Personalized Book Recommendations for Adults Using Deep Learning and Sophisticated Filtering Approaches

Keywords: Book Recommender System, Deep Learning, Filtering

Abstract: Reading improves the reader's vocabulary and knowledge of the world. It can open minds to different ideas which may challenge the reader to view things in a different light. Reading books benefits both physical and mental health of the reader, and those benefits can last a lifetime. It begins in early childhood and continue through the senior years. A good book should make the reader curious to learn more, and excited to share with others. For some readers, their reluctance to read is due to competing interests such as sports. For others, it is because reading is difficult and they associate it with frustration and strain. A lack of imagination can turn reading into a rather boring activity. In order to encourage adults to read, we propose an elegant book recommender for adults based on a deep learning and filtering approaches that can infer the content and the quality of books without utilizing the actual content, which are often unavailable due to the copyright constraint. Our book recommender filters books for adult readers simply based on user ratings, which are widely available on social media, for making recommendations. Experimental results have verified the effectiveness of our proposed book recommender system.

1 INTRODUCTION

Studies have shown that those who read for pleasure have higher levels of self-esteem and a greater ability to cope with difficult situations (National Library, 2023). Reading for pleasure was also associated with better sleeping patterns. Adults who read for just 30 minutes a week are 20% more likely to report greater life satisfaction (Rayner et al., 2012). According to a study from Pew Research from 2021 (Basmo, 2023), 75% of Americans have read a book on a yearly basis. A study by Delgado et al. (Delgado et al., 2018) demonstrated that reading also improves written comprehension skills. Moreover, reading is good for a person, since it improves his focus, memory, empathy, and communication skills, besides reducing stress, improving mental health, and learning new things to help him succeed in his work and relationships (Narvaez et al., 1999). Even though there are many benefits on reading books, statistical data have indicated that U.S. adults are reading roughly two or three fewer books per year than they did between 2001 and 2016. The decline is greater among subgroups that tended to be more avid readers, particularly college graduates but also women and older Americans (Baron and Mangen, 2021). This decline closely tracks the rise of social media and iphones, since reading books is a leisure activity that struggles to compete with the popular online platform. This is an alarming rate, since ordinary people are not taking advantages of enriching themselves by reading and learning. We realize that adult life is busy and even adults who want to read sometimes just do not find the time. In order to motivate adults to read, it is essential to recommend books to them that are accessible and relevant in order to achieve the goal.

Adult learners expect information they are learning to be useful and applicable to their lives or interest. Assigning reading that is not just accessible, but is also relevant and enjoyable can make a big difference in motivating adult learners. Choosing books that relate directly to adult students' lives or careers and offering them the joy of reading can be useful. In helping adults to find appropriate books to read, we have developed *BkRec*, a personalized book recommender for adults, which adopts the *content-based* and *collaborative filtering* approaches to make personalized book recommendations for adults. The user-based and item-based collaborative filtering (CF) approaches are popular techniques for generating personalized recommendations (Desrosiers and Karypis, 2011). When the data sparsity becomes a problem for certain adult users, i.e., when there is not enough data to generate similar user groups or similar item groups to use the CF methods, the content-based filtering approach can be adopted to make personalized recommendations for the users.

BkRec, which is a novel recommender that exclusively targets adult readers, is a self-reliant recommender which, unlike others, does not rely on book metadata, such as book reviews and Library of Congress subject headings, that are either directly retrieved or inferred from the Web. BkRec is unique, since it explicitly considers the ratings and content descriptions on books rated by adults. In addition, BkRec first adapts Recurrent Neural Networks (RNN) for classifying books belonged to the categories that users are interested in reading as candidate books to be considered for making recommendations. Hereafter, content-based and collaborative filtering approaches are applying for ranking candidate books to be suggested to adult readers. Conducted empirical study have validated that the proposed book recommender for adults is effective, since it accurately predicates books preferred by adult readers.

2 RELATED WORK

There are a number of recent publications that are closely related to our work. Saraswt and Srishti (Saraswat and Srishti, 2022) apply RNN for classification books based on book plots and reviews and make book recommendations based on book categories. Our BkRec, on the other hand, only adapts RNN for classifying books into one of the 31 predefined categories. Hereafter, BkRec relies on various filtering approaches to classify books that are preferred by adult readers for making recommendations.

Mathew et al. (Mathew et al., 2016) also propose a book recommender system based on content-based and CF approaches. However, their content-based filtering system relies on the entire content of each book for making recommendation, which more often than not are unavailable due to the copyright constraint, as stated earlier. Moreover, they only consider the item-based CF method, without using the user-based approach, which is inadequate, since the latter often compromises the former to provide accurate prediction on user's preference.

Devika et al. (Devika et al., 2016) extract information from the users' reviews and then score to each book based on the users' feedback. In reality, however, this approach is unreliable, since it is highly unlikely that users provide reviews on books and offer feedback on various books due to the time constraint and commitment. Our book recommender system simply consider book descriptions supplied by book publishers and users' ratings on books that are widely available on social media.

Another book recommender system based on collaborative filtering is presented by Ramakrishnan et al. (Ramakrishnan et al., 2020). The authors assign explicit ratings to users solely based on the implicit feedback given by users to specific books using various algorithms. This approach, however, involves users in the feedback cycle, which is undesirable, since it is explicitly relied on engagement, which is not trustworthy due to the fact that users might not be willing to offer their feedbacks on books, especially with the ones that they are unfamiliar.

Fuli Zhang (Zhang, 2016) introduces a personalized book recommender based on time-sequential collaborative filtering, which is combined with college students' learning trajectories. The author, who relies on the availability of time sequence information of borrowing books and the circulation times of books, does not realize that these information are not widely available to the general public, and thus the adoption of their recommendation method is seriously hindered due to the data constraint.

There are a number of survey papers on book recommendations that have been published in the past , which include Anis et al. (Anis et al., 2021), Alharthi et al. (Alharthi et al., 2018), Gupta et al. (Gupta et al., 2020), etc.

3 OUR BOOK RECOMMENDER

We first utilize a deep neural network model to classify a book B given by a user U who often makes available a number of preferred books in his profile. Based on the category of B, we filter books in a collection that are in the same category as B, called *candidate books*. Hereafter, we combine different filtering approaches to recommend the top-ranked candidate books to U.

3.1 The Recurrent Neural Network (RNN) Model

We employ a recurrent neural network (RNN) as our classifier, since RNNs produce robust models for classification. Similar to other deep neural networks, RNNs are both trained (optimized) by the backpropagation of error and comprised of a series of layers: input, hidden, and output. An *input* layer is a vector or matrix representation of the data to be modeled, and a few *hidden* layers of activation nodes are included. Each of the hidden layer is designed to map its previous layer to a higher-order and higher-dimensional representation of the features which aims to be more useful in modeling the output than the original features. An *output* layer produces the desired output for classification or regression tasks.

RNNs achieve the recurrent pattern matching through its *recurrent layer(s)*. A recurrent layer is one which contains a single recurrent unit through which each value of the input vector or matrix passes. The recurrent unit maintains a *state* which can be thought of as a "memory." As each value in the input iteratively passes through the unit at time step t, the unit updates its state h_t based on a function of that input value x_t and its own previous state h_{t-1} as $h_t = f(h_{t-1}, x_t)$, where f is any non-linear activation.

Recurrent layers are designed to "remember" the most important features in sequenced data no matter if the feature appears towards the beginning of the sequence or the end. In fact, one widely-used implementation of a recurrent unit is thus named "Long-Short Term Memory", or LSTM. The designed RNN accurately classifies our data set of books solely based on their sequential text properties.

3.1.1 Feature Representation

To utilize a RNN, we need to provide the network with sequential data as input and a corresponding ground-truth value as its target output. Each data entry has to first be transformed in order to be fed into the RNN. Attributes of book entries were manipulated as follows:

- **Label**.The label consists of the category of a book, each of which is the top 31 categories pre-defined by Thriftbooks[1]. Since RNN cannot accept strings as an output target, each unique category string is assigned a unique integer value, which is transformed into a one-hot encoding[2] to be used later as the network's prediction target.
- **Features**. Features are extracted from the data set S as the *brief description* of a book, which is called a *sentence* of an entry, and is accessible from the book-affiliated websites such as Amazon[3]. Words in a brief description are transformed into *sequences*, or ordered lists of tokens, i.e., unigrams and special characters such as punctuation marks. Each sequence is padded with an appropriate number of null tokens such that each sequence was of uniform length. We have considered only the first 72 tokens in each sentence when representing the features, since over 90% of sentences in S contain 72 or fewer tokens. We considered the 6,500 most commonly-occurring tokens in S.
- **Text**. While extracting features, we have chosen not to remove stopwords, since we prefer not to lose any important semantic meaning, e.g., 'not', within term sequences nor punctuation, since many abstracts include mathematical symbols, e.g., '|', which especially correlate to certain categories. We did, however, convert all of the text in a sentence to lowercase because the particular *word embedding* which we used did not contain cased characters.

[1] https://www.thriftbooks.com/sitemap/

[2] A one-hot encoding of an integer value i among n unique values is a binarized representation of that integer as an n-dimensional vector of all zeros except the i^{th} element, which is a one.

[3] www.amazon.com

3.1.2 Network Structure

In this section, we discuss our RNN used for classifying book categories. Table 1 summaries different layers, their dimensions, and their parameters in our RNN.

The **Embedding Layer**. A design goal of our neural network is to capture relatedness between different English words (or tokens) with similar semantic meanings. Our neural network begins with an embedding layer whose function is to learn a *word embedding* for the tokens in the vocabulary of our dataset. A word embedding maps tokens to respective n-dimensional real-valued vectors. Similarities in semantic meanings between different tokens ought to be captured in the word embedding by corresponding vectors which are also similar either by Euclidean distance, or by cosine similarity, or both.

The embedding layer contains 1,950,000 parameters, since there are 6,500 vectors, one for each token in the vocabulary, and each vector comes with 300 dimensions, and all of which could be trained. Due to the large amount of time it would take to properly train the word embedding from scratch, we have performed two different tasks: (i) we have loaded into the embedding layer as weights an uncased, 300-dimensional word embedding, GloVe, which has been pre-trained on documents on the Web, and (ii) we have decided to freeze, i.e., not train, the embedding layer at all. The pre-trained vectors from GloVe sufficiently capture semantic similarity between different tokens for our task and they are not required to be further optimized. Since the embedding layer was not trained, it simply served to transform the input tokens into a 300-dimensional space. Therefore, instead of the 72-element vector which we started with, the embedding layer outputs a 72×300 real-valued matrix.

The **Bi-directional GRU Layer**. Following the embedding layer in our network is one type of recurrent layer – a bi-directional GRU, or Gated Recurrent Unit, layer. A GRU is a current state-of-the-art recurrent unit which is able to 'remember' important patterns within sequences and 'forget' the unimportant ones. This layer effectively 'reads' the text, or 'learns' higher-order properties within a sentence, based on certain ordered sequences of tokens. The number of trainable parameters in a single GRU layer

Table 1: Dimensions and number of parameters of layers in the RNN.

Layer	Output Dimensions	Total Parameters	Trainable Parameters
Input	72	0	0
Embedding	72×300	1,950,000	0
Bi-directional GRU	72×128	140,160	140,160
Global Max Pooling (1D)	128	0	0
Dropout 1	128	0	0
Dense Hidden	64	8,256	8,256
Dropout 2	64	0	0
Dense Output	31	845	845
Total		**2,099,261**	**149,261**

is $3 \times (n^2 + n(m+1))$, where n is the output dimension, or the number of time steps through which the input values pass, and m is the input dimension. In our case, $n = 64$, since we have chosen to pass each input through 64 time steps, and $m = 300$ which is the dimensionality of each word vector in the embedding space. Since our layer is bi-directional, the number of trainable parameters is twice that of a single layer, i.e., $2 \times 3 \times (64^2 + 64 \times 301) = 140{,}160$, the greatest number of trainable parameters in our network.

The recurrent layer outputs a 72×128 matrix, where 72 represents the number of tokens in a sequence, and 128 denotes the respective output values of the GRU after each of 64 time steps in 2 directions.

The **Global Max-Pooling Layer (1D)**. At this point in the network, it is necessary to reduce the matrix output from the GRU layer to a more manageable vector which we eventually use to classify the token sequence into one of the 31 categories. In order to reduce the dimensionality of the output, we pass the matrix through a *global max-pooling* layer. This layer simply returns as output the maximum value of each column in the matrix. Max-pooling is one of several pooling functions, besides sum- or average-pooling, used to reduce the dimensionality of its input. Since pooling is a computable function, not a learnable one, this layer cannot be optimized and contains no trainable parameters. The output of the max-pooling layer is a 128-dimensional vector.

The **Dropout Layer 1**. Our model includes at this point a dropout layer. Dropout, a common technique used in deep neural networks which helps to prevent a model from overfitting, occurs when the output of a percentage of nodes in a layer are suppressed. The nodes which are chosen to be dropped out are probabilistically determined at each pass of data through the network. Since dropout does not change the dimensions of the input, this layer in our network also outputs a 128-dimensional vector.

The **Dense Hidden Layer**. Our RNN model includes a dense, or fully-connected, layer. A *dense layer* is typical of nearly all neural networks and is used for discovering hidden, or latent, features from the previous layers. It transforms a vector x with N elements into a vector y with M inputs by multiplying x by a $M \times N$ weight matrix W. Throughout training, weights are optimized via backpropagation.

The **Dropout Layer 2**. Before classification, our RNN model includes another dropout layer to again avoid overfitting to the training sequences.

The **Dense Output Layer**. At last, our RNN model includes a final dense layer which outputs 31 distinct values, each value corresponding to the relative probability of the input belonging to one of the 31 unique categories. Each instance is classified according to the category corresponding to the highest of the 31 output values.

3.2 Our Filtering Approaches

Our book recommender considers two different filtering approaches in making book recommendations for adult readers: the content-based and collaborative-based filtering (CF) approaches. The content-based approach suggest books similar in *content* to the ones a given user has liked in the past, whereas recommendations based on the user-based CF approach identifies a group of users S whose preferences represented by their ratings are similar to those of the given user u and suggests books to u that are likely appealing to u based on the ratings of S.

3.2.1 The Content-Based Filtering Methods

The content-based filtering approach recommend items to a user that are *similar* to the ones that the user prefers in the past. The approach can be adopted for identifying the common characteristics of books being liked by user u and recommend to u new books that share these characteristics. The *similarity of books* is computed by using the designated features applicable to the books to be compared. For example, if u offers very high ratings on books in the domain of adventure or a particular author, then the content-based filtering approach suggests other books to u based on the same domain, i.e., adventure, or author. In this section, we propose two different content-based filtering approaches, using either the Vector Space model, denoted *CB-VSM*, or the Naïve Bayes model, denoted *CB-NBM*. The detailed design of these two content-based filtering approaches are presented below.

The Content-Based Filtering Approach Using the Vector Space Model (CB-VSM). The content-based filtering method analyzes the descriptions of books rated by an adult u and construct a profile of u based on the descriptions which are used for predicting the ratings of books unknown to u. This method does not require the *ratings* on books given by other users as in the CF approach to predict ratings on unknown books to u. A user profile is a vector representation of the corresponding user u's interests, which can be constructed using Equation 1.

$$X_u = \Sigma_{i \in \tau_u} r_{u,i} X_i \tag{1}$$

where τ_u is the set of books rated by user u, $r_{u,i}$ denotes the rating provided by user u on book i, and X_i is the vector representation of the book description D on i with the *weight* of each keyword k in D computed by using the *term frequency (TF)* and *inverse document frequency (IDF)* of k.

The Cosine Similarity, denoted *CSim*, of the vector space model (VSM) is used to predict the *rating* of a book B unknown to user u using the profile P of u, denoted $CSim(P, B)$, as defined in Equation 2. The profile representation of P is computed using Equation 1, whereas the vector representation of the description of B is determined similarly as X_i in Equation 1, and the *weight* of each keyword k in the description of B is calculated by using the TF/IDF of k.

$$CSim(P, B) = \frac{\Sigma_{i=1}^{t}(P_i \times B_i)}{\sqrt{\Sigma_{i=1}^{t} P_i^2 \times \Sigma_{i=1}^{t} B_i^2}} \tag{2}$$

where t is the dimension of the vector representation of P and B.

The Content-Based Filtering Approach Using the Naïve Bayes Model (CB-NBM). Besides using the vector space model for content-based filtering, machine-learning techniques have also been widely used in inducing content-based profiles. In using the machine-learning approach for text (which can be adopted for profile) classification, an inductive process automatically constructs a text classifier by learning from a set of training documents, which have already been labeled with the corresponding categories. Indeed, learning to classify user profiles can be treated as a binary text categorization problem, i.e., each item is classified as either interesting or not interesting with respect to the user preferences as specified by the attributes in a user profile. In this section, we discuss the Naïve Bayes classifier, which is a widely-used machine-learning algorithm in content-based filtering approach. Even though a constraint imposed on using the machine-learning approach for content-based filtering is that items must be labeled with their respective classes during the training process, after the classifier has been trained, it can be used to automatically infer profiles based on the trained model.

The Naïve Bayes model is a probabilistic approach to inductive learning. The probabilistic classifier developed by the Naïve Bayes model is based on the *Bayes' rule* as defined below.

$$\begin{aligned} P(C|D) &= \frac{P(D|C) \times P(C)}{P(D)} \\ &= \frac{P(D|C) \times P(C)}{\sum_{c \in C} P(D|C=c) P(C=c)} \end{aligned} \tag{3}$$

where C (D, respectively) is a random variable corresponding to a class (document[4], respectively).

Based on the term, which is either an attribute or a keyword in an item, independence assumption, the Naïve Bayes rule yields

$$\begin{aligned} P(c|d) &= \frac{P(d|c) \times P(c)}{\sum_{c \in C} P(d|c) P(c)} \\ &= \frac{\prod_{i=1}^{n} P(w_i|c) \times P(c)}{\sum_{c \in C} \prod_{i=1}^{n} P(w_i|c) \times P(c)} \end{aligned} \tag{4}$$

where w_i $(1 \leq i \leq n)$ is a term in d, and $\sum_{c \in C} \prod_{i=1}^{n} P(w_i|c) \times P(c)$ is a *chain rule.*

The classification process is based on the following computation:

$$\begin{aligned} Class(d) &= arg\ max_{c \in C} P(c|d) \\ &= arg\ max_{c \in C} \frac{P(d|c) \times P(c)}{\sum_{c \in C} P(d|c) \times P(c)} \end{aligned} \tag{5}$$

where $P(d|c)$ is the *probability* that d is observed, given that the class is known to be c, and $P(c)$ is the *probability* of observing class c, which is defined as

$$P(c) = \frac{N_c}{N} \tag{6}$$

[4]From now on, unless stated otherwise, a document is treated as an item, such as a book.

where N_c is the number of training items in class c, and N is the total number of training items.

In estimating $P(d|c)$ in Equation 5, the *Multiple-Bernoulli model* is applied, since the Multiple-Bernoulli distribution is a natural way to model the distributions over binary vectors. $P(d|c)$ is computed in the *Multiple-Bernoulli model* as

$$P(d|c) = \prod_{w \in V} P(w|c)^{\delta(w,d)} (1 - P(w|c))^{1-\delta(w,d)} \quad (7)$$

where $\delta(w,d)$ = 1 if and only if term w occurs in d.

Note that $P(d|c)$ = 0 if there exists a $w \in d$ that never occurs in c in the training set, which is the *data sparseness* problem (Clarke and Wheaton, 2007) and can be solved by using the *Laplacian smoothing* estimate (Osher et al., 2022) as defined below.

$$P(w|c) = \frac{df_{w,c} + 1}{N_c + 1} \quad (8)$$

where $df_{w,c}$ denotes the number of items in c which includes term w, and N_c is the number of items belonged to the class c.

In designing BkRec, we have provided the option to adopt either the *Vector Space Model* or the *Naïve Bayes Model*, since the former is simply an algebraic model for representing text documents as vectors of identifiers, whereas the latter soley relies on a trained model using a pre-defined labeled item set. Thus, we offer the user the choice between an algebaric model or a machine learning model to be considered in developing BkRec or other book recommender systems.

3.2.2 The Collaborative Filtering Approaches

The collaborative filtering (CF) approaches rely on the ratings of a user and other users. The predicted rating of a user u on a book i is likely similar to the rating of user v on i if both users have rated other books similarly. In this paper, we consider both the *user-based CF* approach, denoted *UB-CF*, and the *item-based CF* approach, denoted *IB-CF*.

The User-Based CF Approach. The user-based CF approach determines the interest of a user u on a book i using the ratings on i by other users, i.e., the *neighbors* of u who have similar rating patterns (Zhao and Shang, 2010). We apply the Cosine Similarity measure as defined in Equation 9 to calculate the similarity between two users u and v and determine user pairs which have the lowest rating difference among all the users. Equation 9 can be applied to compute the *similarity* between a user and each one of the other users to find out the *similarity group* of each user. Two users who have the lowest difference rating value between them means that they are the *closest neighbors*.

$$USim(u,v) = \frac{\Sigma_{s \in S_{u,v}} (r_{u,s} \times r_{v,s})}{\sqrt{\Sigma_{s \in S_{u,v}} r_{u,s}^2} \times \sqrt{\Sigma_{s \in S_{u,v}} r_{v,s}^2}} \quad (9)$$

where $r_{u,s}$ ($r_{v,s}$, respectively) denotes the rating of user u (user v, respectively) on item s, and $S_{u,v}$ denotes the set of books rated by both users u and v.

Upon determining the K-nearest neighbors (i.e., KNN) of a user u using Equation 9, we can compute the predicted rating on a book s for u using Equation 10.

$$\hat{r_{u,s}} = \bar{r_u} + \frac{\Sigma_{v \in S_{u,v}} (r_{v,s} - \bar{r_u}) \times USim(u,v)}{\Sigma_{v \in S_{u,v}} USim(u,v)} \quad (10)$$

where $\hat{r_{u,s}}$ stands for the predicted rating on book s for user u, $\bar{r_u}$ is the average rating on books provided by user u, $S_{u,v}$ is the group of closest neighbors of u, $r_{v,s}$ is the rating of user v on book s, and $USim(u,v)$ is the similarity measure between users u and v as computed in Equation 9.

The Item-Based CF Approach. Contrast to the user-based CF approach which relies on similar user groups to recommend books, the item-based CF approach computes the similarity values among different books and determine sets of books with similar ratings provided by different users. The item-based CF approach predicts the rating of a book i for a user u based on the ratings of u on books similar to i. The *adjusted cosine similarity matrix* (Wang et al., 2006) is applied to compute the similarity values among different books and assign books with similar ratings into the same similar-item group as defined in Equation 11.

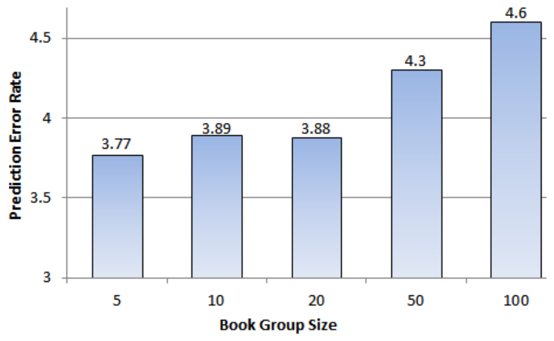
$$ISim(i,j) = \frac{\Sigma_{u \in U} (r_{u,i} - \bar{r_u}) \times (r_{u,j} - \bar{r_u})}{\sqrt{\Sigma_{u \in U} (r_{u,i} - \bar{r_u})^2} \times \sqrt{\Sigma_{u \in U} (r_{u,j} - \bar{r_u})^2}} \quad (11)$$

where $ISim(i,j)$ denotes the similarity value between books i and j, $r_{u,i}$ ($r_{u,j}$, respectively) denotes the rating of user u on book i (j, respectively), $\bar{r_u}$ is the average rating for user i on all books u has rated, and U is the set of books rated by u.

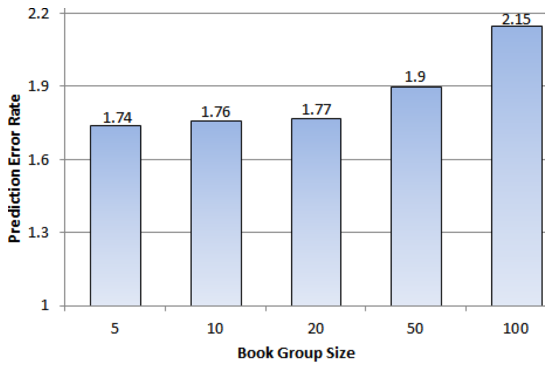
Equation 11 is used to compute the similarity between two books, whereas Equation 12 predicts the rating for user u on book i.

$$S_{u,i} = \bar{r_u} + \frac{\Sigma_u (r_{u,i} - \bar{r_u})}{u(i) + r} + \frac{\Sigma_{j \in I(u)} (r_{u,j} - \frac{\Sigma_u (r_{u,j} - \bar{r_u})}{u(j)})}{ISim(i,j) + r} \quad (12)$$

where $S_{u,i}$ denotes the predicted rating on book i for user u, $\bar{r_u}$ denotes the *average rating* on all books u



(a) Errors using only 70% book descriptions



(b) Errors using 100% book descriptions

Figure 1: Prediction errors of the content-based approach on the books in the BookCrossing dataset.

has rated, $r_{u,i}$ ($r_{u,j}$, respectively) is the rating on book i (j, respectively) provided by u, $I(u)$ is the set of books u has rated, $u(i)$ ($u(j)$, respectively) is the number of users who has rated i (j, respectively), and r is the book rating which is used to decrease the extremeness when there are not enough ratings available.

The 1^{st} component on the right-hand side of Equation 12 is called the *global mean* which is the average rating on all the books u has rated. The 2^{nd} component is called the *item offset* which is the score for user u on book i, whereas the 3^{rd} component is called the *user offset*, i.e., the user prediction on book i.

4 EXPERIMENTAL RESULTS

In this section, we first introduce the datasets used for the empirical study conducted to assess the performance of BkRec, our personalized book recommender for adult readers. Hereafter, we present the results of the empirical studies on BkRec.

4.1 Datasets

We have chosen a number of book records included in the BookCrossing dataset to conduct the performance evaluation of our recommender[5]. The BookCrossing dataset was collected by Cai-Nicolas Ziegler (Ziegler et al., 2005) between August and September of 2004 with data extracted from BookCrossing.com. It includes 278,858 users who provide, on the scale of 1 to 10, 1,149,780 ratings on 271,379 books. Each book record includes a user_ID, the ISBN of a book, and the rating provided by the user (identified by user_ID) on the book. We used Amazon.com AWS advisement API to verify that the ISBNs from the BookCrossing dataset are valid. The 271,379 books in the BookCrossing dataset is denoted BKC_DS.

Besides using the books and their ratings in the BookCrossing dataset, we extracted the book description of 30% of the BookCrossing books from Amazon.com, since they were missing in the books and needed for our content-based filtering approach. The Amazon dataset yields the additional dataset used for evaluating the performance of BkRec. Figure 1 shows the differences in terms of prediction errors using only 70% versus 100% of book descriptions generated by the content-based filtering approach.

4.2 Accuracy of Our RNN Classifier

Using a 80/10/10% training/validation/test split of the data as mentioned in Section 4.1, we achieved 79% classification accuracy on the book dataset. The accuracy could not be higher likely because of the high amounts of overlap between distinct keywords in the brief description of books with different categories, such as "Deep Learning Computing" and "Theory of computation". With 79% accuracy, we still successfully classify 4 out of 5 book records, which is way above the baseline "best-guesser" classifier. Other bag-of-words modeling techniques with which we have experimented, i.e., logistic regression, Support Vector Machine (SVM), and Multinomial Naïve Bayes (Ricci et al., 2011), showed lower results.

4.3 Performance Evaluation of BkRec

In our empirical study, we considered the similar user group size of *ten* users in the user-based CF approach, since LensKit[6], which has implemented the

[5]Other datasets can be considered as long as they contain user_IDs, book ISBNs, and ratings.

[6]http://lenskit.org/

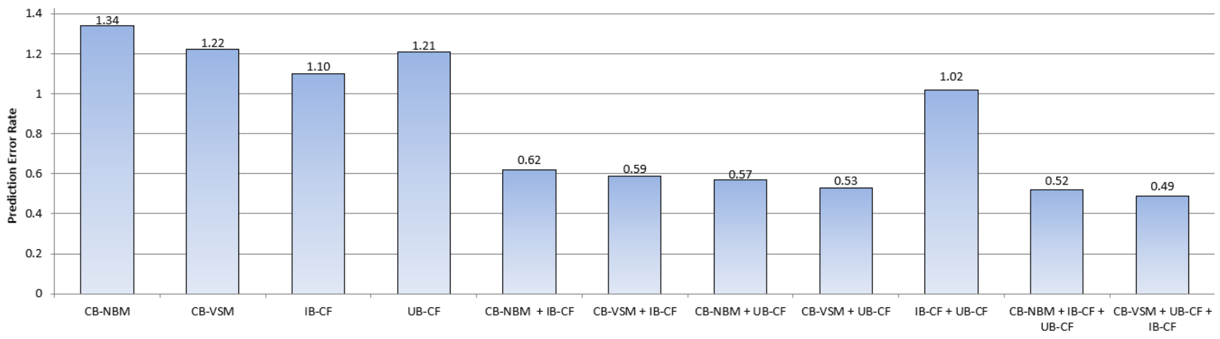


Figure 2: Prediction error rates of the various filtering approaches and their combinations, where CB, IB, and UB denote the Content-Based, Item-Based, and User-Based models, respectively.

user-based CF method and has been cited in a number of published papers, has demonstrated that ten is an ideal group size in predicting user ratings. We have also chosen *ten* to be the group size of books used in the content-based and item-based CF approaches, since the prediction error rate using this group side is the most ideal, in terms of size and accuracy, as demonstrated in our empirical study and reported in Figure 1.

To evaluate the performance of BkRec, we computed the prediction error of BkRec for each user U in CBC_DS by taking the absolute value of the difference between the real and predicted ratings on each book U has rated in the dataset. These prediction errors were added up and divided it by the total number of predictions. Figure 2 shows the prediction error rate of each filtering approaches and their combinations.

As shown in Figure 2, the combined content-based using VSM, item-based, and user-based CF approach, denoted CIU, outperforms individual and other combined prediction models in terms of obtaining the lowest prediction error rates among all the models[7]. CIU achieves the highest prediction accuracy, which is only *half* a rating (out of 10) away from the actual rating, since the item-based and content-based filtering approach compensates the user-based CF approach when user ratings are *sparse* and vice versa. The prediction error rate, i.e., accuracy ratio, achieved by CIU is statistically significant ($p < 0.04$) over the ones based on the combined content-based and item-based CF approach and the combination of the content-based and user-based approaches, the next two models with prediction error rate lower than *one*, which are determined using the Wilcoxon test signed-ranked test (Rey and Neuhäuser, 2011). The experimental results have verified that CIU is the most accurate recommendation tool in predicting ratings on books for adults, which is the most suitable choice for making book recommendations for adults based on the rating prediction.

[7]From now on, whenever we refer to *BkRec*, our book recommender, we mean the recommender system based on CIU.

4.4 Comparing Book Recommender Systems

In this section, we compared our recommender with exiting book recommenders that achieve high accuracy in recommendations on books based on their respective model.

- **MF**. Yu et al. (Yu et al., 2009) and Singh et al. (Singh and Gordon, 2008) predict ratings on books and movies based on matrix factorization (MF), which can be adopted for solving large-scale collaborative filtering problems. Yu et al. develop a non-parametric matrix factorization (NPMF) method, which exploits data sparsity effectively and achieves predicted rankings on items comparable to or even superior than the performance of the state-of-the-art low-rank matrix factorization methods. Singh et al. introduce a collective matrix factorization (CMF) approach based on relational learning, which predicts user ratings on items based on the items' genres and role players, which are treated as unknown values of a relation between entities of a certain item using a given database of entities and observed relations among entities. Singh et al. propose different stochastic optimization methods to handle and work efficiently on large and sparse data sets with relational schemes. They have demonstrated that their model is practical to process relational domains with hundreds of thousands of entities.
- **ML**. Besides the matrix factorization methods, probabilistic frameworks have been introduced for rating predictions. Shi et al. (Shi et al., 2010) propose a joint matrix factorization model for making

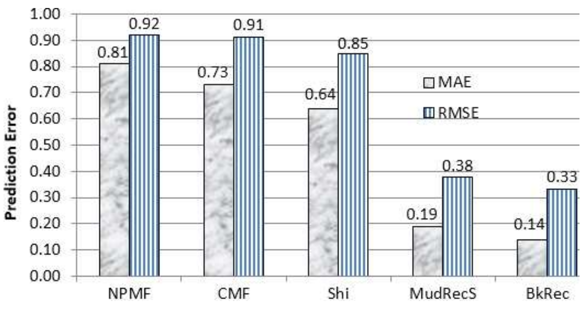


Figure 3: The MAE and RMSE scores for various book recommendation systems based on BKC_DS, the BookCrossing dataset.

context-aware item recommendations[8]. Similar to ours, the matrix factorization model developed by Shi et al. relies not only on factorizing the user-item rating matrix but also considers contextual information of items. The model is capable of learning from user-item matrix, as in conventional collaborative filtering model, and simultaneously uses contextual information during the recommendation process. However, a significant difference between Shi et al.'s matrix factorization model and ours is that the contextual information of the former is based on mood, whereas ours makes recommendations according to the contextual information on books.

- MudRecS (Qumsiyeh and Ng, 2012) makes recommendations on books, movies, music, and paintings similar in content to other books, movies, music, and paintings, respectively that a MudRecS user is interested in. MudRecS does not rely on users' access patterns/histories, connection information extracted from social networking sites, collaborated filtering methods, or user personal attributes (such as gender and age) to perform the recommendation task. It simply considers the users' ratings, genres, role players (authors or artists), and reviews of different multimedia items. MudRecS predicts the *ratings* of multimedia items that match the interests of a user to make recommendations.

Figure 3 shows the *Mean Absolute Error* (MAE) and *Root Mean Square Error* (RMSE) scores of our and other recommender systems on the BKC_DS dataset. MAE and RMSE are two performance metrics widely-used for evaluating rating predictions on multimedia data. Both MAE and RMSE measure the *average magnitude* of *error*, i.e., the average prediction error, on incorrectly assigned ratings. The error values computed by MAE are *linear* scores, i.e., the absolute values of individual differences in incorrect assignments are weighted equally in the average, whereas the error rates of RMSE are squared before they are summed and averaged, which yield a relatively *high* weight to errors of *large* magnitude.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f(x_i) - y_i| \quad (13)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(f(x_i) - y_i)^2}{n}} \quad (14)$$

where n is the total number of items with ratings to be evaluated, $f(x_i)$ is the rating predicted by a system on item x_i ($1 \leq i \leq n$), and y_i is an expert-assigned rating to x_i.

As the MAE and RMSE scores shown in Figure 3, our book recommender significantly outperforms other book recommender systems on rating predictions of the respective books based on the Wilcoxon Signed-Ranks Test ($p \leq 0.05$).

4.5 An Online Evaluation on the Performance of BkRec

Besides conducting an offline evaluation, we also perform an online evaluation to verify the novelty of *BkRec* in recommending books to adults. The online evaluation determines whether its suggestions are perceived as preferable by ordinary users, i.e., adults, which offers another perspective on the performance of the recommender. This additional evaluation is based on real users' assessments of the book recommender which goes beyond the offline performance analysis conducted and presented in previous subsections. We first determine the ideal number of appraisers and test cases extracted from the BookCrossing dataset for the evaluation of *BkRec* so that they are *reliable* and *objective*.

4.5.1 The Number of Appraisers

In statistics, two types of errors, Types I and II, are defined (Jones and Kenward, 2003). Type I errors, also known as α errors, or *false positives*, are the *mistakes* of *rejecting* a null hypothesis when it is true, whereas Type II errors, also known as β errors, or *false negatives*, are the *mistakes* of *accepting* a null hypothesis when it is false. We apply the formula in (Jones and Kenward, 2003) below to determine the ideal number of appraisers, n, which is dictated by the probabilities of occurrence of Types I and II errors, to evaluate the performance of *BkRec* online.

$$n = \frac{(Z_{\frac{\alpha}{2}} + Z_{\beta})^2 \times 2\sigma^2}{\triangle^2} + \frac{(Z_{\frac{\alpha}{2}})^2}{2} \quad (15)$$

[8]The system was originally designed to predict ratings on *movies* but was implemented by Qumsiyeh and Ng (Qumsiyeh and Ng, 2012) for additional comparisons on *books* as well.

where $\triangle$ is the *minimal expected difference* to compare books extracted from the constructed dataset by our recommendation approach with manually-chosen books, which is set to 1 in our study, since we expect our recommendation approach to suggest books to users is as good as the ones chosen manually; σ^2 is the *variance* of the extracted books and is set to be 1.95 in our study. It is computed by averaging the sum of the square difference between the mean and the actual number of useful suggested created for each one of the 100 test cases (extracted from BookCrossing database for verifying σ^2), which are computed on a *simple random sample* and do not change with a larger sample set of test cases; α (β, respectively) denotes the probability of making a Type I (II, respectively) error, which is set to be 0.05 (0.20, respectively), and 1 - β determines the probability of a false null hypothesis that is correctly rejected, and Z is the value assigned to the standard *normal distribution* of extracted books. Based on the standard normal distribution, when α = 0.05, $Z_{\frac{\alpha}{2}}$ = 1.87, and when β = 0.20, Z_β = 0.85. When α = 0.05 and β = 0.20, they imply that we have 95% *confidence* on the correctness of our analysis and that the probability of avoiding false negatives/positives) of our statistical study is 80%. According to (Kazmier, 2003), 0.05 is the commonly-used value for α, whereas 0.80 is a conventional value for 1 - β, and a test with β = 0.20 is considered to be statistically powerful. Based on these assigned values, the ideal number of appraisers used for our study is

$$n = \frac{(1.87+0.85)^2 \times 2 \times 1.95}{1^2} + \frac{1.87^2}{2} \cong 30 \quad (16)$$

The results collected from the 30 appraisers are expected to be comparable with the results that are obtained by the actual population (Jones and Kenward, 2003), i.e., ordinary adult book readers. We recruited 30 students, who are either graduate-level or undergraduate-level students in STEM major, from our university to serve as appraisers of the study.

4.5.2 The Number of Offline Test Cases

To determine the ideal number of test cases to be used in the controlled experiments, we rely on two different variables: (i) the *average attention span* of an adult and (ii) the *average number of search queries* that a person often creates in one session when using a web search engine. As mentioned in (Rozakis, 2002), the average attention span of an adult is between 40 to 60 minutes. Moreover, Jansen et al. (Jansen et al., 2000), who have evaluated web users' behavior especially on (i) the amount of time web users spend on a web search engine, and (ii) the average number of queries submitted by a user, estimate that the average number of queries created by each user in one session on a web search engine is 5. Based on these studies, each appraiser was asked to evaluate our recommender using *five* test cases, since evaluating the recommender system on the retrieved results, i.e., 5 books, of each one of the five test cases takes approximately 60 minutes, which falls into an adult time span. Moreover, each appraiser contributed three separated hours for the offline evaluation. We randomly selected *450* (= 30 × 5 × 3) test cases from the BookCrossing dataset for the performance evaluation.

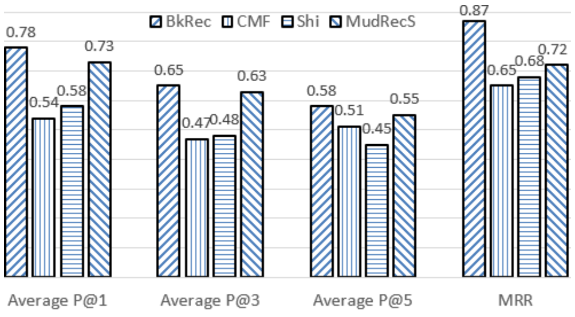


Figure 4: The $P@1$, $P@3$, $P@5$, and MRR values for the online evaluation of various book recommenders systems and *BkRec*.

4.5.3 Online Performance Evaluation

After the gold standard for each test case provided by each one of the 30 appraisers were determined, we computed the average $P@1$, $P@3$, $P@5$, and *MRR* values for the three existing recommender systems, i.e., *CMF*, Shi, and MudRec, and *BkRec*, our book recommender based on the content-based model using NBM, which outperforms VSM in terms of content-based filtering, that are involved in our online empirical study. Figure 4 shows the performance metrics for the average $P@1$, $P@3$, $P@5$, in addition to *MRR*, which is the average of the reciprocal ranks at which the *first* correct answer (among the top-5 ranked answers retrieved by a book recommender system) for each test case is made. The $P@1$, $P@3$, $P@5$, and MRR scores of *BkRec* are higher than the corresponding ones of the three book recommender systems, and the results are statistically significant based on the Wilcoxon Signed-Rank test ($p < 0.035$).

5 CONCLUSIONS

We can all agree that reading cannot happen without focus and in order to fully understand the story, we have to concentrate on each page that we read. In a world where gadgets are only getting faster and shortening our attention span, we need to constantly practice concentration and focus. Reading is one of the few activities that requires your undivided attention, therefore, improving your ability to concentrate. While reading, we also have to remember different characters and settings that belong to a given story. Even if one enjoys reading a book in one sitting, (s)he has to remember the details throughout the time (s)he take to read the book. Therefore, reading is a workout for your brain that improves memory function. Even though we recognize the benefits of reading aside from leisure and education, searching for a book of interest and entertaining is not an easy task due to the fact that there are hugh amount of books available to choose from these days.

It is important to develop a book recommender system for adults that lightens their burden to search for desirable books to read for either educational or entertaining purpose, or both. Unlike many of the existing book recommender systems, our proposed book recommender for adults, *BkRec*, simply relies solely on a deep learning model and different filtering approaches to make personalized book recommendations. BkRec is simple and unique, since it relies only on user ratings on books to determine books preferred by adults. It is also easy to interpret any book recommended by BkRec, which is simply based on the common interests of other users who share the same passion to a particular adult.

The empirical study conducted to evaluate the performance of BkRec demonstrates that our recommender outperforms well-known book recommenders for adults, and the results are statistically significant.

REFERENCES

Alharthi, H., Inkpen, D., and Szpakowicz, S. (2018). A Survey of Book Recommender Systems. *Intelligent Information Systems*, 51:139–160.

Anis, M., Kavitha, V., Juby, M., and Sam, G. (2021). Survey Paper on Book Recommendation System and Comparative Study on Types of Algorithms used. *International Journal of Creative Research Thoughts (IJCRT)*, 9(6):120–126.

Baron, N. and Mangen, A. (2021). Doing the Reading: the Decline of Long Long-Form Reading in Higher Education. *Poetics Today*, 42(2):253–279.

Basmo (2023). 15 Statistics About Reading Books You Must Know in 2023. https://basmo.app/reading-statistics/#:∼:text=Now%20that%20we%20know%20that,couple%20of%20things%20in%20common.

Clarke, P. and Wheaton, B. (2007). Addressing Data Sparseness in Contextual Population Research: Using Cluster Analysis to Create Synthetic Neighborhoods. *Sociological Methods & Research*, 35(3):311–351.

Delgado, P., Vargas, C., Ackerman, R., and Salmeron, L. (2018). Don't Throw Away your Printed Books: A Meta-Analysis on the Effects of Reading Media on Reading Comprehension. *Educational Research Review*, 25:23–38.

Desrosiers, C. and Karypis, G. (2011). *Recommender Systems Handbook*, chapter A Comprehensive Survey of Neighborhood-based Recommendation Methods, pages 107–144. Springer.

Devika, P., Jisha, R., and Sajeev, G. (2016). A Novel Approach for Book Recommendation Systems. In *Proceedings of the 2016 IEEE international Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–6. IEEE.

Gupta, S., Jain, Y., Laad, S., Khandelwal, S., and Saklecha, P. (2020). A Survey Report on Book Recommendation Techniques. *Computer Engineering & Information Technology*, 9(4):1–20.

Jansen, B., Spink, A., and Saracevic, T. (2000). Real Life, Real Users, and Real Needs: a Study and Analysis of User Queries on the Web. *IPM*, 36(2):207–227.

Jones, B. and Kenward, M. (2003). *Design and Analysis of Cross-Over Trials, 2nd Ed.* Chapman and Hall.

Kazmier, L. (2003). *Schaum's Outline of Business Statistics*. McGraw-Hill.

Mathew, P., Kuriakose, B., and Hegde, V. (2016). Book Recommendation System through Content Based and Collaborative Filtering Method. In *Proceedings of the 2016 International Conference on Data Mining and Advanced Computing*, pages 47–52. IEEE.

Narvaez, D., Broek, P. V. D., and Ruiz, A. (1999). The Influence of Reading Purpose on Inference Generation and Comprehension in Reading. *Educational Psychology*, 91(3):488.

National Library (2023). Reading for Pleasure – a Door to Success. https://natlib.govt.nz/schools/reading-engagement/understanding-reading-engagement/reading-for-pleasure-a-door-to-success.

Osher, S., Wang, B., Yin, P., Luo, X., Barekat, F., Pham, M., and Lin, A. (2022). Laplacian Smoothing Gradient Descent. *Research in the Mathematical Sciences*, 9(3):55.

Qumsiyeh, R. and Ng, Y. (2012). Predicting the Ratings of Multimedia Items for Making Personalized Recommendations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–484.

Ramakrishnan, G., Saicharan, V., Chandrasekaran, K., Rathnamma, M., and Ramana, V. (2020). Collaborative Filtering for Book Recommendation System. In *Soft Computing for Problem Solving (SocProS), Volume 2*, pages 325–338. Springer.

Rayner, K., Pollatsek, A., Ashby, J., and Clifton, C. (2012). *Psychology of Reading, 2nd Ed.* Psychology Press.

Rey, D. and Neuhäuser, M. (2011). Wilcoxon-signed-rank test. In *International Encyclopedia of Statistical Science*, pages 1658–1659. Springer.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. (2011). *Recommender Systems Handbook*. Springer.

Rozakis, L. (2002). *Test Taking Strategies and Study Skills for the Utterly Confused*. McGraw-Hill.

Saraswat, M. and Srishti (2022). Leveraging Genre Classification with RNN for Book Recommendation. *Information Technology*, 14(7):3751–3756.

Shi, Y., Larson, M., and Hanjalic, A. (2010). Mining Mood-Specific Movie Similarity with Matrix Factorization for Context-Aware Recommendation. In *Workshop on CARS*, pages 34–40.

Singh, A. and Gordon, G. (2008). Relational Learning via Collective Matrix Factorization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 650–658.

Wang, J., de Vries, A., and Reinders, M. (2006). Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 501–508.

Yu, K., Zhu, S., Lafferty, J., and Gong, Y. (2009). Fast Nonparametric Matrix Factorization for Large-Scale Collaborative Filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 211–218.

Zhang, F. (2016). A Personalized Time-Sequence-Based Book Recommendation Algorithm for Digital Libraries. *IEEE Access*, 4:2714–2720.

Zhao, Z. and Shang, M. (2010). User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM KDD)*, pages 478–481.

Ziegler, C., McNee, S., Konstan, J., and Lausen, G. (2005). Improving Recommen-dation Lists Through Topic Diversification. In *Proceedings of the World Wide Web*, pages 22–32.