An Embedding-based Deep Learning Approach for Movie Recommendation

Conference Paper · June 2020

DOI: 10.1109/ICCES48766.2020.9137998

3 authors, including:

Ram Murti Rawat
Delhi Technological University
17 PUBLICATIONS **24** CITATIONS

SEE PROFILE

Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020)
IEEE Conference Record # 48766; IEEE Xplore ISBN: 978-1-7281-5371-1

An Embedding-based Deep Learning Approach for Movie Recommendation

Ram Murti Rawat
Assistant Professor, Department of COE, Delhi Technological University, Delhi, India.

Vikrant Tomar, Vinay Kumar
Undergraduate Student, Department of COE, Delhi Technological University, Delhi, India.

Abstract - **Recommender systems are employed either as tools or algorithms, whose key task is to efficiently predict the ratings for items and recommend items, using the data generated by users. It assists users in finding items that they will like. Hence, recommendation systems are becoming an essential part of some applications, e-commerce websites, and online streaming services, etc. This paper emphasizes on recommendation system for movies whose main objective is to propose a movie recommendation system through a deep learning technique. The size and complication of websites have increased due to the rapid growth of the internet. On these websites, it has become time-consuming and extremely difficult for the users to find the information that they are searching for. Therefore, a collaborative filtering-based movie recommendation system using deep learning and embedding is proposed. The proposed system is evaluated by calculating the RMSE and MAE values. The proposed method is compared with other machine learning algorithms and some state-of-the-art methods. Our model gave an MAE score of 0.7305 and an RMSE score of 0.9311. It performs better than some previous approaches to collaborative-filtering based movie recommendation.**

Keywords – Recommendation system, artificial neural network, collaborative filtering, embedding, machine learning

I. INTRODUCTION

A recommendation system is now a central part of all e-commerce applications and online entertainment services. The recommendation system gains data about the user's preferences for a particular item (e.g. songs, movies, games, music, etc.) in two different ways. An implicit way of obtaining user data involves observing the user's past behavior such as order history, watched movies, and search queries. Another way of obtaining data is done by gathering the user's previous ratings and likes. The size and complication of websites have increased due to the exponential growth of the internet. On these websites, it has become time-consuming and extremely difficult for the users to find the information that they are searching for. Recommendation systems assist users in finding the information that is within their interests by some interaction with the user. A Recommendation system predicts the rating for items that the user has not rated yet based on his previous ratings and ratings by similar users. Recommendation systems have been a field of research for decades now, but the curiosity remains high because of the plenty of practical applications and the problem abundant domain. Recommender systems deliver tailored information by the familiarity of the user's interests by interaction or experience with that user. As online streaming services are becoming a part of our lives, the recommendation systems have gained importance. With this in mind, our objective is to work on a collaborative filtering-based deep learning strategy for the movie recommendation system. Collaborative filtering is an approach for recommendation systems that depends on the ratings given by a particular user along with the ratings for items given by other similar users. New items of interest can be recommended to the users if the ratings are precisely predicted for an item that the user has not yet rated. In this way, movies are recommended that are different than the user's previously watched movies. Whereas, a Content-Based Recommender only recommends movies based on the similarity between items. The basic idea behind content-based filtering is that if someone likes an item, then he or they will also like a "similar" item. It usually works well when it's easy to determine the properties of each item. Collaborative filtering helps users find diverse items if there are users with similar preferences. Different ways of performing collaborative filtering are present and have been implemented. One way is to create a user profile based on user's ratings and watch history and then find similar users using the k-nearest neighbor algorithm. Similarity can be defined in various ways between the users. The user profile can be defined as a vector of continuous values and then use some comparison measures like cosine similarity. Matrix factorization is one other known method for performing collaborative filtering. In this algorithm, a user-item matrix is composed, then it is factorized into two different matrices such that the inner dimension of the matrices denotes some latent factors. The representation of these factors in the above factorization helps recommend new unique items to the users. Now, deep learning is proving to be quite accurate in recommendation tasks. The popularity increase of deep learning in recommendation systems is due to its state-of-the-art performance and highly accurate predictions. Deep learning better represents the user's interest, features of the item, and past interactions between them as compared to other previous approaches to recommendation systems. A movie recommendation system using the deep learning technique and embedding layers are used.

II. Related Work

Recommender systems are classified into three different techniques that are content-based [5], [6], collaborative filtering [7], [9]-[10], [20], hybrid [13], [14]. Also, clustering algorithms are used for recommendation purposes as it helps cluster similar users into groups. In a movie recommendation model, the system learns the previous user ratings then these ratings help the system to recommend news movies to users using different approaches. Collaborative filtering [7], [9]-[10], [20] is becoming extremely important as this technique is used by many of the recommender systems. Hybrid approaches use the combination of both the content and collaborative-based filtering techniques. Memory-based and model-based are the two main types of collaborative filtering techniques. Memory-based collaborative filtering [4], [16] discoveries similar users in the user space corresponding to a specific user and then recommend movies to that user dynamically. There are two problems related to this approach that are high computational complexity and sparsity of data. An item-based method for collaborative filtering was proposed by different authors to resolve these issues. They calculated the similarity amongst the nearby region items about a fixed entity. This usually uses the k-nearest neighbor algorithm to find groups of similar items. This technique reduced the time complexity to some extent and made accurate predictions. Model-based collaborative filtering [3], [8] approach prepares a model in advance based on the previous item ratings by the user and then the model delivers item recommendation. The model is prepared using different machine learning algorithms. This approach is scalable and solves data sparsity, but time complexity is higher than other techniques. The most general method of achieving collaborative filtering is to use a k-nearest-neighbor approach between users [7]. Clustering also helps in finding similar groups in userspace. K-Means clustering is the most common clustering algorithm. An effective collaborative movie recommendation system was proposed which uses a hybrid K-Means clustering with a Cuckoo search algorithm for optimization [1]. There is another popularly known technique matrix factorization for performing collaborative filtering [19]. Deep learning has reformed various fields of computer engineering, and now it is being extensively used in recommendation systems [2], [15], [20]. Wang et al. [20] built a collaborative model for recommendation using deep learning which works both on content data and rating matrix. A recommendation system is proposed by Elkahky et al. [15] using deep learning which uses search queries and web browsing history provided by users. But there is a limitation of this method, as it needs history and search queries which are not accessible all the time. Wei et al. [21] proposed a movie recommendation model using deep neural networks for prediction of movie ratings by mining the characteristics of items and is particularly used to cold start items. A deep learning technique for collaborative filtering based on autoencoders precisely predicts the movie ratings [2]. There is another extraordinary approach to applying collaborative filtering known as matrix completion (MC). In this technique, the rating matrix is approximated to a low-rank rating matrix. This is a mathematically complex approach. In 2014, Kalofolias et al [23] introduced a regularization of the MC model known as the geometric matrix completion (GMC) model by adding additional information in the form of user and item graphs. In [11], a more efficient graph-regularized method for matrix completion was proposed using alternating least squares optimization method (GRALS). More understandable and simpler approach with Embedding layers and deep neural networks is proposed.

III. Proposed Work

A. Dataset Description

MovieLens 100K data set is used in the proposed model. The MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota [22]. This data set contains 100,000 ratings in the range of 1 to 5 for 1682 movies from a user base of 943. The data set has been preprocessed and includes only those users who have rated more than 20 different movies. It also contains simple demographic information of users (age, gender, occupation, zip) and information of movie genres. 80% of the dataset will be used for training and rest 20% will be used for testing. The data of users who did not have complete information were removed. **u.data** file is the main file used in building the model. It contains the rating data from a user for a particular movie. It has 4 columns UserID, MovieID, Rating, Timestamp.

The distribution of ratings in range 1 to 5 is shown in Fig 1.

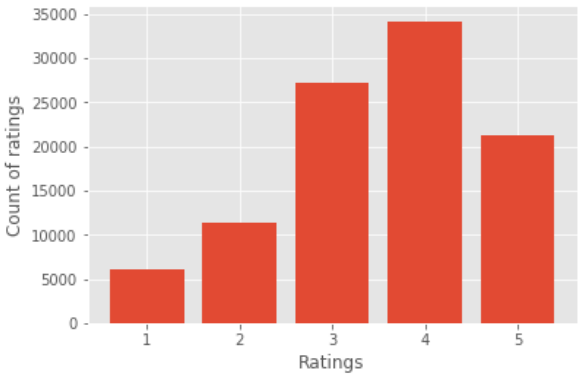

Fig. 1. Distribution of ratings

B. The architecture of the Model

After observing and pre-processing the dataset, 943 unique users and 1,682 unique movies is detected in the MovieLens dataset. However, the users and movie IDs are passed directly into our deep learning network because the model will try to find the relation between IDs that are not present. Our algorithm uses the embeddings to convert these IDs to a continuous vector of fixed size as machine learning algorithms

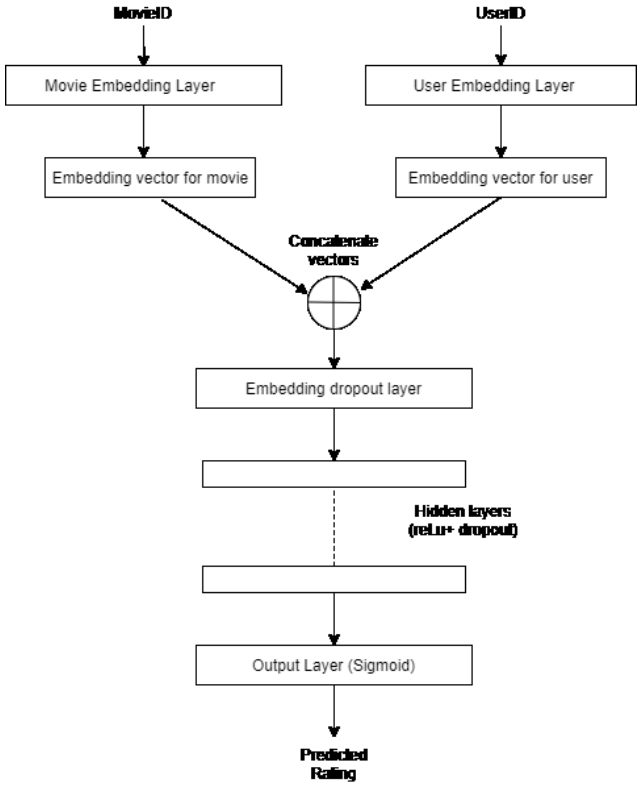

Fig. 2. Structure of network

expects numerical values as input. An embedding vector will be produced for a userID and one embedding vector for a movieID from the embedding space of 943 and 1682 respectively. Then, the vectors are concatenated into one and then passed to the dropout layer. The dropout layer is used to avoid the over fitting of a model. In each training epoch, dropout randomly sets outgoing edges of the layer to 0 with some predefined probability. Next, several fully-connected hidden layers are added with a dropout probability and reLU activation function. Each hidden layer can have a different value of output features and dropout probability. The input features are decided based on the output features of the previous layer. Finally, the predicted rating should be sent as an output for the input user and model IDs. For this, an output layer with a sigmoid activation function is employed and later rescale in the range of 1 to 5. The weights of the neurons in each layer are updated in each training epoch with a back propagation algorithm. The structure of our proposed network used as our model for predicting ratings is as shown in Fig.2.

C. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) is a category of artificial neural networks (ANN). ANN a technique encouraged by the biological neuron network that makes up the human brain. This network learns to accomplish tasks by examining examples, without being explicitly programmed. The MLP is made up of several layers (an input layer, an output layer, and one or more hidden layers) that contains nodes with different activation functions. MLPs are fully connected, which means each node in a layer is connected to every node of the immediate next layer. The input layer receives the input data. No information processing is done on this layer. The received input is conveyed to the following layer, the hidden layer. The data directed from the input layer is processed in the hidden layers and passed to the next layer i.e. output layer. In MLP, many artificial neurons and hidden layers can be obtained. MLP can solve more complex tasks by adding a large number of neurons and layers for processing. However, it will increase the computation time. The output layer is the final layer, every node of the last hidden layer is connected to the final output layer, and hence, the result of the processed data from the hidden layer is produced at this layer. Fig. 3 shows a simple multi-layer perceptron. The advantages of MLP are:

1) It can solve highly complex and complicated problems.
2) It can handle missing data.
3) It learns the weights on its own using the backpropagation algorithm. Thus, it solves a wider range of problems than the algorithms.

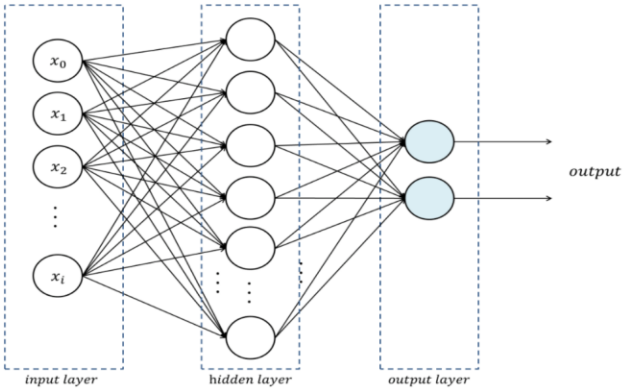

Fig. 3. Simple Multi-Layer Perceptron Model

D. Embeddings

An embedding is a conversion of discrete or categorical variables into a vector of continuous numerical values. In the view of deep learning, embeddings are multi-dimensional vector with learned continuous values that represent discrete or categorical variables. The embeddings are used in reducing the dimension of categorical variables and conveniently represent categories in a transformed space that can be used as input to neural networks and other machine learning algorithms. Embeddings have 3 basic purposes:

1. Finding nearest neighbors in the embedding space. It can be used in making recommendations based on item properties and user interests.
2. It can be used as input to various machine learning algorithms for a supervised learning problem.
3. Used in visualizing of relations between categorical variables and their characteristics.

Neural network embeddings overcome the problems of another simple method for transforming categorical variables, that is, one-hot encoding.

E. Root Mena Square Error

Root Mean Square Error (RMSE) is a common evaluation metric used in calculating the accuracy of a system. RMSE is calculated by taking the square root of the mean squared errors. The mean value is calculated by dividing the sum of the squared difference between actual and predicted output by the number of samples. Equation (1) represents the RMSE value.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{N}(predicted_i - actual_i)^2}{N}} \quad (1)$$

N is the total number of instances. The smaller RMSE value means better performance for the system. Thus, a system which produces smaller RMSE value will provide better recommendations.

F. MAE

Mean Absolute error (MAE) is a similar evaluation metrics as RMSE. It is calculated by the mean of absolute error values. Absolute error is the positive value of the difference between the predicted output and the actual output. The equation for MAE is mentioned in (2).

$$\text{MAE} = \frac{\sum_{i=0}^{N}|predicted_i - actual_i|}{N} \quad (2)$$

IV. Experimental Result & Analysis

First, some pre-processing of the data is done on the dataset for the desired input for machine learning algorithms. The dataset is divided into two parts in which 80% is for training data and the rest 20% is test data for verification of the result. Our network has 4 basic parameters. To check the performance of our recommender system model, the RMSE and MAE is calculated corresponding to various values of these parameters.

Table 1. RMSE and MAE value for different parameter values.

k	hidden	emb_dropout	Dropouts	MAE	RMSE
100	[300, 200, 100]	0.02	[0.25, 0.25, 0.25]	0.7375	0.9371
100	[200, 100]	0.01	[0.15, 0.15]	0.7324	0.9321
75	[100, 200]	0.02	[0.25, 0.25]	0.7349	0.9353
50	[200, 300, 400]	0.02	[0.20, 0.20, 0.20]	0.7374	0.9355
50	[400, 300]	0.02	[0,25, 0.25, 0.25]	0.7319	0.9315
125	[100, 200]	0.02	[0.20, 0.25]	0.7337	0.9340
125	**[500, 400, 300]**	**0.01**	**[0.15, 0.15, 0.15]**	**0.7305**	**0.9311**
200	[300, 200, 300]	0.02	[0.25, 0.25, 0.20]	0.7353	0.9356
150	[200, 300, 200]	0.02	[0.25, 0.25, 0.25]	0.7402	0.9345

The size of embedding vectors for each unique user and movie is denoted by k. The hidden parameter is a list of values each corresponds to the number of output features of the hidden layers. The emp_dropout represents the dropout probability of the embedding dropout layer. The dropout parameter is a list of dropout probabilities for each hidden layer. Table 1 shows the RMSE and MAE for different parameter values.

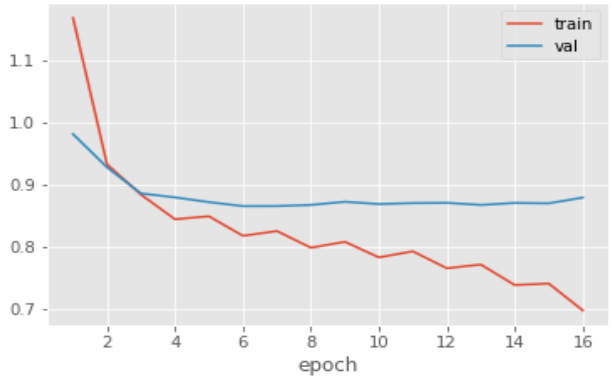

Fig 4. Training and Validation Error vs the number of Epoch

Fig. 4 shows how the training and validation errors are changing with increasing training epochs of the model. The training error is decreasing continuously but the validation error first decreases till epoch 6 but almost stays the same after that. Table 2 shows the corresponding RMSE and MAE values for a single parameter configuration but only changing k value, i.e. the size of output embedding from the embedding layer. The parameter configuration for the comparison is two hidden layers have [400, 300] output features, emp_dropout value is 0.02, dropout values are [0.25, 0.25].

Table 2. Evaluation metric values for different k.

k	*MAE*	*RMSE*
50	0.7359	0.9315
75	0.7330	0.9313
100	0.7370	0.9339
125	0.7315	0.9308
150	0.7338	0.9313
175	0.7328	0.9324
200	0.7331	0.9306

The proposed algorithm is compared with different algorithms for collaborative filtering. KNN is the K-nearest neighbor algorithm, which is a supervised machine learning algorithm for classification. PCA (Principal Component Analysis) is a technique for dimensionality reduction. A genetic algorithm (GA) is a nature-inspired algorithm. SOM refers to Self-organizing Map which maps a higher dimension input into a lower dimension. Clustering is an unsupervised learning algorithm that groups together similar items. GMC (geometric matrix completion) and GRALS (Graph regularized matrix completion method) are both state-of-the-art methods used for

collaborative filtering. KMEANS CUCKOO is a combination of K-Means clustering and Cuckoo search optimization technique. EMBEDDING+ANN is our proposed approach which uses Artificial Neural networks and Embedding. Table 3 shows the MAE and RMSE values for different algorithms. Our algorithm performs better than all of the algorithms in terms of RMSE value and has a larger value of MAE than just the KMEANS CUCKOO algorithm. Fig. 5 and 6 show the plot for different algorithms vs RMSE and MAE respectively.

Table 3. RMSE and MAE values for different algorithms

ALGORITHM	MAE	RMSE
KNN	-	1.124
PCA-GAKM	0.94	-
PCA-SOM	0.98	-
SOM-CLUSTER	0.75	-
GAKM-CLUSTER	0.76	-
KMEANS CUCKOO	0.68	1.2363
GMC	-	0.996
GRALS	-	0.945
EMBEDDING + ANN	**0.7305**	**0.9311**

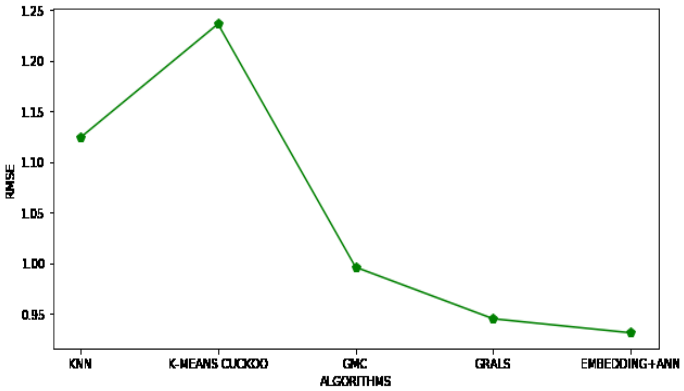

Fig 5. RMSE vs Different Algorithms

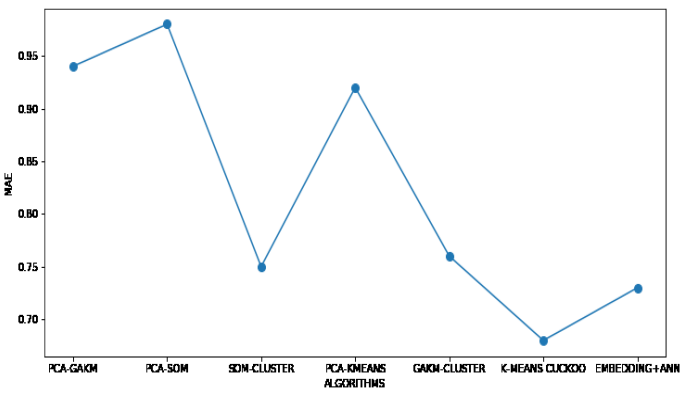

Fig 6. MAE vs Different Algorithms

V. CONCLUSION

In this paper, a collaborative recommender system using an embedding-based deep learning approach is implemented. MovieIDs and UserIDs are compared at the embedding layer to a vector of continuous values of a specific size. Then we add various layers with certain weights for processing of our input data. The hidden layers have a reLu activation function and have a dropout probability for better results. Finally, an output layer with sigmoid activation is added for the prediction of ratings. The movie ratings are predicted for the users in test data and calculated the evaluation metrics based on those predictions. Based on the evaluation metrics, our model gives better results than other machine learning algorithms in terms of MAE and RMSE values. Also, the proposed technique is compared with some state-of-the-art methods and provided more accurate results. While recommending new movies to a user, the rating based on the model is predicted and recommend the movie with higher ratings to the user. The deep learning technique with embedding layers has an additional advantage of scalability. In the future, a hybrid model will be built to give better recommendations. A combination of various machine learning algorithms can help make the recommendation system more efficient and accurate. Ensemble learning is also one of the techniques that will exploit in movie recommendation systems.

REFERENCES

[1] Katarya R, Verma OP. An effective collaborative movie recommender system with cuckoo Search. Egyptian Informatics Journal 18 (2017) 105–112.
[2] Jeffrey Lund, Yiu-Kai Ng. Movie Recommendation using deep learning approach. IEEE Computer Society, 2018.
[3] Lu J, Mao M, Wu D, Zhang G, Wang W. Recommender system application developments: a survey. Decision Support Sys 2015; 74:12–32.
[4] Gutiérrez a, Bobadilla J, Hernando a, Ortega F. Recommender systems survey. Knowledge Based System 2013; 46:109–32.
[5] Semeraro G, De Gemmis M, Lops P. Content-based recommender systems: state of the art and trends. Recommender System Handbook 2011:1–33.
[6] Billsus D, Pazzani MJ. Content-based recommendation systems 2007:325–41.
[7] W. Croft, T. Strohman, and D. Metzler. Search Engines: Information Retrieval in Practice. Addison Wesley, 2010.
[8] Zhang R, Wang Y, Bao H, Liu X, Sun H. Recommender systems based on ranking performance optimization. Front Computer Sci China 2015;1–11.
[9] Su X, Khoshgoftaar TM. A survey of collaborative filtering techniques. Adv Artificial Intelligence 2009; 2009:1–19.
[10] Zeng D, Huang Z, Chen H. A comparison of collaborative-filtering algorithms for e-commerce. IEEE Intelligent Systems 2007.
[11] Nikhil Rao, Inderjit S. Dhillon, Pradeep K. Ravikumar, Hsiang-Fu Yu. Collaborative filtering with graph information: Consistency and scalable methods. In C. Cortes, R. Garnett, N. D. Lawrence, M. Sugiyama, D. D. Lee, and editors, Advances in Neural Info Processing Syst 28, pages 2107–2115. Curran Associates, Inc., 2015.
[12] Ekstrand MD. Collaborative filtering recommender systems. Found Trends_Human-Comput Interact 2010; 4:81–173.
[13] Burke R. Hybrid web recommender systems. Adapt Web 2007.
[14] Kim BM. Clustering approach for hybrid recommender system. In: Proc IEEE/WIC Int Conf Web Intell (WI 2003). p. 33–8.
[15] A. Y. Song, Elkahky, and X. He. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. In WWW, pages 278–288, 2015.

[16] Chen G, Chen L, Wang F. Recommender systems based on user reviews: the state of the art. User Model. User-Adapt Interact 2015; 25:99–154.
[17] Jannach D, Nilashi M, Ithnin N, Bin Ibrahim O. Clustering and regression based multi-criteria collaborative filtering with incremental updates. Inf Sci(Ny) 2014;293:235–50.
[18] Tsapatsoulis N, Georgiou O. Improving the scalability of recommender systems by clustering using genetic algorithms. Lecture Notes Comput Sci 2010; 6352:442–9.
[19] C. Volinsky, R. Bell, Y. Koren. Matrix Factorization Techniques for Recommender Systems. Computer, 42(8):30–37, 2009.
[20] N. Wang, D.-Y. Yeung, H.Wang. Collaborative Deep Learning for Recommender Systems. In KDD, pages 1235–1244, 2015.
[21] J. Wei, K. Chen, J. He, Z. Tang, Y. Zhou. Collaborative Filtering and Deep Learning Based Recommendation System for Cold Start Items. Expert Systems with Applications, 69:29–39, 2017.
[22] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS).
[23] Vassilis Kalofolias, Pierre Vandergheynst, Michael Bronstein, Xavier Bresson. Matrix completion on graphs. arXiv preprint arXiv:1408.1717,2014.