Reliable Book Recommender System: An Evaluation and Comparison of Collaborative Filtering Algorithms

Aldin Kovačević(✉) and Zerina Mašetić

Faculty of Engineering and Natural Sciences, Department of Information Technologies, International Burch University, Sarajevo, Bosnia and Herzegovina
aldin.kovacevic@stu.ibu.edu.ba

Abstract. Nowadays, a vast number of online businesses and services use recommender systems to provide relevant product suggestions to their customers. Recommender systems can be defined as systems employing various data mining techniques to predict users' interest in certain items or topics, among a tremendous amount of available data. While there exist varying methodologies to construct a recommender system, one of the most potent and commonly used techniques is collaborative filtering (CF), which evaluates items based on the opinions of other users. This paper examines several commonly used collaborative filtering models with the goal of creating a reliable book recommendation system. The research focuses on training, evaluating and comparing several neighborhood-based and matrix factorization-based recommender algorithms, using a combination of two book datasets: the BookCrossing dataset compiled in 2004, and the Goodbooks "most popular books" dataset from 2017. Grid search with cross-validation is used to find the best performing combination of several hyper-parameters and different CF models. Ultimately, a matrix factorization-based SVD++ approach is selected as the most performant model, with MAE of 0.57 and RMSE of 0.752, making it a considerably good model based on the provided data. Moreover, a manual comparison between a user's book history and their recommended book titles shows that users are generally recommended books of the same or similar genres. Lastly, the paper discusses certain shortcomings of the proposed model and offers several suggestions for future improvements.

Keywords: Recommender systems · Collaborative filtering · Book recommendations · Neighborhood-based approaches · Matrix factorization-based approaches

1 Introduction

With the ever-growing amount of information and data available on the internet, users have an increasing need of tools to find and access appropriate information. Recommender systems are an example of one of the most successful tools for that purpose. In technical terms, a recommendation system (or recommendation engine) can be defined

N. Ademović et al. (Eds.): IAT 2021, LNNS 316, pp. 264–280, 2022.
https://doi.org/10.1007/978-3-030-90055-7_20

as a system which operates on a business environment, taking a user's online footprint as input and generating a probable output footprint consisting of item predictions for that user, which are as close to reality as possible [1]. Recommender systems have quickly become an important aspect of the information and e-commerce landscape, allowing users to quickly filter through large swaths of information and product spaces in order to find the relevant data [2].

Throughout history, humans have always relied on "recommendation systems", in one way or another. Before the information age took a considerable hold in society, people used recommendations or advice from their peers or experts in a particular field to guide their decision-making process and discover new material. However, this approach was limited in the sense that there might exist an "independent item" that a person would like, but if no one close to that person is familiar with the item, it would be unavailable to said person. This led to the creation of computer-based recommendation systems that can "expand" the circle of users from whom a person can obtain recommendations, and mine users' history and preferences to uncover new patterns, providing a more finely-tuned experience [2]. Over the past 20 years, there has been a multitude of various approaches to building recommender systems [3], with the most commonly used (and most performant) approaches being content-based algorithms and collaborative filtering (CF) algorithms [4]. Content-based algorithms attempt to recommend items similar to the items the user has liked in the past, correlating with the user's preferences [5]. On the other hand, collaborative filtering algorithms recommend items to users by evaluating the preferences, opinions and ratings that other users have expressed regarding those items [6]. While both of these approaches have their benefits and limitations, this paper is going to focus on collaborative filtering. Since content-based filtering relies on the properties of actual content to be recommended, its results might not transfer from one domain to another. On the other hand, collaborative filtering relies only on the user rating matrices for particular items, which are usually similar in form across multiple domains [8]. Moreover, collaborative filtering has been met with major success in both research and practice, being widely used in information filtering and e-commerce applications [7].

The topic of recommender systems has been the subject of numerous research papers for quite some time, primarily focusing on movie, music and product recommendations [8]. However, few papers have explored the field of book recommendations [1, 9]. This paper presents an exploratory study which aims to produce a satisfactory recommender system model that can be employed for relevant book recommendations on a bookshop or library website. After performing an exploratory analysis of two popular book datasets, BookCrossing and Goodbooks-10k, this research paper bases its approach around the collaborative filtering (CF) methodology. Namely, since CF recommender systems can depend on a variety of factors and initial parameters, grid search with cross-validation is applied in order to find the best performing set of model hyper-parameters. In short, the main contributions of this paper can be summarized as follows:

1. creating an acceptable book recommendation system by testing a variety of collaborative filtering approaches with a combination of parameters, and deducing the best performing model
2. fine-tuning of the recommender model and discussing possible future improvements.

2 Literature Review

The prototypes of recommender systems occurred fairly early in the history of computing, with Grundy [10], a computer-based librarian, being an early step towards automatic recommendation systems. Grundy functioned by grouping users into common "stereotypes" based on short interview questions and using hard-coded information about various stereotypes to generate favorable book recommendations. While fairly primitive in its execution (and according to today's standards), it represented one of the first major entries in the space of recommender systems.

First major strides regarding collaborative filtering started occurring in the early 1990s, when CF-based approaches began to arise as a solution to online information overload [2]. One of the first examples were Tapestry [11], a manual collaborative filtering system for relevant data retrieval, and GroupLens [12], which identified Usenet articles that could be interesting to a user, based on other users' ratings. Soon afterwards, various collaborative recommender systems, such as BellCore [13] for movies, Ringo [14] for music and Jester [15] for jokes followed. Moreover, recommender systems were also recognized in the marketing sphere for the purposes of increasing sales and improving customer experience [16], culminating in Amazon's adoption of recommender technology which could suggest relevant items to customers based on their previous purchase history, and the history of similar customers [3].

Due to the popularity and well-documented nature of collaborative filtering methods, there exists a plethora of relevant research papers on recommendation systems. In their 2005 paper, Adomavicius and Tuzhilin [3] made an extensive categorization of currently known recommender algorithms into content-based, collaborative and hybrid approaches, summarizing their possible extensions. On the other hand, Su and Khoshgoftaar [17] gave more focus to CF-based methods, including memory-based, model-based and hybrid methods. Their survey outlines some of the most sophisticated recommendation algorithms available as of 2009. Another excellent survey about collaborative filtering systems was produced by Ekstrand et al. [2], explaining the process which goes into creation and evaluation of CF algorithms in great detail. Apart from these theoretical surveys, a couple of experimental studies have also been conducted. A study by Breese et al. [18] compares the performance of two widely used memory-based methods – Pearson correlation and vector similarity – against two classical model-based approaches – clustering and Bayesian network. The comparison was done on three different datasets, indicating that Bayesian network and correlation-based methods feature a better performance. An experimental study into CF algorithms for e-commerce was conducted by Huang et al. [19], comparing user-based, item-based, dimensionality reduction-based and a few other model-based methods. This study uses precision, recall and F1 score as its evaluation metrics, ignoring the standard MAE and RMSE error scores commonly used in other papers. Recommender systems research was additionally reinvigorated in 2006, with the launch of Netflix's 1-million-dollar prize competition [22] to come up with an improved movie recommendation algorithm. The winning approach implemented a hybrid method which used temporal dynamics to account for the timestamps of movie ratings [23], and the Netflix dataset is still used today as a standard dataset for evaluating the performance of CF algorithms [8].

With the advent of the Netflix competition and the rise of e-commerce, most of the research into recommendation systems has been focused on movies, businesses and commercial products [2]. In terms of book recommender systems, a few relevant studies have been conducted. Kurmashov et al. [9] implemented a collaborative filtering model using the Pearson correlation as its similarity method. Based on user feedback, the quality of their recommender system was rated at 77%. Parvatikar and Joshi [30] combined collaborative filtering with association mining techniques to fill in the missing ratings, creating a system with an average MAE score of 0.7. Another collaborative filtering system with association mining was independently proposed and implemented by Patil et al. [31]. Rana and Deeba [32] attempted a slightly different collaborative filtering approach, by utilizing Jaccard similarity - the ratio of common book readers vs. the readers who rated the books individually. Their approach yielded an RMSE of 1.504 on a dataset with a 10-based rating scale. Uko et al. [33] proposed a collaborative filtering model with adjusted cosine similarity, averaging an RMSE of 0.9. On the other hand, Sase et al. [1] suggested a hybrid recommendation approach, combining collaborative filtering, relevant book properties and user demographics. Cho et al. [34] tested a variety of recommendation system approaches, including a k-NN collaborative filtering model with adjusted cosine similarity which yielded an RMSE of 1.29 on a 10-rating scale.

In recent years, recommender techniques other than collaborative filtering have begun to emerge, namely content-based approaches rooted in information retrieval, Bayesian inference and case-based reasoning methods [20]. Content-based methods work with the actual content or properties of items in the recommender system. These algorithms can use item content to supplement user rating patterns, or completely replace them. Apart from that, as various recommender strategies matured, hybrid recommender systems [21] emerged, combining multiple algorithms into composite systems that build upon the strengths of its individual parts. Despite all of that, collaborative filtering has remained an effective recommendation approach, both on its own and as a hybridized approach with additional content-based properties.

3 Materials and Methods

Among a variety of recommender system approaches, this paper focuses on building a collaborative filtering (CF) model. Collaborative filtering is a popular family of recommendation algorithms that gives out recommendations based on the ratings or behavior of other users in the system [2]. With CF, the ratings of other users in the system can be gathered and selected in order to predict another user's likely preferences. In other words, if a group of users agree about their preference for certain items, they are also likely to agree about other items which they might not have seen yet. To elaborate further, if a group of users prefers the same items as user A, then user A is likely to prefer the items that the group likes, which the user has not seen yet. This approach will be the key principle examined throughout the research paper. Figure 1 showcases an overview of the primary steps taken during the research process.

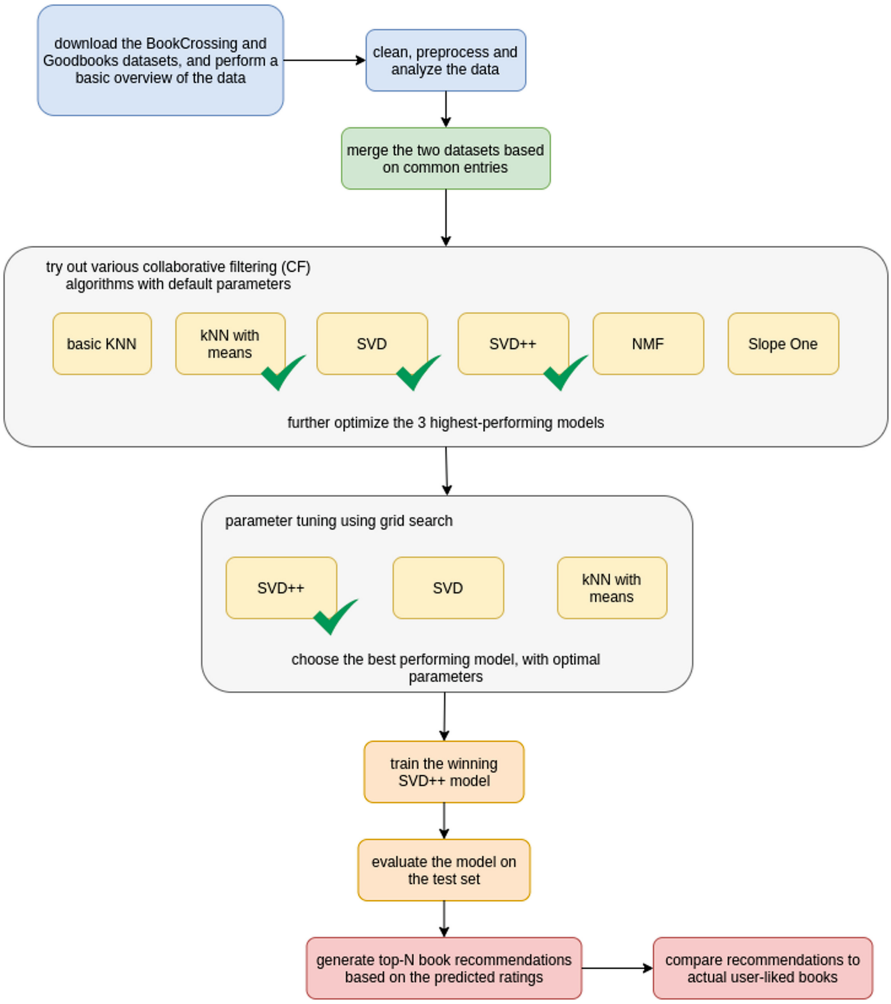

Fig. 1. Model diagram with key steps

3.1 Dataset Overview

Two different book datasets are used as the basis of this research: the BookCrossing [24] and Goodbooks-10k "top 10,000 books" [25] datasets. The BookCrossing dataset was compiled from the BookCrossing forum over a 4-week period in 2004, and comprises 271,379 books, 275,858 users with anonymized demographics and 1,149,780 implicit and explicit book ratings. On the other hand, the Goodbooks-10k dataset consists of 10,000 most popular books up to 2017 crawled from a book review website "Goodbooks", with around 6 million ratings and 53,424 users. Additionally, this dataset contains a list of user-defined book tags, which can be used to infer literary genres.

Both of the datasets contain several key files. The "books.csv" file of the BookCrossing defines the ISBN number, book title and author, publisher and year of publication,

and links to the book cover image in three formats (small, medium and large). The user ratings are defined separately in a "ratings.csv" file, with user ID, book ISBN (foreign key) and the actual rating from 0 to 10. Additionally, the set contains a "users.csv" file with anonymized user demographics (user ID, location and age). The Goodbooks-10k "books.csv" equivalent defines the same columns as BookCrossing, with a few additions. The file contains a "book ID" (which is later used to join the file with ratings), and several other IDs, such as Goodreads ID and edition IDs. Moreover, the file has information about the book language, number of reviews and average rating. Goodbooks-10k's "ratings.csv" is similar to the BookCrossing's file, with the exception of using the book ID as the foreign key, not the ISBN number. Moreover, ratings are given on a scale from 1 to 5. Lastly, this dataset also contains a "tags.csv" file which comprises a collection of user-given tags for various books. With some preprocessing, this collection can be used to infer book genres.

3.2 Data Cleaning and Preprocessing

After performing an initial exploratory and statistical analysis of the two datasets, they are processed and prepared for use with the recommender model. Standard data cleaning and preprocessing steps are performed, including the removal of duplicate items, replacement of N/A and outlier values with median/mode values and standardization of column names (since the two datasets use different names for their attributes).

Special preprocessing steps are undertaken with the rating datasets. First of all, the BookCrossing ratings also contain implicit ratings, or books "rated" with a 0. Since a 0-rating is not indicative of a user's preference for that item, implicit ratings are removed from the dataset. Secondly, since BookCrossing ratings are presented on a scale from 1 to 10, they are transformed down to a rating scale of 0.5 to 5, to be in line with Goodbooks-10k ratings. The resulting rating distributions are showcased on Fig. 2. The distribution of book ratings for both datasets suggests that users are generally more likely to give a positive rating (3.0 and up) to a book than a non-favorable rating.

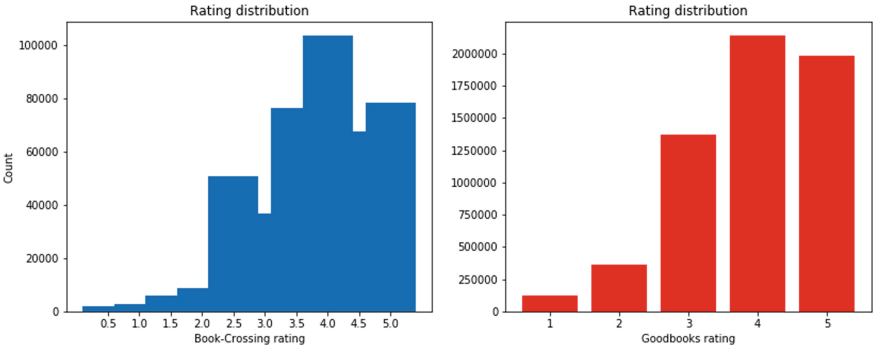

Fig. 2. Rating distribution across both datasets

Lastly, to have a balanced representation of both datasets in the final model, a combined set of ratings is generated and limited to around 30,000 ratings. To ensure statistical

significance, the ratings are extracted from users who have rated at least 50 books and books which have received at least 25 ratings. One of the reasons for the dataset reduction is the computational constraints involved with subsequent model training and evaluation. Through repeated experimentation and testing, a dataset of ~30,000 ratings was found to be computationally feasible, producing satisfactory results.

3.3 Collaborative Filtering Techniques

Based on their approach, collaborative filtering (CF) recommender systems are generally categorized into two distinct families: memory-based and model-based methods [8]. Memory-based algorithms load in the entire rating matrix into memory when making recommendations. These algorithms try to find relevant recommendations based on the similarity between a given user/item and the remainder of the rating matrix. Memory-based algorithms can adapt to changes in the data, but their processing time can be large, depending on the size of the input dataset. On the other hand, model-based algorithms fit a parameterized model to the rating matrix, later providing recommendations based on the produced model. This family of algorithms has constant time regardless of input size, but the model needs to be recomputed whenever data changes.

Neighborhood-based methods are among the most used memory-based CF techniques. These k-NN inspired algorithms predict ratings by looking at the users with ratings similar to the input user, or at items that are similar to the input item. The main assumption behind neighborhood-based approach is that if two users have similar preferences regarding a certain set of items, they are likely to have similar preferences about the remaining items. Alternatively, if two items are rated similarly by a portion of the users, the two items are likely to be rated in a similar manner by the remaining users. Based on these two different approaches (user similarity and item similarity), neighborhood-based algorithms can be further classified into user-based CF methods [5] and item-based CF methods [18]. As the name suggests, the user-based approach identifies users that are similar to the input user, and estimates the user's preference for an item to be the average rating given out by these similar users. Alternatively, the item-based approach identifies items that are similar to the input item and predicts the item's rating to be the average rating of these similar items. The user/item similarity can be calculated in a variety of ways, with the most commonly used similarity measures being Pearson correlation, (adjusted) cosine similarity and mean-squared difference (MSD) [2].

On the other hand, model-based methods fit a parametric model to the training data. This model can then be used to predict unseen ratings and provide reasonable item recommendations. Model-based approaches are more varied than neighborhood-based algorithms, with some popular options being cluster-based CFs, Bayesian classifiers, regression-based methods and matrix factorization-based approaches [8]. In recent years, matrix factorization-based methods have become particularly popular, primarily due to the success of the SVD (singular value decomposition) algorithm during the "Netflix Prize" competition [23]. The main approach of matrix factorization-based algorithms is to factorize a rating matrix into a product of two low-rank matrices (user-profile and item-profile) which are then used to estimate the missing ratings [26]. In an SVD approach, each user is a row in a matrix structure, each column is an item and elements of the matrix are ratings given to the items by users. Factorizing this matrix using SVD results

in the extraction of latent factors, mapping each user and item into a lower-dimensional latent space. This makes it easier to clearly represent the relationships between users and items, and calculate the expected user ratings [35].

In order to create a suitable book recommender system, this paper examines the most popular algorithms from both the memory-based and model-based algorithm families, giving a special focus to neighborhood-based and matrix factorization-based methods, as the most common representatives of their respective algorithm families. Moreover, the research also tests whether a user-based or an item-based approach yields better results for a neighborhood-based model.

3.4 Recommender Algorithms

The training and evaluation of recommender models was done using Surprise [27], a scikit-based Python library for building and analyzing recommender systems. Several representative models of major collaborative filtering approaches were tested, namely neighborhood-based and matrix factorization-based models. The neighborhood-based models work on the principle of finding similarities between items or users. Two neighborhood-based (memory-based) algorithms were selected, basic k-NN and k-NN with means, with the difference being that the latter takes the mean ratings of each user into account. Matrix factorization-based models extract features and correlation from user and item matrices. Out of matrix factorization-based (model-based) algorithms, the paper examined SVD [26], SVD++ [23], which is a variant of SVD that captures implicit ratings (the fact that a user rated an item, regardless of the rating value) in addition to explicit ratings and NMF [28], an algorithm similar to SVD, which prevents negative elements in factorized matrices. Lastly, a linear regression-based Slope One [29] algorithm was also tested.

All candidate models were fitted to the training set generated from relevant ratings using Surprise (80-20 split), and evaluated through a 5-fold cross-validation process. The best performing models were later fine-tuned and optimized using the grid search approach to select the optimal combination of parameters. Table 1 showcases the parameters used to build the initial models.

Table 1. Initial recommender model parameters

Recommender model	Chosen parameters
Basic k-NN	{k = 40, similarity = 'cosine', user_based = True}
k-NN with means	{k = 40, similarity = 'cosine', user_based = True}
SVD	{n_epochs = 20, lr_all = 0.005, reg_all = 0.02}
SVD++	{n_epochs = 20, lr_all = 0.007, reg_all = 0.02}
NMF	{n_epochs = 20, biased = False, reg = 0.06}
Slope one	No special parameters

4 Model Optimization and Evaluation

Various evaluation measures for prediction accuracy have been used throughout the available research into collaborative filtering systems. The most common ones are the mean absolute error (MAE) and the root mean squared error (RMSE). MAE is measured on the same scale as the original rating, and represents a measure of how much a predicted rating differs from the original rating:

$$MAE = \frac{1}{n} \sum_{u,i} \left| p_{u,i} - r_{u,i} \right| \tag{1}$$

where $p_{u,i}$ is the predicted and $r_{u,i}$ the actual rating a user u has given to an item i. On the other hand, RMSE places a greater emphasis on larger errors, penalizing them more. Since errors are squared before being averaged, RMSE gives a higher weight to large errors [2].

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

4.1 Initial Model Evaluation

Since this paper aims to minimize potential large errors in predicted ratings, RMSE was used as the primary performance metric to evaluate the accuracy of the recommender model, as well as determine the best combination of parameters during the grid search phase. However, the MAE value was calculated for every algorithm as well, since it represents the average offset between predicted and actual ratings. The lower the value of RMSE/MAE is, the better accuracy is expected from the recommender model. The performance of the initial models was evaluated using a 5-fold cross-validation on the training set. Figure 3 shows a unified plot of MAE (red) and RMSE (blue) values for all six trained algorithms.

Based on the RMSE plot, SVD++ and SVD are the best performing algorithms, with RMSE values of 0.765 and 0.769, respectively. Table 2 showcases the full list of MAE and RMSE scores for all tested algorithms.

Looking at both performance metrics (RMSE and MAE), SVD++ and SVD algorithms are in the lead, with SVD++ having a slight advantage (0.769 vs. 0.765 for RMSE, and 0.582 vs. 0.59 for MAE). Out of the remaining models, the closest to them is k-NN with means, having an RMSE of 0.837 and MAE of 0.623. The remaining models (NMF, basic k-KNN and Slope One), while not displaying an overly bad performance, still lag behind the top algorithms. In general, it seems that matrix factorization-based recommendation models outperform neighborhood-based and regression-based algorithms, at least for the particular ratings dataset used in this paper.

4.2 Grid Search and Parameter Tuning

SVD++, SVD and k-NN with means achieved the best performance metrics during the initial testing, and therefore merit further research. Using grid search with 5-fold cross-validation, these three algorithms will be trained and evaluated with different parameter

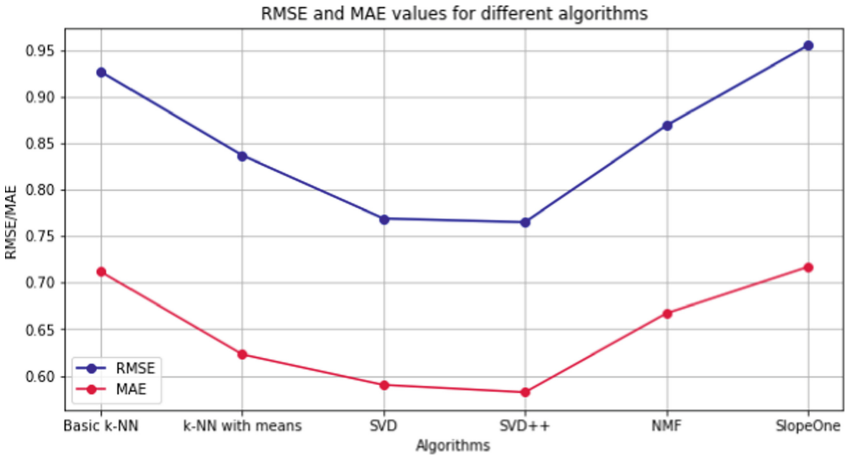

Fig. 3. MAE and RMSE scores for various algorithms

Table 2. RMSE and MAE scores for different models

Recommender model	RMSE	MAE
Basic k-NN	0.927	0.712
k-NN with means	0.837	0.623
SVD	0.769	0.59
SVD++	0.765	0.582
NMF	0.869	0.667
Slope one	0.955	0.717

grids, to see if hyper-parameter tuning can lead to better (lower) RMSE and MAE scores than those obtained in the initial iterations. The algorithms and their parameter grids are outlined in Table 3.

Table 3. Grid search parameter grids for different models

Recommender model	Parameter grid
SVD++	{n_epochs = [20, 25, 40], lr_all = [0.005, 0.007, 0.009, 0.01], reg_all = [0.2, 0.4, 0.6]}
SVD	{n_epochs = [20, 25, 40], lr_all = [0.005, 0.007, 0.009, 0.01], reg_all = [0.2, 0.4, 0.6]}
k-NN with means	{k = [15, 20, 25, 30, 40, 50, 80, 100], similarity = [‘pearson’, ‘cosine’], user_based = [True, False]}

For SVD++ and SVD, the relevant parameters are the number of epochs (n_epochs), the learning rate (lr_all) and the regularization term (reg_all). In k-NN with means, there exist two different kinds of adjustable parameters. First of all, a parameter inherent to k-NN models is the number of neighbors to be examined (k). However, all memory-based models also depend on the similarity function and the approach to collaboration: user-based or item-based. Neighborhood-based models in Surprise use the cosine similarity and user-based filtering by default, so the grid search will examine the possibility of utilizing a different similarity function and/or item-based filtering to increase performance. Like in the initial phase, the algorithms were fitted and evaluated on the training set. RMSE was used as the metric for choosing the best subset of hyper-parameters. The optimal parameters for each model, along with RMSE and MAE values, are available in Table 4.

Table 4. RMSE, MAE and optimal parameters after grid search

Recommender model	RMSE	MAE	Optimal parameters
SVD++	0.759	0.581	{n_epochs = 25, lr_all = 0.007, reg_all = 0.2}
SVD	0.761	0.582	{n_epochs = 20, lr_all = 0.009, reg_all = 0.2}
K-NN with means	0.828	0.62	{k = 100, similarity = 'cosine', user_based = False}

Looking at the newly computed performance metrics, SVD++ once again shows the best results, with RMSE of 0.759 and MAE of 0.581. It is closely followed by regular SVD, with 0.759 and 0.581 for RMSE and MAE, respectively. k-NN with means, while performing better than the original model, does not stack up to SVD-based algorithms. However, it is worth noting that grid search evaluation found item-based filtering to be the more performant approach (user_based = False), suggesting that item-based filtering is the preferred option for models based on similarity (such as k-NN), given the contents of the examined book ratings dataset. In conclusion, after conducting a grid search hyper-parameter optimization, SVD++ is the best performing algorithm, so it will be used throughout the remainder of this paper.

5 Experimental Results

After initial tests and evaluation, the selected recommender model was the matrix factorization-based SVD++, with 25 epochs, learning rate of 0.007 and regularization term of 0.2. Experimental results obtained by using this model are outlined in the following sections.

5.1 Test Set Evaluation

Cross-validation and tuning of hyper-parameters were done on the previously extracted training subset of data, whereas 20% of the original data was withheld for the test set. The test set is used to get an unbiased estimation of the model's performance by applying

it to unknown samples. After re-training the SVD++ model on the training set using the previously obtained optimal parameters, rating predictions were generated for the samples in the test set. Upon evaluation, RMSE for the test set was 0.752, and its MAE was 0.57. These metrics closely resemble the cross-validation metrics calculated via the training set, which indicates that the model is unbiased and can generalize well to unseen data. Figure 4 presents a visualization of actual (blue) versus predicted (red) ratings for 50 randomly sampled books, given by different users.

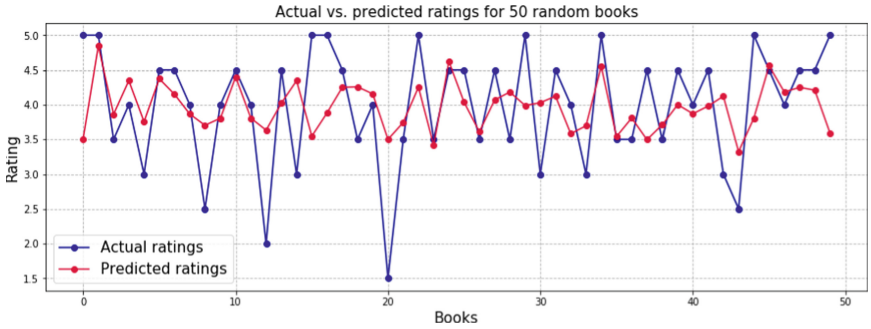

Fig. 4. Actual vs. predicted ratings for randomly sampled books

Based on the rating plot, it seems that the predicted ratings generally follow the trend of actual ratings: higher predicted ratings mostly correspond to higher actual ratings, and vice versa (except for a few outlier cases), sometimes even closely matching them.

5.2 Generating Top-N Recommendations

Since the chosen recommender model performs well on unseen data, it is suitable for use as the crux of the main recommender system which generates top-N recommended books for a given user. First, a so-called "anti-test set" is constructed. Since not every user has rated every book, a collection of all (user, book) pairs which are not in the original ratings dataset is constructed. This ensures that the user will not be recommended books which they have already read. SVD++ is then fitted to the anti-test set and rating predictions are generated for all (user, book) tuples. When a user wants to receive book recommendations, top N books (N being the number of books the user requests) with the highest predicted ratings by that user are selected to be displayed. Table 5 showcases an overview of the top-10 manually rated books (by the user) and the top-10 books recommended by the system for a randomly sampled user from the dataset, with an ID of 270713.

Judging from the top-rated books, user 270713 prefers books that are more fictitious in nature, such as sci-fi and fantasy genres (making up half of the selection). Additionally, they also enjoy an occasional drama or an autobiographical work. When it comes to the recommended books, it seems like the system picked up on the user's preferences, with half of the recommendations being sci-fi or fantasy books. Moreover, the remaining books are mostly related to drama and autobiographical genres, which are also present

Table 5. Top-rated and top-recommended books for a randomly selected user

Highest rated books by the user		Top books recommended by the system	
R	Title (genre)	R	Title (genre)
5.0	The Autobiography of Malcolm X (autobiography)	4.59	Weirdos From Another Planet! (sci-fi, comic)
5.0	Anne Frank: The Diary of a Young Girl (autobiography)	4.57	A Tree Grows in Brooklyn (semi-autobiography)
5.0	The World According to Garp (drama)	4.55	The Two Towers (fantasy)
5.0	Xenocide (sci-fi)	4.53	A Wrinkle in Time (sci-fi)
5.0	Ender's Game (sci-fi)	4.51	The Return of the King (fantasy)
5.0	Watership Down (adventure)	4.5	84 Charing Cross Road (drama)
5.0	The Red Tent (drama)	4.49	Where the Sidewalk Ends (poetry)
5.0	Speaker for the Dead (sci-fi)	4.47	Lolita (drama)
5.0	The Mists of Avalon (fantasy)	4.46	The Little Prince (novella)
5.0	Ender's Shadow (sci-fi)	4.45	Good Omens (fantasy)

in the books previously read by the user. All in all, it appears that the recommender system tends to produce books similar to the user's previous reading history.

6 Discussion

Based on the performance metrics evaluated using the final SVD++ recommender model, this paper produced satisfactory results. With RMSE of 0.752 and MAE of 0.57, the model seems to exhibit a better [30, 33] or comparable [32, 34] performance in comparison with related research. On the other hand, certain papers utilized different scoring metrics [9] or did not provide metrics at all [1, 31] and focused only on the proposal of the algorithm, making them difficult or impossible to adequately compare with the results obtained in this research. Moreover, even among the papers with usable performance metrics, different book datasets and rating scales are used, so one should be careful when analyzing these studies, making sure to examine them in context. For an easier overview, Table 6 showcases a comparison between previous research with relevant metrics and the results obtained during this study ("–" stands for "not reported or not applicable").

Of the two metrics, it is generally more difficult to interpret the meaning of RMSE in a practical sense (due to it stemming from the squared error). However, MAE is straightforward: it represents the average difference between predicted and original ratings. With a MAE of 0.57, the predictions generated by this model, on average, differ from the actual book ratings by ± 0.57. Since the ratings in the dataset are scaled from 0.5 to 5, with steps of 0.5, this can be considered a great metric, as it means that the average

Table 6. Metrics reported by previous research vs. this paper

Research	Method	RMSE	MAE	Other metrics
Kurmashov et al. [9]	CF with Pearson correlation	–	–	77% quality (user feedback)
Parvatikar and Joshi [30]	CF with association mining	–	0.7	–
Uko et al. [33]	CF with cosine similarity	0.9	–	–
Cho et al. [34]	CF using k-NN and cosine similarity (0–10 rating scale)	1.29	–	88.66% precision & 91.16% recall (using a content-based approach)
Rana and Deeba [32]	CF with Jaccard similarity (0–10 rating scale)	1.504	–	–
This paper	**CF using SVD++**	**0.752**	**0.57**	–

predictions do not deviate too much from the original ratings. While it is not feasible to manually inspect the recommended books for every user in the dataset, a showcase of predictions for a few random users, e.g. the user in Table 5, yields an interesting observation: even though ratings are the only feature used in the recommender model, the model still manages to suggest books of similar genres to the ones the user has enjoyed in the past. This fact, along with low error scores, indicates that the constructed recommender system has considerable quality and relevance.

Looking at Fig. 4, which plots the actual vs. predicted ratings for a random subset of books, a peculiar detail can be noticed: while the predicted ratings generally follow the trend of actual ratings (with a few outlier cases), the actual low ratings differ greatly from predicted low ratings. For several original ratings at the lower end of the scale, such as 2.5, 2 or 1.5, while the predicted ratings are lower than usual, they are nowhere near the actual rating. In fact, the lowest predicted rating on the plot is somewhere around 3.3, which is (at this scale) very far away from the actual lowest ratings. On the other hand, actual higher ratings (3.5+) generally have correspondingly high predicted ratings. This represents a drawback of the current model, and this disparity can be attributed to the uneven distribution of ratings in the input dataset. In Fig. 2, it can be seen that more than two-thirds of given ratings are neutral or positive (3.0+), meaning that users are generally more likely to submit positive ratings than unfavorable ones. Due to an insufficient number of low ratings in the training dataset, the model does not generalize well to them, leading to significant discrepancies between actual and predicted low ratings, as observed in Fig. 4. Using a dataset with more evenly distributed positive and negative ratings could be a way to mitigate this issue.

Despite the good performance and metrics exhibited by this model, there is still a lot of room for future work and improvements. First of all, the approach examined in this paper is a collaborative filtering algorithm relying solely on user ratings to recommend relevant books. Generally, CF algorithms use only the rating matrix and similarities

between user/item ratings to generate predictions. In the previous paragraph, it was also discussed how this can lead to skewed results, if the ratings are not evenly distributed. However, the dataset used in the paper contains some additional interesting properties, such as the book author, genre, user demographics (age and location) and book publication year; all of which are factors that could contribute to the user's preference for a book. Another potentially useful (but perhaps not immediately obvious) factor could be a book's "page number/price" ratio. Even though a book might fit a user's preference based on genre and other factors, some users might also want to ensure that they would be getting "their money's worth" from the book; in other words, that the amount they are paying is proportional to the amount of content provided. Using this ratio would require enriching the current dataset with book prices (since book prices can vary between different outlets and bookstores, a base reference should be established) or finding a different input dataset containing price data. Meaningfully incorporating all of these properties into the recommendation algorithm could lead to the creation of a more reliable hybrid collaborative filtering/content-based recommender system which might yield better performance and offer improved recommendations.

Another aspect which could be improved is the size of the input dataset. Due to computational constraints, the ratings dataset was limited to ~30,000 relevant entries for the duration of this study. A richer and more extensive dataset (along with a better machine) could lead to increased performance metrics and more variety in recommendations. Lastly, one of the issues of collaborative filtering in general is trust: since the recommendation system relies on the ratings given by users, is the feedback given by a user genuine or not? However, as this is a broad and general issue, it was not examined during this research. Inclusion of additional content-based parameters (and the creation of a hybrid algorithm) into the recommender model could offset issues related to genuineness of user ratings.

7 Conclusion

Overall, this study managed to yield several interesting results. Firstly, it resulted in a collaborative filtering book recommender system, based on the matrix factorization approach using SVD++, with RMSE of 0.752 and MAE of 0.57. Since the ratings in the input dataset range from 0.5 to 5, with a rating step of 0.5, this basically means that, on average, predicted ratings differ from the original ratings by ~0.57, which is an acceptable deviation. Moreover, upon inspecting the list of predicted books for a user and comparing them to the user's previous book resume, it appears that the algorithm does indeed recommend books of similar genres and ratings to the ones previously enjoyed by the user. While the model presents good overall results, there is still significant room for improvement. Inclusion of a larger dataset with more balanced positive/negative ratings could result in better generalization and more accurate predictions for lower ratings. Additionally, collaborative filtering could potentially be combined with other hybrid recommendation approaches; for example, incorporating user demographics, book genres and prices into the recommendation algorithm for more relevant results. All in all, this paper presents an interesting insight into collaborative filtering algorithms and makes a case that book recommender systems based on SVD++ definitely merit additional research.

References

1. Sase, A., et al.: A proposed book recommender system. IJARCCE, 481–483 (2015). https://doi.org/10.17148/ijarcce.2015.42108
2. Ekstrand, M.D.: Collaborative filtering recommender systems (2011). https://doi.org/10.1561/9781601984432
3. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005). https://doi.org/10.1109/tkde.2005.99
4. Almazro, A., et al.: A Survey Paper on Recommender Systems. Concordia Institute for Information Systems Engineering, Montreal (2010)
5. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10
6. Sarwar, B., et al.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the Tenth International Conference on World Wide Web – WWW 2001 (2001). https://doi.org/10.1145/371920.372071
7. Han, J., Kamber, M., Pei, J.: In Data Mining: Concepts and Techniques. Elsevier/Morgan Kaufmann, Amsterdam (2012)
8. Lee, J., Sun, M., Lebanon, G.: Comparative study of collaborative filtering algorithms. In: Proceedings of the International Conference on Knowledge Discovery and Information Retrieval (2012). https://doi.org/10.5220/0004104001320137
9. Kurmashov, N., Latuta, K., Nussipbekov, A.: Online book recommendation system. In: 2015 Twelve International Conference on Electronics Computer and Computation (ICECCO) (2015). https://doi.org/10.1109/icecco.2015.7416895
10. Rich, E.: User modeling via stereotypes*. Cogn. Sci. **3**(4), 329–354 (1979). https://doi.org/10.1207/s15516709cog0304_3
11. Goldberg, D., et al.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992). https://doi.org/10.1145/138859.138867
12. Resnick, P., et al.: GroupLens. In: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, CSCW 1994 (1994). https://doi.org/10.1145/192844.192905
13. Hill, W., et al.: Recommending and evaluating choices in a virtual community of use. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1995 (1995). https://doi.org/10.1145/223904.223929
14. Shardanand, U., Maes, P.: Social information filtering. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1995 (1995). https://doi.org/10.1145/223904.223931
15. Goldberg, K.: In Eigentaste: A Constant Time Collaborative Filtering Algorithm. Electronics Research Laboratory, College of Engineering, University of California, Berkeley (2000)
16. Ansari, A., Essegaier, S., Kohli, R.: Internet recommendation systems. J. Mark. Res. **37**(3), 363–375 (2000). https://doi.org/10.1509/jmkr.37.3.363.18779
17. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. Artif. Intell. **2009**, 1–19 (2009). https://doi.org/10.1155/2009/421425
18. Breese, J., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of Uncertainty in Artificial Intelligence (1998)
19. Huang, Z., Zeng, D., Chen, H.: A comparison of collaborative-filtering recommendation algorithms for E-commerce. IEEE Intell. Syst. **22**(5), 68–78 (2007). https://doi.org/10.1109/mis.2007.4338497
20. Schafer, J.B., Konstan, J.A., Riedl, J.: E-commerce recommendation applications. Appl. Data Min. Electron. Commerce **5**, 115–153 (2001). https://doi.org/10.1007/978-1-4615-1627-9_6

21. Burke, R.: Hybrid web recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_12
22. Bell, R.M., Koren, Y., Volinsky, C.: All together now: a perspective on the NETFLIX PRIZE. Chance **23**(1), 24 (2010). https://doi.org/10.1007/s00144-010-0005-2
23. Koren, Y.: Factorization meets the neighborhood. In: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008 (2008). https://doi.org/10.1145/1401890.1401944
24. Ziegler, C.-N., et al.: Improving recommendation lists through topic diversification. In: Proceedings of the 14th International Conference on World Wide Web, WWW 2005 (2005). https://doi.org/10.1145/1060745.1060754
25. Zajac, Z.: Goodbooks-10k: a new dataset for book recommendations, FastML (2017)
26. Pazzani, M., Billsus, D.: Learning collaborative information filters. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 46–54 (1998)
27. Hug, N.: Surprise, a Python library for recommender systems, Surprise (no date). http://surpriselib.com/. Accessed 20 Jan 2021
28. Luo, X., et al.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Industr. Inf. **10**(2), 1273–1284 (2014). https://doi.org/10.1109/tii.2014.2308433
29. Lemire, D., Maclachlan, A.: Slope one predictors for online rating-based collaborative filtering. In: Proceedings of the 2005 SIAM International Conference on Data Mining (2005). https://doi.org/10.1137/1.9781611972757.43
30. Parvatikar, S., Joshi, B.: Online book recommendation system by using collaborative filtering and association mining. In: 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC) (2015). https://doi.org/10.1109/iccic.2015.7435717
31. Patil, A.E., et al.: Online book recommendation system using association rule mining and collaborative filtering. IJCSMC **8**(4), 83–87 (2019)
32. Rana, A., Deeba, K.: Online book recommendation system using collaborative filtering (with Jaccard similarity). J. Phys. Conf. Ser. **1362**, 012130 (2019). https://doi.org/10.1088/1742-6596/1362/1/012130
33. Uko, E., Eke, B.O., Asagba, P.O.: An improved online book recommender system using collaborative filtering algorithm. Int. J. Comput. Appl. **179**(46), 41–48 (2018). https://doi.org/10.5120/ijca2018917193
34. Cho, J., et al.: Book recommendation system. Bachelor's thesis. Carleton College (2017)
35. Kumar, V.: Singular value decomposition (SVD) in recommender system. Anal. India Mag. (2021). https://analyticsindiamag.com/singular-value-decomposition-svd-application-recommender-system/. Accessed 10 May 2021