trending and best-selling books
5. Hybrid model

1. Content-Based Filtering:

In this approach, I have built a user profile and an item profile based on the ratings given by the user to the item. So, we get a matrix containing user preferences and attributes of the book. So, in future, if a book has an attribute that the user has preferred or rated highly, this system will recommend that book to the user.

a. Build item profile:

I have used one hot encoding to generate the book-to-attribute matrix.

```
#Build Item profile
isbn_category = fulldata[['isbn','categories']]
isbn_category.shape
```

```
(39784, 2)
```

The sample attributes from categories are:

```
1 isbn_category.categories.unique()
```

```
array(['Computers', 'Photography', 'Law', 'Art', 'Philosophy',
       'Education', 'Nature', 'History', 'Social Science', 'COMPUTERS',
       'Literary Criticism', 'Biography & Autobiography', 'Pets',
       'Psychology', 'Business & Economics', 'Young Adult Fiction',
       'Self-Help', 'Design', 'Travel', 'Architecture', 'Gardening',
       'Small business', 'Science', 'Entrepreneurship', 'House & Home',
       'Medical', 'Marketing', 'Language Arts & Disciplines',
       'Technology & Engineering', 'Mathematics',
       'Foreign Language Study', 'C# (Computer program language)',
       'Computer programming', 'Python (Computer program language)',
       'Juvenile Nonfiction', 'Comics & Graphic Novels',
       'Comic books, strips, etc', 'Fiction', 'Literary Collections',
       'Family secrets', 'Juvenile Fiction', 'Cloning',
       'Cartoon characters', 'Family & Relationships',
       'Adventure stories', "Children's stories",
       'COMICS & GRAPHIC NOVELS', 'Young Adult Nonfiction', 'Humor',
       'Study Aids', 'Body, Mind & Spirit', 'Plague',
       'East Indian Americans', 'Imaginary places', 'Performing Arts',
       'True Crime', 'Religion', 'Elves', 'Bands (Music)', 'Drama',
       "Children's stories, American.", 'Braille books',
       'Crafts & Hobbies', 'JUVENILE FICTION', 'Bible stories, English',
       "Children's fiction", 'Brothers and sisters',
       'JUVENILE NONFICTION', 'Abused children',
       'BIOGRAPHY & AUTOBIOGRAPHY', 'Sports & Recreation',
       'Books and reading', 'Self-esteem', 'Cooking',
       'FAMILY & RELATIONSHIPS', 'Child rearing', 'Health & Fitness',
       'Child development', 'Only child', 'Communication in marriage',
       'Infants', 'Exercise therapy', 'Mothers', 'Athletes', 'Metabolism',
```

The dataframe of books (ISBN) to attributes (Category) after one hot encoding is:

```
1 isbn_category.head(100)
```

isbn	Abused children	Adventure stories	Adventure travel	Aerospace engineers	Architecture	Art	Art and music	Assertiveness (Psychology)	Athletes	Australia	...	Study Aids	Technology & Engineering	Travel	Travel writers
9780007198238	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
9780007236350	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
9780007340484	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
9780007509263	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
9780008291730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

b. Build user profile:

Screenshot of ISBN to user ID rating matrix:

```
#Ratings for isbn
#isbn_ratings = pd.pivot_table(filteredisbn, values='user_ratings', index=['isbn'], columns = ['user_id'])
isbn_ratings = pd.pivot_table(fulldata, values='user_ratings', index=['isbn'], columns = ['user_id'])
isbn_ratings.sort_index(axis=1, inplace=True)
```

```
print(isbn_ratings)
print(isbn_ratings.head(5))
```

```
user_id          1  1000037  10001960  1000231  100026730  10003774  10005457  \
isbn
9780007198238 NaN      NaN       NaN      NaN        NaN       NaN       NaN
9780007236350 NaN      NaN       NaN      NaN        NaN       NaN       NaN
9780007340484 NaN      NaN       NaN      NaN        NaN       NaN       NaN
9780007509263 NaN      NaN       NaN      NaN        NaN       NaN       NaN
9780008291730 NaN      NaN       NaN      NaN        NaN       NaN       NaN
```

Screenshot of ISBN to categories matrix:

```
ratings_matrix.head()
```

isbn	1	1000037	10001960	1000231	100026730	10003774	10005457	10006359	10007949	1000903	...	999171	999220	9992398	99935	999436
9780007198238	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
9780007236350	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
9780007340484	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
9780007509263	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
9780008291730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

Screenshot of user ID to categories matrix:

```
#user profile
user_category.head()
```

	Abused children	Adventure stories	Adventure travel	Aerospace engineers	Architecture	Art	Art and music	Assertiveness (Psychology)	Athletes	Australia	...	Study Aids	Technology & Engineering	Travel	Travel writers
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1000037	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
10001960	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
1000231	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
100026730	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

Then I have used TF-IDF to generate the resultant matrix for content-based filtering and saved the resultant matrix in content_based_rating_matrix.pkl file.

Finally, I got top 10 recommendations using content-based filtering:

- Get user id
- For all books, if user has not rated the book, calculate rating as per resultant matrix, using TFIDF.
- Sort this resultant matrix in descending order of ratings.
- Return top10 books.

Results of content-based filtering:

```
recommendations = contentbasedtop10('269235')
isbn_recommendations = top10isbn(recommendations)
print("Content based filtering recommendations : ")
for isbnitem in isbn_recommendations :
    print("ISBN :", str(isbnitem), "Title :", str(isbn_title_dict[isbnitem]))
```

```
Content based filtering recommendations :
ISBN : 9781893361928 Title : Winter
ISBN : 9781627796347 Title : Being a Beast
ISBN : 9781101663189 Title : The Snow Leopard
ISBN : 9780684852331 Title : Out of the Noosphere
ISBN : 9780544341647 Title : Cry of the Kalahari
ISBN : 9780486138664 Title : Travels of William Bartram
ISBN : 9780345806291 Title : The Invention of Nature
ISBN : 9780316396684 Title : Kings of the Yukon
ISBN : 9780240818832 Title : Digital Wildlife Photography
ISBN : 9781984815767 Title : Warcross
```

Advantages:

This approach eliminates the cold start problem, where enough ratings are not available for an item, either because it is new or fewer users have rated it. But since it matches the preference of target users, there is a higher chance that user will like it.

2. Collaborative Filtering

Below are the steps that I followed in this approach:

Got user ID, ISBN and ratings:

```
isbn_uers_rating = reviews[['user_id', 'isbn', 'user_ratings']]
isbn_uers_rating.head()
```

	user_id	isbn	user_ratings
0	779608	9780321948540	4.0
1	753824	9780321948540	4.0
2	35798331	9780321948540	4.0
3	2100772	9780321948540	1.0
4	1794100	9780321948540	5.0

Generated user ID to ISBN matrix:

```
print(ratings_matrix.shape)
ratings_matrix.head()
```

```
(30675, 2670)
```

isbn	9780006548393	9780007198238	9780007222582	9780007224487	9780007236350
user_id					
1	NaN	NaN	NaN	NaN	NaN
1000037	NaN	NaN	NaN	NaN	NaN
10001960	NaN	NaN	NaN	NaN	NaN
1000231	NaN	NaN	NaN	NaN	NaN
100026730	NaN	NaN	NaN	NaN	NaN

Sample of ratings matrix after filling missing values with 0:

```
ratings_matrix.head(5)
```

isbn	9780006548393	9780007198238	9780007222582	9780007224487
user_id				
1	0	0	0	0
1000037	0	0	0	0
10001960	0	0	0	0
1000231	0	0	0	0
100026730	0	0	0	0

5 rows × 2670 columns

Saved this matrix in collaborative_ratings_matrix.pkl file.

Found sparsity of the matrix.

```
#Calculate sparsity of ratings matrix
sparsity=1.0-len(ratings_matrix)/float(ratings_matrix.shape[0]*ratings_matrix.shape[1])
print ("sparsity percentage : " +  str(sparsity*100))
```

```
sparsity percentage : 99.9625468164794
```

Since the data is very sparse, I have used cosine similarity to find top N nearest neighbors.

a. User-Based Collaborative Filtering:

I have defined a function ubcftop10 (user) for user-based filtering using the formula:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

```python
#Get top user baed collborative filtering recommendations for user
def ubcftop10(user):
    ratings_rankings = []
    for i in range(ratings_matrix.shape[1]):
        ratinglist = []
        if ratings_matrix[str(ratings_matrix.columns[i])][user] ==0:
            ratinglist.append(int(0))
            ratinglist.append(str(ratings_matrix.columns[i]))
        else:
            ratinglist.append(int((predict_ratings_user(user, str(rat
            ratinglist.append(str(ratings_matrix.columns[i]))
        ratings_rankings.append(ratinglist)

    ratings_rankings.sort(key=lambda x: x[0])
    ratings_rankings.reverse()
    top10_ratings = ratings_rankings[:10]
    return top10_ratings
```

b. Item-Based Collaborative Filtering:

I have defined a function ibcftop10(user) for item-based filtering using the formula:

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

```
#Get top item baed collborative filtering recommendations for user
def ibcftop10(user):
    ratings_rankings = []
    for i in range(ratings_matrix.shape[1]):
        ratinglist = []
        if ratings_matrix[str(ratings_matrix.columns[i])][user] ==0:
            ratinglist.append(int(0))
            ratinglist.append(str(ratings_matrix.columns[i]))
        else:
            ratinglist.append(int((predict_ratings_item(user, str(ra
            ratinglist.append(str(ratings_matrix.columns[i]))
        ratings_rankings.append(ratinglist)

    ratings_rankings.sort(key=lambda x: x[0])
    ratings_rankings.reverse()
    top10_ratings = ratings_rankings[:10]
    return top10_ratings
```

Finally, got top 10 recommendations for both the approaches:

- Get user id
- For all books, if user not rated book, calculate rating as per k nearest neighbors
- Sort this resultant matrix in descending order of ratings.
- Return top10 books

Results of user-based filtering:

```
recommendations = ubcftop10('269235')
isbn_recommendations = top10isbn(recommendations)
print("User based collaborative filtering recommendations : ")
for isbnitem in isbn_recommendations :
    print("ISBN :", str(isbnitem), "Title :", str(isbn_title_dict[isbnitem]))
```

```
User based collaborative filtering recommendations :
ISBN : 9781613126219 Title : El Deafo
ISBN : 9781770461987 Title : SuperMutant Magic Academy
ISBN : 9781770461536 Title : Over Easy
ISBN : 9781596436626 Title : Red Handed
ISBN : 9781466858527 Title : This One Summer
ISBN : 9781442465978 Title : Through the Woods
ISBN : 9780679462712 Title : Into Thin Air
ISBN : 9780618871711 Title : Fun Home
ISBN : 9780307772947 Title : The Postman Always Rings Twice
ISBN : 9780307767516 Title : The Maltese Falcon
```

Results of item-based filtering:

```
recommendations = ibcftop10('269235')
isbn_recommendations = top10isbn(recommendations)
print("Item based collaborative filtering recommendations : ")
for isbnitem in isbn_recommendations :
    print("ISBN :", str(isbnitem), "Title :", str(isbn_title_dict[isbnitem]))
```

```
Item based collaborative filtering recommendations :
ISBN : 9781939799081 Title : Infomaniacs
ISBN : 9781910593554 Title : I Feel Machine
ISBN : 9781770461536 Title : Over Easy
ISBN : 9781596438736 Title : Hidden: A Child's Story of the Holocaust
ISBN : 9780224063975 Title : Jimmy Corrigan
ISBN : 9781770463219 Title : A Bubble
ISBN : 9781770463165 Title : Sabrina
ISBN : 9781770461987 Title : SuperMutant Magic Academy
ISBN : 9781683961895 Title : Alienation
ISBN : 9781613126219 Title : El Deafo
```

Advantages:

As we can see, both the approaches gave different results. So, more recommendations are provided for the same user-book datapoint.

3. Deep Learning Model:

Below are the steps I followed in this approach:

Read reviews dataframe:

```
reviews_dataset.head(5)
```

	isbn	user_id	user_name	user_ratings	user_review_text
0	9780321948540	779608	Erin	4.0	There are a lot of great tips in this volume, ...
1	9780321948540	753824	Rolf Häsänen	4.0	Not as many useful tips as volume1 unless you ...
2	9780321948540	35798331	Valery	4.0	Not as helpful as the first, but does go into ...
3	9780321948540	2100772	Bruce	1.0	OMG. Surely the absolute worst, amature photo ...
4	9780321948540	1794100	Icepick	5.0	The format may not be for everybody, but I rea...

After dropping null values, the dataframe looks like:

```
reviews_dataset.head(50)
```

	isbn	user_id	user_ratings
0	9780321948540	779608	4.0
1	9780321948540	753824	4.0
2	9780321948540	35798331	4.0
3	9780321948540	2100772	1.0
4	9780321948540	1794100	5.0

Applied one hot encoding for user ID in order to convert the users into categorical data:

```
1  isbn_dict
```

```
{'9780321948540': 1,
 '9780133856934': 2,
 '9780133856880': 3,
 '9781429957113': 4,
 '9780134385273': 5,
```

Applied one hot encoding for ISBN in order to convert the books into categorical data:

```
1  reviews_dataset.head(50)
```

	isbn	user_id	user_ratings
0	1	779608	4.0
1	1	753824	4.0
2	1	35798331	4.0
3	1	2100772	1.0
37	2	9348396	4.0
38	2	25369063	5.0
60	3	27610501	4.0
61	3	56801456	5.0
62	3	53222157	4.0

Sample of new dataframe:

```
1  reviews_dataset.head(10)
```

	isbn	user_id	user_ratings
0	1	1	4.0
1	1	2	4.0
2	1	3	4.0
3	1	4	1.0
4	1	5	5.0

Split the dataset into training and testing datasets:

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(reviews_dataset, test_size=0.2, random_state=42)
```

Created neural network model using Keras. Size of input for deep learning layer is:

```
n_users = len(reviews_dataset.user_id.unique())
n_users
```

```
30675
```

```
n_books = len(reviews_dataset.isbn.unique())
n_books
```

```
2670
```

Embedding is the first layer. Second is flattening. I have done this for users and books. The I concatenated both the layers and attached 2 fully connected layers. Finally, compiled the model using Adam Optimizer for mean squared error.

```
from keras.layers import Input, Embedding, Flatten, Dot, Dense, Concatenate
from keras.models import Model


# creating book embedding path
book_input = Input(shape=[1], name="Book-Input")
book_embedding = Embedding(n_books+1, 5, name="Book-Embedding")(book_input)
book_vec = Flatten(name="Flatten-Books")(book_embedding)

# creating user embedding path
user_input = Input(shape=[1], name="User-Input")
user_embedding = Embedding(n_users+1, 5, name="User-Embedding")(user_input)
user_vec = Flatten(name="Flatten-Users")(user_embedding)

# concatenate features
conc = Concatenate()([book_vec, user_vec])

# add fully-connected-layers
fc1 = Dense(128, activation='relu')(conc)
fc2 = Dense(32, activation='relu')(fc1)
out = Dense(1)(fc2)

# Create model and compile it
model2 = Model([user_input, book_input], out)
model2.compile('adam', 'mean_squared_error',metrics=['accuracy'])
```

Trained the model for 50 epochs. Below is the sample for 2 epochs:

```
import matplotlib.pyplot as plt
%matplotlib inline

history = model2.fit([train.user_id, train.isbn], train.user_ratings, epochs=2, verbose=1)
```

```
WARNING:tensorflow:From /Users/sameerdeekshit/anaconda3/lib/python3.7/site-packages/tensorflow
:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a f
Instructions for updating:
Use tf.cast instead.
Epoch 1/2
38560/38560 [==============================] - 9s 221us/step - loss: 2.1233 - acc: 0.3004
Epoch 2/2
38560/38560 [==============================] - 8s 210us/step - loss: 0.9672 - acc: 0.4685
```

Calculated accuracy as:

```
print(history.history.keys())
```

```
dict_keys(['loss', 'acc'])
```

```
model2.evaluate([test.user_id, test.isbn], test.user_ratings)
```

```
9641/9641 [==============================] - 0s 35us/step

[1.558443599241352, 0.3483041178548303]
```

Saved the model as deep learning model in deeplearningmodel file.

For testing, I loaded this model and entered input as user ID. For each book, model.predict() calculates ratings for items which user has not rated.

```
user = '269235'
bookratings = []
for i in usersbooks:
    allbooks = usersbooks[i]
    for item in allbooks :
        #print("mapped book : ", item)
        #print("mapped user : ", user)
        ud = user_dict[user]
        isd = isbn_dict[item]
        #print("actual book : ", isd)
        #print("actual user : ", ud)
        x = pd.Series([ud])
        #print(type(x))
        #print(x)
        y = pd.Series([isd])
        #print(type(y))
        #print(y)

        predictions = model2.predict([x, y])
        #print(predictions[0][0])
        bookratings.append([predictions[0][0],item])
```

For recommendation:

- Get user id
- For all books, if user not rated book, calculate rating as per k nearest neighbors
- Sort this resultant matrix in descending order of ratings.
- Return top10 books

Results of deep learning model:

```
deeplrecommended = top10deep('269235')
isbn_recommendations = top10isbn(deeplrecommended)
print("Deep learning based model recommendations books : ")
for isbnitem in isbn_recommendations :
    print("ISBN :", str(isbnitem), "Title :", str(isbn_title_dict[isbnitem]))
```

```
Deep learning based model recommendations books :
ISBN : 9781453237915 Title : West with the Night
ISBN : 9781599908755 Title : Starcross
ISBN : 9781596431317 Title : Drawing Words and Writing Pictures
ISBN : 9780143111597 Title : The Left Hand of Darkness
ISBN : 9782080201331 Title : Master Photographers
ISBN : 9780385372091 Title : Green Eggs and Ham: Read & Listen Edition
ISBN : 9780757305603 Title : The Sleepeasy Solution
ISBN : 9780821215517 Title : Examples
ISBN : 9780936861579 Title : Captives of Blue Mountain
ISBN : 9780805800586 Title : The Psychology of Learning Mathematics
```

Advantages:

This approach is useful when a greater number of items and user ratings are available. Deep learning model learns from the historic data to find hidden patterns and give more diverse results.

4. Trending and Best-Selling Recommendation

I followed below steps in this approach:

Fetched current trending books and current best-selling books from Google API.

```
print("Start time : ", datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
for item in range(len(keywords)):
    fetch_item = keywords[item]
    print("Fetch : ", fetch_item )
    #Get single book
    url = 'https://www.googleapis.com/books/v1/volumes?q=' + str(fetch_item) + ' &key=' + str(DEVELOPER_KEY)
    print("url : ",url)
    fetch_data(url)
print("End time : ", datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
```

Followed all the steps in “Data preprocessing” to generate a dataframe. Store trending books dataframe in trending_books_data.pkl file. From the trending and best-selling items, analyzed the description of the content to calculate the items that are most similar to those preferred by the user.

Used TF-IDF to convert the attributes into a vector:

```
tfidfv = TfidfVectorizer(analyzer='word',stop_words=set(stopwords.words('english')))
vocab = tfidfv.fit_transform(combineddata['description'])
similarity = linear_kernel(vocab, vocab)
isbns = pd.Series(combineddata['isbn']).to_dict()
isbns = dict((v,k) for k,v in isbns.items())
```

Used cosine similarity to find the top N similar items:

```
def trending_similar(isbn):
    bookid = isbns[isbn]
    bookslist = list(enumerate(similarity[bookid]))
    bookslist.sort(key=lambda x: x[1], reverse=True)
    top10TrendingBooks = bookslist[1:11]
    return top10TrendingBooks
```

Finally made recommendations of top 10 trending and similar or related items:

```
def trendingTop10(user):
    booklist = usersbooks[user]
    #print(booklist)
    recommendedbooks = []
    for book in booklist:
        #print(book)
        newbooks = trending_similar(book)
        #print("newbooks : " ,newbooks)
        recommendedbooks = recommendedbooks + newbooks
    recommendedbooks = list(set(recommendedbooks))
    recommendedbooks.sort(key=lambda x: x[1], reverse=True)
    return recommendedbooks[:10]
```

For recommendation:

- Get user id
- For each book rated by use TF-IDF to get content-based similarity from trending books.
- Sort this resultant matrix in descending order of ratings.
- Return top10 books

Results of trending now and best-selling recommendation:

```
trendingrecommended = trendingTop10('269235')
isbn_recommendations = top10isbn(trendingrecommended)
print("Recommendations based on trending books : ")
for isbnitem in isbn_recommendations :
    print("ISBN :", str(isbnitem), "Title :", str(combined_isbn_title_dict[isbnitem]))
    #print("ISBN :", str(isbnitem))
```

```
Recommendations based on trending books :
ISBN : 9788186775097 Title : The Five Love Languages
ISBN : 9788175110625 Title : Answer is Blowing in the Wind
ISBN : 9788175110625 Title : Answer is Blowing in the Wind
ISBN : 9783319326931 Title : Human Dignity of the Vulnerable in the Age of Rights
ISBN : 9783319326931 Title : Human Dignity of the Vulnerable in the Age of Rights
ISBN : 9783319012223 Title : Most-Cited Scholars in Criminology and Criminal Justice, 1986-2010
ISBN : 9781948584081 Title : The Catalain Book of Secrets
ISBN : 9781943562183 Title : Rescuing Wendy
ISBN : 9781943562183 Title : Rescuing Wendy
ISBN : 9781895431162 Title : New World Order & Third World
```

Advantages:

This approach helps to get latest the recommendations, apart from the items in the dataset. So, if new relevant items are available, that are not present in dataset, still user has a chance of getting recommendations for that item. This approach is also helpful in case the user is new and no previous data is available.

5. Hybrid Model:

This approach combines 2 recommendations from each of the above-mentioned approaches and recommends 10 books considering all the approaches. I have shown these results in the next part.

VI. Testing and Evaluation:

I have written a Python script that uses the pickle files for each of the approaches above. I loaded the pickle file for each method and got top 10 recommendations from each approach. Finally, 10 books are recommended. The running time for whole process is 2-5 minutes. Below is the screenshot of recommended books from all the approaches:

```
Using TensorFlow backend.
Welcome to Book Recommendation System :
Enter user id : 269235
Entered userid is :  269235

Calculating content based filtering books recommendations

Content based filtering recommendations :
ISBN : 9781524742393 Title : America's Reluctant Prince
ISBN : 9781501194313 Title : Howard Stern Comes Again
ISBN : 9780399590511 Title : Educated
ISBN : 9781501181009 Title : The Outsider
ISBN : 9781466861244 Title : The Calculating Stars
ISBN : 9781466847705 Title : How to Walk Away
ISBN : 9781250301710 Title : The Silent Patient
ISBN : 9781101967683 Title : The Whistler
ISBN : 9780812988918 Title : The Last Days of Night
ISBN : 9780804137263 Title : Armada

Calculating user based collaborative filtering books recommendations

User based collaborative filtering recommendations :
ISBN : 9780553448122 Title : Artemis
ISBN : 9781524731663 Title : Bad Blood
ISBN : 9781429915649 Title : A Wrinkle in Time
ISBN : 9781250301710 Title : The Silent Patient
ISBN : 9780804137263 Title : Armada
ISBN : 9780544609716 Title : The Whole30
ISBN : 9788186775097 Title : The Five Love Languages
ISBN : 9781400201662 Title : Girl, Wash Your Face
ISBN : 9781449335564 Title : You Don't Know JS: Scope & Closures
ISBN : 9780812988918 Title : The Last Days of Night
```

```
Calculating item based collaborative filtering books recommendations

Item based collaborative filtering recommendations :
ISBN : 9781466861244 Title : The Calculating Stars
ISBN : 9780553448122 Title : Artemis
ISBN : 9781466847705 Title : How to Walk Away
ISBN : 9781250301710 Title : The Silent Patient
ISBN : 9781449335564 Title : You Don't Know JS: Scope & Closures
ISBN : 9780544609716 Title : The Whole30
ISBN : 9788186775097 Title : The Five Love Languages
ISBN : 9780735219113 Title : Where the Crawdads Sing
ISBN : 9781524731663 Title : Bad Blood
ISBN : 9781101202661 Title : The Little Book of Atheist Spirituality

Calculating trending books recommendations

Recommendations based on trending books :
ISBN : 9789042911833 Title : Spirituality
ISBN : 9789042019485 Title : Cyberculture, Cyborgs and Science Fiction
ISBN : 9788186775097 Title : The Five Love Languages
ISBN : 9788175110625 Title : Answer is Blowing in the Wind
ISBN : 9788175110625 Title : Answer is Blowing in the Wind
ISBN : 9783737538268 Title : Animal Paradise
ISBN : 9783540533467 Title : Mathematics — The Music of Reason
ISBN : 9783319326931 Title : Human Dignity of the Vulnerable in the Age of Rights
ISBN : 9783319326931 Title : Human Dignity of the Vulnerable in the Age of Rights
ISBN : 9783319012223 Title : Most-Cited Scholars in Criminology and Criminal Justice, 1986–2010
```

```
Deep learning based model recommendations books :
ISBN : 9781453237915 Title : West with the Night
ISBN : 9781599908755 Title : Starcross
ISBN : 9781596431317 Title : Drawing Words and Writing Pictures
ISBN : 9780143111597 Title : The Left Hand of Darkness
ISBN : 9782080201331 Title : Master Photographers
ISBN : 9780385372091 Title : Green Eggs and Ham: Read & Listen Edition
ISBN : 9780757305603 Title : The Sleepeasy Solution
ISBN : 9780821215517 Title : Examples
ISBN : 9780936861579 Title : Captives of Blue Mountain
ISBN : 9780805800586 Title : The Psychology of Learning Mathematics

Hybrid approach based books recommendations

ISBN : 9781524742393 Title : America's Reluctant Prince
ISBN : 9781501194313 Title : Howard Stern Comes Again
ISBN : 9780553448122 Title : Artemis
ISBN : 9781524731663 Title : Bad Blood
ISBN : 9781466861244 Title : The Calculating Stars
ISBN : 9780553448122 Title : Artemis
ISBN : 9789042911833 Title : Spirituality
ISBN : 9789042019485 Title : Cyberculture, Cyborgs and Science Fiction
ISBN : 9781453237915 Title : West with the Night
ISBN : 9781599908755 Title : Starcross
```

VI. Conclusion:

Using various approaches of recommendation, we can get better and more accurate results. I have tried to implement each of the approach and shared the link to Google drive in the email.

Google drive link:
https://drive.google.com/drive/folders/1jUPEDHxnEXvFcykRtYV-EagJfOq8M13n?usp=sharing

GitHub link:
https://github.com/PurvaDeekshit/CMPE-256-Individual-Project