Recommender System

Rauf Fatali, Javidan Abdullayev, Aliisa Alizada
Department of Applied Computer Science, UFAZ, Azerbaijan,
rauf.fatali@ufaz.az, javidan.abdullayev@ufaz.az,
aliisa.alizada@ufaz.az

Abstract. The subject of recommender systems has become one of the top facilities of nowadays technology for successful business in e-commerce. Recommender systems help users to get recommendations of certain products according to their preferences that users appreciate with it. Huge business companies such as Amazon, Netflix, Spotify, Pandora, Linkedin have begun to use several recommender systems approaches to recommend their products in the recent two decades, and they have played a major role in the development of these approaches.

Recommender systems have several approaches to implement, and mainly they are covered by 3 main approaches: Content-based filtering, Collaborative filtering, and Hybrid approaches. These approaches differ from each other due to different resource types. For example, the Content-based filtering approach uses features of items, the Collaborative filtering approach uses users' relationships to predict users' preferences, and the final one which is the Hybrid approach contains the combination of Content-based filtering and Collaborative filtering approaches.

In this report, we mainly focused on the concept of implementation of these recommender systems approaches on the basis of books. After the application of the data preprocessing phase, each approach was implemented with the helping of several strategies, and all experimental results of each approach were covered.

Keywords. Recommender Systems, Book Recommender Systems; Content-based filtering; Collaborative filtering; Hybrid Approach.

Introduction

E-commerce is one of the constantly evolving areas in the industry and ever-growing e-commerce is one of the key factors that develop the industry of each country. With the increasing demand for the Web, this evolving field has benefited from the use of various technologies to provide high-quality services to users. One of those technologies is recommendation systems (RS). In recent decades interest in RS has been grown since the presence of the first works on this topic in the mid-1990s [1]. At that time the ACM Conference was especially noteworthy because of contributions to many advanced results on RS [2].

RS can be defined as the recommendation of various items to users that suit users' previous preferences. The main goal of RS is to help users find the products more quickly and easily that they are looking for. From this aspect, the successful implementation of the RS is very crucial for both users and e-commerce [3]. It is possible to sell more products and services by increasing user satisfaction with giving more accurate recommendations, on the other hand, we can also lose many and many users with failed recommendations. The development of such RS mainly depends on the domain and the characteristics of the available resources. These domains may vary for each e-commerce field.

The next section first discusses the approaches that are used by leading companies today, and then covers information on the book RS and approaches we will work on.

1. State of the Art

Based on real-world examples, many companies like Netflix, Spotify, Pandora, Linkedin, and Amazon use several approaches of RS to promote their products and services and increase their revenue. Netflix uses a two-tiered row-based approach for its ranking system [4]. For example, each row has its own theme and contains several movies related to that theme. The strongest recommendation is located on the left for each row, and for each theme strongest one is located on top. This explained example is only one of the approaches that Netflix uses [4]. Another example we can give from RS of Amazon company. One of the used approaches in

Amazon is item-to-item collaborative filtering [5]. Amazon personalizes the online store for each user with the helping of RS and each store can change at any time depending on user preferences [5]. Providing recommendations in the best way for each company can play a crucial role in developing themselves from the beneficial side. For example, with the implementation of RS approaches, Netflix saves up to $ 1 billion in a year [4]. Amazon also derives 35% of its revenue from this aspect like Netflix [5].

In this project, we aimed to make RS based on the domain of the books. The increasing demand for books by readers in recent years has encouraged us to contribute to this field. We can see that the number of readers is developing day by day depending on different aspects and the main goal of each reader is to get items from reliable sources that meet their needs. If the user successfully obtains the desired item from the identified source from the first search, the next step is already on the RS. In this case, RS should begin to give the appropriate recommendations to the user based on previous preferences [6]. As we mentioned before, the development of such effective RS depends on the strong implementation of 3 main approaches together are Content-based filtering (CB), Collaborative filtering (CF), and Hybrid approach.

1.1 Content-based filtering

Firstly, we start with *CB* filtering which makes recommendations based on mainly book description and some metadata information like book's genre and author. In *CB* filtering, we do not need user ratings or any other information. We only need description and meta-data of books to recommend them to the users [7]. In this approach, we modeled books as a set of numbers. That's we created a quantitative model to represent each book from our dataset. The dataset has a column **description** that describes what is the content of book. Using that column, we first create a quantitative model, and then in further steps to recommend books to users we consider some metadata information like book's genre and author.

It is hard to recommend items to new users and it is also hard to recommend new items to the users. This method is also useful when we have products in a dataset that have few or no ratings which are called the **Cold Start** problem [7]. *CF* filtering can’t handle such situations well therefore *CB* filtering is the indispensable approach in such cases [7]. *CB* filtering also has some drawbacks like the “**serendipity problem**” in which this approach doesn’t consider what users have already rated therefore some recommendations can be dissimilar to users’ interests.

1.2 Collaborative filtering

CF is the most extensively used approach in today’s systems and the aim is to suggest useful items to users. In CF, systems recommend books based on similarities among books’ properties. CF systems consider the past relationships between users and interdependencies among items. CF techniques are very popular especially for e-commerce systems [9]. It can be seen from the implementation of the CB approach that it recommends items based on their description and metadata. Therefore, the CB approach is effective when user data is in text format [9]. When the data is in multimedia or other formats, there are some difficulties in the analysis of items by machine. Although it is possible to solve the problem to some extent using human assistance in the analysis of items, we can say that CB filtering is not enough to work with most of the data which is available today. Another drawback of the CB approach is the lack of ability to estimate the quality of items. In CF, the system does not analyze the content of items and recommendations depend on only users' opinions. We can say that the above-mentioned drawbacks are not felt in CF, and this approach allows us to work with all types of data that are available today [9].

In this step, we give general information on the implementation of CF. Usually, CF is classified into two broad categories which are called user-based and item-based collaborative filtering. In our project, we implement both approaches and both implementations are covered in the experimental section.

1.2.1 User-based filtering

In the user-based approach, we find out users that have similar behaviors that they have bought and liked similar books in the past. The advantage of this method is that it does not require any information besides users’ historical priorities on a set of items [10]. Accuracy and efficiency of recommendation systems increase parallelly with user interactions [11]. Let’s say there are two users *A* and *B* in our system that they like and dislike similar books. If user *A* buy a new book and liked it then there would be more probability that user *B* also would like the new book. Therefore, such systems immediately will recommend that book to user *B*.

1.2.2 Item-based filtering

The main idea behind the item-based approach is that if a group of people rated two books similarly then those books are similar. If a user bought one of those books then our system will recommend another one to that user. Let's if a group of users gave 5 ratings to the books *A* and *B*. Then if another user buys the book *A* then our system will recommend the book *B* to that user or vice versa [7].

User-based and item-based approaches are also called the memory-based approach. Usually, their results are based on similarity matrices that are these filterings learn parameters from the data to the data and we do not use machine learning algorithms in our approach. The third and last approach of RS that we covered is the Hybrid approach.

1.3 Hybrid approach

Nowadays, huge e-commerce companies as like as Netflix increasingly utilizes the Hybrid approach. Such companies increase both their revenue and user satisfaction with this approach [7]. In general, Hybrid approach is an approach that results from a combination of two or more approaches of RS (cf. Figure 1). The implementation of Hybrid approach increases the possibility of overcoming the disadvantages of a single approach and generalizing the advantages of different approaches. If we take into consideration the CB and CF approach separately, in the CF approach we can see that the problem arises when there are limited data, and also, we are not able to use the rating data in the CB approach [12]. The next section covers a brief description of each dataset we implemented for our RS.

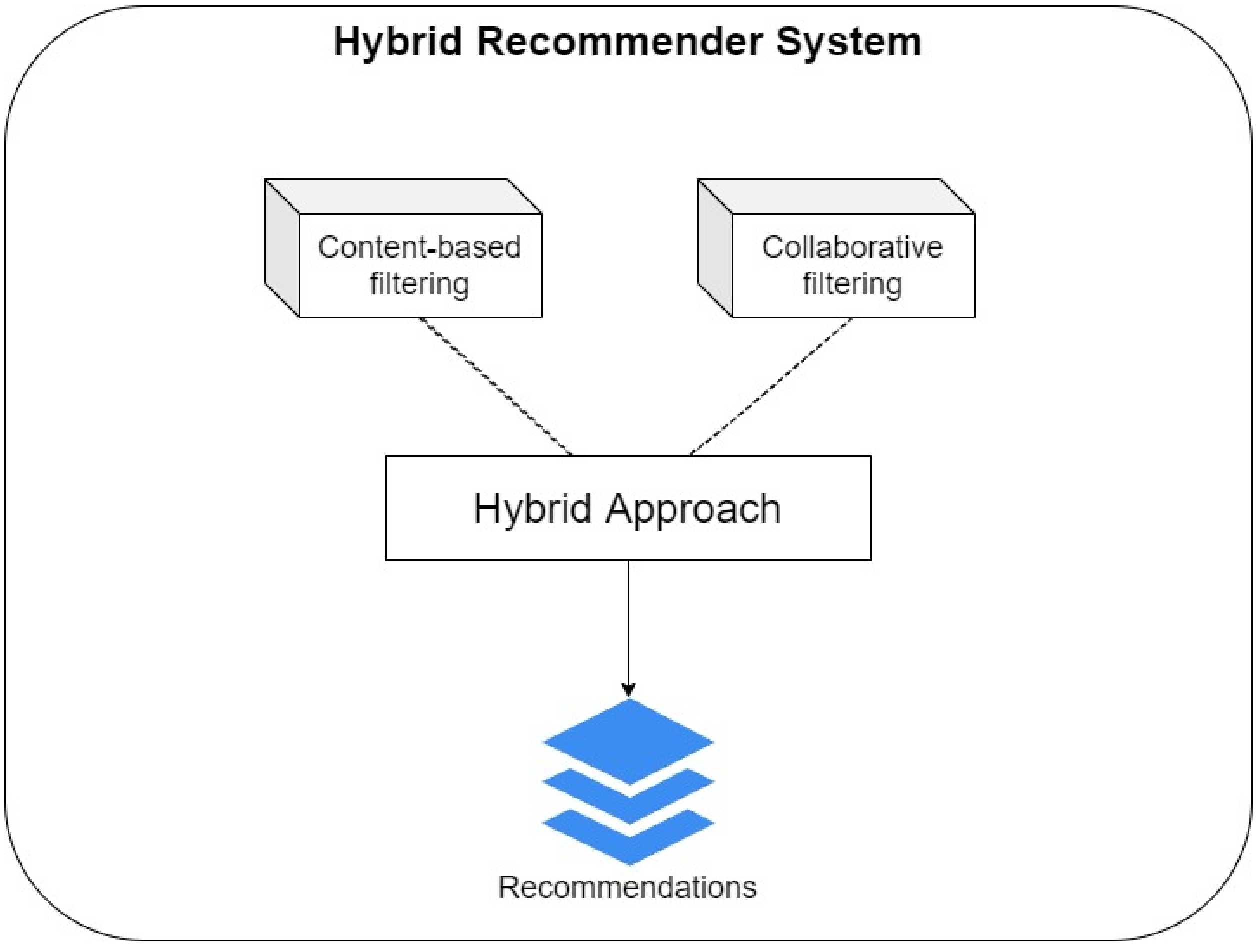


Figure 1. The architecture of Hybrid Recommender Systems.

2. Data Description

For our book recommendation system, we used three different datasets in CSV format and each dataset has its own characteristic features related to books: The first dataset is *description.csv*, the second dataset is *listing.csv*, and the third one is *books.csv*.

description.csv contains 3 columns and each column is the feature for our domain. The first feature which is common for all datasets is the ID number for each book. Others contain the name and brief description of each book. description.csv dataset consists of 122 unique books. The data types for these features are ***int64*** according to the ID and ***object*** according to the name and content. The second dataset which is *listing.csv* contains 4 features. ID number and name of the books still remaining the same in this dataset. *listing.csv* database consists of 30 different genres and covers many books that are dominated by ***travel*** genre with 90 books, ***business & money*** genre with 77 books, ***children's books*** genre with 64 books (cf. Figure 2). With the helping of this dataset, we mainly used the meta-data which is the genre for our implementation. In general, we utilized *description.csv* and *listing.csv* datasets for a CB filtering approach.

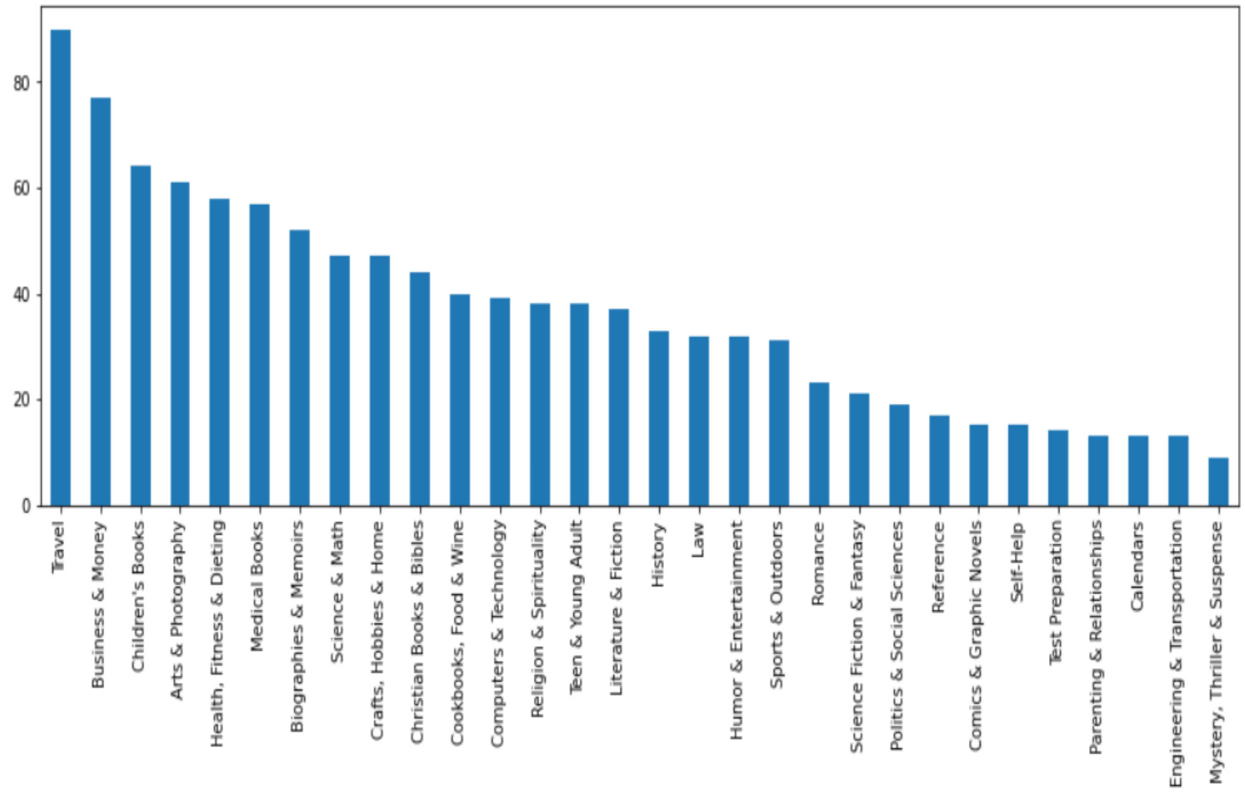


Figure 2. Histogram of genres and counts according to *listing.csv*.

The last dataset which is called *books.csv* can be considered as the crucial one for our RS due to the description of the ratings for each book. This dataset contains 5 features and the data type for each feature is *int64*. Features are ID number of books, average rating, number of ratings, ID number of users, and users' rating. According to the *books.csv* dataset 190 users rated 710 books between the range of 0 and 5. The dataset *books.csv* is used in the CF approach and all datasets are used in the Hybrid approach.

3. Experiments and Results

3.1 Content-based filtering

3.1.1 Applied Approaches

Before starting actual code implementation, we preprocessed the text of all descriptions. We converted all letters to *lower case* and removed *ASCII characters, stop words*, *html,* and *punctuation* from the description column. We started our implementation by modeling each book using a *TF-IDF* vectorizer. Using *TF-IDF* we modeled each book as a vector of numbers. Then we used the *cosine similarity* [16] metric to measure similarity between books' vectors. After doing a test on our model, we realized that the model doesn't capture the semantic meaning, and also it gave us a sparse matrix to process. To handle this embedding ***word2vec*** [8] difficulties we search on the internet and found a very excellent method called word

Currently, Neural Networks are the most famous architecture to capture semantic meaning of word representations which is called *word embedding* [13]. Another advantage of word embedding is that it creates dense, low dimensional features that processing it can be much easy in contrast to *TF-IDF* which creates very sparse, high dimensional features. Like *TF-IDF* the embedding *word2vec* takes words and then returns a vector in *D-dimensional space.* In many cases, D would be between 50 and 500 [8]. We decided to use 300-dimension vectors for this recommendation engine. This isn't a sharp choice for dimension. We just selected that value as an average value between 50 and 500. Usually using embedding word2vec outperforms *TF-IDF* [8].

We don't need to train our own word embedding *word2vecs* because this is a very expensive process and it also requires a large dataset. Our dataset only for about 120 books has a description which is less information. Therefore, we will load *Google pre-trained word embedding* which was trained on a large corpus, including Wikipedia, news articles, and much more [8].

Using pre-trained word embeddings, we can get the word vector for each word. We can think of it as a dictionary in python which has key-value pairs and each key represent the word and each value represent the corresponding word vector.

We first convert the description of each book to a vector and then find the similarities among book vectors using a cosine similarity measure. To do so, we have tried two methods: average Word2vec and TF-IDF Word2Vec [14] [15].

In our TF-IDF Word2vec implementation, we applied different strategies that we considered metadata information of book in our recommendations. After finding the similarity score between two books we check that if the genre of the two books is different. If so, we then divided that similarity score by 2. Another hint is that if the author of the two books is the same then we multiply similarity score by 1.1 just to increase a bit the similarity score to have better recommendations.

We would say another interesting idea in our implementation. The idea was that it can be better to partition the dataset into clusters and recommend books to users based on that clusters. Let's say if I have a book in our dataset that has an "average rating" value equals to 1 then it isn't a good idea to recommend that book.

To handle this situation, we used the *k-means* method from the **scikit-learn** library. Using the *elbow* method, we found the best **k** to partition the dataset. In our case was 2 but we didn't use that value to partition our dataset. Our dataset is imbalanced therefore we chose k as 10 to balance the dataset. After partitioning the dataset, we took only the cluster which has the most "average rating" and the most "number of ratings" values. Then we recommended books to users considering only that cluster and skip the other cluster.

3.1.2 Results

In this part, we are going to explain different results that we got by applying different methods. First, we applied average Word2Vec approach. Then we put the book "Fantasy League" as input in the recommendation function and that function recommended 5 books to us. The genre of the book "Fantasy League" is "Children's Books" but there is no recommended book in that genre. The recommended four books are in the genre "Biographies & Memoirs" and only one book is in the genre "Teen & Young Adult". From the user tester's point of view, it isn't satisfied with recommendations.

Then we applied a new approach which is called TF-IDF Word2Vec. Similarly, we got five recommendations for the book "Fantasy League". In this case, the recommended three books are in the genre "Biographies & Memoirs" and the other one is in the genre "Teen & Young Adult" and the last one has the genre "Romance".

It is not satisfactory to the user tester. In general TF-IDF Word2vec works better than average Word2vec [8].

To handle this situation, we applied a different strategy. As we explained in the implementation section, we first check the genre of each book with the book in question and if the genres are different then we divide the similarity score by 2 and if the authors are the same then we multiply the similarity score by 1.1 just to increase that score a bit. After applying this strategy inside TF-IDF Word2vec method we got 5 recommendations which all of them are in the genre “Children's Books”. According to the user tester, this was an excellent result.

Actually, we had a little bit problem here that inside our recommendation we had a book with an average rating of 3.12. In general, this is not a good recommendation because the average rating is less to recommend. Till now we recommend books from the whole dataset which had 122 books with descriptions. Therefore, it is possible to recommend books with a low average rating. To overcome this issue, we used partitioned datasets. We partitioned the dataset into ten clusters and combined nine clusters as one cluster. The cluster which is the combination of nine clusters has 47 books. The reason for combining the nine clusters into one cluster is that originally our dataset is imbalanced to get a bit balanced dataset we used such an approach. These books inside the cluster have more average ratings and more people rated them. Therefore, recommending from that cluster is better to overcome previous issue. Then we put the book “Fantasy League” as input in the recommendation function and that function again recommended 5 books to us. Three of the recommended books are in genre of “Children's Books” and two others are in the genre of “Biographies & Memoirs”. Because of the genre property of books, it may not be good as previous which all recommended books are in genre of “Children’s Book”, but the minimum average rating among recommended books is 3.95 which is better. We benefited from the data partitioning approach in this case.

3.2 Collaborative Filtering

3.2.1 Applied Approaches

As we said before, we applied both the user-based and item-based approaches for our system. In this section, we covered the approaches that we applied for our system step by step.

User-based approach

We have implemented the user-based CF on our dataset in the following steps:

i. Create a set of ratings given to different items by users
ii. Evaluate the similarities among users using cosine similarity
iii. Calculate the predicted value of rating using the neighbor ratings.

We used the *book.csv* file to find similarities among users. We need three columns from this dataset such as **book_id**, **user_id,** and **user_rating** and for this reason, other columns dropped from the dataset. We obtain a set of users, a set of books, and ratings which given by users to books. In this dataset, 710 books have been rated by 84 users and the shape of a new dataset is (2189, 3). For the next step, the dataset is split to train and test datasets with scikit-learn's *train_test_split* function. We allocated 25% of the data for the test part and 75% for the train part and our target variable is **user_id** which data stratified according to the **user_id**. Before stratify data, some restrictions have to add to the **user_id**. Users who rated only one book have to be deleted from the dataset because of stratifying principles. To evaluate the performance of regressors, we used the *Root Mean Squared Error* metric and it can be implemented from the scikit-learn library [7]. In the CF model, **user_id** and **book_id** are input values, and **user_rating** is an output value that is rated between 1 and 5. In user-based CF we find similar users based on ratings which they rated books. We used the pivot-table function of the pandas library to see users' ratings for certain books. In the table, some not a number (NaN) values may obstacle with the operation of the cosine similarity function and they replaced by zeros. Based on the pivot-table, the cosine score assesses with scikit-learn's *cosine_similarity* function[7]. After the data preprocessing phase, the *cosine_similarity* function can be used to determine the similarity between users based on the rate which they give to different books and books can be recommended to the users based on these similarities. We do not give equal weight to all users, so we prefer users with ratings that are similar to the user in question than the other users whose ratings are not.

$$r_{u,m} = \frac{\sum_{u',u'\neq u} sim(u,u') \cdot r_{u',m}}{\sum_{u',u'\neq u} |sim(u,u')|}$$

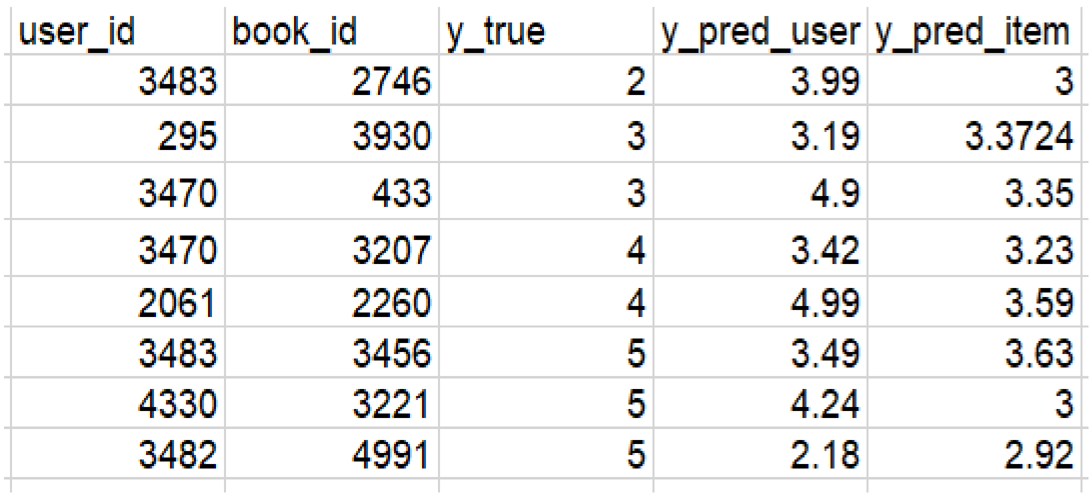
In this formula, $r_{u,m}$ represents the rating given by user *u* to book *m*. When we split the data into two parts as train and test, we did stratify according to the **user_id**. Therefore, it is possible that some books do not have in similarity matrix, and we need to check this case. If the book is not in the matrix, we define a rating of 3, regardless of the user and the book. Otherwise, we calculate the weight mean score. Implementation of the item-based approach is very close to the implementation of the user-based approach.

Item-based approach

In this approach, we considered the similarities between the books, in the previous approach this similarity was based on users. All stages in the user-based approach are repeated. In this section, we talked about the differences between the implementation of the two approaches. Firstly, when the data is split into the train and test sets, the data is stratified according to the book. For this reason, we drop books that are repeated only one time in the dataset for stratify strategy. Evaluate similarities between items, we use the pivot table as before. The only difference is that the index and column change positions. After filling missing values with zeros, we found similarities of books with cosine metrics. Again, we used the same strategies as in the user-based approach. Then we predict user ratings based on books, not users.

3.2.2 Results

Our dataset contains ratings that users give to books. We predict user ratings using the weight mean formula and calculate RMSE with real value of rating and predicted value of rating. For our user-based model, RMSE value is 2.0011 and for the item-based model, this value is 1.9596. For this part, we randomly chose 8 books ("Mexican-American War", "Sex, Sorcery, and Spirit: The Secrets of Erotic Magic", "The Best American Essays of the Century", "Texas Birds: A Folding Pocket Guide to Familiar Species", "Greyhawk Adventures: From the Ashes/Boxed Set", "No Trespassing!: Squatting, Rent Strikes, and Land Struggles Worldwide", "Investing in Duplexes, Triplexes, and Quads", "EMS Field Guide, BLS Version") from the test dataset and we took the *id* of the user and book from the test dataset and called them id pairs (**id_pairs**). The first value in the id pairs is user id (**user_id**). For instance, according to the (3483, 2746) id pair, the real value of rating (y_true) is 2, which means that 3483rd user gave 2 ratings to the 2746th book. In the user-based model predicted value of rating (y_pred_user) is 3.99, and in the item-based model predicted value of rating (y_pred_item) is 3. In some cases, we can say that the difference between the prediction value and the real value is not so great, but in some cases the difference is large. The item-based model is a little bit better than the user-based approach. The results of the experimental part are shown in Table 1 and visualized in Figure 3.



user_id	book_id	y_true	y_pred_user	y_pred_item
3483	2746	2	3.99	3
295	3930	3	3.19	3.3724
3470	433	3	4.9	3.35
3470	3207	4	3.42	3.23
2061	2260	4	4.99	3.59
3483	3456	5	3.49	3.63
4330	3221	5	4.24	3
3482	4991	5	2.18	2.92

Table 1. Results of the experimental part

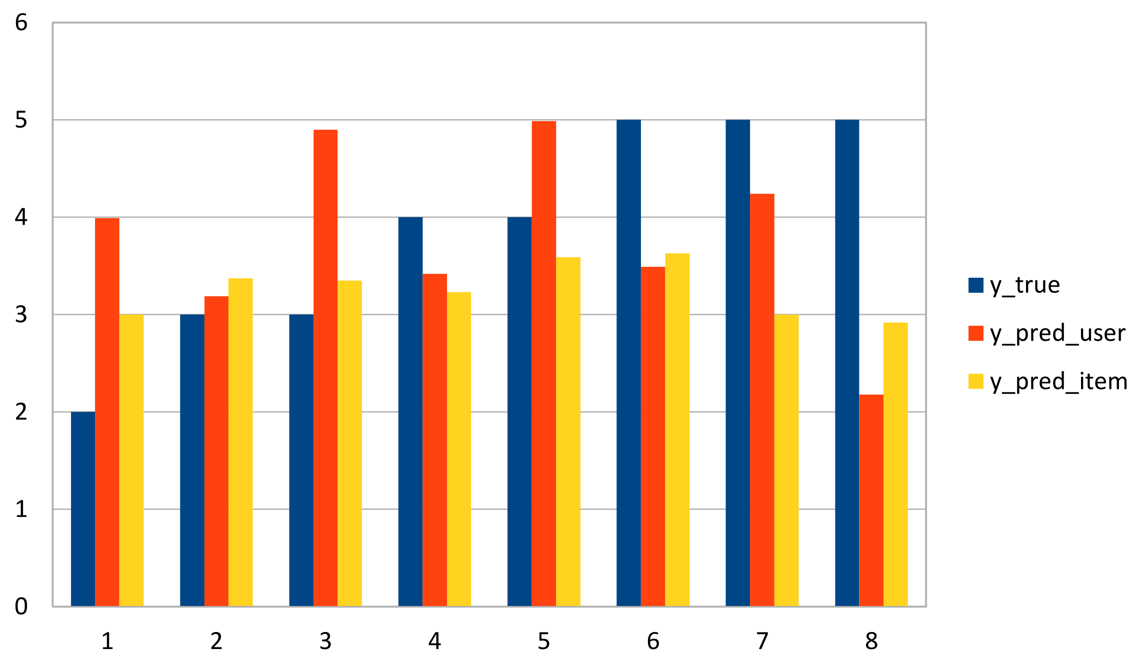


Figure 3. Visualizations of the results

In **Figure 3**, blue indicates the value of *y_true*, orange the value of *y_pred_user*, and yellow the value of *y_pred_item*. It can be seen from the chart and the table, in some cases the differences between real value and predicted value are similar but most of the time differences are big. According to **Figure 3** predicted value in the item-based model is more similar to the real value. If we compare user-based and item-based approaches, we can see that the RMSE value is a bit greater in the user-based approach. Finally, we calculated the mean of predicted values for both models based on ratings rated by users as 5. This value was 2.71 for the user-based model and it was 2.44 for the item-based model. We can say that the user-based model is a little bit better for the recommendation engine in this case. This case also is repeated for other scales of ratings where the real value of ratings are 4, 3, 2, and 1.

3.3 Hybrid Approach

3.3.1 Applied Approaches

Initially, the datasets that we used in this approach were: *description.csv*, *listing.csv*, and *books.csv*. We used descriptions of each book from the *description.csv*, the meta-data of each book from *listing.csv*, and the ratings of each user for each book from the *books.csv*. We built a new system by using the previous strategies that we used in CB and CF approaches for this Hybrid approach. As mentioned, the Hybrid approach is a combination of several approaches and there are several ways to combine these approaches to achieve efficient Hybrid RS. The integration of the CF approach into the CB filtering approach can be an example of one of these ways and we have established our Hybrid RS using this example. The workflow of this system is as follows:

i. We took into consideration the name of the book and the user as input.
ii. We used the CB approach to find the most similar 20 books according to input. In this case, as mentioned before, we determine the similarities based on the descriptions and some metadata information like genres and authors of books.
iii. With the helping of the user-based filtering of the CF approach, we estimated the ratings that users can give to these books.
iv. In the end, we got the top 5 results that provide the books with the highest predicted ratings and recommended these books to the user.

We considered all results from the implementation of Hybrid approache in the next section.

3.3.2 Results

We conducted some experiments to measure the quality of the Hybrid system that we applied. The testing part is especially important to measure the quality of such systems based on accurate results. There are two main inputs that we will consider in this system, first the name of the book and then the user's ID number. In the case of this experiment, we take into consideration the name of the book as **“Fantasy League”** and the user's ID number as **3466**. Having obtained the most similar results to the book we assigned, we predicted the ratings that these books will be given by user **3466**. At last, we got the highest results on the basis of predictions. Based on the results, we observed that books read and rated by the user can also be found in the list of recommendations. Thus, we added the restriction that if there are any books that users read and rated, remove them from the recommendations list. If we take a quick look at the results of CB approach, we can see that we mostly get the results based on metadata. For the book “Fantasy League” in the genre of “Children's Books”, in contrast to the CB approach, in this system only two books in the same genre were recommended to the user. Books in other genres were recommended both based on the CB approach and the approximate ratings which we predicted with the using of CF approach that the user can give. The case of taking into consideration several approaches together is one of the robust sides of our Hybrid RS. As we said, we built the Hybrid system using the CF approach integration into the CB filtering approach. This case forced us to work with more limited data in this system. We came to the conclusion that we need larger datasets to get more efficient results from our Hybrid RS.

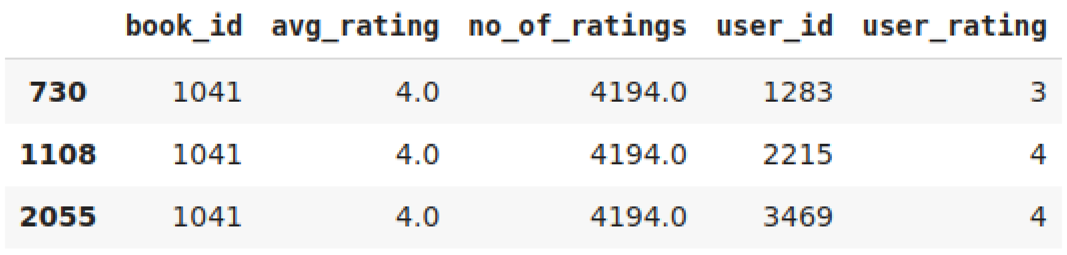
Discussion and Perspective

Generally, our implementation worked as we want. We would say that Content-Based Filtering worked better than we expected but it isn’t the case for Collaborative Filtering. Collaborative filtering sometimes predicted poor results. When we searched on the internet, we found that the weighted mean approach is one of the best approaches for CF but in practice, it wouldn’t be so for us. There could be different reasons for such behavior of CF. We would say that may be the main problem is related to the dataset itself. Maybe the *rating data matrix* was very sparse and because of that, we had poor predictions sometimes. Also for CB, we had only 122 examples which are low. However, we didn’t have poor recommendations but we thought that if we have a good dataset it would be much better for us.

After this, we think about improving our project. We think that to improve this project initially, we should start with another powerful, good dataset. We also think that it may be better to add a sentiment analysis feature to our project in the CB part. If we will work on Amazon’s Book Review dataset then the sentiment analysis approach can be a nice idea. For CF, we didn’t apply supervised machine learning methods but it may be better to add such an approach to our project. In our project, we also applied a sequential data mining algorithm but unfortunately, it wouldn’t be successful. We think that the main reason for that failure is our dataset. It isn’t a transactional dataset and our assumptions didn’t work as we expected. If we had a nice dataset, we are sure that we can mine useful patterns from that dataset. We think that all things that we mentioned now can be very useful to improve our project.

Appendix A

Additionally, we applied a data mining approach to our problem. After a long discussion among ourselves, we decided that maybe it can be useful to apply sequential data mining algorithms to our dataset. In this case, we had a problem that the dataset didn't have a column like a timestamp. After analyzing our dataset, we thought that it may be possible to take the order of index of dataset as a sequence of events. Let's put an example:



	book_id	avg_rating	no_of_ratings	user_id	user_rating
730	1041	4.0	4194.0	1283	3
1108	1041	4.0	4194.0	2215	4
2055	1041	4.0	4194.0	3469	4

Above figure shows which users rated the book with id 1041. Here we have two assumptions:

The first assumption is that: "If a user gave a rating to some book then it means that the user bought that book in the past".

As you see, there are three index numbers 730, 1108, and 2055 for this dataset.

The second assumption is that: "The first user with id 1283 bought this book and after that, the user with id 2215 bought this book and again after that the user with id 3469 bought this book".

Therefore, we think the order of indexes as the order of transactions. We could continue our work only after having such assumptions.

We applied **GSP** sequential data mining algorithm to our dataset. We would say that the result is not satisfactory from our point of view, because there are seven lengths 2 sequences and only one length 3 sequences for minimum support 4. At that time, we think that we wouldn't be successful in our approach. Also, minimum support 4 is much less for our mining algorithm, and the patterns also very poor. Maybe we have problems with our assumptions. For this case, we think that the main reason is our dataset.

References

[1] Laurent Candillier, Kris Jack, Françoise Fessant, and Frank Meyer. State-of-the-art recommender systems, Collaborative and Social Information Retrieval and Access-Techniques for Improved User Modeling, 2009.

[2] Charu C. Aggarwal, Recommender Systems: The Textbook, IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, 2016.

[3] Kemal Ozkan, Zuhal Kurt, Erol Seke, An Image-based Recommender System Based on Image Annotation, CNR Group, Istanbul, Turkey, 2019.

[4] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. 6, 4, Article 13 (December 2015), 19 pages.

[5] Greg Linden, Brent Smith, and Jeremy York, Amazon.com Recommendations: Item-to-Item Collaborative Filtering, IEEE INTERNET COMPUTING, 2003.

[6] Zafar Ali, Shah Khusro, Irfan Ullah, A Hybrid Book Recommender System Based on Table of Contents (ToC) and Association Rule Mining, INFOS '16, Giza, Egypt, 2016.

[7] Rounak Banik, Hands-On Recommendation Systems with Python: Start building powerful and personalized, recommendation engines with Python, Birmingham, UK, 2018.

[8] https://www.kdnuggets.com/2020/08/content-based-recommendation-system-word-embeddings.html

[9] Fidel Cacheda, Víctor Carneiro, Víctor Carneiro, Vreixo Formoso, Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems, University of A Coruña, Spain, 2011.

[10] https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

[11] Laurent Candillier, Kris Jack, Françoise Fessant, Frank Meyer, State-of-the-Art Recommender Systems, France Telecom R&D Lannion, France, 2009.

[12] Manisha Chandaka, Sheetal Giraseb, Debajyoti Mukhopadhyay, Introducing Hybrid Technique for Optimization of Book Recommender System, Department of IT, Maharashtra Institute of Technology, Kothrud, India, 2015.

[13] http://jalammar.github.io/illustrated-word2vec/

[14] https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1

[15] https://thedatasingh.medium.com/introduction-713b3d976323

[16] https://www.machinelearningplus.com/nlp/cosine-similarity/