Book Recommendation System

Anshika Pal
Department of Computer Science
PES University

Channabasavanna Hatti
Department of Computer Science
PES University

Rohan
Department of Computer Science
PES University

Abstract:

The main objective is to create a book recommendation system for users. Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. Collaborative Filltering, Content-Based Filtering is used. And in Collaborative filtering, user-Based and Item-Based Collaborative filtering is used.

" A good recommender system has to consider how users interact with the recommendations."

A recommendation system broadly recommends items to the user best suited to their tastes and traits. It uses the user's previous data and other user's data to give new recommendations.

I. INTRODUCTION

Recommendation systems are used in hundreds of different services - everywhere from online shopping to music to movies. For instance, the online retailer Amazon had a heavy hand in developing collaborative filtering algorithms that recommend items to users. Music services like Pandora identify up to 450 uniquely identifying characteristics of songs to find music similar to that of their users' preferences. Other music streaming services, such as Spotify, heavily rely upon the music selections of similar users to make weekly song recommendations and personalized radio stations. Netflix, a popular television and movie streaming service, uses these systems to recommend movies that viewers may enjoy. We can see how recommendation systems have a surprisingly large impact on the materials consumers engage with over the course of their daily lives.

For our senior comprehensive project, we analyzed three systems that predict how users will rate specific books. Our system that we created makes these predictions based on data gathered from the BookCrossing dataset. To accurately predict users' reactions to books, we've integrated several strategies in the field of recommendation systems.

II. BACKGROUND

There are three major approaches for recommendation systems: (i) content-based, (ii) collaborative, and (iii) Machine learning based model. Broadly, recommendation systems that implement a content-based (CB) approach recommend items to a user that are similar to the ones the user preferred in the past. On the other hand, recommendation systems that implement collaborative filtering (CF) predict users' preferences by analyzing relationships between users and interdependencies among items; from these, they extrapolate new associations.

III. DATA

For this Project, we used **Book-Crossing(BX) Dataset.** This Dataset contains 278,858 users(anonimyzed but with demographic information) providing 1,149,780 ratings(explicit and implicit) about 271,379 books.

Dataset Description:

i. BX-Books.csv has 8 columns which include information about books.We can uniquely identify each book with the help of ISBN. The tile, author and publisher and year of publication of each book has been listed. The last three columns include URLs for different sizes of images.
ii. BX-Users.csv: lists the age and location of a user along with their User-ID.
iii. BX-Book-Ratings.csv: contains book ratings given by users. The User-ID and ISBN have also been provided.

IV. DATA PREPROCESSING

We had some extra columns in BX-Users.csv which are not required for our task like image URLs. And we had renamed the columns of each file as the name of the column contains space, and uppercase letters so we will correct as to make it easy to use. Each algorithm has constraints, which has caused us to sample the dataset

If we take all the books and all the users for modeling, Don't you think will it create a problem? So what we have to do is we have to decrease the number of users and books because we cannot consider a user who has only registered on the website or has only read one or two books. On such a user, we cannot rely to recommend books to others because we have to extract knowledge from data.

We will limit this number and we will take a user who has rated at least 200 books and also we will limit books and we will take only those books which have received at least 50 ratings from a user. This will give us 105283 peoples have given a rating among 278000. And then we will extract the ratings of those user-ids. We got 900 users who have given 5.2 lakh rating and this we wanted. Then we merged ratings with books on basis of ISBN so that we will get the rating of each user on each book id and the user who has not rated that book id the value will be zero. We got our dataset size as 4.8 lakh and dropped the duplicate values.

Finally, we have a dataset with that user who has rated more than 200 books and books that received more than 50 ratings. the shape of the final dataframe is 59850 rows and 8 columns.

Libraries used: pandas,missingo, surprise, numpy, matplotlib, pycountry_convert, plotly, wordcloud.

V. DATA ANALYTICS

The primary goal of EDA is to support the analysis of data prior to making any conclusions. It may aid in the detection of apparent errors, as well as a deeper understanding of data patterns, the detection of outliers or anomalous events, and the discovery of interesting relationships between variables.

In Fig1, we have displayed the unique users accross the globe . We defined the function get_alpha3() and get_name() to convert the full form of country's name to it's short form and vice versa.

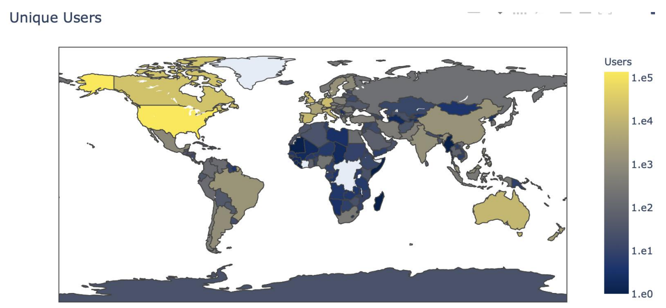


Fig1 – Unique Users

In Fig2, we have displayed the top 10 user's cities in terms of barplot. This gives us the clear picture that London has most numbers of users followed by barcelona and toronto.

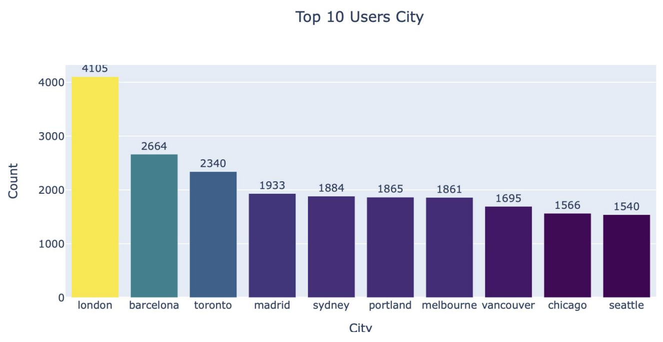


Fig2 – Top Ten Users City

In Fig3, we have displayed the top ten books published by years which will give help us to identify the books which is widely liked by users and has given higher ratings by the users.

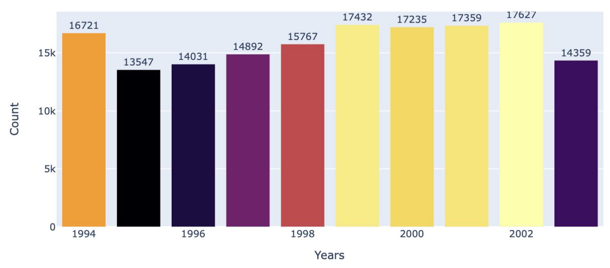


Fig3 – Top Ten Number Of Books Published By years

In Fig4, we have displayed the barplot of Top 10 Publishers which will help us to recommend the book to the user in terms of Content-Based Filtering.

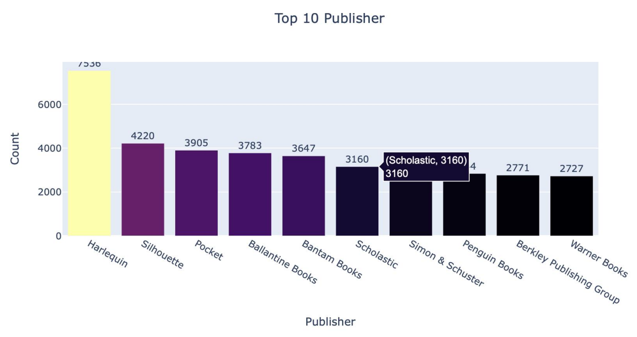


Fig4 – Top 10 Publishers

In Fig4, Mpst Reviewed Books have been displayed and we can see 'The Lovely Bones' is very popular book followed by 'Wild Animus'.

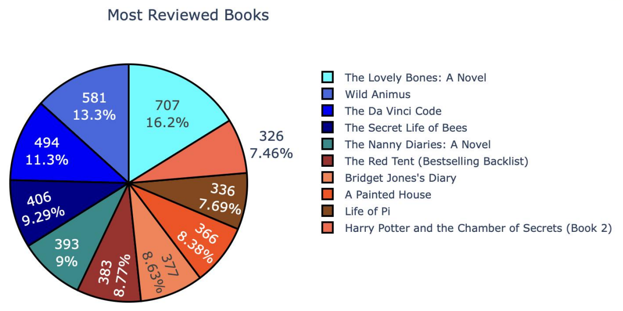


Fig5 – Most Reviewed Books

In Fig6, Top ten Best Books is displayed which are books are given highest ratings.

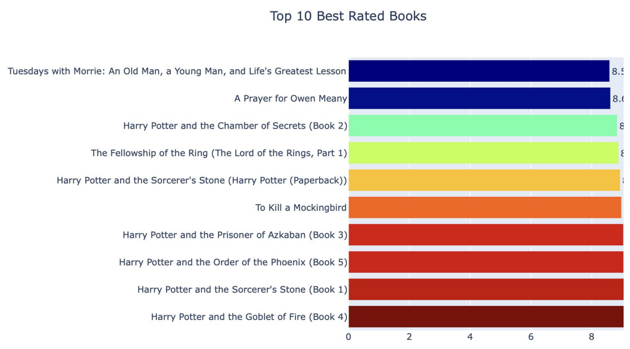


Fig6 – Top Ten Best Rated Books

VI. MODELS AND RESULTS

COLLABORATIVE-BASED FILTERING

We used KNN Algorithm for Collaborative-Based Filtering(Fig7).Collaborative Filtering Using k-Nearest Neighbors (kNN) kNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors.
Further, there are several types of collaborative filtering algorithms —

i. User-User Collaborative Filtering
ii. Item-Item Collaborative Filtering

In K Nearest Neighbors for collaborative filtering, we used the number of k people who most similar to the person we are looking for to find good recommendations. The best value for k depends on the problem. We have used KNN with Means algorithm for building user-based recommender system. This algorithm takes into account the mean ratings of each user. The prediction $\hat{r}_{ui}$ is set as:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} \text{sim}(u, v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} \text{sim}(u, v)}$$

where sim(u,v) is the similarity of two users u and v. We use cosine similarity measure to compute the closeness of users with each other.
UserBased Filtering:

The current implementation get neighbors of target user based on similarity measure. We find k nearest neighbors and use their ratings to recommend the items to the target user. In this method, k nearest neighbors for the target user is fixed and won't change
We have created the class:
UserBasedCollaborativeFiltering() and defined normalize(), compute_similarity(), create_similarity_matrix(), get_neighbors(), score_item() and recommend() functions in it.

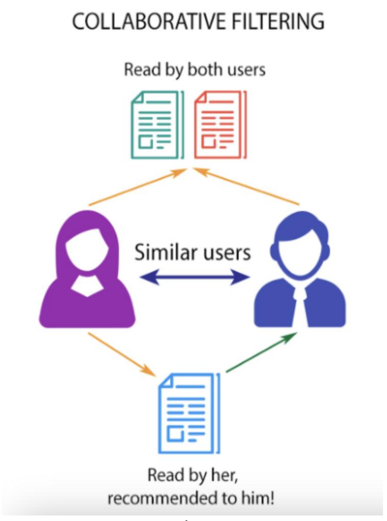
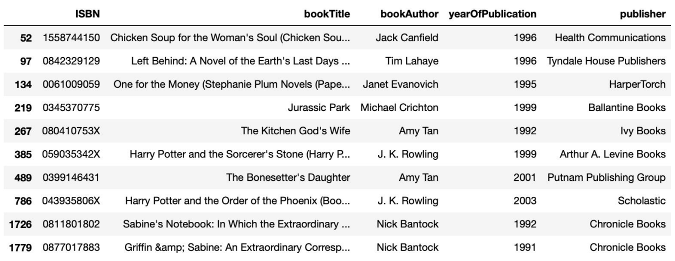


Fig7

normalise() function is summing the rowsand counting non-zero values and then taking the mean. By iterating the columns, we are subtracting the mean values.

compute_similarity() is calculating the cosine similarity of two users. create_similarity_matrix() is creating the cosine similarity matrix between all the users. get_neighbors() is calculating the neighbours on the basis of matrix created. recommend() is recommending the top users that a user is matched with.

For e.g., if UserId = 23872, then top ten recommendations are shown below.



	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher
52	1558744150	Chicken Soup for the Woman's Soul (Chicken Sou...	Jack Canfield	1996	Health Communications
97	0842329129	Left Behind: A Novel of the Earth's Last Days ...	Tim Lahaye	1996	Tyndale House Publishers
134	0061009059	One for the Money (Stephanie Plum Novels (Pape...	Janet Evanovich	1995	HarperTorch
219	0345370775	Jurassic Park	Michael Crichton	1999	Ballantine Books
267	080410753X	The Kitchen God's Wife	Amy Tan	1992	Ivy Books
385	059035342X	Harry Potter and the Sorcerer's Stone (Harry P...	J. K. Rowling	1999	Arthur A. Levine Books
489	0399146431	The Bonesetter's Daughter	Amy Tan	2001	Putnam Publishing Group
786	043935806X	Harry Potter and the Order of the Phoenix (Boo...	J. K. Rowling	2003	Scholastic
1726	0811801802	Sabine's Notebook: In Which the Extraordinary ...	Nick Bantock	1992	Chronicle Books
1779	0877017883	Griffin &amp; Sabine: An Extraordinary Corresp...	Nick Bantock	1991	Chronicle Books

Fig8

DRAWBACKS OF USER-USER COLLABORATIVE FILTERING:

Imagine you have an online music service that has an internal recommmnder system. Where can the problem occur? Suppose this system has a million users. Every time you want to make an offer for a person, you have to compare the distance of this person with 1000000 other people.

Now suppose we want to make several recommendations to different people every second. In this way, the calculations will be very complex and difficult. Except when you spend a lot of money to buy servers, the system will slow down. In more formal terms, latency can be a major problem in neighborhood-based recommender systems. Two main problems with working with user-based recommender systems are as follows:

i. scalability

As the number of users increases, so does the number of calculations. User-based methods may work well with thousands of users, but with millions of users, their "scalability" is problematic.

ii. sparsity

Many of recommender systems have a large number of users and items. But each user, on average, rates a small percentage of their products. For example, Amazon has millions of books, but on average, the users have rated a handful of these books. For this reason, neighborhood-based algorithms may not be able to find any close neighbors for a person. Because of these two problems, it is better to use item-based filtering.

Item Based Filtering:

In the user-based method, we considered a user and were trying to find the person most similar to him / her among other users so that we can find recommendations for this user by the rating of the others. But in item-based filtering, before the recommendation time, we find the most similar items and combine them with user ratings so that we can form recommendations.

Now, we used Weighted Slope One algorithm for item-based filtering.In this algorithm, it's defined a concept called deviation for items dev(i,j) is defined as the average difference between the ratings of i and those of j:

$$\operatorname{dev}(i,j) = \sum_{u \in S_{i,j}(X)} \frac{u_i - u_j}{card(S_{i,j}(X))}$$

Where card(S) is the number of elements in S and X is the number of all ratings. So card(Si,j(X)) is the number of people who have rated both items i and j.

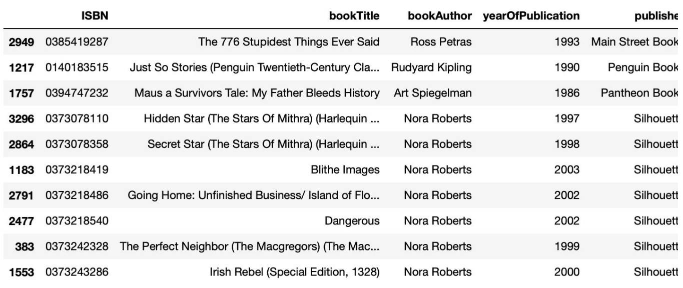
After calculating the deviations between all items, we used the following formula for predicting the rating of user u to item j. The prediction r̂uj is set as:

$$\hat{r}_{uj} = \frac{\sum_{i \in S(u)-j} (\operatorname{dev}(j,i) + u_i) c_{j,i}}{\sum_{i \in S(u)-j} c_{j,i}}$$

Now, we defined ItemBasedCollaborativeFiltering class in python to work with all the functions with ease and inside that we defined prepare_data(), compute_deviations(), slope_one_recommend(), recommend() functions.

prepare_data() function is preparing data by passing user ratings into the list called user_ratings and returing it as dictionary. compute_deviations() is calculating the deviation of the ratings. recommend() function will recommend the top books to the user on the basis of item-item based filtering.

For e.g., if user_idex = 1, then result will be(Fig9):



	ISBN	bookTitle	bookAuthor	yearOfPublication	publishe
2949	0385419287	The 776 Stupidest Things Ever Said	Ross Petras	1993	Main Street Book
1217	0140183515	Just So Stories (Penguin Twentieth-Century Cla...	Rudyard Kipling	1990	Penguin Book
1757	0394747232	Maus a Survivors Tale: My Father Bleeds History	Art Spiegelman	1986	Pantheon Book
3296	0373078110	Hidden Star (The Stars Of Mithra) (Harlequin ...	Nora Roberts	1997	Silhouett
2864	0373078358	Secret Star (The Stars Of Mithra) (Harlequin ...	Nora Roberts	1998	Silhouett
1183	0373218419	Blithe Images	Nora Roberts	2003	Silhouett
2791	0373218486	Going Home: Unfinished Business/ Island of Flo...	Nora Roberts	2002	Silhouett
2477	0373218540	Dangerous	Nora Roberts	2002	Silhouett
383	0373242328	The Perfect Neighbor (The Macgregors) (The Mac...	Nora Roberts	1999	Silhouett
1553	0373243286	Irish Rebel (Special Edition, 1328)	Nora Roberts	2000	Silhouett

Fig9

CONTENT-BASED FILTERING

Content-based filtering, also referred to as cognitive filtering, recommends items based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document. A content-based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make recommendations to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

After cleaning and preparing the data, we can use TF-IDF representation to make our recommender system.
TF is simply the frequency of a word in a document. IDF is the inverse of the document frequency among the whole corpus of documents. TF-IDF is used mainly because of two reasons: Suppose we search for “the rise of analytics” on Google. It is certain that “the” will occur more frequently than “analytics” but the relative importance of analytics is higher than the search query point of view. In such cases, TF-IDF weighting negates the effect of high frequency words in determining the importance of an item (document).

Here is a funny description of TF-IDF representation(Fig10):

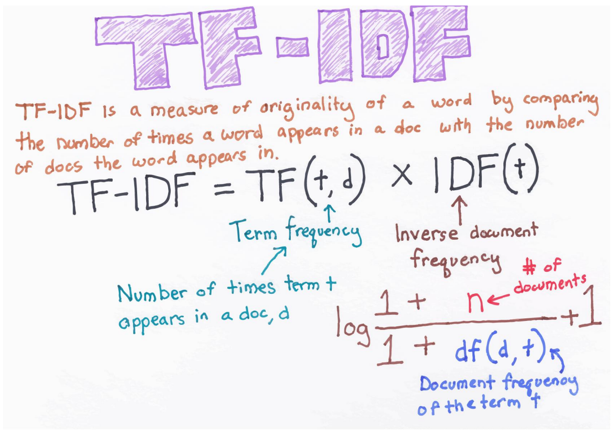
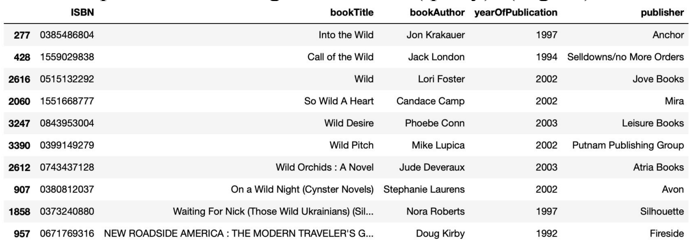

TF-IDF
TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.
TF-IDF = TF(t, d) × IDF(t)
Term frequency
Inverse document frequency
Number of times term t appears in a doc, d
# of documents
log (1 + n) / (1 + df(d, t)) + 1
Document frequency of the term t

Fig10

We have defined a class ContentBasedFiltering() and defined the followinf functions: create_embedding_matrix(), create_kernel(), create_indices(), recommend().

In create_embedding_matrix(), TfidfVectorizer(python) is used to create the tf-idf matrix and min_df = 3, max_features = None, strip_accents = 'unicode', analyzer = 'word', token_pattern = r'\w{1,}', ngram_range = (1, 3), stop_words = 'english' features are passed into the function and returning the matrix. In create_kernel() sigmoid_kernel is returned. recommend() is used to recommend the top books to the user.

For e.g., query ‘Wild Animus’ is passed, it will recommend the the top 10 books using recommend(query). (Fig 11)



	ISBN	bookTitle	bookAuthor	yearOfPublication	publisher
277	0385486804	Into the Wild	Jon Krakauer	1997	Anchor
428	1559029838	Call of the Wild	Jack London	1994	Selldowns/no More Orders
2616	0515132292	Wild	Lori Foster	2002	Jove Books
2060	1551668777	So Wild A Heart	Candace Camp	2002	Mira
3247	0843953004	Wild Desire	Phoebe Conn	2003	Leisure Books
3390	0399149279	Wild Pitch	Mike Lupica	2002	Putnam Publishing Group
2612	0743437128	Wild Orchids : A Novel	Jude Deveraux	2003	Atria Books
907	0380812037	On a Wild Night (Cynster Novels)	Stephanie Laurens	2002	Avon
1858	0373240880	Waiting For Nick (Those Wild Ukrainians) (Sil...	Nora Roberts	1997	Silhouette
957	0671769316	NEW ROADSIDE AMERICA : THE MODERN TRAVELER'S G...	Doug Kirby	1992	Fireside

Fig11

MACHINE LEARNING BASED MODEL

We will limit the books in this whose ratings are given atleast 5 times. We are normalizing the ratings on the basis of gaussian normalization.

Rating Normalization(Gaussian Normalization):

$$R^{u_i}_{norm}(b) = \frac{R_b - R^{u_i}_{mean}}{\sqrt{\sum_j (R_{b_j} - R^{u_i}_{mean})}}$$

We are filling the na values by 0 and normalizing the mean user ratings.
Surprise is a Python scikit for building and analyzing recommender systems that deal with explicit rating data. Provide various ready-to-use prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based (SVD, PMF, SVD++, NMF), and many others. Also, various similarity measures (cosine, MSD, pearson…) are built-in. By iterating through all the models on surprise i.e. SVD(), SVDpp(),SlopeOne(), NMF(), NormalPredictor(), KNNBaseline(), KNNBasic(), KNNWithMeans(), KNNWithZScore(), BaselineOnly(), CoClustering, we are getting the RMSE values, fit time and test time we are checking. We chose Baseline Algorithm as it gave us the smallest RMSE values. Due to the previous section, we choose Baseline Only algorithm which predicts the baseline estimate for given user and item. A baseline is a fixed point of reference that is used for comparison purposes.

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

where bu and bi are the estimation biases for user and item, respectively.
If user u is unknown, then the bias bu is assumed to be 0. The same applies for item i with bi.

Baselines can be estimated in two different ways:
1. Stochastic Gradient Descent (SGD)
2. Alternating Least Squares (ALS)

We use SGD procedure for our project.
RMSE values that we got:-

```
Estimating biases using sgd...
RMSE: 0.6161

0.6160890840544051
```

In our project we are recommending user_id = 36938.
Results:

	ISBN	Rating	bookTitle	bookAuthor	yearOfPublication	publisher
3367	0345342968	3.035087	Fahrenheit 451	RAY BRADBURY	1987	Del Rey
675	0446310786	3.051375	To Kill a Mockingbird	Harper Lee	1988	Little Brown &; Company
2739	0345361792	3.032526	A Prayer for Owen Meany	John Irving	1990	Ballantine Books
6117	059035342X	3.043820	Harry Potter and the Sorcerer's Stone (Harry P...	J. K. Rowling	1999	Arthur A. Levine Books
6730	0671027360	3.032648	Angels &amp; Demons	Dan Brown	2001	Pocket Star
3227	0316666343	3.047491	The Lovely Bones: A Novel	Alice Sebold	2002	Little, Brown
1892	0385504209	3.061188	The Da Vinci Code	Dan Brown	2003	Doubleday
5715	0142001740	3.057554	The Secret Life of Bees	Sue Monk Kidd	2003	Penguin Books
8264	0156027321	3.034544	Life of Pi	Yann Martel	2003	Harvest Books
4925	043935806X	3.031039	Harry Potter and the Order of the Phoenix (Boo...	J. K. Rowling	2003	Scholastic

Acknowledgment

We would like to thank PES University for giving us this opportunity to research on this topic. We would like to thank, in particular Prof K S Srinivas, who gave his invaluable guidance, without whom this would be impossible to accomplish.

REFERENCES

1. Book Recommendation System through content based and collaborative filtering method Praveena Mathew; Bincy Kuriakose; Vinayak Hegde
2. Book recommendation system using opinion mining technique Shahab Saquib Sohail; Jamshed Siddiqui; Rashid Ali
3. Online book recommendation system
4. Collaborative Filtering for Book Recommendation System Gautam Ramakrishnan, V. Saicharan, K. Chandrasekaran, M. V. Rathnamma & V. Venkata Ramana
5. A novel approach for book recommendation systems P Devika; R C Jisha; G P Sajeev