Book Recommendation System for Young Adults

Jessica Stow (STWJES003@MYUCT.AC.ZA)

2024-09-25

View this report on my GitHub profile!

This report's repository can be viewed on my GitHub profile.

Plagiarism declaration

- I know that plagiarism is wrong.
- Plagiarism is to use another's work and pretend that it is one's own.
- I have used the required convention for citation and referencing.
- Each contribution to and quotation in this assignment from the work(s) of other people has been attributed, and has been cited and referenced.
- This assignment is my own work.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
- I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Objectives

The objectives of this report were as follows:

1. Build recommender systems that predict the rating a user will give to a book based on each of:

a) item-based collaborative filtering,

b) user-based collaborative filtering, and

c) matrix factorisation.

2. Assessment and ensemble model:

a) Assess the accuracy of the matrix factorisation recommender system, using a single train/test sample.

b) Assess the accuracy of the matrix factorisation recommender system with and without regularisation.

c) Create a final model that ensembles the predictions from the three approaches, and then assess the accuracy of the ensemble predictions.

In this report I have opted to build specific recommender systems for young adults between the ages of 18 and 25.

Introduction to Data Mining and Recommender Systems

In the era of big data, recommender systems have become an integral part to many online systems, providing users with personalised suggestions aimed at enhancing user engagement and satisfaction. These systems utilise data mining techniques to offer personalised suggestions by analysing patterns in user preferences and behaviours (Data Mining: Concepts and Techniques, 2012).

The collaborative approach, one of the most widely used approaches, is a filtering approach that specifically focuses on identifying users with similar tastes or preferences, recommending items based on the opinions and actions of those with shared interests. This method may also take into account a user's social environment to enhance the relevance of the recommendations. Here we will focus on the use and application of the User-Based, Item-Based and Matrix Factorisation collaborative filtering approaches. But first, it is important to understand the concept of *cosine similarity* and its relevance to the former two approaches.

Cosine Similarity

Cosine similarity is a metric used in both user-based and item-based collaborative filtering to measure how similar two vectors are.

Given two vectors, x and y, the cosine similarity is defined as:

$$cos(\theta) = \frac{x \cdot y}{||x||\ ||y||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

Cosine similarity ranges from 0 to 1, with higher values indicating greater similarity. As two vectors become more aligned, the angle between them decreases, and the cosine similarity approaches 1, reflecting highly similar user preferences. Conversely, a larger angle results in a cosine similarity closer to 0, indicating that the preferences are very different.

User-Based Collaborative Filtering

User-Based Collaborative Filtering accounts for a user's interests by identifying similar users and recommending items that those users have shown interest in (Grus, 2015).

To get an idea of how similar two users are we need to use the cosine similarity metric, which quantifies how alike any two users are based on their preference vectors.

Item-Based Collaborative Filtering

Item-based collaborative filtering (IBCF) takes an alternative approach by computing similarities between items, rather than users. Recommendations are then generated for each user by aggregating items that are similar to the ones the user has shown interest in (Grus, 2015).

Cosine similarity is again used to calculate similarity. If two items are of interest to the same users, their similarity will be closer to 1. If no users show interest in both items, their similarity will be closer to 0. Recommendations are generated by summing the similarities of items related to the user's current interests.

Collaborative filtering with matrix factorisation

Matrix factorisation offers a different approach to collaborative filtering, rooted in linear algebra, where the goal is to fill in missing values within a matrix. Also known as matrix decomposition, it involves representing a matrix as the product of two smaller matrices. The key concept behind this method is the discovery of **latent factors** — hidden features that capture meaningful patterns in the data.

In recommendation systems, matrix factorisation decomposes the user-item ratings matrix into two smaller matrices in such a way that the known ratings are closely approximated. A key advantage of this approach is that, while the original ratings matrix is incomplete (with missing entries), the decomposed matrices are fully populated. This allows for predicting the missing values in the original matrix, effectively filling in the blanks and making recommendations based on the latent factors derived from the data.

Data description

The dataset used in this report was obtained from Kaggle's freely available Book Recommendation Dataset. The data was collected by Cai-Nicolas Ziegler in a four-week-long crawl between August and September 2004 from the Book-Crossing community with kind permission from Ron Hornbaker, CTO of Humankind Systems. It contains 278 858 users (anonymised, but with demographic information) providing over 1 million ratings (explicit/implicit) about 271 379 books.

The dataset consists of three files:

1. **Users:** which contains the user information:

`User.ID`: the unique, anonymised user identifier.

`Location`: the location of the user (in the format of city, state, country).

`Age`: the age of the user.

2. **Books:** which contains the book and content based information (which have been obtained from Amazon Web Services):

`ISBN`: the unique identifier for each book.

`Book.Title`: the book title.

`Book.Author`: the book author (in the case of several authors, only the first is provided).

`Year.Of.Publication`: the year of publication.

`Publisher`: the publisher of the book.

`Image-URL-S`, `Image-URL-M`, and `Image-URL-L`: URLs linking to cover images of the books in size small, medium and large, respectively. These URLs point to the Amazon web site.

3. **Ratings:** which contains the book rating information:

`User.ID`: the unique user id of the user rating the book.

`ISBN`: the ISBN (identifier) of the book rated.

`Book.Rating`: the rating given by the user. These ratings are either explicit (expressed on a scale of 1-10 where higher values indicated higher appreciation), or implicit, expressed by 0.

Exploratory data analysis

No duplicate entries were identified in any of the datasets.

The `Books` dataset contains information on 271,360 unique books, distinguished by their ISBNs.

The `Users` dataset includes details for 278,858 unique users, identified by their user IDs.

The `Ratings` dataset consists of 1,149,780 ratings. Within this dataset, 105,283 unique users provided ratings (both explicit and implicit), representing approximately 38% of the total user base. Interestingly, these users rated 340,556 books, a number that exceeds the total listed in the `Books` dataset.

Univariate analysis

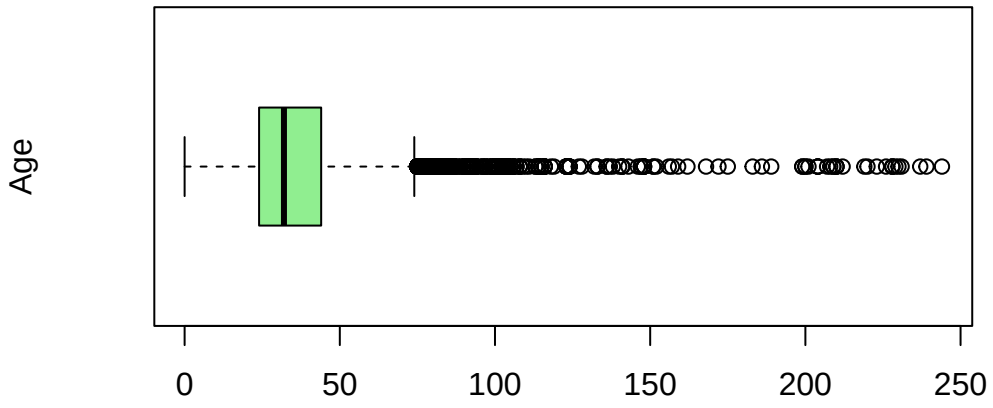
The investigation focused on two key variables: user age and book ratings.

User location and additional book details were excluded from the analysis, as they were deemed irrelevant to the objectives of this recommender system exercise.

User Age

User ages ranged from 0 to 244 years, with a mean of 35 and a median of 32. This wide range presents significant issues, as it is implausible for users to be 0 or 244 years old. These outliers are likely due to data entry errors. After excluding these extreme values, the age distribution appears largely symmetrical and uniform.

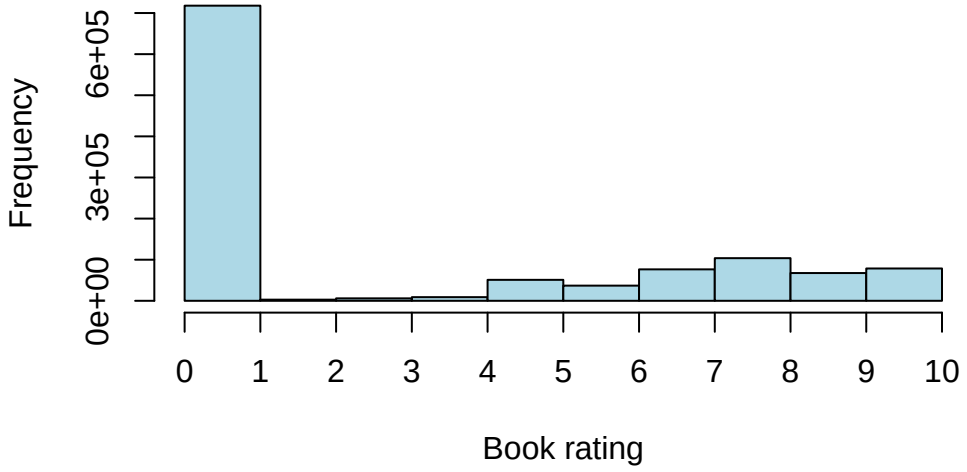
Figure 1: Boxplot of User Age



Book ratings

Book ratings ranged from 0 to 10, with a rating of 0 signifying an implicit interaction, indicating that the user may have read or interacted with the book without providing an explicit rating. Ratings between 1 and 10 were explicit, reflecting direct user input. As illustrated in *Figure 2* below, the majority of ratings were implicit, which will be less valuable for our recommender system as we proceed.

Figure 2: Distribution of Book rating



Bivariate analysis

Ratings by user

It was observed that one user (ID 11676) rated an impressive 13,607 books, a figure significantly higher than the average of just 10 book ratings per user. Upon further investigation, it was noted that this user did not provide any location or age information.

Ratings by book

The most rated book was "Wild Animus" by Rich Shapero, published in 2004, which received a total of 2502 ratings.

Several books had an average rating of 10, though in many cases these ratings were based on input from only a single user.

Methods

Data manipulation

Ratings

Implicit ratings (those equal to 0) were excluded, as they are not useful for building the recommender model. Additionally, only books with 5 or more ratings were retained, in order to exclude books that are rarely read.

User ratings were further filtered to include only those from users who had rated 10 or more books (i.e., users who rated at least the average number of books). This approach ensured a focus on active readers, providing a more reliable understanding of their preferences. After applying these filtering methods, the resulting ratings data frame contained 111 008 rows, meaning approximately 1 million observations (~90% of the original ratings data) were excluded.

Users

The analysis was narrowed to focus on the preferences of young adults between the ages of 18 and 25 (inclusive), with all other age groups excluded from the data frame. Additionally, user location information was discarded, as it did not contribute meaningfully to the recommender model development.

Books

It was ensured that all books had a title. An investigation was conducted into instances where the Year of Publication field contained non-numeric values, as this would prevent converting the column to an integer. It was discovered that, for three books, the Book Author and Year of Publication fields had been mistakenly swapped. This issue was resolved by correcting the swapped information. The Year of Publication column was then converted to an integer type.

The ISBN, book title, author, and year of publication information were retained, while the publisher and image URLs were removed from the `books` data frame.

Joining of data frames

The `Books` data frame was merged with the `Ratings` data frame using a left join on the common variable `ISBN`. This approach ensured that all ratings information was retained while adding the corresponding book titles and authors to the dataset.

To resolve the earlier discrepancy, where more books were rated than were listed in the `Books` data frame, a check was performed for missing book titles. It was found that 7249 ratings lacked corresponding book titles or any associated information. Consequently, these ratings were removed from the data frame.

The user information was then merged into the new data frame using a left join, matching on the shared user IDs. A left join was selected to ensure that all data related to the books and their ratings remained intact. Following this, it was noted that 97 202 ratings lacked age information and, as a result, these records were removed from the data frame.

The final merged data frame contained the following variables: the book's ISBN, title, author, year of publication, the user's rating, and their age. It comprised 10 768 ratings.

Collaborative filtering with matrix factorisation

Matrix factorisation was implemented using the `recosystem` package. To prepare the data in the required format, we structured the ratings dataset so that each row represented a unique rating, with columns for the user ID, book ISBN, book title, and rating score. It was necessary to convert both the user IDs and book ISBNs into factors, as the matrix factorisation model in `recosystem` relies on categorical encoding of users and items. This step ensured that the model could properly identify and differentiate between users and books.

After preparing the data, we initialised a matrix factorisation model using the `recosystem` package by creating an instance of the `Reco()` object. This object serves as the model that will be trained on our data to learn user-item interactions.

Next, we split the data into training and test sets to evaluate the model's performance. We used an 80/20 split, where 80% of the data was randomly selected for training and the remaining 20% was used for testing. To ensure reproducibility so that the same random split can be generated each time the code is run, we set a random seed.

To apply the `recosystem` package for matrix factorisation, the training and testing data must be converted into a format compatible with the package. This is achieved using the `data_memory()` function, which transforms the data into memory-efficient objects that `recosystem` can process.

For the training data:

- The `user_index` (user ID), `item_index` (ISBN for each book), and the rating (the actual rating provided by the user) are specified.

- This establishes a memory-based data object that connects users, books, and ratings, which is then utilised for model training.

For the testing data:

- A similar process was applied to the test set. The `user_index` and `item_index` were specified along with the actual ratings, enabling the evaluation of the model's performance on unseen data.
- The training set comprised 8614 observations, which were utilised to train the model.

Collaborative filtering was subsequently conducted using matrix factorisation, both with and without the inclusion of a regularisation term.

Without regularisation term

The matrix factorisation model was trained on the training set using the `train()` function from the `recosystem` package. In the absence of a regularisation term, the model exclusively learned latent factors for users and items without penalising large or complex factor values. This approach increased the risk of overfitting, particularly with sparse data, as the model could fit the training data too closely, capturing noise rather than identifying generalisable patterns.

After completing the training phase, predictions were generated for the test set using the `predict()` function. These predicted ratings were stored for later evaluation. However, due to the lack of regularisation, while the model might have performed well on the training set, it may have struggled to generalise to unseen data in the test set, potentially resulting in reduced predictive accuracy.

With regularisation term

To improve the performance of matrix factorisation in collaborative filtering, L2 regularisation was introduced. L2 regularisation incorporates a penalty term into the loss function, which discourages large parameter values. This technique helps mitigate overfitting, particularly in sparse datasets, a common issue in recommendation systems. In the `recosystem` package, L2 regularisation is applied to both user and item latent factors by adjusting the `costp_l2` and `costq_l2` parameters during model training. These parameters regulate the strength of regularisation for the user and item factors, respectively.

A grid search was conducted to determine the optimal combination of `costp_l2` and `costq_l2` values that would minimise the RMSE for the predictions.

The following values were tested:

- `costp_l2`: 0.001, 0.01, 0.1, 0.5, 0.6, 0.8, 1
- `costq_l2`: 0.001, 0.01, 0.1, 0.5, 0.6, 0.8, 1

This method enabled the identification of the most effective regularisation settings to enhance prediction accuracy and prevent overfitting.

The best combination of hyperparameters were a `costp_l2` of 0.5 and a `costq_l2` of 0.01, which resulted in a RMSE of ~1.83.

This combination was selected for the final model with L2 regularisation to generate predictions.

User-Based Collaborative Filtering

To compute user similarities, the ratings matrix was first converted into a wide format, where each row corresponded to a user (User.ID) and each column represented a book (ISBN).

Given that users exhibit different rating tendencies, with some consistently assigning higher or lower ratings, mean-centering was applied to standardise the data. Without this normalisation, users with similar preferences could appear dissimilar due to variations in their rating styles. Mean-centering adjusted the ratings to reflect how much a user liked or disliked a book **relative to their own average**, thereby aligning their rating patterns. This step ensured the data was appropriately prepared for calculating cosine similarity between users, capturing their true preferences and facilitating more accurate recommendations.

Any `NA` values, representing missing ratings for a given book, were replaced with zero. This step was essential, as cosine similarity cannot process missing values. Following this, the matrix was converted into a sparse format, which is more computationally efficient when dealing with matrices that contain a high proportion of zero values.

Cosine similarities between users, based on their book ratings, were calculated using the `simil()` function from the `proxyC` package, which is optimised for efficiently handling sparse matrices. The resulting user similarities sparse matrix was then converted into a dense matrix to facilitate easier computation and predictions in subsequent steps.

Predictions

To evaluate the predictions of the user-based collaborative filtering model, predictions were made on the same test set used in the matrix factorisation model. A function was created to calculate the user-based predicted rating while accounting for each user's average rating.

Within the function, the cosine similarity scores for the target user were first retrieved and then standardised so that they summed to one, but only across users who had rated the specific book. The predicted rating for the book was computed as a weighted sum of the

ratings provided by other users, with the weights corresponding to the standardised similarity scores. To return the prediction to its original scale, the user's average rating was added to the result.

This approach ensured that both user similarities and individual rating patterns were accounted for when generating predictions.

The function was then applied to the same test set used in the collaborative filtering model with matrix factorisation.

Item-Based Collaborative Filtering

To compute item similarities, the ratings matrix was first transformed into a wide format, where each row represented a book (ISBN) and each column represented a user (User.ID).

Given that books may receive varying ratings from different users, mean-centering was applied to standardise the data. Without this normalisation, books with similar levels of popularity could appear dissimilar due to differences in user rating habits. By applying mean-centering, the ratings were adjusted to reflect how well a book was rated relative to its average rating across all users. This step ensured that the data was properly prepared for calculating cosine similarity between items, capturing each book's true popularity and enabling more accurate recommendations.

Any `NA` values, representing missing ratings for a given user, were replaced with zero. This step was essential as cosine similarity cannot process missing values. The matrix was then transformed into a sparse format, improving computational efficiency when working with matrices containing a significant number of zero values.

Cosine similarities between books, based on their ratings from users, were calculated using the `simil()` function from the `proxyC` package, which is optimised for efficiently handling sparse matrices. The resulting item similarities sparse matrix was then converted into a dense matrix to facilitate easier computation and predictions in subsequent steps.

Predictions

To evaluate the item-based (IB) collaborative filtering model's predictions, the same test set used for the matrix factorisation model was employed. A function was developed to calculate the predicted rating for each user, while adjusting for the user's average rating.

The function first retrieved the cosine similarity scores for the target item and standardised them so that they summed to one, but only across items that had been rated by the specific user. The predicted rating for the item was then calculated as a weighted sum of the ratings provided for similar items, with the weights determined by the standardised similarity scores.

Finally, the predicted rating was adjusted by adding the user's average rating to return it to the original scale.

This approach ensured that both item similarities and individual rating tendencies were accounted for when generating predictions.

The function was subsequently applied to the same test set used in the matrix factorisation-based collaborative filtering model.

Ensemble model

The final model was an ensemble, which averaged the predictions from the user-based, item-based, and matrix factorisation models (without regularisation). Predictions were only included in the averaging process if they fell within the (valid) range of 1 to 10 (inclusive).

Accuracy assessment

The predictions from the five models were compared with the actual values, and their accuracy was assessed using mean squared error (MSE) and root mean squared error (RMSE) as evaluation metrics.

Results

The accuracy results have been represented in *Table 1* below, along with a summary of the key insights.

Table 1: Collaborative Filtering Model Performance Comparison

Model	MSE	RMSE
Matrix Factorisation (without reg.)	3.39	1.84
Matrix Factorisation (with reg.)	3.30	1.82
User-based Collaborative Filtering	11.75	3.43
Item-based Collaborative Filtering	3861.38	62.14
Ensemble Model	**1.41**	**1.19**

1. **CF with MF (without regularisation):** The matrix factorisation model without regularisation performs reasonably well, but the absence of regularisation may lead to overfitting. This can cause the model to perform better on training data but slightly worse on unseen data compared to regularised models.
2. **CF with MF (with regularisation):** The inclusion of L2 regularisation marginally improves the performance, reducing both MSE and RMSE. This suggests that regularisation helps prevent overfitting, allowing the model to generalise better to unseen data by penalising overly complex latent factors.
3. **User-based CF:** The user-based collaborative filtering model performs significantly worse compared to the matrix factorisation models. The high MSE and RMSE indicate that this approach might struggle to capture the complex relationships between users and books, especially in cases of sparse data.
4. **Item-based CF:** The item-based collaborative filtering model performs poorly, with a dramatically higher MSE and RMSE compared to other methods. This suggests that item similarity is not as effective in predicting ratings in this dataset, likely due to the large number of sparse entries or poor similarity measures among items.
5. **Ensemble Model:** The ensemble model, which averages predictions from the user-based, item-based, and matrix factorisation models, performs the best overall. Its MSE and RMSE are the lowest, indicating that combining the strengths of different models results in improved prediction accuracy. The ensemble approach seems to balance the shortcomings of individual models, especially the weaknesses of user-based and item-based collaborative filtering.

In summary:

- The **ensemble model** outperforms all other approaches, demonstrating that a combined approach leads to more accurate predictions.
- **Matrix factorisation with regularisation** is the second-best performing model, as it effectively generalises the data and avoids overfitting.
- **User-based** and especially **item-based collaborative filtering** perform poorly, highlighting the limitations of these traditional techniques, particularly in datasets with sparsity or diverse rating patterns.

Discussion

The results of the five collaborative filtering models highlight significant differences in performance across the methods implemented. Each model offers its own unique strengths and weaknesses, and the results provide insights into the effectiveness of different recommendation techniques, particularly when applied to the specific demographic of young adult readers.

Matrix Factorisation Models

The matrix factorisation models demonstrated solid performance in both the regularised and non-regularised model forms. The model without regularisation achieved a Root Mean Squared Error (RMSE) of **1.84**, but with the introduction of L2 regularisation, the RMSE was reduced to **1.82**. This small improvement illustrates the benefit of regularisation in preventing overfitting by discouraging overly complex latent factors.

The ability of matrix factorisation to capture latent relationships between users and books makes it particularly effective in handling sparse data, a common challenge in recommender systems. However, the relatively small improvement with regularisation suggests that the model may already be robust, or that the dataset used does not suffer significantly from overfitting. Future work could explore different types of regularisation or deeper grid searches to optimise hyperparameters more precisely.

User-Based Collaborative Filtering

User-based collaborative filtering (UBCF) performed notably worse compared to the matrix factorisation models, with an RMSE of **3.43**. This result is expected, as UBCF relies heavily on the availability of dense user interactions. In the presence of sparse data, UBCF can struggle to identify meaningful relationships between users, especially in a dataset like this, where only a small proportion of users have rated a large number of books.

Additionally, user-based approaches can be more susceptible to issues of variability in individual rating behaviour, which can distort predictions. Despite applying mean-centering to standardise user ratings, the model still struggled to capture the full complexity of user preferences.

Item-Based Collaborative Filtering

The item-based collaborative filtering (IBCF) model yielded the worst performance with a significantly higher RMSE of **62.14**. This high error indicates that item-based similarities were not sufficient to generate accurate predictions in this dataset. One possible explanation

is that the dataset may contain too many sparsely-rated books, leading to inaccurate similarity calculations between items.

This method could benefit from further data preprocessing such as only including books that have a ratings count above a certain threshold.

Ensemble Model

The ensemble model, which averaged predictions from the user-based, item-based, and matrix factorisation models, performed the best overall, achieving an RMSE of **1.19**. This result suggests that the ensemble approach successfully combined the strengths of each model, mitigating their individual weaknesses. By leveraging the power of multiple collaborative filtering techniques, the ensemble model was able to provide more accurate predictions than any individual model alone.

One reason for this success could be that the ensemble model effectively balances the over-reliance on latent factors in matrix factorisation with the direct similarity measures in user- and item-based filtering. This highlights the advantage of using ensemble methods in recommender systems, as they can provide more robust predictions by incorporating diverse approaches.

Limitations and recommendations

Despite the success of the ensemble model, there are several limitations to this analysis. The most significant challenge was the sparsity of the dataset, with a large proportion of books and users having very few ratings. This affected the performance of both user-based and item-based collaborative filtering.

Another potential limitation is the relatively basic hyperparameter tuning process used for the matrix factorisation model. While L2 regularisation improved performance, a more extensive grid search could yield further improvements.

Conclusion

In conclusion, the results demonstrate the power of ensemble models in creating robust recommender systems, particularly when combining collaborative filtering approaches. Future work could expand on this by incorporating additional data sources, experimenting with more advanced algorithms, and fine-tuning model parameters to further enhance performance.

References

Grus, J. (2015). Data Science from Scratch: First Principles with Python. 1st ed. O'Reilly Media.

Han, J., Kamber, M. & Pei, J. (2012). Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, CA, USA.