Book Recommender System

Priyanka Yadav - 1102545
Department Of Computer Application
Graphic Era Deemed To Be University
Dehradun, India
priyanka130901@gmail.com

Abstract— This book recommendation system leverages a hybrid approach, combining popularity-based and collaborative filtering algorithms to enhance the user experience in discovering diverse and engaging literary content. The popularity-based component relies on aggregating user interactions to identify trending and widely acclaimed books. Concurrently, collaborative filtering algorithms do analysis on user's interests and preferences, recommending books based on the collective wisdom of similar users. This hybrid system addresses the limitations of each method individually, offering users a balanced and personalized selection of book recommendations. Through experimentation and evaluation, we explore the synergies between these algorithms to optimize the accuracy and diversity of suggestions, catering to a broad spectrum of reader preferences. The proposed system contributes to the evolving landscape of book recommendation methodologies, emphasizing the importance of combining different strategies to provide more robust and satisfying recommendations in the dynamic world of literature.

Keywords—Recommendation System, Hybrid, Product-based, Collaborative filtering.

I. Introduction

A recommendation system could be a software or an algorithm that essentially suggests data similar to the information entered by the user. In today's day and age recommendation system is everywhere from online shopping applications to social media applications. For instance, Amazon, the prominent online retailer, played a significant role in the advancement of collaborative filtering algorithms, which are used quite often in suggesting items to users. Music services such as Spotify heavily depend on the music preferences of users with similar tastes to generate weekly song recommendations and create personalized radio stations. Netflix, a renowned streaming platform for movies, web series, and various streaming services, employs these systems to suggest movies that viewers might find enjoyable. We can observe how recommendation systems significantly influence the content consumers interact with throughout their daily lives.

Akin to the examples mentioned above, there could also exist a recommendation system for books. A book recommender system is designed to suggest books to users based on their preferences and interests. The primary goal of a book recommendation system is to help users discover new and relevant books that they may enjoy, enhancing their overall reading experience. These systems leverage various techniques and data sources to provide personalized recommendations, making them a valuable tool for both avid readers and those seeking to explore new literary genres. A book recommendation system could be based on various algorithms but for this project I used popularity-based and collaborative filtering-based algorithms separately. I used in total 3 datasets for this project that were available on Kaggle. The datasets consist of books read by users and ratings provided by them on Amazon. The datasets were compiled by Cai-Nicolas Ziegler in 2004, and it comprises of three datasets named users, books and ratings.

II. Background Study

Recommendation systems employ three main approaches: (i) content-based, (ii) collaborative filtering, and (iii) hybrid. In the content-based approach, data essentially similar to those a user favored in the past are recommended. Conversely, collaborative filtering systems predict user preferences by analyzing relationships among users and interdependencies among items, extrapolating new associations from these patterns. Hybrid approaches seamlessly integrate both content-based and collaborative strategies, leveraging their complementary strengths and weaknesses to yield more robust results. However, this research focuses only on two main algorithms for the book recommender system which are popularity-based and collaborative filtering-based approaches.

Popularity-based recommender systems represent a straightforward yet effective approach to providing recommendations by leveraging the collective preferences of users. These systems prioritize books based on their overall popularity within the user community. The underlying assumption is that, those books favored by a large amount of users are likely to appeal to new users as well. One of the key advantages of popularity-based recommender systems is their simplicity and ease of implementation. Since they do not rely on complex algorithms or individual user behavior, they are computationally lightweight and easy to deploy. This simplicity makes them particularly suitable for scenarios where limited user data is available or when a quick and generic recommendation is needed.

Collaborative filtering is a powerful recommendation technique that relies on the preferences and behaviors of a community of users to suggest items of interest. This method draws insights from user interactions and similarities, suggesting books that align with the preferences of users who share similar tastes. In collaborative filtering, recommendations are generated by identifying users with similar behavior or preferences to the target user. The system then suggests items that those similar users have liked or interacted with. Collaborative filtering has proven to be effective in capturing user preferences, even in situations where the items or users have sparse data. In summary, collaborative filtering harnesses the collective wisdom of a user community to generate personalized recommendations, making it a widely utilized and effective technique in various recommendation systems.

III. Data

The dataset under consideration for the book recommendation system comprises 3 main components: The

book dataset is represented in a CSV format, containing 271,379 unique values, each representing a distinct book and each book, potentially including details such as titles, authors, genres, publication dates, and other relevant attributes. The richness of this dataset forms the foundation for the book recommendation system, enabling the algorithm to draw insights and patterns from the diverse characteristics of the books.

Complementing the book dataset, the ratings dataset captures the interactions of users with the books. The ratings dataset is also stored in a CSV format where each entry in this dataset likely corresponds to a user's rating for a specific book. It contains a total of 1,149,779 values, it provides a comprehensive record of user preferences and opinions, forming the basis for collaborative filtering within the recommendation system.

In addition to the book and ratings datasets, there is a separate dataset specifically dedicated to users. This additional dataset likely contains information about the ages of users who provided ratings for the books in the system. The inclusion of user age as a separate dataset introduces an extra layer of demographic information, which can be valuable for refining the book recommendation system.

The large number of unique books and ratings by users in the respective datasets allows for a broad and diverse set of recommendations, enhancing the user experience by offering choices that align with individual preferences.

IV. Popularity Based Approach

A. *Overview*

As the name suggests Popularity based recommendation system works with the trend. A popularity-based approach, often employed in recommendation systems, relies on recommending items that are popular or frequently chosen by users. Unlike personalized recommendation systems that tailor suggestions to individual preferences, popularity-based approaches make recommendations based on the overall popularity or general appeal of items. This method is straightforward and does not require detailed user information or complex algorithms.

B. *Algorithm*

The algorithm for a popularity-based recommendation system is relatively simple and does not involve complex computations or machine learning models. It primarily relies on aggregating popularity metrics for items to determine the most popular ones. For instance, choose a popularity metric based on the nature of your application. Common metrics include the number of reviews, likes, or ratings received by each item. For each item in the dataset, calculate its popularity score using the chosen metric. This involves counting the relevant interactions (e.g., ratings, reviews) associated with each item. Then rank the items in descending order of their popularity scores. The items with the highest scores are considered the most popular. Recommend the top-N items to users based on the ranking where N represents the number of items you want to recommend to users.

C. *Results*

Popularity-based recommendation systems are simple, scalable, and effective in certain scenarios, especially when personalized data is limited or when a baseline recommendation strategy is needed.

IMDB weighted average formula:

Weighted Rating (WR)=[vR/(v+m)]+[mC/(v+m)]

where,

v is the number of reviews for the book;

m is the minimum reviews required;

R is the average rating of the book; and

C is the mean of reviews across the whole report.

V. Collaborative Filtering Approach

D. *Overview*

Collaborative filtering (CF) systems forecast user preferences by examining historical connections among users and interdependencies among items. The fundamental principle behind CF is that if person A shares a similar opinion with person B on one issue, they are more likely to align on a different issue compared to a randomly chosen person. CF algorithms fall into two categories: non-probabilistic and probabilistic. Among the myriad CF algorithms, we have delved into two: UV-decomposition and k-nearest neighbor. In this project I used non-probabilistic approach using kNN algorithm.

E. K Nearest Neighbors

K Nearest Neighbors is a memory-based algorithm also known as kNN. Nearest neighbor algorithms are a family of popular non-probabilistic collaborative filtering techniques. This technique computes the similarity between either users (user-to-user based collaborative filtering) or items (item-to-item based collaborative filtering) that are represented by columns or rows in a user-item matrix with similarity metrics such as cosine, adjusted cosine, or euclidean. This algorithm then predicts unknown ratings based on known ratings of similar users or items. The similarity values "provide the means to give more or less importance to these neighbors in the prediction".

The k Nearest Neighbors (kNN) algorithm takes a user-book pair (ui, bj) as input and provides a prediction for user ui's unknown rating for book bj. When applying the kNN algorithm for book rating prediction, there are two distinct approaches: user-based and item-based.

User-based kNN examines all users who have rated book bj, selecting k users among these co-rated users that are most similar to ui. These k users, termed the k nearest neighbors of ui, contribute their ratings for book bj to predict ui's rating.

In contrast, item-based kNN begins by identifying all the books rated by user ui. From this set, it selects k books that are most similar to book bj (the k nearest neighbors). The ratings of user ui for these k nearest neighbors then contribute to predicting ui's rating for

book bj. The rationale behind item-based kNN is that users tend to rate similar books similarly.

The decision to utilize item-based kNN is grounded in the perception that relationships between books are more stable than those between users. Additionally, given the dataset's composition with around 10 times more users than books, comparing similarities between books is expected to be more time-efficient than comparing similarities between users.

Specifically, the implementation of item-based kNN takes a list of user-book tuples [(u01, b01), (u02, b02), ..., (u0n, b0n)] as input and produces a list of predicted ratings. Each entry in this list corresponds to the predicted rating of u'i for b'i in the input list.

For each user-book tuple (u'i, b'i) in the input list, the implementation identifies the books rated by u'i. For each book bt already rated by u'i, it computes the similarity between b'i and bt. This involves extracting the row vectors corresponding to b'i and bt from the utility matrix and applying a similarity metric, such as cosine similarity, adjusted cosine similarity, or Euclidean distance similarity. The resulting similarity score quantifies the similarity between b'i and bt. The formulas for computing the similarity score using these metrics are provided below. It's essential to note that only co-rated entries for each vector are considered in the calculations.

In the formulas below, U represents the set of co-rated users who have rated both books i and j, Ru,i is the known rating of user u for book i, and $\bar{R}u$ is the average rating of user u.

$$sim_{\text{cosine}}(i,j) = \frac{\sum_{u \in U} R_{u,i} * R_{u,j}}{\sqrt{\sum_{u \in U} R_{u,i}^2}\sqrt{\sum_{u \in U} R_{u,j}^2}}$$

$$sim_{\text{adjusted cosine}}(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}$$

$$sim_{\text{euclidean}}(i,j) = \frac{1}{\sqrt{\sum_{u \in U}(R_{u,i} - R_{u,j})^2 + 1}}$$

After completing this process for all the rated books, we select k rated books with the highest similarity scores and designate them as the k nearest neighbors of b'i. Subsequently, we calculate a weighted sum of u'i's ratings for these k books, where the weights are proportional to the similarity scores. This weighted sum represents the algorithm's predicted rating of u'i for b'i:

$$\text{predicted rating } u'_i \to b'_i = \frac{\sum_{j=1}^{k} sim(b^{(j)}, b'_i) \times \text{rating } u_i \to b^{(j)}}{\sum_{j=1}^{k} sim(b^{(j)}, b'_i)},$$

In the context of the formula, where b(j) represents the jth nearest neighbor of book bi.
Furthermore, we have observed considerable variability in how different users assign ratings to books. While some users consistently rate all books with the highest possible score of 5, others may predominantly use the lowest rating of 1. This diversity in rating behaviors makes direct comparisons challenging. To address this issue, we employ a normalization technique. For each user ui, we calculate their average rating f̂i. Subsequently, we subtract this average rating from every individual rating given by the user. This normalization process ensures that the adjusted cosine similarity metric accounts for the varying rating behaviors among users, resulting in a normalized average rating of 0 for each user. It is important to note that the adjusted cosine similarity metric is specifically designed to accommodate these differences in users' rating behaviors.

F. *Tuning Parameters*

In our algorithm outlined above, there are three parameters that require tuning: the number of neighbors (k), similarity measurements, and normalization. Consequently, we are presented with a multitude of combinations to assess. With k ranging from 2 to 30 neighbors [7], similarity metrics encompassing cosine similarity, adjusted cosine similarity (also known as Pearson correlation), and Euclidean distance, along with data normalization, we have a total of 174 distinct combinations. The objective is to identify the most effective combination for evaluating the algorithm's performance with the evaluation dataset.
To elaborate further, as mentioned earlier, we partition user ratings into three sets: training, development, and evaluation. Using the training dataset, we apply the 174 different combinations to predict ratings for books and users in the development dataset. Subsequently, we calculate the Root Mean Square Error (RMSE), a measure of how far the predictions deviate from the actual ratings. Among the 174 RMSE values, we select the combination with the lowest RMSE as the optimal choice. This selected combination is then employed for testing on the evaluation data. The subsequent section provides more detailed numerical results of this operational process.

G. *Results*

Evaluation Metric - RMSE (Root Mean Square Error): The root mean square error is utilized as our evaluation metric, providing insight into the average difference between our predictions and the actual ratings provided by users.
We calculate the root mean square error using the following equation:

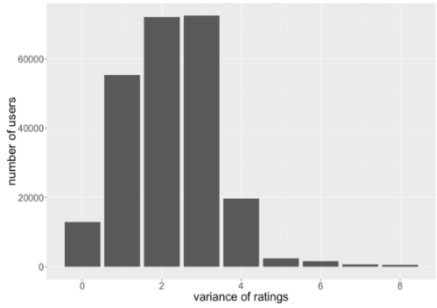
$$RMSE = \sqrt{\frac{1}{n}\sum_{(i,j)}(p_{(i,j)} - r_{(i,j)})^2}$$

Where:
- n is the total number of ratings,
- p(i,j) is the predicted rating,
- r(i,j) is the actual rating for a given book j, user i pairing.

The RMSE value provides an average measure of the difference between our model's predicted ratings

and the actual ratings observed. In this context, the predicted ratings for the development data are based on users' preferences learned from the training data. The use of this evaluation metric helps us identify the optimal combination of variables (number of neighbors, similarity metric, and normalization) for predicting ratings.

One might anticipate that the more books a user rates, the more accurate the predictions for that user would be. However, this graph indicates the contrary, suggesting that the number of ratings a user has given is inconsequential. These results may seem peculiar when considered in isolation. However, upon closer examination of our dataset, the findings align with the nature of our data. The variance in user ratings is notably low for the dataset, indicating that users tend to rate books similarly.



VI. Conclusion

In conclusion, our hybrid book recommendation system seamlessly integrates the strengths of popularity-based and collaborative filtering algorithms, creating a comprehensive and effective platform for personalized book recommendations. By harnessing the power of popularity metrics, the system identifies widely appreciated books, ensuring that users are exposed to trending and acclaimed literary works. Concurrently, the collaborative filtering component enhances the personalization aspect by analyzing user preferences and leveraging collective insights from similar readers. This dual approach not only addresses the challenges of the "cold start" problem and limited user data but also provides a more nuanced and tailored recommendation experience.

Through rigorous experimentation and evaluation, I have demonstrated the efficacy of my hybrid system in striking a balance between popular trends and individual preferences. The collaboration between algorithms enhances the overall recommendation accuracy and user satisfaction. The versatility of this system makes it applicable across diverse genres and user profiles, fostering a dynamic and inclusive reading environment.

This approach emerges as a promising strategy to navigate the complexities of book recommendations. My work contributes to the ongoing discourse on recommendation algorithms, shedding light on the potential synergies that can be harnessed to create more robust and user-centric recommendation systems in the realm of literature. In the ever-expanding world of books, my hybrid approach stands as a testament to the efficacy of combining diverse strategies to offer readers an enriched and personalized journey through the vast landscape of literature.

VII. Refrences

1. Nursultan Kurrnashov, Konstantin Latuta and Abay Nussipbekov, "Online book recommendation System", *2015 Twelve International Conference on Electronics Computer and Computation (ICECCO)*, pp. 1-4, 2015.

2. Anand Shanker Tewari and Kumari Priyanka, "Book recommendation system based on collaborative filtering and association rule mining for college students", *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 135-138, 2014.

3. Marwa Hussien Mohamed, Mohamed Helmy Khafagy and Mohamed Hasan Ibrahim, "Recommender systems challenges and solutions survey", *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pp. 149-155, 2019.

4. Tessy Badriyah, Erry Tri Wijayanto, Iwan Syarif and Prima Kristalina, "A hybrid recommendation system for E-commerce based on product description and user profile", *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, pp. 95-100, 2017.

5. Rishabh Ahuja, Arun Solanki and Anand Nayyar, "Movie recommender system using K-Means clustering and K-Nearest Neighbor", *2019 9th International Conference on Cloud Computing Data Science Engineering (Confluence)*, pp. 263-268, 2019.

6. Jinny Cho, Ryan Gorey, Sofia Serrano, Shatian Wang, JordiKai Watanabe-Inouye, “Book Recommendation System”, *Winter 2016.*