Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 855 – 870
© TÜBİTAK
doi:10.3906/elk-2107-145

A new similarity-based multicriteria recommendation algorithm based on autoencoders

Zeynep BATMAZ[*], **Cihan KALELİ**
Department of Computer Engineering, Faculty of Engineering, Eskişehir Technical University, Eskişehir, Turkey

Received: 31.07.2021 • **Accepted/Published Online:** 07.10.2021 • **Final Version:** 21.03.2022

Abstract: Recommender systems provide their users an efficient way to handle information overload problem by offering personalized suggestions. Traditional recommender systems are based on two-dimensional user-item preference matrix constructed depending on the users' overall evaluations over items. However, they have begun to present their preferences under various circumstances. Thus, traditional recommendation techniques fail to process multicriteria ratings during the recommendation process. Multicriteria recommender systems are an extension of traditional recommender systems that utilize multicriteria-based user preferences. Multicriteria recommender systems provide more personalized and accurate predictions compared to traditional recommender systems. However, the increased amount of data dimension causes sparsity to be a major problem of such systems. Especially, the similarity-based multicriteria recommender systems may fail to find similar neighbors to an active user due to the lack of corated items among users. Therefore, we propose a new similarity-based multicriteria collaborative filtering approach based on autoencoders. In order to handle sparsity, the proposed method extracts nonlinear, low-dimensional, dense features from raw and sparse users'/items' preferences. Our experimental outcomes show that the proposed work can amortize the negative impacts of sparsity over the accuracy comparing with the state-of-the-art multicriteria recommendation techniques.

Key words: Multicriteria, collaborative filtering, autoencoders, sparsity, accuracy

1. Introduction

Recommender systems (RS) are influential ways of dealing with the information overload problem caused by intensive use of the Internet. Satisfaction of the customers increases with the help of RS by matching the customers with related products without losing money and time. Moreover, RS improve usability and continuity of the systems. Collaborative filtering (CF) is the most popular technique used for producing recommendations. CF is based on the idea that people with similar behavioral leanings tend to agree in the future.

Users' overall preferences of products are considered during the recommendation processes of traditional CF techniques. With the increasing number of CF applications, the perspectives of users' evaluations over products have changed over time. In addition to make a general assessment of a product, users have begun to evaluate it considering various subcriteria. With this way, customers obtain more personalized recommendations. Additionally, product providers have opportunities to improve their weaknesses with the help of detailed feedback provided by the users. In order to handle the evaluations over multiple criteria, multicriteria collaborative filtering (MCCF) techniques as a new continuation of traditional CF are introduced [1]. MCCF allows for producing predictions using users' evaluations in multiple directions. Moreover, since the relations among users

[*]Correspondence: zozdemir@eskisehir.edu.tr

are obtained utilizing ratings in multiple dimensions instead of one general aspect, MCCF provides more accurate predictions comparing with the traditional CF [1, 2]. Despite the improvements in accuracy, sparsity has become more noticeable in MCCF systems [3]. In real life, a limited number of users evaluate a service/product for a single criterion. Since users represent their evaluations over multiple criteria, this ratio decreases further with the increasing number of criteria. Especially, for similarity-based MCCF systems, sparsity causes the number of corated items among users to decrease dramatically. As a result, similarity-based approaches fail to find similar neighbors to an active user [3]. Thus, this sparsity problem causes a negative impact on the accuracy of the predictions and may even cause an inability to produce predictions.

Researchers have been working to absorb the negative impacts of sparsity on the accuracy of the predictions in MCCF systems. However, most of these studies are based on linear methods. Nonlinear methods provide obtaining more complex and unexpected information from data with respect to linear approaches [4]. There are a limited number of studies that utilize nonlinear assumptions; however, they need extra information about users or items which is not always possible to gather. Even though there are nonlinear approaches based on deep learning (DL) techniques that are utilized in overall rating-oriented RS, these approaches operate on single rating dimension. Thus, they cannot be directly applied in MCCF systems. Therefore, there is a need for producing predictions on a sparse data set utilizing nonlinear approaches without using extra information. Thus, we proposed a similarity-based MCCF algorithm (AE-simMCCF) based on autoencoders in order to prevent the adverse effect of sparsity over the accuracy of the produced predictions in this work. AE-simMCCF utilizes autoencoders to extract nonlinear, hidden, and low-dimensional features from raw, high-dimensional users'/items' preferences for each criterion. AE-simMCCF uses these features to compute similarities among users/items instead of sparse users'/items' preferences in order to produce overall predictions. With this way, AE-simMCCF manages to deal with sparsity problem. Contributions provided by the proposed study to the literature can be recorded as follows:

- A new similarity-based MCCF approach which utilizes autoencoders is proposed.
- Raw, sparse, high-dimensional criterion-based users'/items' preferences are reduced to low-dimensional, nonlinear, complex, dense features with autoencoders.
- Extracted dense and low-dimensional features by autoencoders are utilized to find similar neighbors. With this way, the proposed method prevents the negative impact of sparsity over the accuracy of the produced overall criterion-based referrals.

The organization for the rest of the study is specified as follows. Existing MCCF techniques especially focus on handling sparsity are presented in Section 2. Basic MCCF techniques and utilized DL technique are introduced in Section 3. Section 4 presents the proposed approach. Section 5 presents the experimental outcomes. Conclusions and future work are given in Section 6.

2. Related work

DL is often utilized in many research areas, including RS due to the developed fast processing units and increased need in processing big data [4]. DL-based methods are utilized in overall rating-oriented RS in order to enhance the accuracy of referrals by extracting nonlinear, latent, and complex relations among users and items [5–8] and extracting nonlinear features from content/review information [9–11]. DL-based methods are also used in RS to handle the sparsity problem by combining user preferences, and side information in hybrid approaches [12, 13].

Reducing the dimension of user-item preference matrix with DL-based techniques is also utilized in RS to deal with sparsity [14, 15]. Despite the abundance of studies based on DL techniques in single rating-oriented RS, the number of DL-based approaches aiming to deal with sparsity and accuracy issues in MCCF is limited.

Multicriteria-based recommendation techniques are grouped as similarity-based and aggregation function-based approaches [1]. Several methods are proposed to deal with problems of MCCF systems [16]. Most of the existing studies are focused on improving accuracy. Several techniques are proposed for computing similarities such as Euclidean distance [17], Mahalanobis distance [18] grey relational analysis [19, 20] for enhancing the accuracy of referrals in similarity-based methods. To improve the accuracy of predictions in aggregation function-based MCCF systems, a part of the researchers try to produce more accurate criterion-based predictions [21–26]. Criterion-based predictions are generated by matrix factorization [23], fuzzy Bayesian approach [21], autoencoders [25], multilayer neural networks [26]. Rest of the researchers try to enhance the accuracy of predictions by integrating more precise aggregation function [27–31]. Support vector regression [28], feed-forward neural networks [26, 29], autoencoders [30], adaptive genetic algorithm [31], genetic programming [32], and tensor factorization [33, 34] are utilized for learning aggregation function. All of the mentioned studies focused solely on improving accuracy without addressing the sparsity problem.

Even though AE-simMCCF is based on autoencoders, it is completely different from our previous work AE-MCCF presented in [25]. The previous work AE-MCCF is an aggregation function-based method focusing on improving accuracy of the predictions, whereas AE-simMCCF is a similarity-based MCCF approach which focuses on handling the sparsity issue. AE-MCCF utilizes autoencoders to produce criterion-based predictions utilizing the produced data at the outermost decoder layer of the autoencoder. Unlike the previous work, AE-simMCCF utilizes autoencoders aiming to map raw, sparse, high-dimensional users'/items' preferences into low-dimensional, dense, nonlinear, and complex features in order to compute similarities among users/items. Thus, AE-simMCCF utilizes the produced data at the outermost encoder layer of the autoencoder. Moreover, the proposed work differs from AE-MCCF by utilizing other mechanisms such as dropout and batch normalization to prevent overfitting and provide better learning. Since the dropout regularization decreases the coadaptation, the network tends to learn more robust features [8]. Furthermore, AE-MCCF is limited to several activation functions at the output layer since the input data is not normalized. However, any activation function can be utilized in any layer of AE-simMCCF, since the input data is normalized to make the input range to be proper with the chosen activation function.

Researchers propose some approaches based on integrating semantic and content information to deal with sparsity problem for MCCF systems. Both users' content information and ontological semantic filtering are utilized to solve sparsity problem in [35]. Content features extracted by stacked denoising autoencoders are integrated into tensor factorization in [36]. Even though the work presented in [36] utilizes denoising autoencoders to extract features from content information, the work is still based on linear assumption during prediction process. The authors in [37] propose using preference-based similarity instead of computing correlations over sparse rating profiles to deal with sparsity issue. Additionally, reducing dimensions of users'/items' sparse preferences into low-dimensional dense space helps to deal with both sparsity and scalability issues. Principle component analysis (PCA) and higher-order singular value decomposition are utilized to reduce dimensions of the multidimensional user-item matrix into low-level space to alleviate sparsity and scalability issues in [38]. PCA is utilized in many studies with the purpose of dimensionality reduction for handling multicollinearity and scalability issues [39, 40]. The neuro-fuzzy system is utilized to deal with sparsity in [41] since more precise features can be obtained with that system. Even though sparsity is a main problem of MCCF domain, the number of studies focused on dealing with sparsity is limited. Furthermore, nearly all of these existing studies are based on linear assumptions. Nonlinear assumptions are only utilized for extracting information from content data in a limited number of studies; thus, the extracted relations among users' and items' interactions are still based on linear assumptions. Therefore, there is still a need for approaches based on nonlinear assumptions to deal with the sparsity issue in MCCF systems.

3. Background

3.1. Multicriteria collaborative filtering

In MCCF systems, a user's preference relation is specified with a rating function as $R : UsersXItems \rightarrow R_0XR_1XR_2X \ \ ...XR_k$, where R_0 represents overall ratings which are assigned to the items by users. R_c represents c^{th} criterion ratings of users for items, with $c \in 1, 2, ..., k$. MCCF techniques are classified as aggregation function-based and similarity-based approaches [1]. In similarity-based methods, MCCF problem is mapped into a traditional CF problem by aggregating correlations among users/items with regard to criteria and overall ratings. With this purpose, criteria-based relations among users/items are computed utilizing any correlation measure. For computing aggregated correlations among users/items, either worst-case or average similarity methods are utilized. Among the computed discrete similarities, the minimum one is assigned as aggregated similarity in worst-case similarity technique. The aggregated similarity is computed as the average of the discrete similarities in the average similarity approach. The aggregated similarity among two users/items as u and v are computed using average and worst-case similarity methods as in Eq. 1 and Eq. 2, respectively:

$$sim_{avg}(u,v) = \frac{\sum_{c=0}^{c=k} sim_c(u,v)}{k+1} \tag{1}$$

$$sim_{min}(u,v) = min_{c=0,...,k} sim_c(u,v) \tag{2}$$

where sim_c is any similarity function such as Pearson and cosine similarities for the c^{th} criterion. In aggregation function-based approaches, an aggregation function is learned to find out the relations among criteria ratings and overall evaluations. Aggregation function-based methods consist of two major steps as predicting criteria-votes and learning the aggregation function [1]. In the former one, predictions are generated for each criterion using anyone of the recommendation methods. In the second step, aggregation function f is learned with several methods such as data mining, machine learning, and domain knowledge. Then, the overall predictions R_0 can be obtained using the criterion-based predictions ($R_1, R_2, ..., R_k$) and f as shown in Eq. 3:

$$R_0 = f(R_1, R_2, ..., R_k) \tag{3}$$

3.2. Autoencoders

An autoencoder is a kind of artificial neural network. An autoencoder is used for extracting nonlinear features, dimensionality reduction, and computing deficient values in RS [4]. An autoencoder tries to regenerate its input at the output layer [42]. A simple autoencoder can be represented with three layers as the input layer, hidden layer, and output layer. An autoencoder consists of two sections as encoder and decoder parts. The autoencoder takes its input at the input layer, encodes it at the hidden layer as shown in Eq. 4 and decodes it at the output layer as in Eq. 5. The output of the encoder part can be utilized as feature engineering and dimensionality reduction. The learning process of an autoencoder for a given input set X is shown in Eq. 6

$$f(x) = \theta(W_1 x + b_1) \tag{4}$$

where θ represents a nonlinear function, W_1 represents the weight matrix between the input and hidden layers, x is the input of the autoencoder, and b_1 represents the biased vector belonging to the hidden layer.

$$g(f(x)) = \delta(W_2 f(x) + b_2) \tag{5}$$

where δ is a nonlinear function, W_2 represents the weight matrix between the hidden and output layers, $f(x)$ is the encoded data, and b_2 represents the biased vector belonging to the output layer.

$$\sum_{x \in X} \|x - g(f(x))\|_2^2 \tag{6}$$

4. AE-simMCCF

AE-simMCCF is a similarity-based MCCF method that utilizes autoencoders. AE-simMCCF consists of two parts as feature extraction/dimensionality reduction part and prediction part. Figure 4 shows a general representation of the procedure of user-based AE-simMCCF (U-AE-simMCCF). Algorithm 1 presents the pseudocode of AE-simMCCF procedure.

AE-simMCCF uses autoencoders in feature extraction/dimensionality reduction part to obtain low-dimensional, dense, nonlinear, hidden features from raw, sparse, and high-dimensional user/item preferences for handling sparsity in MCCF systems. For the users set U and items set I in a k-dimensional multicriteria recommender domain, the rating space is k x n x m where the number of users in U is represented with n and the number of items in I is specified as m. In order to compute similarities among users/items, the rating space for k-dimensional multicriteria problem is decomposed into k single rating problems where each one has a rating space as n x m. Each criterion-based n x m user-item matrix contains sparse user/item preferences. Low-dimensional, dense, and complex features are extracted from these preferences by constructing an autoencoder for each one of the criterion-based user-item matrices. Each user u and item i are represented with sparse vectors as $r_u = r_{u1}, r_{u2}, ..., r_{um}$ and $r_i = r_{i1}, r_{i2}, ..., r_{in}$, respectively in a criterion-based n x m user-item matrix. For a user-based autoencoder, each r_u in the matrix is considered an input of the autoencoder after applying several preprocessing operations mentioned in the steps from 4 to 6 of Algorithm 1 to the matrix. For this purpose, missing ratings are assigned to zero as in [6]. In order to disallow the network to be punished because of the missing ratings, sign function is applied to the user-item matrix and the result is saved into another matrix Y. Moreover, to make the input range compatible with the output range of the autoencoders activation functions, the observed ratings in the user-item matrix is mapped into the corresponding range depending on the chosen activation function. After feeding the autoencoder with r_u, the input is encoded and converted into a nonlinear, dense, and low-dimensional form using the Eq. 4 with the autoencoder's encoder part. Then, the input of the autoencoder is reconstructed at the output layer with its decoder part using the Eq. 5. The error value is computed for each neuron at the outermost layer and it is multiplied with the related cell in Y. With the help of Y, it is guaranteed that the network is penalized due to only the loss values of observed votes. Furthermore, L_2 regularization is used and dropout is added to each encoder and decoder layer of the autoencoder to prevent overfitting. Additionally, batch normalization is used before activation

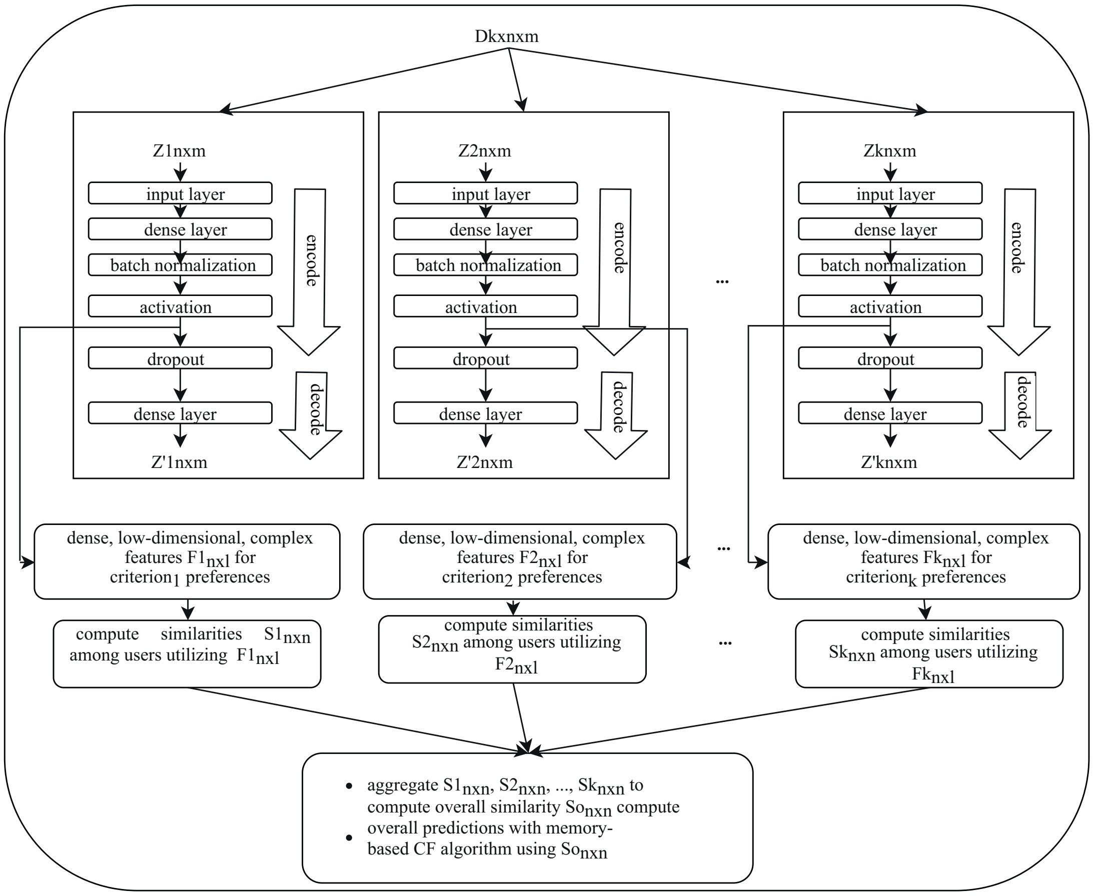
Dkxnxm
Z1nxm
input layer
dense layer
batch normalization
activation
dropout
dense layer
Z'1nxm
encode
decode
Z2nxm
Z'2nxm
Zknxm
Z'knxm
...
dense, low-dimensional, complex features F1nxl for criterion1 preferences
dense, low-dimensional, complex features F2nxl for criterion2 preferences
dense, low-dimensional, complex features Fknxl for criterionk preferences
compute similarities S1nxn among users utilizing F1nxl
compute similarities S2nxn among users utilizing F2nxl
compute similarities Sknxn among users utilizing Fknxl
aggregate S1nxn, S2nxn, ..., Sknxn to compute overall similarity Sonxn compute overall predictions with memory-based CF algorithm using Sonxn

Figure. General representation of U-AE-simMCCF.

functions in each dense layer except the outermost layer of the autoencoder aiming to prevent overfitting, to provide more precise features and better learning [42, 43]. Batch normalization also alleviates faster learning and better generalization for the network. L_2 regularization is utilized. Eq. 7 represents the loss function for the autoencoder.

$$\sum_{u_O \in U} \|u_O - g(f(u_O))\|_2^2 + \lambda(\|W_1\|_2^2 + \|W_2\|_2^2) \tag{7}$$

where u_O is the observed ratings vector for user u, and λ represents the regularization term. After the training process is completed, the network is fed with r_u, and nonlinear and dense features are obtained from the autoencoders encoder part. Since the number of neurons l in the encoding layer is much smaller than m, it is guaranteed that obtained features are more low-dimensional than r_u. This situation decreases the number of computations during similarity computation. The extracted criterion-based features f_c for all users are utilized for computing similarities with cosine similarity as given in Eq. 8 for each criterion c where $c \in 1, 2, ..., k$ in the prediction part. The aggregated similarity So is computed either with worst-case or average similarity methods. The prediction of item i for user u is produced as given in Eq. 9 using So.

$$S_c(u,v) = \frac{\sum_{i=1}^{l} F_c(u,i)F_c(v,i)}{\sqrt{\sum_{i=1}^{l} F_c(u,i)^2}\sqrt{\sum_{i=1}^{l} F_c(v,i)^2}} \tag{8}$$

$F_c(u,i)$ is the i^{th} feature for user u obtained by the autoencoder in terms of criterion c.

$$P(u,i) = \frac{\sum_{v \in N(u)} So(u,v)R(v,i)}{\sum_{v \in N(u)} |So(u,v)|} \tag{9}$$

where $N(u)$ is the set of top-n neighbors of user u, and $R(v,i)$ is the rating of item i given by user v.

Algorithm 1 U-AE-simMCCF algorithm

```
    Input: $D_{kXnXm}$                      ▷ Multidimensional user-item matrix for $k$-criteria
    Output: $P_{nXm}$                       ▷ overall evaluations-based predictions
 1: First section:
 2:     Separate $D$ into k $nXm$ matrices with regard to criteria
 3:     for each $nXm$ criterion-based matrix $Z$ do
 4:         Assign 0 to lost values in $Z$
 5:         Compute $Y = \text{Sign}(Z)$
 6:         Map the ratings in $Z$ into the range [0, 1] or [-1, 1] depending on the chosen activation function
 7:         Design an autoencoder $A$ for $Z$
 8:         Network training part:
 9:             Feed $A$ with each user preference $u$ in $Z$
10:             Utilize Eq. 4 aiming to encode the user preference
11:             Utilize Eq. 5 with the purpose of decoding the encoded data at the decoder layer
12:             Compute error values $E = |u - g(f(u))|$
13:             Compute observed error values $E_O = |u_O - g(f(u_O))| = E \odot Y$
14:             Use the loss function given in Eq. 7 and preferred optimizer for updating biases and weights for
    $A$
15:         Attain the trained autoencoder $A'$ after finishing the training process for $A$
16:         Feature extraction/dimensionality reduction part:
17:             Use each user preference in $Z$ as input of the trained autoencoder $A'$
18:             Utilize Eq. 4 to obtain low-dimensional, dense, complex features at the encoder layer whose
    length is $l$.
19:         Obtain the feature matrix $F_{nxl}$ after feeding $A'$ with all users preferences in $Z$
20:     end for
21: Second section:
22:     for each $nXl$ criterion-based feature matrix $F$ do
23:             Compute similarities $S_{nxn}$ among users utilizing the $F_{nxl}$ with cosine similarity as in Eq. 8
24:     end for
25:     Use computed criterion-based similarities in order to calculate the aggregated similarity $So_{nxn}$ either
    with worst-case or average similarity method.
26:     Compute $P$ utilizing traditional memory-based CF algorithm and $So_{nxn}$ as given in Eq. 9
```

5. Experimental work and discussions

Several trials are performed on two real data sets to show how effectively AE-simMCCF handles sparsity with regard to accuracy and coverage in MCCF systems. In the first part of the experiments, the impacts of the changing parameters such as activation functions and hidden layer numbers on the performance of AE-simMCCF are shown. In the second part, AE-simMCCF is compared with the state-of-the-art similarity-based and aggregation function-based MCCF algorithms to show using nonlinear, low-dimensional, dense features in terms of raw, high-dimensional sparse user/item preferences during the prediction process prevents adverse effects of sparsity over the accuracy of the predictions. Moreover, AE-simMCCF is compared with the baseline algorithms in terms of their running times in seconds. The utilized baseline MCCF methods for comparison are below:

- Similarity-based approaches: Minimum similarity method-based traditional MCCF approach (TMCCF-MinSim) [1] and average similarity method-based traditional MCCF approach (TMCCF-AvgSim) [1]
- Aggregation function-based approaches: Support vector machines-based traditional MCCF approach (TMCCF-SVM) [28] and AE-MCCF [25]

All the above baseline methods are designed as both user- and item-based. For the baselines TMCCF-MinSim, TMCCF-AvgSim, and TMCCF-SVM, the number of neighbors is experimentally set to 25 and cosine correlation is utilized considering our data sets and experimental methodology. Radial basis function is used as kernel function for TMCCF-SVM. Since AE-MCCF's experimental methodology is the same as the proposed work, the parameters which provide the best accuracy results specified in [25] are protected.

5.1. Data sets and evaluation measures

In the trials, two real data sets Yahoo!Movies (YM) and TripAdvisor (TA) which are gathered in [28] and [44], respectively are benefited. A subset including users and movies which possesses at least five ratings is chosen from YM data set for the trials. The subset includes 63,027 multicriteria votes of 4377 users for 2565 movies with a sparsity rate of 99.4386%. YM data set contains evaluations for the criteria as story, acting, direction, and visuals besides overall evaluations. YM data set's letter-based 13-level rating scale (A+ to F) is converted into discrete numeric 13-level rating scale from 1 to 13. A subset consisting of 4798 multicriteria ratings from 1346 users for 1289 hotels with a sparsity rate of 99.7235% is chosen from TA data set for the trials. Each user in the subset has at least three votes. The users evaluate the hotels under the criteria as value, rooms, location, cleanliness, check-in, service, and business service besides overall evaluations with a numeric five-star rating scale. Each user's ratings in YM data set are split into two sets as training and testing votes in the ratio of 80% and 20%, respectively for user-based approaches. For TA data set, the whole set is divided into training and testing votes in the ratio of 80% and 20%, respectively instead of each user's ratings for user-based approaches since each user has at least three votes. Five distinct train and test set pairs are attained by repeating the mentioned procedure five times for each data set. For item-based approaches, we performed experiments only for YM data set. TA data set has lots of hotels which has only one rating, which prevents designing item-based approaches. Thus, to perform item-based experiments for YM data set, the whole process for user-based approaches is repeated to obtain train and test sets pairs but this time each item's ratings are divided as training and testing votes in the ratio of 80% and 20%, respectively. Additionally, to provide reliable results, each analysis is repeated three times for each one of the pairs and the overall outcome is obtained by averaging all the results.

Sparsity negatively impacts accuracy of the produced predictions. Moreover, it may even disallow to produce predictions which is measured by coverage. Thus, to measure the performance of AE-simMCCF with regard to accuracy and coverage, mean absolute error (MAE), root mean squared error (RMSE), and coverage metrics are utilized. Coverage is related to the number of available items in the system covered by the produced predictions [45]. Eq. 10, Eq. 11, and Eq. 12 describe coverage, MAE, and RMSE, respectively. R_t and $|R_t|$ symbolize the test ratings and the number of test ratings, respectively. $|R_p|$ represents the number of produced predictions. Moreover, to measure the performances of algorithms in terms of their running times, several analyses are provided in seconds.

$$Coverage = \frac{|R_p|}{|R_t|} \tag{10}$$

$$MAE = \frac{1}{|R_p|} \sum_{(i,j)\in R_p} |R_{t,ij} - R_{p,ij}| \tag{11}$$

$$RMSE = \sqrt{\frac{1}{|R_p|} \sum_{(i,j)\in R_p} (R_{t,ij} - R_{p,ij})^2} \tag{12}$$

5.2. Experimental outcomes

5.2.1. Impacts of network parameters

In order to construct an autoencoder for each criterion, Keras 2.1.5 with TensorFlow backend is benefited. Several fixed hyperparameters such as batch size, λ, learning rate, dropout regularization are used during the training processes of the autoencoders. For both of the data sets, λ is experimentally set to 0.001. Weights and biases are initialized with He normal distribution. Mean squared error is the used loss function for the autoencoders. Biases and weights are optimized with Adam optimizer. Adam optimizer's default parameter values are protected except learning rate and decay. Both learning rate and decay are experimentally set to 0.0001. Batch size is set to 30. Dropout regularization is set to 0.2 and 0.1 for YM and TA data sets, respectively.

Aiming to show how the performance of AE-simMCCF is affected by changing network parameters such as activation function and hidden layer number, various experiments are performed. For representing how changing number of encoder layers impresses the performance of AE-simMCCF, encoder layer numbers are specified as 1, 2, and 4. $1/5^{th}$, $1/8^{th}$, and $1/12^{th}$ of the input size are the neuron numbers in the encoder layers, respectively. The most known nonlinear activation functions as sigmoid, hyperbolic tangent (Tanh), and exponential linear unit (ELU) are used in the trials to represent the influences of changing activation functions over the performance of AE-simMCCF. The input data is scaled to the range [0, 1] or [-1, 1] for sigmoid and Tanh functions, respectively. Since ELU can deal with negative input values and mean of the network input to be close to zero provides faster convergence [46], the input data is scaled to the range [-1, 1] for ELU function.

Table 1 shows how varying activation functions and encoder layer numbers affects the performance of U-AE-simMCCF with regard to accuracy and coverage on YM data set. It is obvious that the best accuracy results are generally obtained with two encoder layers for all activation functions except ELU. The capacity of the network is an important factor that causes this situation to occur. When the capacities of machine learning algorithms are enough in terms of the complexity of the corresponding function and the quantity of input data, they will commonly perform their best [42]. Coverage results generally improve with increasing encoder layer numbers. Moreover, it is seen that tanh function provides more accurate recommendations than sigmoid and ELU functions. The reason for this situation may be that tanh function provides faster convergence and better generalization compared with sigmoid function since tanh produces outputs having average close to zero [47]. Considering both coverage and accuracy metrics, the best performance results for U-AE-simMCCF is obtained with tanh function for 4 encoder layers.

Table 1. Effects of changing activation functions and encoder layer numbers over the performance of U-AE-simMCCF with regard to accuracy and coverage on YM data set.

Number of encoder layers	Activation function	Aggregation type	MAE	RMSE	Coverage
1 encoder layer	Sigmoid	worst-case	2.4919	2.6540	0.2822
		avg	2.5021	2.6626	0.2809
	Tanh	worst-case	2.1324	2.2730	0.3322
		avg	2.1421	2.2812	0.3238
	ELU	worst-case	2.2037	2.3311	0.2803
		avg	2.2267	2.3568	0.2871
2 encoder layers	Sigmoid	worst-case	2.3510	2.4912	0.2461
		avg	2.3455	2.4912	0.2580
	Tanh	worst-case	2.0897	2.2497	0.4655
		avg	2.0946	2.2291	0.4609
	ELU	worst-case	2.1695	2.3141	0.4153
		avg	2.1732	2.3214	0.4179
4 encoder layers	Sigmoid	worst-case	2.4029	2.5548	0.2738
		avg	2.4111	2.5683	0.2816
	Tanh	worst-case	2.0926	2.2680	0.5188
		avg	2.0696	2.2411	0.5192
	ELU	worst-case	2.1643	2.3501	0.4821
		avg	2.1624	2.3330	0.4552

Table 2 shows impacts of varying activation functions and encoder layer numbers on the performance of U-AE-simMCCF with regard to accuracy and coverage on TA data set. As it is obvious, the best accuracy results are obtained with two encoder layers for all activation functions. As it is said, this situation is related with the capacity of the network. Compared to YM results, it is seen that the structure of the data set such as the number of samples for training the network and sparsity ratio affects the capacity of the network. Coverage results generally improve with increasing encoder layer numbers like the coverage results of YM data set. Moreover, for TA data set, it is obvious that the best performance results for U-AE-simMCCF in terms of accuracy and coverage are obtained with tanh function for 2 encoder layers.

Table 3 shows how different activation functions and encoder layer numbers impress the performance of I-AE-simMCCF in the way of accuracy and coverage on YM data set. The best accuracy and coverage results are obtained with tanh activation function for four encoder layers. Comparing I-AE-simMCCF with U-AE-simMCCF, it is observed that coverage and accuracy results for I-AE-simMCCF are better than the ones for U-AE-simMCCF. This situation may be caused by sparsity ratios of input samples. Since the number of movies in YM dataset is smaller than the number of users, movie preferences are denser than the user preferences. On the other hand, since the number of input samples in training data for I-AE-MCCF is smaller than the ones for U-AE-simMCCF, increasing encoder layer numbers may not improve accuracy for some of the activation functions. The number of input samples is not enough to attain more accurate predictions with increasing encoder layers for ELU. As it is said before, the capacity of the network specifies its performance. This capacity is related to the amount of input data and utilized hyperparameters such as layer numbers.

Table 2. Impacts of changing activation functions and encoder layer numbers over the performance of U-AE-simMCCF with regard to accuracy and coverage on TA data set.

Number of encoder layers	Activation function	Aggregation type	MAE	RMSE	Coverage
1 encoder layer	Sigmoid	worst-case	1.0385	1.0422	0.0639
		avg	1.0875	1.0928	0.0661
	Tanh	worst-case	0.8822	0.8860	0.0976
		avg	0.8944	0.8970	0.0983
	ELU	worst-case	0.9099	0.9148	0.0958
		avg	0.9166	0.9193	0.0953
2 encoder layers	Sigmoid	worst-case	0.9694	0.9760	0.0543
		avg	0.9795	0.9841	0.0581
	Tanh	worst-case	0.9461	0.9502	0.1025
		avg	0.8485	0.8525	0.1034
	ELU	worst-case	0.9262	0.9313	0.1024
		avg	0.8728	0.8773	0.1028
4 encoder layers	Sigmoid	worst-case	0.9641	0.9698	0.0523
		avg	0.9106	0.9155	0.0603
	Tanh	worst-case	1.1321	1.2159	0.1026
		avg	0.8602	0.9287	0.1022
	ELU	worst-case	1.1332	1.1476	0.1000
		avg	0.8982	0.9025	0.0972

Table 3. Effects of changing activation functions and encoder layer numbers over the performance of I-AE-simMCCF with regard to accuracy and coverage on YM data set.

Number of encoder layers	Activation function	Aggregation type	MAE	RMSE	Coverage
1 encoder layer	Sigmoid	worst-case	2.3496	2.6425	0.3818
		avg	2.4023	2.6954	0.3837
	Tanh	worst-case	2.0143	2.2940	0.4792
		avg	2.0077	2.2857	0.4790
	ELU	worst-case	2.0125	2.2785	0.4431
		avg	2.0434	2.3104	0.4435
2 encoder layers	Sigmoid	worst-case	2.2590	2.5148	0.3146
		avg	2.2516	2.5116	0.3319
	Tanh	worst-case	1.9999	2.2785	0.5269
		avg	1.9865	2.2592	0.5275
	ELU	worst-case	2.0574	2.3278	0.4979
		avg	2.0610	2.3324	0.5033
4 encoder layers	Sigmoid	worst-case	2.2708	2.5255	0.3407
		avg	2.2725	2.5349	0.3529
	Tanh	worst-case	1.9933	2.2886	0.5554
		avg	1.9832	2.2794	0.5613
	ELU	worst-case	2.0739	2.3748	0.5394
		avg	2.0395	2.3375	0.5470

5.2.2. Comparison with state-of-the-art algorithms

To show the effectiveness of AE-simMCCF with regards to accuracy and coverage of the predictions, it is compared with state-of-the-art algorithms on YM and TA data sets. Table 4 and Table 5 represents the attained outcomes for YM and TA data sets, respectively. As it is seen in the tables, AE-simMCCF can better absorb the negative impacts of sparsity issue over the accuracy of the produced predictions in terms of accuracy comparing with the baseline algorithms for both data sets. The reason for this situation is that AE-simMCCF utilizes nonlinear, dense, complex, and low-dimensional features extracted from high-dimensional, raw, sparse users'/items' preferences by autoencoders during the prediction process.

Table 4. Comparison of AE-simMCCF with state-of-the-art algorithms on YM data set.

Method	Aggregation Type	MAE	RMSE	Coverage
U-AE-simMCCF	worst-case	2.0926	2.2680	0.5188
U-AE-simMCCF	avg	2.0696	2.2411	0.5192
U-TMCCF-MinSim	worst-case	2.5766	2.7373	0.5002
U-TMCCF-AvgSim	avg	2.5573	2.7382	0.5003
U-TMCCF_SVM	-	2.3599	2.4899	0.4259
U-AE-MCCF	-	2.2553	2.4450	1.0000
I-AE-simMCCF	worst-case	1.9933	2.2886	0.5554
I-AE-simMCCF	avg	1.9832	2.2794	0.5613
I-TMCCF-MinSim	worst-case	2.6061	2.8395	0.2865
I-TMCCF-AvgSim	avg	2.6088	2.8425	0.2867
I-TMCCF_SVM	-	2.5164	2.6974	0.2510
I-AE-MCCF	-	2.4430	2.6602	1.0000

Table 5. Comparison of AE-simMCCF with state-of-the-art algorithms on TA data set.

Method	Aggregation Type	MAE	RMSE	Coverage
U-AE-simMCCF	worst-case	0.9461	0.9502	0.1025
U-AE-simMCCF	avg	0.8485	0.8525	0.1034
U-TMCCF-MinSim	worst-case	0.9382	0.9435	0.0894
U-TMCCF-AvgSim	avg	0.9360	0.9412	0.0894
U-TMCCF_SVM	-	0.9348	0.9401	0.0894
U-AE-MCCF	-	0.8946	0.9318	1.0000

Neighborhood-based methods suffer from coverage values due to the lack of neighbors' ratings. Except AE-MCCF, the other baseline methods and AE-simMCCF are based on neighborhood-based method which uses similarities among neighboring users/items and ratings of neighbors during the prediction process. Since AE-MCCF is not a neighborhood-based approach, it does not directly use neighbors' ratings. AE-MCCF utilizes autoencoders to directly compute lost values in users'/items' preferences. As it is shown in Tables 4 and 5, AE-simMCCF provides better coverage results comparing with the other neighborhood-based baseline methods. AE-simMCCF utilizes autoencoders to map the users'/items' preferences into a low-dimensional, latent, and dense space instead of directly computing lost values. Thus, AE-simMCCF computes similarities among users/items utilizing dense features instead of sparse ratings. This is why coverage values provided by AE-simMCCF is better than the ones obtained by the other neighborhood-based baselines. Since the second part of AE-simMCCF based on producing predictions utilizing the computed similarities among users/items over extracted features and the ratings given to the targets by the neighboring users/items, coverage value for AE-simMCCF is lower than the ones for AE-MCCF. Even though AE-MCCF has higher coverage values, it provides less accurate predictions comparing with AE-simMCCF.

In Tables 6 and 7, the running times of algorithms are presented in seconds for YM and TA data sets, respectively. Training parts of the algorithms are performed by using the hyperparameter values that provide the best accuracy and coverage results on a GPU with 4GB RAM and 640 NVIDIA CUDA cores. All other computations are performed on a machine with 32 GB RAM and Intel Xeon E5-1630 CPU (3.70 GHz). AE-simMCCF is a similarity-based MCCF algorithm which consists of off-line and on-line parts. AE-simMCCF creates a model by extracting low dimensional, complex, dense features from high-dimensional, sparse, raw users'/items' preferences with autoencoders in its off-line part. In the on-line part, it computes correlations among users/items utilizing the hidden features obtained by the generated model in the off-line part aiming to produce predictions. The off-line part of a model-based approach is only run once at the beginning in order to generate models to use them in the on-line parts. Then using this model, predictions can be produced when a request occurs for a real-time multicriteria recommender system. Thus, AE-simMCCF provides an acceptable and fast enough prediction runtime as well as better accuracy and coverage results comparing with other baseline algorithms.

Table 6. Running times of algorithms in seconds for YM data set.

Method	Off-line part	On-line part
U-AE-simMCCF	5394.7543	372.1016
U-TMCCF-Sim	-	1671.8210
U-TMCCF_SVM	183.7234	1682.7444
U-AE-MCCF	3334.4855	90.7223
I-AE-simMCCF	4217.7150	149.5060
I-TMCCF-Sim	-	890.9988
I-TMCCF_SVM	134.7366	895.0468
I-AE-MCCF	2409.0059	81.0620

Table 7. Running times of algorithms in seconds for TA data set.

Method	Off-line part	On-line part
U-AE-simMCCF	2550.9133	37.3408
U-TMCCF-Sim	-	98.0602
U-TMCCF_SVM	102.6172	99.0788
U-AE-MCCF	1789.7764	21.6494

6. Conclusions and future work

The increasing amount of criteria causes sparsity to be a main problem of MCCF systems. Especially, similarity-based MCCF approaches suffer from sparsity while finding similar neighbors due to the lack of corated items. In order to prevent the damaging impressions of sparsity on the accuracy of the produced predictions, we proposed a novel similarity-based MCCF approach, AE-simMCCF based on autoencoders. One of the major contributions of AE-simMCCF is dealing with sparsity by extracting nonlinear, latent, dense, and low-dimensional features from raw, high-dimensional, and sparse users'/items' preferences with autoencoders while producing predictions. Similarities among users/items are computed utilizing these features instead of high-dimensional and sparse users'/items' preferences. Several experimental analyses are performed on two real data sets for presenting the efficiency of AE-simMCCF with regard to accuracy and coverage. Experimental outcomes show that AE-simMCCF can better absorb the negative impacts of sparsity issue over the accuracy of the generated referrals compared with baseline methods. Additionally, AE-simMCCF improves coverage values compared to other neighborhood-based MCCF techniques.

Extracting nonlinear features from content information of items and reviews of users by deep learning techniques and integrating those extracted features into the multicriteria prediction process can be considered for our future work.

Acknowledgment

This work is supported by Eskişehir Technical University under grant 19ADP032.

References

[1] Adomavicius G, Kwon Y. New recommendation techniques for multicriteria rating systems. IEEE Intelligent Systems 2007; 22 (3): 48-55. doi: 10.1109/MIS.2007.58

[2] Bilge A, Kaleli C. A multi-criteria item-based collaborative filtering framework. In: Proceedings of the 11th International Joint Conference on Computer Science and Software Engineering; Chonburi, Thailand; 2014. pp. 18-22.

[3] Adomavicius G, Manouselis N, Kwon Y. Multi-criteria recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB (Eds.). Recommender Systems Handbook 1st ed. Springer, 2011, pp. 769-803

[4] Batmaz Z, Yurekli A, Bilge A, Kaleli C. A review on deep learning for recommender systems: challenges and remedies. Artificial Intelligence Review 2019; 52 (1): 1-37. doi: 10.1007/s10462-018-9654-y

[5] Sedhain S, Menon AK, Sanner S, Xie L. Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web; Florence, Italy; 2015. pp. 111-112.

[6] Strub F, Mary J. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In: Proceedings of the NIPS Workshop on Machine Learning for eCommerce; Montreal, Canada; 2015.

[7] Suzuki Y, Ozaki T. Stacked denoising autoencoder-based deep collaborative filtering using the change of similarity. In: Proceedings of the 31st International Conference on Advanced Information Networking and Applications Workshops; Taipei, Taiwan; 2017. pp. 498-502.

[8] Valcarce D, Landin A, Parapar J, Barreiro A. Collaborative filtering embeddings for memory-based recommender systems. Engineering Applications of Artificial Intelligence 2019; 85: 347-356. doi: 10.1016/j.engappai.2019.06.020

[9] Xing S, Liu F, Wang Q, Zhao X, Li T. Content-aware point-of-interest recommendation based on convolutional neural network. Applied Intelligence 2019; 49 (3): 858-871. doi: 10.1007/s10489-018-1276-1

[10] Chen X. The application of neural network with convolution algorithm in western music recommendation practice. Journal of ambient Intelligence and Humanized Computing 2020. doi: 10.1007/s12652-020-01806-5

[11] Khan ZY, Niu Z, Nyamawe AS, ul Haq I. A deep hybrid model for recommendation by jointly leveraging ratings, reviews and metadata information. Engineering Applications of Artificial Intelligence 2021; 97: 104066. doi: 10.1016/j.engappai.2020.104066

[12] Kim D, Park C, Oh J, Yu H. Deep hybrid recommender systems via exploiting document context and statistics of items. Information Sciences 2017; 417: 72-87. doi: 10.1016/j.ins.2017.06.026

[13] Deng X, Wu YJ, Zhuang F. Trust-embedded collaborative deep generative model for social recommendation. The Journal of Supercomputing 2020; 76 (11): 8801-8829. doi: 10.1007/s11227-020-03178-1

[14] Unger M, Bar A, Shapira B, Rokach L. Towards latent context-aware recommendation systems. Knowledge-Based Systems 2016; 104 (C): 165-178. doi: 10.1016/j.knosys.2016.04.020

[15] Du Y-p, Yao C-q, Huo S-h, Liu J-x. A new item-based deep network structure using a restricted boltzmann machine for collaborative filtering. Frontiers of Information Technology & Electronic Engineering 2017; 18 (5): 658-666. doi: 10.1631/FITEE.1601732

[16] Yalcin E, Bilge A. Binary multicriteria collaborative filtering. Turkish Journal of Electrical Engineering & Computer Sciences 2020; 28 (6): 3419-3437. doi: 10.3906/elk-2004-184

[17] Wasid M, Ali R. An improved recommender system based on multi-criteria clustering approach. Procedia Computer Science 2018; 131: 93-101. doi: 10.1016/j.procs.2018.04.190

[18] Wasid M, Ali R. Multi-criteria clustering-based recommendation using mahalanobis distance. International Journal of Reasoning-based Intelligent Systems, 2020; 12 (2): 96-105. doi: 10.1504/IJRIS.2020.106803

[19] Hu Y-C. Neighborhood-based collaborative filtering using grey relational analysis. The Journal of Grey Systems 2014; 26 (1): 99-114.

[20] Hu Y-C, Chiu Y-J, Liao Y-L, Li Q. A fuzzy similarity measure for collaborative filtering using nonadditive grey relational analysis. The Journal of Grey Systems 2015; 27 (2): 93-103.

[21] Kant V, Jhalani T, Dwivedi. Enhanced multi-criteria recommender system based on fuzzy bayesian approach. Multimedia Tools and Applications 2018; 77 (10): 12935-12953. doi: 10.1007/s11042-017-4924-2

[22] Farokhi N, Vahid M, Nilashi M, Ibrahim O. A multi-criteria recommender system for tourism using fuzzy approach. Journal of Soft Computing and Decision Support Systems 2016; 3 (4): 19-29.

[23] Majumder GS, Dwivedi P, Kant V. Matrix factorization and regression-based approach for multi-criteria recommender system. In: Proceedings of International Conference on Information and Communication Technology for Intelligent Systems; Ahmedabad, India; 2017. pp. 103-110.

[24] Ashraf M, Hussain MZ. Multi-criteria decision based recommender system using fuzzy linguistics model for e-commerce. International Journal of Scientific Research in Science and Technology 2018; 4 (5): 61-67. doi: 10.32628/IJSRST1738526

[25] Batmaz Z, Kaleli C. AE-MCCF: An autoencoder-based multi-criteria recommendation algorithm. Arabian Journal for Science and Engineering 2019; 44 (11): 9235-9247. doi: 10.1007/s13369-019-03946-z

[26] Nassar N, Jafar A, Rahhal Y. A novel deep multi-criteria collaborative filtering model for recommendation system. Knowledge-Based Systems 2020; 187: 104811. doi: 10.1016/j.knosys.2019.06.019

[27] Jannach D, Gedikli F, Karakaya Z, Juwig O. Recommending hotels based on multi-dimensional customer ratings. In: Proceedings of the 2012 International Conference on Information and Communication Technologies in Tourism; Helsingborg, Sweden; 2012. pp. 320-331.

[28] Jannach D, Karakaya Z, Gedikli F. Accuracy improvements for multi-criteria recommender systems. In: Proceedings of the 13th ACM Conference on Electronic Commerce; Valencia, Spain; 2012. pp. 674-689.

[29] Hassan M, Hamada M. A neural networks approach for improving the accuracy of multi-criteria recommender systems. Applied Sciences 2017; 7 (9): 868. doi: 10.3390/app7090868

[30] Tallapally D, Sreepada RS, Patra BK, Babu KS. User preference learning in multi-criteria recommendations using stacked auto encoders. In: Proceedings of the 12th ACM Conference on Recommender Systems; Vancouver, BC, Canada; 2018. pp. 475-479.

[31] Kaur G, Ratnoo S. Adaptive genetic algorithm for feature weighting in multi-criteria recommender systems. Pertanika Journal of Science & Technology 2019; 27 (1): 306-314.

[32] Gupta S, Kant V. An aggregation approach to multi-criteria recommender system using genetic programming. Evolving Systems 2020; 11: 29-44. doi: 10.1007/s12530-019-09296-3

[33] Wang S, Yang J, Chen Z, Yuan H, Geng J et al. Global and local tensor factorization for multi-criteria recommender system. Patterns 2020; 1 (2): 100023. doi: 10.1016/j.patter.2020.100023

[34] Hong M, Jung JJ. Multi-criteria tensor model for tourism recommender systems. Expert Systems with Applications 2021; 170: 114537. doi: 10.1016/j.eswa.2020.114537

[35] Kermany NR, Alizadeh SH. A hybrid multi-criteria recommender system using ontology and neuro-fuzzy techniques. Electronic Commerce Research and Applications 2017; 21: 50-64. doi: 10.1016/j.elerap.2016.12.005

[36] Chen Z, Gai S, Wang D. Deep tensor factorization for multi-criteria recommender systems. In: Proceedings of 2019 IEEE International Conference on Big Data; Los Angeles, CA, USA; 2019. pp. 1046-1051.

[37] Fan J, Xu L. A robust multi-criteria recommendation approach with preference-based similarity and support vector machine. In: Proceedings of the 10th International Symposium on Neural Networks; Dalian, China; 2013. pp. 385-394.

[38] Kumar Bokde D, Girase S, Mukhopadhyay D. An item-based collaborative filtering using dimensionality reduction techniques on mahout framework. In: Inproceedings of the 4th Post Graduate Conference for Information Technology; Sangamner, Maharashtra, India; 2015. pp. 2394-0905.

[39] Nilashi M, Esfahani MD, Roudbaraki MZ, Ramayah T, Ibrahim O. A multi-criteria collaborative filtering recommender system using clustering and regression techniques. Journal of Soft Computing and Decision Support Systems 2016; 3 (5): 24-30.

[40] Nilashi M, Bagherifard K, Rahmani M, Rafe V. A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques. Computers & Industrial Engineering 2017; 109: 357-368. doi: 10.1016/j.cie.2017.05.016

[41] Nilashi M, Ibrahim O, Ithnin N, Zakaria R. A multi-criteria recommendation system using dimensionality reduction and neuro-fuzzy techniques. Soft Computing 2015. 19 (11): 3173-3207. doi: 10.1007/s00500-014-1475-6

[42] Goodfellow IJ, Bengio Y, Courville AC. Deep Learning, Adaptive Computation and Machine Learning. MIT Press, 2016.

[43] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning; Lille, France; 2015. pp. 448-456.

[44] Wang H, Lu Y, Zhai C. Latent aspect rating analysis without aspect keyword supervision. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; San Diego, CA, USA; 2011. pp. 618-626.

[45] Ge M, Delgado-Battenfeld C, Jannach D. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In: Proceedings of the 2010 ACM Conference on Recommender Systems; Barcelona, Spain; 2010. pp. 257-260.

[46] LeCun YA, Bottou L, Orr GB, Müller KR. Efficient backprop. In: Montavon G, Orr GB, Müller KR (Eds.). Neural Networks: Tricks of the Trade 2nd ed. Springer, 2012, pp. 9–48.

[47] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015; 521: 436-444. doi: 10.1038/nature14539