Deep Exercise Recommendation Model

Tuanji Gong and Xuanxia Yao

***Abstract*—In online education scenario, recommending exercises for students is an attractive research topic. In this paper, we propose a new hybrid recommendation model that combines deep collaborative filtering (DeepCF) component with wide linear component. The former incorporates stacked denoising auto-encoder(SDAE) into matrix factorization and the latter is general linear component. In DeepCF component, we employ SDAE to learn low dimension latent feature of a student's feature and an item's feature and use matrix factorization method to predict the rating that a student rates an item. In wide linear model, we incorporate some meta properties of an item, such as difficulty, type and knowledge components(KCs). The two components are combined by linear approach. We use negative sampling method to generate the training dataset. An item is corrupted by Gaussian noise and is feed into the SDAE net ,which consists of encoder and decoder with multiple layers. We use tightly couple model to combine SDAE model and collaborative filter model. Experimental results show that the proposed model achieves a 10% relative improvement in AUC metric compared to the traditional collaborative filter method.**

***Index Terms*—Deep collaborative filtering, recommend system, stacked denoising autoencoder, exercise.**

I. Introduction

With the rapid development of online education, recommender systems play a key role in implementing personalized learning. In K12 online education scenario, there are hundreds of thousands of exercises. It is impossible and unnecessary for a student to do all exercises. The aim of exercise recommender system is to filter and pick up proper items for a student. It not only consolidates the KC that the student has just learned, but also can review learned KCs that she has high probability to forget.

The recommender system can improve the efficient and effort of doing exercises as it can know what she knows or what she does not know. It is also able to save the student's time to select proper exercises.

Recommender systems are usually classified into three categories: content based method, collaborative filter and hybrid method [1]. Collaborative filter based recommender systems have good performance and high accuracy, but suffer from cold start and sparse problem. Due to powerful computation and new algorithm, deep learning has achieved great success in many fields [2]. Inspired by the success of deep learning applied on speech recognition, image recognition, and machine translation, many researchers have employed deep learning on recommender systems [3]. Deep learning learns multiple levels of representations and abstractions from data, and can solve both supervised and unsupervised learning tasks [4]. Auto-encoder (AE) is an unsupervised deep learning model attempting to reconstruct its input data in the output layer. In general, the bottleneck layer (the middle-most layer) is used as a salient feature representation of the input data. There are many variants of auto-encoders such as denoising auto-encoder, marginalized denoising auto-encoder, sparse auto-encoder, contractive auto-encoder and variational auto-encoder (VAE).

In exercise recommender system, traditional collaborative filter models predict preference of a student based on matrix factor technology, but they do not utilize useful side information, such as content of items, abilities of KCs of a student. So the traditional collaborative filtering has poor performance and sparse issue. Based on deep model and wide mode, our method mitigates the problem and has a significant improvement in performance. Compared to traditional exercised system, our model has the following merits: 1) It is able to know what a student learned and recommend proper items to solid the knowledge component; 2) It is able to know how well a student grasped the KC and recommend proper difficult items ; 3) It can know what KC a student has a high probability to forget and suggest her to review the KC. The proposed model has been deployed on K12 online education platform and recommended thousands of items for students.

The contribution of this paper is two-folds: 1) The proposed model incorporates DeepCF model and wide linear model to recommend exercises for students; 2) We use negative sampling technology to generate training samples.

In the rest of the paper, we will discuss related work in Section II and present the model in Section III. The experiments and the corresponding results will be given in Section IV. Finally, we will conclude this paper and discuss the future work in Section V.

II. Related Work

Collaborative filtering(CF) base methods has extreme performance, winning the Netflix Prize competition [5]. CF makes use of usage or history data, such as user ratings on items, to recommend items that people with similar tastes and preferences have liked in the past. The two primary methods of collaborative filtering are the neighborhood method and latent factor model. Matrix factorization is one of the most successful implementation of the latent factor model. In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. However, CF suffers from the sparse problem. In order to alleviate the problem, Salakhutdinov *et al.* [6] proposed probabilistic matrix factorization(PMF)

Manuscript received July 23, 2018; revised December 19,2018.

The authors are with the School of Computer and Communication Engineering, University of Science and Technology Beijing 10083,China (e-mail: gongtuanji@foxmail.com,yaoxuanxia@163.com).

DOI: 10.7763/IJMO.2019.V9.677

model which scaled linearly with the number of observations and performed well on the large, sparse, and very imbalanced dataset. It modeled latent factors of users and items by Gaussian distributions. To improve the recommendation performance of PMF, Bayesian probabilistic matrix factorization method (BPMF)[7] considered a full Bayesian treatment of the parameter space instead of a point estimate used in PMF.

Learning effective features is critical in matrix factorization. Deep learning based methods have emerged as a powerful tool for learning representation. Denoising auto-encoders take a partially corrupted input whilst training to recover the original undistorted input. The goal of auto-encoder is to force the hidden layer to discover more robust features and to prevent it from simply learning the identity function. Vincent *et al.* [8] presented an stacked denoising auto-encoders with a local denoising criterion to learning useful representations. However, the learned latent factor may not be very effective due to the sparse nature of the ratings and the side information. To mitigate the problem, Li *et al* [9] proposed a model by combining probabilistic matrix factorization with marginalized denoising stacked auto-encoders. Because precious work could not learn a good representation from content for recommendation task or considered only text modality of the content in multimedia scenario, Li *et al* [10] proposed a Bayesian generative model called collaborative variant auto encoder (CVAE) that considered both rating and content for recommendation in multimedia scenario.

There are two approaches of integrating auto-encoder with traditional recommender system: tightly coupled model and loose coupled model [3]. Tightly coupled model learns the parameters of auto encoder component and recommender component simultaneously, which enables recommender model to provide guidance for auto encoder to learn more semantic features. It directly couples matrix factorization with deep learning models. Wang *et al.* [11] proposed a hierarchical Bayesian model which integrated stacked denoising autoencoder (SDAE) into probabilistic matrix factorization. Loosely coupled model is performed in two steps: learning salient feature representations via auto encoders, and then feeding these feature representations to recommender system. HRCD [12] is a hybrid collaborative model based on auto-encoder and timeSVD++ in loosely coupled approach. It is a time-aware model which uses SDAE to learn item representations from raw features and aims at solving the cold item problem.

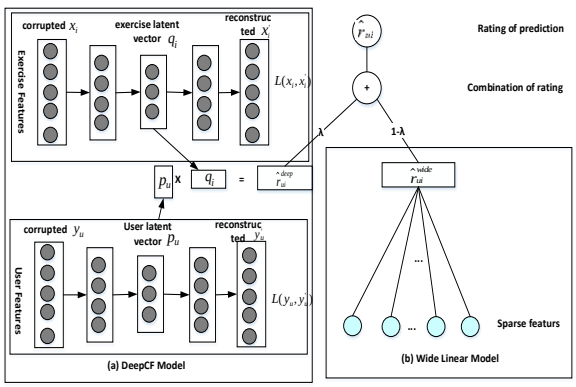

Fig. 1. The architecture of the proposed model. (a) DeepCF component,(b)wide linear component.

In order to use the advantage of wide linear model and deep learning model ,Cheng *et al.* [13] presented wide & deep learning--jointly trained wide linear models and deep neural networks--to combine the benefits of memorization and generalization for recommender systems. Guo *et al.* [14] proposed a combined model called DeepFM which combined the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture.

III. METHODOLOGY

A. Problem Definition

Given a set of users $U = \{u_i \mid i = 1, \ldots, N\}$, a set of items $I = \{i_j \mid j = 1, \ldots, M\}$, and an observed record of the users' past exercises of items is a 4-tuple, $O = (t, u, i, r_{ui})$, where r_{ui} is the preference that user u rates item i. It is binary values with 0 meaning that user u does not need to exercise item i and 1 indicating user u need to exercise. And t denotes the date that user u exercised item i. We use O^+ to denote the set of observed log, and O^- to denote the set of unobserved log. The goal of the recommender system is to pick for user u a set of items that the predicted values are most likely to be 1. In our case, $\hat{r}_{ui}$ is the predicted ratings in the range of [0, 1]. In the rest of the paper, we use u to index a user, and i and j to index items.

B. Model Architecture

Our model consists of two components: deep CF component and wide linear component. The architecture of the proposed model is illustrated in Fig.1. Fig.1(a) is the DeepCF component and it incorporates two SDAE nets into matrix factorization. Fig.1(b) is the wide linear component.

The two components are combined by linear approach with $\hat{r}_{ui}^{deep}$ denoting predicted rating by DeepCF component and $\hat{r}_{ui}^{wide}$ by wide component. The predicted rating of user u to item i is calculated as follows :

$$\hat{r}_{ui} = \sigma(\lambda \hat{r}_{ui}^{wide} + (1-\lambda)\hat{r}_{ui}^{deep}) \tag{1}$$

where $\hat{r}_{ui}$ is the predicted rating, $\hat{r}_{ui}^{wide}$ is the wide component of rating and $\hat{r}_{ui}^{deep}$ is the rating of deep CF component. $\sigma(.)$ is sigmoid activation function. λ is the control factor that balances the two components.

Our objective is to predict the rating that all user u on items in observed dataset. We use cross entropy loss as loss function as follows:

$$L = \prod_{(u,i) \in O \cup O^-} \hat{r}_{ui}^{\,r_{ui}} \cdot (1-\hat{r}_{ui})^{(1-r_{ui})} \tag{2}$$

C. Deep CF Component

We use two stacked denoising auto-encoder nets to learn low dimension representation of user and item respectively. The two SDAE nets have same architecture with multiple layers and the bottleneck layer of two nets has same size. Fig. 1 (a) illustrates the SDAE architecture. The first two layers encode the input to latent vector, the latter two layers decode the latent vector to reconstruct the input. Because using denoising technology can enhance the robust of feature, the input to SDAE net is corrupted with different type of corrupted method, such as Gaussian noise, mask noise, and salt-and-pepper noise. In low dimension latent space, we use matrix factorization method to predict the rating.

The SDAE model has N weight layers, first N/2 layers encode the input x to latent vector, last N/2 layers decode to reconstruct the input . The loss function of SDAE is the sum of squared error:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N} E_{p(x_i'|x_i)}\ell(x_i, x_i') \tag{3}$$

$$\begin{aligned}
&\ell(x,x') = (x-x')^2 + \beta R(\boldsymbol{\theta}) \\
&h_i = \sigma(W_i h_{i-1} + b_i), i = 1..N \\
&h_0 = x, h_N = x', h_{N/2} = z \\
&R(\boldsymbol{\theta}) = \sum_i \|W_i\|_2^2 + \|b_i\|_2^2
\end{aligned} \tag{4}$$

where h_i is ith hidden layer, W_i is the ith weight, b_i is ith bias, $\sigma(.)$ is sigmoid activation function, $R(\boldsymbol{\theta})$ is regular term with L2. β is smoothing factor.

The gradient of parameters are calculated by stochastic gradient decent and back propagation[15] .

Taking log for Eq.2 and adding L2 regularize term, we get loss objective function as follows:

$$\begin{aligned}
l = &-\sum_{(u,i)\in O\cup O^-} r_{ui}\log\hat{r}_{ui} + (1-r_{ui})\log(1-\hat{r}_{ui}) + \\
&\beta[\sum_j (\|W_j^u\|_2^2 + \|W_j^i\|_2^2 + \|b_j^u\|_2^2 + \|b_j^i\|_2^2) + \|b_u\|_2^2 + \|b_j\|_2^2 + \|W^{wide}\|_2^2 + \|b^{wide}\|_2^2]
\end{aligned} \tag{5}$$

$$\hat{r}_{ui}^{deep} = p_u \cdot q_i + b_u + b_i \tag{6}$$

$$p_u = f(W_2^u \cdot f(W_1^u u + b_1^u) + b_2^u) \tag{7}$$

$$p_i = f(W_2^i \cdot f(W_1^i \tilde{i} + b_1^i) + b_2^i) \tag{8}$$

where p_u, q_i is the latent vector of user u and item i respectively. u is corrupted input of user and $\tilde{i}$ is corrupted item input. $W_*^u, b_*^u, W_*^i, b_*^i$ stand for parameters of user and item SDAE net. The gradients of parameter of deep CF model are calculated as follows:

$$\begin{aligned}
\frac{\partial l}{\partial W_j^u} &= G\frac{\partial \hat{r}_{ui}}{\partial W_j^u} + 2\beta W_j^u \\
&= G\frac{\partial \hat{r}_{ui}}{\partial \hat{r}_{ui}^{deep}}\frac{\partial \hat{r}_{ui}^{deep}}{\partial p_u}\frac{\partial p_u}{\partial W_j^u} + 2\beta W_j^u \\
&= \lambda GHq_i\frac{\partial p_u}{\partial W_j^u} + 2\beta W_j^u
\end{aligned} \tag{9}$$

$$\begin{aligned}
\frac{\partial l}{\partial b_j^u} &= G\frac{\partial \hat{r}_{ui}}{\partial b_j^u} + 2\beta b_j^u \\
&= G\frac{\partial \hat{r}_{ui}}{\partial \hat{r}_{ui}^{deep}}\frac{\partial r_{ui}^{deep}}{\partial p_u}\frac{\partial p_u}{\partial b_j^u} + 2\beta b_j^u \\
&= \lambda GHq_i\frac{\partial p_u}{\partial b_j^u} + 2\beta b_j^u
\end{aligned} \tag{10}$$

$$\frac{\partial l}{\partial b_u} = G\frac{\partial \hat{r}_{ui}}{\partial b_u} = G\frac{\partial \hat{r}_{ui}}{\partial \hat{r}_{ui}^{deep}}\frac{\partial r_{ui}^{deep}}{\partial b_u} = \lambda GH \tag{11}$$

$$\frac{\partial l}{\partial W_j^i} = \lambda Gp_u\frac{\partial q_i}{\partial W_j^i} + 2\beta W_j^i \tag{12}$$

$$\frac{\partial l}{\partial b_j^i} = \lambda GHp_u\frac{\partial q_i}{\partial b_j^i} + 2\beta b_j^i \tag{13}$$

where

$$\begin{aligned}
&G = (r_{ui} - \hat{r}_{ui})/(\hat{r}_{ui}(1-\hat{r}_{ui})) \\
&H = \sigma(\lambda\hat{r}_{ui}^{wide} + (1-\lambda)\hat{r}_{ui}^{deep})(1-\sigma(\lambda\hat{r}_{ui}^{wide} + (1-\lambda)\hat{r}_{ui}^{deep})) \\
&\frac{\partial \hat{r}_{ui}}{\partial \hat{r}_{ui}^{deep}} = \lambda H
\end{aligned} \tag{14}$$

D. Input of SDAE net

The input of user SDAE is the abilities of KCs in a discipline that a student achieves and the ability value ranges from 0 to 1. The input size depends on the number of KCs in one discipline, for example, the number of KCs of math course in high school is 1786. The input of item SDAE consists of property features and content feature of an item. The content is embedded into low dimension feature and property features are transformed into real-valued type. The low-dimension content feature and transformed property features are concatenated as input of item SDAE.

E. Wide Linear Component

Similar to wide component in [13], our wide linear component is a generalized linear model of the form $y = W_w^T X + b_w$, as illustrated in Fig.1(b), y is the prediction of rating, $\mathbf{x} = [x_1, x_2, ..., x_d]$ is a vector of d features, $\mathbf{w}_{wide} = [w_1, w_2, ..., w_d]$ is weight parameters and b_{wide} is the bias.

The weight of parameters of wide component are calculated as follows:

$$\hat{r}_{ui}^{wide} = \sigma(W_{wide}^T X + b_{wide}) \quad (15)$$

$$\begin{aligned}\frac{\partial l}{\partial W_{wide}} &= G\frac{\partial \hat{r}_{ui}}{\partial W_{wide}} + 2\beta W_{wide} \\ &= G\frac{\partial \hat{r}_{ui}}{\partial \hat{r}_{ui}^{wide}}\frac{\partial \hat{r}_{ui}^{wide}}{\partial W_{wide}} + 2\beta W_{wide} \\ &= (1-\lambda)GH\sigma(W_{wide}^T X + b_{wide})(1-\sigma(W_{wide}^T X + b_{wide}))X + 2\beta W_{wide}\end{aligned} \quad (16)$$

$$\begin{aligned}\frac{\partial l}{\partial b_{wide}} &= G\frac{\partial \hat{r}_{ui}}{\partial W_{wide}} + 2\beta b_{wide} \\ &= G\frac{\partial \hat{r}_{ui}}{\partial \hat{r}_{ui}^{wide}}\frac{\partial \hat{r}_{ui}^{wide}}{\partial b_{wide}} + 2\beta b_{wide} \\ &= (1-\lambda)HG\sigma(W_{wide}^T X + b_{wide})(1-\sigma(W_{wide}^T X + b_{wide})) + 2\beta b_{wide}\end{aligned} \quad (17)$$

where $\sigma(.)$ is sigmoid activation function. W_{wide} is weight parameters and b_{wide} is bias,

$$\frac{\partial \hat{r}_{ui}}{\partial \hat{r}_{ui}^{wide}} = (1-\lambda)H$$

E. Feature Processing

There are three types of raw features in our model: numerical features, categorical feature and text feature.

(1).Numerical feature is normalized with 0 mean and standard deviation by normalized formula

$$x' = \frac{x-u}{\sigma}$$

(2).Categorical feature is coded by using one-hot coding .

(3).Text feature is embedded into low dimension representation. First, the content of item is segment into phrases by usying HanLP1 tool. Second, embedding vectors of phrases are get by looking up embedding dataset trained by [16]using Skip-gram[17] model. The embedding vectors of phrases are averaged as the feature vector of content.

F. Network Training

Given a set of training set T consisting of N tuples, we optimize the model through Adam[18] over shuffled mini-batches. Adam is an adaptive version of gradient descent which adaptively controls the step size with respect to the absolute value of the gradient and learning rate. The updating rules for parameter set θ of the networks are introduced in the next section.Weight matrix is initialized by using Glorot algorithm[19].

IV. EXPERIMENTS

A. Dataset Setup

To evaluate the performance of the model in real-world applications, we collected an exercised dataset at an online education company. The dataset is collected from exercise logs, which contains more than 50K user's exercise logs. For our purpose, we analyze the data from September 1, 2016 to Feb 1, 2018. We pick these students who are senior and have exercised on math courses. Our model can also be applied to other courses. For items ,the math course contains 1786 KCs and for every KC we select number of items ranging from 5 to 10 with difficulty between 0.2 to 0.9.The dataset amounts to 25860 items. We randomly select 8000 users who have exceeded 50 exercise times. We divide the dataset into three subsets of training set, validation set, and test set. We use 80% of the dataset as the training set, 10% as validation set to tune the hyper-parameters, and the rest is used as the test set.

B. Fill Missed Rating

Since a user can do only a small proportion of items, there are a lot of missed rating data. According to exercise logs, we use negative sampling method to generate rating matrix. The rating matrix has binary value with 1 indicating the user need to exercise and 0 indicating the user do not need to exercise the item. The sampling algorithm follows the below rules:

(1). Items will be rated as 0 if they have covering same knowledge component and less or equal difficult than the item that an user responded correctly.

(2). Items will be rated as 1 if

a) Difficulty of the item is higher than items that an user can respond correctly.

b) The item covers the KCs that an user do not exercise before and

c) The item has approximate difficult and same knowledge component with items that the user failed to answer correctly.

C. Feature Descriptions

User features consist of ability feature and property feature. The ability features include the ability of each KC in math course and course ability. Course ability of a student is weight average of abilities of all KCs in one course. The math course in high school has 1786 KCs. Property features of user feature is shown in Tab.1.

Item feature consists of property features and content feature. Content feature is the content of an item, which is text type of feature.

D. Evaluation Metrics

We use the receiver operating characteristic (ROC) curve and the area under ROC (AUC-ROC) as evaluation metrics. The true positive rate (TPR) and false positive rate (FPR) used for generating ROC curves are defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

where TP represents the true positives, FN represents the false negatives, TN represents the true negatives, and FP represents the false positives. We evaluate the performance of each compared method on the exercise dataset.

[1] https://github.com/hankcs/HanLP

TABLE I: PROPERTY OF AN ITEM AND PROPERTY OF AN USER

Item		User	
Feature	**Description**	**Feature**	**Description**
grade	It denotes the grade that the item is adapted to. One of twelve values ranging from 1 to 12.	grade	It denotes the grade of the student.
course	It denotes the course that the item belongs to. One of nine courses ranging from 1 to 9.	course	It denotes the course that the student exercises.
type	It denotes the type of an item. One of four types including selection, filling, computation, proven.	IQ	It denotes IQ of the student.
KCs	It denotes the KCs covered in an item	#KCs	It denotes number of KCs the student has exercised.
# KCs	It denotes the number of covered KCs in an item	Average difficulty	It denotes average difficulty of items that a student failed to answer in every KCs in last exercise.
difficulty	It denotes the difficulty of an item	-	-

E. Experimental Results

a) Baseline system

We use probabilistic matrix factorization[6] as the baseline system.

b) Number layers of SDAE

We conduct experiment with different layer and different size of latent vector. Result shows that the SDAE net with more layer has more performance and size of latent vector has a slight effort. In our experiment, we found that SDAE with 4 layer and size of latent vector with 300 attains highest performance with 0.837 AUC, shown in Tab.2. The reason behind is that deep layers have more powerful representation.

TABLE II: PERFORMANCE COMPARISON OF DIFFERENT LAYER AND SIZE OF LATENT VECTOR

#SDAE Layer	Size of Latent vector	AUC
2	200	0.819
2	300	0.821
4	200	0.836
4	300	**0.837**

F. Performances

In our dataset, we conduct an experiment to compare the performance of four models, as shown in Fig.2. The baseline model obtains 0.75 AUC. The wide model with 0.772 AUC has a slight performance improvement. Compared to baseline model, DeepCF model has a 4% relative increase. The hybrid model that combining DeepCF and wide model achieves the best performance with 0.825 AUC and 10% relative increase.

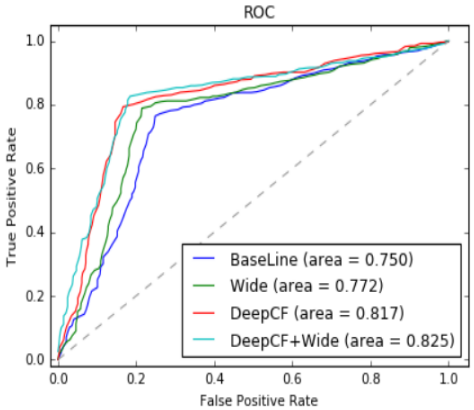

Fig. 2. Comparison of performance for four models.

V. CONCLUSION

In this paper, we propose a hybrid model combining DeepCF model and wide linear model. DeepCF model uses SDAE net integrating side information and property information of items to learn a robust low dimension representation of items that include side information. The hybrid model using advantages of deep model and wide model achieves the best performance with an 10% relative increase than baseline model.

For future work, we will use novel models, e.g. CNN or RNN, to improve performance of exercise recommendation.

REFERENCES

[1] F. O. Bobadilla, A. Hernando, and A.Gutierrez, "Recommender systems survey," *Knowledge-Based Sytems*, vol. 46, p. 33, 2013.

[2] Y. L. Cun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015.

[3] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," 2017.

[4] L. Deng and D. Yu, "Deep learning: Methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, pp. 197-387, 2014.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, pp. 30-37, 2009.

[6] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Porc. International Conference on Neural Information Processing Systems*, 2007, pp. 1257-1264.

[7] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Porc. International Conference on Machine Learning*, 2008, pp. 880-887.

[8] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371-3408, 2010.

[9] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Porc. ACM International on Conference on Information and Knowledge Management*, 2015, pp. 811-820.

[10] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *Porc. the ACM SIGKDD International Conference*, 2017, pp. 305-314.

[11] H. Wang, N. Wang, and D. Yeung, "Collaborative deep learning for recommender systems," in *Porc. Knowledge Discovery and Data Ining*, 2015, pp. 1235-1244.

[12] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Systems With Applications*, vol. 69, pp. 29-39, 2017.

[13] H. T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, *et al.*, "Wide and deep learning for recommender systems," pp. 7-10, 2016.

[14] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," pp. 1725-1731, 2017.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," vol. 323, pp. 399-421, 1986.
[16] S. Li, Z. Zhao, R. Hu, W. Li, T. Liu, and X. Du, "Analogical reasoning on chinese morphological and semantic relations," arXiv preprint arXiv:1805.06504, 2018.
[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, 2013.
[18] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Porc. the International Conference on Learning Representations*, 2015.
[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol. 9, pp. 249-256, 2010.

Tuanji Gong is a PhD student in computer science and technology at School of Computer and Communication Engineering, University of Science and Technology Beijing, China.

His research interests include machine learning, data mining, user modeling and recommender system.

He has 5 papers published in journals or international conferences.

Xuanxia Yao is an associate professor at School of Computer and Communication Engineering, University of Science and Technology Beijing, China.

Her research interests include network and information security, clouding computing, machine learning, machine vision and big data.

She has 33 papers published in journals or internal conferences, as well as one book.