Knowledge-Based Systems 187 (2020) 104811

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A novel deep multi-criteria collaborative filtering model for recommendation system☆

Nour Nassar*, Assef Jafar, Yasser Rahhal

Higher Institute for Applied Sciences and Technology, Damascus, Syria

ARTICLE INFO

Article history:
Received 28 December 2018
Received in revised form 9 May 2019
Accepted 24 June 2019
Available online 25 June 2019

Keywords:
Collaborative filtering
Deep learning
Deep neural network
Multi-criteria
Recommender system

ABSTRACT

Recommender systems have been in existence everywhere with most of them using single ratings in prediction. However, multi-criteria predictions have been proved to be more accurate. Recommender systems have many techniques; collaborative filtering is one of the most commonly used.

Deep learning has achieved impressive results in many domains such as text, voice, and computer vision. Lately, deep learning for recommender systems began to gain massive interest, and many recommendation models based on deep learning have been proposed. However, as far as we know, there is not yet any study which gathers multi-criteria recommendation and collaborative filtering with deep learning. In this work, we propose a novel multi-criteria collaborative filtering model based on deep learning. Our model contains two parts: in the first part, the model obtains the users and items' features and uses them as an input to the criteria ratings deep neural network, which predicts the criteria ratings. Those criteria ratings constitute the input to the second part, which is the overall rating deep neural network and is used to predict the overall rating. Experiments on a real-world dataset demonstrate that our proposed model outperformed the other state-of-the-art methods, and this provides evidence pointing to the success of employing deep learning and multi-criteria in recommendation systems.

1. Introduction

Recommender systems (RSs) are software tools that make suggestions for items that might be of interest to a user. They can be seen everywhere: there are recommender systems for movies [1], music [2], tourism [3], books [4], news [5], research articles [6], and products in general, and they even have become an important component in websites such as Google, Amazon, Netflix, YouTube, and others [7]. RSs use many techniques such as content-based ones [8], collaborative filtering [8], and trust-based recommender systems [9,10]. Collaborative filtering techniques are the most commonly used; they do not need any previous knowledge about users or items, instead, they make recommendations based on interactions between them. Although they are efficient and simple, they suffer from a number of problems, like cold start [5], prediction accuracy [11] and inability of capturing complex interactions between the user and the item [12].

Most RSs use single ratings in predictions and this fact is considered as a limitation because when a user makes a choice he might take into consideration more than one aspect, so additional aspects may increase the accuracy of the prediction [13]. Thus, Multi-Criteria Recommender System (MCRS) that uses multi-criteria ratings can lead to recommendations, which may be more accurate; because it provides users the possibility to rate an item based on multiple criteria, and that means it would be able to represent more predilections of each user. For an example, in a music recommender system, some users may like music based on its rhythm, beat, or timbre, while others may like the same music but for its melody, tempo, texture, or any other combinations of the distinct attributes of that music.

In recent years, Deep Learning (DL) has achieved tremendous success in many fields such as computer vision, language processing, and speech recognition. Recently, deep learning for recommender systems began to gain enormous attention. Some researchers have proposed recommendation models based on deep learning, as it has proved capable of capturing complex relations [12], which led to improvements in the RS performance, but most of these models make use of single ratings in predictions.

In this paper, we propose a novel solution, which gathers multi-criteria recommendation and deep learning, in order to effectively improve the performance of collaborative filtering. Our model comprises two stages: in the first stage, we obtain the

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.06.019.

* Corresponding author.
E-mail addresses: nournassar123@gmail.com, nour.nassar@hiast.edu.sy (N. Nassar).

https://doi.org/10.1016/j.knosys.2019.06.019
0950-7051/

users and items' features and use them as an input to a deep neural network that predicts the criteria ratings, and in the second one, we use a deep neural network to learn the relationship between the overall rating and the criteria ratings.

By conducting experiments on a real-world dataset, we show that our proposed model achieves better results than other state-of-the-art methods.

The main contributions of this work are as follows:

1. We present a deep multi-criteria collaborative filtering model, which combines the strengths of both techniques, in order to improve the performance of collaborative filtering. To our knowledge, this is the first attempt to integrate deep learning and multi-criteria into collaborative filtering.
2. We do extensive experiments on a real-world dataset in order to show the effectiveness of our model and the prominence of using multi-criteria and deep learning in collaborative filtering.

The paper is organized as follows: Section 2 reviews the related work. Section 3 describes the system overview. Section 4 provides the results of our experiments and Section 5 includes the conclusion and possible future work.

2. Related work

Collaborative filtering can be divided into two general classes: neighborhood and model-based methods. Neighborhood-based methods use user–item ratings directly to make recommendations by calculating the similarities between users (user-based) [14] or between items (item-based) [15]. Model-based methods use these user–item ratings in order to learn a predictive model and later use this model to make recommendations. There are plenty of model-based methods, such as Latent Semantic Analysis (LSA) [16], Bayesian Clustering [17], Support Vector Machines (SVM) [18], Latent Dirichlet Allocation (LDA) [19], and Singular Value Decomposition (SVD) [20]. Among the various collaborative filtering methods, the Singular Value Decomposition (SVD) is the most popular one. This method maps both users and items to a joint latent factor space of the same dimension; the latent space represents the latent features of the users or items [8].

Multi-criteria recommendation techniques can be classified by their utility function into two categories: memory-based and model-based techniques. In memory-based techniques, the similarity can be computed from the multi-criteria ratings in two ways: the first approach aggregates traditional similarities values that are calculated separately on each criterion into a single similarity using one of the following aggregation methods (average [21], worst-case [21], and weighted sum of individual similarities [22]). The second approach calculates the distance between multi-criteria ratings directly using multidimensional distance metrics (Euclidean, Manhattan, and Chebyshev distance metrics) [21]. Model-based approaches build a predictive model in order to predict unknown ratings, such as the Aggregation-function-based approach [21] coming from the assumption that an item's overall rating is not independent of other ratings but, rather, there exists a relation between the overall rating and the multi-criteria ratings; Probabilistic Modeling [23,24], Support Vector Regression (SVR) [25], Multilinear Singular Value Decomposition (MSVD) [26], and Genetic Algorithm (GA) [27].

There are very few researches on applying deep learning to collaborative filtering model. Salakhutdinov et al. [28] used Restricted Boltzmann Machines (RBM's) to model the correlation between item ratings, then Georgiev et al. [29] extended the previous work by modeling both user–user and item–item correlations. Ouyang et al. [30] proposed an Autoencoder-based Collaborative Filtering (ACF) which use autoencoder to model users explicit ratings on items. Wu et al. [31] presented Collaborative Denoising AutoEncoder (CDAE) that utilizes the Denoising Autoencoders in Collaborative Filtering. He et al. [32] presented Neural Collaborative Filtering (NCF) framework that uses Deep Neural Network (DNN) in order to learn the nonlinear relationship between the users and items.

3. System overview

The proposed model follows the aggregation-function-based approach that Adomavicius et al. [21] presented, as shown in Fig. 1. It consists of three steps:

1. Predict unknown k multi-criteria ratings using any recommendation technique.
2. Learn aggregation function f using statistical or machine learning techniques.
3. Predict unknown overall ratings using f and predicted multi-criteria ratings.

In our model, we used deep neural networks to predict the k multi-criteria ratings in the first step, and to learn the aggregation function$f$ in the second step, as illustrated in Fig. 2.

3.1. Predict criteria ratings

In this step, we will predict the multi-criteria ratings, where we use a deep neural network to predict the criteria ratings for a user on an item. Our work concentrates on pure collaborative filtering, so we use only item ID or user ID as a feature. Still, with such a generic feature representation, our model can be simply modified to solve the cold start problem by using user and item content features. The ID is a categorical feature that does not have a logical order; therefore we used the feature representation method that He et al. [32] proposed in their NCF framework. We converted first the ID into a dense continuous valued and low dimensional vector called an embedding vector. The embedding vectors are initialized with random values, and then during the model training, the values are adjusted so that to minimize the loss function.

The criteria ratings DNN is shown in Fig. 3 where, in the input layer, the input vector x is given by:

$$x = Concatenate(v_u, v_i) \tag{1}$$

where v_u and v_i are user and item embedding vectors.

Then, followed by a number of hidden layers, we choose the Rectified Linear Units (ReLU) as the activation function (Eq. (2)) because it is the most efficient [33].

$$ReLU\ (z) = max\ (z, 0) \tag{2}$$

The output of a hidden layer l is formulated as:

$$h_l = ReLU\ (W_l h_{l-1} + b_l) \tag{3}$$

where W_l and b_l are the weight matrix and bias vector for the layer l and $h_1 = x$.

In the output layer, we predict the user criteria ratings $r_1, r_2, \ldots, r_k$ using Eqs. (4) and (5):

$$y_{ui} = ReLU\ (W_L h_{L-1} + b_L) \tag{4}$$

$$[r_1, r_2, \ldots, r_k]^T = y_{ui} \tag{5}$$

where L is the number of layers.

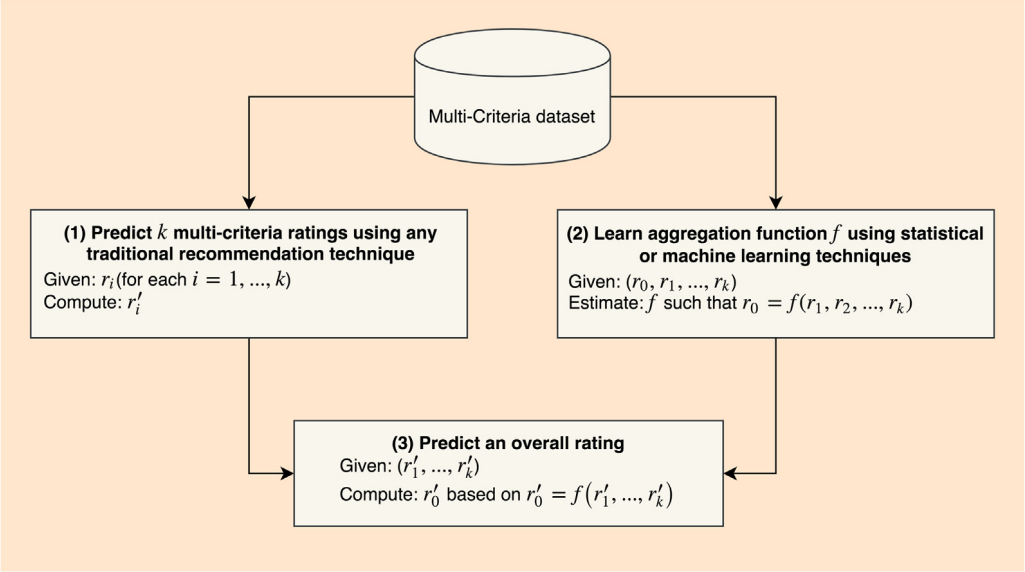

Fig. 1. The aggregation-function-based approach.

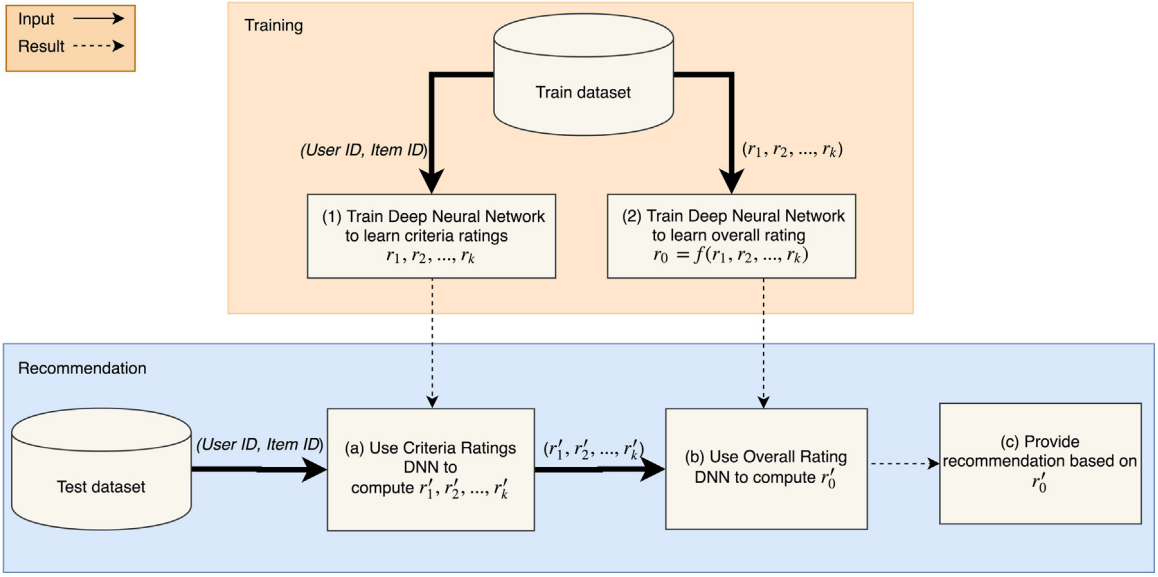

Fig. 2. The framework of the proposed model, including two stages: the Training and the Recommendation stage.

3.2. Learn aggregation function

In the aggregation-function-based approach, the general and intuitive assumption is that the overall rating can be estimated by an aggregation function of multi-criteria ratings. In this step, we aim to learn the relationship between the overall rating r_0 and the criteria ratings $r_1, r_2, \ldots, r_k$.

$$r_0 = f(r_1, r_2, \ldots, r_k) \tag{6}$$

and for this purpose, we shall use a DNN as it proved to be capable of estimating any function [34].

In the input layer, as shown in Fig. 3, the input vector is the criteria ratings $r_1, r_2, \ldots, r_k$ for user u and item i, and since the DNNs are sensitive to the inputs distribution and scaling [35], we normalize the continuous features$r_1, r_2, \ldots, r_k$. The normalization of a sample r_i is calculated as:

$$z_i = \frac{r_i - m}{s} \tag{7}$$

with m the mean of the training samples and $s$the standard deviation of the training samples. The input vector becomes:

$$x = [z_1, z_2, \ldots, z_k]^T \tag{8}$$

Then, followed by a number of dense ReLU layers, the output of the hidden layer is given again by Eq. (3).

In the output layer, we predict the overall rating r_0 in Eq. (4), where:

$$r_0 = y_{ui} \tag{9}$$

In both DNNs, we used MAE loss function and Adam optimizer [36].

3.3. Recommendation

After the completion of the model training, where the model parts were trained separately without knowing each other, we can now use the model to predict the user overall rating on the

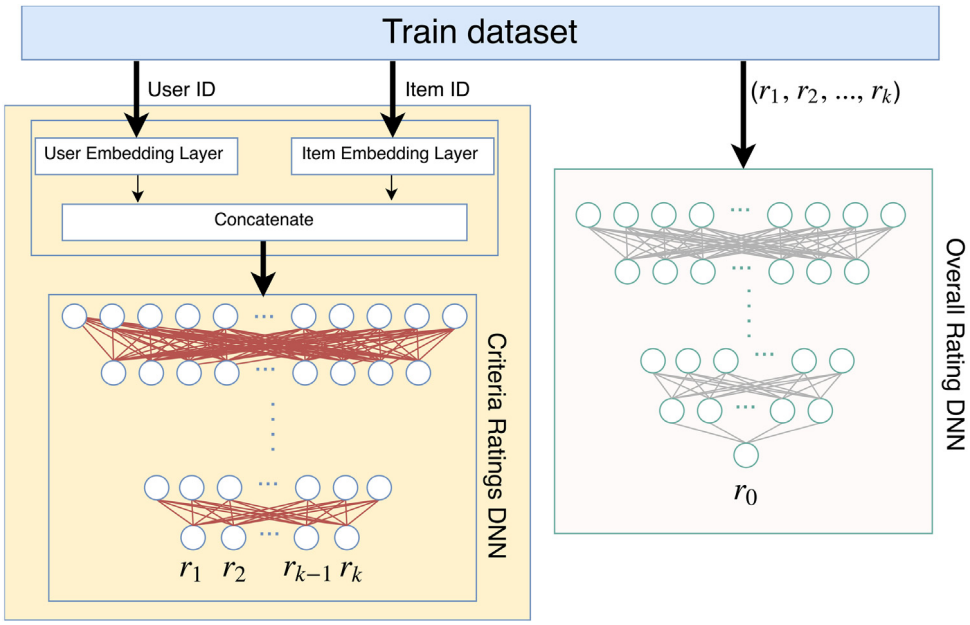

Fig. 3. The architecture of the two deep neural networks: Criteria Ratings DNN and Overall Rating DNN.

Table 1
TripAdvisor dataset statistics.

Field description	Value
Number of users	72,119
Number of hotels	1,850
Number of ratings	81,085
Data sparsity	99.94%

Table 2
Ratings distribution.

RATINGS	1	2	3	4	5
Value	4268	5853	10374	24696	35894
Rooms	4002	5517	10493	25019	36054
Location	1464	3141	8503	22177	45800
Cleanliness	2591	3585	8232	19951	46726
Check in / front desk	3775	4468	11083	19201	42558
Service	4086	4462	10036	20114	42387
Business service	4525	5849	19624	21661	29426
Overall	4246	6052	8078	24180	38529

The number of examples of the values for each criteria rating and for the overall rating in the dataset.

items that he has not rated yet. The recommendation process occurs as shown in Fig. 2:

Step (a): Compute criteria ratings. First, we get the user ID and item ID pairs and feed them as inputs to the Criteria Ratings DNN, and then we predict the criteria ratings $r'_1, r'_2, \ldots, r'_k$.

Step (b): Compute overall ratings. In this step, we normalize the criteria ratings $r'_1, r'_2, \ldots, r'_k$ computed in step (a), feed them as inputs to the Overall Rating DNN, and then predict the overall ratings r'_0.

Step (c): Provide recommendation. Finally, we recommend the optimal items to the user using the overall ratingr'_0 as in traditional single rating recommender systems.

4. Experiments and results

4.1. Dataset

We evaluated the performance of our model on the TripAdvisor[1] dataset, a multi-criteria rating dataset, which contains users' ratings for hotels. It includes seven criteria ratings (Value, Rooms, Location, Cleanliness, Check in/front desk, Service, and Business service) and an overall rating, the rating range between 1 and 5. The statistics of the dataset are shown in Table 1, and the distribution of values for the different aspects and for the global score are shown in Table 2.

[1] The dataset is crawled from the TripAdvisor website: https://www.tripadvisor.com/.

4.2. Evaluation

We used several metrics in order to evaluate the performance of our model:

(1) Mean Absolute Error (MAE)

$$MAE = \frac{1}{N}\sum_{u,i}\left|r_{ui} - r'_{ui}\right| \tag{10}$$

where N is the size of the test set, r'_{ui} is the predicted overall rating of user u for item i, and r_{ui} is the true overall rating.

(2) F-measure (F_β)

Given top K recommendations for each user:

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \tag{11}$$

$$precision = \frac{1}{U}\sum_{u}\frac{|\{relevant\ and\ recommended\ items\ in\ top\ K\}|}{|\{recommended\ items\ in\ top\ K\}|} \tag{12}$$

$$recall = \frac{1}{U}\sum_{u}\frac{|\{relevant\ and\ recommended\ items\ in\ top\ K\}|}{|\{relevant\ items\}|} \tag{13}$$

Table 3
Optimizers results.

Optimizer	MAE
Adadelta	0.9112 ± 0.0170
Adagrad	0.7565 ± 0.0216
Adam	**0.7552** ± **0.0050**
Adamax	0.7919 ± 0.0112
RMSprop	0.7612 ± 0.0059
SGD	3.2805 ± 0.2129

Experimental results for different optimizers.

where U is the number of users.

(3) Fraction of Concordant Pairs (FCP)

$$FCP = \frac{n_c}{n_c + n_d} \tag{14}$$

$$n_c = \sum_u n_c^u \tag{15}$$

$$n_d = \sum_u n_d^u \tag{16}$$

$$n_c^u = \left|\left\{(i,j)\,|r'_{ui} > r'_{uj} \text{ and } r_{ui} > r_{uj}\right\}\right| \tag{17}$$

$$n_d^u = \left|\left\{(i,j)\,|r'_{ui} \geq r'_{uj} \text{ and } r_{ui} < r_{uj}\right\}\right| \tag{18}$$

(4) Mean Average Precision (MAP)

Given top K recommendations for each user, MAP is the mean of the AP for all users.

$$AP = \frac{\sum_{n=1}^{K} Precision(n) \times rel\,(n)}{min\,\{K, |\{relevant\ items\}|\}} \tag{19}$$

where $Precision(n)$ is the precision up to cutoff$n$ in the top K and $rel\,(n)$ is a function which returns 1 if the item at rank n is relevant, and 0 if not relevant.

(5) Mean Reciprocal Rank (MRR)

Given top K recommendations for each user:

$$MRR = \frac{1}{U} \sum_u \frac{1}{rank_{firstrelevant}} \tag{20}$$

4.3. Settings

To implement our model, we used Keras[2] with TensorFlow[3] as backend.

(1) Criteria Ratings DNN Settings

We randomly initialized the DNN parameters using a normal distribution with mean of 0 and standard deviation of 0.05. We experimented with a number of optimizers as illustrated in Table 3. In consequence, we used Adam optimizer, with 0.001 learning rate and parameters values as provided in [36]. We used batch size of 512. We tested the embedding vector size as shown in Fig. 4 [8,16,32,64,128], and hence we set each of user and item embedding vector size to 64. We tried different hidden layer configurations as demonstrated in Table 4; and as a result, we selected the [128→64→32→16→8] hidden layers, whereas, in the output layer, there are 7 neurons, equal to the number of the criteria ratings.

(2) Overall Rating DNN Settings

We initialized randomly the DNN parameters like the previous Criteria Ratings DNN, using a normal distribution with mean of 0 and standard deviation of 0.05. We also used Adam optimizer with 0.001 learning rate and the same parameters values. While we set the batch size to 512, we used the [64→32→16→8] hidden layers. In the output layer, there is 1 neuron for the overall rating.

[2] https://github.com/keras-team/keras.

[3] https://github.com/tensorflow/tensorflow.

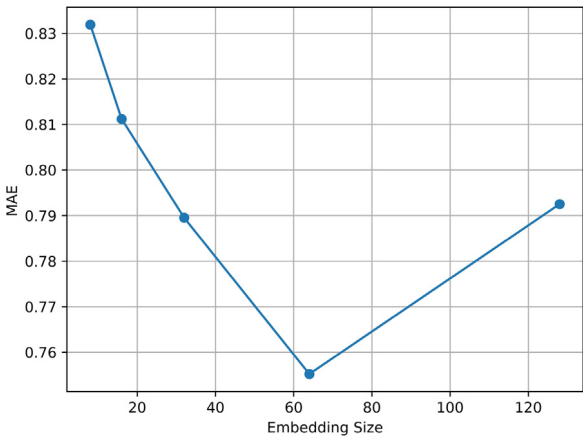

Fig. 4. The impact of embedding vector size.

Table 4
Hidden layers results.

Hidden layers	MAE
[8]	3.4703 ± 0.0708
[8,16]	3.0678 ± 0.5612
[8,16,32]	0.9481 ± 0.0636
[8,16,32,64]	0.7776 ± 0.0146
[8,16,32,64,128]	**0.7552** ± **0.0050**
[8,16,32,64,128,256]	0.7830 ± 0.0629

Experimental results for different hidden layers configurations.

Table 5
Criteria ratings error.

Criteria rating	MAE
Value	0.8195 ± 0.0082
Rooms	0.7612 ± 0.0082
Location	0.6581 ± 0.0077
Cleanliness	0.6324 ± 0.0112
Check in / front desk	0.7582 ± 0.0143
Service	0.7401 ± 0.0103
Business service	0.8774 ± 0.0053
Mean	0.7495 ± 0.0073

4.4. Results

We used 5-fold cross-validation method, to split the data randomly into 80% training set including a 1% validation set and a 20% test set. We repeated this process 5 times and calculated the mean value of the MAE for each criteria rating and their mean in the Criteria Ratings DNN as shown in Table 5. We calculated the mean value of each metric for the overall rating in the whole model. We compared the performance of our model to a number of famous single rating recommendation methods such as SVD [37], SVD++ [38], SlopeOne [39], Baseline Estimates [40], and K Nearest Neighbors (KNN) with Baseline [40]. In addition, we compared it to a single deep neural network that predicts the overall rating directly. The best results of this DNN were obtained from settings similar to the criteria ratings DNN, but with 256 as the size of each user and item embedding vector, [512→256→128→64→32→16→8] hidden layers, and 1 neuron at the output layer for the overall rating. This comparison was done on the overall rating. The results illustrated in Table 6, demonstrate that our proposed model outperformed the other

Table 6
Evaluation results.

Metric	SVD	SVD++	SlopeOne	Baseline estimates	KNN with baseline	Single DNN	Our
MAE	0.8071 ± 0.0043	0.8222 ± 0.0071	0.9160 ± 0.0053	0.8094 ± 0.0019	0.8115 ± 0.0041	0.7855 ± 0.0153	**0.7552 ± 0.0050**
F_1	0.8370 ± 0.0035	0.8390 ± 0.0035	0.8666 ± 0.0018	0.8391 ± 0.0020	0.8388 ± 0.0019	0.8572 ± 0.0024	**0.8769 ± 0.0043**
F_2	0.7951 ± 0.0031	0.7991 ± 0.0057	0.9248 ± 0.0011	0.7986 ± 0.0029	0.7984 ± 0.0031	0.9176 ± 0.0057	**0.9295 ± 0.0014**
FCP	0.5024 ± 0.0210	0.5092 ± 0.0056	0.5096 ± 0.0661	0.5025 ± 0.0117	0.5097 ± 0.0127	0.4957 ± 0.0102	**0.5126 ± 0.0160**
MAP	0.3935 ± 0.0057	0.3483 ± 0.0049	0.3006 ± 0.0026	0.3881 ± 0.0023	0.3865 ± 0.0041	0.4481 ± 0.0091	**0.4643 ± 0.0072**
MRR	0.7272 ± 0.0054	0.7513 ± 0.0018	0.7502 ± 0.0037	0.7374 ± 0.0028	0.7360 ± 0.0010	0.7244 ± 0.0092	**0.7664 ± 0.0039**

Evaluation results for our model against the other algorithms. F_1, F_2, *FCP*, *MAP*, and *MRR* the higher the better while *MAE* the lower the better.

state-of-the-art methods on all the evaluation metrics. According to the results, MAE of our model is at least 5% better than the other methods. In F_1 and F_2, our model excelled the other compared methods. In FCP, our model is the best. Our model surpasses the other models in MAP at least by 7% and it exceeds them at MRR. Moreover, our model with two DNNs for criteria ratings and for overall rating outperformed the single DNN, because the overall rating provides information about how much the user likes the item, and multi-criteria ratings help in understanding why they do like it. Thus, multi-criteria ratings enable estimating the similarity between two users more accurately. Therefore, we used another DNN for the multi-criteria ratings. The results are very promising, showing the advantage of using multi-criteria with deep networks for collaborative filtering recommendation system.

5. Conclusion and future work

In this paper, we proposed a novel multi-criteria collaborative filtering model based on deep learning. First, we discussed the implementation details of applying deep learning and multi-criteria to collaborative filtering, where our model contains two branches: one for predicting the criteria ratings and the other for predicting the overall rating. Then we showed the effectiveness of our model, where the experimental results demonstrated that our proposed model achieved good performance by outperforming the other methods for all the evaluation metrics, which proved that the use of multi-criteria and deep learning in collaborative filtering recommendation system is a successful attempt. Finally, our framework is general, easy to implement, and model-independent. Therefore, we can try to enhance it using different deep learning techniques.

As a follow up in the future, we want to improve the performance of our model, either by studying different deep learning methods such as Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), and Autoencoder or by using more complex deep networks or other feature representation methods.

Acknowledgments

We thank N. Chamoun for careful reading of the manuscript. Special thanks are due to S. Tarbouche for advice and support.

References

[1] W. Carrer-Neto, M.L. Hernández-Alcaraz, R. Valencia-García, F. García-Sánchez, Social knowledge-based recommender system. application to the movies domain, Expert Syst. Appl. 39 (2012) 10990–11000.
[2] D. Bogdanov, M. Haro, F. Fuhrmann, A. Xambó, E. Gómez, P. Herrera, Semantic audio content-based music recommendation and visualization based on user preference examples, Inf. Process. Manag. 49 (2013) 13–33.
[3] M. Al-Hassan, H. Lu, J. Lu, A semantic enhanced hybrid recommendation approach: A case study of e-government tourism service recommendation system, Decis. Support Syst. 72 (2015) 97–109.
[4] H.K. Kim, H.Y. Oh, J.C. Gu, J.K. Kim, Commenders: A recommendation procedure for online book communities, Electron. Commer. Res. Appl. 10 (2011) 501–509.
[5] S. Cleger-Tamayo, J.M. Fernández-Luna, J.F. Huete, Top-n news recommendations in digital newspapers, Knowl. Based Syst. 27 (2012) 180–189.
[6] J. Son, S.B. Kim, Academic paper recommender system using multilevel simultaneous citation networks, Decis. Support Syst. 105 (2018) 24–33.
[7] G. Koutrika, Modern recommender systems, in: Proc. 2018 Int. Conf. Manag. Data - SIGMOD '18, 2018, pp. 1651–1654, http://dx.doi.org/10.1145/3183713.3197389.
[8] F. Ricci, Lior, R. Bracha (Eds.), Recommender Systems Handbook, second ed., 2015, http://dx.doi.org/10.1007/978-1-4899-7637-6.
[9] J. O'Donovan, B. Smyth, Trust in recommender systems, in: Proc. 10th Int. Conf. Intell. User Interfaces, 2005, pp. 167–174.
[10] C. Porcel, A. Ching-López, G. Lefranc, V. Loia, E. Herrera-Viedma, Sharing notes: An academic social network based on a personalized fuzzy linguistic recommender system, Eng. Appl. Artif. Intell. 75 (2018) 1–10.
[11] M. Wasid, R. Ali, An improved recommender system based on multi-criteria clustering approach, Procedia Comput. Sci. 131 (2018) 93–101.
[12] M. Fu, H. Qu, Z. Yi, L. Lu, Y. Liu, A novel deep learning-based collaborative filtering model for recommendation system, IEEE Trans. Cybern. (2018) http://dx.doi.org/10.1109/TCYB.2018.2795041.
[13] M. Hassan, New Machine Learning Methods for Modeling a Multi-Criteria Recommender System, 2018.
[14] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, SIGIR Forum. 51 (2017) 227–234, http://dx.doi.org/10.1145/3130348.3130372.
[15] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proc. 10th Int. Conf. World Wide Web, ACM, New York, NY, USA, 2001, pp. 285–295, http://dx.doi.org/10.1145/371920.372071.
[16] T. Hofmann, Collaborative filtering via Gaussian probabilistic latent semantic analysis, in: Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Dev. Informaion Retr., ACM, New York, NY, USA, 2003, pp. 259–266, http://dx.doi.org/10.1145/860435.860483.
[17] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proc. Fourteenth Conf. Uncertain. Artif. Intell., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 43–52, http://dl.acm.org/citation.cfm?id=20740942074100.
[18] M. Grcar, B. Fortuna, D. Mladenic, M. Grobelnik, Knn versus SVM in the collaborative filtering framework, in: V. Batagelj, H.-H. Bock, A. Ferligoj, A. Žiberna (Eds.), Data Sci. Classif., Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 251–260.
[19] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022, http://dl.acm.org/citation.cfm?id=944919.944937.
[20] R. Bell, Y. Koren, C. Volinsky, Modeling relationships at multiple scales to improve accuracy of large recommender systems, in: Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., ACM, New York, NY, USA, 2007, pp. 95–104, http://dx.doi.org/10.1145/1281192.1281206.
[21] G. Adomavicius, Y. Kwon, New recommendation techniques for multicriteria rating systems, IEEE Intell. Syst. 22 (2007) 48–55, http://dx.doi.org/10.1109/MIS.2007.58.
[22] T.Y. Tang, G. Mccalla, The pedagogical value of papers: A collaborative-filtering based paper recommender, J. Digit. Inf. 10 (2009) http://dx.doi.org/10.1109/WKDD.2009.132.
[23] L. Si, R. Jin, Flexible mixture model for collaborative filtering, in: Proc. Twent. Int. Conf. Int. Conf. Mach. Learn., AAAI Press, 2003, pp. 704–711, http://dl.acm.org/citation.cfm?id=3041838.3041927.
[24] N. Sahoo, R. Krishnan, G. Duncan, J. Callan, The halo effect in multicomponent ratings and its implications for recommender systems: The case of yahoo! movies, Inf. Syst. Res. 23 (2012) 231–246, http://dx.doi.org/10.1287/isre.1100.0336.
[25] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, V.N. Vapnik, Support vector regression machines, Adv. Neural Inf. Process. Syst. 9 (1) (1997) 155–161, doi:10.1.1.10.4845.
[26] M. Hassan, M. Hamada, A neural networks approach for improving the accuracy of multi-criteria recommender systems, Appl. Sci. 7 (2017) 868, http://dx.doi.org/10.3390/app7090868.

[27] M. Hassan, M. Hamada, Genetic algorithm approaches for improving prediction accuracy of multi-criteria recommender systems, Int. J. Comput. Intell. Syst. 11 (2018) 146–162, http://dx.doi.org/10.2991/ijcis.11.1.12.
[28] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering, in: Proc. 24th Int. Conf. Mach. Learn., ACM, New York, NY, USA, 2007, pp. 791–798, http://dx.doi.org/10.1145/1273496.1273596.
[29] K. Georgiev, P. Nakov, A non-IID framework for collaborative filtering with restricted Boltzmann machines, in: S. Dasgupta, D. McAllester (Eds.), Proc. 30th Int. Conf. Mach. Learn., PMLR, Atlanta, Georgia, USA, 2013, pp. 1148–1156, http://proceedings.mlr.press/v28/georgiev13.html.
[30] Y. Ouyang, W. Liu, W. Rong, Z. Xiong, Autoencoder-based collaborative filtering, in: C.K. Loo, K.S. Yap, K.W. Wong, A.T. Beng Jin, K. Huang (Eds.), Neural Inf. Process., Springer International Publishing, Cham, 2014, pp. 284–291.
[31] Y. Wu, C. DuBois, A.X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: Proc. Ninth ACM Int. Conf. Web Search Data Min. - WSDM '16, 2016, pp. 153–162, http://dx.doi.org/10.1145/2835776.2835837.
[32] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural Collaborative Filtering, 2017, http://dx.doi.org/10.1145/3038912.3052569.
[33] O. Fuentes, J. Parra, E. Y. Anthony, V. Kreinovich, Why Rectified Linear Neurons Are Efficient: Symmetry-Based, Complexity-Based, and Fuzzy-Based Explanations, Dep. Tech. Reports (CS). (2017). https://digitalcommons.utep.edu/cs_techrep/1171.
[34] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[35] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training By Reducing Internal Covariate Shift, 2015, http://dx.doi.org/10.1007/s13398-014-0173-7.2.
[36] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2014, pp. 1–15, http://dx.doi.org/10.1145/1830483.1830503.
[37] Y. Koren, R. Bell, C. Volinsky, Matrix Factorization Techniques for Recommender Systems, 2009, pp. 42–49, http://dx.doi.org/10.1109/MC.2009.263.
[38] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, in: Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., ACM, New York, NY, USA, 2008, pp. 426–434, http://dx.doi.org/10.1145/1401890.1401944.
[39] D. Lemire, A. Maclachlan, Slope One Predictors for Online Rating-Based Collaborative Filtering, 2007, http://dx.doi.org/10.1137/1.9781611972757.43.
[40] Y. Koren, Factor in the neighbors, ACM Trans. Knowl. Discov. Data. 4 (2010) 1–24, http://dx.doi.org/10.1145/1644873.1644874.