PTADisc: A Cross-Course Dataset Supporting Personalized Learning in Cold-Start Scenarios (Appendix)

Contents

Our code, dataset, and detailed description for the dataset are available at `https://github.com/wahr0411/PTADisc.git`.

A Related Work

A.1 Cognitive Diagnosis and Knowledge Tracing

The goal of cognitive diagnosis (CD) is to assess students' level of proficiency of different knowledge concepts through the prediction process of student performance, given students' exercise records, *aka* response logs. Traditional psychometric-based methods include Item Response Theory (IRT) [18], MIRT [19] and DINA [4, 22]. These methods depend on manually designed functions and the effectiveness requires a large number of psychological experiments to verify, which are labor-intensive and lack of ability to capture complex relationships between students, problems, and knowledge concepts. Recent years, with the development of artificial intelligence [10], deep learning-based cognitive diagnosis models have been developed [23, 6, 9, 28]. Specifically, NCD [23] incorporates neural networks to learn the complex exercising interactions. And RCD [6] models the interactive and structural relations via a multi-layer student-problem-concept relation map.

The goal of knowledge tracing (KT) is to dynamically model students' knowledge proficiency through her historical learning records, so as to predict her performance to new problems. Traditional probabilistic KT models assume students' knowledge state as a set of binary variables where each variable represents whether a student masters an individual concept or not, such as Bayesian Knowledge Tracing (BKT) [3]. Recent years, deep learning-based KT models are proposed for learning valid representations especially when large amounts of data are available, such as DKT [21] and DKVMN [26]. Further, from the perspective of model structure, a few methods based on transformers (SAKT [15]), GNNs (SGKT [24]), and pre-training frameworks (PEBG [11]) are proposed.

A.2 Cross-Domain Recommendation

Cross-domain recommendation is a promising method to alleviate data sparsity and the cold-start problem [30, 27, 2]. Several models have been proposed, including CMF [20], which uses shared parameters for all domains, and CST [14], which transfers knowledge about users and items from auxiliary data sources. Mapping-based methods have been shown to be effective in solving cold-start recommendation problems [12], by learning a mapping function from the source domain to the target domain. However, these methods have limited generalization ability for cold-start items or users. To address this issue, TMCDR [29] introduces meta learning to improve the generalization ability and PTUPCDR [30] further improves TMCDR by learning personalized bridges for each user. While the cross-domain problem has been widely explored in the recommendation domain, there is limited research on cross-course learner modeling in personalized learning.

A.3 Other educational applications

Prerequisite discovery refers to the task of identifying and establishing the sequence or order in which concepts or topics should be learned or presented, ensuring that foundational concepts are understood before more advanced ones [13]. Suppose a MOOC corpus is composed by n courses in the same subject area, denoted as $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_i, \cdots, \mathcal{D}_n\}$, where $\mathcal{D}_i$ signifies an individual course. Course concepts are subjects taught in the course, i.e., the concepts not only mentioned but also discussed and taught in the course. Let us denote the course concept set of $\mathcal{D}$ as $\mathcal{C} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_n$, where $\mathcal{C}_i$ representing the concepts intrinsic to $\mathcal{D}_i$. Prerequisite relation learning in MOOCs is formally defined as follows. Given a MOOC corpus $\mathcal{D}$ and its corresponding course concepts $\mathcal{C}$, the objective is to learn a function $\mathcal{P} : \mathcal{C}^2 \to \{0, 1\}$ that maps a concept pair $\langle a, b\rangle$, where $a, b \in \mathcal{C}$, to a binary class that predicts whether a serves as a foundational prerequisite for concept b.

Computerized adaptive testing (CAT) is an emerging testing format in many standardized examinations, aiming to rapidly and accurately diagnose a candidate's level of knowledge mastery through personalized test items [1]. Let's conceptualize a set of students represented by $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, a problem set represented by $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$ and a set of knowledge concepts represented by $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$ related to the problems. We denote the record of student s_i answering problem p_j as a triplet $r_{ij} = \langle s_i, p_j, a_{ij}\rangle$, where a_{ij} equals 1 if s_i answers p_j correctly, and 0 otherwise. Problem set $\mathcal{P}$ is divided into a tested set $\mathcal{P}_T$ and an untested set $\mathcal{P}_U$. When introduced to a novel student $s_i \in \mathcal{S}$, a problem pool $\mathcal{P}$ with knowledge concepts $\mathcal{C}$, the challenge is to architect a strategy $\mathcal{A}$ to select a X-size question set $\mathcal{P}_T = \{p^*_1, p^*_2, \ldots, p^*_X\}$ step by step that has the maximum quality and diversity. Prior to the testing phase, we set up an abstract cognitive diagnosis model $\mathcal{M}$ with parameters θ capturing knowledge states. During testing, at step $t(1 \leq t \leq X)$, we select one question $p_t^* = \mathcal{A}(\mathcal{P}_U, \mathcal{M})$, then observe a new interaction test record $r_{it}^* = \langle s_i, p_t^*, a_{it}^* \rangle$ and update the knowledge states, i.e., $\boldsymbol{\theta}$, in $\mathcal{M}$ instantly. After testing, we measure the effectiveness of $\mathcal{A}$ by computing $\mathrm{Inf}(\mathcal{A})$ and $\mathrm{Cov}(\mathcal{A})$, where $\mathrm{Inf}(\mathcal{A})$ denotes the measurement of quality and $\mathrm{Cov}(\mathcal{A})$ denotes the measurement of diversity.

Educational recommendation lies in constructing a recommend system that can process the interactions between students and questions. This system should be capable of making appropriate learning suggestions to students [8]. In the context of a digital educational platform, assume there are N students and P problems. We record the exercising process of a certain student $n = \{(p_1, r_1), (p_2, r_2), \cdots, (p_t, r_t)\}, n \in \mathcal{N}$, where $p_t \in \mathcal{P}$ represents the problems that student n practices at her time step t, and r_t denotes the corresponding score. Conventionally, a correct response to problem p_t is denoted by r_t equals to 1, , and an incorrect response by r_t equals to 0. Each problem $p \in \mathcal{P}$ is characterized by a triplet $p = \{w, c, d\}$. Specifically, the element w represents its text content as a word sequence $p = \{w_1, w_2, \ldots, w_W\}$. $c \in \mathcal{C}$ describes its knowledge concept coming from all K concepts. And d means its difficulty factor.

B Dataset Staticties

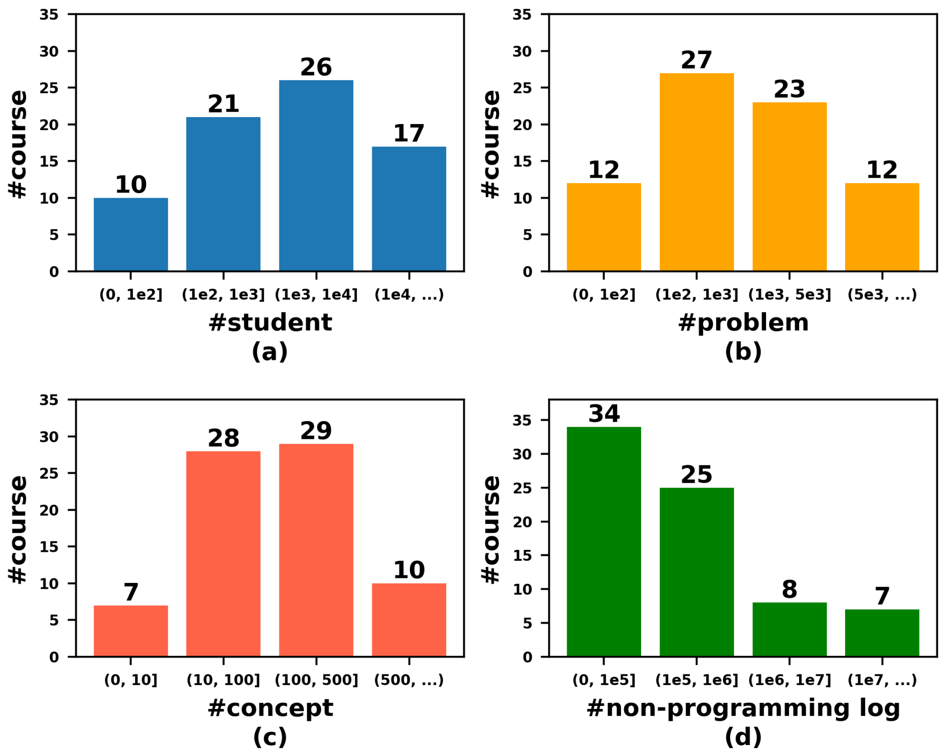

Figure 1: Distribution of the number of students, problems, concepts and non-programming logs over courses in PTADisc.

Table 1: Detailed statistics of 68 courses in PTADisc.

course name	#student	#problem	#concept	#non-programming log	#programming log
C++ programming	362,585	20,786	617	110,641,195	4,058,080
Computer App. foundation	37,280	7,304	527	110,200,887	216,146
C programming	1,074,901	43,140	1,069	97,385,606	164,459,382
DS. and algorithm analysis	294,236	29,914	897	41,667,618	-
Python programming	277,621	28,396	817	39,087,014	21,827,297
Java programming	198,335	24,524	906	23,542,208	8,461,930
Computational thinking foundation	34,519	4,500	401	16,871,472	2,212
Information technology	36,700	6,499	516	9,935,894	159
Computational thinking	45,329	8,399	477	6,504,414	19,085
Database principle	33,031	10,428	758	5,219,955	1,039
Information processing technology and App.	17,152	2,428	221	3,127,821	7,536
Computer network	20,333	7,372	609	2,451,121	1,023
Database technology and App.	12,646	3,616	325	2,363,206	-
Introduction to computer science	26,884	4,077	419	2,108,904	13,159
Operating system	18,237	8,563	726	1,777,233	64,947
Object oriented programming java	7,552	2,993	315	944,474	395,830
Principles of computer composition	11,699	5,474	475	755,982	34,456
Web front-end technology	6,612	4,247	438	572,879	876
Compilation principle	5,635	1,830	263	493,449	65,249
Thinking by data	2,597	1,027	156	445,636	-
Multivariate statistical analysis	1,704	790	83	416,427	62,393
Linux system	4,398	2,672	284	391,434	1,026
Computer and problem solving	14,241	1,518	226	370,605	522
Software engineering	4,197	2,815	335	270,692	-
Assembly language programming	2,830	1,300	206	200,231	26
Machine learning	2,073	1,984	331	197,685	500
Csharp programming	2,134	1,981	91	194,984	76,053
Java web	4,783	1,547	215	189,171	332
Big data processing technology	1,211	1,055	155	165,573	-
VB programming	2,032	1,273	113	157,360	5,829
Discrete mathematics and App.	3,516	1,278	123	147,522	17,855
Digital image processing	1,984	956	218	142,497	-
English	1,952	660	19	139,328	-
Software project management	850	1,753	133	133,824	970
Scala programming	1,378	616	124	127,382	9,367
Literature and history	2,306	886	31	126,748	-
Discrete mathematics	1,694	608	92	125,378	58,724
Fortran programming	1,398	858	106	123,498	248,698
Intro to algorithm competition	2,699	1,403	229	119,565	190,772
Data warehouse and data mining	689	777	97	117,717	-
Practice of statistics	210	394	35	99,217	-
Principles of information security	1,844	1,298	201	93,303	10,055
Software design and architecture	496	360	43	82,141	16,658
Single chip microcomputer principle and App.	784	505	88	80,977	-
Network programming technology	1,236	362	71	77,571	-
Digital logic	1,805	348	64	76,986	382
Introduction to computer	1,801	546	99	63,889	-
Numerical analysis	610	1,111	241	54,558	17,348
Big data management	217	640	65	51,724	-
Psychology	296	111	1	50,180	-
Probability and statistic	557	1,054	247	46,106	1,413
Problem solving fundation	1,037	373	63	38,440	323
Artificial intelligence	582	353	77	28,088	-
Intro to artificial intelligence	740	301	58	24,879	11
Software testing and quality assurance	454	172	4	24,342	-
Linear algebra	494	420	100	17,341	8,811
PHP programming	165	632	153	14,462	340
Object oriented analysis and design	265	178	47	11,872	-
Introduction to internet of things	217	291	14	10,949	-
Microcomputer principle and interface tech.	422	41	19	10,213	-
Signals and systems	378	38	12	8,502	-
Calculus	154	357	120	7,024	9,085
Matlab simulation	249	76	10	6,030	6,089
Japanese	54	190	19	5,375	-
Computer system fundamentals	67	91	25	3,643	-
Introduction to cloud computing	104	59	39	3,571	-
Wireless network	60	54	26	2,242	-
Swift programming	31	102	14	2,170	-
Data visualization	87	53	6	2,105	513
Fundamentals of analogy electron technique	36	29	19	1,476	-
Politics	56	29	3	1,450	-
Tourism	22	30	1	1,320	-
Software requirement analysis and design	331	21	20	21	5,492
Haskell programming	98	3	3	-	302

C Implementation Details

CCLMF is a model-agnostic framework that can be applied to various CD or KT models. Here, we take NCD as an example and showcase the implementation details of CCLMF based on NCD, namely CC-NCD. After pre-training the NCD model in the source course, the student's proficiency representation in the source course can be obtained by extracting the corresponding row from the matrix $\mathbf{A}^s$ given the student ID i:

$$\boldsymbol{u}_i^s = \mathbf{A}_{\text{NCD}}^s[i], \tag{1}$$

where $\mathbf{A}^s$ is the student representation matrix learned by NCD.

In the meta stage, we used a two-layer perceptron (MLP) as the meta network. This meta network then generates a transformation matrix for each student as the personalized mapping function:

$$\boldsymbol{T}_{K^s \times K^t} = \text{MLP}(\boldsymbol{u}_i^s; \theta), \tag{2}$$

where θ is the parameters of MLP, and $\boldsymbol{T}_{K^s \times K^t}$ is the transformation matrix. K^s and K^t denote the dimensionality of the student proficiency representation in the source and target course respectively. Specifically, the dimensionality of the student representation is determined by the number of knowledge concepts considered.

The transformation matrix $\boldsymbol{T}_{K^s \times K^t}$ is then used to map student proficiency representation to the target course using matrix multiplication:

$$\boldsymbol{u}_i^t = \boldsymbol{u}_i^s \cdot \boldsymbol{T}_{K^s \times K^t}. \tag{3}$$

The final output $\hat{r}_i$ of CC-NCD is formulated as:

$$\hat{r}_i = L(\boldsymbol{Q}_p^t \circ (\boldsymbol{u}_i^t - \boldsymbol{h}^{diff}) \times h^{disc}; \theta_l), \tag{4}$$

where $\boldsymbol{Q}_p^t \in \{0,1\}^{1 \times K^t}$ is the concept relevancy of the problem p in the target course. $\boldsymbol{h}^{diff} \in (0,1)^{1 \times K^t}$, $h^{disc} \in (0,1)$ denotes concept difficulty and problem discrimination learned from the NCD model using data of the target source. $L(\cdot)$ denotes the Linear Layers in NCD which is shown in full paper Figure 5 and θ_l is the parameters of $L(\cdot)$.

Given the ground truth value r from $\mathcal{R}^t$, all learnable parameters are trained together with the meta network and mapping function by optimizing the cross-entropy loss function as:

$$loss_{CC-NCD} = -\sum_i (r_i \log \hat{r}_i + (1 - r_i) \log (1 - \hat{r}_i)). \tag{5}$$

During the inference stage, given a cold-start student s_j in the target course, we can get the latent proficiency representation in the target course as:

$$\boldsymbol{u}_j^t = \mathbf{A}_{\text{NCD}}^s[j] \cdot \text{MLP}(\mathbf{A}_{\text{NCD}}^s[j]; \theta), \tag{6}$$

which can be utilized to predict the student's performance in the target course via Equation (4).

D Baseline Model Details

Cognitive diagnosis models:

DINA [4, 22] is a traditional method that is well-suited for binary scoring items, and it can effectively account for student errors due to guessing or slipping.

IRT [5] is an important psychological and educational theory rooted in psychometrics, which employs a linear function to model the features of both students and problems.

MIRT [19] is a multidimensional extension of IRT, modeling multiple knowledge proficiency.

NCD [23] is the first attempt to introduce neural networks for Cognitive Diagnosis, which can model high-order and complex student-problem interaction.

RCD [6] models the interactive and structural relations via a multi-layer student-problem-concept relation map and infers students' proficiency through the representations from this map.

Knowledge tracing models:

DKT [17] is the first approach applying deep learning to knowledge tracing tasks, making use of the recurrent neural network in the process of modeling students' behavior.

DKVMN [26] makes use of a memory network, a static matrix to store all concepts and a dynamic matrix to update students' knowledge states of those concepts.

SAKT [16] employs a self-attention mechanism to capture the connections between exercises and student responses.

AKT [7] utilizes an attention mechanism to analyze the temporal gap between questions and students' prior interactions to better understand their past engagement.

GIKT [25] makes use of a bipartite graph to model the input information, namely problems and concepts, and uses graph convolutional neural network (GCN) to process the data. Then the results were sent to LSTM and get the final prediction.

SGKT [24] uses a session graph and during the process of students' answering, a gated graph neural network was set to handle the problem-concept graph.

PEBG [11] utilizes pre-training model to get the low-dimensional problem embeddings and models the relation between problems and concepts as a bipartite graph.

E Supplementary Experiment Results

We conducted experiments between the selected five courses in Figure 4(a) of the paper. The experimental results are presented in Table 2. We chose *C++ Programming* as the target course due to its wide range of correlation coefficients with other courses. The source courses are marked within brackets and are ranked from lowest to highest correlation coefficient with *C++ Programming*: 0.54 for *Python Programming* (Python), 0.59 for *Data Structure and Algorithm Analysis* (DS), 0.64 for i (Java), 0.79 for *C Programming* (C). To simulate cold-start scenarios, we sampled 5% of each student's response logs in *C++ Programming* to form the target course.

From Table 2, we can observe that CCLMF achieves a certain improvement of the two models in all source courses. Notably, the results of the NCD model reveal that the extent of model improvement is related to the correlation coefficient between the source course and the target course. The source course with the highest correlation coefficient (0.79 for *C Programming*) exhibits the most significant improvement, while the source course with the weakest correlation coefficient (0.54 for *Python Programming*) demonstrates relatively less improvement.

Table 2: CCLMF results on MIRT and NCD, taking *C++ Programming* as the target course. Source courses are marked within brackets.

Metrics	Model	no dropout	10% dropout	20% dropout	30% dropout	40% dropout	50% dropout
AUC	MIRT	0.6272	0.6218	0.6157	0.6124	0.6051	0.6057
	CC-MIRT (Python)	**0.7150 (+0.0878)**	**0.7123 (+0.0905)**	**0.7102 (+0.0945)**	0.6864 (+0.0740)	0.6801 (+0.0750)	0.6716 (+0.0659)
	CC-MIRT (DS)	0.6912 (+0.0640)	0.6933 (+0.0715)	0.6904 (+0.0747)	**0.7012 (+0.0888)**	**0.7042 (+0.0991)**	**0.6990 (+0.0933)**
	CC-MIRT (Java)	0.7132 (+0.0860)	0.7093 (+0.0875)	0.7065 (+0.0908)	0.7004 (+0.0880)	0.6975 (+0.0924)	0.6890 (+0.0833)
	CC-MIRT (C)	0.6996 (+0.0724)	0.7021 (+0.0803)	0.6935 (+0.0778)	0.6912 (+0.0788)	0.6870 (+0.0819)	0.6768 (+0.0711)
ACC	MIRT	0.7493	0.7479	0.7479	0.7471	0.6825	0.6860
	CC-MIRT (Python)	**0.7738 (+0.0245)**	**0.7721 (+0.0242)**	**0.7714 (+0.0235)**	0.7657 (+0.0186)	0.7637 (+0.0812)	0.7606 (+0.0746)
	CC-MIRT (DS)	0.7668 (+0.0175)	0.7681 (+0.0202)	0.7665 (+0.0186)	**0.7694 (+0.0223)**	**0.7706 (+0.0881)**	**0.7685 (+0.0825)**
	CC-MIRT (Java)	0.7719 (+0.0226)	0.7707 (+0.0228)	0.7706 (+0.0227)	0.7672 (+0.0201)	0.7671 (+0.0846)	0.7641 (+0.0781)
	CC-MIRT (C)	0.7703 (+0.0210)	0.7719 (+0.0240)	0.7695 (+0.0216)	0.7683 (+0.0212)	0.7671 (+0.0846)	0.7643 (+0.0783)
RMSE	MIRT	0.4919	0.4935	0.4935	0.4931	0.511	0.5106
	CC-MIRT (Python)	**0.4009 (-0.0909)**	**0.4016 (-0.0919)**	**0.4022 (-0.0913)**	0.4104 (-0.0827)	0.4118 (-0.0992)	0.4151 (-0.0955)
	CC-MIRT (DS)	0.4089 (-0.0830)	0.4074 (-0.0861)	0.4086 (-0.0849)	**0.4052 (-0.0879)**	**0.4035 (-0.1075)**	**0.4054 (-0.1052)**
	CC-MIRT (Java)	0.4021 (-0.0898)	0.4027 (-0.0908)	0.4035 (-0.0900)	0.4055 (-0.0876)	0.4062 (-0.1048)	0.4085 (-0.1021)
	CC-MIRT (C)	0.4050 (-0.0869)	0.4039 (-0.0896)	0.4062 (-0.0873)	0.4071 (-0.0860)	0.4085 (-0.1025)	0.4112 (-0.0994)
AUC	NCD	0.6981	0.6960	0.6926	0.6873	0.6846	0.6791
	CC-NCD (Python)	0.7008 (+0.0028)	0.6979 (+0.0019)	0.6943 (+0.0017)	0.6931 (+0.0058)	0.6873 (+0.0027)	0.6807 (+0.0016)
	CC-NCD (DS)	0.7225 (+0.0244)	0.7189 (+0.0229)	0.7164 (+0.0238)	0.7128 (+0.0255)	0.7078 (+0.0232)	0.6997 (+0.0206)
	CC-NCD (Java)	0.7154 (+0.0173)	0.7123 (+0.0163)	0.7079 (+0.0153)	0.7058 (+0.0185)	0.6989 (+0.0143)	0.6907 (+0.0116)
	CC-NCD (C)	**0.7663 (+0.0682)**	**0.7627 (+0.0667)**	**0.7598 (+0.0672)**	**0.7541 (+0.0668)**	**0.7486 (+0.0640)**	**0.7423 (+0.0632)**
ACC	NCD	0.7619	0.7602	0.7616	0.7552	0.7556	0.7558
	CC-NCD (Python)	0.7693 (+0.0074)	0.7675 (+0.0073)	0.7666 (+0.0050)	0.7674 (+0.0122)	0.7668 (+0.0112)	0.7668 (+0.0110)
	CC-NCD (DS)	0.7747 (+0.0128)	0.7702 (+0.0100)	0.7677 (+0.0061)	0.7673 (+0.0121)	0.7675 (+0.0119)	0.7671 (+0.0113)
	CC-NCD (Java)	0.7661 (+0.0043)	0.7695 (+0.0093)	0.7687 (+0.0071)	0.7667 (+0.0115)	0.7668 (+0.0112)	0.7660 (+0.0102)
	CC-NCD (C)	**0.7854 (+0.0235)**	**0.7875 (+0.0273)**	**0.7833 (+0.0217)**	**0.7759 (+0.0207)**	**0.7734 (+0.0178)**	**0.7710 (+0.0152)**
RMSE	NCD	0.4116	0.4124	0.4115	0.4174	0.4164	0.4156
	CC-NCD (Python)	0.4102 (-0.0014)	0.4094 (-0.0030)	0.4111 (-0.0004)	0.4115 (-0.0059)	0.4123 (-0.0041)	0.4130 (-0.0026)
	CC-NCD (DS)	0.4018 (-0.0098)	0.4059 (-0.0065)	0.4081 (-0.0034)	0.4133 (-0.0041)	0.4123 (-0.0041)	0.4181 (0.0025)
	CC-NCD (Java)	0.4088 (-0.0028)	0.4066 (-0.0058)	0.4075 (-0.0040)	0.4097 (-0.0077)	0.4104 (-0.0060)	0.4130 (-0.0026)
	CC-NCD (C)	**0.3877 (-0.0239)**	**0.3881 (-0.0243)**	**0.3913 (-0.0202)**	**0.3956 (-0.0218)**	**0.3967 (-0.0197)**	**0.4028 (-0.0128)**

References

[1] Haoyang Bi, Haiping Ma, Zhenya Huang, Yu Yin, Qi Liu, Enhong Chen, Yu Su, and Shijin Wang. Quality meets diversity: A model-agnostic framework for computerized adaptive testing. In *ICDM*, pages 42–51. IEEE, 2020.

[2] Ye Bi, Liqiang Song, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. Dcdir: A deep cross-domain recommendation system for cold start users in insurance domain. In *SIGIR*, pages 1661–1664, 2020.

[3] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[4] Jimmy De La Torre. Dina model and parameter estimation: A didactic. *Journal of educational and behavioral statistics*, 34(1):115–130, 2009.

[5] Susan E Embretson and Steven P Reise. *Item response theory*. Psychology Press, 2013.

[6] Weibo Gao, Qi Liu, Zhenya Huang, Yu Yin, Haoyang Bi, Mu-Chun Wang, Jianhui Ma, Shijin Wang, and Yu Su. Rcd: Relation map driven cognitive diagnosis for intelligent education systems. In *SIGIR*, pages 501–510, 2021.

[7] Aritra Ghosh, Neil Heffernan, and Andrew S Lan. Context-aware attentive knowledge tracing. In *SIGKDD*, pages 2330–2339, 2020.

[8] Zhenya Huang, Qi Liu, Chengxiang Zhai, Yu Yin, Enhong Chen, Weibo Gao, and Guoping Hu. Exploring multi-objective exercise recommendations in online education systems. In *CIKM*, pages 1261–1270, 2019.

[9] Jiatong Li, Fei Wang, Qi Liu, Mengxiao Zhu, Wei Huang, Zhenya Huang, Enhong Chen, Yu Su, and Shijin Wang. Hiercdf: A bayesian network-based hierarchical cognitive diagnosis framework. In *SIGKDD*, pages 904–913, 2022.

[10] Mengze Li, Tianbao Wang, Haoyu Zhang, Shengyu Zhang, Zhou Zhao, Wenqiao Zhang, Jiaxu Miao, Shiliang Pu, and Fei Wu. Hero: Hierarchical spatio-temporal reasoning with contrastive action correspondence for end-to-end video object grounding. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 3801–3810, 2022.

[11] Yunfei Liu, Yang Yang, Xianyu Chen, Jian Shen, Haifeng Zhang, and Yong Yu. Improving knowledge tracing via pre-training question embeddings. *arXiv preprint arXiv:2012.05031*, 2020.

[12] Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. Cross-domain recommendation: An embedding and mapping approach. In *IJCAI*, volume 17, pages 2464–2470, 2017.

[13] Liangming Pan, Chengjiang Li, Juanzi Li, and Jie Tang. Prerequisite relation learning for concepts in moocs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1447–1456, 2017.

[14] Weike Pan, Evan Xiang, Nathan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, volume 24, pages 230–235, 2010.

[15] Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

[16] Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. In *12th International Conference on Educational Data Mining, EDM 2019*, pages 384–389. International Educational Data Mining Society, 2019.

[17] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. *NIPS*, 28, 2015.

[18] Georg Rasch. Studies in mathematical psychology: I. probabilistic models for some intelligence and attainment tests. 1960.

[19] Mark D Reckase and Mark fD Reckase. *Multidimensional item response theory models*. Springer, 2009.

[20] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *SIGKDD*, pages 650–658, 2008.

[21] K Sohn, H Lee, X Yan, C Cortes, N Lawrence, and D Lee. Advances in neural information processing systems. *NIPS*, pages 3483–3491, 2015.

[22] Matthias Von Davier. The dina model as a constrained general diagnostic model: Two variants of a model equivalency. *British Journal of Mathematical and Statistical Psychology*, 67(1):49–71, 2014.

[23] Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yuying Chen, Yu Yin, Zai Huang, and Shijin Wang. Neural cognitive diagnosis for intelligent education systems. In *AAAI*, volume 34, pages 6153–6161, 2020.

[24] Zhengyang Wu, Li Huang, Qionghao Huang, Changqin Huang, and Yong Tang. Sgkt: Session graph-based knowledge tracing for student performance prediction. *Expert Systems with Applications*, 206:117681, 2022.

[25] Yang Yang, Jian Shen, Yanru Qu, Yunfei Liu, Kerong Wang, Yaoming Zhu, Weinan Zhang, and Yong Yu. Gikt: a graph-based interaction model for knowledge tracing. In *PKDD*, pages 299–315. Springer, 2021.

[26] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks for knowledge tracing. In *WWW*, pages 765–774, 2017.

[27] Cheng Zhao, Chenliang Li, Rong Xiao, Hongbo Deng, and Aixin Sun. Catn: Cross-domain recommendation for cold-start users via aspect transfer network. In *SIGIR*, pages 229–238, 2020.

[28] Yuqiang Zhou, Qi Liu, Jinze Wu, Fei Wang, Zhenya Huang, Wei Tong, Hui Xiong, Enhong Chen, and Jianhui Ma. Modeling context-aware features for cognitive diagnosis in student learning. In *SIGKDD*, pages 2420–2428, 2021.

[29] Yongchun Zhu, Kaikai Ge, Fuzhen Zhuang, Ruobing Xie, Dongbo Xi, Xu Zhang, Leyu Lin, and Qing He. Transfer-meta framework for cross-domain recommendation to cold-start users. In *SIGIR*, pages 1813–1817, 2021.

[30] Yongchun Zhu, Zhenwei Tang, Yudan Liu, Fuzhen Zhuang, Ruobing Xie, Xu Zhang, Leyu Lin, and Qing He. Personalized transfer of user preferences for cross-domain recommendation. In *WSDM*, pages 1507–1515, 2022.