Information Sciences 503 (2019) 521–532

Contents lists available at ScienceDirect

Information Sciences

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

# Deep Latent Factor Model with Hierarchical Similarity Measure for recommender systems

Jiayu Han[a,b,c], Lei Zheng[b], He Huang[b], Yuanbo Xu[a,c], Philip S. Yu[b], Wanli Zuo[a,c,*]

[a] Jilin University, Changchun, Jilin, China
[b] University of Illinois at Chicago, IL, USA
[c] Key Laboratory of Symbolic Computation and Knowledge Engineering for the Ministry of Education, Jilin University, Changchun, Jilin, China

ARTICLE INFO

*Article history:*
Received 4 January 2019
Revised 4 July 2019
Accepted 6 July 2019
Available online 8 July 2019

*Keywords:*
Recommender systems
Latent Factor Model
Hierarchical similarity measure
Deep learning

ABSTRACT

Latent Factor Model(LFM), as an effective feature mapping method, is widely applied in recommender systems. One challenge of LFM is previous methods usually use the inner product to calculate the similarity between users and items in the latent space, which cannot characterize different impacts of various latent factors. Another challenge is the performance of LFM will be negatively affected when facing data sparsity problem. In this paper, we propose a model named DLFM-HSM(Deep Latent Factor Model with Hierarchical Similarity Measure) to overcome the challenges above. More specifically, we introduce a hierarchical similarity measure to calculate an impact score which can better represent the similarity between a user and an item than the inner product. Also, in order to ease the data sparsity problem, we extract latent representations of users and items using deep neural networks from items' content information instead of only from user-item rating records. By representing users with items they purchased, our model guarantees that users and items are mapped into a common space and thus they are directly comparable. Extensive experiments on five real-world datasets show significant improvements of DLFM-HSM over the state-of-the-art methods and demonstrate the effectiveness of our model for alleviating the data sparsity problem.



## 1. Introduction

People are often inundated with information and choices in the big data era. Recommender Systems (RS), as an effective way to tackle the information overload problem, provide people with valuable suggestions and help people focus on the items they may be interested in. It has been developed for a variety of applications, like e-government, e-business, e-commerce, e-learning, etc. [21]

RS can be clustered into three common main categories. Content-based Filtering, Collaborative Filtering,and Hybrid Filtering [3]. Content-based Filtering (CBF) makes recommendations according to the previous users' choosing. Collaborative Filtering (CF), the state-of-the-art method to build RS, predicts a user's rating or preference on a candidate item based on other similar users and items. Hybrid Filtering combines CBF and CF together aiming to exploit the merits of each technique.

* Corresponding author at: Jilin University, Changchun, Jilin, China.
*E-mail addresses:* jyhan15@mails.jlu.edu.cn (J. Han), wanli@jlu.edu.cn (W. Zuo).

https://doi.org/10.1016/j.ins.2019.07.024
0020-0255/

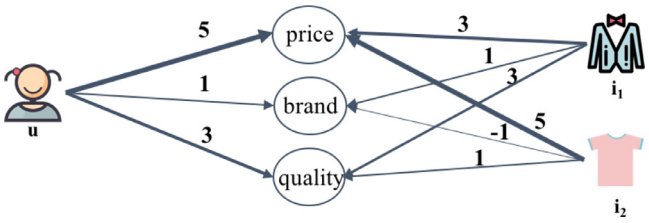



**Fig. 1.** An example which illustrates the challenge of capturing users' preference on difference latent factors using the traditional Latent Factor Model (LFM).

Recently, fuzzy tools and deep learning techniques are becoming popular and effective, and fuzzy-based recommender systems [34] and deep learning based recommender systems [35] attract many researchers' attention.

Among various CF methods, Latent Factor Model (LFM) is the most popular one due to its effectiveness [17,25]. The traditional method of LFM first maps both users and items into a shared latent space and then models users' preferences on items by computing the inner products of their latent factor vectors. The traditional LFM has its advantage in efficiency but there are some challenges in its processing procedure. The inner product method is one of the challenges. Let us first recall the working mechanism of the inner product method. It first multiplies the corresponding elements (latent factors) of two latent factor vectors and then sums these values together to obtain a score between a user and an item. Further to explain, the element-wise multiplication on two latent factor vectors (one is for representing a user and another one is for representing an item) represents the interactions between a user and an item on these latent factors and the summation means integration of these interactions. We call this score as the *impact score*. By analyzing the above procedure, we find that the traditional LFM is limited by the inner product. Because it treats different latent factors in the same way, while different latent factors may have different significance for different users. Let us give a specific example to better explain this challenge. As shown in Fig. 1, we assume that "*price*", "*brand*" and "*quality*" are the latent factors in the latent space, and the number on each line represents the extent of a user's preference or an item's property on a certain latent factor. Let us focus on the user $u$ and two candidate items $i_1$ and $i_2$, and consider which one should be recommended to the user $u$. According to Fig. 1, we can easily write the latent vectors of the user and items as follow: $u = [5, 1, 3], i_1 = [3, 1, 3], i_2 = [5, -1, 1]$ and have the inner product between them as $u \cdot i_1 = 25 < u \cdot i_2 = 27$. If we use the traditional LFM method, item $i_2$ will be recommended to the user because of its higher score than item $i_1$. In fact, compared to item $i_2$, item $i_1$ may be more suitable for the user, because $i_1$ matches the user well in each factor and it achieves a good balance between "*price*" and "*quality*" that the user is interested in, while item $i_2$ only matches the user on the "*price*" factor.

The core reason for the above challenge is that the traditional LFM uses the inner product to model interactions between a user and an item, and combines the interactions between different factors with the same weight to get the impact score. If the interaction on a certain factor is too large or too small, the impact score will have a bias influenced by this factor. In order to face this challenge, some researchers use factorization machines [28] to model the importance of all interactions between latent factors. Attentional mechanisms are also proposed to weight interactions between latent factors [5,10]. Despite they have a good performance when meeting this challenge, they bring many parameters into the learning process. Different from them, we propose a heuristic similarity metric called *Hierarchical Similarity Measure* (HSM), which is able to distinguish the different significance of different interactions between users and items. In addition, our proposed HSM is a parameter-free method and thus it does not require extra training as previous methods.

Another challenge is the data sparsity. It deteriorates the accuracy of the rating prediction for traditional LFM. Some scholars aim to use content-based filtering methods [24,26] or use hybrid filtering methods which combine content-based filtering with collaborative filtering methods [7,9,13] to alleviate this problem. Through analyzing the real world data, we observe that items are often associated with descriptions which can describe items' properties. and can also describe users' interests. Based on this observation, we propose a novel method to represent the users and the items to solve this challenge. We utilize deep neural networks to learn both users' interests and items' properties from items' descriptions, which can get more representative features than other methods which do not use deep neural networks. As we know, each user has rated more than one item, therefore, we propose to use latent matrices instead of latent vectors to represent the users. In addition, this method can also ensure that users and items are mapped into a common feature space and they can be compared directly.

The main contributions of this paper are summarized as follows:

- We propose a Hierarchical Similarity Measure which is a parameter-free method and can better integrate the interactions between the users and the items.
- We utilize deep neural networks to learn both users' interests and items' properties from items' descriptions, which can better alleviate the data sparsity problem. Furthermore, we propose to use the latent factor matrices instead of using the latent factor vectors in order to better represent the users.

**Table 1**
Symbol appointments.

| Symbol | Explanation |
|---|---|
| $\mathcal{U}$ | Set of users |
| $\mathcal{I}$ | Set of items |
| $\mathcal{I}_u$ | Set of items which are rated by user u |
| $\mathcal{I}_u^-$ | Set of items which are not rated by user u |
| $D$ | The set of items' descriptions |
| $\mathcal{R}$ | Set of user and item pairs with observed rating |
| $\mathbf{P_u}$ | Latent matrix of user $u$ |
| $\mathbf{q_i}$ | Latent vector of item $i$ |
| $\hat{r}_{ui}$ | The predicted rating of user u for item i |
| $r_{ui}$ | The actual rating of user u for item i |
| $\odot$ | An defined operation on a matrix and a vector |
| $\oplus$ | An defined operation on a vector |

- We conduct extensive experiments on five datasets to validate the effectiveness of our model. The results show that our model outperforms the state-of-the-art baselines and can achieve a better performance than baselines when facing the data sparsity problem.

The rest of paper is organized as follows: We first give a formal problem definition and a preliminary introduction in Section 2. Then the proposed model DLFM-HSM is described in detail in Section 3 and the extensive experiments and the discussions are presented in Section 4. Next, we briefly review the recent and classic works about latent factor model in Section 5. Finally, we give a conclusion about our paper in Section 6.

## 2. Preliminaries

### 2.1. Problem definition

Our paper is centered on the rating prediction task, which predicts the ratings on unseen items based on users' previous transactions using side content information. It can be formulated as follows. Given a user $u \in \mathcal{U}$, $\mathcal{I}_u$ represents the set of items that user $u$ has rated, and $\mathcal{I}_u^-$ stands for the set of unrated items of user $u$, and $r_{ui} \in \mathcal{R}$ denotes the rating that the user $u$ gave to the item $i$. We aim to utilize both the content information of $\mathcal{I}_u$ and the rating records $\mathcal{R}$ to predict the rating that the user $u$ may give to an item in $\mathcal{I}_u^-$. The symbols we use to introduce our model are listed in Table 1.

### 2.2. Latent Factor Model

Suppose there are $n$ users $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$ and $m$ items $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$. For each user $u \in \mathcal{U}$ and each item $i \in \mathcal{I}$, their corresponding latent vectors are $\mathbf{p_u} \in \mathbb{R}^k$ and $\mathbf{q_i} \in \mathbb{R}^k$ which are learned from user-item ratings. And the predicted rating on an unrated item can be modeled as the inner product of the two latent vectors:

$$\hat{r}_{ui} = \mathbf{p_u}^\top \mathbf{q_i}.$$

To learn the latent factor vectors, the following objective function is often employed:

$$\min_{\mathbf{q}_*, \mathbf{p}_*} \sum_{(u,i)\in\mathcal{R}} (r_{ui} - \mathbf{p_u}^\top \mathbf{q_i})^2 + \lambda(\| \mathbf{p_u} \|^2 + \| \mathbf{q_i} \|^2),$$

where $\mathcal{R}$ represents the set of user and item pairs with observed rating, $r_{ui}$ represents the actual rating of user $u$ on item $i$, and $\lambda$ is used to control the strength of regularization in order to prevent the overfitting problem.

## 3. Deep Latent Factor Model with Hierarchical Similarity Measure

In this section, we introduce our model – **D**eep **L**atent **F**actor **M**odel with **H**ierarchical **S**imilarity **M**easure (DLFM-HSM) in detail. We first describe the overall framework of our model and then discuss the details about how to extract latent factors from content information and how to build representations for users and items respectively. Finally, we will cover the process of measuring the similarity between users and items and the proposed hierarchical similarity measure will be presented at the end of this section.

### 3.1. DLFM-HSM framework

The framework of DLFM-HSM is illustrated in Fig. 2. Intuitively, users' interests are hidden in the items they have rated, and therefore, we use the content information of rated items to represent users' preferences. As shown in Fig. 2, DLFM-HSM takes one candidate item $i$'s description and a set of descriptions of some other items which are rated by user $u$ as input

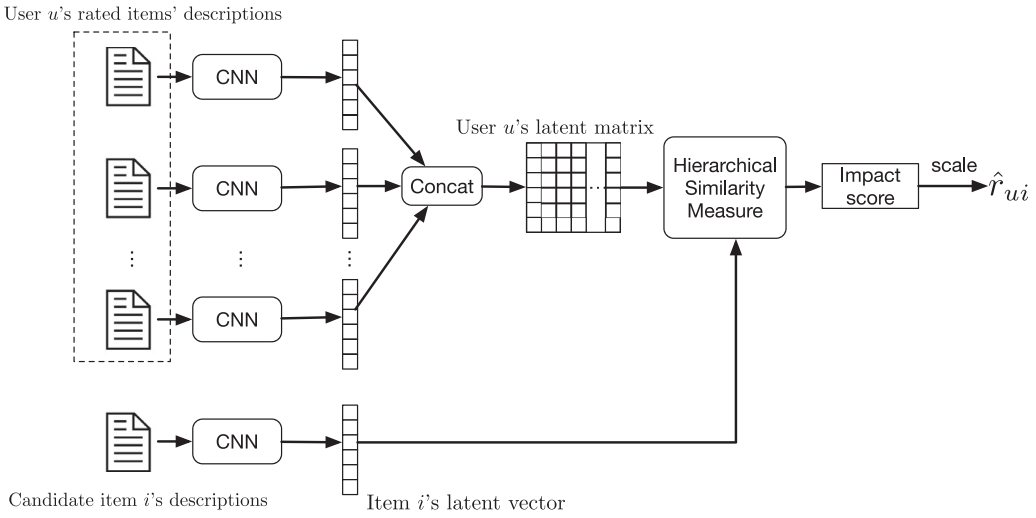



**Fig. 2.** The architecture of our proposed model: Deep Latent Factor Model with Hierarchical Similarity Measure (HSM). All CNNs share the same set of parameters. The structure of CNN and HSM are shown in Figs. 3 and 4 respectively.

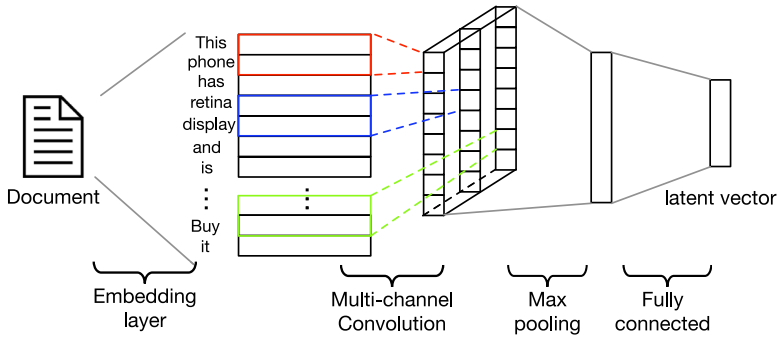



**Fig. 3.** The structure of the convolutional neural network [15] used in our model.

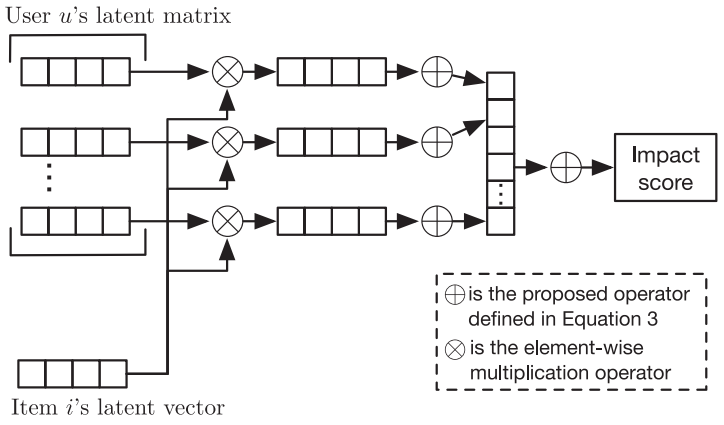



**Fig. 4.** The structure of Hierarchical Similarity Measure $\odot$.

data. For each piece of description, we use a classic convolutional neural networks model (CNN)[15] to process its content and obtain a latent vector for this description. The details of how to construct users' and items' latent representations are described in Section 3.2. By using CNN, we can get an abstract representation for each item and then we put the latent vectors of a user's rated items together into a matrix to represent a user's profile. In order to compute the similarity between a user matrix and an item vector, we propose a Hierarchical Similarity Measure, denoted by $\odot$, to get an impact

score between a user and an item, and then based on this impact score we further get the predicted rating for a given item. The details of the Hierarchical Similarity Measure is presented in Section 3.3.

### 3.2. Learning latent features for items and users

Traditional LFM [18] usually uses matrix factorization to learn the latent factors for items and users, which does not make use of any items' content information. In order to fully take advantage of the content information, we use convolutional neural networks model [15] to learn the latent expressive semantic features for each item from its description, which can mitigate the data sparsity problem by extracting latent factors from content information. The process of how to learn a latent vector is illustrated in Fig. 3. It contains four hidden layers: Embedding layer, Convolutional layer, Max-pooling layer, and Fully Connected layer. The input to this model is a set of descriptions $D = \{d_1, d_2, \ldots, d_{|D|}\}$.

First, each description will be fed into the Embedding Layer which can learn the embedding representation for each word. Then the embedding vectors are fed into the Convolutional Layer with different kinds of filters. It can extract contextual features among words. Next, the features will be put into the Max-pooling Layer which is used to further choose more important features. Finally, the chosen features will be fed into the Fully Connected Layer to get the final latent vector for each item.

A user's interests can be depicted from several aspects and each aspect is corresponding to an item which the user has rated. Using items' descriptions to construct users' latent vectors enables mapping users and items into the same feature space. And then, in this new feature space, we can compare their feature vectors directly.

For each user $u$ with rated history $\mathcal{I}_u$, we use a latent matrix instead of a latent vector to represent a user. It can be represented as following:

$$\mathbf{P_u} = [\mathbf{q_i}],$$

where $i \in \mathcal{I}_{ud}$; each column $\mathbf{q_i}$ is the latent factor vector of the item $i$. Previous methods such as the inner product always operated on vectors, so how to compute the similarity between one user's matrix and one item's vector is a challenge. One simple way is to obtain the latent vector for the user $i$ by averaging all the latent vectors of his rated items:

$$\mathbf{p_u} = \frac{1}{|\mathcal{I}_{ud}|} \sum_i \mathbf{q_i}$$

where $i \in \mathcal{I}_{ud}$. However, as discussed in the introduction, a user's interests can be diverse and the simple way cannot characterize these varieties. In the next section, we will introduce a hierarchical way to compute the similarity between a user's matrix and an item's vector.

### 3.3. Hierarchical Similarity Measure

Until now, we have obtained the final latent representation for users and items. The next step in traditional LFM is to model the interactions between users and items to predict ratings for candidate items. Given the user $u$'s latent matrix $\boldsymbol{P_u}$ and the candidate item $i$'s latent vector $\mathbf{q_i}$, we cannot directly use the inner product to model the interaction between them directly. To this end, we define a Hierarchical Similarity Measure method containing two-level hierarchy to calculate the impact score between a matrix and a vector and its structure is shown in Fig. 4. The first level is used to calculate the impact score of the interactions between each item vector in the matrix and the candidate item vector. The second level is used to get a final impact score which can characterize the user's preference on the candidate item.

From the Fig. 4, we see that there are two operations. One is a simple element-wise multiplication operator $\otimes$, the other one is a new operation $\oplus$ defined in our paper and it is also the key element of the Hierarchical Similarity Measure method. There are two meanings of the 'hierarchical': the structure is hierarchical and the new operation is hierarchical. We first talk about what is the new hierarchical operation and how to use the new operation to measure the similarity between two vectors.

Previous works [14,18] often use the inner product of two latent vectors to model the interactions. Let us first recall the basic principle of the inner product. Given two vectors $\mathbf{x}$ and $\mathbf{y}$, the inner product can be written as follow:

$$\mathbf{x} \cdot \mathbf{y} = \sum_j v_j = \sum_j x_j \times y_j.$$

It performs an element-wise multiplication between two vectors to obtain an intermediate vector $\mathbf{v}$. After that, it adds each element in the vector $\mathbf{v}$ to get a final value. We call the vector $\mathbf{v}$ as the *interaction vector* because each element $v_j$ in vector $\mathbf{v}$ can be treated as an interaction between two vectors on the latent factor $j$ and the value of $v_j$ determines the relevance between two vectors on the latent factor $j$. Therefore, the main idea of the inner product is to measure the interaction between two vectors on each latent factor and integrate these interactions together to give a final evaluation. However, the inner product gives each interaction the same weight, which is easily affected by a certain latent factor. If an interaction on one latent factor is too high or too low, the final evaluation will be decided by this latent factor and omit other interactions.

In order to alleviate the problem above, we define a new operation $\oplus$ on an interaction vector $\mathbf{v}$ instead of simply adding the elements in $v$ as:

$$\oplus\mathbf{v} = \sigma\left(\sigma\left(\sum_j v_j\right) + \sigma(\max_j v_j) - (1 - \sigma(\min_j v_j))\right), \tag{1}$$

where $\sigma$ represents the *sigmoid* function. Eq. (1) contains three parts and each part is squeezed to the interval (0,1) by using sigmoid functions. Eq. (1) generates a score which is used to characterize the interactions between two latent vectors from three aspects. The first part, $\sigma(\Sigma_j v_j)$, like the inner product, considers the interaction in each latent factor and sum them to gain a basic score for measuring the overall impact. To better explain the meaning of the rest parts in Eq. (1), we use the example in Fig. 1. We find that the interaction (derived from the element-wise multiplication) between the user $u$ and the item $i_2$ on the factor "*price*" is far greater than the interaction on the factor "*brand*". So if we treat the interactions equally, the negative impact of the weak interaction will be omitted and it will further affect the overall evaluation. To capture the negative impact of this weak interaction, we introduce an element: $\sigma(\min_j v_j)$which represents the weakest interaction, reflecting the most negative aspect, e.g., the "*brand*" factor on item $i_2$ (Fig. 1). Because the value of $\sigma(\min_j v_j)$ will be small if the interaction is weak, we use $1 - \sigma(\min_j v_j)$ to represent its negative impact to the whole and we call it as the *weak impact*. A low value on the weakest interaction means that the user may have no interest in this item, so we subtract the *weak impact* from the basic score. Similar to this, we introduce the element *strong impact:* $\sigma(\max_j v_j)$ which represents the impact of the strongest interaction to distinguish from the case with many moderate interactions over a large number of factors, instead of a strong single factor. The strongest interaction means that the user may have a strong interest in this item, so we add the *strong impact* into the basic score. And then, we get the final impact score.

Based on $\oplus$, the extension of the inner product can be written as follows:

$$\mathbf{x} \bullet \mathbf{y} = \oplus\mathbf{v} = \sigma\left(\sigma\left(\sum_j x_j * y_j\right) + \sigma(\max_j x_j * y_j) - (1 - \sigma(\min_j x_j * y_j))\right), \tag{2}$$

where the $\mathbf{v}$ is the interaction vector which is equal to the element-wise multiplication between the $\mathbf{p}$ and $\mathbf{q}$. We use the new inner product to calculate the similarity between the user $u$ and item $i_1$ and item $i_2$ (the example in introduction) separately and now we can get $\mathbf{u} \cdot \mathbf{i}_1 > u \cdot \mathbf{i}_2$.

Based on the new operation $\oplus$ and the new inner product $\bullet$, the Hierarchical Similarity Measure denoted as $\odot$, can be formalized as follows:

$$\mathbf{P}_{\boldsymbol{u}} \odot \mathbf{q}_{\mathbf{i}} = \oplus[\mathbf{p}_{\mathbf{uj}} \bullet \mathbf{q}_{\mathbf{i}}],$$

where $\mathbf{p}_{\mathbf{uj}}$ represents the column vector of the matrix $\mathbf{P}_u$, "$[\cdot]$" is an operation which constructs a new vector by concatenating all elements in it. It first calculates the interaction between each vector in the matrix $\boldsymbol{P_u}$ and the candidate item vector $\mathbf{q}_{\mathbf{i}}$ by the new inner product function. After that, we get $m$ impact scores and we integrate these scores together to construct a new vector. The new vector contains the interactions between the matrix and the vector and then we use the new operation $\oplus$ on the new vector to get a final impact score which can characterize the user's preference on the candidate item.

### 3.4. Model fitting

Our paper is centered on the rating prediction task and we train our model by fitting the observed ratings. Given a batch of training examples $\mathcal{R}$, in order to estimate the parameters set $\theta$, one can solve the least squares problem with the hierarchical similarity measure, which we call it Hierarchical Loss:

$$\mathcal{L}_H = \min_\theta \sum_{(u,i)\in\mathcal{R}} (r_{ui} - \hat{r_{ui}})^2 = \min_\theta \sum_{(u,i)\in\mathcal{R}} (r_{ui} - scale \times \sigma(\mathbf{P}_{\boldsymbol{u}} \odot \mathbf{q}_{\mathbf{i}}))^2, \tag{3}$$

where *scale* represents the rating scale (usually is 5). Specifically, during the training process, the parameters in the Embedding Layer are jointly learned with other layers in the neural networks rather than using a pre-trained embedding. The objective function is minimized by the gradient decent approach and we employ Adam[16] optimization method to accelerate the training process.

## 4. Experiments

In this section, we conduct abundant experiments on five datasets to validate the efficiency of our proposed model. We firstly compare our proposed model DLFM-HSM with the state-of-the-art ones, then we investigate how does Hierarchical Similarity Measure work and further analyze the performance of the DLFM-HSM from different aspects.

**Table 2**
Characteristics of datasets.

| Dataset | #users | #items | #ratings | density |
|---|---|---|---|---|
| ML-1M | 6040 | 3544 | 993,482 | 4.641% |
| CDs & Vinly | 75,258 | 64,443 | 1,097,592 | 0.02% |
| Baby | 857,664 | 147,472 | 1,415,622 | 0.001% |
| Video games | 826,767 | 50,210 | 1,324,753 | 0.003% |
| Musical instruments | 339,231 | 83,046 | 500,176 | 0.001% |

### 4.1. Experimental setting

#### 4.1.1. Datasets

Because we need user-item rating records and items' descriptions as input to build our model, we choose five public datasets containing both information to validate the performance of our model. The statistics of the five datasets are listed in Table 2. The details of each dataset are as following:

**Movielens 1M(ML-1M)**, is a movie dataset that has been widely used in rating prediction task. It has 1 million ratings from 6000 users on 4000 movies. Due to the original dataset does not have any content information about movies, we adopt another edition[1] used in [14]. The authors in [14] collected these descriptions of movies from IMDB.[2]

**Amazon**, as far as we know, is the largest public recommendation datasets which include more than 143 million rating records with textual information.[3] It contains 21 categories which are independent of each other and each category can be treated as a separate dataset that is still large and sparse. Therefore, we choose four datasets: CDs & Vinly, Baby, Video Games and Musical Instruments, and the density of the sparsest dataset is only 0.001%.

#### 4.1.2. Evaluation metrics

To compare with baselines, we use mean average error (MAE) and mean square error (MSE) as evaluation metrics, which are frequently used in the rating prediction task. The definitions are as follows:

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|r_{ui} - \hat{r_{ui}}|, \tag{4}$$

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(r_{ui} - \hat{r_{ui}})^2, \tag{5}$$

where $r_{ui}$ is the observed rating score of user $u$ to item $i$ and $\hat{r_{ui}}$ is its corresponding predicted rating score.

Both of them can measure the distance between the real rating and the predicted rating and they are negatively-oriented scores, which means lower values are better. Since the errors are squared before they are averaged, the MSE gives a relatively high weight to large errors.

#### 4.1.3. Baselines

In order to evaluate our proposed model, we use the following methods as comparison baselines:

**SVD++**: **S**ingular **V**alue **D**ecomposition++ [18] is a classic LFM that maps both users and items into a joint latent factor space. It tries to mix the strengths of the latent factor model as well as the neighborhood model.

**PMF**: **P**robabilistic **M**atrix **F**actorization [25] is another common matrix factorization method which models each latent factor for users and items by Gaussion Distribution.

**ConvMF**: **C**onvolutional **M**atrix **F**actorization [14] is a method which integrates convolutional neural network into probabilistic matrix factorization.

**DeepCoNN**: **Deep Co**operative **N**eural **N**etwork [36] uses two parallel neural networks to jointly learn the latent factors for users and items and then connects the two parallel neural networks through a top shared layer.

**FM**: **F**actorization **M**achines [28] can model all interactions between variables using factorized parameters.

**DLFM**: **D**eep **L**atent **F**actor **M**odel is another version of our model which is simply based on the inner product.

#### 4.1.4. Experimental setups

We follow the convention in [11] to split each dataset into three sub-categories: a training set, a validation set, and a test set with the ratio: 80%, 10%, 10%. The training set is used to train a model, the validation set is used to tune the hyperparameters, and the test set is used to evaluate the performance of the model. And then we need to preprocess the textual data as follows: 1) setting the maximum length of the description $l$ of each item as 50; 2) removing stop words;

[1] http://dm.postech.ac.kr/~cartopy/ConvMF/data/movielens.tar.
[2] Plot summaries are available at http://www.imdb.com/.
[3] http://jmcauley.ucsd.edu/data/amazon/links.html.

**Table 3**
The MSE / MAE results on various datasets.

| Model / Dataset | ML-1M | Musical instruments | Video game | CDs & Vinly | Baby |
|---|---|---|---|---|---|
| SVD++[18] | 0.898/0.705 | 1.314 / 0.863 | 1.638 / 0.998 | 0.929 / 0.674 | 1.475 / 0.929 |
| PMF [25] | 0.897/0.702 | 1.302 / 0.864 | 1.575 / 0.983 | 0.941 / 0.692 | 1.440 / 0.925 |
| ConvMF [14] | 0.853/0.684 | 1.134 / 0.806 | 1.337 / 0.977 | 0.938 / 0.721 | 1.330 / 0.923 |
| DeepCoNN [36] | 0.851/0.673 | 1.100 / 0.790 | 1.321 / 0.979 | 0.940 / 0.709 | 1.329 / 0.914 |
| FM [28] | **0.795**/0.686 | 1.481 / 0.911 | 1.865 / 1.079 | 1.366 / 0.808 | 1.935 / 1.127 |
| DLFM | 0.842/0.677 | 1.010 / 0.843 | 1.323 / 0.976 | 1.019 / 0.744 | 1.304 / 0.935 |
| **DLFM-HSM** | 0.824/**0.668** | **0.954 / 0.741** | **1.294 / 0.973** | **0.925 / 0.671** | **1.292 / 0.900** |
| **Improvement over best baseline** | **−3.51%/0.63%** | **13.27%/6.2%** | **2.04%/0.41%** | **0.43%/0.45%** | **2.78%/1.53%** |

3) constructing vocabulary by selecting the word whose frequency is larger than 5; 4) representing each description as a word-index vector.

There are some hyperparameters requiring tuning. We will introduce the validation process in the later part (see details in Section 4.3). We first give the hyperparameters setting as follows: Batch size is 256 and the maximum epoch is 3; the maximum length of description is 50; the dimension of word embedding is set to be 100 and the word embedding vectors will be trained through the optimization process; in the convolutional layer, we use three kinds of filters with different window size: 3, 4, 5 and each has 64 filters; the final dimension of the latent factor is 45. Finally, in order to keep the reliability of the model, we do the whole process five times and the average test errors are reported.

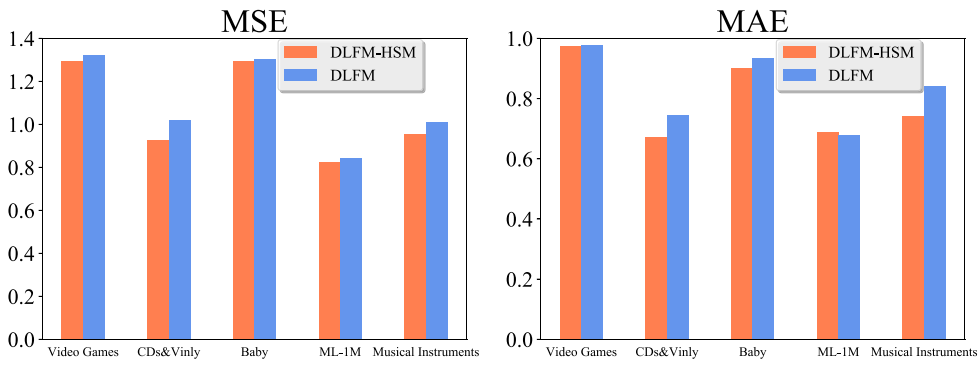
### *4.2. Performance comparison*

We first present the overall performance compared with all the baselines. Table 3 shows the MSE and the MAE results for all models on the five datasets and the best results are indicated in bold. We also show the improvements of our model compared with the best baseline on the different datasets. From the results we can observe that:

(1) Our proposed model achieves the best performance on all the datasets. Specifically, it achieves the biggest improvement on the Musical Instruments dataset which is the sparsest dataset (see Table 2). It indicates that our model can better alleviate the data sparsity problem. The reason is that our model learns the latent factors from item descriptions instead of from a rating matrix, which means that our model can learn more semantic factors. Therefore, when the rating matrix is sparse, our model is less affected.
(2) The baselines can be divided into two categories: with content information (ConvMF [14] and DeepCoNN [36]) and without content information (SVD++ [17], PMF [25]). By comparing the two kinds of methods, we find that utilizing content information can actually improve the performance of the model. Although PMF performs better than SVD++ on most datasets, neither of them achieves better performance compared to other baselines which utilize side content information. When the user-item rating matrix is sparse, the gap between them will be further widened.
(3) Compared with FM [28], we both try to model interactions between latent factors. From the results, we see that FM performs better than our model on the ML-1M dataset because ML-1M is far denser and has less number of users and items than the other four datasets. When the data is getting extremely sparser, the performance of FM is the worst one among all compared method. This can demonstrate that our model can better alleviate the data sparsity problem than the FM model.
(4) We find that ConvMF, DeepCoNN and our model utilize convolutional neural networks (CNN) to learn features for users and items. ConvMF uses CNN to guide the training process of PMF and DeepCoNN uses CNN to learn latent features for users and items respectively. Different from the models above, our model learns latent factors directly from item descriptions and also make users and items map into the same latent space which guarantees that users and items can be compared directly. And with hierarchical similarity measure, DLFM-HSM shows consistent improvements over ConvMF and DeepCoNN.

### *4.3. Model analysis*

In this section, we try to answer the following questions: (1) Does the Hierarchical Similarity Measure actually improve the performance of the model? (2) When the data gets sparser, how is the performance of the models? (3) Why we choose these hyperparameters? We conduct five sets of experiments to further analyze the performance of the DLFM-HSM. (1) Hierarchical Similarity Measure is the key to our model DLFM-HSM. To evaluate the effectiveness of Hierarchical Similarity Measure, we compare DLFM-HSM with DLFM which is a variant of our model and the results are in Fig. 5. DLFM just simply uses the traditional inner product instead of the hierarchical similarity measure to obtain the impact score. From the results, we see that DLFM-HSM has always outperformed DLFM, which verifies that HSM can help the model to explore deeper interactions between users and items than DLFM. Through further analyzing the results in Table 3, we see that the DLFM can achieve a better performance than other baselines on some datasets with abundant content information, which can indirectly verify our model's ability – extracting deep and effective factors.




(a) MSE results (b) MAE results

**Fig. 5.** Comparison between DLFM-HSM and DLFM.

**Table 4**
MSE results over different percentages of the Musical Instruments dataset.

| Model/Percentage | 20% | 30% | 40% | 50% | 60% | 70% | 80% |
|---|---|---|---|---|---|---|---|
| DLFM-HSM | **1.018** | **1.018** | **1.014** | **1.017** | **1.014** | **1.004** | **0.995** |
| FM | 1.519 | 1.503 | 1.480 | 1.491 | 1.452 | 1.470 | 1.481 |
| SVD++ | 1.381 | 1.392 | 1.405 | 1.387 | 1.377 | 1.362 | 1.354 |
| PMF | 1.366 | 1.376 | 1.387 | 1.373 | 1.374 | 1.352 | 1.352 |
| ConvMF | 1.311 | 1.308 | 1.325 | 1.261 | 1.154 | 1.141 | 1.139 |
| DeepCoNN | 1.118 | 1.118 | 1.116 | 1.114 | 1.110 | 1.107 | 1.103 |

**Table 5**
MAE results over different percentages of the Musical Instruments dataset.

| Model/Percentage | 20% | 30% | 40% | 50% | 60% | 70% | 80% |
|---|---|---|---|---|---|---|---|
| DLFM-HSM | **0.853** | **0.846** | **0.844** | **0.845** | **0.833** | **0.818** | **0.741** |
| FM | 0.918 | 0.918 | 0.914 | 0.917 | 0.917 | 0.909 | 0.912 |
| SVD++ | 0.894 | 0.901 | 0.907 | 0.899 | 0.899 | 0.888 | 0.884 |
| PMF | 0.890 | 0.897 | 0.899 | 0.895 | 0.895 | 0.887 | 0.879 |
| ConvMF | 0.862 | 0.860 | 0.853 | 0.836 | 0.817 | 0.811 | 0.806 |
| DeepCoNN | 0.871 | 0.851 | 0.842 | 0.858 | 0.820 | 0.804 | 0.790 |

(2) Furthermore, to demonstrate that DLFM-HSM has an ability to deal with the sparse problem, we apply the DLFM-HSM to different percentages of the training dataset. The results are reported in Tables 4 and 5. Compared with other baselines, DLFM-HSM can also achieve good performance when only using 20% of the training data, which indicates that our model can work well in the sparse situation. The reason is that though the dataset has few ratings, it still has some content information and our model can extract latent factors for users and items using a deep model from them and then alleviate the rating sparsity problem.

(3) In Fig. 6, we show the performance of DLFM-HSM on the validation sets of five different datasets with varying embedding dimensions from 25 to 200 and latent vector dimensions from 10 to 60 to investigate its sensitivity. As can be seen, it does not improve the performance much when the embedding dimension and latent vector dimension is greater than 100 and 45. Thus, we fix the embedding dimension as 100 and the latent vector dimension as 45.

(4) Lastly, we report the MSE of DLFM-HSM for each iteration on four datasets in Fig. 7. From the results, we can see DLFM-HSM reaches convergence quickly, which indicates that DLFM-HSM can improve performance without sacrificing efficiency.

## 5. Related work

In this section, we will give a brief review of the previous works which are closely related to LFM and distinguish our work from them.

Learning effective representations of users and items is the core of building recommender systems. A good feature representation can directly improve the quality of recommendations. LFM, as an effective way to learn representations, is the most widely used collaborative filtering approach, which exploits the hidden factors to represent users' interests and items' properties.

One of the most successful realizations of LFM is Matrix Factorization (MF) [18]. It is a classic model which many other models are based on. MF model maps users and items into a joint latent space and utilizes the inner product to capture

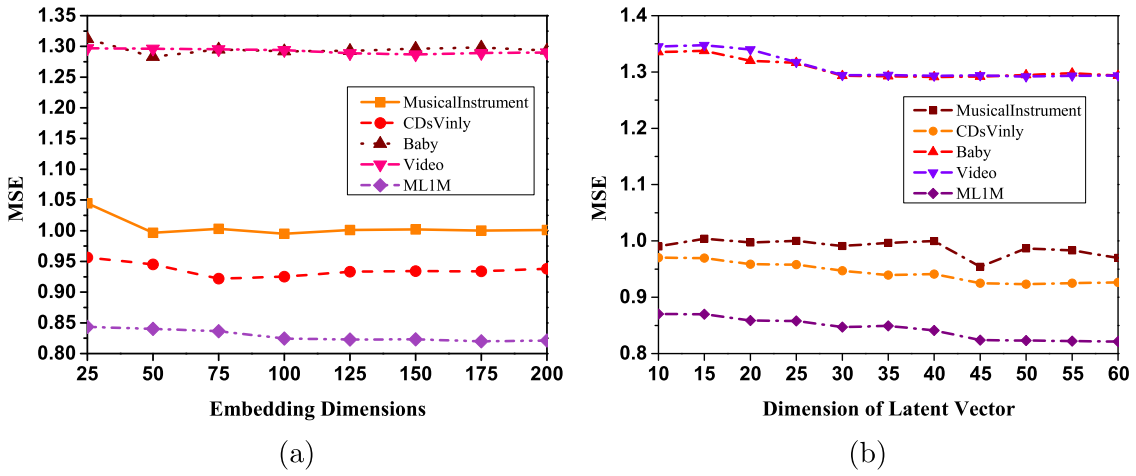



**Fig. 6.** The impact of the dimension of embedding vectors and latent vectors on the performance of DLFM-HSM in terms of MSE on different datasets.

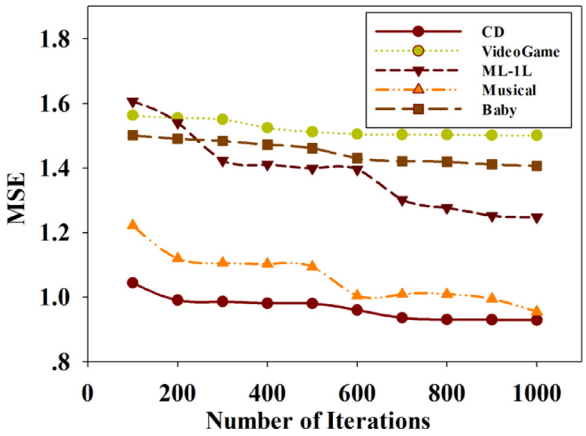



**Fig. 7.** Validation MSE of DLFM-HSM w.r.t the number of iterations on different datasets.

the interactions between users and items. The idea of MF is intuitive but effective for a recommendation. After this, many extensions of MF come out. Some people observe that there are some biases among ratings, therefore, they introduce the biases into the rating modeling process. SVD++ [17] is an example. It merges the factor and neighborhood models and exploits both explicit and implicit feedback by the users; Minh and Salakhutdinov [25] improve MF from another aspect. They proposed a model called Probability Matrix Factorization which models each latent factor by Gaussian Distribution to scale linearly with the number of observations. There is an obvious problem caused by "latent". Due to the "latent factor, these models cannot give specific explanations about why they recommend these items. Therefore, there are two kinds of improvements to LFM. One is focusing on how to improve the accuracy of recommendations, the other one is studying how to provide explanations for recommendations.

We first review the works aim to improve the accuracy of recommendations. Because of the huge users and items on the internet, the data sparsity problem always exists. To tackle this problem, some models [1,23] try to incorporate side information (such as reviews) and have shown promising performance in collaborative filtering. In recent years, deep learning, as an effective technique extract expressive features, has been widely applied in many research areas, such as computer vision [19], natural language processing [30], etc. Because of its strong ability in representation learning, many researchers introduce deep learning to recommender systems [20,22] which can also alleviate the data sparsity problem to some extent. Some methods [20,29,33] employ Auto-Encoder and Restricted Boltzmann Machine to accomplish collaborative filtering based on the user-item rating matrix and the authors of [27] design a deep convolutional neural network to learn the latent factors directly from music data to make music recommendations. These methods just try to combine deep learning method with recommender systems together, not further consider how to utilize deep learning technique in depth. With the rapid development of deep learning technique, more and more researchers put the emphasis on the improvement of algorithms of recommender systems using deep learning technique, not the simple combination between algorithms and deep learning. For example, ConvMF [14] seamlessly integrates CNN into PMF to capture contextual information which can improve the accuracy of rating prediction. Different from them, Zheng et al. [36] want to exploit abundant review data which contains

much available information can be used to make recommendations. In order to make the learned features beneficial to the rating prediction, they propose a joint deep learning model DeepCoNN which can jointly learn item properties and user behaviors from the review data. Catherine and Cohen [4] extend the DeepCoNN by introducing an additional latent layer representing the target user-target item pair even when the target user's review for the target item. He et al. [12] propose a model named Neural Collaborative Filtering (NCF). It leverages a multi-layer perceptron to capture the interactions between users and items instead of using the inner product. The motivation of NCF is similar to our paper, the difference between NCF with our model is reflected in two aspects: 1. The objective task is different, NCF is proposed for the item ranking task, while our model is proposed for the rating prediction task. 2. Compared with NCF, our model does not introduce many parameters (which need to train) into the training process.

Different from the above works, some researchers want to give an explicit explanation about the item they recommend. Therefore, they want to add some mechanism to make latent factors can be interpretative. To this end, some works [27,31] learn latent factors from textual information and incorporate topic LDA [2] into the training process. Authors in [8] propose a probabilistic model based on collaborative filtering and topic model to extract aspects and sentiments of users and items. Recently, Wang et al. [32] propose a novel solution named Tree-enhanced Embedding Method that combines the strengths of embedding based and tree-based models. They first employ a tree-based model to learn explicit decision rules (aka. cross features) from the rich side information, then design an embedding model that can incorporate explicit cross features and generalize to unseen cross features on user ID and item ID. At the core of their embedding method is an easy-to-interpret attention network, making the recommendation process fully transparent and explainable. Cheng et al. [6] apply a proposed aspect-aware topic model (ATM) on the review text to model user preferences and item features from different aspects, and estimate the aspect importance of a user towards an item. The aspect importance is then integrated into an aspect-aware latent factor model (ALFM), which learns user's and item's latent factors based on ratings. ALFM introduces a weighted matrix to associate those latent factors with the same set of aspects discovered by ATM, such that the latent factors could be used to estimate aspect ratings. There is a common problem among them. The process of latent factor learning and the user aspect learning are not jointly modeled, which may lose some interaction information between them.

## 6. Conclusion

In this work, we devise an effective model DLFM-HSM to solve the rating prediction problem. Specifically, in order to better measure the similarity between users and items, we design a Hierarchical Similarity Measure to replace the popular inner product. Furthermore, in order to make the model less susceptible to the data sparsity problem, we employ deep neural networks to learn users' preferences and items' profiles from items' descriptions. Extensive experiments have been conducted on five real-world datasets, and our results demonstrate the improvement of DLFM-HSM over state-of-the-art baselines.

From the experiment section, we can see that DLFM-HSM has a good performance when content information is abundant. But the performance of DLFM-HSM will be affected by lacking content information. Besides items' descriptions, there are many kinds of other content information, like users' comments on items. In the future, we will further explore how to merge these different kinds of content information together to better represent users and items.

## Conflict of interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

## Intellectual property

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

## Research ethics

We further confirm that any aspect of the work covered in this manuscript that has involved human patients has been conducted with the ethical approval of all relevant bodies and that such approvals are acknowledged within the manuscript.

IRB approval was obtained (required for studies and series of 3 or more cases).

Written consent to publish potentially identifying information, such as details or the case and photographs, was obtained from the patient(s) or their legal guardian(s).

## CRediT authorship contribution statement

**Jiayu Han:** Writing - original draft. **Lei Zheng:** Methodology. **He Huang:** Writing - review & editing. **Yuanbo Xu:** Writing - review & editing. **Philip S. Yu:** Supervision. **Wanli Zuo:** Supervision.

## Acknowledgment

This work is sponsored by the Scientific and Technological Development Program of Jilin Province (20180101330JC, 20190302029GX), the National Natural Science Foundation of China (61602057).

## References

[1] Y. Bao, H. Fang, J. Zhang, Topicmf: simultaneously exploiting ratings and reviews for recommendation., in: AAAI, 14, 2014, pp. 2–8.
[2] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (Jan) (2003) 993–1022.
[3] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowl.-Based Syst. 46 (2013) 109–132.
[4] R. Catherine, W. Cohen, Transnets: learning to transform for recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, ACM, 2017, pp. 288–296.
[5] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, T.-S. Chua, Attentive collaborative filtering: multimedia recommendation with item-and component-level attention, in: Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval, ACM, 2017, pp. 335–344.
[6] Z. Cheng, Y. Ding, L. Zhu, M. Kankanhalli, Aspect-aware latent factor model: rating prediction with ratings and reviews, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2018, pp. 639–648.
[7] A.S. Das, M. Datar, A. Garg, S. Rajaram, Google news personalization: scalable online collaborative filtering, in: Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 271–280.
[8] Q. Diao, M. Qiu, C.-Y. Wu, A.J. Smola, J. Jiang, C. Wang, Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars), in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014, pp. 193–202.
[9] M.D. Ekstrand, J.T. Riedl, J.A. Konstan, et al., Collaborative filtering recommender systems, Found. Trends® Hum.–Comput. Interact. 4 (2) (2011) 81–173.
[10] Y. Gong, Q. Zhang, Hashtag recommendation using attention-based convolutional neural network., in: IJCAI, 2016, pp. 2782–2788.
[11] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommendation tasks, J. Mach. Learn. Res. 10 (Dec) (2009) 2935–2962.
[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
[13] T. Hofmann, Latent semantic models for collaborative filtering, ACM Trans. Inf. Syst. (TOIS) 22 (1) (2004) 89–115.
[14] D. Kim, C. Park, J. Oh, S. Lee, H. Yu, Convolutional matrix factorization for document context-aware recommendation, in: Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016, pp. 233–240.
[15] Y. Kim, Convolutional neural networks for sentence classification, arXiv:1408.5882 (2014).
[16] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980 (2014).
[17] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 426–434.
[18] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009).
[19] Y. LeCun, K. Kavukcuoglu, C. Farabet, Convolutional networks and applications in vision, in: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, IEEE, 2010, pp. 253–256.
[20] S. Li, J. Kawale, Y. Fu, Deep collaborative filtering via marginalized denoising auto-encoder, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM, 2015, pp. 811–820.
[21] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: a survey, Decis. Support Syst. 74 (2015) 12–32.
[22] J. Manotumruksa, C. Macdonald, I. Ounis, Matrix factorisation with word embeddings for rating prediction on location-based social networks, in: European Conference on Information Retrieval, Springer, 2017, pp. 647–654.
[23] J. McAuley, J. Leskovec, Hidden factors and hidden topics: understanding rating dimensions with review text, in: Proceedings of the 7th ACM conference on Recommender systems, ACM, 2013, pp. 165–172.
[24] E. Michlmayr, S. Cayzer, Learning user profiles from tagging data and leveraging them for personal (ized) information access(2007).
[25] A. Mnih, R.R. Salakhutdinov, Probabilistic matrix factorization, in: Advances in Neural Information Processing Systems, 2008, pp. 1257–1264.
[26] R.J. Mooney, L. Roy, Content-based book recommending using learning for text categorization, in: Proceedings of the Fifth ACM Conference on Digital Libraries, ACM, 2000, pp. 195–204.
[27] A. Van den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, in: Advances in Neural Information Processing Systems, 2013, pp. 2643–2651.
[28] S. Rendle, Factorization machines with libfm, ACM Trans. Intell. Syst.Technol. (TIST) 3 (3) (2012) 57.
[29] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted Boltzmann machines for collaborative filtering, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 791–798.
[30] R. Socher, Y. Bengio, C.D. Manning, Deep learning for nlp (without magic), Tutorial Abstracts of ACL 2012, Association for Computational Linguistics, 2012. 5–5
[31] C. Wang, D.M. Blei, Collaborative topic modeling for recommending scientific articles, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 448–456.
[32] X. Wang, X. He, F. Feng, L. Nie, T.-S. Chua, Tem: tree-enhanced embedding model for explainable recommendation, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2018, pp. 1543–1552.
[33] Y. Wu, C. DuBois, A.X. Zheng, M. Ester, Collaborative denoising auto-encoders for top-n recommender systems, in: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, ACM, 2016, pp. 153–162.
[34] R. Yera, L. Martinez, Fuzzy tools in recommender systems: a survey, Int. J. Comput. Intell.Systems 10 (1) (2017) 776–803.
[35] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: a survey and new perspectives, ACM Comput. Surv. (CSUR) 52 (1) (2019) 5.
[36] L. Zheng, V. Noroozi, P.S. Yu, Joint deep modeling of users and items using reviews for recommendation, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, ACM, 2017, pp. 425–434.