Books Recommendation System Using Collaborative Filtering

Nour Saleh Rabee'
Bachelor of Computer Engineering
Birzeit, Ramallah, Palestine
rabee.nour123@gmail.com

Najwa Nidal Bsharat
Bachelor of Computer Engineering
Birzeit, Ramallah, Palestine
najwabsharat112@gmail.com

Lojain Jalal Abdalrazaq
Bachelor of Computer Engineering
Birzeit, Ramallah, Palestine
lojain.jalal@gmail.com

Abstract— **In today's digital world, where new books are constantly being published on various topics, finding a book that matches your interests can be a challenging and time-consuming task. Search engines and online bookstores often fall short in providing accurate recommendations that precisely fit your preferences or highlight highly recommended books.**

To address this issue, our project focuses on developing a recommender system that suggests books to users based on their preferences. By incorporating collaborative filtering techniques, we analyze user ratings and interactions with books to identify patterns and similarities between users. This enables us to recommend books to users based on the preferences and behaviors of others with similar tastes. This collaborative filtering approach enhances the accuracy and relevance of the book recommendations, helping users discover new books that align with their interests and preferences.

Keywords—Collaborative Filtering, Recommendation System, NLP, Cosine Similarity, Recall, Precision, PR Curve.

I. Introduction

There are three main types of recommendation algorithms used in recommendation systems. Collaborative Filtering, Content-based Filtering, and hybrid of them. Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user.[1] Content-based filtering, on the other hand, uses similarities in products, services, or content features, as well as information accumulated about the user to make recommendations.[2]

Hybrid recommender system is the one that combines multiple recommendation techniques together to produce the output. If one compares hybrid recommender systems with collaborative or content-based systems, the recommendation accuracy is usually higher in hybrid systems. The reason is the lack of information about the domain dependencies in collaborative filtering, and about the people's preferences in content-based system. The combination of both leads to common knowledge increase, which contributes to better recommendations. The knowledge increase makes it especially promising to explore new ways to extend underlying collaborative filtering algorithms with content data and content-based algorithms with the user behavior data.

So, why is collaborative filtering chosen as a method for this project's recommendation system? Collaborative filtering is chosen for several reasons. Firstly, collaborative filtering is content-independent, meaning it doesn't rely on specific item attributes or content. Instead, it focuses on user ratings and preferences, allowing for a more comprehensive and unbiased assessment of item quality. Additionally, collaborative filtering enables effective recommendations by leveraging the concept of user similarity. By identifying users with similar tastes and preferences, the system can suggest items that have been positively rated by those similar users. This user-centric approach enhances the relevance and accuracy of the recommendations. [3]

Considering these advantages, collaborative filtering is deemed the most suitable approach for our project's recommendation system. It not only allows for a robust assessment of item quality but also ensures that recommendations are tailored to individual user preferences, ultimately enhancing the overall user experience.

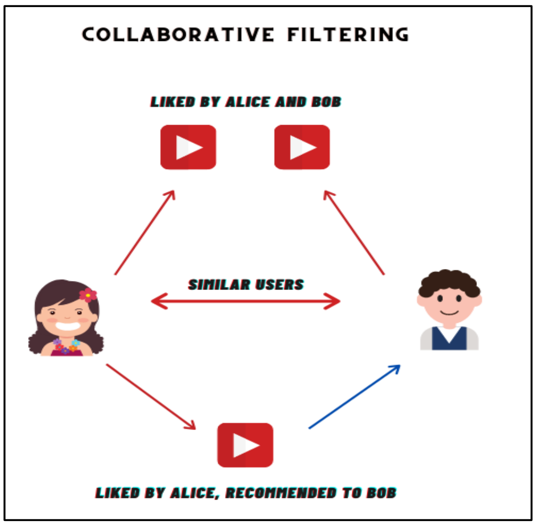


Fig 1. Collaborative Filtering

A. *Similarity Calculation*

Within collaborative filtering, there are specific techniques employed, and one notable example is ***Cosine Similarity***. Cosine similarity is the measurement of how similar two products or two documents are irrespective of their shape or size. It is the dot product of the two vectors divided by the two vectors' magnitude. Cosine similarity is expressed by the below equation:

$$\text{Similarity}(p,q) = \cos\theta = \frac{p \cdot q}{\|p\|\|q\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}$$

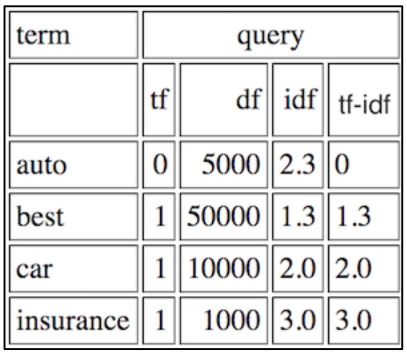
Fig 2. Cosine Similarity Equation [4]

The value of cosine varies from -1 to +1 where the -1 value refers to items that are opposite in nature and +1 refers to the item which are very similar. [4]

B. *Data Preprocessing*

Text is mostly in unstructured form. Lot of noises will be present in it. In data preprocessing we will remove the noises associated with it. It is not possible to analyze the data without properly preprocessing it.[5]The project incorporated various data processing techniques, including the utilization of ***TF-IDF*** and ***sparse matrix***.

TF-IDF, short for Term Frequency-Inverse Document Frequency, is a handy algorithm that uses the frequency of words to determine how relevant those words are to a given document.[6] It takes into account both the term's frequency within a document (TF) and its rarity across the entire document collection (IDF). TF-IDF is commonly used in information retrieval and text mining tasks to determine the importance of terms and aid in document ranking or similarity calculations.



term	query			
	tf	df	idf	tf-idf
auto	0	5000	2.3	0
best	1	50000	1.3	1.3
car	1	10000	2.0	2.0
insurance	1	1000	3.0	3.0

Fig 3. TF-IDF Matrix

A sparse matrix, on the other hand, is a data structure that efficiently represents matrices where the majority of elements are zero. In contrast to a dense matrix, which stores all elements explicitly, a sparse matrix only stores the non-zero elements, resulting in reduced memory usage and computational efficiency. Sparse matrices are commonly employed in various data processing tasks, including collaborative filtering, where the user-item rating matrix often exhibits sparsity due to the large number of possible items and users.

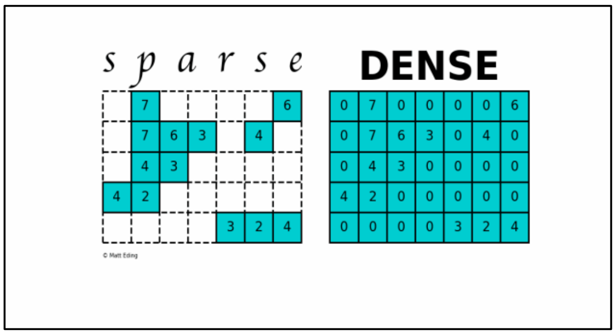


Fig 4. Sparse Matrix Vs. Dense Matrix

However, prior to analysis, the parsed data underwent crucial preprocessing steps. This involved converting the text to lower case to standardize it and treat words uniformly. Additionally, non-alphanumeric characters were removed to focus on meaningful content and eliminate noise. Furthermore, whitespace removal ensured a compact representation of the text. These preprocessing steps played a vital role in cleaning the data, reducing noise, and preparing it for effective analysis, ultimately enhancing the performance of the recommendation system.

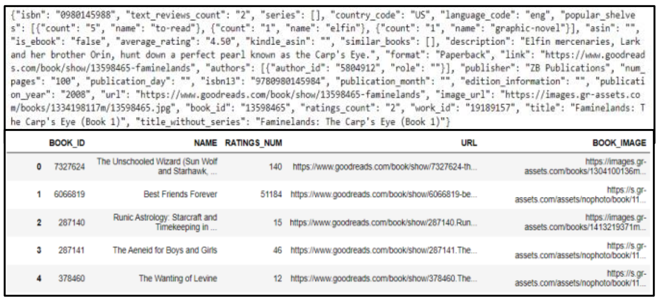
II. Material and Methods

A. *METADATA USED*

The book recommender system uses information from Goodreads, a popular website where people share their reading activities, to make better recommendations. Researchers from the University of California San Diego collected and organized the data from Goodreads. They used four main files for the recommendation system. The first file, "goodreads_interactions.csv," contains details about how users interacted with books, such as user IDs, book IDs, and ratings from 0 to 5. The second file, "goodreads_books.json," provides more information about each book. Each line in the file represents a different book. The last dataset, "book_id_map.csv," helps keep the book ID references consistent across the different datasets. By using all of this data, the recommender system can make better recommendations by considering how users interacted with books in the past.

B. *DATA ANALYSIS AND PREPROCESSING*

1) Parsing Book Metadata: In the first step of the project, we focus on organizing the book information from the "goodreads_books.json" file. We look for specific details that are important for the recommender system. This helps us create a structured format for the data. The main fields we are interested in are the book's unique identifier (BOOK_ID), title (NAME), number of ratings it has received (RATINGS_NUM), the web address that provides more information about the book (URL), and the web address of the book's cover image (BOOK_IMAGE). By extracting only these important details, we make the book information more concise and easier to work with.



```
{"isbn": "0980145988", "text_reviews_count": "2", "series": [], "country_code": "US", "language_code": "eng", "popular_shelve
s": [{"count": "5", "name": "to-read"}, {"count": "1", "name": "elfin"}, {"count": "1", "name": "graphic-novel"}], "asin": "",
"is_ebook": "false", "average_rating": "4.50", "kindle_asin": "", "similar_books": [], "description": "Elfin mercenaries, Lark
and her brother Orin, hunt down a perfect pearl known as the Carp's Eye.", "format": "Paperback", "link": "https://www.goodread
s.com/book/show/13598465-faminelands", "authors": [{"author_id": "5804912", "role": ""}], "publisher": "ZB Publications", "num_
pages": "100", "publication_day": "", "isbn13": "9780980145984", "publication_month": "", "edition_information": "", "publicati
on_year": "2008", "url": "https://www.goodreads.com/book/show/13598465-faminelands", "image_url": "https://images.gr-assets.co
m/books/1334198117m/13598465.jpg", "book_id": "13598465", "ratings_count": "2", "work_id": "19189157", "title": "Faminelands: T
he Carp's Eye (Book 1)", "title_without_series": "Faminelands: The Carp's Eye (Book 1)"}
```

	BOOK_ID	NAME	RATINGS_NUM	URL	BOOK_IMAGE
0	7327624	The Unschooled Wizard (Sun Wolf and Starhawk, ...	140	https://www.goodreads.com/book/show/7327624-th...	https://images.gr-assets.com/books/1304100136m...
1	6066819	Best Friends Forever	51184	https://www.goodreads.com/book/show/6066819-be...	https://s.gr-assets.com/assets/nophoto/book/11...
2	287140	Runic Astrology: Starcraft and Timekeeping in ...	15	https://www.goodreads.com/book/show/287140.Run...	https://images.gr-assets.com/books/1413219371m...
3	287141	The Aeneid for Boys and Girls	46	https://www.goodreads.com/book/show/287141.The...	https://s.gr-assets.com/assets/nophoto/book/11...
4	378460	The Wanting of Levine	12	https://www.goodreads.com/book/show/378460.The...	https://s.gr-assets.com/assets/nophoto/book/11...

Fig 5. Data Parsing: Extracting Book Metadata from Goodreads

2) Title Processing and Cleaning: After extracting the necessary information from the "goodreads_books.json" file, we move on to the next step of creating a search engine to find books based on their titles. To make this process easier, we perform some steps on the book titles:

a) *Cleaning the text:* We remove any characters that are not letters or numbers, including common words like "the" or "and".
b) *Lowercasing:* We convert all the letters in the titls to lowercase.
c) *Removing duplicated spaces*: If there are mutiple spaces in a row, we reduce them to just one space.
d) *Considering titles with length greater than 1*: We only include titles that have more than one character.

For example, the book title "The Golden Compass(His Dark Materials,#1)" goes through these steps and becomes "the golden compass his dark materials 1". This modified title is easier to compare and match with other titles, as it removes special characters and follows a standard format. Finally, we save the processed book data, including the extracted information and adjusted titles, in a JSON file named "book_titles.json" for future use.

C. Book Search Engine

The goal of creating a book search engine is to find books in the "Processed_book_metadata.json" file that we have processed earlier, based on books we have read and liked. This allows us to retrieve additional information like the book's unique identifier (BOOK_ID), number of ratings it has received (RATINGS_NUM), its web address (URL), and the cover image (BOOK_IMAGE). Since we have a large dataset with millions of books, a search engine is necessary. The search engine works by taking an input query, which is the name of a book, and then finding the closest matches among the books in our dataset.

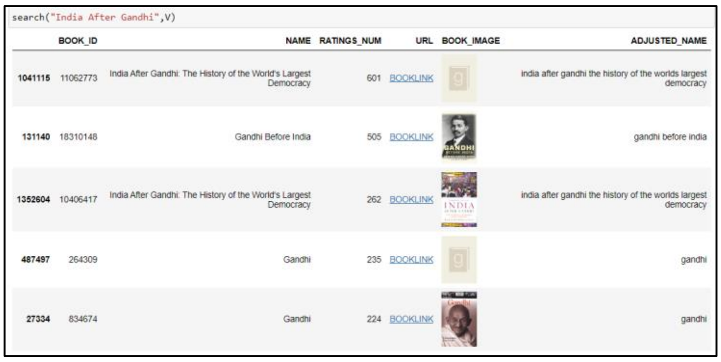
There are a few approaches used in the search engine:

*a) TI*IDF matrix:* The book titles in our dataset are converted into numbers, and the search engine matches the query to a list of these numbers.

b) Cosine similarity approach: It calculates the similarity between the input query vector and the tf-idf matrix.

To perform these calculations, we use helpful libraries such as scikit-learn. Specifically, we utilize the TfidfVectorizer class from the sklearn.feature_extraction.text module to transform the processed book titles into a tf-idf matrix representation. This matrix allows for efficient searching and matching of book titles. Additionally, the cosine_similarity function from the sklearn.metrics.pairwise module is used to calculate the cosine similarity scores, which help determine the most relevant book matches.

In the end, the search engine takes an input query with a book title and provides a list of the most matched books ranked based on the number of ratings they have received.



```
search("India After Gandhi",V)
```

	BOOK_ID	NAME	RATINGS_NUM	URL	BOOK_IMAGE	ADJUSTED_NAME
1041115	11062773	India After Gandhi: The History of the World's Largest Democracy	601	BOOKLINK		india after gandhi the history of the worlds largest democracy
131140	18310148	Gandhi Before India	505	BOOKLINK		gandhi before india
1352604	10406417	India After Gandhi: The History of the World's Largest Democracy	262	BOOKLINK		india after gandhi the history of the worlds largest democracy
487497	264309	Gandhi	235	BOOKLINK		gandhi
27334	834674	Gandhi	224	BOOKLINK		gandhi

Fig 6. Relevant Book Matches for the Search 'India after Gandhi'

D. FINDING SIMILAR USERS

In the process of finding similar users with the same taste in books, several steps are followed.

A) *Mapping Book ID Values:* After using the book search engine to create a personal dataset of the books that have been read and liked, the book IDs in the "liked_books.csv" file are mapped to the corresponding actual book IDs using the "book_id_map.csv" file. This mapping allows for accurate identification of the liked books.

b) *Finding similar users:* The next step involves reading the user data from the input file "goodreads_interactions.csv" and identifying users who share similar tastes in books. This is achieved by checking if there are any mutual books between each user and the target user. If a mutual book is found, the user is added to a dictionary, and if the user is already present, the count of mutual books is incremented. This process results in determining the number of users who have at least one book overlap with the target user.

c) *Filtering Users:* The filtration process is implemented to refine the list of similar users. Users who have a specific amount of book overlap, such as at least 20% of similar books, are retained. This filtering step ensures that the final list consists of users who exhibit a significant similarity in reading preferences.

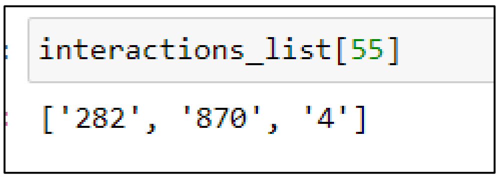
The output of this process is the obtained list of users who share a similar taste in books. The total number of similar users after the filtration step is **1258 users**, out of the initial **316341 users**. This list of similar users forms the basis for further analysis and personalized book recommendations in the collaborative filtering approach.

E. *SIMILAR USERS RATINGS*

Since it's not enough to find the similar users that have read mutual books, it is important to have a similar ratings values to these books. So we have built matrix that have the user_id book_id, ratings attributes for each similar user:

User_id	Book_id	ratings

Using "goodreads_interactions.csv" the history of the overlapped users that have the similar taste such as us was uploaded. The following figure shows the interaction for the user at index 55:

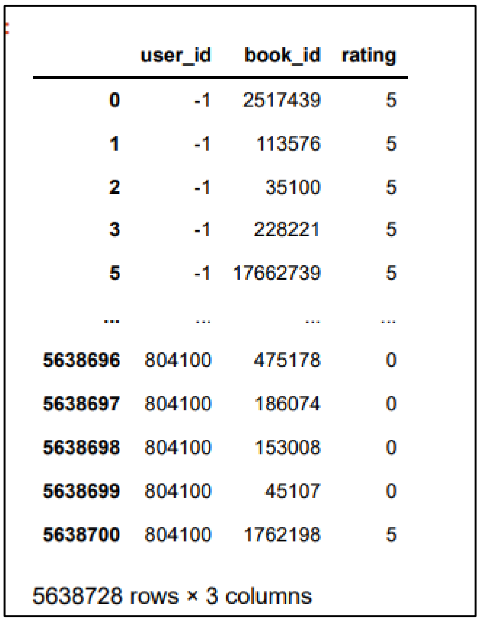


```
interactions_list[55]
['282', '870', '4']
```

Fig 7. the interaction for the user at index 55

F. *CREATING BOOK/USER MATRIX*

We will construct a user-book matrix, where each row represents a different user and each column represents a different book. The cells in the matrix will contain the ratings given by users to the corresponding books. As the number of users and books can be large, the matrix can quickly become extensive. To address this, we filtered down our user list in a previous step to manage the matrix size.



```
         user_id   book_id  rating
0             -1   2517439       5
1             -1    113576       5
2             -1     35100       5
3             -1    228221       5
5             -1  17662739       5
...          ...       ...     ...
5638696   804100    475178       0
5638697   804100    186074       0
5638698   804100    153008       0
5638699   804100     45107       0
5638700   804100   1762198       5

5638728 rows × 3 columns
```

Fig 8. Book/USER Matrix

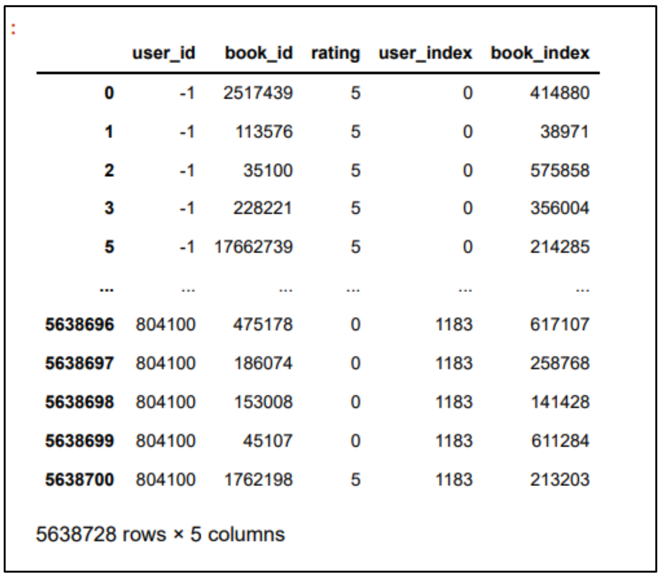
From the figure above, we can observe that our user ID is (-1). The initial rows in the matrix contain our own ratings, which are ratings from other users.

Next, we need to convert the user and book IDs to corresponding positions. This means we assign a specific position to each unique user ID and book ID as shown in the table below.

		Book position		
user position	-	**0**	**1**	**2**
	0	5		2
	1	6	3	
	2			2

Fig 9. Mapping each user index with user id and each book index with book id.

When constructing our matrix which is a concatenating of figure 8 and Figure 9, we utilize a sparse matrix. The reason for using a sparse matrix is that it doesn't consume excessive memory or storage space.



```
         user_id   book_id  rating  user_index  book_index
0             -1   2517439       5           0      414880
1             -1    113576       5           0       38971
2             -1     35100       5           0      575858
3             -1    228221       5           0      356004
5             -1  17662739       5           0      214285
...          ...       ...     ...         ...         ...
5638696   804100    475178       0        1183      617107
5638697   804100    186074       0        1183      258768
5638698   804100    153008       0        1183      141428
5638699   804100     45107       0        1183      611284
5638700   804100   1762198       5        1183      213203

5638728 rows × 5 columns
```

Fig 10. Book/USER Matrix using Sparse Matrix

In Figure 10, we observe that the ratings in column 3 have values of either 5 or 0. This indicates that the ratings are essentially binary, representing a "good" or "bad" evaluation.

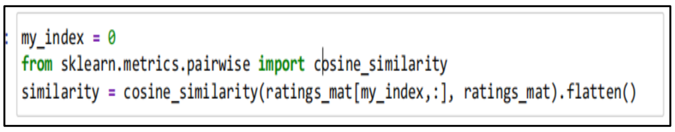
G. *Similar users to us*

To find users with similar tastes and preferences, we employed the cosine similarity method. This involves creating a matrix where rows represent users and columns represent books, capturing the user-book interactions or ratings.

We use cosine similarity to measure how similar users are based on their book preferences. This helps us identify users who have similar tastes. The similarity matrix we create has values between 0 and 1, where 1 indicates identical preferences. By comparing the book preferences of each pair of users, we generate this similarity matrix. We then find the most similar users to a target user by selecting the rows with the highest similarity values, considering users who have about 20% overlap in the books they have read. We obtain the user IDs of these similar users and analyze their book interactions or ratings. This allows us to generate a list of potential book recommendations specifically tailored for the target user. This approach taps into the collective knowledge of similar users to offer personalized suggestions, enhancing the user's experience and encouraging further exploration of books aligned with their interests.

The cosine similarity process calculates the similarity between each row (user) in the matrix. In this code, we specify the index as my user_id, which is -1, and calculate the similarity between my user and all other users in the matrix.

It is obvious from figure 10 that the user_inder to -1 is 0.

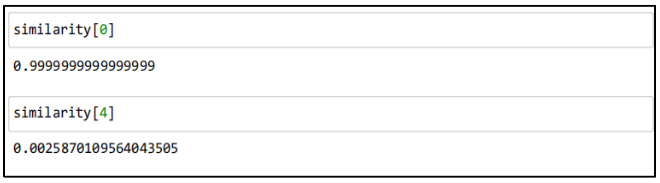


```
my_index = 0
from sklearn.metrics.pairwise import cosine_similarity
similarity = cosine_similarity(ratings_mat[my_index,:], ratings_mat).flatten()
```

Fig 11. Setting my_index to '0'

In the given figure, we calculate the cosine similarity between index 0 and itself, resulting in a similarity score of 0.999999999, indicating extremely high similarity. This suggests that the book preferences of the user at index 0 closely match their own preferences. The value is close to 1, indicating a strong alignment between the user's ratings and interactions with the books.

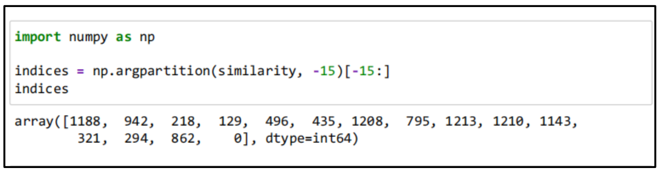
On the other hand, when we calculate the cosine similarity between user_index 0 and user_index 4, the resulting similarity score is 0.0025870109564043505. This indicates a very low similarity between these two users in terms of their book preferences. The value is significantly smaller, close to 0, suggesting that user 0 and user 4 have distinctly different tastes and preferences when it comes to the books they have rated or interacted with.



```
similarity[0]

0.9999999999999999

similarity[4]

0.0025870109564043505
```

Fig 12. Cosine Similarity between index 0 and itself and between 0 and 4.

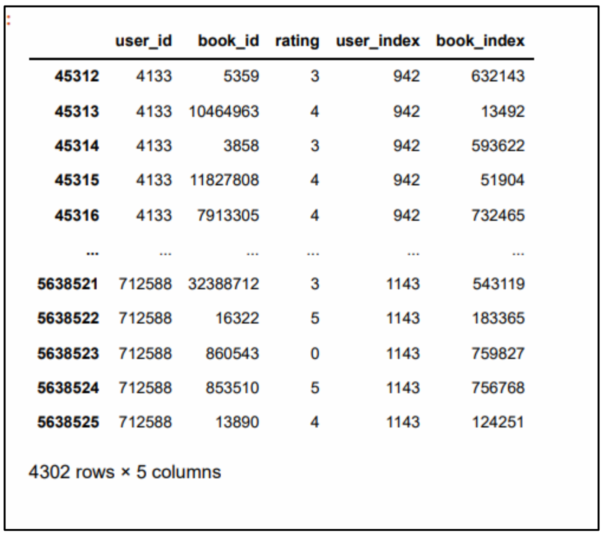
Our objective is to find the users who are most similar to our user with index -1. To accomplish this, we utilize the numpy function `np.argpartition` to identify the 15 users with the highest similarity scores. These users have the closest book preferences to our own user. Figure 13 displays the indices of 15 users that have most similar book taste to index 0.



```
import numpy as np

indices = np.argpartition(similarity, -15)[-15:]
indices

array([1188,  942,  218,  129,  496,  435, 1208,  795, 1213, 1210, 1143,
        321,  294,  862,    0], dtype=int64)
```

Fig 13. Top 15 Users with Similar Book Preferences to User at Index 0

Once we have obtained the indices of the top 15 users who share similar tastes with the user at index 0, we proceed to retrieve their corresponding user IDs. We then display this information in a table format, as depicted in Figure 13.



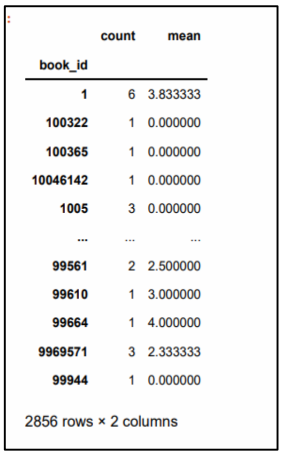
	user_id	book_id	rating	user_index	book_index
45312	4133	5359	3	942	632143
45313	4133	10464963	4	942	13492
45314	4133	3858	3	942	593622
45315	4133	11827808	4	942	51904
45316	4133	7913305	4	942	732465
...	...	...	...	...	...
5638521	712588	32388712	3	1143	543119
5638522	712588	16322	5	1143	183365
5638523	712588	860543	0	1143	759827
5638524	712588	853510	5	1143	756768
5638525	712588	13890	4	1143	124251

4302 rows × 5 columns

Fig 14. User IDs of Top 15 Users with Similar Book Preferences (User at Index 0)

H. Recommended Books

Once we have obtained the user IDs of similar users, we can filter their interactions or ratings to identify the books they have enjoyed. From this, can create a list of potential book recommendations for the target user. To form the recommendation list, we consider the frequency of each book's appearance in these recommendations (count) and the average rating (mean) given to it. For example, if all similar users have read a book and rated it poorly (e.g., rating of 1), that book will not be recommended.

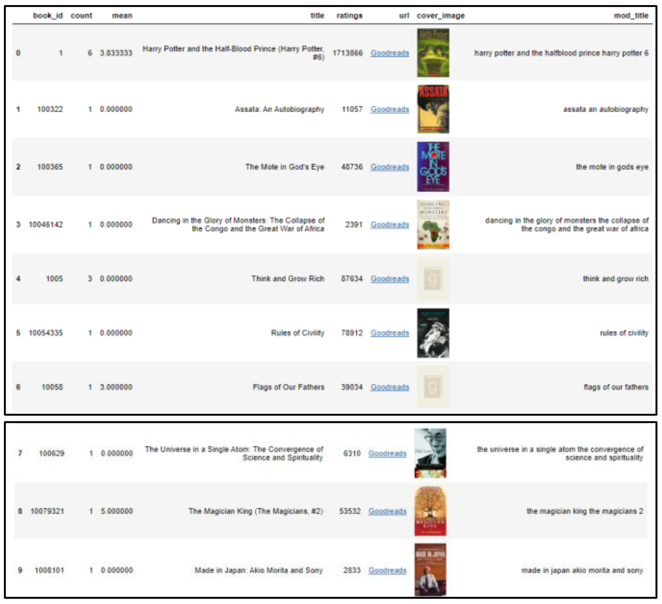


book_id	count	mean
1	6	3.833333
100322	1	0.000000
100365	1	0.000000
10046142	1	0.000000
1005	3	0.000000
...	...	...
99561	2	2.500000
99610	1	3.000000
99664	1	4.000000
9969571	3	2.333333
99944	1	0.000000

2856 rows × 2 columns

Fig 15. The count and The average rating for each book

In the next step, we enhance the usefulness of the recommendations by adding the book titles. This involves merging the book titles, url, cover image, and the modified title, with the previously generated list of recommended books (Figure 15). By doing so, we obtain the names or titles of specific books, providing more comprehensive information to the user.



	book_id	count	mean	title	ratings	url	cover_image	mod_title
0	1	6	3.833333	Harry Potter and the Half-Blood Prince (Harry Potter, #6)	1713866	Goodreads		harry potter and the halfblood prince harry potter 6
1	100322	1	0.000000	Assata: An Autobiography	11057	Goodreads		assata an autobiography
2	100365	1	0.000000	The Mote in God's Eye	48736	Goodreads		the mote in gods eye
3	10046142	1	0.000000	Dancing in the Glory of Monsters: The Collapse of the Congo and the Great War of Africa	2391	Goodreads		dancing in the glory of monsters the collapse of the congo and the great war of africa
4	1005	3	0.000000	Think and Grow Rich	87634	Goodreads		think and grow rich
5	10054335	1	0.000000	Rules of Civility	78912	Goodreads		rules of civility
6	10058	1	3.000000	Flags of Our Fathers	39034	Goodreads		flags of our fathers
7	100629	1	0.000000	The Universe in a Single Atom: The Convergence of Science and Spirituality	6310	Goodreads		the universe in a single atom the convergence of science and spirituality
8	10079321	1	5.000000	The Magician King (The Magicians, #2)	53532	Goodreads		the magician king the magicians 2
9	1008101	1	0.000000	Made in Japan: Akio Morita and Sony	2833	Goodreads		made in japan akio morita and sony

Fig 16. Merging Book Titles, URLs, and Cover Images for Enhanced Suggestions

By incorporating the book titles into the recommendation system, we enhance its user-friendliness and overall effectiveness. This improvement allows users to easily identify and explore specific books of interest. The inclusion of book titles adds valuable context and improves the user experience, making the recommendation system more intuitive and user-friendly.

I. *Ranking Books*

After obtaining 2854 recommendations, we focus on determining the most relevant ones. To achieve this, we begin by creating an adjusted account that normalizes the book's appearances among users similar to us compared to a broader audience. This accounts for the popularity of a book within our specific preferences.

Next, we identify the popularity of each book by examining its frequency across all users. We prioritize books that are popular among users like us but not as popular among users who have different preferences.

We proceed with two filtering steps:

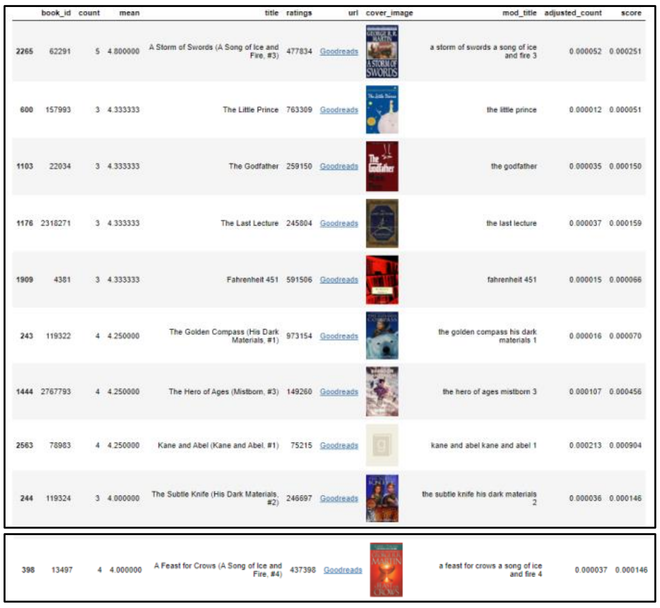
a) Removing books that we have already read by comparing their book IDs with our reading history.

b) Eliminating books with titles that match the titles of books we have read, ensuring we exclude duplicates or variations. We ensure the titles are in a standardized format for accurate comparison.

Additionally, we remove books with low popularity, specifically those recommended by two or fewer users, to prioritize books with a higher number of recommendations and readers.

To determine the ranking of recommendations, we calculate the mean reading score for each book, which represents the average rating from users like us (index 0). This score is multiplied by the book's popularity, combining user ratings and popularity in the ranking process.

Finally, we sort the recommendations based on their adjusted score (mean reading score multiplied by popularity) in descending order. This results in a ranked list of books, with the most relevant recommendations appearing at the top.



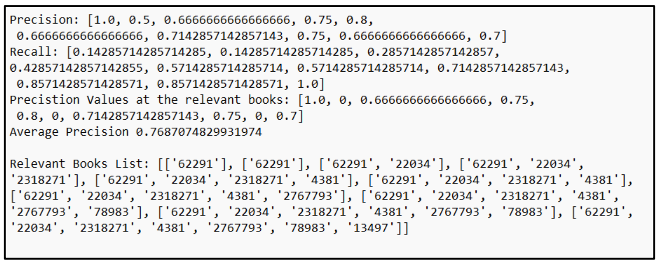
	book_id	count	mean	title	ratings	url	cover_image	mod_title	adjusted_count	score
2265	62291	5	4.800000	A Storm of Swords (A Song of Ice and Fire, #3)	477834	Goodreads		a storm of swords a song of ice and fire 3	0.000052	0.000251
600	157993	3	4.333333	The Little Prince	763309	Goodreads		the little prince	0.000012	0.000051
1103	22034	3	4.333333	The Godfather	259150	Goodreads		the godfather	0.000035	0.000150
1176	2318271	3	4.333333	The Last Lecture	245804	Goodreads		the last lecture	0.000037	0.000159
1909	4381	3	4.333333	Fahrenheit 451	591506	Goodreads		fahrenheit 451	0.000015	0.000066
243	119322	4	4.250000	The Golden Compass (His Dark Materials, #1)	973154	Goodreads		the golden compass his dark materials 1	0.000016	0.000070
1444	2767793	4	4.250000	The Hero of Ages (Mistborn, #3)	149260	Goodreads		the hero of ages mistborn 3	0.000107	0.000456
2563	78983	4	4.250000	Kane and Abel (Kane and Abel, #1)	75215	Goodreads		kane and abel kane and abel 1	0.000213	0.000904
244	119324	3	4.000000	The Subtle Knife (His Dark Materials, #2)	246697	Goodreads		the subtle knife his dark materials 2	0.000036	0.000146
398	13497	4	4.000000	A Feast for Crows (A Song of Ice and Fire, #4)	437398	Goodreads		a feast for crows a song of ice and fire 4	0.000037	0.000146

Fig 17. Books after ranking

III. Evaluations

We evaluate the performance of the recommendation system by assessing key metrics such as precision, recall, and average precision. These metrics provide valuable insights into the system's effectiveness in generating relevant recommendations. Precision measures the accuracy of the recommended books, recall captures the system's ability to identify all relevant books, and average precision accounts for both precision and the order in which relevant books are presented. By analyzing these evaluation metrics, we gain a comprehensive understanding of the recommendation system's quality and its ability to provide valuable recommendations to users.

These are the recommended books after ranking: ['62291', '157993', '22034', '2318271', '4381', '119322', '2767793', '78983', '119324', '13497'] (figure 17). The binary relevance of these books, indicating whether they are relevant or not, is given as [1, 0, 1, 1, 1, 0, 1, 1, 0, 1]. The binary relevance was calculated based on user feedback. So the results were as follows:



```
Precision: [1.0, 0.5, 0.6666666666666666, 0.75, 0.8,
 0.6666666666666666, 0.7142857142857143, 0.75, 0.6666666666666666, 0.7]
Recall: [0.14285714285714285, 0.14285714285714285, 0.2857142857142857,
0.42857142857142855, 0.5714285714285714, 0.5714285714285714, 0.7142857142857143,
 0.8571428571428571, 0.8571428571428571, 1.0]
Precistion Values at the relevant books: [1.0, 0, 0.6666666666666666, 0.75,
 0.8, 0, 0.7142857142857143, 0.75, 0, 0.7]
Average Precision 0.7687074829931974

Relevant Books List: [['62291'], ['62291'], ['62291', '22034'], ['62291', '22034',
'2318271'], ['62291', '22034', '2318271', '4381'], ['62291', '22034', '2318271', '4381'],
['62291', '22034', '2318271', '4381', '2767793'], ['62291', '22034', '2318271', '4381',
'2767793', '78983'], ['62291', '22034', '2318271', '4381', '2767793', '78983'], ['62291',
'22034', '2318271', '4381', '2767793', '78983', '13497']]
```

Fig 18. Precision, Recall, Average precision after ranking books.

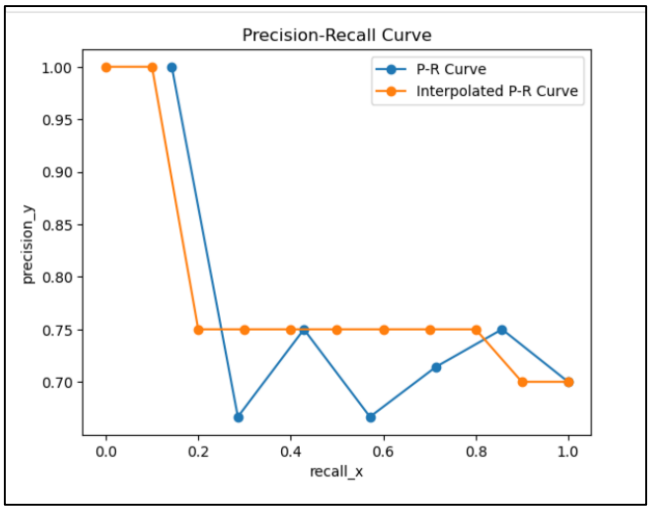
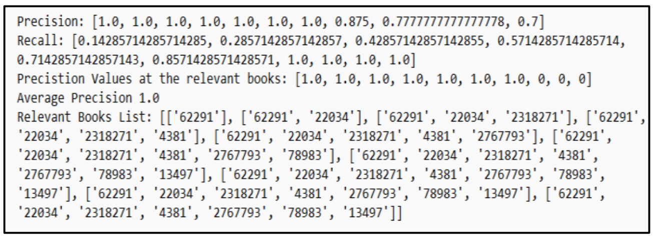


Fig 19. PR Curve before optimization.

After modifying the ranking of the recommended books, the binary relevance has been improved to match the desired sequence of [1, 1, 1, 1, 1, 1, 1, 0, 0, 0]. The new modified ranked recommended books are ['62291','22034', '2318271', '4381', '2767793', '78983', '13497', '157993', '119322', '119324']. This adjustment has resulted in improved precision, recall, and average precision scores, indicating a better alignment between the recommended books and the user's preferences or search criteria as shown in the figure below.



```
Precision: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.875, 0.7777777777777778, 0.7]
Recall: [0.14285714285714285, 0.2857142857142857, 0.42857142857142855, 0.5714285714285714,
0.7142857142857143, 0.8571428571428571, 1.0, 1.0, 1.0, 1.0]
Precistion Values at the relevant books: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0, 0, 0]
Average Precision 1.0
Relevant Books List: [['62291'], ['62291', '22034'], ['62291', '22034', '2318271'], ['62291',
'22034', '2318271', '4381'], ['62291', '22034', '2318271', '4381', '2767793'], ['62291',
'22034', '2318271', '4381', '2767793', '78983'], ['62291', '22034', '2318271', '4381',
'2767793', '78983', '13497'], ['62291', '22034', '2318271', '4381', '2767793', '78983',
'13497'], ['62291', '22034', '2318271', '4381', '2767793', '78983', '13497'], ['62291',
'22034', '2318271', '4381', '2767793', '78983', '13497']]
```

Fig 20. Precision, Recall, Average precision after modifying ranking books.

IV. Improvments

Suggestions on how to enhance the book recommendation system:

a) Implement hybrid approaches: Combine multiple recommendation techniques, such as collaborative filtering, content-based filtering, and demographic-based filtering, to create a hybrid recommendation system. This can provide a more comprehensive and accurate set of recommendations.

b) Consider contextual factors: Take into account factors such as the user's current mood, the time of year, or recent trends in the literary world. This can help provide more relevant and timely recommendations to users.

c) Regularly update the system: Keep the recommendation system up to date by adding new books, removing outdated ones, and adjusting the algorithms based on user feedback and evolving trends.

V. Conclusion

To sum up, in conclusion, our project aims to address the challenge of finding personalized book recommendations in today's vast digital landscape. By developing a recommender system based on collaborative filtering techniques, we have focused on analyzing user ratings and interactions to identify patterns and similarities among users. Through this process, we have successfully enhanced the accuracy and relevance of book recommendations by considering the preferences and behaviors of users with similar tastes. Additionally, we have conducted evaluations using precision, recall, and average precision metrics to assess the performance of our recommendation system. These evaluations have provided valuable insights into the system's effectiveness and allowed us to optimize the ranking and binary relevance of the recommendations. Overall, our project strives to provide users with a more tailored and satisfactory book discovery experience, ensuring that they can explore and enjoy books that align with their interests and preferences.

VI. References

[1] Real Python. (2022, August 18). Build a recommendation engine with collaborative filtering. Real Python. https://realpython.com/build-recommendation-engine-collaborative-filtering/#:~:text=beautiful%20%2B%20Pythonic%20code.-,What%20Is%20Collaborative%20Filtering%3F,similar%20to%20a%20particular%20user.

[2] What content-based filtering is &amp; why you should use it | upwork. (n.d.). https://www.upwork.com/resources/what-is-content-based-filtering

[3] chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https:/iopscience.iop.org/article/10.1088/1742-6596/1000/1/012101/pdfAvastHTML/Shell/Open/Command

[4] uniQin.ai. (2022, May 28). Cosine similarity for item based collaborative filtering. Cosine Similarity for Item Based Collaborative Filtering.
https://blog.uniqin.ai/p/cosine-similarity-for-item-based

[6] Person. (2021, October 6). Understanding TF-IDF for Machine Learning. Capital One.
https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/#:~:text=TF%2DIDF%20(Term%20Frequency%20%2D,for%20a%20variety%20of%20tasks