Personalized and Explainable Employee Training Course Recommendations: A Bayesian Variational Approach

CHAO WANG, Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
HENGSHU ZHU and PENG WANG, Baidu Talent Intelligence Center, Baidu Inc., Beijing, China
CHEN ZHU, University of Science and Technology of China, Hefei, China
XI ZHANG, College of Management and Economics, Tianjin University, Tianjin, China
ENHONG CHEN, School of Computer Science, University of Science and Technology of China, Hefei, China
HUI XIONG, Artificial Intelligence Thrust, The Hong Kong University of Science and Technology, Guangzhou, China

As a major component of strategic talent management, learning and development (L&D) aims at improving the individual and organization performances through planning tailored training for employees to increase and improve their skills and knowledge. While many companies have developed the learning management systems (LMSs) for facilitating the online training of employees, a long-standing important issue is how to achieve personalized training recommendations with the consideration of their needs for future career development. To this end, in this article, we present a focused study on the explainable personalized online course recommender system for enhancing employee training and development. Specifically, we first propose a novel end-to-end hierarchical framework, namely Demand-aware Collaborative Bayesian Variational Network (DCBVN), to jointly model both the employees' current competencies and their career development preferences in an explainable way. In DCBVN, we first extract the latent interpretable representations of the employees' competencies from their skill profiles with autoencoding variational inference based topic modeling. Then, we develop an effective demand recognition mechanism for learning the personal demands of career development for employees. In particular, all the above processes are integrated into a unified Bayesian inference view for obtaining both accurate and explainable recommendations. Furthermore, for handling

This work was accomplished when Chao Wang was an intern at Talent Intelligent Center, Baidu Inc. supervised by the second and seventh authors.
This is a substantially extended and revised version of [58], which appears in the *Proceedings of the 29th Web Conference (WWW'20)*.
This work was supported by grants from the National Natural Science Foundation of China (No. 91746301, 61836013, U1605251, 71722005, 71571133), and Natural Science Foundation of Tianjin (No. 18JCJQJC45900).
Authors' addresses: C. Wang, Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China, 230026, China; email: wdyx2012@mail.ustc.edu.cn; H. Zhu (corresponding author) and P. Wang, Baidu Talent Intelligence Center, Baidu Inc., 100085, China; emails: {zhuhengshu, wangpeng40}@baidu.com; C. Zhu, University of Science and Technology of China, 230026, China; email: zc3930155@gmail.com; X. Zhang, College of Management and Economics, Tianjin University. 300072, China; email: jackyzhang@tju.edu.cn; E. Chen, School of Computer Science, University of Science and Technology of China, 230026, China; email: cheneh@ustc.edu.cn; H. Xiong (corresponding author), Artificial Intelligence Thrust, The Hong Kong University of Science and Technology, 511458, China; email: xionghui@ust.hk.

1046-8188/2021/12-ART70 $15.00
https://doi.org/10.1145/3490476

the employees with sparse or missing skill profiles, we develop an improved version of DCBVN, called the *Demand-aware Collaborative Competency Attentive Network (DCCAN) framework*, by considering the connectivity among employees. In DCCAN, we first build two employee competency graphs from learning and working aspects. Then, we design a graph-attentive network and a multi-head integration mechanism to infer one's competency information from her neighborhood employees. Finally, we can generate explainable recommendation results based on the competency representations. Extensive experimental results on real-world data clearly demonstrate the effectiveness and the interpretability of both of our frameworks, as well as their robustness on sparse and cold-start scenarios.

CCS Concepts: • **Information systems → Data mining;**

Additional Key Words and Phrases: Employee training course recommendation, recommender system, intelligent education

ACM Reference format:
Chao Wang, Hengshu Zhu, Peng Wang, Chen Zhu, Xi Zhang, Enhong Chen, and Hui Xiong. 2021. Personalized and Explainable Employee Training Course Recommendations: A Bayesian Variational Approach. *ACM Trans. Inf. Syst.* 40, 4, Article 70 (December 2021), 32 pages.
https://doi.org/10.1145/3490476

1 INTRODUCTION

In strategic talent management, learning and development (L&D) aims at improving the individual and organization performance through planning tailored training for employees to increase and hone their skills and knowledge, which is of great importance for companies to maintain their competitive edges in the fast-pace business environments [54]. According to the research reports from the *Association for Talent Development*,[1] U.S. organizations spent $1,296 per employee on L&D in 2018, with an average of 34.1 learning hours. Therefore, in recent years, more and more companies have built the learning management systems (LMSs) for facilitating the online training of employees, which can provide not only large cost savings, but also an effective way to deliver engaging development for talents due to the benefits of reach, scale, and timeliness [9]. However, along this line, a long-standing challenge is how to offer the talents with personalized training course recommendations.

Unlike traditional recommendation scenarios (e.g., movies or product recommendations), the learning motivation of employees heavily depends not only on their current competencies but also on their goals of future career development. In the literature, current course recommender systems mostly focus on the scenario of student education [2, 65]. Recently, some efforts were made to enhance the learning practices of individuals and organizations in talent management [47]. However, none of them has considered both their current competencies and career development demands. Figure 1 shows a motivating example of building a personalized employee training recommender system. Specifically, based on the skill profiles and historical course records, we can guess that Alice studied the course *Java Performance Optimization* due to the demand of honing existing skills, while Bob studied *From Hadoop to Spark* for increasing his current skill set. Meanwhile, although Cindy has a similar skill profile to Alice, she made a very different choice of training course, i.e., *Deep Learning in NLP*, which is far away from her existing competencies. This might be because she wanted to master skills in a new field for achieving her future career goal.

Therefore, it is critically important to simultaneously consider the current competencies and the career development demands of employees for the recommendation. Unfortunately, there are many technical and domain challenges along this line. First, the employees' competencies and development demands are closely connected, which requires a unified way to automatically

[1] https://www.td.org/.

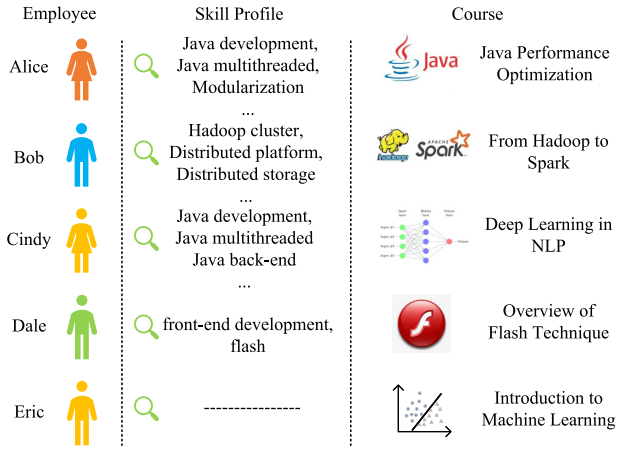


Fig. 1. A motivating example of personalized training recommendations for employees.

understand and represent the characteristics of employees. Also, in the educational domain, interpretable models play an important role in learning and progress, designing of better instruction, and possibly intervention to address individual and group needs [10, 19, 20, 30, 31]. Providing explainable analysis on the employees' competencies and recommendation results can stimulate their learning interests and help them better understand their competency status. Consequently, it is vital and challenging to make the latent representation in our frameworks have clear physical meaning for both employees and courses. Moreover, we usually have to face extreme data noise and sparseness problems on the LMS. On the one hand, the skill profiles may not completely reflect employees' skills and contain both fine-grained and coarse-level skill labels. On the other hand, employees' skill profiles may be quite sparse or even missing so we can have only a little information about them. It is a great challenge to learn the true competencies and demands of employees from such noisy and sparse data.

To directly achieve the primary goal of jointly modeling both the current competencies and the career development demands of employees, in our preliminary work [58], we propose a personalized online course recommender system for enhancing employee training. Specifically, the recommender system is based on a novelly designed end-to-end hierarchical framework, namely ***D*emand-aware *C*ollaborative *B*ayesian *V*ariational *N*etwork (DCBVN)**. Considering that the original skill profiles may be sparse and ambiguous, they could not be readily utilized. Thus, in DCBVN, we first extract the latent interpretable representations of employees' competencies from their skill profiles with autoencoding variational-inference-based topic modeling [46]. In this way, the auxiliary skill information helps us deepen the understanding of employees and hence, alleviates the sparsity and cold-start problem. Then, by exploiting the observed course records and collaborative learning behaviors, we develop an effective demand recognition mechanism for learning the personal demands of career development. Each dimension of both the latent competence and demand variable represents an interpretable skill topic in reality. Finally, the most appropriate training courses are recommended through an adapted collaborative filtering algorithm. All the above processes are integrated into a unified Bayesian collaborative filtering way to make sure both the recommendation accuracy and explainability at the same time.

During the modeling process of DCBVN, we have to first extract latent competency representations from the skill profiles to identify the current competency of each employee. Thus, it could not work well without the usage of skill information. However, in real data, the employees' skill profiles may be quite sparse or even missing. For example, as shown in Figure 1, Dale's skill profile

only contains two skills, which results in huge challenges for modeling his competency. Moreover, the Eric's skill profile is missing; thus, we can know nothing about his competency information. To tackle this problem, in this article, we further present the improved *Demand-aware Collaborative Competency Attentive Network (DCCAN)* framework for handling the absence of employees' skill profiles. Actually, DCBVN can be viewed as a part of DCCAN. DCCAN is capable of modeling the competency and demand representations for employees with sparse and even zero skill labels by utilizing the competency information of neighborhood employees. On the one hand, the employees who have learned the same courses may have similar skill profiles. On the other hand, the employees doing similar work may share similar competencies. According to these two observations, we first construct two undirected and unweighted graphs, based on the above two employee neighborhood definitions, respectively. Then, we design a graph-attentive network architecture to explore the potential competency of an employee from her neighbors. Furthermore, we propose a novel multi-head integration mechanism to combine the competency variables learned from two graphs to obtain the final supplementary competency variables. Similar to DCBVN, we can then transform the supplementary competency variables into demand variables by the demand recognition mechanism and further generate explainable recommendation results. Finally, we conduct extensive experimental results on a large-scale real-world dataset in both general and cold-start scenarios. The experimental results clearly demonstrate the effectiveness and the superior interpretation power of DCBVN and DCCAN frameworks.

Overview. The rest of this article is organized as follows: In Section 2, we introduce the real-world dataset used in this article. In Section 3, we introduce in detail how to generate explainable and accurate recommendations with our proposed DCBVN framework. Then, we extend DCBVN with DCCAN framework for handling the absence of employees' skill profiles in Section 4. The recommendation performance of our frameworks is evaluated in Section 5, with some further discussions on cold-start scenarios. Some case studies are also presented in Section 5 to show the interpretability of our frameworks. In Section 6, we briefly introduce some related work. Finally, we conclude this article in Section 7.

2 DATA DESCRIPTION

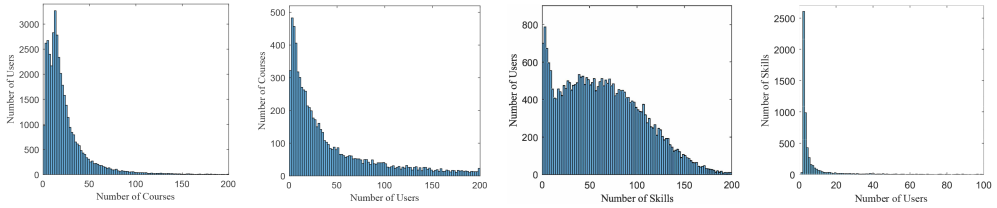
In this section, we introduce the real-world dataset shown in this article. Specifically, our dataset *DLearner* consists of two main components, namely learning records and skill profiles of employees, which are provided by a major high-tech company in China. Note that, all of the sensitive information in the dataset has been removed or made anonymous for privacy prevention purposes.

Learning Records. In the dataset, the learning records were collected from the LMS for employee training. Table 1 shows the statistical information of the dataset. Most courses are in the form of videos and slides. Different from student education, courses on the LMS usually focus on specific job skills. Thus, there is usually no clear sequential relationship (i.e., without course dependency) among the courses on the LMS. One can choose and study any course on the website of LMS without special restrictions.

Specifically, the learning records were collected from May 2016 to August 2019, containing 40,411 users and 8,693 courses. First, we need to exclude the noisy records that users only click on the online course but do not spend time learning. Consequently, only when a user had studied more than half of the video or slides, we considered it as a valid record. In this way, we can represent the entire learning records in the form of 0/1 rating matrix, where 1 means the valid record and 0 otherwise. There are totally 981,796 valid records in the *DLearner* dataset. Therefore, we can find that learning records are quite sparse as only 0.28% of the rating matrix entries contain valid records. Figures 2(a) and 2(b) show the distributions of the learning records from the user and course perspectives, respectively. It can be observed that most courses were studied by a

Table 1. The Statistical Information of the Dataset

The number of total learning records	981,796
The number of users	40,411
The number of users with skill profiles	30,662
The number of users without skill profiles	9,749
The number of courses	8,693
The number of skills	6,504



(a) User perspective (b) Course perspective (c) User perspective (d) Skill perspective

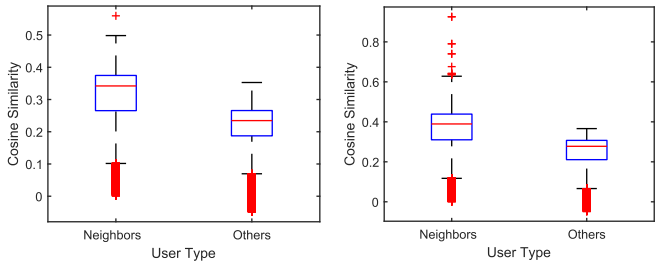
Fig. 2. The two figures (a) and (b) show the distributions of the number of learning records from user and skill perspectives, respectively. The two figures (c) and (d) show the distributions of the number of different job skills from user and skill perspective, respectively.

small number of users while the long tail effect is quite obvious in the distribution from the course perspective.

Skill Profiles. In the dataset, we also have detailed skill profiles for a large proportion of employees, which indicates the professional job skills the employee has already mastered before online training. Specifically, there are totally 6,504 unique job skills related to 30,662 users in the *DLearner* dataset, while the rest 9,749 users who have no skill profile. Besides, the employees are classified into different groups according to their departments and job positions in our dataset. There are totally 557 departments and 5 job positions (i.e., Technology, Product, User Interface, Manager, and Others). To show more statistical characteristics of the skill data, Figures 2(c) and 2(d) present the distributions of the number of the employees' skills from user and skill perspectives, respectively. We can see that the numbers of users with respect to the different numbers of skills show the double crest variation. The average number of skills per user mastered is 63.94. However, on the other hand, the data are quite sparse from the skill perspective. It is mainly because that the profile data contains both quite fine-grained and coarse-grained skill labels at the same time. As a result, some of the skill labels may have similar meanings but different names. Other skill labels may belong to contain relationships. Moreover, it is likely that the skills of each employee are not completely recorded in the profile data. Consequently, about 75% skills can only be found in less than 10 users' profiles in our dataset.

Based on the above, data sparsity and skill ambiguity raise great challenges to the design of the recommendation algorithm. Directly using the primary skill labels is improper. It is necessary to propose an appropriate approach to extract the effective competency representations from the high-dimensional and intensive noise data.

Besides, as shown in Table 1, we have collected 40,411 users in our dataset. Among them, 30,662 users have skill profiles. Hence, the ratio of missing skill profiles in the current dataset is about 1/4. However, this ratio in the practical industrial environment will be much larger, since we have filtered out the employees who rarely use the learning management system for better validating the recommendation methods. The users filtered out are mostly composed of outsourcing employees and interns, who usually do not have skill profiles. Consequently, the ratio of missing skill



(a) Learning perspective (b) Working perspective

Fig. 3. Figures (a) and (b) show the box plot of the average cosine similarity to other employees for each user from learning and working perspectives, respectively.

profiles will be much larger than 1/4 in practice. For those employees who have no skill profile, we are unable to learn their competency representations directly. To this end, in this article, we propose to infer the employees' competency variables from their "neighbors". Intuitively, there are two natural neighborhood relations for employees, i.e., learning and working neighborhoods. First, from the learning perspective, if two employees have learned the same course, they may have similar current competencies. Second, from the working perspective, if two employees are doing similar work, they may also share similar competencies. In the rest of this section, we aim to validate whether these two types of neighborhoods could help us complete the competencies. For each user i, we first divide the rest of the users into two groups according to whether or not they are neighbors of the user i. Then we use a 6,504-dim vector to denote the skill profile for every user. Each dimension of the vector represents a specific skill, where the value is set as 1 only if this skill is in the user's skill profile; otherwise, the value is 0. For a user i, we calculate the cosine similarity between each user and her neighbor's skill profile vectors and then averaged the similarities. We also randomly sample 100 users who are not in user i's neighborhood group and calculate the average cosine similarity. The results are shown in Figures 3(a) and 3(b). We can easily observe from Figures 3(a) and 3(b) that the similarities between one employee and her neighbors are actually much larger than the similarities between her and other employees from both learning and working perspectives. We also conducted the paired t-test to verify that the differences between the similarities to two user groups are statistically significant for $p < 0.005$, from both learning and working perspectives. By comparing the results in Figures 3(a) and 3(b), we can find that the similarities to neighbors in working perspective seems to be larger than those in the learning perspective, which implies that employees with similar works may have more similar competencies.

3 DCBVN FRAMEWORK

In this section, we will present the details of our proposed ***D*emand-aware *C*ollaborative *B*ayesian *V*ariational *N*etwork (DCBVN)** framework. First, we provide the probabilistic definitions of our DCBVN framework for Bayesian inference. Then, we introduce how to extract effective and interpretable latent competency representations from the skill profiles of employees. We further propose a demand recognition mechanism to transform the competency variables embeddings from latent skill space to latent demand space. Finally, collaborative filtering is performed for generating the recommendation results.

3.1 Probabilistic Modeling Paradigm

Following the famous **latent factor models (LFMs)** [37, 56], we suppose that the process of users selecting courses is influenced by two perspectives, i.e., user demand perspective and course

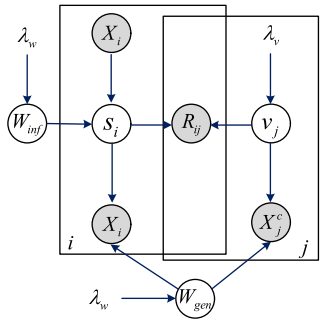


Fig. 4. The graphical model of DCBVN.

property perspective. Hence, we represent the user i by a latent variable $u_i \in \mathbb{R}^K$ and course j by a latent variable $v_j \in \mathbb{R}^K$ in a shared low-dimensional space with dimension K. Then, the rating r_{ij} of user i on course j is drawn from the Normal distribution centered at the inner product of the two latent variables:

$$r_{ij} \sim \mathcal{N}\left(u_i^T v_j, c_{ij}^{-1}\right), \tag{1}$$

where c_{ij} is the precision parameter. It is usually set higher when $r_{ij} = 1$ than when $r_{ij} = 0$ ($c_{ij} = a$ if $r_{ij} = 1, c_{ij} = c$ if $r_{ij} = 0, a >> c$) [56, 59], indicting we have more confidence on the rating when $r_{ij} = 1$, since $r_{ij} = 0$ means the user may either be uninterested or unaware of the course.

Then, we put the Normal distribution as the prior of each latent course variable v_j:

$$v_j \sim \mathcal{N}\left(0, \lambda_v^{-1} I\right). \tag{2}$$

In traditional LFMs, the user demand variable u_i is also drawn from the Normal prior like v_j. However, according to common sense, the user skill backgrounds have dominated influences in the course selection procedure. Comprehending users' competency is of great benefit to model their demands. Along this line, we utilize the latent competency variable $s_i \in \mathbb{R}^K$ to represent the competency of user i by building a probabilistic generative progress using VAE-LDA [46]. Thus, the skill profile x_i of user i is generated through a generative network parameterized by ψ:

$$x_i \sim p_\psi(x_i|s_i). \tag{3}$$

On the other hand, as discussed before, users who have analogous employee skills and hence have similar latent competency variables may differ greatly in personal learning preferences depending on their goals of career development. It is inappropriate to only consider the competency for training course recommending. We can draw support from historical records to recognize users' learning goals. Noticed that it is also improper to only consider the course records for recommendation, since the suitable courses are also different for people who have distinct competencies. Hence, in this article, we propose the demand recognition mechanism $G(\cdot)$ to transform the original competency variable s_i into the latent user demand variable u_i by considering both collaborative learning information and personalized employee competency information:

$$u_i = G(s_i). \tag{4}$$

In summary, Figure 4 illustrates the graphical model of DCBVN which comprehensively combines the conventional LFM-based collaborative filtering method with autoencoding variational inference for topic modeling.

3.2 Variational Autoencoder Network

In this subsection, we discuss how to extract semantic representations from the employee skills and build the collaborative network in Figure 5 for personalized training course recommendation. In industrial practice, we usually adopt the offline strategy by retraining the model periodically. For example, we can update the recommendation results by retraining the model every few weeks with the latest collected data to satisfy the practical needs. Therefore, in this work, we focus on modeling the personalized employees' training and development preferences through an explainable variational autoencoder network based on the input of current learning records and user profiles.

3.2.1 Topic Modeling of Skill Profiles. The physical meanings of user embeddings are quite important in the educational domain. Therefore, we have to carefully choose the appropriate embedding approach for achieving high recommendation performance while maintaining good interpretability at the same time. In the literature, the topic modeling algorithm is a quite popular probabilistic generative model for learning latent and interpretable representations [3, 45, 64]. However, traditional topic models may not perform well on the sparse data [67]. On the other hand, the **variational autoencoder (VAE)** has demonstrated its great inference abilities in many content embedding tasks, e.g., texts, labels, and images [24, 39]. Thus, we employ VAE-LDA in our framework for constructing a hierarchical and explainable course recommender system, which can combine the advantages of VAE and topic models.

Following the classic topic modeling algorithm [3], we assume every skill profile is generated from a mixture of K topics $\beta = (\beta_1, \ldots, \beta_K)$. Each topic $\beta_k \in \mathbb{R}^M$ represents a probability distribution over the entire M skills. For example, in the topic related to machine learning, skills like "SVM" and "random forest" would have high probabilities while the probabilities of "product design" and "market research" would be very small. The sum of the probabilities of all the skills should be equal to 1 for each topic.

Then the topic proportions $\theta_i \in \mathbb{R}^K$ of user i over the skills are supposed to obey Dirichlet distribution: $\theta_i \sim Dirichlet(\alpha)$, where each dimension represents the proportion of the corresponding topic. For example, $\theta_i = (0.1, 0.3, 0.6)$ means that the skill profile is related to the three topics with the proportions (10%, 30%, 60%), respectively.

Consequently, the skill profile $x_i = (x_{i1}, \ldots, x_{iN_i})$ with length N_i for user i can be drawn from the Multinomial distribution: $x_{in} \sim Multinomial(1, \beta\theta_i)$. Under this assumption, the marginal likelihood of the skill profile x_i can be given by:

$$p(x_i|\alpha, \beta) = \int_{\theta_i} \left(\prod_{n=1}^{N_i} p(x_{in}|\beta, \theta_i) \right) p(\theta_i|\alpha) d\theta_i. \tag{5}$$

The Dirichlet prior in LDA is significant for obtaining interpretable topics [55]. Nevertheless, it is problematic to implement the reparameterization trick for the Dirichlet distribution so that incapable to take gradients through the sampling process in VAE. To solve the problem, we utilize a Laplace approximation to the softmax basis of Dirichlet prior, which supports unconstrained optimization of the cost function [34].

Hence, we use s_i to denote the competency variable and we have $\theta_i = \sigma(s_i)$, where $\sigma(\cdot)$ is the softmax function. Each dimension of s_i is related to a specific topic in full accord with θ_i.

Then, following the Laplace approximation of the softmax basis in [15], the off-diagonal elements of the covariance matrix are suppressed with $O(1/K)$, leading to approximately diagonal covariance matrix for large K. Accordingly, the Laplace approximation $p(s_i)$ over the competency variable s_i can be given as a multivariate Normal with mean μ and covariance Σ where:

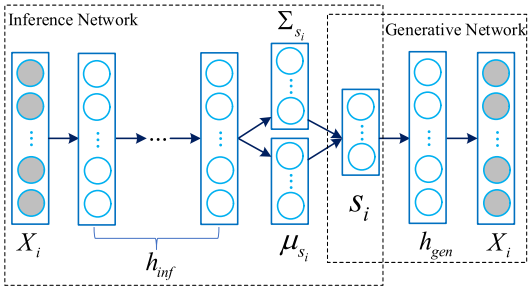


Fig. 5. The network architecture of autoencoding variational inference based topic modeling.

$$\begin{aligned} \mu_k &= \log \alpha_k - \frac{1}{K}\sum_{l=1}^{K} \log \alpha_l, \\ \Sigma_{kk} &= \frac{1}{\alpha_k}\left(1 - \frac{2}{K}\right) + \frac{1}{K^2}\sum_{l=1}^{K}\frac{1}{\alpha_l}. \end{aligned} \tag{6}$$

Recalling that competency variable s_i is the softmax basis of Dirichlet prior, we can thereupon approximate the simplex basis with the logistic normal distribution $p(\theta_i|\alpha) \approx \mathcal{LN}(\theta_i|\mu, \Sigma)$ [26].

3.2.2 Generative Process. Now we can present the generative process for user skill profiles:
(1) For the layer h_{gen} of the generative network:
(a) Draw the m_{th} column of weight matrix W_{gen}:

$$W_{gen,*m} \sim \mathcal{N}\left(0, \lambda_w^{-1} I_K\right);$$

(b) The topic matrix $\beta = \sigma(W_{gen})$;
(c) Obtain the ith row of hidden state h_{gen} by:

$$h_{gen,i*} = (\beta\theta_i)^T.$$

(2) For each user i, draw x_i from:

$$x_i \sim Multinomial(N_i, h_{gen,i*}).$$

Here, we also take advantage of the softmax basis W_{gen} of topic matrix β for the ease of unconstrained optimization. We use $\sigma(W_{gen})$ to denote the softmax transformation separately on every column of weight matrix W_{gen}.

It is noticed that there are two kinds of data collection modes for skill profile x_i in real scenes, depending on whether one skill label could appear once or multiple times in a list. For the situation that all the skills only hold at most once in the list per user, the skill generation follows a multivariate hypergeometric distribution instead of a multinomial distribution. Fortunately, the multivariate hypergeometric distribution converges to the multinomial distribution with large skill size [4]. Accordingly, we can still apply the above generative process for a large dataset in such a situation.

Owing to the above process, we can obtain the explainable competency variable s_i. However, the course property variable v_j still remains unexplainable. Due to the various sources of online courses, it may be quite manpower-consuming to collect the labels for each course. Moreover, it would be much more beneficial for helping users make decisions if the courses could be labeled with the same topics as user competency variables. Therefore, we can make use of the skill profiles of users who have learned the course to represent the property of this course.

Let x_j^c and s_j^c denote the skill profile and latent skill variable of course j, respectively. To construct x_j^c, we first count the numbers of occurrences of each skill among users who have learned course j. Then, we only keep the top e (e is a constant) frequent skills so that the popular courses would not have much noise skills. As for the latent skill variable s_j^c, we can transform the original unexplainable course property variable v_j into a suitable latent space to obtain the explainable vector. Here, we employ the 1-layer MLP for the transformation process:

$$s_j^c = \sigma\left(W^c v_j + b^c\right). \tag{7}$$

Then the generative process of course skill profile x_j^c with length N_j^c can be given by:

$$x_j^c \sim Multinomial\left(N_j^c, \beta s_j^c\right). \tag{8}$$

Here we share the same topic matrix β with the generative process of user skill profile to make sure that the topics in both user and course perspectives have exactly the same meanings.

3.2.3 Demand Recognition. The learned competency variables are very useful for understanding the employees' skill backgrounds. Intuitively, they are also highly relevant to the career development demands of employees. However, as discussed before, it is obviously inappropriate and limited to only utilize s_i for generating the final demand u_i. Supposing there are two technical employees with similar skill profiles using LMS for online learning. One wants to become a senior Java programmer, and the other one intends to transfer to an AI developer. Certainly, they should accept totally different recommendations to fit their actual demands.

Therefore, we propose a demand recognition mechanism as shown in Figure 6 to bridge the competency variable s_i and demand variable u_i by exploiting the historical course records and collaborative information. Since each dimension of s_i represents one of the K topics, we can model the pattern of demand transformation on each topic by combining transfer variable $d_t \in \mathbb{R}^K$ with s_i. Here, d_t reflects the changes in the proportions of topics. Specifically, each transfer pattern can be interpreted as one kind of learning trend. Then we have transformed matrix $Z_i = (z_{i1}, \ldots, z_{iT}) \in \mathbb{R}^{K\times T}$, where $z_{it} = s_i + d_t$, supposing there are T patterns.

In order to comprehensively analyze users' true demands, we need to measure the relations between each transfer pattern and users' real demand by calculating the importance score of every transfer pattern. Larger importance score ω_{it} would indicate more critical influence of pattern t on user i. In this way, we are able to capture the most relevant transfer patterns and pay more attention to them. In this article, we propose two ways to automatically compute the importance score $\omega_i = (\omega_{i1}, \ldots, \omega_{iT}) \in \mathbb{R}^T$, namely individual-based and group-based importance score.

For individual-based importance score ω_i, we only employ the historical course records of user i to scoop out his/her demands:

$$\omega_i = \sigma\left(Z_i^T V R_{i*}^T\right), \tag{9}$$

where R_{i*} is the ith row of the course record matrix R. Here, we take advantage of the softmax function to make sure that the sum of the T patterns' score is equal to 1.

However, the course records of a single user may be too few to find his/her real demand, since the data are quite sparse. Therefore, we further take the influence of user groups into consideration due to the common sense that employees doing similar work may share similar demands of career development. To this end, we first divide the users into different groups based on their departments and job positions. Let g_i denote the group that employee i belongs to. Then we compute the group-based record vector $\tilde{R}_{i*}$ as follows:

$$\tilde{R}_{i*} = \frac{\sum_f R_{f*} I(g_f = g_i)}{sum\left(\sum_f R_{f*} I(g_f = g_i)\right)}, \tag{10}$$

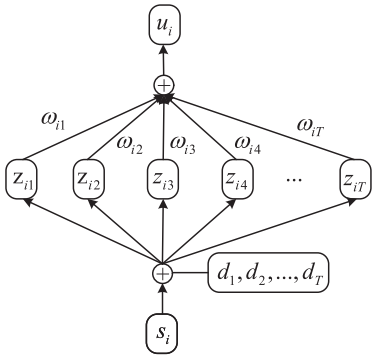


Fig. 6. The demand recognition mechanism.

where $sum(\cdot)$ denotes the sum of every column of the vector and $I(\cdot)$ is the indicator function. Here we employ the element-wise summation function $sum(\cdot)$ to prevent the record vectors of large groups being far greater than those of small groups. Hence, we could calculate the grouped-based importance score ω_i as follows:

$$\omega_i = \rho \cdot \sigma\left(Z_i^T V R_{i*}^T\right) + (1-\rho) \cdot \sigma\left(Z_i^T V \tilde{R}_{i*}^T\right), \tag{11}$$

where ρ is a balance parameter to adjust the influence from individual and user group perspectives.

Finally, by combing the influence of T patterns with the importance score $\omega_i \in \mathbb{R}^T$, we are able to comprehensively analyze users' true demands. Specifically, the final latent demand variable u_i can be obtained by the weighted summation:

$$u_i = Z_i \omega_i = \sum_{t=1}^{T} \omega_{it} z_{it}. \tag{12}$$

3.2.4 Inference Process. After building the generative model, we could provide the joint probability of the full data:

$$p(R, X, X^c, V, S) = \prod_{i,j} p(r_{ij}|u_i, v_j) p_\psi(x_i|s_i) p(s_i) p(x_j^c|v_j) p(v_j).$$

We need to obtain the posterior distribution of the latent variables to inference the model. Nevertheless, it is difficult to get an analytical solution. To address this problem, we utilize a popular approximation, i.e., **Stochastic Gradient Variational Bayes (SGVB)** estimator, to construct a stochastic gradient ascent solution for our proposed framework. Specifically, we realize an inference network parameterized by ϕ for efficient posterior inference of competency variable s_i. The **multi-layer perception network (MLP)** with L layers is chosen as the inference network structure. Thus, the variational distribution q is parameterized by ϕ and x_i:

$$q(S) = \prod_i q_\phi(s_i|x_i) = \prod_i \mathcal{N}(\mu_\phi(x_i), \Sigma_\phi(x_i)). \tag{13}$$

Then the inference process can be defined as follows:

(1) For each layer l of the inference network:

(a) Draw the mth column of weight matrix W_l:

$$W_{l,*m} \sim \mathcal{N}\left(0, \lambda_w^{-1} I_{K_l}\right).$$

(b) Draw the bias vector b_l from $b_l \sim \mathcal{N}(0, \lambda_w^{-1} I_{K_l})$.

(c) Draw the ith row of hidden state h_l by:

$$h_{l,i*} \sim \mathcal{N}\left(Sigmoid(h_{l-1,i*})W_l + b_l, \lambda_s^{-1} I_K\right).$$

(2) Draw mean and covariance of latent variable by:

$$\begin{aligned} \mu_{s_i} = \mu_\phi(x_i) &\sim \mathcal{N}\left(h_L W_\mu + b_\mu, \lambda_s^{-1} I_K\right), \\ \Sigma_{s_i} = \Sigma_\phi(x_i) &\sim diag\left(\mathcal{N}\left(h_L W_\Sigma + b_\Sigma, \lambda_s^{-1} I_K\right)\right). \end{aligned}$$

Draw latent variable s_i from:

$$s_i \sim \mathcal{N}(\mu_{s_i}, \Sigma_{s_i}),$$

where λ_w, λ_s are the hyperparameters and K_l is the number of columns of the hidden state h_l. Besides, λ_s is often supposed to be infinite in VAE so that the Normal distribution would degrade to Dirac delta function [48]. In such a case, VAE could work like other common neural networks and improve computing efficiency.

With the help of reparameterization trick [43], we could easily gain a differentiable sampling process of s_i. Specifically, we draw sample by $\epsilon \sim \mathcal{N}(0, I)$ and let $s_i = \mu_{s_i} + \epsilon \Sigma_{s_i}$.

Thus, the **Evidence Lower Bound (ELBO)** of our DCBVN framework is formulated as:

$$\begin{aligned} \mathcal{L}(q) = &\log p(R|U, V) + \mathbb{E}_q[\log p_\psi(X|S)] + \log p(X^c|V) \\ &+ \log p(V) - \mathbb{KL}(q_\phi(S|X)||p(S)). \end{aligned} \tag{14}$$

It can be seen that the ELBO is separated into three pieces: prediction loss, reconstruction loss, and prior loss.

First, the prediction loss is frequently utilized in LFMs to maximize the log-likelihood of records:

$$\log p(R|U, V) = -\sum_{i,j} \frac{C_{ij}}{2} \left(r_{ij} - u_i^T v_j\right)^2.$$

Second, the reconstruction loss measures the generative quality of skill profiles of users and courses. For the user-generative process, in consideration of the intractability of computing expected values, we employ Monte Carlo sampling from ϵ following Law of the Unconscious Statistician:

$$\begin{aligned} \log p(X^c|V) &= \sum_j p\left(x_j^c|v_j\right), \\ E_q[\log p(X|S)] &= \sum_i \frac{1}{D} \sum_d p_\psi\left(x_i|s_i^{(d)}\right). \end{aligned}$$

Finally, the prior loss can be viewed as a regularization term:

$$\log p(V) = -\frac{\lambda_v}{2} \sum_j \|v_j\|^2,$$

$$\mathbb{KL}(q_\phi||p) = \frac{1}{2} \sum_i \left[(\mu - \mu_{s_i})^T \Sigma^{-1} (\mu - \mu_{s_i}) + tr(\Sigma^{-1} \Sigma_{s_i}) + \log \frac{|\Sigma|}{|\Sigma_{s_i}|} - K\right].$$

Thus, stochastic optimization methods such as Adam [23] can be used to operate the ELBO.

4 DCCAN FRAMEWORK

DCBVN can effectively deal with the problem of predicting employees' learning demands. During the modeling process, DCBVN has to first extract latent competency representation from the skill profile to identify the current competency of each employee. However, in real-world scenarios, we may be not able to collect all employees' skill profiles. In such a case, the proposed DCBVN framework cannot analyze employees' current competencies and thus is unable to recommend appropriate courses. Collaborative filtering models without the usage of skill information can still be applied to these employees for recommendations. However, it would be quite hard to capture the real user demand when ignoring their skill distribution. Moreover, such recommender systems will lack interpretability. Besides, some existing skill profiles may be very sparse and lack enough competency information. When recommending courses to these employees, the sparsity problem would greatly disturb the modeling performance of DCBVN. Consequently, it is crucial and meaningful to infer the competency and demand representations for those employees who have few and zero skill labels.

Along this line, based on the proposed DCBVN framework, we further present the extended DCCAN framework for handling the absence of employees' skill profiles. In fact, DCBVN can be viewed as a component of DCCAN. DCCAN is capable of modeling the competency and demand representations for employees with sparse and even no skill labels for better recommendation results by incorporating the competency information of existing users. Intuitively, the competency representation of the employee with no skill label can be inferred from their neighbors. Our DCCAN framework is able to handle many types of neighbors concurrently. In this article, we mainly adopt two types of neighborhoods. On the one hand, the employees who have learned the same courses may have similar skill profiles. On the other hand, the employees doing similar work may share similar competencies. We have validated that users have more similarities in the skill profiles of their neighbors in Section 2. According to this observation, we first describe the whole user communities as two undirected and unweighted graphs, based on the two user neighborhood definitions, respectively. Then we design a graph-attentive network architecture to explore the potential competencies of a user from her neighbors' competency variables. Next, we propose a novel multi-head integration mechanism to combine the competency variables learned from two graphs to obtain the supplementary competency variables. The aggregation process of the graph attentional layer and multi-head integration is illustrated by Figure 7. Thus, we can further transform the supplementary competency variables into demand variables by the introduced demand recognition mechanism. Similar to DCBVN, we can then employ the latent variables to generate explainable recommendations.

4.1 Employee Competency Graph

We will start by defining the two types of employee competency graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, representing the user relationship from learning and working aspects, respectively. For constructing either of the two graphs, each user i in our dataset is viewed as a node and we denote $\mathcal{N}_i$ as the neighborhoods of node i. In the learning relationship graph $\mathcal{G}_1$, one user is connected to the other one only when they have learned at least one shared course. On the other hand, in the working relationship graph $\mathcal{G}_2$, the connection between two users means they are doing similar work. Here, we continue to use the partitioning method introduced in Section 3.2.3, i.e., dividing users into different groups based on their departments and positions. By comparison, $\mathcal{G}_2$ is composed of many disconnected subgraphs and more sparse than $\mathcal{G}_1$.

Users in the datasets can be divide into three sets, $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$, depending on the sparsity of their skill profiles. Specifically, users in $\mathcal{I}_1$ have sufficient skill profiles (i.e., the number of skill labels

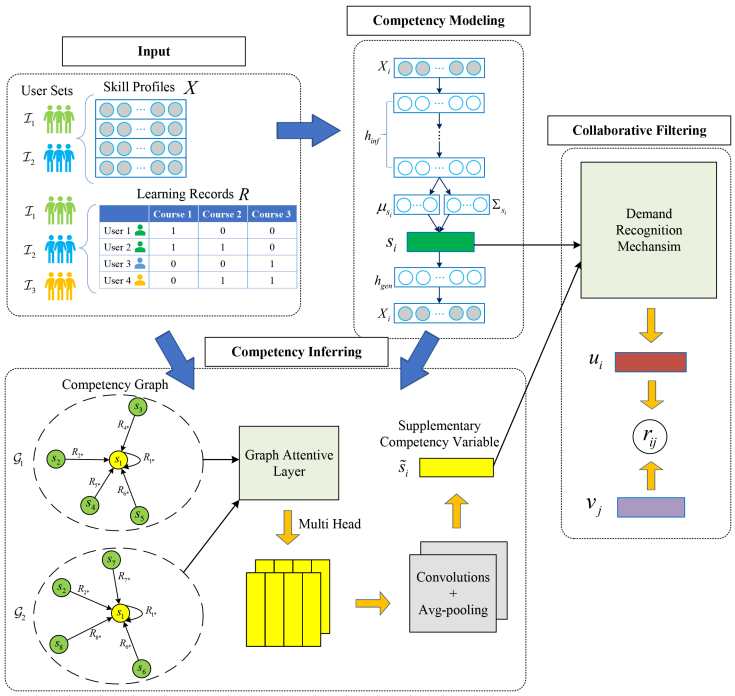


Fig. 7. The overview architecture of the DCCAN framework.

exceeds a predefined threshold) while users in $\mathcal{I}_2$ have sparse skill profiles (i.e., the number of skill labels is below the threshold). As for users in $\mathcal{I}_3$, they have no skill label. Obviously for users in $\mathcal{I}_1$, we can employ the proposed DCBVN framework to obtain the competency variables. However, DCBVN has to face great challenges when handling users in $\mathcal{I}_2$ with sparse skill profiles. Moreover, DCBVN is incapable of handling users in $\mathcal{I}_3$. To solve the problem, DCCAN aims to exploit the competency variables of the neighborhood users $l \in \mathcal{N}_i \cap \mathcal{I}_1$ for inferring the competency information of user $i \in \mathcal{I}_2 \cup \mathcal{I}_3$ by a graph attentive layer.

Methodology-wise, the attention variables for users is modeled as vectors by weighted summations of the competency variables from their neighborhood nodes. Formally, the input to the graph attentive layer is the set of competency variables $\{s_i | i \in \mathcal{I}_1\}$ and the set of rating vectors $\{R_{i*} | \forall i\}$. Then, the graph-attentive layer would produce a new set of competency variables $\{s_i | i \in \mathcal{I}_2 \cup \mathcal{I}_3\}$. Note that the initial value of $s_i, i \in \mathcal{I}_1$ could be obtained easily by a pre-trained DCBVN model, and then these variables would be updated under the unified DCCAN framework. Different from the common graph attentive network [51] which has only one type of feature, our graphs contain two types of input feature, where rating vectors are used for calculating attention scores and competency variables are used for weighted summation.

Specifically, we first perform a shared linear transformation parametrized by a weight matrix W_a on the rating vector R_{i*}. Such transformation is beneficial for reducing redundant information and capturing key feature similarities. Then, the self-attention mechanism is applied to the nodes. We calculate the attention score e_{il} by

$$e_{il} = f_a(W_a R_{i*}, W_a R_{l*}), \tag{15}$$

where $f_a(\cdot)$ is a non-linear feed-forward layer. Following [51], in our experiment, we let

$$f_a(W_a R_{i*}, W_a R_{l*}) = LeakyReLU(\delta^T (W_a R_{i*} \| W_a R_{l*})), \tag{16}$$

where $\delta \in \mathbb{R}^{2K}$ is a learnable weight vector, $\|$ means the concatenation treatment, and $LeakyReLU(\cdot)$ represents the $LeakyReLU$ activation function.

The attention score e_{il} implies the potential influence of user j's competency to user i. Since considering all the other users' influence would bring much redundant and irrelevant information, as well as increase the model complexity, we only need to consider the neighborhood users. Thus, for a specific user $i \in \mathcal{I}_2 \cup \mathcal{I}_3$, we normalize the attention scores across $\forall l \in \mathcal{N}_i \cap \mathcal{I}_1$ with the *softmax* function:

$$\hat{e}_{il} = \frac{\exp e_{il}}{\sum_{l' \in \mathcal{N}_i \cap \mathcal{I}_1} \exp e_{il'}}. \tag{17}$$

Note that for user $i \in \mathcal{I}_2$, we also add the node i itself to the set of neighborhood nodes in Equation (17). Thus, the attention mechanism would also take her own sparse skill profile into consideration. As for user $i \in \mathcal{I}_3$, there is no available skill profile for herself. Finally, the attentive competency variable is given by:

$$\hat{s}_i = \sum_{l \in \mathcal{N}_i \cap \mathcal{I}_1} \hat{e}_{il} s_l, \tag{18}$$

4.2 Multi-head Integration

For both graphs, we can obtain the attentive competency variables for target users by the graph-attentive layer. The next challenge is how to balance the influence from learning and working perspectives for each user. To this end, in this subsection, we propose a multi-head integration mechanism to produce the final supplementary competency variable. It is noticed that our multi-head Integration mechanism can actually handle multiple types of competency graphs. In this article, we take the two graphs as an example.

First, we adopt the multi-head technique [50] to enhance the representational ability and model robustness of attention results. In other words, we concurrently perform n independent attention mechanisms to obtain $2n$ attentive competency variables from two graphs based on Equation (18). In this way, each head could be viewed as one particular view to exploit similar skill topics from the neighborhood nodes. Accordingly, all the attentive competency variables from one graph could be depicted by a matrix $S_G \in \mathbb{R}^{K \times n}$. Considering that each dimension of the competency variable $\hat{s}_i$ represents a real skill topic, each row of S_G could be viewed as a n dimensions feature vector for the topic. When further combining the variables obtained from two graphs, they actually form a 2-channel feature tensor $S_a = \{S_{G_1}, S_{G_2}\} \in \mathbb{R}^{K \times n \times 2}$ for each user. If there are more graphs, we can easily extend the tensor to more channels.

Then, we choose to use a CNN [22] layer to exploit the interactions among multiple views of each skill topic at large scales, since CNN is quite suitable for extracting dominated information of feature matrix from local to global views [19]. It is worth noting that every dimension of the competency variable is related to a specific skill topic. Therefore, we must ensure the interpretability for each dimension across the whole convolution process. To achieve this goal, we need to be meticulous in designing the convolution kernel to prevent the interaction among each topic. Here, we let all the kernels' sizes in the form of $1 \times \eta$, where $\eta \leq n$. In this way, the convolution operation would only be applied to a single skill topic at one time. For each kernel, we then employ an avg-pooling operation to make sure the output's dimension is equal to $K \times 1$. Finally, we average all the convolution kernels' output vector to derive the supplementary competency variable $\tilde{s}_i \in \mathbb{R}^K$. Thus, we can automatically catch the important competency information from multiple views for

the employees' competency variables with the help of multi-head integration mechanism. Moreover, the interpretability of each dimension of the competency variable has still remained during the proposed mechanism.

After that, we can adopt the demand recognition mechanism in DCBVN to transform the supplementary competency variable $\tilde{s}_i$ into the user-demand variable u_i for the rating prediction task similar to DCBVN:

$$\begin{aligned} \tilde{u}_i &= G(\tilde{s}_i), \\ r_{ij} &\sim \mathcal{N}\left(u_i^T v_j, c_{ij}^{-1}\right). \end{aligned} \tag{19}$$

4.3 Unified Modeling

So far, we have introduced how to infer the supplementary competency variables from users' neighbors and then make rating predictions. However, users can be divided into three user sets as discussed before and the training objective for each user set is also different from each other. Therefore, it is necessary to arrange a reasonable optimization process. In this subsection, we aim to build a unified Bayesian model training paradigm for all user sets based on the introduced techniques.

First, for users in $\mathcal{I}_1$, we can still employ the ELBO loss function in Equation (14) for optimization just like what we do in DCBVN framework. Unlike in DCCAN, we also use the employee competency graphs to produce the supplementary competency variables. Therefore, we add a term in the loss to measure the Euclidean distances between the supplementary competency variables and the variables learned from real skill profiles:

$$\begin{aligned} \mathcal{L}_1 = -\sum_{i \in \mathcal{I}_1, j} \Big[&\log p(r_{ij}|u_i, v_j) + \mathbb{E}_q[\log p_\psi(x_i|s_i)] + \log p(v_j) \\ &+ \log p(x_j^c|v_j) - \mathbb{KL}(q_\phi(s_i|x_i)||p(s_i)) + \|\tilde{u}_i - u_i\|^2 \Big]. \end{aligned} \tag{20}$$

Second, for users in $\mathcal{I}_2$, we need to learn the competency variables from both real skill profiles and employee competency graphs, but we do not have to measure their distances. The loss function is given by combing the variational inference and potential skill-label inferring:

$$\begin{aligned} \mathcal{L}_2 = -\sum_{i \in \mathcal{I}_2, j} \Big[&\log p(r_{ij}|\tilde{u}_i, v_j) + \mathbb{E}_q[\log p_\psi(x_i|s_i)] + \log p(v_j) \\ &+ \log p(x_j^c|v_j) - \mathbb{KL}(q_\phi(s_i|x_i)||p(s_i)) \Big]. \end{aligned} \tag{21}$$

Third, for users in $\mathcal{I}_3$, we only need to train the supplementary competency variables. The loss function is as follows:

$$\mathcal{L}_3 = -\sum_{i \in \mathcal{I}_3, j} \Big[\log p(r_{ij}|\tilde{u}_i, v_j) + \log p(v_j) + \log p(x_j^c|v_j) \Big]. \tag{22}$$

Note that, in all of the three loss functions above, the course j will be involved in a loss only when it has been learned by at least one user in the corresponding user set. Finally, we can combine the three loss functions to build a unified loss for all users.

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3. \tag{23}$$

In the training process of DCCAN, we first initialize the model parameters and competency variables for users in $\mathcal{I}_1$ and $\mathcal{I}_2$ by a pre-trained DCBVN model. Then, we will alternately update the gradients for the three user sets. Specifically, in each training epoch, we first update the users in $\mathcal{I}_1$ and then fix their competency variables. Then, we update the users in $\mathcal{I}_2$ with the trained

competency variables for $\mathcal{I}_1$. Lastly, we update the users in $\mathcal{I}_3$. Besides, course vectors will be updated during the training epoch when the users have learned the courses.

4.4 Prediction

In this subsection, we introduce how to generate the predictions by our proposed DCBVN and DCCAN frameworks.

First, in general scenarios, let Y denote the observed data. On the basis of the trained model (either based on DCBVN or DCCAN), we can make the estimation by:

$$\mathbb{E}[r_{ij}|Y] = \mathbb{E}[u_i|Y]^T \mathbb{E}[v_j|Y] = \mathbb{E}[G(s_i)|Y]^T \mathbb{E}[v_j|Y]. \tag{24}$$

For the point estimation, we approximate the prediction as:

$$r^*_{ij} = G(\mathbb{E}[s_i])^T v_j, \tag{25}$$

where $\mathbb{E}[s_i] = \mu_{s_i}$, i.e., the mean variable in inference network. For users who have sparse or missing skill profiles, we replace the competency variable s_i with $\tilde{s}_i$.

The cold-start issue is a common and tough problem in real-world data. Actually, there may be two types of cold-start problems in our course recommender system. First, for the cold-start situation from the skill-label perspective, we have introduced how to use DCCAN framework to infer the potential competencies of employees from their neighborhood nodes. Second, in the cold-start situation from the learning record perspective, we will have no course records for the new users and cannot meet their real demands. However, DCBVN and DCCAN can still make recommendations based on the competency variable s_i. Moreover, we can also draw help from user group information to solve the cold-start problem. Specifically, we adopt the group-based importance score and set the balance parameter ρ as 0. In this way, we can still make recommendations like in the normal position. We will further discuss the different strategies for cold-start problems in the experimental part. To sum up, our DCBVN framework can greatly alleviate the cold-start problem.

5 EXPERIMENTS

In this section, we demonstrate the effectiveness of our proposed DCBVN and DCCAN frameworks from the following aspects: (1) the overall recommendation performances compared with several state-of-the-art baselines; (2) the recommendation performances of DCCAN on the cold-start problem from the skill-label perspective; (3) the analysis on cold-start scenarios from learning record perspective; (4) the analysis on the two types of employee competency graphs and multi-head integration mechanism; (5) the parameter analysis; and (6) some case studies to visualize the interpretability of our frameworks.

5.1 Baseline Approaches

To verify the effectiveness of DCBVN and DCCAN frameworks, we compare them with several baseline methods including two traditional models (i.e., WMF and CTR), five state-of-the-art neural models (i.e., NeuMF, DeepMF, LightGCN, CDL, and CVAE) and a variant of DCBVN:

- **WMF** [18]**:** This is a classic collaborative model using linear low-rank factorization for recommendation.
- **CTR** [56]**:** This is a classic hybrid model, which leverages LDA for modeling content information and combines it with traditional collaborative LFM.
- **NeuMF** [14]**:** This is a state-of-the-art collaborative model for implicit feedback, which leverages a multi-layer perceptron to learn the user-item interaction function with cross entropy loss.

- **DeepMF** [62]**:** This is a state-of-the-art collaborative model, which implements matrix factorization with multi-layer perception network.
- **LightGCN** [13]**:** This is a state-of-the-art graph based recommendation model, which learns user and item embeddings by propagating them on the user-item interaction graph.
- **CDL** [59]**:** This is a state-of-the-art hybrid recommender utilizing stacked denoising autoencoder for extracting deep latent representations.
- **CVAE** [27]**:** This is a state-of-the-art hybrid method and can be viewed as the improved version of CDL, which improves the content representations by applying variational autoencoder with Bayesian inference.
- **DCBVN-O:** This is a variant of our proposed DCBVN framework by ignoring the group information of users and only adopting the individual-based importance score in the modeling process. By comparing DCBVN-O with DCBVN, which adopts the group-based importance score, we can verify the usefulness of group information.

Among the baselines, WMF, NeuMF, DeepMF, and LightGCN only make use of course records for matrix factorization while all the other methods employ both record information and skill profile information for collaborative filtering.

5.2 Evaluation Metric

To measure the performances of recommendation results, we adopt the widely used evaluation metrics in recommender systems, i.e., R@P, P@P, and MAP@P [27, 28, 59]. R (Recall)@P counts the ratio of successfully predicted items among top-P items to all positive items for each user. On the other hand, P (Precision)@P counts the ratio of successfully predicted items among top-P items to P for each user. However, both recall and precision metrics ignore the ranking of recommendation results, while, in the practical scenarios, the ranking of courses is very important. Consequently, we adopt MAP (Mean Average Precision)@ P to consider the ranking of correct prediction items among top-P items. The final results on R@P, P@P, and MAP@P are all reported by the average value over all users. Generally, the larger the values of the three metrics are, the better the recommendation results we have.

5.3 Experimental Settings

In the experiments, we evaluated and compared the models under both the normal setting and label-missing setting. In the normal setting, we only used the 30,662 users who have skill profiles in our dataset for training and test. We selected 70% and 10% course records for each user to construct the training and validation set respectively according to the chronological order, since it might be inappropriate to use an employee's future course selection for training and then recommending past course selection during the testing process. The rest course records composed the test set. All the baseline methods would be compared in the normal setting. For DCCAN, we set the threshold as 10 skill labels to divide users into the sets $\mathcal{I}_1$ and $\mathcal{I}_2$ as introduced in Section 4.1. There is no user in $\mathcal{I}_3$ in the normal setting.

In the label-missing setting, all the 40,411 users in our dataset were used in the training stage for evaluating the recommendation results on those who have no skill profiles. Since CTR, CDL, CVAE, and DCBVN must use competency information for recommendations, we only compare the performances of WMF, NeuMF, DeepMF, and DCCAN in this setting. We still selected 70% and 10% course records for each user to construct the training and validation set, respectively, according to the chronological order. However, the construction of the test set in the label-missing setting was a little different from the normal setting. The first test set T1 contained the rest 20% course records of all the 9,749 users who have no skill profile. Thus, we are able to use

T1 for evaluating the performances of different approaches. However, we cannot know the recommendation performance disparity of DCCAN between employing and not employing skill profiles when only testing on T1. As a result, for the 30,662 users who have skill profiles, we randomly selected the rest of the course records of 10% users to compose the set T2. In both the training and test stages, all the baselines and DCCAN ignored the skill profiles of users in T2. Further, we trained another DCCAN model, named DCCAN-S, which employed the skill profiles of users in T2 for training and test stages. By comparing the results between DCCAN and DCCAN-S, we could easily validate the effectiveness of skill label inferring in DCCAN. Similar to the normal setting, we set the threshold as 10 skill labels to divide users for DCCAN model, and users who have no skill labels composed $\mathcal{I}_3$ in the label-missing setting.

For all the baseline methods, we set the dimension of latent space K as 50 for a fair comparison. Following the settings in their papers, we set the precision parameters c_{ij} as $a = 1$ and $c = 0.01$ and pretrained CTR, CDL, and CVAE with an LDA, a two-layer SDAE network, and a two-layer VAE network, respectively, to get the parameter initialization. The noise level in CDL was set to be 0.3. Then we explored the corresponding parameters of all the baselines, such as regularization parameters, learning rates, and other parameters.

In our DCBVN and DCCAN models, we also set $K = 50$ and chose the 2-layer MLP network as the inference network architecture for a fair comparison with baselines. The dimensions of the two layer were set as 200 and 100, respectively, which is the same setting in CDL and CVAE. Similar to Li and She [27], we added a balance parameter λ_r to adjust the penalty of reconstruction loss with respect to prediction terms. Then the precision parameter $\tilde{c}_{ij}$ could be set as $a = \lambda_r, b = 0.1\lambda_r, c = 0.01\lambda_r$. Thus, tuning the parameter λ_r is equivalent to tuning the precision parameter. Besides, the value of α_k in Equation (6) was set as $1/K$ for each k and the maximum number e of skills for each course as 200. Finally, we tuned the value of λ_v from the candidate set {0.01, 0.1, 1, 10, 100}.

Though using the same user inference network architecture with CDL and CVAE, we found that DCBVN could still work well without the pre-trained network parameters for content embedding. This might be because we performed the stochastic optimization methods for all the parameters synchronously, while CDL and CVAE optimized the content variables and collaborative variables alternately. Thus, DCBVN does not necessarily need the pre-training process.

5.4 Experimental Results

5.4.1 Recommendation Performance in the Normal Setting. In this part, we investigate the performance of DCBVN, DCCAN, and baseline methods in the *DLearner* dataset. As discussed in Section 5.3, we first evaluate the methods in the normal setting.

Table 2 shows the recommendation performance results of all models in the normal setting. It can be observed that our proposed DCCAN method achieves the best performances on all the metrics. Specifically, DCCAN could outperform the best baseline method, i.e., CVAE, by a relative boost of 15.19%, 11.27%, and 14.14% for the metrics R@20, P@20, and MAP@20. Besides, by comprehensively modeling the skill backgrounds and personal demands of employees, our proposed DCBVN method also performs better than all of the other methods except DCCAN. The difference between DCBVN and DCCAN in the normal setting is that DCBVN could adopt neighborhood users' competency information for completing the competency variables of users with sparse skill profiles. As a result, DCCAN could perform slightly better than DCBVN, which demonstrates the effectiveness of competency inferring process in our DCCAN method. Moreover, we can find that DCBVN-O, i.e., the variant of DCBVN, also has better performance than other baselines. DCBVN-O is a little worse than DCBVN, which demonstrates that user group information is actually beneficial when modeling the user demands. Among the baselines, CVAE, CDL, and CTR also use the

Table 2. The Overall Recommendation Performance of Different Approaches in the Normal Setting

Methods	R@10	R@20	P@10	P@20	MAP@10	MAP@20
WMF	0.1342	0.1577	0.0605	0.0429	0.0366	0.0203
CTR	0.1678	0.2150	0.0721	0.0496	0.0428	0.0238
NeuMF	0.1693	0.2178	0.0719	0.0495	0.0427	0.0238
DeepMF	0.1736	0.2228	0.0731	0.0503	0.0431	0.0244
LightGCN	0.1801	0.2296	0.0749	0.0518	0.0464	0.0254
CDL	0.1806	0.2311	0.0740	0.0511	0.0458	0.0250
CVAE	0.1890	0.2401	0.0754	0.0521	0.0468	0.0257
DCBVN-0	0.2023	0.2571	0.0791	0.0549	0.0503	0.0291
DCBVN	0.2105	0.2668	0.0810	0.0568	0.0522	0.0306
DCCAN	**0.2177**	**0.2750**	**0.0839**	**0.0590**	**0.0541**	**0.0320**

skill profile information for recommendations. However, they simply consider users' skill labels as auxiliary content information, while ignoring the close relationship between users' competencies and learning demands. They are also incapable of exploring courses' characteristics by skill profile data. Consequently, our methods could achieve better recommendation results. Meanwhile, we can find that the performances of CVAE are only lower than our methods, which demonstrate the embedding ability of variational autoencoder. By comparison, CTR, which leverages LDA in its model, does not perform well due to the sparsity of skills. CDL, which employs a common autoencoder, also performs a little worse. Besides, NeuMF and DeepMF perform better than CTR and WMF due to the superior embedding performance of neural networks. Moreover, LightGCN also exploits the graph neural network for recommendations. One important difference between LightGCN and our methods is that LightGCN adopts graph neural networks for the user-item interaction graph, while we design a graph attentive network architecture to explore the potential correlations for user competency graphs. From Table 2, we can observe that LightGCN is able to achieve comparable performance to CDL, while outperforming all the other baseline methods which only use course records. However, since LightGCN does not exploit the user skill profile information, it cannot perform better than our models.

5.4.2 Recommendation Performance in the Label-Missing Setting. Next, we evaluate the methods in the label-missing setting. Unlike the normal setting, here we mainly aim to validate the recommendation performances on users who have no skill label. As introduced in Section 5.3, we adopt two test sets T1 and T2. Table 3 and Table 4 present the recommendation performance results on T1 and T2, respectively. We can observe that DCCAN achieves obvious superior results on all three metrics. Specifically, DCCAN could outperform the best baseline method, by a relative boost of 14.08%, 18.01%, and 6.76% for the metrics R@20, P@20, and MAP@20. This is because DCCAN is capable of drawing support from the neighborhood users when modeling the user with zero skill labels, while the other methods could only make predictions based on the learning records. The superior performance clearly demonstrates the effectiveness of DCCAN on users with missing skill profiles. Further, to investigate the competency inferring ability, we also adopt the test set T2. Different from other methods, DCCAN-S uses the real skill labels for users in T2 to train the model. We can observe from Table 4 that the performance of DCCAN is close to DCCAN-S, implying that DCCAN has a good ability on inferring missing skill profiles and make recommendations based on the supplementary competency variables.

Table 3. The Recommendation Performance of Different Approaches on Test Set T1 in the Label-Missing Setting

Methods	R@10	R@20	P@10	P@20	MAP@10	MAP@20
WMF	0.1268	0.1708	0.0720	0.0514	0.0442	0.0261
NeuMF	0.1399	0.1868	0.0803	0.0588	0.0480	0.0281
DeepMF	0.1419	0.1877	0.0817	0.0593	0.0494	0.0285
LightGCN	0.1508	0.1974	0.0850	0.0622	0.0513	0.0296
DCCAN	**0.1736**	**0.2252**	**0.0981**	**0.0734**	**0.0538**	**0.0316**

Table 4. The Recommendation Performance of Different Approaches on Test Set T2 in the Label-Missing Setting

Methods	R@10	R@20	P@10	P@20	MAP@10	MAP@20
WMF	0.1318	0.1781	0.0543	0.0381	0.0321	0.0181
NeuMF	0.1522	0.1906	0.0670	0.0453	0.0425	0.0234
DeepMF	0.1519	0.2019	0.0661	0.0472	0.0420	0.0245
LightGCN	0.1629	0.2121	0.0687	0.0490	0.0435	0.0254
DCCAN	**0.1856**	**0.2370**	**0.0750**	**0.0535**	**0.0464**	**0.0271**
DCCAN-S	**0.1976**	**0.2508**	**0.0801**	**0.0562**	**0.0525**	**0.0309**

5.4.3 Cold-Start Scenarios with Zero Learning Records. In this subsection, we consider the cold-start problem from the learning record perspective, which is a common hard problem in recommender systems that new users have no historical record. It becomes even more serious on the LMS due to the massive new employees in many modern fast-paced companies [41]. Without the user-item interactions, most collaborative filtering methods would fail to make predictions. However, if the users have skill profiles, we can still make recommendations based on the competency information by our proposed methods. In this part, we provide some analysis on the cold-start scenarios with zero learning records.

We propose three different approaches to handle the cold-start problem under DCCAN framework, namely DCCAN-C1, DCCAN-C2, and DCCAN-C3. Note that we can also apply these strategies to the DCBVN framework. In DCCAN-C1, we only use the employees' competency variables for the recommendation, that is, the latent user variables are exactly the same as the skill variables of new users. In DCCAN-C2, we place no assumption on different transfer patterns to calculate the demand variables of each new user, that is, we let every transfer pattern have the same importance score. In DCCAN-C3, for each new user, we only employ the user group information to calculate the importance scores of the employees, that is, we adopt the group-based importance score and set the balance parameter ρ as 0 in Equation (11). For comparison, we also provide the result of DCCAN without cold-start restriction. Besides, most baseline recommendation methods are incapable of handling the cold-start scenarios with zero learning records. since CVAE could still make recommendations solely based on the user profiles, we also employ CVAE as the baseline method in the cold-start setting.

In this experiment, we only use the users with skill profiles to avoid disturbance from the skill-label perspective. First, we selected 50% of the users in the training stage and the rest of the users were assumed to be new users. In order to be consistent with the normal setting, we randomly chose 70% course records of training users to construct the training set for DCCAN and the three variants. The rest of the 30% course records were used for validation. Then we selected 30% course records of new users as the training set only for DCCAN. Notice that, for CVAE, DCCAN-C1,

Table 5. The Recommendation Performance Results in Cold-Start Scenarios

Methods	R@20	P@20	MAP@20
CVAE	0.1754	0.0409	0.0177
DCCAN-C1	0.1780	0.0418	0.0198
DCCAN-C2	0.1869	0.0433	0.0214
DCCAN-C3	0.1973	0.0450	0.0232
DCCAN	0.2517	0.0562	0.0301

DCCAN-C2, and DCCAN-C3, these data would not be used. The rest of the 70% course records of new users compose the test set for all four methods. The performance results are shown in Table 5.

From Table 5 we can easily find that DCCAN outperforms all the variants a lot, which clearly demonstrates the necessity of considering the personal demands of employees for the recommendation. Even DCCAN-C3 has utilized user group information, it still performs much worse than DCCAN, showing the dominant effect of individual historical records. Among the three cold-start variants, it can be observed that DCCAN-C1 still performs well, which verifies the effectiveness of our framework even on the cold-start scenario. Besides, DCCAN-C2 performs better than DCCAN-C1, which demonstrates again that it is improper to directly treat skill variable as the user demand variable. Finally, we can observe that DCCAN-C3 outperforms the other two variants. This shows the effectiveness of user group information even on the cold-start scenario. In addition, we can find that the performance of CVAE is slightly worse than DCCAN-C1, which also only uses the employees' competency variables for the recommendations. DCCAN-C2 and DCCAN-C3 both perform much better than CVAE, which demonstrates the effectiveness of user demand modeling.

5.4.4 Graph Attention Analysis. In this subsection, we mainly discuss the influences of the two employee competency graphs and the multi-head integration mechanism. We first conduct the ablation experiments to validate the importance of each competency graph and the multi-head integration mechanism. Then we will discuss the parameter sensitiveness of our designed multi-head integration mechanism.

As mentioned before, DCCAN can draw support from neighborhood users in two employee competency graphs to predict the competency variables and then transform them into the final supplementary competency variables by multi-head integration mechanism. Here, we design three different variants of DCCAN, named DCCAN-G1, DCCAN-G2, and DCCAN-G3. In DCCAN-G1, we only input the employee competency graph $\mathcal{G}_1$, which contains the user relationships from the learning perspective, during the modeling training and testing. Similarly, in DCCAN-G2, we choose to only input the employee competency graph $\mathcal{G}_2$, which contains the user relationships from the working perspective. As for DCCAN-G3, we input both of the graphs, but we simply apply the average optimization for the outputs of graph attentive layers instead of the convolution optimization. The performance results on test sets T1 and T2 are shown in Table 6. For comparison, we also provide the result of the original DCCAN model.

From Table 6, we can observe that the employee competency graph $\mathcal{G}_1$ is more helpful than $\mathcal{G}_2$ for improving the recommendation performance on both test sets, though we have found that neighbors from the working perspective may have more similarities in skill profiles as discussed in Section 2. The possible reason is that $\mathcal{G}_2$ is much sparser than $\mathcal{G}_1$, thus we could not always extract enough useful competency information from the neighborhood users in $\mathcal{G}_2$. By combining the outputs of $\mathcal{G}_1$ and $\mathcal{G}_2$, DCCAN-G3 performs better than DCCAN-G1 and DCCAN-G2 on both test sets. However, the simple average optimization in DCCAN-G3 cannot fully exploit the competency information from multi views. Thus, DCCAN-G3 performs worse than the original DCCAN model.

Table 6. The Recommendation Performance Results of Different Variants of our Methods

(a) The results on T1 in label-missing setting.

Methods	R@20	P@20	MAP@20
DCCAN-G1	0.2101	0.0656	0.0295
DCCAN-G2	0.1966	0.0631	0.0290
DCCAN-G3	0.2132	0.0674	0.0298
DCCAN	0.2252	0.0734	0.0316

(b) The results on T2 in label-missing setting.

Methods	R@20	P@20	MAP@20
DCCAN-G1	0.2243	0.0511	0.0262
DCCAN-G2	0.2122	0.0494	0.0257
DCCAN-G3	0.2289	0.0518	0.0265
DCCAN	0.2370	0.0535	0.0271

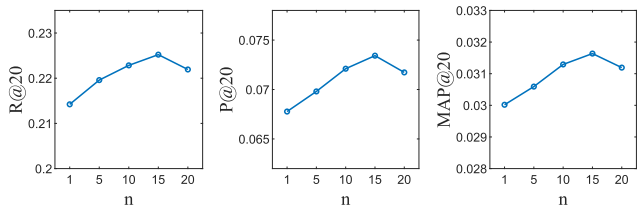


Fig. 8. The recommendation performance with different values of n.

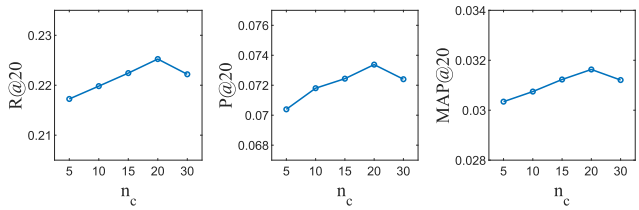


Fig. 9. The recommendation performance with different values of n_c.

In the multi-head integration mechanism, there are two main important parameters, i.e., the number of attentive heads n and the number of convolution kernels n_c. First, we present the recommendation performance results for different values of n in test set T1 in Figure 8. The n attentive heads can be viewed as n views for inferring the competency variable from one user's neighborhood nodes. When n is too large, there may be many meaningless views. Obviously, n should be smaller than the dimension K of the competency variables. In this experiment, we fix all the other parameters as the same values and tune the value of n. When $n = 1$, the multi-head attention structure actually degrades into normal attention architecture. We can observe from Figure 8 that when n is smaller than 15, DCCAN does not perform very well. However, if n is too large, the performance of DCCAN also decreases.

Next, we discuss the influence of the number of convolution kernels n_c. Usually, more convolution kernels can extract more useful features. However, since, in our DCCAN framework, the size of convolution kernel is limited. Also, the dimension of each skill topic (which is equal to the number of attentive heads n) is not large. Thus, too many convolution kernels would result in the over-fitting problem. We present the performance results for different values of n_c in test set T1 in Figure 9. It can be observed that the performance of DCCAN first increases and then decreases when n_c becomes larger.

5.4.5 Parameter Analysis. In this subsection, we mainly discuss some parameters that influence the loss functions and our demand recognition mechanism. First, we evaluate the parameter λ_r in

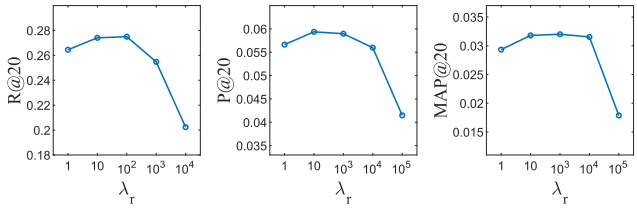


Fig. 10. The recommendation performance with different values of λ_r.

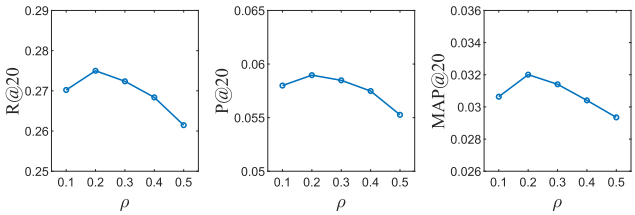


Fig. 11. The recommendation performance with different values of ρ.

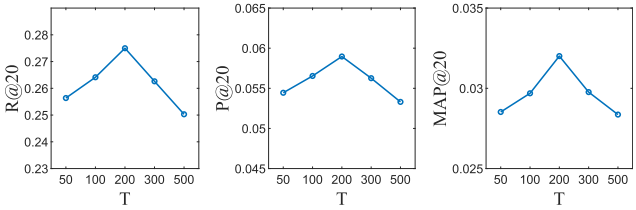


Fig. 12. The recommendation performance with different values of T.

our methods, which represents whether we concentrate more on prediction accuracy or reconstruction error. Figure 10 presents the results for different values of λ_r. When λ_r is small, the penalty of prediction loss declines. Thus, our frameworks tend to reduce the effectiveness of collaborative filtering. Moreover, it requires a larger iterative number for our methods to convergence with small λ_r. On the other hand, when λ_r grows larger, the quality of latent representations could not remain good due to the over-fitting problem.

Then we discuss the influence of balance parameter ρ, which shows the demand recognition mechanism focusing more on individual/group information. Figure 11 presents the results for different values of ρ. We can observe that, when $\rho = 0.2$, our frameworks get the best performance results. Notice that $\rho = 0.2$ does not imply that we pay much more attention to group information than individual information because we have normalized the group course records $\tilde{R}_{i*}$ so that the absolute value of each dimension of $\tilde{R}_{i*}$ is much smaller than the individual records R_{i*}.

Lastly, we investigate the influence of the number of demand transfer patterns. We assume that there are T chief patterns in the practical scenes. With small T, we could not model the demand transferring process very well. Nevertheless, if T is too large, it would lead to over-fitting. As shown in Figure 12, $T = 200$ seems to be an appropriate choice. We find that our frameworks are robust for over-fitting since they are inherently a Bayesian-generative model learning the latent distributions rather than the point estimates of variables.

Table 7. Case Studies on the Interpretable Recommendation

Top 3 topics of skills
a. distributed, hadoop, system design, apache, spark, hive, scheduling, storage, operation b. products design, product operation, promotion, impact, planning, scheme, disassembly c. retrieval, data flow, trigger, capture packets, stability, query, automation, building database
Selected courses
a. *Information security awareness training* b. *From technological backbone to manager*
Top 3 topics of demands
1. distributed, hadoop, system design, apache, spark, hive, scheduling, storage, operation 2. business, service, management, guidance, strategy, organize, examine, implement 3. achieve, dispose, service, system, solution, develop, platform, workflow, leadership
Suggested courses
1. *Office network security training* 2. *Application cases of big data on predictions* 3. *What are the success factors of a high-tech company* 4. *Engineering leadership talk: platform governance* 5. *How to use the tailor-made data storage system* 6. *Introduction to safety specification* 7. *The technology platforms in the company* 8. *Engineering leadership and strategy*

5.5 Case Study

In this subsection, we aim to provide the interpretable insights of the recommendations obtained by our proposed frameworks from both employee and course perspectives.

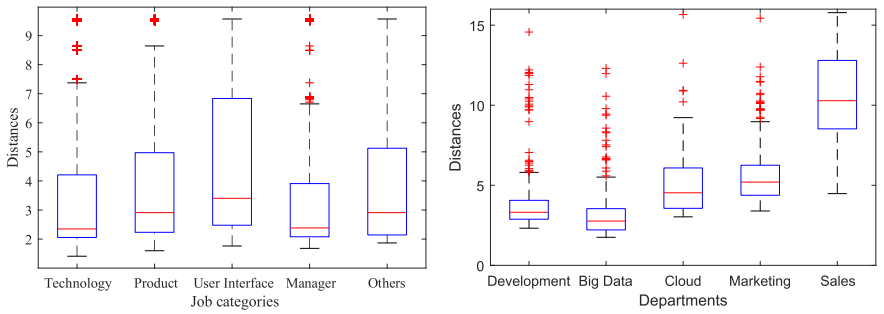
5.5.1 Employee Perspective. Table 7 shows a real user case study in our *DLearner* dataset. We first present the top 3 topics of the user's skills. From Table 7, we can speculate that she might be an R&D engineer since her current skills are mostly related to data processing and programming. Meanwhile, from the historical learning records, we can find that she preferred to learn about how to grow into a manager of the team (*course* b). Besides, she was also interested in office privacy and security (*course* a).

Then we provide the top 3 topics of the demands learned by our model and the top recommended courses in Table 7. It can be directly found from the *topic* 2 and *topic* 3 that the user-demand variable correctly captures the differences of user's current competencies and personal demands. In this way, we successfully understand her career development goal, i.e., become a good manager. Thus, we recommend some courses to enhance her business horizons (*courses* 2 and 3) and develop her leadership abilities (*courses* 4 and 8). We also suggest some technical courses (*courses* 5 and 7) to help her improve her current competencies. Besides, *courses* 1 and 6 are recommended due to her demand for information safety awareness training.

One important characteristic of DCCAN is its ability to infer the competency variable of a user from her neighbors. Here we present a case study to show the skill-topic inferring power of DCCAN. Note that DCCAN could both generate the competency variable from the real skill profile and the supplementary variable from neighbors for users. Since we have ensured the interpretability for each dimension of the competency variables in the process of graph attention and multi-head integration, the skill topics of these two types of variables have exactly the same meanings. Thus, we provide the top 3 topics of both the variables for a user in Table 8. First, since most of this user's colleagues in the same department and position have the skill topic *a*, which is relevant to

Table 8. Case Studies on Skill-topic Inferring

Top 3 topics of skills obtained by real skill profile
a. front-end development, css, nodejs, react, review, linux, mysql, database, stability, redis, html
b. project management, business understanding, programme planning, market research, planning
c. rebuilding, git, usability, video, open source, visualization, interface, programming, web, docker
Top 3 topics of skills obtained by supplementary competency variable
1. front-end development, css, nodejs, react, review, linux, mysql, database, stability, redis, html
2. test tool, function test, jenkins, continuous integration, quality assurance, stress test, usability
3. project management, business understanding, programme planning, market research, planning



(a) Learning perspective (b) Working perspective

Fig. 13. The box plots of the Euclidean distances between competency and demand variables of employees in different job categories and departments, respectively.

front-end techniques, DCCAN successfully predict the topic *a* as the top competency for her. DCCAN also predicts another popular topic of her neighbors, which is related to testing, as the top competency. Actually, this topic is in the top five topics of the competency variable obtained by real skill profile. Besides, many users who have learned the same courses with her have the skill topic *b*, which is related to management, DCCAN also predicted the topic *b* as one of the top three topics. This visualization hints that DCCAN has a good capability for capturing key information for skill-topic inferring.

Next, we present some interesting results on the differences in employees' skill backgrounds and personal career development demands. Figure 13 shows the Euclidean distances between competency and demand variables of employees in different job categories and departments, respectively. The small distance means they mainly choose courses based on their current competencies while large distance implies the high impact of various career development demands. From Figure 13(a), we can observe that the distances of employees in Technology and Manager categories are quite smaller than employees in other categories. This may be because their career development goals are tightly related to their current career directions. Meanwhile, employees in User Interface and Product categories usually have a wide variety of learning preferences, which indicates they are more likely to seek different career developments. We also present the Euclidean distances for five departments that hold a large number of employees in Figure 13(a). Here we adopt the department functions to denote the five departments but not their actual names due to the privacy prevention purpose. We can find that employees in the departments related to development and big data tend to focus on improving their current competencies. Meanwhile, employees in the departments related to cloud and marketing have more wide variety of learning preferences. We can also find

Table 9. Case Studies on Course Labeling

Exploring Selenium
a. function test, test tool, continuous integration, quality assurance, code, coverage rate, review, jenkins
b. debug, compile, shell, programming, sdk, encapsulation,match, hadoop, open source, reconfiguration
c. product design, redis, encapsulation, http, cache,spark, com, reviewing, cooperation, sdk, offline
Mr. Wu's experience sharing
1. management, product, data, business, business requirements, scheme, service, design
2. plan, innovate, management, data analysis, put on market, project management, reposition
3. mining, orientation, data analysis, reposition, trans-department, solution, investigation & research

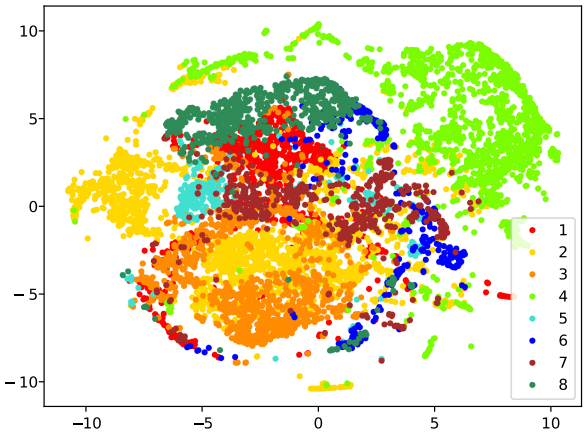


Fig. 14. The visualization of course clustering. (The clusters are distinguished by different colors.)

that the distances of employees in sales department are quite larger than all the other departments, which may because salesmen need to learn about knowledge in various fields.

5.5.2 Course Perspective. Our frameworks are also able to label the courses with the topics of skills, which is very helpful for users to make decisions. Especially when the course titles cannot directly reflect the content, users would be quite confused and most likely miss the suitable courses. For example, Table 9 presents the top 3 topics of two real courses in *DLearner* dataset. If one does not know what Selenium is, he/she may feel puzzled about the course *Exploring Selenium*. However, with the help of the skill labels learned by our models, he/she could guess that the course is about an automated testing tool (In fact, Selenium is an open-source Web automated testing tool). Similarly, if one does not know who is Mr. Wu, he/she would absolutely have no idea about the course *Mr. Wu's experience sharing*. Actually, Mr. Wu is an excellent grassroots manager in the company and we can easily infer this from the given topics.

Furthermore, we provide an overall view of the courses in *DLearner* dataset to show the interpretability of our methods. We first performed k-means clustering [1] to partition all the 8,693 courses into eight clusters according to their latent skill variables $\{s_j^c, j = 1, \ldots, 8{,}693\}$ obtained by our methods. Then we utilize the t-SNE algorithm [33] to transform the original 50-dimensional vector into a 2-dimensional space for visualization, as shown in Figure 14. We can observe that *clusters* 5, 6, and 7 are quite close. Actually, courses in these three clusters are all technical courses while *cluster* 5 focuses on underlying architecture and database; *cluster* 6 focuses on application and algorithm; *cluster* 7 focuses on web and mobile devices. Moreover, *clusters* 2 and 3 are also very close to, and are interlocked with, each other in the low-dimensional space. In fact, the courses in

these two clusters are both relevant to product and marketing. Besides, *cluster* 1 mainly contains courses about management and *cluster* 8 is mainly composed of courses helping improve the employees' personal qualities, such as communication and leadership. Lastly, *cluster* 4, which is far away from the other clusters, consists of some miscellaneous courses without a clear theme.

6 RELATED WORK

In this section, we first introduce some general recommendation algorithms and then explainable recommender systems. Lastly, we will focus on the domain of course recommender systems.

General Recommenders. For general recommenders, **Collaborative Filtering (CF)** methods have been regarded as the most popular and successful technique for mining the relevance between users and items from the historical interactions [11, 16, 29, 32, 42]. Among various CF methods, the **latent factor models (LFMs)** [6, 37, 57] are the most widely used approaches due to their advanced recommendation performance compared with the traditional neighborhood-based methods [44]. For example, the **probabilistic matrix factorization (PMF)** [37], as a representative LFM, aims to factorize the rating matrix into the product of user and item latent matrices in a low-rank space. Furthermore, Wang and Blei [56] proposed an approach to combine LFM with classic topic models [3] for integrating content information. Recently, with the successful adoption of deep learning methods in many fields [19, 40, 66], some researchers focused on building advanced hybrid recommendation models to blend LFM and neural network for collaborative information modeling [60]. For example, Wang et al. [59] exploited the Stacked Denoising Autoencoders for achieving collaboratively content embeddings with latent content variables. Furthermore, Li and She [27] utilized the Variational Autoencoder for modeling content information to construct a Bayesian hybrid recommendation model. However, it is still a big challenge to combine the competency modeling with course recommendations.

Explainable Recommenders. There are mainly two types of methods for constructing explainable recommender systems in the literature. One is post-hoc method, which chooses to separate the recommending and interpreting processes and the explanations are picked from a group of predefined templates [53]. The other is the embedded method, which tried to build a unified model to integrate both the recommending and interpreting processes [5, 36, 61]. The existing embedded methods are mainly based on reviews. For example, McAuley and Leskovec [36] obtained interpretable textual labels for latent rating dimensions from product reviews. Besides, Chen et al. [5] introduced an attention mechanism to explore the usefulness of reviews and produce review-level explanations. Furthermore, Gao et al. [12] built an explainable deep hierarchy network with an attentive multi-view learning framework. Chen et al. [7] designed a hierarchical co-attentive selector to optimize accuracy and explainability in a joint way. However, these works tend to give the explanation according to the items' characteristics. In this article, for course recommendations, we aim to provide explainable results on both the users' competencies and demands, as well as courses.

Course Recommenders. Current course recommender systems mostly focus on the scenario of student education [2, 65]. For example, Parameswaran et al. [38] studied the problem of constraint-based course recommendations for students; Vialardi et al. [52] recommended how many and which courses to study on the basis of previous students; Thai-Nghe et al. [49] proposed to predict the student performance with LFM for recommendations; and Chu et al. [8] proposed to recommend courses on the web with rule-based methods. Besides, there are also some recent works that focused on online learning platforms, such as Massive Open Online Courses. For example, Zhang et al. [63] designed a course recommender system for MOOC by association rules mining. Jing and Tang [21] proposed a content-aware algorithm framework based on content-based users' access behaviors. Hou et al. [17] paid attention to the sequence of learning curriculum and proposed a systematic recommendation methodology. Recently, some efforts were made to

enhance the learning practices of individuals and organizations in talent management [35]. For example, Klašnja-Milićević et al. [25] recognized different patterns of learning style and utilized neighborhood methods for online training recommendations. Srivastava et al. [47] studied the scenario of industrial training in organizations with sequence matching and mining.

Unlike the authors of the above works, we studied the problem of explainable personalized training course recommendations with employees' career development awareness.

7 CONCLUSIONS

In this article, we focused on studying the personalized online course recommender system for improving employees' training and development. We designed two novel explanatory end-to-end hierarchical frameworks, the Demand-aware Collaborative Bayesian Variational Network (DCBVN) framework and the Demand-aware Collaborative Competency Attentive Network (DCCAN) framework. A unique perspective of our systems is that we jointly model the employees' current competencies and their sustainable career development. Specifically, in DCBVN, we extracted latent interpretable representations from their skill profiles and then learn the personal demands of career development for different employees. Also, we designed an adapted collaborative filtering algorithm for recommending the most appropriate training courses for employees. Moreover, all the above processes are integrated into a unified Bayesian inference view. In the improved DCCAN framework, we further considered the employees who have sparse and even missing skill profiles. Specifically, we designed a graph-attentive network and a multi-head integration mechanism to infer one's competency information from her neighborhood nodes. Finally, we conducted extensive experiments on real-world data to demonstrate the effectiveness and the interpretation power of both of our proposed frameworks, as well as their robustness on sparse and cold-start scenarios.

Besides, the currently proposed models were designed for solving the personalized learning demands in the practical industrial environment. However, our proposed frameworks can also be applied to build the personalized recommender system for other learning platforms with adequate user skill-profile information. Since the current open datasets usually do not contain such user skill-profile data, we may not be able to directly transfer our frameworks to these datasets. In the future, we will try to find more scenarios to transfer our models.

REFERENCES

[1] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1027–1035.

[2] Narimel Bendakir and Esma Aïmeur. 2006. Using association rules for course recommendation. In *Proceedings of the AAAI Workshop on Educational Data Mining*, Vol. 3.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3, (Jan 2003), 993–1022.

[4] Youngchul Cha and Junghoo Cho. 2012. Social-network analysis using topic models. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 565–574.

[5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1583–1592.

[6] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient neural matrix factorization without sampling for recommendation. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–28.

[7] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-attentive multi-task learning for explainable recommendation. In *Proceedings of IJCAI*.

[8] Ko-Kang Chu, Maiga Chang, and Yen-Teh Hsia. 2003. Designing a course recommendation system on web based on the students' course selection records. In *EdMedia: World Conference on Educational Media and Technology*. Association for the Advancement of Computing in Education (AACE), 14–21.

[9] Renée E. Derouin, Barbara A. Fritzsche, and Eduardo Salas. 2005. E-learning in organizations. *Journal of Management* 31, 6 (2005), 920–940.
[10] Louis V. DiBello, Louis A. Roussos, and William Stout. 2006. A review of cognitively diagnostic assessment and a summary of psychometric models. *Handbook of Statistics* 26 (2006), 979–1030.
[11] Jingtao Ding, Guanghui Yu, Yong Li, Xiangnan He, and Depeng Jin. 2020. Improving implicit recommender systems with auxiliary data. *ACM Transactions on Information Systems (TOIS)* 38, 1 (2020), 1–27.
[12] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable recommendation through attentive multi-view learning. In *Proceedings of AAAI*.
[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648.
[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.
[15] Philipp Hennig, David Stern, Ralf Herbrich, and Thore Graepel. 2012. Kernel topic models. In *Artificial Intelligence and Statistics*. 511–519.
[16] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.
[17] Yifan Hou, Pan Zhou, Jie Xu, and Dapeng Oliver Wu. 2018. Course recommendation of MOOC with big data support: A contextual online learning approach. In *Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 106–111.
[18] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining, 2008 (ICDM'08)*. IEEE, 263–272.
[19] Zhenya Huang, Qi Liu, Enhong Chen, Hongke Zhao, Mingyong Gao, Si Wei, Yu Su, and Guoping Hu. 2017. Question difficulty prediction for READING problems in standard tests. In *Proceedings of AAAI*. 1352–1359.
[20] Zhenya Huang, Qi Liu, Yuying Chen, Le Wu, Keli Xiao, Enhong Chen, Haiping Ma, and Guoping Hu. 2020. Learning or forgetting? A dynamic approach for tracking the knowledge proficiency of students. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–33.
[21] Xia Jing and Jie Tang. 2017. Guess you like: Course recommendation in MOOCs. In *Proceedings of the International Conference on Web Intelligence*. 783–789.
[22] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[23] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[24] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).
[25] Aleksandra Klašnja-Milićević, Boban Vesin, Mirjana Ivanović, and Zoran Budimac. 2011. E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & Education* 56, 3 (2011), 885–899.
[26] John D. Lafferty and David M. Blei. 2006. Correlated topic models. In *Advances in Neural Information Processing Systems*. 147–154.
[27] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 305–314.
[28] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1734–1743.
[29] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 689–698.
[30] Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, and Guoping Hu. 2019. EKT: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering* 33, 1 (2019), 100–115.
[31] Qi Liu, Runze Wu, Enhong Chen, Guandong Xu, Yu Su, Zhigang Chen, and Guoping Hu. 2018. Fuzzy cognitive diagnosis for modelling examinee performance. *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, 4 (2018), 48.
[32] Zheng Liu, Xing Xie, and Lei Chen. 2018. Context-aware academic collaborator recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD 2018),* (London, UK, August 19-23, 2018). ACM, 1870–1879.

[33] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[34] David J. C. MacKay. 1998. Choice of basis for Laplace approximation. *Machine Learning* 33, 1 (1998), 77–86.
[35] Nikos Manouselis, Hendrik Drachsler, Riina Vuorikari, Hans Hummel, and Rob Koper. 2011. Recommender systems in technology enhanced learning. In *Recommender Systems Handbook*. Springer, 387–415.
[36] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 165–172.
[37] Andriy Mnih and Ruslan R. Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.
[38] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. 2011. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Transactions on Information Systems (TOIS)* 29, 4 (2011), 20.
[39] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. 2016. Variational autoencoder for deep learning of images, labels and captions. In *Advances in Neural Information Processing Systems*. 2352–2360.
[40] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Chao Ma, Enhong Chen, and Hui Xiong. 2020. An enhanced neural network approach to person-job fit in talent recruitment. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–33.
[41] Chuan Qin, Hengshu Zhu, Chen Zhu, Tong Xu, Fuzhen Zhuang, Chao Ma, Jingshuai Zhang, and Hui Xiong. 2019. DuerQuiz: A personalized question recommender system for intelligent job interview. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2165–2173.
[42] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Transactions on Information Systems (TOIS)* 38, 3 (2020), 1–23.
[43] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).
[44] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, 285–295.
[45] Dazhong Shen, Hengshu Zhu, Chen Zhu, Tong Xu, Chao Ma, and Hui Xiong. 2018. A joint learning approach to intelligent job interview assessment. In *Proceedings of IJCAI*. 3542–3548.
[46] Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488* (2017).
[47] Rajiv Srivastava, Girish Keshav Palshikar, Saheb Chaurasia, and Arati Dixit. 2018. What's next? A recommendation system for industrial training. *Data Science and Engineering* 3, 3 (2018), 232–247.
[48] Robert S. Strichartz. 2003. *A Guide to Distribution Theory and Fourier Transforms*. World Scientific Publishing Company.
[49] Nguyen Thai-Nghe, Lucas Drumond, Artus Krohn-Grimberghe, and Lars Schmidt-Thieme. 2010. Recommender system for predicting student performance. *Procedia Computer Science* 1, 2 (2010), 2811–2819.
[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017), 5998–6008.
[51] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[52] Cesar Vialardi, Javier Bravo, Leila Shafti, and Alvaro Ortigosa. 2009. Recommendation in higher education using data mining techniques. In *Proceedings of the International Working Group on Educational Data Mining* (2009).
[53] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: Explaining recommendations using tags. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*. ACM, 47–56.
[54] Richard A. Voorhees. 2001. Competency-based learning models: A necessary future. *New Directions for Institutional Research 2001*, 110 (2001), 5–13.
[55] Hanna M. Wallach, David M. Mimno, and Andrew McCallum. 2009. Rethinking LDA: Why priors matter. In *Advances in Neural Information Processing Systems*. 1973–1981.
[56] Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 448–456.
[57] Chao Wang, Qi Liu, Runze Wu, Enhong Chen, Chuanren Liu, Xunpeng Huang, and Zhenya Huang. 2018. Confidence-aware matrix factorization for recommender systems. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.
[58] Chao Wang, Hengshu Zhu, Chen Zhu, Xi Zhang, Enhong Chen, and Hui Xiong. 2020. Personalized employee training course recommendation with career development awareness. In *Proceedings of the Web Conference 2020*. 1648–1659.
[59] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[60] Hao Wang, Shi Xingjian, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *Advances in Neural Information Processing Systems*. 415–423.
[61] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 587–596.
[62] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems.. In *Proceedings of IJCAI*. 3203–3209.
[63] Hao Zhang, Tao Huang, Zhihan Lv, SanYa Liu, and Zhili Zhou. 2018. MCRS: A course recommendation system for MOOCs. *Multimedia Tools and Applications* 77, 6 (2018), 7051–7069.
[64] Chen Zhu, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li. 2018. Person-job fit: Adapting the right talent for the right job with joint representation learning. *ACM Transactions on Management Information Systems (TMIS)* 9, 3 (2018), 1–17.
[65] Fan Zhu, Horace H. S. Ip, Apple W. P. Fok, and Jiaheng Cao. 2007. PeRES: A personalized recommendation education system based on multi-agents & SCORM. In *Proceedings of the International Conference on Web-Based Learning*. Springer, 31–42.
[66] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. 2017. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*. 465–476.
[67] Yuan Zuo, Junjie Wu, Hui Zhang, Hao Lin, Fei Wang, Ke Xu, and Hui Xiong. 2016. Topic modeling of short texts: A pseudo-document view. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2105–2114.

Received January 2021; revised July 2021; accepted September 2021