applied sciences

MDPI

Article

Feature Extracted Deep Neural Collaborative Filtering for E-Book Service Recommendations

Ji-Yoon Kim [1] and Chae-Kwan Lim [2,*]

[1] Contents AI Research Center, Romantique, 27 Daeyeong-ro, Busan 49227, Republic of Korea; rlawldbs2918@nate.com
[2] Department of Distribution and Logistics, Tongmyong University, Busan 48520, Republic of Korea
* Correspondence: cklim@tu.ac.kr

Abstract: The electronic publication market is growing along with the electronic commerce market. Electronic publishing companies use recommendation systems to increase sales to recommend various services to consumers. However, due to data sparsity, the recommendation systems have low accuracy. Also, previous deep neural collaborative filtering models utilize various variables of datasets such as user information, author information, and book information, and these models have the disadvantage of requiring significant computing resources and training time for their training. To address this issue, we propose a deep neural collaborative filtering model with feature extraction that uses minimal data such as user number, book number, and rating information. The proposed model comprises an input layer for inputting and embedding the product and user data, a feature extraction layer for extracting the features through data correlation analysis between the embedded user and product data, a multilayer perceptron, and an output layer. To improve the performance of the proposed model, Bayesian optimization was used to determine hyperparameters. To evaluate the deep neural collaborative filtering model with feature extraction, a comparative analysis experiment was conducted with currently used collaborative filtering models. The goodbooks-10k public dataset was used, and the results of the experiment show that the low accuracy caused by data sparsity was considerably improved.

Keywords: collaborative filtering; electronic publishing; feature extraction; recommender systems

check for updates

Citation: Kim, J.-Y.; Lim, C.-K. Feature Extracted Deep Neural Collaborative Filtering for E-Book Service Recommendations. *Appl. Sci.* **2023**, *13*, 6833. https://doi.org/10.3390/app13116833

Academic Editor: Luis Javier Garcia Villalba

Received: 7 March 2023
Revised: 30 May 2023
Accepted: 2 June 2023
Published: 5 June 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The number of Internet users constantly increases, and the use of the electronic commerce market is growing accordingly [1]. As a result, the electronic publication market, based on Internet technologies, is growing. General consumers use various platforms to purchase electronic books (e-books), and students use e-books for learning. Moreover, as recent studies have found that e-books can help students learn, student e-book use is expected to increase [2]. Therefore, a customized recommendation system that can help increase e-book sales is essential for the survival of companies.

Due to technological developments, online sales markets have grown, and corporations and researchers have studied various recommendation algorithms [3–7]. The collaborative filtering (CF) system, commonly used in recommendation systems, analyzes the correlation between the user and the product and recommends new products based on past experiences and behavior. Based on the rating of a certain product, CF uses the item-based CF to search for products similar to the corresponding product; the information of the user and that of users that purchased and assessed similar products constitutes user-based CF. The user-based CF predicts the user rating of a product that the user has never purchased [8,9]. However, a limitation of CF is that it consumes considerable hardware resources to execute the recommendation system. Therefore, studies on recommendation systems based on matrix decomposition have been conducted to reduce resource use. These

propose efficient resource models by defining latent vector features between the user and the product. However, the matrix decomposition-based recommendation system has a data sparsity problem [10].

Therefore, to enhance the existing recommendation systems, recent studies have investigated deep-learning-based recommendation systems. The user–product data in the universal recommendation system are multidimensional data composed of numbers or texts [11]. Previous studies have used deep learning with multidimensional data to study regression, image and voice classification, and natural language processing and have demonstrated that deep learning outperforms previous algorithms [12]. In particular, the recommendation systems are characterized by deep-learning embedding [13] and multilayer perceptron [14] for processing the user and product data. The embedding transforms the input data into vectors to compute the similarity between the data. In addition, multilayer perceptron can analyze nonlinear data by using a hidden layer. Various deep-learning-based recommendation systems have been proposed based on these models. However, models that apply efficient product–user data feature extraction based on the nature of the deep-learning model are still lacking. Furthermore, presented deep learning-based recommendation systems have exploited diverse types of data such as user, product, and creator information to improve the accuracy of the models. However, various data require more computing resources, training times to train the model, and considerable time to optimize the parameters.

To resolve the abovementioned problems, this paper proposes deep neural CF with feature extraction for e-book service recommendations. The model utilizes minimal data, such as user information numbers, book numbers and ratings, and it is composed of four parts. First is the data input, which includes the user data and the input layer that embeds the product data. Second, the feature extraction layer analyses the correlation between the embedded user and product data. The last two are the multilayer perceptron layer and the output layer. To measure the model's accuracy, the root mean square error (RMSE) and mean absolute error (MAE) were chosen. In addition, Bayesian optimization was used to find the optimized model parameters, which can reduce parameter searching and testing time. The contributions of this study are summarized as follows:

- A deep neural CF model with feature extraction for e-book service recommendations was designed by reflecting the user's and product's features as much as possible.
- Bayesian optimization was used to optimize the model parameters, and the proposed model for e-book service recommendations was used with the optimized model parameters and activation function.
- A comparative analysis experiment with other CF models was conducted to assess the performance of the proposed model, and the results showed that the proposed model outperformed the comparison models.

The remainder of the paper is structured as follows: Section 2 introduces the relevant research. Then, Section 3 presents the proposed model, and Section 4 the optimization process. Finally, Section 5 discusses the results, and Section 6 offers the conclusion.

2. Related Works

2.1. Recommendation Systems

Deep learning is continuously researched and thus is used in various sectors. For example, neural CF, a deep-learning-based recommendation model, was proposed in [15]. The model was used to interpret the complex relationship between the user and item data; the complex relationship is a disadvantage of the matrix factorization model [15]. In addition, the CF method (ConvNCF) uses an interactive map with various 2-dimensional convolution neural networks (CNNs) and an embedding layer to demonstrate that the deep-learning-based recommendation models outperformed the conventional models at the time [16]. ConvMF embedded CNNs to generate recommendations with user review data [17]. It was also used with continuous features, such as the user's age, app installation, and session, and categorical features, such as the mobile device and demographics of the

user. The corresponding model is accurate and expandable, and the input data features were improved [18]. In a novel personalized long short-term memory-based matrix factorization approach for online quality-of-service prediction, the latent features were extracted to represent many users and services. In addition, the newly input data were used to update the model [19] progressively. The deep hybrid CF approach for the service recommendation model used the similarity of the text information in the multilayer perceptron [20] and learned the nonlinear relationships of the services and mashups of the web service that retain complex call relationships [21]. In [22], an image-based service recommendation system was used to extract the features from JPEG-type image data, and a model that used a CNN and random forest ensemble algorithm was proposed. The dual-embedding-based deep latent factor model was proposed to obtain implicit feedback. The existing embedding of the recommendation system used the user- and item-embedding layers, and models that apply the user–item interaction into four types of embedding have also been proposed [23].

2.2. Feature Extraction Systems

In [24], a convolutional graph network was used to learn features from knowledge graphs instead of embeddings to improve the performance of the recommendation system. In addition, to enhance the convolutional neural networks gated recurrent units were used. In [25], the feature of personalized preference information was extracted, and the structure of the deep-learning model that could explicate the user's decision-making was proposed. A feature learning process that overlooks the past interaction information and a model that applies attention was also presented. Chen et al. [26] proposed a heterogeneous information network model that uses deep learning. As a feature, rich secondary data containing the user review data and user rating were used; the secondary data were presented to the models that train with the features of entities through the network embedding process. Zeng et al. [27] proposed a deep-learning model for the recommendation system that uses the co-occurrence embedding structure with the rating, user, and item metrics to analyze the correlation between the user and item data [27]. Finally, in [28], the fusion recommendation model that utilizes the item rating and user review data was presented. One limitation of the traditional fusion recommendation model is its complexity; thus, a subnetwork that separated the user review data and rating data was used to reduce the complexity.

Aside from the traditional deep-learning-based recommendation system structure that uses the multilayer perceptron, an immediate item was recommended using the fused features. Huang et al. [29] proposed a neural explicit factor model to increase the explainability of the recommendation systems of the existing CF models that use user and item data. The explainability of the item and user vectors was increased by applying the user-feature attention matrix and an n-item-feature quality matrix to the user–item rating matrix. Additionally, using a 1-dimensional CNN and feedforward neural network, the features from the item-feature vector, user, and item were extracted [29].

The proposed model is structurally closest to the deep neural CF model with feature extraction, but the deep neural CF model with feature extraction underwent the feature selection process of the user items, and the residual connection of the item data and the embedded users is performed for the feature map data that rearrange the selected features. Therefore, the role of the model is different, and the number of parameters used in the computation of deep learning shows a significant difference.

3. Proposed Model

This section describes the deep neural model with feature extraction, and each part is described in detail. Figure 1 represents the structure of the deep neural CF with feature extraction composed of four layers: input, feature extraction, multilayer perceptron, and output. The input layer receives user data and book data. Concatenated embeddings perform concatenation to enable user data and book data to be processed in the feature extraction layer. The feature extraction layer utilizes user-book data to analyze the correlations between the data and output essential features. The multi-layer perceptron learns

the nonlinear relationships among the extracted data. Finally, the output layer outputs the predicted evaluation value of a specific book for a specific user.

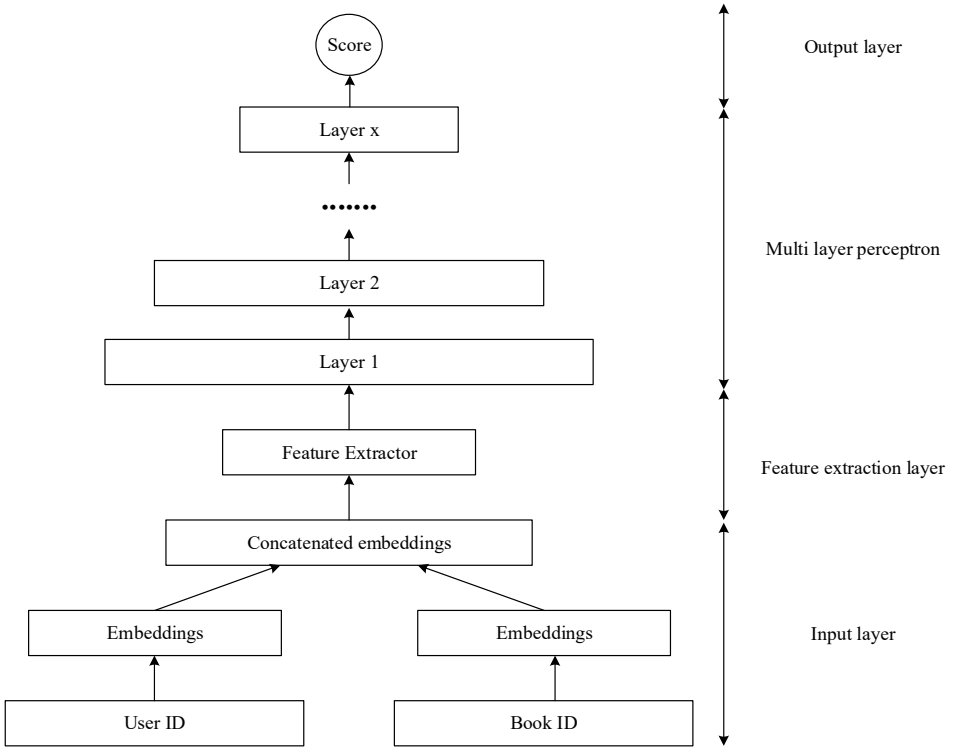


Figure 1. Structure of the feature extracted deep neural collaborative filtering.

3.1. Input Layer

In the input layer, the user and book data are input individually. An embedding process using one-hot encoding is performed for the input data, and the sparsely input user and book data can be densely expressed. The embedding method used in natural language processing is applied. The latent user vector is expressed as $X_u = \{x_u^1, x_u^2, \cdots x_u^n\}$. In addition, the n in x_u^n indicates the features of the nth user. The latent book vector is expressed as $X_b = \{x_b^1, x_b^2, \cdots x_b^m\}$. Here, m in x_b^m represents the user features of the mth user. Each embedded user and book data instance are concatenated so that their features can be extracted in the feature extraction layer. The concatenation can be expressed as Equation (1).

$$z = concatenate(X_u, X_b) = \begin{bmatrix} X_u \\ X_b \end{bmatrix} \quad (1)$$

where z represents data that underwent concatenation.

3.2. Feature Extraction Layer

The residual block [30], used in image processing, is utilized as the feature extraction layer. Furthermore, the feature extraction layer includes a novel process in which only salient features that affect the correlation of the user–book data can be selected. The feature extraction layer includes feature selection operation, feature rearrangement operation and residual connection.

- Feature selection operation

The feature selection operation extracts the input data features and selects only the data necessary to determine the user–book data correlation. Then, the convolution layer is used to analyze the data features. The convolution layer has kernels to create feature maps

from input data. The features extracted by each kernel in the 1D-convolution layer [31] are expressed by Equation (2).

$$Convolution\ layer\ (z) = z \cdot k(z) \tag{2}$$

Here, k denotes a 1-dimensional input vector, k refers to the number of convolution kernels and $\cdot$ is the convolution operator. Calculated convoluted features from l^{th} can be expressed by Equation (3).

$$x_i^l = f\left(\sum_j x_j^{l-1} \cdot k_{ij}^l + b_i^l\right) \tag{3}$$

Here, x_j^i denotes i^{th} feature data of l^{th} layer, f is the activation function, x_j^{l-1} represents j^{th} feature data of the previous layer, k_{ij}^l indicates the related kernel between i^{th} and j^{th} features of l^{th} layer. Finally, b_i^l refers to the bias values of k_{ij}^l features.

To prevent overfitting of extracted feature data and for convenience of convolutional layer interpretation, the corresponding data are input after average pooling [32]. Lastly, the batch normalization [33] process is performed to prevent the overfitting and the destruction of the convolutional layer and the slope of the data that underwent average pooling.

Computations of feature data after global average pooling and batch normalization and input data are required; therefore, the feature data must undergo a feature rearrangement operation to have data maps identical to those of the input data.

- Feature rearrangement operation

The feature rearrangement operation involves two fully connected layers. The first layer uses the rectified linear unit (*ReLU*) activation function [34] to compute the selected feature data. The *ReLU* activation function can be expressed as Equation (4).

$$ReLU(x) = \begin{cases} x, if\ x > 0 \\ 0, if\ x \leq 0 \end{cases} \tag{4}$$

The fully connected layer output value performs computations with the input data of the feature extraction layer. Therefore, selecting an activation function that can deliver accurate feature results of the user and item data is essential. We use the scaled exponential linear unit (*SeLU*) activation function [35] in the fully connected layer. The *SeLU* activation function possesses internal normalization to resolve the gradient loss of the exponential linear unit activation function [36]. Moreover, an advantage of *SeLU* is the conservation of the average and variance from the previous layer. The *SeLU* activation function can be expressed as Equation (5).

$$SeLU(x) = \lambda \begin{cases} x, if\ x > 0 \\ \alpha e^x - \alpha, if\ x \leq 0 \end{cases} \tag{5}$$

Here, α and λ as previously designated constants. The function is characterized by iterative multiplication by constant λ to maintain the input data variance and average. In addition, if the input data point is identical or below 0, the variance is reduced to prevent gradient loss.

- Residual connection

The value output of the *SeLU* activation function residually connects [37]—the feature extraction layer input data. Therefore, the residual connection can be expressed as Equation (6).

$$rc = F(x) + x \tag{6}$$

Here, $F(x)$ refers to the function that performs the feature selection and rearrangement operations, and x represents the input data of the feature extraction layer. Figure 2 shows the feature extraction layer.

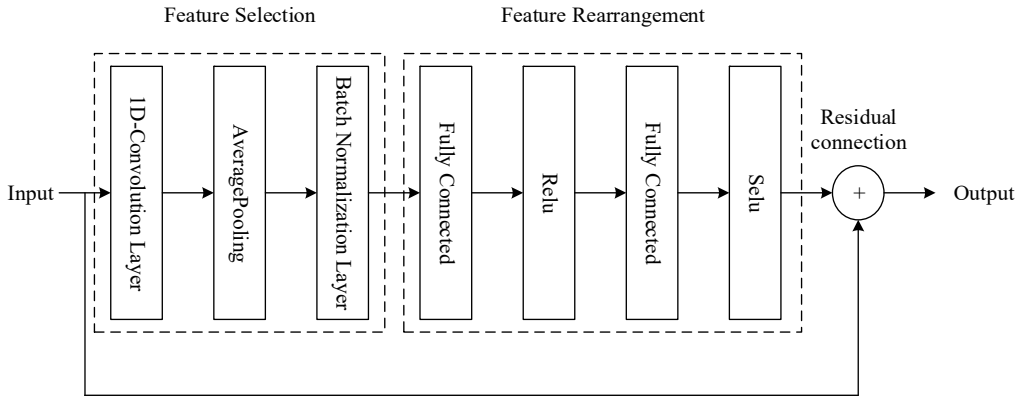


Figure 2. Feature extraction layer.

3.3. Multi-Layer Perceptron

The multilayer perceptron is advantageous for its ability to visualize nonlinear data relationships. The first layer of the multilayer perceptron receives an *rc* of data after feature extraction, and the computation is performed after the data passes through hidden layers. The multilayer perceptron is expressed as follows:

$$\begin{aligned} \theta_1 &= ReLU\left(W_1^T rc + b_1\right) \\ &\cdots \\ \theta_n &= ReLU\left(W_n^T \theta_{n-1} + b_n\right) \end{aligned} \tag{7}$$

Here, θ_1 refers to the first layer. *ReLU* is the activation function of the multilayer perceptron: W and b refer to the weight and bias, respectively.

3.4. Output Layer

The output layer outputs a certain user's predicted rating of a certain book. The output layer has one output, expressed as Equation (8).

$$\hat{y}_{u,b} = ReLU\left(W_n^T x + b_n\right) \tag{8}$$

Here, $\hat{y}_{u,b}$ refers to the predicted value and includes the user and book information used for the prediction. Additionally, *ReLU* refers to the *ReLU* activation function used in the output layer.

4. Parameter Search Using Bayesian Optimization

A deep-learning model is composed of and dependent on many parameters, such as the number of neurons, learning rate, convolution layer, filter, and kernel numbers. Therefore, deep-learning model performance depends on the selection of suitable hyperparameters. The hyperparameters of the proposed model are related to the data used in the model training [38]. This section presents the parameter selection process using Bayesian optimization [39].

4.1. Bayesian Optimization Model

Grid search and random search are techniques used in existing deep-learning model hyperparameter tuning, and they have an internal problem in finding the optimal hyperparameters [40,41]. However, Bayesian optimization evaluates and estimates based on probability models [42]. The search for the optimal hyperparameter of the proposed model involved finding the minima of test and validation losses. Let the search space be P, the size of the embedding layer of the internal hyperparameters N_e, the method of normalizing

the embedding N_r, and the size of the filter of the 1D convolution layer N_f. Then, the object function F can be expressed as Equation (9).

$$F : P\left(N_e, N_r, N_f, \ldots N_n\right) \subset \mathbb{R}^n \rightarrow \mathbb{R} \tag{9}$$

where P is the search space and p^* is the hyperparameters of the optimized model. In addition, p^* can be expressed as Equation (10).

$$p^* = \underset{p \in P}{argmin} F(p) \tag{10}$$

The observations of the object function can be expressed $D_{1:n} = \left(p_{1:n}, F_{p1:n}\right)$. The observation area used to create the probability model $F(p)$, which helps find the next position of P in the sample Bayes optimization, calculates the posterior distribution of the objective function based on Bayes' theorem. Then, the next combination of hyperparameters is selected from the calculated posterior distribution. In addition, the former sampling point information is used to ascertain the shape of the object function and find hyperparameters that can output the maximum function for the black-box model.

4.2. Optimizer Selection

Optimizer selection is a significant factor in deep learning model-making. For example, in the adaptive moment estimation, Adam, the optimizer, understands the data, requires little memory, and exhibits high computing efficiency. Therefore, Adam was selected as the optimizer for training the deep neural CF with feature extraction [43].

4.3. Selection of Target Search Parameters

For the selection of the optimal hyperparameters of the proposed model, the hyperparameter search space was defined, and the hyperparameter search results were identified using the convergence plot. Finally, the optimal hyperparameter values of the proposed model, which were derived using Bayesian optimization, were visualized.

Adam was chosen as the optimizer for the proposed model, and the number of layers in the multilayer perceptron was set to four. The input layer, feature extraction, and multilayer perceptron hyperparameter search spaces were arranged as shown in Table 1.

Table 1. Hyperparameter search space for the deep neural CF model with feature extraction.

Kind of Data	Search Space
Size of embedding in the input layer	10–100
Regularization method type in the embedding input layer	L1, L2, L1L2
Number of 1D convolution layer filters of feature extraction layer	4–258
Size of 1D convolution layer kernel of feature extraction layer	4–258
Number of neurons in the first fully connected layer of feature extraction layer	4–258
Activation function type in the second fully connected layer	Sigmoid, SoftMax, ReLU, tanh, *SeLU*, ELU
Number of neurons in the first fully connected layer of the multilayer perceptron	4–258
Number of neurons in the second fully connected layer of the multilayer perceptron	4–258
Number of neurons in the third fully connected layer of the multilayer perceptron	4–258
Number of neurons in the fourth fully connected layer of the multilayer perceptron	4–258
Type of loss function	Huber, MeanAbsoluteError, MeanSquaredError
Value of learning rate	0.001–0.000001

The evaluation metrics for determining the hyperparameters were chosen as the functions used in the model for performance assessment: *RMSE* and *MAE*.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{11}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \quad (12)$$

5. Results and Discussion

5.1. Dataset

The open user–book dataset goodbooks-10k was used. These data were publicized by Zygmunt [44], and the dataset is accessible through the GitHub website (https://github.com/zygmuntz/goodbooks-10k, (accessed on 1 March 2023)). This dataset includes 53,424 unique users, 10,000 unique book data points, and 5,976,479 ratings. The minimum value of the rating is 1, and the maximum is 5. No additional data preprocessing was performed. The dataset includes user id, book id and rating as shown in Table 2.

Table 2. Dataset of goodbooks-10k.

Attribute	Range
User Id	1–53,424
Book Id	1–10,000
Rating	1–5

5.2. Evaluataion Metrics

To assess the CF model performance, RMSE and MAE were selected as evaluation metrics. RMSE is defined in (10) and MAE in (11). RMSE uses the square function and imposes a penalty when the residual value is significant. MAE ascertains the absolute average of the error by comparing the predicted and actual values to identify their difference on the absolute scale.

5.3. Model Performance Comparision

The commonly used machine- and deep-learning-based CF models and the proposed model were comparatively analyzed. For comparison of all models, the goodbooks-10k dataset was used, and all recommendation models were evaluated using the same evaluation metrics. The descriptions of the selected models are as follows.

ALS: A system that uses the matrix factorization technique and square loss function [45].

SVD: A commonly used recommendation system with matrix factorization [46].

Fast AI Embedding Dot Bias: Fast AI is an upper wrapper of PyTorch made by Jeremy Howard, and it supports the CF function using deep learning. The default model parameters were used in the performance comparison [47].

A simple algorithm for recommendation (SAR): The SAR defines the similarity based on the co-occurrence of data in all input data. In addition, it measures the strength of the relationship between the items that have interacted previously in the data. In the performance comparison, the default model parameters were used [48].

For a fair comparison, the Recommenders, an open-source library of models published by Microsoft, was used to pool all compared models. The corresponding open-source library was in GitHub. Additionally, the proposed model used pandas, NumPy, and TensorFlow libraries.

For the comparative analysis experiment, the dataset was split randomly [49] to obtain a training set of 80%, a validation set of 10%, and a test set of 10% of the data. The training dataset was used in model training, and the validation dataset was used to tune the hyperparameters.

5.4. Selected Parameters of the Model

The hyperparameter optimization for the proposed model was performed using Bayesian optimization, and Figure 3 shows the convergence plot. In the Bayesian optimization algorithm, it can be seen that the minimum value of the loss function was reached at

the 13th iteration. Table 3 shows the importance of the optimal hyperparameters of the proposed model obtained using Bayesian optimization.

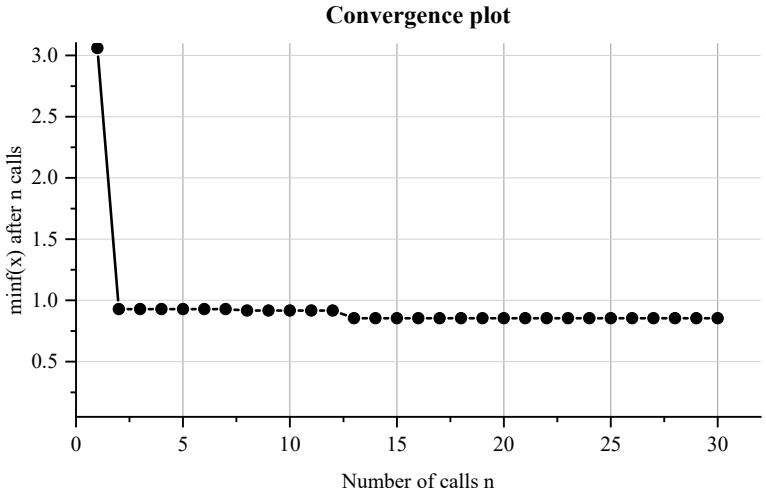


Figure 3. Convergence plot.

Table 3. Selected hyperparameters for the deep neural CF model with feature extraction.

Kind of Data	Search Space
Size of embedding in the input layer	40
Regularization method type in the embedding input layer	L1
Number of 1D convolution layer filters of feature extraction layer	108
Size of 1D convolution layer kernel of feature extraction layer	24
Number of neurons in the first fully connected layer of feature extraction layer	14
Activation function type in the second fully connected layer	*SeLU*
Number of neurons in the first fully connected layer of the multilayer perceptron	60
Number of neurons in the second fully connected layer of the multilayer perceptron	216
Number of neurons in the third fully connected layer of the multilayer perceptron	258
Number of neurons in the fourth fully connected layer of the multilayer perceptron	210
Type of loss function	MeanAbsoluteError
Value of learning rate	0.000495

5.5. Experimental Result and Analysis

All models were trained five times to evaluate the four comparison models and the proposed model. The evaluation functions were RMSE and MAE; the experimental results are shown in Table 4.

Table 4. Performance comparison of the models.

Trial No.	ALS		SVD		Fast AI Embedding Dot Bias		SAR		Feature Extracted Deep Neural CF	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	0.9641	0.7307	0.8560	0.6683	0.9711	0.7781	1.6783	1.4239	0.8424	0.6593
2	0.9686	0.7325	0.8555	0.6684	0.9656	0.7722	1.6755	1.4190	0.8418	0.6587
3	0.9651	0.7309	0.8548	0.6673	0.9652	0.7719	1.6683	1.4138	0.8429	0.6598
4	0.9685	0.7340	0.8565	0.6689	0.9655	0.7722	1.6657	1.4119	0.8424	0.6594
5	0.9683	0.7324	0.8558	0.6681	0.9653	0.7720	1.6682	1.4149	0.8428	0.6597

The best training results of each model were analyzed. The ALS model obtained RMSE of 0.9641 and MAE of 0.7307, better than the similar deep-learning model FAST AI Embedding Dot Bias. The SVD model obtained 0.8448 as RMSE and 0.6673 as MAE, the

highest values among all models excluding the proposed model. The FAST AI Embedding Dot Bias model obtained 0.9652 and 0.7719 as RMSE and MAE values, respectively, and outperformed the SAR model but not the ALS and SVD models. The SAR model obtained 1.6657 as RMSE and 1.4119 as MAE, which was the worst performance among the models used for comparison. Finally, the proposed model obtained 0.8418 as RMSE and 0.6587 as MAE, which means it outperformed all the other models.

The experimental results and analysis show that the proposed model, deep neural CF with feature extraction, obtained the lowest RMSE and MAE values. In particular, the proposed model performed better than the traditionally used matrix-factorization-based CF models. Therefore, it is possible to conclude that the proposed model best analyzed the relationship between user and book data.

6. Conclusions

This paper proposes a novel form of deep neural CF model with feature extraction that can be used by electronic publication service companies. The proposed model analyzed the relationship between the user data and book data using the feature extraction layer. The predicted rating values of the book and user data were output by the multilayer perceptron. Bayes optimization was used to find the optimized parameter values, and a comparative analysis was performed with commonly used models such as ALS, SVD, Fast Ai embedding dot bias and SAR. The results of the comparative analysis showed that the proposed model achieved the best performance. Also, we could recognize that the feature extraction layer has sufficient capacity to analyze features of the data. However, the ability to improve model performance is limited. Thus, as a follow-up study, we will research complex and deep feature extraction layers to enhance the deep neural CF model.

Author Contributions: Conceptualization, Methodology, and Software, J.-Y.K.; Project administration, Funding acquisition, C.-K.L.; All authors have read and agreed to the published version of the manuscript.

Funding: This study results from a study on the "Leaders in Industry-university Cooperation 3.0 Project", supported by the Ministry of Education and National Research Foundation of Korea (202201330001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Internet Usage Statistics. Available online: https://www.internetworldstats.com/stats.htm (accessed on 5 January 2023).
2. Sari, S.Y.; Rahim, F.R.; Sundari, P.D.; Aulia, F. The importance of e-books in improving students' skills in physics learning in the 21st century: A literature review. *J. Phys. Conf. Ser.* **2022**, *2309*, 012061. [CrossRef]
3. Mobasher, B.; Burke, R.; Bhaumik, R.; Williams, C. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans.* **2007**, *7*, 23-es. [CrossRef]
4. Zhen, L.; Huang, G.Q.; Jiang, Z. An inner-enterprise knowledge recommender system. *Expert Syst. Appl.* **2010**, *37*, 1703–1712. [CrossRef]
5. Guy, I.; Carmel, D. Social recommender systems. In Proceedings of the WWW '11: 20th International World Wide Web Conference, Hyderabad, India, 28 March–1 April 2011; pp. 283–384. [CrossRef]
6. Verma, C.; Hart, M.; Bhatkar, S.; Parker-Wood, A.; Dey, S. Improving scalability of personalized recommendation systems for enterprise knowledge workers. *IEEE Access* **2016**, *4*, 204–215. [CrossRef]
7. Ricci, F.; Rokach, L.; Shapira, B. *Recommender Systems: Introduction and Challenge*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 1–34. [CrossRef]
8. Sarwar, B.; Karypis, G.; Konstan, J.; Reidl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web (WWW), Hong Kong, China, 1–5 May 2001. [CrossRef]
9. Zhao, A.D.; Shang, M.S. User-based collaborative-filtering recommendation algorithms on Hadoop. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, Phuket, Thailand, 9–10 January 2010. [CrossRef]

10. Hu, R.; Pu, P. Enhancing collaborative filtering systems with personality information. In Proceedings of the RecSys '11: Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011. [CrossRef]
11. Elahi, M.; Ricci, F.; Rubens, N. A survey of active learning in collaborative recommender systems. *Comput. Sci. Rev.* **2016**, *20*, 29–50. [CrossRef]
12. Zhang, Q.; Yang, L.T.; Chen, Z.; Li, P. A survey on deep learning for big data. *Inf. Fusion* **2018**, *42*, 146–157. [CrossRef]
13. Barkan, O.; Koenigstein, N. ITEM2VEC: Neural item embedding for collaborative filtering. In Proceedings of the 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing, Vietri sul Mare, Italy, 13–16 September 2016. [CrossRef]
14. Zheng, L. A Survey and Critique of Deep Learning on Recommender Systems. 2016. Available online: https://bdsc.lab.uic.edu/docs/survey-critique-deep.pdf (accessed on 6 January 2023).
15. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. *arXiv* **2017**, arXiv:1708.05031. [CrossRef]
16. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Outer product-based neural collaborative filtering. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 9–19 July 2018.
17. Kim, D.; Park, C.; Oh, J.; Lee, S.; Yu, H. Convolutional matrix factorization for document context-aware recommendation. In Proceedings of the 10th ACM Conference, Boston, MA, USA, 15–19 September 2016. [CrossRef]
18. Cheng, H.T. Wide & deep learning for recommender systems. *arXiv* **2016**, arXiv:1606.07792. [CrossRef]
19. Xiong, R.; Wang, J.; Li, Z.; Li, B.; Hung, P.C.K. Personalized LSTM based matrix factorization for online QoS prediction. In Proceedings of the 2018 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, 2–7 July 2018. [CrossRef]
20. Taud, H.; Mas, J. Multilayer perceptron (MLP). In *Geomatic Approaches for Modeling Land Change Scenarios, Lecture Notes in Geoinformation and Cartography*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 451–455. [CrossRef]
21. Xiong, R.; Wang, J.; Zhang, N.; Ma, Y. Deep hybrid collaborative filtering for Web service recommendation. *Expert Syst. Appl.* **2018**, *110*, 191–205. [CrossRef]
22. Ullah, F.; Zhang, B.; Khan, R.U. Image-based service recommendation system: A JPEG-coefficient RFs approach. *IEEE Access* **2020**, *8*, 3308–3318. [CrossRef]
23. Cheng, W.; Shen, Y.; Huang, L.; Zu, Y. Dual-embedding based deep latent factor models for recommendation. *ACM Trans. Knowl. Discov. Data* **2021**, *15*, 85. [CrossRef]
24. Lin, Y.; Du, S.; Zhang, Y.; Duan, K.; Huang, Q.; An, P. A recommendation strategy integrating higher-order feature interactions with knowledge graphs. *IEEE Access* **2022**, *10*, 119290–119300. [CrossRef]
25. Maneechote, N.; Maneeroj, S. Explainable recommendation via personalized features on dynamic preference interactions. *IEEE Access* **2022**, *10*, 116326–116343. [CrossRef]
26. Chen, S.; Tian, J.; Tian, X.; Liu, S. Fusing user reviews into heterogeneous information network recommendation model. *IEEE Access* **2022**, *10*, 63672–63683. [CrossRef]
27. Zeng, W.; Qin, J.; Wei, C. Neural collaborative autoencoder for recommendation with co-occurrence embedding. *IEEE Access* **2021**, *9*, 163316–163324. [CrossRef]
28. Wang, H.; Hong, M.; Hong, Z. Research on BP neural network recommendation model fusing user reviews and ratings. *IEEE Access* **2021**, *9*, 86728–86738. [CrossRef]
29. Huang, H.; Luo, S.; Tian, X.; Yang, S.; Zhang, X. Neural explicit factor model based on item features for recommendation systems. *IEEE Access* **2021**, *9*, 58448–58454. [CrossRef]
30. He, K.; Zhang, X.; Ren, S.; Sun, j. Deep residual learning for image recognition. *arXiv* **2015**, arXiv:1512.03385. [CrossRef]
31. Ha, S.; Choi, S. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016. [CrossRef]
32. Kasagi, A.; Tabaru, T.; Tamura, H. Fast algorithm using summed area tables with unified layer performing convolution and average pooling. In Proceedings of the IEEE Workshop on Machine Learning for Signal Processing, Tokyo, Japan, 25–28 September 2017; pp. 1–6. [CrossRef]
33. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, Lille, France, 6–11 July 2015. [CrossRef]
34. Agarap, A.F. Deep learning using rectified linear units. *arXiv* **2018**, arXiv:1803.08375. [CrossRef]
35. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-normalizing neural networks. In Proceedings of the Self-Normalizing Neural Networks, NIPS'17: 31st International Conference on Neural Information Processing Systems, Montreal, ON, Canada, 3–8 September 2018. [CrossRef]
36. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and accurate, deep network learning by exponential linear units (ELUs). In Proceedings of the ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016. [CrossRef]
37. Jastrzebski, S.; Arpit, D.; Ballas, N.; Verma, V.; Che, T.; Bentio, Y. Residual connections encourage iterative inference. In Proceedings of the ICLR 2018 Conference, Vancouver, ON, Canada, 16 February 2018. [CrossRef]
38. Zhou, Y.; Cahya, S.; Combs, T.A.; Nicolaou, C.A.; Wang, J.; Desai, P.V.; Shen, J. Exploring tunable hyperparameters for deep neural networks with industrial ADME data sets. *Am. Chem. Soc.* **2019**, *59*, 1005–1016. [CrossRef]
39. Frazier, P.I. A tutorial on Bayesian optimization. *arXiv* **2018**, arXiv:1807.02811. [CrossRef]
40. Candelieri, A.; Archetti, I.; Author, F.; Barkalov, K. Turning hyperparameters of a SVM-based water demand forecasting system through parallel global optimization. *Comput. Oper. Res.* **2019**, *106*, 202–209. [CrossRef]

41. Thiede, L.A.; Parlitz, U. Gradient based hyperparameter optimization in Echo State Networks. *Neural Netw.* **2019**, *115*, 23–29. [CrossRef] [PubMed]
42. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; Freitas, N.D. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* **2016**, *104*, 148–175. [CrossRef]
43. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference for Learning Representations, San Diego, CA, USA, 7–9 May 2015. [CrossRef]
44. Goodbooks-10k. Available online: https://github.com/zygmuntz/goodbooks-10k (accessed on 5 January 2023).
45. Ghosh, S.; Nahar, N.; Wahab, M.A.; Biswas, M. Recommendation system for e-commerce using alternating leat squares (ALS) on Apache Spark. In *Intelligent Computing and Optimization, Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2021. [CrossRef]
46. Author, M.G.; Margaritis, K.G. Applying SVD on generalized item-based filtering. In Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), 8–10 September 2005.
47. Howard, J.; Gugger, S. Fastai: A layered API for deep learning. *Information* **2020**, *11*, 108. [CrossRef]
48. Aggarwal, C.C. *Recommender Systems: The Textbook*; Springer: New York, NY, USA, 2016.
49. Meng, Z.; McCreadie, R.; Macdonald, C.; Ounis, I. Exploring data splitting strategies for the evaluation of recommendation models. In Proceedings of the RecSys '20: 14th ACM Conference on Recommender Systems, New York, NY, USA, 22–26 September 2020. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.