Adaptive Deep Modeling of Users and Items Using Side Information for Recommendation

Jiayu Han, Lei Zheng, Yuanbo Xu, Bangzuo Zhang, Fuzhen Zhuang, Philip S. Yu, *Fellow, IEEE*, and Wanli Zuo

***Abstract*— In the existing recommender systems, matrix factorization (MF) is widely applied to model user preferences and item features by mapping the user-item ratings into a low-dimension latent vector space. However, MF has ignored the individual diversity where the user's preference for different unrated items is usually different. A fixed representation of user preference factor extracted by MF cannot model the individual diversity well, which leads to a repeated and inaccurate recommendation. To this end, we propose a novel latent factor model called adaptive deep latent factor model (ADLFM), which learns the preference factor of users adaptively in accordance with the specific items under consideration. We propose a novel user representation method that is derived from their rated item descriptions instead of original user-item ratings. Based on this, we further propose a deep neural networks framework with an attention factor to learn the adaptive representations of users. Extensive experiments on Amazon data sets demonstrate that ADLFM outperforms the state-of-the-art baselines greatly. Also, further experiments show that the attention factor indeed makes a great contribution to our method.**

***Index Terms*— Adaptive user preference model, attention factor, convolutional neural network (CNN), recommendation system.**

I. Introduction

IT IS well known that people are facing an information overload problem due to the increasing amounts of data on the Internet.[1] Recommender systems, as an effective way to help people filter the information, have become more and more important and received a lot of attention not only in industry but also in academia [1]. Recommender system can be applied to many areas such as movie recommendation [2], news recommendation [3], service recommendation [4], etc., by providing people with some personalized suggestions or valuable advice based on their history of transactions.

There are two main research tasks in this field: rating prediction [5] and personalized ranking [6] problem. Rating prediction is the task of predicting a given user's ratings for an item based on the past ratings while personalized ranking is the task of inducing an ordering over an item instance space. This paper mainly addresses the rating prediction problem which is an important task in many real-world recommendation scenarios like movie recommendation and product recommendation. It can estimate the rating that a user may give to a specific item and then make a recommendation according to the estimated rating. There are three common methods to solve this problem: the collaborative filtering (CF) method, the content-based filtering method, and the hybrid method. The CF method is a frequently used method to build recommender systems, which makes the recommendations by analyzing the transactions between users and items. Content-based filtering methods make recommendations based on the textual information of items, which are effective at recommending some items that are similar to the ones that users have rated. Hybrid methods use some strategies to combine the above methods effectively which aim to leverage the merits of each method to make a recommendation.

There are two kinds of methods based on CF methods: memory-based methods and model-based methods. The idea of memory-based algorithms [5], [7] is intuitive since it is based on an assumption that users can always get recommendations from others who share similar tastes with them. When this kind of algorithms predicts the rating, they should compute the whole data set, which is much time-consuming when the data set is large. Different from memory-based algorithms, model-based algorithms aim to learn a model that can predict the rating from the data set. Latent factor model (LFM) [8] is a kind of model-based methods. It is based on an assumption that ratings are deeply influenced by some unknown latent factors. It gives recommendations through modeling users and items as the latent vectors and comparing the distance of these vectors. The existing LFMs are usually based on matrix factorization (MF) [9], [10] which decompose the user-item rating matrix into two lower dimension vectors. Although

Manuscript received May 6, 2018; revised December 16, 2018; accepted March 30, 2019. This work was supported in part by the Scientific and Technological Development Program of Jilin Province under Grant 20180101330JC and Grant 20190302029GX and in part by the National Natural Science Foundation of China under Grant 61602057 and Grant 61773361. *(Corresponding author: Wanli Zuo.)*

J. Han, Y. Xu, and W. Zuo are with the Department of Computer Science and Technology, Jilin University, Changchun 130012, China, and also with the Key Laboratory of Symbolic Computation and Knowledge Engineering for the Ministry of Education, Jilin University, Changchun 130012, China (e-mail: jyhan15@mails.jlu.edu.cn; yuanbox15@mails.jlu.edu.cn; wanli@jlu.edu.cn).

L. Zheng and P. S. Yu are with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60661 USA (e-mail: lzheng21@uic.edu; psyu@uic.edu).

B. Zhang is with the School of Information Science and Technology, Northeast Normal University, Changchun 130117, China (e-mail: zhangbz@nenu.edu.cn).

F. Zhuang is with the Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: zhuangfuzhen@ict.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2909432

[1] http://www.internetworldstats.com/emarketing.htm

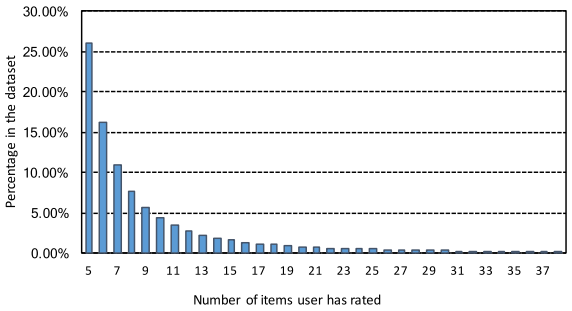


Fig. 1. Distribution of rated items per user has in the Movies data set of Amazon.

these methods have shown reasonably good performance in real-world data sets (e.g., Netflix,[2] Amazon,[3] etc.), relative to the millions of items that need to be recommended, the items rated by users only occupy a small portion.

To illustrate the above point, we count the number of ratings for each user in the Amazon data set, which is shown in Fig. 1. We can find that most of the users only have five ratings. Therefore, data sparsity has always been the main challenge to CF methods. For example, user A has bought three books: *A Brief History of Time*, *Deep Learning*, and *Pattern Recognition and Machine Learning*. However, there are millions of books on the Internet. If we represent the user as a multihot vector like neighborhood-based CF method, we will get an extremely sparse vector for user A. In this situation, LFM still cannot extract effective features from the sparse matrix. Therefore, the traditional CF method cannot get good performance when the data are really sparse. Furthermore, cold-start [11], [12] is an extreme situation of data sparsity which will lead to a poor experience to the new users. When new users or new items join into a recommender system, they do not have any reviews or ratings. In this situation, most of the traditional methods including LFM cannot solve this problem well either.

Dealing with the above problems, some researchers [13], [14] incorporated side information (like demographics of users [15], item reviews [16], and descriptions [17], etc.) into the learning process to alleviate the sparsity problem and achieve a noticeable improvement. However, there is still another drawback, *individual diversity* problem in the existing works, which means the lack of considering individual diversity of users when facing different items. The individual diversity means the preferences of a user for different target items should be different. In order to better clarify what individual diversity is, we will give an example which is shown in Fig. 2. In general, when predicting the rating for book *d* and book *e*, previous models first learn the representations of user and item and then calculate the similarity between the representations to give a prediction. However, previous models generally learn the fixed representations [$\mathbf{p_u}$ in Fig. 2(a)] for users to predict the rate for every item, which lacks flexibility and will lead to some boring and inaccurate recommendations. Therefore, it is very important to exploit the user's

[2]https://www.kaggle.com/laowingkin/netflix-movie-recommendation/data
[3]http://jmcauley.ucsd.edu/data/amazon/links.html

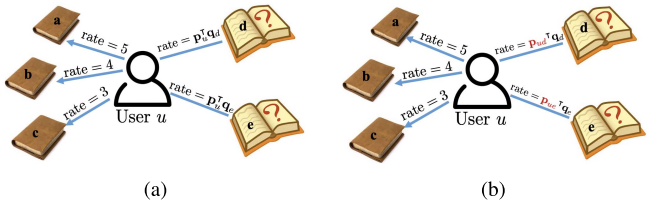


Fig. 2. Comparison between previous models and our model on how to construct users' preference representation. (a) Previous models: without individual diversity. (b) Our model: with individual diversity.

individual representation for each item, respectively [$\mathbf{p_{ud}}$, $\mathbf{p_{ue}}$ in Fig. 2(b)].

Therefore, the challenge of our task is how to construct a good preference representation from the sparse data for each user. We still use the example above. Intuitively, the preferences of users are usually hidden into the rated items. However, from the user's rating records, we can only know the top-level categories that user may be interested in and we cannot get more detailed information about the user's interests. Considering the item description that contains more abundant detailed information about the item. Take *A Brief History of Tim* as an example, the description of this book is that "A landmark volume in science writing by one of the great minds of our time, Stephen Hawking's book explores such profound questions as: How did the universe begin—and what made its start possible?" Therefore, intuitively, in order to get more detailed information about user's interests, we aim to extract the deep semantic information from the descriptions of items, and then we can use this information to construct the users' preference vector.

In recent years, deep learning has yielded immense success in computer vision, speech recognition, and natural language processing. It has attracted increasing attention from the researchers in recommender systems because of its power to learn effective feature representations by discovering the nonlinear relationships between users and items from the data. As such, in this paper, we propose a novel model called adaptive deep LFM (ADLFM) which combines convolutional neural networks (CNNs) with the LFM seamlessly and uses attention mechanism to solve the individual diversity problem. We propose ADLFM based on the heuristic information that different items in user history will play a different role in the process of prediction. Adaptive means that ADLFM can learn a different preference factor representation for a user when facing different items. It is realized by exploiting the deep relations between the users and items from the rating matrix and side information.

From the discussion above, we can see that users and items can share common representations intuitively, which is also the assumption of neighborhood-based CF method. In our proposed method ADLFM, we use descriptions of items a user has rated as the user's initial representation, which contains more semantic information. We use the word embedding technique to map the index of words of descriptions of each item into vectors of real numbers. Through this process, each word will be represented as a feature vector which contains complete information. Compared with "bag-of-words" techniques, it has

more capability to exploit the underlying relationships among words. Second, we use the CNNs [18] to extract local features around each word and then utilize a max operation to combine the local features into a fixed global vector. After learning the latent vector of each item, we construct a user latent vector by considering the user attention for the target item. The attention factor is used to measure the relevance between the user preference and the target item (an unrated item). Note that, in our model ADLFM, a user latent vector is not fixed. It means that when facing different items, the user latent vector will adaptively change.

The challenges we face in our work and the contributions we make in this paper are summarized as follows.

A. Challenge

1) Data Sparsity: Relative to the millions of items that need to be recommended, the items rated by users only occupy a small portion. How can we improve the performance of LFM when the model faces the data sparsity?

2) Individual Diversity: Previous models always adopt a fixed user latent vector to represent the preference of the user. How can we design a reasonable and adaptive user preference model?

B. Contributions

1) We propose a novel model called adaptive LFM that seamlessly combines deep learning technique with the LFM to tackle the sparsity problem.
2) We use item descriptions as side information which contains detail information about items. Also, an adaptive mechanism is proposed to better represent user preferences, which can construct user latent vector adaptively and accurately.
3) We conduct various experiments on Amazon data sets. The extensive results demonstrate that our proposed model outperforms the current state-of-the-art methods in rating prediction and can also mitigate the data sparsity problem.

The rest of this paper is organized as follows. We first review the related works in Section II. Then, we present the problem formulation and some preliminary model in Section III. In Section IV, we describe our proposed model ADLFM in detail. Then, extensive evaluations of our model are presented in Section V, which shows that our model outperforms other state-of-the-art techniques. Finally, we conclude this paper in Section VI.

II. Related Works

This paper is relevant to the following topics: How to model the users and items and how to apply deep learning to making recommendations. In this section, we will give a short review of the previous works that are closely related to our work and distinguish our work from them.

A. Latent Factor Model

Learning effective representations of users and items plays the most important role in recommender system since it can make predictions more accurate. An effective representation means that it can show the user's preferences or the item's properties. LFM is a common technique to solve this problem. Some of the most successful realizations of LFMs are based on MF [9], which tries to explain the ratings by characterizing both items and users on several factors inferred from the rating patterns. It first models users and items into a joint latent vector space and then exploits the relationships between users and items in this common space. There is another classic method called probability MF [19] which models each latent factor by the Gaussian distribution to scale linearly with the number of observations. They only utilize ratings or reviews, so these methods will suffer from sparsity problem which will deteriorate the models' performance. To alleviate this problem, many researchers attend to incorporate more information. Van den Oord *et al.* [20] put the topic model [21] into the learning process and learn the latent factors from textual information; McAuley and Leskovec [13] proposed a method using the topic model to understand the ratings with review texts and they, indeed, made an improvement compared with the above methods. A similar approach is followed in [22], which also tries to utilize both ratings and reviews to align the topic with the rating dimensions to improve prediction accuracy. Ren *et al.* [23] propose a latent variable model, which can generate explanations and predict item ratings based on user opinions and social relations. Recently, Jhamb and Fang [24] propose a dual-perspective LFM for group-aware event recommendation by using two kinds of latent factors to model the dual effect of groups, which can incorporate additional contextual information flexibly. Wei *et al.* [25] combine time-aware CF and deep learning methods to solve the cold start items problem.

Although these works have a good performance, the ability of representations is restricted by the similarity calculation method that uses lexical similarity to compute the textual similarity, which may ignore the deep relationships among words. Therefore, different from previous methods, our model aims to combine more deep semantic relationships into the LFM which can improve the performance while keeping the advantages of the LFM.

B. Recommendation Based on Deep Learning Method

Deep learning [26] has attracted the researcher's attention because of its powerful ability to learn the representative features. Recently, there are many recommender systems based on deep learning. Manotumruksa *et al.* [27] combine MF with word-embedding technique. Salakhutdinov *et al.* [28], Li *et al.* [29], and Wu *et al.* [30] utilize autoencoder and restricted Boltzmann machines model to accomplish CF only based on user-item matrix. He *et al.* [31] propose neural collaborative filtering method that leverages a multilayer perceptron to learn the user-item interaction function. Vartak *et al.* [32] presented a meta-learning strategy to address cold-start problem. To further utilize the auxiliary information (reviews, descriptions, etc.), many researchers study how to use deep learning models to explore the auxiliary information. ConvMF [17] seamlessly integrates CNN into the probabilistic matrix factorization (PMF) model, which can capture contextual

information for rating prediction. Zheng *et al.* [16] propose a joint deep learning model, deep cooperative neural networks (DeepCoNN), to jointly learn the item properties and user behaviors from the review text. Wang *et al.* [33] propose a deep knowledge-aware network (DKN) that incorporates knowledge graph representation into news recommendation.

Although these methods achieve significant improvement, all of them ignored the "individual diversity," which means that users' preference should vary for different items. Specifically, when predicting the rating, a user may rate a target item, the items (user has rated) that are relevant to the target item should provide more information to represent the user. Therefore, the major difference between the previous methods and ours is that we improve the performance of LFM with CNNs and attention mechanism. Under this model, we can construct an adaptive user preference representation that allows us to compare users and items more intuitively and directly.

III. Preliminary

In this section, we will present the basic problem in this paper and give a brief introduction to the model which is related to this problem.

A. Problem Formulation

We formulate the sparse recommendation problem in this paper as follows. Given a user $u \in \mathcal{U}$, we use $\mathcal{I}_u$ to represent the set of items that user u has rated and use $\mathcal{I}_u^-$ to represent the set of unrated items of user u. We aim to use the information from $\mathcal{I}_u$ to predict the rating that user u may give to an item in $\mathcal{I}_u^-$. The symbols we used to introduce our model are listed in Table I.

B. Latent Factor Model

LFM is a prevalent approach which can solve the recommendation problem by trying to explain the ratings by a set of latent factors. It is an effective way to predict ratings for unrated items. The factors can be inferred from transactions between users and items. MF is a common method to realize the LFM. In its basic form, both users and items are mapped into a joint latent vector space and it makes recommendations by measuring the correspondence between user preferences and item properties.

Suppose there are N users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ and aM items $\mathcal{I} = \{i_1, i_2, \ldots, i_M\}$. The user-item rating matrix is R. The size of R is $N \times M$. For each user $u \in \mathcal{U}$ and each item $i \in \mathcal{I}$, their corresponding latent factor vectors are $\mathbf{p_u} \in \mathbb{R}^k$ and $\mathbf{q_i} \in \mathbb{R}^k$, and the predicted rating to an unrated item can be modeled by inner production of vectors

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

From the above equation, we can see that how to estimate the latent vectors $\mathbf{p_u}$ and $\mathbf{q_i}$ is the key challenge in this problem. To learn the latent factor vectors, a basic objective function is employed. It is often defined as follows:

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

TABLE I
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
$\mathcal{I}_{ud}$	set of items which are used to construct user preference
V	Vocabulary of dataset
$\mathbf{W}$	weights of convolution layer
r $\mathcal{R}$	set of observed ratings
$\mathbf{p_u}$	latent vector of user u
$\mathbf{q_i}$	latent vector of item i
s	dimension of embedding vector
w	the window size of filter
k	max length of item's description
θ	parameters of deep neural networks
$\mathbb{R}$	set of real number
f_θ	function of neural layer
$\hat{r}_{ui}$	the predicted rating that user u for item i
r_{ui}	the actual rating that user u for item i
$\mathbf{d_i^e}$	the embedding of item i's description
$\mathbf{c_i^{max}}$	the intermediate vector
$\mathbf{b}$	bias for neural networks

where $\mathcal{R}$ represents the set of observed ratings and λ is used to control the strength of regularization in order to prevent overfitting problem. With $\mathbf{q}_*$ and $\mathbf{p}_*$, we can use inner product between the user and item latent vectors to predict the rating that user may give to the item.

IV. Adaptive Latent Factor Model Based on Neural Networks

The framework of ADLFM is illustrated in Fig. 3. As we see, it can be divided into two parts and they share the same feature extraction model. The lower part of the framework is the basic feature extraction model. It is a CNN model. We feed the item description into the model and it will generate a latent vector for an item. The details of this part will be introduced in Section IV-A. Then, we further learn the user-adaptive latent vector shown in the upper part of the framework which adopts the same model with the lower part and generates k intermediate latent features, respectively. To obtain the final adaptive user latent representation, we use an attention-based model (the details shown in Fig. 4) to integrate the k latent vector into a final user adaptive latent vector. We will introduce this part in Section IV-B.

A. Learning Item Latent Vector

The process of learning item latent vector is shown in Fig. 3 (bottom). We learn the item latent vector from item description using a deep neural network $f_\theta(\cdot)$ with parameter θ. It can be treated by a form of feed-forward neural network that can be denoted as a composition of function $f_\theta^l(\cdot)$ which corresponds with each layer l

$$f_\theta(\cdot) = f_\theta^L(f_\theta^{L-1}(\ldots f_\theta^1(\cdot)\ldots)).$$

In our model, we have four hidden layers. The first hidden layer is the embedding layer. It transfers the one hot encoded word input into a dense feature representation which is an

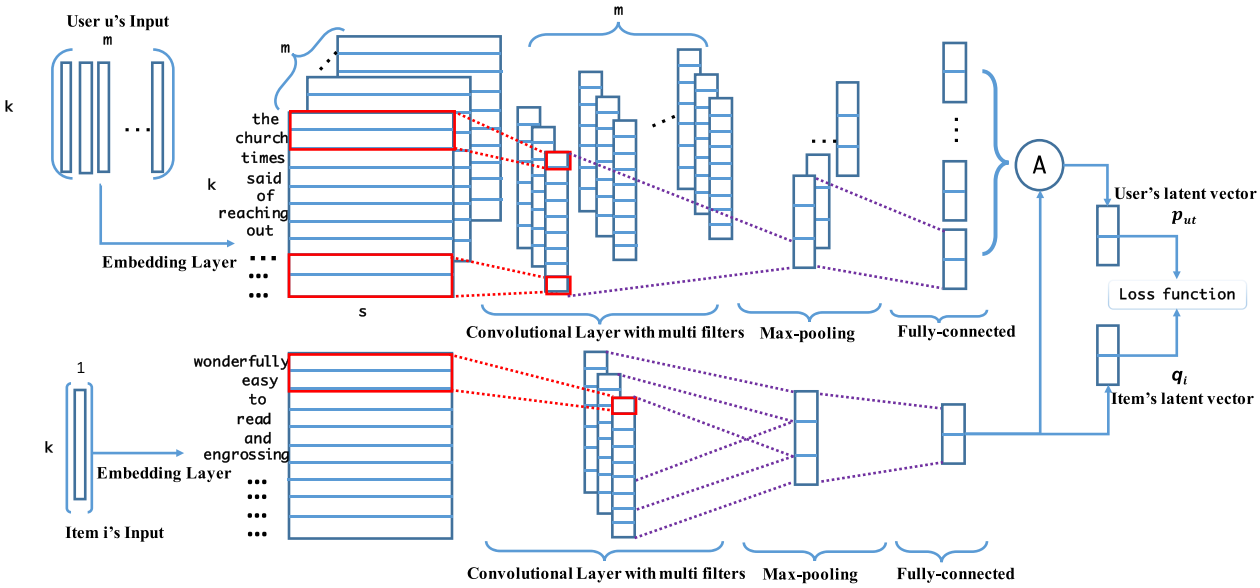


Fig. 3. Framework of ADLFM, where A is the attention module shown in Fig. 4.

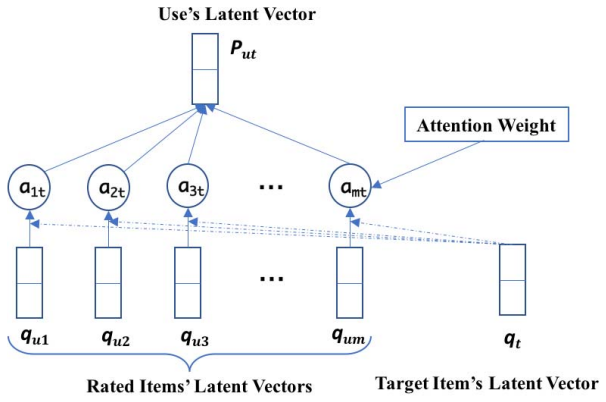


Fig. 4. Details of attention function A.

effective way to represent the semantic relations between words. It can be denoted as a function with parameters

$$f_\theta^1 : V \to \mathbb{R}^{n \times s}$$

where V represents the vocabulary obtained from the corpus, n is the size of V, and d is the dimension of embedding for each word.

Given a discrete set of item description inputs $D = \{d_1, d_2, \ldots, d_{|D|}\}$. Each $d_i \in |V|^k$ contains k words from the vocabulary V. If the length of description is greater than k, we will keep the first k words and discard the rest words. If the length is less than k, we will pad the zero factor for the rest. The raw input d_i is converted into a continuous vector by embedding operation. After embedding, each description is denoted as

$$\mathbf{d}_i^e = d_{w1} \oplus d_{w2} \oplus \ldots \oplus d_{wk}$$

where d_{wi} is the s-dimensional word vector corresponding to the ith word in the description d_i, and $\oplus$ is the concatenation operator which can turn k $1 \times s$ vectors to a $k \times s$ description matrix.

Next, we feed the output vector obtained above into the convolutional layer to extract. It can be formulated as

$$f_\theta^2 : c_i^j = f\left(\mathbf{W} * \mathbf{d}_{ij:j+s-1}^e + b\right)$$

where f is a nonlinear activation function (rectified linear unit is selected as activation function in this paper), $\mathbf{W} \in \mathbb{R}^{w \times d}$ represents a shared weight of convolutional filter, $*$ is a convolution operator, w is the window size of filter, and b is the bias. Filter can be used on each possible window and then we get a feature map

$$\mathbf{c_{i\cdot}} = \left[c_i^1, c_i^2, \cdots, c_i^{k-s+1}\right].$$

One filter function shares the same weights. In order to learn the general features, we use l different filters to extract the features in parallel.

The third hidden layer is the max-pooling layer. After the convolution layer, not all features can help in enhancing the performance. Therefore, we extract the most important contextual features by applying max-pooling layer for each feature map. The max-pooling function is defined as follows

$$f_\theta^3 : c_{i\cdot}^{max} = \max\{\mathbf{c_{i\cdot}}\}.$$

Because we have l different filters, after applying the max-pooling layer to each output of previous layer, we get the representation for each description

$$\mathbf{c_i^{max}} = \{c_{i1}^{max}, c_{i2}^{max}, \cdots, c_{il}^{max}\}.$$

The above vector will be parsed to the last layer—fully connected layer

$$f_\theta^4 : \mathbf{q_i} = \sigma\left(\mathbf{W}_f \mathbf{c_i^{max}} + \mathbf{b}\right)$$

where $\mathbf{W}_f$ is the weight matrix, $\mathbf{b}$ is the bias, σ is the activation function that can be linear or nonlinear. In this paper, we use sigmoid function as the activation function.

The above model can learn the combination of high-level features parsed from the previous layers and give a new representation of each description. Eventually, we obtain the latent vector $\mathbf{q_i}$ for each item i.

B. Learning User Adaptive Latent Factor

When making recommendations for a user, we often base on an assumption that the interests of the user are hidden in the previous rating history. As we observed, the item description often contains much basic descriptive information about an item. Intuitively, user preference is hidden in these descriptions of the items he has consumed or rated. Therefore, in our model, we aim to utilize the items in $\mathcal{I}_u$ to express the user preference of u. Different from the previous work [5], we aim to use the item latent vectors that are learned from item descriptions to represent each user. Therefore, we propose a new method to represent the user preference which can be represented by the item latent vectors

$$\mathcal{I}_{ud} = \{\mathbf{q_i} | i \in \mathcal{I}_u\}$$

where $\mathbf{q_i}$ is learned from the deep neural networks that are introduced in Section IV-A. Furthermore, when predicting the ratings of item a and b, respectively, we think that the user latent vectors can be different. It should adapt to different target item a and b. For example, when the user rates a comedic movie, the movies (he has rated) related to the comedic movie will play a more important role than others in prediction this time and this information should be reflected in the user preference representation. Based on this assumption, we use an attention-based method to learn an adaptive representation of each user.

Inspired by the classic work [34], we introduce an attention factor to implement the above idea (shown in Fig. 4). Assume an item $t \in \mathcal{I}_u^-$ is the target item which needs to measure. The attention factor is used to measure the related degree between the previously consumed items and target item. We compute the attention factor between $\mathcal{I}_{ud}$ and target item latent vector by applying the similarity function to each vector in $\mathcal{I}_{ud}$

$$a_{it} = \text{Similarity}(\mathbf{q_i}, \mathbf{q_t}).$$

Because we want to measure the distance between two latent vectors, in this paper, we use a simple but effective method cosine similarity to compute the attention factor a_i.

Attention mechanism was first proposed to solve computer vision problem [35]. The idea of attention mechanism is that a different part plays a different role in the deciding process and it can decide which part of a source should be paying attention to. In our work, we apply a simple attention mechanism that can take full account of the relationships between users and items to learn a user's profile adaptively.

Based on the attention mechanism, the user-adaptive latent factor can be represented as follows:

$$\mathbf{p_{ut}} = \sum_i a_{it} \mathbf{q_i}, \quad \mathbf{q_i} \in \mathcal{I}_{ud}.$$

In addition, we also propose a model named ADLFM with no attention (ADLFM-N) to be compared with ADLFM. ADLFM-N ignores the attention factor. It models the user preference as follows:

$$\mathbf{p_{ut}} = \frac{1}{|\mathcal{I}_{ud}|} \sum_i \mathbf{q_i}, \quad \mathbf{q_i} \in \mathcal{I}_{ud}.$$

After we obtain the latent vectors of user u and item t, we form the rating prediction as the inner product between the two latent vectors

$$\hat{r}_{ut} = \mathbf{q_t}^T \mathbf{p_{ut}}.$$

C. Model Fitting

All of the latent vectors are real number, so the most common used objective function is to minimize the mean square error (MSE)

$$\min_\theta \sum_{u \in \mathcal{U}, i \in \mathcal{I}_u} (r_{ui} - \mathbf{q_i}^T \mathbf{p_{ui}})^2$$

where θ refers to all the parameters that the model has. Specifically, during the training, the parameters in embedding layers are jointly learned with other layers in the neural networks rather than using a pretrained embedding. To solve the parameters, we employ Adam [36] optimization method to accelerate the training process. There are many tool kits for deep learning which can automatically implement the Adam algorithm, such as Theano[4] and Tensorflow.[5] In this paper, we base on the Tensorflow to use Adam, and we omit the details of solving process.

V. Experiments

A. Data Sets and Evaluation Metric

1) Data Sets: In order to validate the effectiveness of our ADLFM model, we conduct extensive experiments on the real-world Amazon.com data set.[6] As far as we know, Amazon is the largest public recommendation data sets with textual information. It contains a rating scale from 1 to 5 with much abundant textual information. Because our task needs item description, we select 15 data sets from Amazon to validate our model. The characteristics of our data sets are summarized in Table II. From Table II, we can see that these data sets are extremely sparse, so it can better validate the effectiveness of our method.

2) Evaluation Criterion: To compare with baselines, we use mean average error (MAE) MSE as evaluation metrics, which are frequently used in rating the prediction task. The definition is as follows:

$$MAE = \frac{1}{N} \sum_{n=1}^{N} |r_{ij} - \hat{r_{ij}}|$$

$$MSE = \frac{1}{N} \sum_{n=1}^{N} (r_{ij} - \hat{r_{ij}})^2$$

where r_{ij} is the observed rating score of user i to item j, and $\hat{r_{ij}}$ is its corresponding predicted rating score. Both of

[4] http://deeplearning.net/software/theano/

[5] https://www.tensorflow.org/

[6] http://jmcauley.ucsd.edu/data/amazon/links.html

TABLE II
CHARACTERISTICS OF AMAZON DATA SET

Dataset	#users	#items	#ratings	#ratings per user	density
Books	8,026,324	2,330,066	22,507,155	0.2903	0.0001%
Electronics	9,022,832	2,136,029	22,596,470	0.2367	0.0001%
Movies and TV	123,959	50,051	1,697,533	13.6943	0.0274%
Home and Kitchen	66,519	28,237	551,682	8.2936	0.0294%
Kindle Store	68,223	61,934	982,619	14.4030	0.0233%
Sports and Outdoors	35,598	18,357	296,337	8.3245	0.0453%
Cell Phones and Accessories	10,032,122	2,453,176	26,043,719	2.5960	0.0001%
Video Games	826,767	50,210	1,324,753	1.6023	0.0032%
Tools and Home Improvement	1,212,468	260,659	1,926,047	1.5885	0.0006%
Beauty	6,403,006	1,660,119	14,771,988	2.3070	0.0001%
Office Products	909,314	130,006	1,243,186	1.3671	0.0011%
Pet Supplies	10,532,966	2,822,861	28,115,041	2.6692	0.0001%
Automotive	4,250,467	1,183,567	8,813,589	2.0735	0.0002%
Grocery and Gourmet Food	5,752,043	1,410,889	12,748,918	2.2164	0.0002%
Musical Instruments	339,231	83,046	500,176	1.4742	0.0018%

them can measure the distance between the real rating and the predicted rating and they are negatively oriented scores, which means lower value is better. Compared with MAE, the MSE gives a relatively high weight to large errors.

3) Baselines: We choose five state-of-the-art methods that are extremely related to latent vector model to make a comparison with our model ADLFM. We choose MF [9] and PMF [19] to evaluate the effect of using side information. Hidden factors as topics (HFTs) [13] and ratings meet reviews (RMRs) [22] are the two methods that take the side information into the consideration. We choose these two methods to compare with ADLFM to show the effectiveness of applying CNNs. We also select a deep learning-based method DeepCoNN [16] to compare with our method. The last baseline is the variant of our model, ADLFM-N, which does not take the attention factor into account. We compare ADLFM with ADLFM-N to validate the effectiveness of applying the attention factor.

a) MF: **M**atrix **F**actorization [9] is a classic collaborate filtering method that maps both users and items into a joint latent factor space. It only utilizes the rating matrix to make rating prediction.

b) PMF: **P**robabilistic **M**atrix **F**actorization [19] is another common MF method which models each user's and item's latent factor by Gaussian distribution.

c) HFT: **H**idden **F**actors as **T**opics is proposed in [13]. It learns the latent factor from user's or item's reviews by employing topic distribution.

d) RMR: **R**ating **M**eet **R**eviews is introduced in [22]. It is a hybrid method which combines the content-based filtering and CF together.

e) DeepCoNN: **Deep Co**operative **N**eural **N**etworks [16] is a model that consists two parallel neural networks coupled in the last layers to learn item properties and user behaviors jointly.

B. Experimental Settings

Similar to the preprocess method in this paper [17], we preprocess the textual data as follows: 1) set the maximum length of item description k as 50; 2) remove stop words; 3) construct vocabulary by selecting the word whose frequency is larger than five; and 4) represent the item description as word index vector.

We implement ADLFM using Python and Keras library with NVIDIA GeForce Titan X GPU. To train the weights of deep neural networks, we use minibatch based on Adam. The minibatch size is 256. The maximum epoch is 100. The hyperparameters of deep neural networks are set as follows: 1) we set the maximum length of description is 50; 2) we fix the number of user descriptions as five since our method aims to solve the sparse task; 3) the dimension of the word embedding is set to be 50 and the word embedding will be trained through the optimization process; 4) in the convolutional layer, we use 64 filters and the window size of each filter is set to be 3, 4, and 5; and 5) we set the dimension of latent factor to be 15.

To evaluate the performance of each model in each data set, we split each data set into three subcategories: the training set, the testing set, and the validation set. Because we want to test the performance of the model when facing the sparsity situation, we only consider the case that each user only has five item descriptions in general. Therefore, the training set only contains five items for each user, the validation set has one item for each user to evaluate the performance of the model during the training process and the remaining items are all treated as testing items to test the performance of the model. In order to keep the reliability of the model, we repeat the whole process five times and the average test error is reported.

To implement the baselines, we set the parameters following the routine described in their papers [9], [13], [16], [19], [22]. For MF and PMF, we use grid search to find the best values for the number of latent factors from {25, 50, 100, 150, 200} and regularization parameter from {0.001, 0.01, 0.1, 1.0}. For HFT and RMR, the number of topics K we use is 5 and we set hyperparameters $\alpha = 0.1$, $\lambda_u = 0.02$ and $\lambda_v = 10$. For DeepCoNN, the model consists of two parallel neural networks and the two networks share the same framework but different parameter weights and we set the number of latent factors as 50 ($|\mathbf{x}_u| = |\mathbf{y}_i| = 50$) and the number of convolutional kernels as 100.

TABLE III
MSE/MAE RESULTS ON VARIOUS MODELS

Model	Dataset				
	Kindle Store	Office Products	Books	Video Games	Home and Kitchen
MF	1.553/0.971	1.814/1.004	1.107/0.819	1.61/0.989	1.628/0.988
PMF	1.561/0.973	1.796/1.018	1.109/0.802	1.608/0.972	1.610/0.989
HFT	1.437/0.901	1.669/1.021	1.138/0.804	1.528//0.973	1.531/0.987
RMR	1.412/0.905	1.638/0.989	1.113/0.791	1.510/0.967	1.501/0.942
DeepCoNN	1.238/0.816	1.198/0.893	1.025/0.802	1.501/0.952	1.342/0.871
ADLFM	**0.975/0.765**	**1.037/0.871**	**0.995//0.783**	**1.483/0.943**	**1.144/0.858**
Improvement(MSE)	**21.26-37.56%**	**3.57-36.33%**	**2.92-12.53%**	**1.20-7.88%**	**6.58-22.98%**
Improvement(MAE)	**6.25-21.37%**	**2.46-14.69%**	**1.01-4.40%**	**0.95-4.93%**	**1.49-13.2%**

Model	Dataset				
	Movies and TV	Musical Instruments	Cell Phones and Accessories	Gourmet Foods	Sports and Outdoors
MF	1.119/0.823	1.506/0.981	2.23/1.861	1.515/0.981	1.219/0.961
PMF	1.117/0.819	1.52/0.994	2.308/1.872	1.491/0.935	1.223/0.851
HFT	1.119 /0.816	1.395/0.838	2.315/1.897	1.457/0.936	1.138/0.815
RMR	1.12 /0.820	1.374/0.835	2.085/1.791	1.465/0.959	1.129/0.819
DeepCoNN	1.1124/0.812	1.233/0.866	1.341/0.973	1.391/0.889	0.9886/0.836
ADLFM	**1.108/0.809**	**1.025/0.798**	**1.221/0.958**	**1.372/0.842**	**0.915//0.712**
Improvement(MSE)	**0.44-1.116%**	**16.86-32.56%**	**8.94-47.26%**	**1.37-9.45%**	**4.70-22.96%**
Improvement(MAE)	**0.36-1.70%**	**4.43-19.72%**	**1.54-49.5%**	**5.28-14.2%**	**12.6-25.9%**

Model	Dataset				
	Automotive	Tools and Home Improvement	Pet Supplies	Electronics	Beauty
MF	1.57/0.974	1.60/0.991	1.70/1.002	1.828.1.015	1.399/1.042
PMF	1.585/0.981	1.61/0.993	1.70/1.011	1.823/1.053	1.414/0.917
HFT	1.492/0.923	1.51/0.961	1.583/0.978	1.722/1.021	1.358/0.912
RMR	1.432/0.915	1.491/0.927	1.562/0.963	1.722/1.018	1.344/0.923
DeepCoNN	1.335/0.924	1.450/0.957	1.328/0.981	1.506/1.109	1.309/0.866
ADLFM	**1.284/0.909**	**1.229/0.894**	**1.283/0.961**	**1.433/0.942**	**1.256/0.836**
Improvement(MSE)	**3.82-19.02%**	**15.5-23.65%**	**3.38-24.55%**	**4.82-21.58%**	**4.02-11.17%**
Improvement(MAE)	**0.65-7.34%**	**3.56-9.97%**	**0.41-4.95%**	**7.20-15.05%**	**3.46-19.77%**

C. Performance Comparison

We conduct extensive experiments to show that our method outperforms other baselines. In order to show the effectiveness of the adaptive user latent vector, we implement another version of the model: ADLFM-N. It is a variant of ADLFM model that ignores the attention factor when modeling the user latent factor. Table III shows the results of MSE and MAE for baselines and our model ADLFM in different data sets. We emphasize the best performance for each data set. We list the max and min improvement rates compared with baselines. From the results, we can see that ADLFM performs the best and makes a great improvement in the performance. We also list the executing time on each model on five typical data sets (in Table IV). Next, we will give a detailed analysis of the results.

1) Comparison of Different Methods: Table III shows the performance of MSE and MAE on baselines and our model. Compared with these baselines, ADLFM achieves the best performance on both metrics and the improvements are also listed in the table. MF and PMF are always two strong baselines in the rating prediction task. The significant gap between these two methods and ADLFM is due to the sparsity of Amazon data sets. From Table II, we find that Movies and TV data set is the densest data set and the ratings per user have is the largest. In this data set, all of the baselines achieve better performance than in other data sets. However, in Electronics data set, all the ones get the worst performance because of the lowest density of data set. In this sparse situation, MF and PMF cannot learn enough features from the user-item rating matrix to make an accurate rating prediction and underperform other methods which consider the side information. Therefore, it is proven that using some side information indeed relieves the problems caused by sparseness from the results.

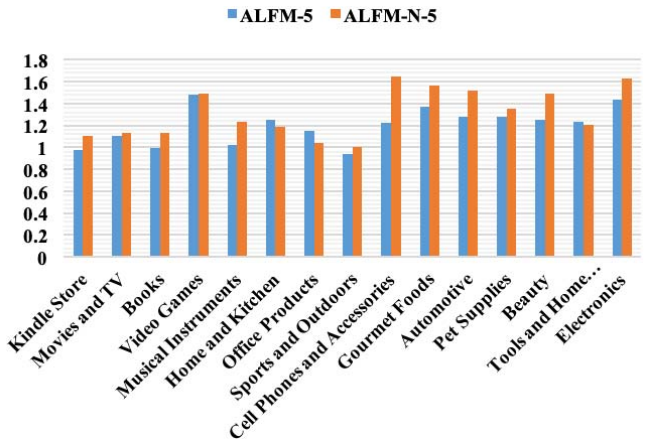


Fig. 5. Comparison between ADLFM and ADLFM-N.

HFT and RMR introduced in [13] and [22] are the models that both utilize textual content and ratings to model the user and item latent vector. The similarity between these two models with our model is that we all put the side information into the learning process. HFT and RMR model the content information using the topic distribution and model the ratings using LFM and then link them together. In addition, RMR model can explicitly learn the item features by

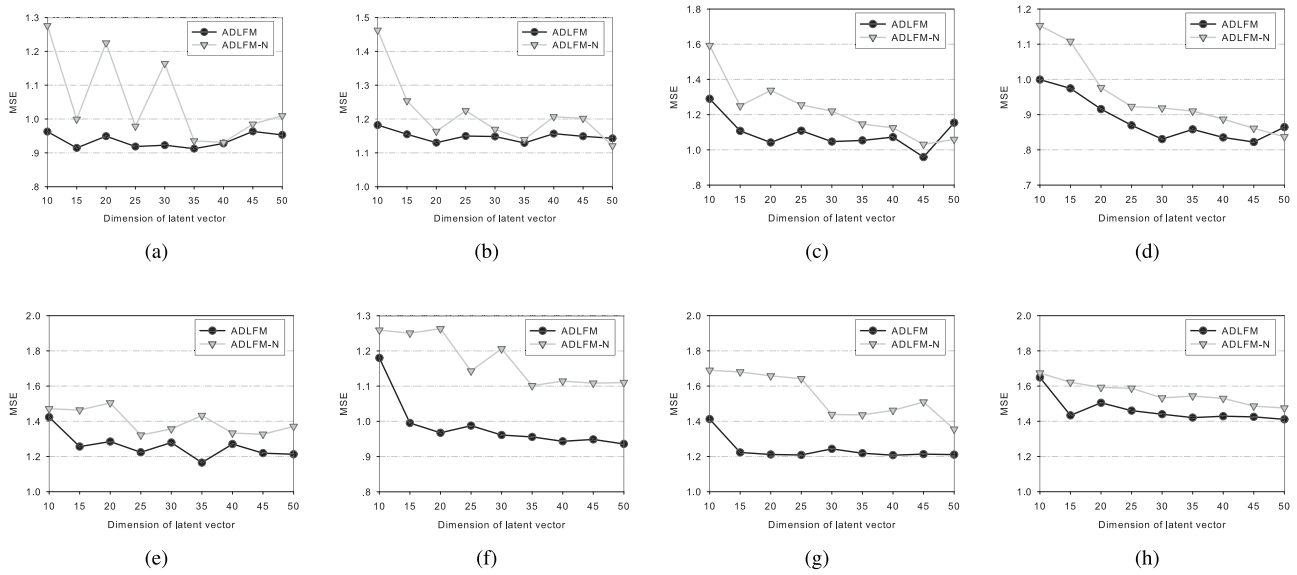


Fig. 6. MSE results over different dimension of latent vector. (a) Sports and outdoor. (b) Home and kitchen. (c) Movies and TV. (d) Kindle store. (e) Beauty. (f) Books. (g) Cell phones and accessories. (h) Electronics.

TABLE IV
COMPARISON OF EXECUTION TIME OF ALL MODELS ON FIVE TYPICAL DATA SETS

Model	MF	PMF	HFT	RMR	DeepCoNN	ADLFM
	time(s)					
Kindle	299	3157	2743	3568	113	141
Books	1946	7533	7367	7697	136	159
Sports and Outdoor	83	324	317	349	119	132
Musical Instuments	155	382	391	426	125	146
Video Games	419	1682	1476	1594	131	142

adopting a fixed one-to-one mapping between the item topic distribution and item feature vector. Different from the above, our model uses CNNs to learn the latent vector for each user and item. For the relative denser data sets, our model can still achieve a big improvement than HFT and RMR. It is because our model can extract more expressive and abstract feature factors from the content. DeepCoNN is another state-of-the-art method which is also based on CNNs. Compared with DeepCoNN, we can get a consistent improvement over it since our model considers individual diversity by constructing an adaptive user latent vector. In general, our model can make a significant improvement than baselines when the data set is extremely sparse and still achieves a little improvement when the data sets are relative dense (the density of the densest data is only 0.0453%).

In order to compare the efficiency of different models, we list the executing time of each model on five typical data sets (Table IV). These data sets have different scale and density, which can better illustrate the overall performance of each model. From the table, we find that MF, PMF, HFT, and RMR will be affected by the scale of data sets, when the data set is getting larger, the performance will be affected. Due to the GPU accelerating, our model and DeepCoNN are not affected.

2) Analysis of Attention Factor: The attention factor is the key to construct adaptive user representation and it can also reduce the space dimensions. We design two sets of experiments to analyze the effectiveness of the attention factor. First, we compare ADLFM with the variant of our model: ADLFM-N to show the necessity of the attention factor. Compared with ADLFM, the user latent vector learned from ADLFM-N is fixed all the time. As shown in Fig. 5, ADLFM works better than ADLFM-N in all data sets. It is because every time we construct a user latent vector, we consider the relationship between the target item latent features and the rated items' latent features in the user representation. The one that is more related to the target item will get a larger attention value than others and play a decisive role in constructing the final user latent vector and other items which are less related will play a complementary role in the constructing process. So in each time of prediction, it has always been those historical preferences that are most relevant to the target items play a decisive role, which can solve the individual diversity problem.

Second, in order to show that ADLFM can help reduce the size of latent space vector and its strong representation ability with attention factor, we select four sparse data sets and four dense data sets from 15 data sets to compare ADLFM and ADLFM-N over the different dimension of the latent vector. Fig. 6 shows the performance of ADLFM and ADLFM-N when the dimension of the latent vector ranges from 10 to 50. From the results, we find that ADLFM can get a better performance when the dimension of latent vectors is only about 15, which means that ADLFM has a strong expressive ability and can greatly reduce the vector space. Although there is a fluctuation of the MSE loss of ADLFM and ADLFM-N, compared with ADLFM-N, the margin fluctuation of loss of ADLFM is smaller. It means that our adaptive user preference

TABLE V
MSE RESULTS OF ADLFM OVER DIFFERENT NUMBER OF DESCRIPTIONS (HOME AND KITCHEN)

#descriptions	3	4	5	6	7	8	9	10	11	12
MSE	1.1831	1.1612	1.1444	1.1102	1.092	1.0825	1.0662	1.0568	1.0527	1.029
#descriptions	13	14	15	16	17	18	19	20	21	22
MSE	1.0312	1.0195	1.0271	1.0279	0.9863	1.0202	0.976	0.9623	0.9591	0.9411

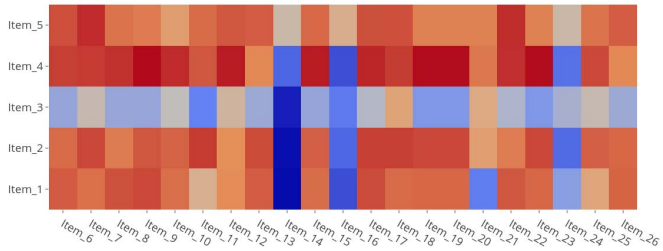


Fig. 7. Visualization of attention weights of ADLFM on an example user. y-axis: rated items. x-axis: candidate items.

model has better representation ability with attention factor and the attention factor can make ADLFM less influenced by the number of factors. From Fig. 6(b)–(d), we find that when the data set is relatively dense, if we extract enough factors, the ADLFM-N can also receive a good performance since CNNs can extract more expressive features from data. But in the sparse data sets, the advantage of our model is significant [see Fig. 6(e)–(h)]. The reason is that we fully take the relationships between users and items into consideration. Therefore, our adaptive user preference model has better representation ability with attention factor especially in the sparse situation and it is an effective way to represent the user preferences and solve the individual diversity problem. In order to make it more intuitive, we use a heat map to visualize the attention weights of our model on an example user from Musical Instruments data set (Fig. 7). The x-axis represents the candidate items of the user and the y-axis represents the rated items. A gradient from dark blue to dark red can represent the degree of relevance between items. The dark blue means the most irrelevant and the dark red means the most relevant. From Fig. 7, we see that our model can actually learn meaningful weights and with the same candidate item, different rated items have different attention weights. It can also validate that our model can make users preference to be adaptive to the candidate item.

3) Impact of the Number of Descriptions: Data sparsity has always been a prevalent problem in recommender systems. According to the statistics (see Fig. 1), we find that at least a quarter of the users only have five or fewer items (the data set only contain the users who rated five items at least). Therefore, we explore the performance of our model when facing fewer items (less than five) and furthermore, we explore the effect of the number of descriptions to ADLFM. Table V shows the performance of ADLFM when the numbers of rated items range from 3 to 22. We are surprised to find that our model can achieve much better than MF, PMF, HFT, and RMR when users only have three or four rating records. Note that our model can be further improved when rated items are abundant. The performance can get a 20.45% improvement compared to the case with fewer ratings.

VI. CONCLUSION

Many existing methods utilize MF to extract user latent representations from the user-item rating matrix. However, the fixed representations lose the ability to model the individual diversity of users and lead an inaccurate recommendation. In this paper, an effective model ADLFM is proposed to model adaptive representations of users from item descriptions and the user-item rating matrix. Moreover, an attentive factor is introduced to realize the adaptive mechanism in ADLFM. To the best of our knowledge, it is the first work which utilizes item descriptions to model user representations with an attention factor. And compelling results of experiments demonstrate the improvement of ADLFM over all the state-of-the-art baselines.

As shown in Table IV that if users have more rated items, our model can make a big improvement. In the future, we should deeply consider how to further improve our performance in the sparse situation by using some data set supplement techniques. Moreover, we intend to develop some complex mechanisms to learn the attention factor, so that we may extract the more accurate relationships between users and items.

ACKNOWLEDGMENT

The implementation of this work is available at GitHub.[7]

REFERENCES

[1] C. C. Aggarwal, *Recommender Systems*. Cham, Switzerland: Springer, 2016.
[2] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004.
[3] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber, "Offline and Online evaluation of news recommender systems at Swissinfo. Ch," in *Proc. 8th ACM Conf. Recommender Syst.*, Oct. 2014, pp. 169–176.
[4] M. B. Blake and M. F. Nowlan, "A Web service recommender system using enhanced syntactical matching," in *Proc. IEEE Int. Conf. Web Services*, Jul. 2007, pp. 575–582.
[5] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, May 2001, pp. 285–295.
[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, Jun. 2009, pp. 452–461.
[7] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 1999, pp. 230–237.

[7] https://gist.github.com/anonymous/00794d5b3d1d4bae9579c761af1ac09a

[8] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discovery Data*, vol. 4, no. 1, Jan. 2010, Art. no. 1.

[9] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[10] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2008, pp. 426–434.

[11] S.-T. Park and W. Chu, "Pairwise preference regression for cold-start recommendation," in *Proc. 3rd ACM Conf. Recommender Syst.*, Oct. 2009, pp. 21–28.

[12] K. Zhou, S.-H. Yang, and H. Zha, "Functional matrix factorizations for cold-start recommendation," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2011, pp. 315–324.

[13] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Proc. 7th ACM Conf. Recommender Syst.*, Oct. 2013, pp. 165–172.

[14] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 193–202.

[15] L. Safoury and A. Salah, "Exploiting user demographic attributes for solving cold-start problem in recommender system," *Lect. Notes Softw. Eng.*, vol. 1, no. 3, pp. 303–307, Aug. 2013.

[16] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 425–434.

[17] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 233–240.

[18] Y. Kim. (2014). "Convolutional neural networks for sentence classification." [Online]. Available: https://arxiv.org/abs/1408.5882

[19] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.

[20] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2643–2651.

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.

[22] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proc. 8th ACM Conf. Recommender Syst.*, Oct. 2014, pp. 105–112.

[23] Z. Ren, S. Liang, P. Li, S. Wang, and M. de Rijke, "Social collaborative viewpoint regression with explainable recommendations," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 485–494.

[24] Y. Jhamb and Y. Fang, "A dual-perspective latent factor model for group-aware social event recommendation," *Inf. Process. Manage.*, vol. 53, no. 3, pp. 559–576, May 2017.

[25] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Syst. Appl.*, vol. 69, pp. 29–39, Mar. 2017.

[26] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. Cambridge, U.K.: MIT Press, 2016.

[27] J. Manotumruksa, C. Macdonald, and I. Ounis, "Matrix factorisation with word embeddings for rating prediction on location-based social networks," in *Proc. Eur. Conf. Inf. Retr.* Cham, Switzerland: Springer, 2017, pp. 647–654.

[28] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn.*, Jun. 2007, pp. 791–798.

[29] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 811–820.

[30] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, Feb. 2016, pp. 153–162.

[31] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 173–182.

[32] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6907–6917.

[33] H. Wang, F. Zhang, X. Xie, and M. Guo. (2018). "DKN: Deep knowledge-aware network for news recommendation." [Online]. Available: https://arxiv.org/abs/1801.08284

[34] S. Sukhbaatar, J. Weston, and R. Fergus, "End-to-end memory networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2440–2448.

[35] D. Bahdanau, K. Cho, and Y. Bengio. (2014). "Neural machine translation by jointly learning to align and translate." [Online]. Available: https://arxiv.org/abs/1409.0473

[36] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.698

Jiayu Han received the B.Sc. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, in 2012 and 2015, respectively, where she is currently pursuing the Ph.D. degree with the Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education.

Her current research interests include data mining, data fusion, recommender system, and machine learning.

Lei Zheng received the bachelor's degree from Jilin University, Changchun, China, in 2010, and the master's degree from the Harbin Institute of Technology, Harbin, China, in 2012. He is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Illinois at Chicago (UIC), Chicago, IL, USA.

His current research interests include machine learning, data mining, and recommender systems.

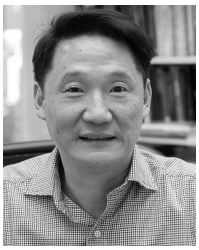
Yuanbo Xu received the B.Sc. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, where he is currently pursuing the Ph.D. degree with the Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education.

His current research interests include applications of data mining, recommender system, and mobile computing.

Bangzuo Zhang was born in 1971. He received the bachelor's degree in computer science from Northeast Normal University, Changchun, China, in 1995, and the master's and Ph.D. degrees in computer application technology from Jilin University, Changchun, in 2002 and 2009, respectively.

He is currently an Associate Professor with the School of Information Science and Technology, Northeast Normal University. His current research interests include information retrieval, recommender systems, machine learning, and data mining.

Fuzhen Zhuang is currently an Associate Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He has authored or coauthored more than 70 papers in some prestigious refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON CYBERNETICS, *ACM Transactions on Intelligent Systems and Technology, Information Sciences*, IJCAI, AAAI, WWW, ICDE, ACM CIKM, ACM WSDM, SIAM SDM, and IEEE ICDM. His current research interests include transfer learning, machine learning, data mining, multitask learning, and recommendation systems.



Philip S. Yu received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, and the M.B.A. degree from New York University, New York, NY, USA.

He is currently a Distinguished Professor of computer science with the University of Illinois at Chicago, Chicago, IL, USA, where he holds the Wexler Chair in information technology. His current research interests include big data, including data mining, data stream, database, and privacy.

Dr. Yu was a recipient of the ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion, and anonymization of big data, the IEEE Computer Society's 2013 Technical Achievement Award for pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining, and anonymization of big data and the Research Contributions Award from the IEEE International Conference on Data Mining (ICDM) in 2003 for his pioneering contributions to the field of data mining, the ICDM 10-year Highest-Impact Paper Award, and the EDBT Test of Time Award in 2014. He was also a recipient of several IBM honors including two IBM Outstanding Innovation Awards, an Outstanding Technical Achievement Award, two Research Division Awards, and the 94th Plateau of Invention Achievement Awards.

Wanli Zuo received the B.Sc., M.Eng., and Ph.D. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, in 1982, 1984, and 2005, respectively.

He is currently a Professor with Jilin University. He has authored or coauthored more than 160 journal papers and conference papers. His current research interests include machine learning, deep learning, information retrieval, natural language processing, and causality learning. He has accomplished three NSFC programs and five provincial/department programs and is currently undertaking a causality learning program sponsored by Jilin Provincial Key Research Funding.