AdaptKT: A Domain Adaptable Method for Knowledge Tracing

Song Cheng[1,2], Qi Liu[1], Enhong Chen[1], Kai Zhang[1], Zhenya Huang[1,*]
Yu Yin[1], Xiaoqing Huang[1], Yu Su[1,3]
[1]Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology, University of Science and Technology of China, [2]DingTalk, Alibaba Group, [3]iFLYTEK Research, iFLYTEK, Co., Ltd
{chsong,sa517494,yxonic,xqhuang}@mail.ustc.edu.cn;{qiliuql,cheneh,huangzhy}@ustc.edu.cn;yusu@iflytek.com

ABSTRACT

Knowledge tracing is a crucial and fundamental task in online education systems, which can predict students' knowledge state for personalized learning. Unfortunately, existing methods are domain-specific, whereas there are many domains (e.g., subjects, schools) in the real education scene and some domains suffer from the problem of lacking sufficient data. Therefore, exploiting the knowledge among other domains to improve the model's performance in the target domain remains pretty much open. We term this problem as Domain Adaptation for Knowledge Tracing (DAKT), which aims to transfer knowledge from the source domain to the target one for knowledge tracing. In this paper, we propose a novel adaptable method, namely Adaptable Knowledge Tracing (AdaptKT), which contains three phases to explore this problem. Specifically, phase I is instance selection. Given the question texts of two domains, we train an auto-encoder to select and embed similar instances from both domains. Phase II is distribution discrepancy minimizing. After obtaining the selected instances and their linguistic representations, we train a knowledge tracing model and adopt the Maximum Mean Discrepancy (MMD) to minimize the discrepancy between the distributions of the domain-specific knowledge states. Phase III is fine-tuning of the output layer. We replace the output layer of the model that trained in phase II with a new one to make the knowledge tracing model's output dimension match the number of knowledge concepts in the target domain. The new output layer is trained while other parameters are frozen. We conduct extensive experiments on two large-scale real-world datasets, where the experimental results clearly demonstrate the effectiveness of AdaptKT for solving the DAKT problem.

CCS CONCEPTS

• **Information system → Information exraction**; • **Computing methodologies** → *Knowledge representation and reasoning*.

KEYWORDS

domain adaptation, knowledge tracing, deep learning

*Corresponding Author.

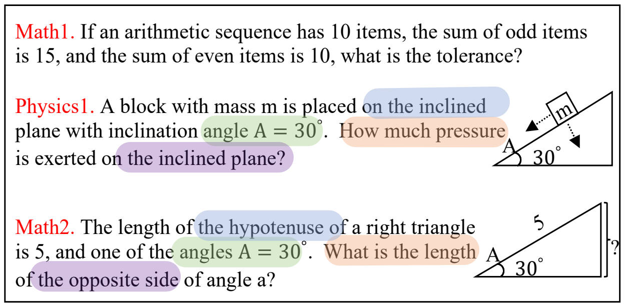
ACM Reference Format:
Song Cheng[1,2], Qi Liu[1], Enhong Chen[1], Kai Zhang[1], Zhenya Huang[1], and Yu Yin[1], Xiaoqing Huang[1], Yu Su[1,3]. 2022. AdaptKT: A Domain Adaptable Method for Knowledge Tracing. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22), February 21–25, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3488560.3498379

1 INTRODUCTION

In recent decades, computer-aided education systems (CAE) have developed rapidly, such as massive open online courses [2] and intelligent tutoring systems [5]. Holding a large volume of students' learning records, these platforms are able to recommend adaptive learning materials and arrange personalized plans for learners. To achieve this, researchers introduce the knowledge tracing problem which aims to track and estimate the knowledge states of students (e.g., the proficiencies of students on knowledge concepts), to predict their scores on the next question.

In the literature, many efforts have been made for knowledge tracing. For traditional methods [6, 7, 17, 28, 36], they mainly leverage the temporal information (e.g., learners' learning sequences) to trace the knowledge state of each concept. Simultaneously, for recent deep approaches [11, 23, 30, 37, 38], the recurrent neural networks are used to model the states of all knowledge concepts jointly. However, despite their great success, some problems restrict limit their applications in the real scenario. First, some domains usually suffer from a lack of sufficient data. For example, many schools have no systems to collect students' learning data and some new subjects (e.g., big data science) have no sufficient accumulated data. Second, existing methods are domain-specific, even if there is enough data. It is not a good choice to directly apply a model trained on a domain (e.g., math) to the other (e.g., physics), because it may lead to a significant performance drop. Unfortunately, in real world applications, the educational domains can be specified as subjects, schools, or grades. In fact, it cannot build models for each one since it is labor-intensive and resource-intensive. Therefore, utilizing the knowledge in other domains to assist training knowledge tracing model for the target one is worth exploring. To better explore this problem, we propose and define it as Domain Adaptation for Knowledge Tracing (DAKT).

However, it is a highly challenging task due to the following issues. First, different domains have domain-specific data. Some data is similar among domains, while some of them are not and even lead to negative transfer [26] during applications. Thus, we should select similar instances from these domains instead of applying all of them. As shown in Figure 1, the structure and even numerical conditions of math question in Math2 bear a striking similarity to



Math1. If an arithmetic sequence has 10 items, the sum of odd items is 15, and the sum of even items is 10, what is the tolerance?

Physics1. A block with mass m is placed on the inclined plane with inclination angle A = 30°. How much pressure is exerted on the inclined plane?

Math2. The length of the hypotenuse of a right triangle is 5, and one of the angles A = 30°. What is the length of the opposite side of angle a?

Figure 1: Different questions from two domains.

the physics question in Physics1 to some extent (e.g., marked with fluorescence), while in Math1 is totally different from Physics1. Unfortunately, current knowledge tracing models are unable to solve this problem because they only use the index of the question as features, which cannot be utilized to distinguish the different question characteristics. Second, the distributions of students' knowledge states are different from domain to domain, because the students' abilities in each domain (e.g., different subjects or different grades) are not the same. Therefore, minimizing the distribution discrepancy between domains is essential before applying the model trained on a domain to the other one. Third, as we all know, the output of knowledge tracing model is the predicted probability that the student masters the knowledge concepts, where the dimension equals the number of the knowledge concepts. However, the number of knowledge concepts varies from domain to domain, i.e., the output layer in a specific domain cannot be used in the other one.

To address the above challenges one by one, in this paper, we provide a three-phase adaptable solution, namely Adaptable Knowledge Tracing (AdaptKT). Specifically, phase I is instance selection. Given the question texts of the source and target domain, we design an auto-encoder with Bi-directional LSTM as encoder and original LSTM as decoder. To select and embed similar questions, we train it on both domains with the reconstruction error as loss function and the training process is independent from the following phases. Phase II is domain discrepancy minimizing. After obtaining the linguistic representations of the selected questions in phase I, we first only train a knowledge tracing model on the source domain. Then, we remove the model's output layer and add an adaptation layer to it. After that, we retrain the model on both domains with Maximum Mean Discrepancy (MMD) [4] as the optimization objective to minimize the discrepancy between the distributions of the domain-specific knowledge states. Phase III is fine-tuning of the output layer [13, 40]. Since the target domain does not have the same number of knowledge concepts as the source domain, the output layer we removed in phase II cannot be used for the target domain. Therefore, we add a new output layer behind the adaptation layer and train it on the target data by freezing the parameters before the output layer. We conduct extensive experiments on two large-scale real-world datasets. The experimental results demonstrate that AdaptKT is an adaptable and effective model under our three phases to achieve domain adaptation for knowledge tracing.

2 RELATED WORK

In this section, we summarize the related work from two perspectives, including knowledge tracing and domain adaptation.

Knowledge Tracing. Recent years have witnessed the development of knowledge tracing [10, 19, 27, 31, 44], which is a task of modeling students' knowledge states over time so that we can accurately predict how students will perform on future questions. One of the classic methods is Bayesian Knowledge Tracing (BKT) [7]. It is a two-stage Dynamic Bayesian Network and adopts binary variables of the Hidden Markov Model to trace the knowledge states, which indicates whether or not the student masters the knowledge concept. Another one is Performance Factor Analysis (PFA) [28] and Learning Factor Analysis (LFA) [6] which can offer a higher sensitivity to students' performance rather than their ability.

Recently, since deep learning methods [29] outperform conventional ones, Deep Knowledge Tracing (DKT) is proposed based on recurrent neural networks to model students' knowledge states in a high-dimensional and continuous representation [39]. Dynamic Key-Value Memory Networks (DKVMN) [41] is also an important deep model from another angle, which uses a key memory matrix to store knowledge concepts and a value memory matrix to store and update the knowledge state. On the basis that, many variants of them and new methods are proposed and achieved great success, such as DKT-tree [39], DKT+forgetting [23], SKVMN [1], GKT [24], etc.

Domain Adaptation To address the problem of lacking sufficient data, transfer learning technique [42, 43] develops fastly and has been widely applied in many areas [9, 13, 21]. The goal of the developed models is to transfer the latent representations obtained from a large labeled source dataset to a smaller unlabeled target dataset. Domain adaptation is a branch of transfer learning. One of the research directions of domain adaptation focuses on minimizing the discrepancy of certain feature distributions between the source and target domain. In prior studies, the most used method is maximum mean discrepancy (MMD) [4] which is a distribution discrepancy measurement. Due to the simplicity and effectiveness of MMD, researchers propose a variety of ways [20, 32] to apply it. Generally, transfer learning techniques, especially domain adaptation, have made a great success.

However, to the best of our knowledge, this is one of the few attempts to introduce domain adaptation into knowledge tracing, which can make knowledge tracing more widely applied. Therefore, domain adaptation for knowledge tracing is still an open problem. In this paper, we first adopt this problem in detail and provide a three-phase adaptable solution that integrates the question texts to solve it.

3 PRELIMINARIES

Knowledge Tracing. In an intelligent education system, suppose there are |M| students, |Q| questions and |C| knowledge concepts for one domain. We record the interaction process of a student as $\mathcal{I} = \{\varsigma_1, \varsigma_2, ..., \varsigma_T\}, \varsigma_t = (x_t, r_t)$, where x_t represents the the question that practiced by the student at interaction step t, and r_t denotes the corresponding score. Generally, if the student answers the question x_t correctly, r_t equals 1, otherwise r_t equals 0. Formally, knowledge tracing can be formulated as follows:

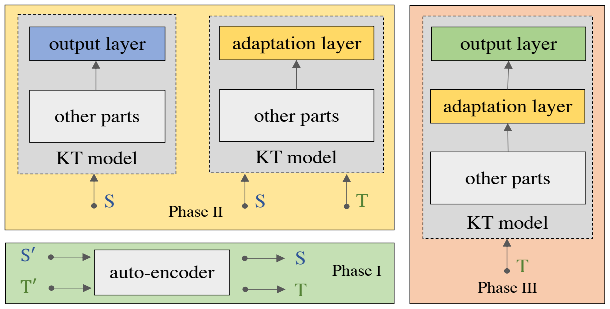


Figure 2: The flowchart overview of our work.

Definition 3.1 (Knowledge Tracing). Given the question interaction sequence $\mathcal{I}$ of each student and the materials of each question from interaction step 1 to T, the goal is two-fold: (1) tracking the change of his knowledge states and estimating how much he masters knowledge concepts from step 1 to T; (2) using the current knowledge state of him to predict the score r_{T+1} on the next candidate question x_{T+1}.

Domain Adaptation. Domain adaptation aims to solve the problem of lacking sufficient data in the target domain. Given the source data $\mathcal{D}_S = \{(x_i, y_i)\}_{i=1}^{n_S}$ and target data $\mathcal{D}_T = \mathcal{D}_{Tl} \cup \mathcal{D}_{Tu}$, where $\mathcal{D}_{Tl} = \{(x_i, y_i)\}_{i=1}^{n_{Tl}}$ is the limited labeled part and $\mathcal{D}_{Tu} = \{x_i\}_{i=1}^{n_{Tu}}$ is the abundant unlabeled part. Domain adaptation assumes that the feature space, label space of the source domain are the same as the target one, i.e., $\mathcal{X}_S = \mathcal{X}_T$, $\mathcal{Y}_S = \mathcal{Y}_T$. Meanwhile, as for the conditional probability distribution and the marginal distribution, at least one of them is inconsistent between the two domains, i.e., $\mathcal{Q}_S(y_S|x_S) \neq \mathcal{Q}_T(y_T|x_T)$ or $\mathcal{P}_S(x_S) \neq \mathcal{P}_T(x_T)$. Formally, domain adaptation can be defined as follows:

Definition 3.2 (Domain Adaptation). Given the source data $\mathcal{D}_S$ in the source domain $\mathcal{S}$ and the tarfet data $\mathcal{D}_T$ in target domain, we hope to leverage both $\mathcal{D}_S$ and $\mathcal{D}_T$ to well learn a task function $f : x_T \mapsto y_T$.

4 ADAPTABLE KNOWLEDGE TRACING

In this section, we first introduce the definition of DAKT problem, followed by an overview of our three-phase adaptable solution. Then, we introduce the technical details of AdaptKT.

4.1 Problem Definition

In our setup, based on the preliminaries in section 3, we define $\mathcal{I}_S$ as the students' interaction sequences in the source domain $\mathcal{S}$, and $\mathcal{I}_T$ as the interaction sequences in the target domain $\mathcal{T}$, respectively. However, in real education scenarios, only the limited labeled part $\mathcal{I}_{Tl}$ is existing while there is no the abundant unlabeled part $\mathcal{I}_{Tu}$. Meanwhile, the marginal distribution $\mathcal{P}(x)$ varies from $\mathcal{S}$ to $\mathcal{T}$, while the conditional distribution $\mathcal{Q}(r|x)$ of two domains is consistent because the students' scores follow the normal distribution under any questions. In general, these conditions meet the requirements of domain adaptation, therefore, we define DAKT as:

Definition 4.1 (domain adaptation for knowledge tracing). Given the source interaction sequences $\mathcal{I}_S$ in the source domain and the limited labeled part $\mathcal{I}_{Tl}$ in the target domain, we hope to leverage both $\mathcal{I}_S$ and $\mathcal{I}_{Tl}$ to well learn a knowledge tracing model $\mathcal{M}$.

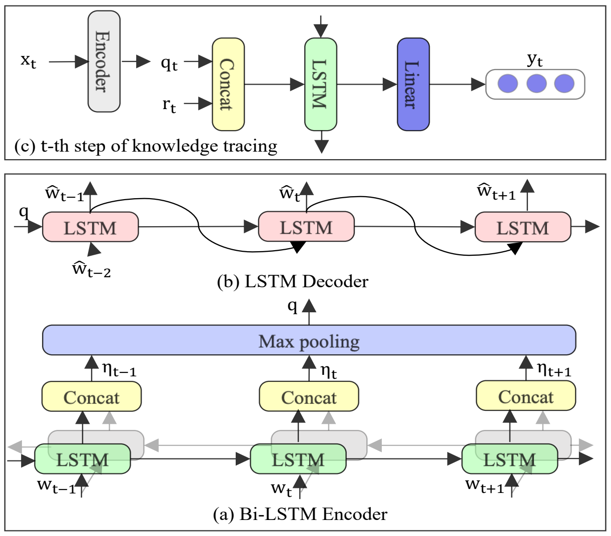


Figure 3: The architecture of the AdaptKT.

4.2 An Overview of AdaptKT

As shown in Figure 2, our solution contains three phases. Specifically, phase I is instance selection. With the reconstruction error as loss function, we use both source question texts S' and target question texts T' to train an auto-encoder, which can select similar questions from both domains and embed them as S and T. Phase II is domain discrepancy minimizing and composed of two substeps. First, we only utilize source data S to train a knowledge tracing model. Then, we remove the output layer of the KT model and add an adaptation layer to it. By treating Maximum Mean Discrepancy (MMD) as the optimization objective, we retrain this model on both S and T simultaneously. Phase III is fine-tuning of the output layer. After phase II, we obtain an incomplete KT model with no output layer. Therefore, we add a new output layer behinds the adaptation one and train it only on target data T with freezing the parameters of the adaptation layer and other parts.

4.3 Details of AdaptKT

AdaptKT contains three phases. The first one is instance selection, the second one is domain discrepancy minimizing and the third one is fine-tuning of the output layer. In this section, we first introduce the architecture of our framework AdaptKT in detail. Then, we comprehensively discuss the three phases of AdaptKT that we proposed to transfer the knowledge from the source domain to the target one.

4.3.1 The Architecture of AdaptKT. Since the index of the question that used in previous knowledge tracing models contains a little information of the questions and cannot measure the similarity that required in phase I between different questions. Therefore, we design an auto-encoder [3] to integrate the linguistic information of the question texts [10] into the knowledge tracing model and provide a technical support for achieving instance selection in phase I. As shown in Figure 3, the formal definition of it is as follows:

$$\begin{aligned} \text{encoder:} \quad & q = \pi_e(x), \\ \text{decoder:} \quad & \hat{x} = \pi_d(q), \end{aligned} \tag{1}$$

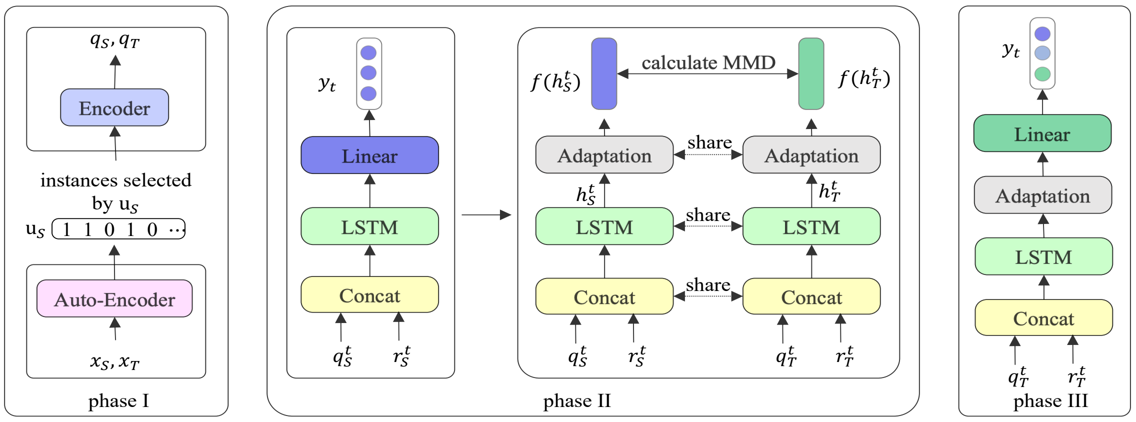


Figure 4: The three phases for transferring AdaptKT from a domain to the other one.

where $x = (w_1, w_2, ..., w_L)$ is the complete texts of the question, w is the single word which embedded by a pre-trained Word2Vec [22], and L is the number of the words. q is the embedding of the question texts and used as the representation of the question in knowledge tracing model. $\hat{x} = (\hat{w}_1, \hat{w}_2, ..., \hat{w}_L)$ is the reconstructed result of x, and $\hat{w}$ is the same.

Specifically, for the encoder π_e, we choose the Bi-LSTM that is shown in Figure 3 (a), because it can make the most of contextual content information of the question text from both forward and backward directions [12]. At each step t, the hidden states of the two directions are $\overrightarrow{h}_t^{enc}$ and $\overleftarrow{h}_t^{enc}$, respectively. Since the hidden state at each direction only contains one-side context, it is beneficial to combine the hidden state from both directions into one vector to capture the linguistic information at each step. Therefore, we obtain semantic representation at each time step t as:

$$\eta_t = \text{concatenate}(\overrightarrow{h}_t^{enc}, \overleftarrow{h}_t^{enc}), \tag{2}$$

and then combine all representations at all time steps into one single vector q with max-pooling. More formally:

$$q_j = \max(\eta_{1j}, \eta_{2j}, ..., \eta_{Lj}) \tag{3}$$

specifically, the j-th dimension of the embedding vector q can be formalized as q_j and η_{ij} is the j-th dimension of the i-th step semantic representation η_i. As for the decoder π_d, since the state-of-the-art results of LSTM in many sequence modeling problems [8, 33, 35], we adopt it that shown in Figure 3(b) as the decoder.

After pre-training the above auto-encoder, we can obtain the higher-level linguistic embeddings q of the question texts x. However, methodology-wise, after obtaining the q, we still have to integrate the score with it. As shown in Figure 3(c), inspired by [18], at each time step t, we aggregate the semantic understanding of the question texts according to the following steps. First, we give a zero vector $\mathbf{0} = (0, 0, ..., 0)^{\mathrm{T}}$, which has the same dimension as the t-th question embedding q_t, then we concatenate q_t and $\mathbf{0}$ to obtain ς_t at each time step t as:

$$\varsigma_t = \begin{cases} [q_t \oplus \mathbf{0}], & r_t = 1, \\ [\mathbf{0} \oplus q_t], & r_t = 0, \end{cases} \tag{4}$$

where $\oplus$ is the operation of concatenating two vectors. Given the interaction sequence $\mathcal{I} = \{\varsigma_1, \varsigma_2, ..., \varsigma_N\}$ of a student, where N is the number of the questions that the student answered, a LSTM architecture as described in DKT is then utilized to model the student learning processes. With the hidden state h_t of the LSTM as the simulated knowledge state of the student, following DKT, we adopt a linear prediction layer to predict the probabilities $y_t \in \mathbb{R}^Q$ of answering correctly for all questions (they are concepts in DKT). Formally, we define the probabilities y_t as follows:

$$y_t = \text{sigmoid}(\mathrm{W}_{out} \cdot h_t + \mathrm{b}_{out}), \tag{5}$$

where W_{out} and b_{out} are the parameters of the output layer.

4.3.2 The Three Phases for Transferring AdaptKT. For tackling the DAKT problem, we design three phases for AdaptKT. In this subsection, we discuss them comprehensively.

Phase I: Instance Selection. The intuition of phase I comes from a real scene. If the error of a model is small on the data of two domains simultaneously, the data of two domains have high similarity. Therefore, following previous work [34], we design a pair of encoder π_e and decoder π_d as shown in Figure 3 that introduced in section 4.3.1, to select similar questions via minimizing reconstruction error on both source and target data.

As shown in phase I in Figure 4, we train the auto-encoder independent from Phase II and III on both source domain data x_S and target domain data x_T with the reconstruction error as loss function, which is defined as follow:

$$\mathcal{R}(\hat{x}_i, x_i) = \frac{1}{L}\sum_{t=1}^{L} \|\hat{w}_t - w_t\|_2^2, \tag{6}$$

however, in order to select the similar questions, we add a filter variable $\mathbf{u}_S$ into the loss function. After that, the formulation of the optimization objective is as follow:

$$\mathcal{J}_E(\pi_e, \pi_d, \mathbf{u}_S) = \frac{1}{n_S}\sum_{i=1}^{n_S} u_S^i \mathcal{R}(\hat{x}_S^i, x_S^i) + \frac{1}{n_T}\sum_{i=1}^{n_T} \mathcal{R}(\hat{x}_T^i, x_T^i) + \Gamma(\mathbf{u}_S), \tag{7}$$

where x_S^i and x_T^i are the questions in source target domains, $\hat{x}_S^i, \hat{x}_T^i$ are the reconstructions of x_S^i and x_T^i. The $\mathbf{u}_S = (u_S^1, u_S^2, ..., u_S^{n_S})$ is the selection indicator where $u_S^i \in \{0, 1\}$ indicates (e.g., 1 for selected, 0 for unselected) whether the question i in source domain is selected or not. At the beginning of training process, all dimensions of it are initialized with 1, that is $\mathbf{u}_S = (1, 1, ..., 1)$. During the training process, the $\mathbf{u}_S$ is updated to minimize $\mathcal{J}_E$. The $\Gamma(\mathbf{u}_S)$ in the objective function is a regularization term on $\mathbf{u}_S$ to avoid the case that all values in $\mathbf{u}_S$ are zero, it is defined as follows:

$$\Gamma(\mathbf{u}_S) = -\frac{\lambda}{n_S}\sum_{i=1}^{n_S} u_S^i, \tag{8}$$

where $\lambda \in [0,1]$ represents and controls the importance of the regularization term.

The training strategy in this phase is alternate, that is, we update the parameters Θ_{auto} of auto-encoder and $\mathbf{u}_S$ alternately. When $\mathbf{u}_S$ is fixed, Θ_{auto} can be updated with back-propagation algorithm. Comparatively, when Θ_{auto} is fixed, the solution of $\mathbf{u}_S$ can be obtained as follows:

$$u_S^i = \begin{cases} 1, & \mathcal{R}(\hat{x}_S^i, x_S^i) < \lambda, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

the reasonable explanation of this alternate training strategy is two folds: (1) fixing Θ_{auto} to update $\mathbf{u}_S$, the useless questions in the source domain are removed; (2) updating Θ_{auto} with $\mathbf{u}_S$ fixed, the auto-encoder is only trained on the selected questions.

After training the auto-encoder with the above strategy, similar questions are selected by the indicator $\mathbf{u}_S$. Meanwhile, the auto-encoder is able to represent the question texts well. Thus, we abandon the decoder π_d and use the encoder π_e to embed the selected questions. That is, we adopt the linguistic embedding q that is shown in Figure 3 as the question representation. Up to now, we have completed the selection and embedding of the questions.

Phase II: Domain Discrepancy Minimizing. In fact, applying a deep learning model that trained on a domain to another one directly is not appropriate, because the data probability distributions of two domains are inconsistent while the essence of deep learning method is to learn the distribution of data. As described in the section 4.1, to transfer the knowledge tracing model from the source domain to the target one, minimizing the discrepancy between the distributions of the domain-specific knowledge states is necessary. Therefore, the phase II contains two steps. The first one is training a knowledge tracing model on source data. The second one is minimizing the distribution discrepancy between both two domains for knowledge tracing model.

As for the first step, the architecture of the knowledge tracing model that to be trained in this phase is introduced in section 4.3.1. Following previous works [23, 39], the objective function that we use to train the knowledge tracing model is defined as:

$$\mathcal{J}_{KT} = \sum\nolimits_{i=1}^{n} \sum\nolimits_{t} l(y_t^{(i)} \cdot \delta(\varsigma_{t+1}^{(i)}), r_{t+1}^{(i)}), \tag{10}$$

where n is the number of students, i and t represent the i-th student and t-th step, respectively. The $l(.,.)$ is the cross-entropy loss and $\delta(\cdot)$ is the one-hot encoding of which question is answered at time $t+1$ by the student, $\varsigma_{t+1}^{(i)}$ is the representation of the question q_t that described in section 4.3.1 , $y_t^{(i)}$ is the predicted probability that the student i answers the questions correctly at time t, $r_{t+1}^{(i)}$ is the real score that the student i gets at $t+1$ time.

In the second step, for domain discrepancy minimizing, there is a widely applied distribution distance measurement: Maximum Mean Discrepancy (MMD) [4]. It is an effective criterion that compares distributions without their density functions. More formally, with two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ on common feature space $\mathcal{X}$, MMD is defined as follows:

$$\mathcal{MMD}(\omega, \mathcal{P}, \mathcal{Q}) = \sup\nolimits_{f \in \omega}(\mathbb{E}_{\tau \sim \mathcal{P}}[f(\tau)] - \mathbb{E}_{\upsilon \sim \mathcal{Q}}[f(\upsilon)]), \tag{11}$$

where ω is the function space of the feature mapping function $f : \mathcal{X} \mapsto \mathbb{R}$, and $\mathbb{E}[\cdot]$ denotes the mean of the samples. Therefore, to minimize the discrepancy between the distributions of the knowledge states $\mathbf{h}_S = \{h_S^i\}_{i=1}^{n_S}$ and $\mathbf{h}_T = \{h_T^j\}_{j=1}^{n_T}$, we need to find an optimal feature mapping function f which can make the following objective reaches minimum value,

$$\mathcal{MMD}(\mathbf{h}_S, \mathbf{h}_T) = \left\| \frac{1}{n_S} \sum\nolimits_{i=1}^{n_S} f(h_S^i) - \frac{1}{n_T} \sum\nolimits_{j=1}^{n_T} f(h_T^j) \right\|, \tag{12}$$

As shown in Figure 4, we remove the knowledge tracing model's output layer and add a new layer, namely adaptation, to it. The adaptation layer is also a simple linear layer and the dimension of it is the same as h_S and h_T. After that, we train the KT model that has no output layer again and use the data that from both domains with the Formula (12) as the optimization objective. Taking the back-propagation algorithm to minimize the $\mathcal{MMD}(\mathbf{h}_S, \mathbf{h}_T)$, the adaptation layer will be updated and can be seen as the optimal function f, that is:

$$\alpha_t = \text{adaptation}(h_t; \Theta_{adp}) = f(h_t), \tag{13}$$

where α_t represents the output result of the adaptation layer and Θ_{adp} is the parameters of it. In general, the final definition of the objective function of phase II can be expressed as:

$$\mathcal{J}_{AdaptKT} = \mathcal{J}_{KT} + \gamma \mathcal{MMD}^2(\mathbf{h}_S, \mathbf{h}_T), \tag{14}$$

where $\gamma \in [0,1]$ is the regularization parameter that controls the importance of MMD.

Phase III: Fine-tuning of the Output Layer. After phase II, we can obtain a well-trained model which can model two domains that have different data distributions. However, since the number of knowledge concepts $|C|$ varies from the source domain to the target one, the dimensions of the output layer are different between the two domains. Therefore, before applying the model to the target domain, we have to abandon the output layer (completed in phase II) that trained on the source data and train a new one behind the adaptation layer with target data.

In phase III, we first add a new output layer which is also a full-connected layer behind the adaptation one. Then, we freeze all the parameters that in front of the output layer. That is, the frozen parameters will not be updated during the training process. Finally, we train the model (actually only the output layer is trained) on the target data by minimizing Formula (10).

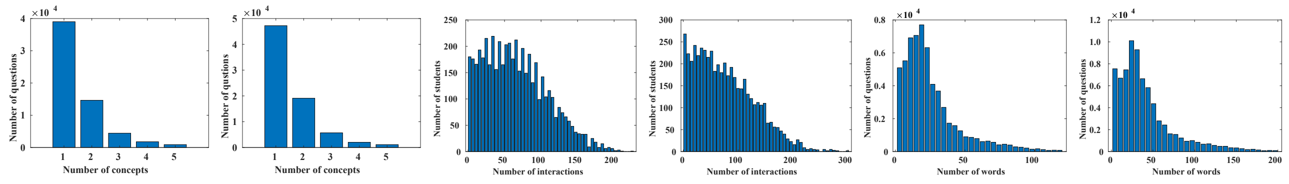
In summary, with the above three phases for AdaptKT, we can well train a domain adaptable knowledge training model. As for its performance in the real application scene, we are to validate in the following sections.

5 EXPERIMENTS

In this section, we conduct extensive experiments with AdaptKT on some knowledge tracing tasks to demonstrate our adaptable knowledge tracing method's effectiveness.

5.1 Experimental Setup

5.1.1 Datasets. We use two datasets in the experiments, namely zx.math and ax.physics, which are collected from iFlyTEK's online learning system Zhixue.com. The basic statistical information is



(a) zx.math (b) zx.physics (c) zx.math (d) zx.physics (e) zx.math (f) zx.physics

Figure 5: Distributions of the numbers of concepts per question, interactions per student and words per question.

Table 1: The statistics of the datasets.

Dataset	Questions	Students	Interactions	Concepts
zx.math	60,752	5,000	329,533	732
zx.physics	75,141	5,000	397,263	859
School-A	708	66	4.673	21
School-B	694	58	4,066	23
School-C	611	62	3,892	18
School-D	764	51	4,398	29

summarized in Table 1. The distributions of the number of knowledge concepts per question, interactions per student, and words per question are shown in Figure 5. By the way, we select and construct four school datasets from the math subject dataset. The detailed descriptions of them are as follows:

- **zx.math.** It is a math dataset that contains mathematical data. It is a large-scale dataset and we randomly selected 329,533 interactions from 5,000 students with 60,752 questions.
- **zx.physics.** It is a physics dataset, where we randomly selected 397,263 interactions from 5,000 students with 75,141 questions.
- **School A,B,C and D.** These datasets are four schools' math subject data. They are selected and constructed from zx.math. The goal of building them is to conduct the school transferring experiment, for evaluating the transferring effectiveness of AdaptKT between different schools. The meta information of them is shown in Table 1.

5.1.2 *AdaptKT Setup.*

For embedding, we incorporate Word2Vec [22] on the whole corpus to get a word to vector mapping with size 100. The hidden state dimension of the Bi-LSTM encoder is set to 50, to keep the output size of Bi-LSTM the same as the input size of the decoder. We evaluate the importance of regularization term via setting λ as 0.0, 0.25, 0.5, 0.75, 1.0. To evaluate the sensitivity of the penalty parameter γ, we set it as 0.0, 0.25, 0.5, 0.75, 1.0.

As for the transferring scenario, we utilize $X \longrightarrow Y$ represents the transferring from source domain X to target domain Y. During model training, all data of the domain X and 10% data of the domain Y are used, this setting simulates the real scene where there is amounts of source data but little target data. The left 90% data of the domain Y is used to evaluate whether the transferred model works or not on the domain Y.

Before any training step, we set the learning rate and batch size as 1e−5 and 64. The parameters are randomly initialized with uniform distribution in the range between $-\sqrt{6/(nin/nout)}$ and $\sqrt{6/(nin/nout)}$ which follows Orr et at. [25], where nin and $nout$ are the numbers of input and output size. Then, at the training process, parameters are updated by Adam optimization algorithm [14].

Table 2: Comparison methods.

Method	Question Selection	MMD	Fine Tuning
Original	-	-	-
DKT+F	-	-	✔
SKVMN+F	-	-	✔
GKT+F	-	-	✔
DKVMN+F	-	-	✔
AdaptKT-QM	-	-	✔
AdaptKT-Q	-	✔	✔
AdaptKT-M	✔	-	✔
AdaptKT	✔	✔	✔

5.1.3 *Comparison Methods.*

We compare AdaptKT with several representation methods. All these methods are able to trace knowledge and then be applied to transferring task in some ways. Specifically, these methods are:

- **Original** refers to the scenario where no transferring operations are performed. Its experimental results are the best performance among the following baselines, which are learned on the source domain and directly applied to the target domain. Only the output layer is replaced to solve the problem of different output dimensions of domains, without any transferring processes.
- **DKT** [30] is an important KT model, which applies a recurrent neural network to model the student learning process for estimating students' mastery of concepts.
- **GKT** [24] is a GNN-based knowledge tracing method, which only adopts prerequisite relationships to construct the knowledge structure.
- **DKVMN** [41] has the capability of exploiting the relationships between underlying concepts and directly outputs the learner's proficiency on each concept.
- **SKVMN** [1] unifies the strengths of recurrent modelling capacity and memory capacity. It can better discover the correlation between latent concepts and questions and trace students' knowledge state.

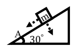
The methods above are not transferable because the output dimensions between different domains are not consistent. We add fine-tuning operation to make them transferable. The variants of them, AdaptKT and its variant are listed in Table 2, where DKT+F, GKT+F, DKVMN+F and SKVMN+F refer to DKT, GKT, DKVMN and SKVMN that add fine-tuning operation; AdaptKT-QM refers to AdaptKT with the only fine-tuning process; AdaptKT-Q refers to AdaptKT that only includes distribution discrepancy minimize and fine-tuning processes; AdaptKT-M refers to AdaptKT that only includes question selection and fine-tuning processes; AdaptKT is the complete method we proposed. We tune the parameters of all

Table 3: AUC of comparison methods on school transferring tasks.

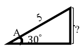
Method	A→B	B→A	A→C	C→A	A→D	D→A	B→C	C→B	B→D	D→B	C→D	D→C
Original	0.6574	0.6614	0.6583	0.6474	0.6311	0.6393	0.6422	0.6364	0.6318	0.6332	0.6302	0.6256
DKT+F	0.6936	0.6833	0.6872	0.6737	0.6724	0.6749	0.6758	0.6731	0.6694	0.6706	0.6687	0.6767
SKVMN+F	0.7087	0.7001	0.6981	0.6759	0.6792	0.6812	0.6787	0.6783	0.6739	0.6772	0.6737	0.6811
GKT+F	0.7025	0.6924	0.6943	0.6816	0.6799	0.6831	0.6824	0.6804	0.6757	0.6783	0.6762	0.6831
DKVMN+F	0.7004	0.6913	0.6923	0.6692	0.6731	0.6752	0.6704	0.6744	0.6707	0.6713	0.6702	0.6751
AdaptKT-QM	0.7013	0.6954	0.7016	0.6873	0.6872	0.6884	0.6861	0.6829	0.6719	0.6856	0.6822	0.6858
AdaptKT-Q	0.7441	0.7350	0.7438	0.7315	0.7326	0.7359	0.7365	0.7349	0.7322	0.7420	0.7321	0.7356
AdaptKT-M	0.7123	0.7028	0.7135	0.7184	0.7008	0.7097	0.7002	0.7113	0.7009	0.7284	0.7101	0.7112
AdaptKT	**0.7637**	**0.7524**	**0.7617**	**0.7582**	**0.7539**	**0.7512**	**0.7501**	**0.7529**	**0.7541**	**0.7640**	**0.7513**	**0.7551**

M1. If an arithmetic sequence has 10 items, the sum of odd items is 15, and the sum of even items is 10, what is the tolerance?

P1. A block with mass m is placed on the inclined plane with inclination angle $A = 30^{\circ}$. How much pressure is exerted on the inclined plane?

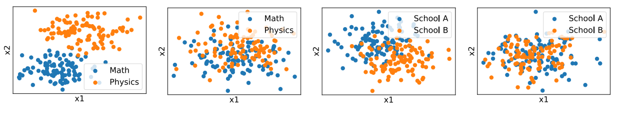


M2. The length of the hypotenuse of a right triangle is 5, and one of the angles $A = 30^{\circ}$. What is the length of the opposite side of angle a?



λ	0.0	0.5	1.0
M1	✗ unselected	✗ unselected	○ selected
M2	✗ unselected	○ Selected	○ selected

Figure 6: An example of instance selection during phase I.



(a) before MMD (b) after MMD (c) before MMD (d) after MMD

Figure 7: The distribution of the knowledge state that visualized by tSNE before and after minimizing MMD on $M \longrightarrow P$, i.e., (a) and (b), and $A \longrightarrow B$, i.e., (c) and (d).

methods in Table 2 to achieve the best performance for a fair comparison, and implement them with PyTorch on a Linux server with a Tesla K20m GPU.

5.1.4 Evaluation. For evaluating, we using 5-fold cross-validation. Specifically, we use the area under the receiver operating characteristic (ROC) curve which is termed as AUC [16], as measurement. For AUC, the larger, the better.

5.2 Selecting Instances

As described above, our AdaptKT framework includes three phases. Among them, the first phase is instance selection via pre-training an auto-encoder. That is, phase I plays a key role during the whole transferring process. The quality of similar question selection has a direct influence on the finally transferring results. Therefore, we conduct an experiment to evaluate the effectiveness of phase I. As shown in Figure 6, we choose a physics problem P1 as the anchor, and compare the selection result of two math problems M1 and M2 after Phase I. We can easily see that if we set λ as 0.0, both M1 and M2 are unselected, that is, selecting no questions can make the loss of the auto-encoder optimal. On the contrary, when the λ equals 1.0, both M1 and M2 are selected, that is, the condition $\mathcal{R}(\hat{x}_S^i, x_S^i) < \lambda$ is always satisfied and the loss is optimal. Fortunately, if we set $\lambda = 0.5$, M2 is selected while M1 is not, the explanation is selecting similar questions can work instead of adopting or abandoning all of them. In general, adopting the designed auto-encoder to select similar questions is reasonable and effective.

5.3 Minimizing Distribution Discrepancy

The second phase is domain discrepancy minimizing. We conduct experiments on $M \longrightarrow P$ and $A \longrightarrow B$ tasks, respectively, to validate the effectiveness of using MMD as optimization objective for minimizing distribution discrepancy in phase II. We compare the difference of the knowledge state's distributions of the source and target domain before and after minimizing MMD. The visualization results of dimension reduction with tSNE [15] are shown in Figure 7.

There are two interesting conclusions on both $M \longrightarrow P$ and $A \longrightarrow B$ tasks. First, before minimizing the MMD between the source and target domain, their knowledge state's distributions are not consistent because the points from different domains are far apart. However, after training the model by minimizing the MMD, the distribution of the two domains overlaps completely. This proves that minimizing distribution discrepancy is an effective method to achieve domain adaptation and it can be well achieved by minimizing MMD. Second, before minimizing MMD, the discrepancy between different subjects is greater than that between different schools because school A and school B are selected from the same subject, i.e., Math. This shows that the students' knowledge states on different subjects are obviously different, while the knowledge states on the same subject are close though the students come from different schools.

5.4 Transferring Between Domains

5.4.1 Transferring Between Schools. To investigate the performance of comparison methods on knowledge and semantic transferring between domains, we adopt four schools datasets from the same subject that shown in Table 1 as four individual domains, and term them as A, B, C and D. Therefore, there are $P_4^2 = 12$ transferring tasks (e.g., $A \longrightarrow B$) that need to be conducted. The experimental results of AUC are shown in Table 3. We can easily see that AdaptKT has the best performance on school transferring tasks among all methods. This proves that AdaptKT gains a superior transferability. However, there is more to be explained in the tables. First, comparing DKT+F, SKVMN+F, GKT+F, and DKVMN+F with original, we can find that fine-tuning with a small number of target data is helps improve knowledge tracing performance. Second, AdaptKT-QM outperforms DKT+F slightly because DKT+F and AdaptKT-QM are very similar in structure while AdaptKT-QM gets the help of question text embedding. Third, AdaptKT outperforms all its variants, which agrees with the intuition that the three transfer phases are effective.

Table 4: AUC on subject transferring tasks.

Method	Original	DKT+F	SKVMN+F	GKT+F	DKVMN+F	AdaptKT-QM	AdaptKT-Q	AdaptKT-M	AdaptKT
$M \to P$	0.5085	0.5621	0.5731	0.5713	0.5707	0.5633	0.6548	0.6030	**0.7014**
$P \to M$	0.5072	0.5603	0.5729	0.5709	0.5708	0.5628	0.6537	0.6011	**0.7132**

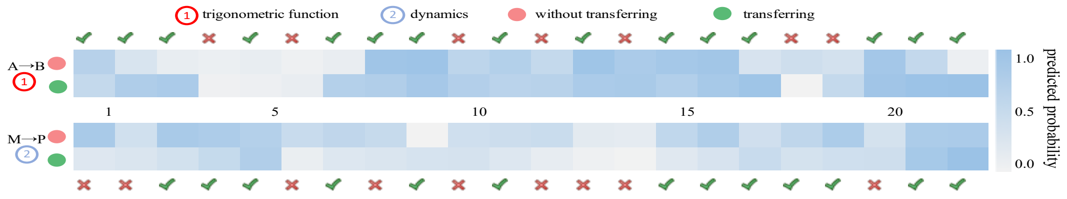


Figure 8: Visualization of knowledge states on trigonometric function and dynamics, with $A \to B$ and $M \to P$ tasks respectively.

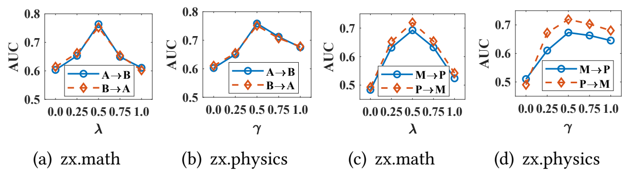


Figure 9: Analysis of the regularization parameters.

5.4.2 Transferring Between Subjects. To further explore the transferability of comparison methods under a more challenging scenario, we choose two subjects, i.e., math and physics, as two domains and term them as M and P and the datasets of them are described in table 1. Therefore, there are P_2^2 transferring tasks, i.e. $M \to P$ and $P \to M$, that need to be conducted. The experimental results in Table 4 indeed show many interesting conclusions. First, we can easily get the same conclusions as school transferring experiments; that is, AdaptKT performs best and the three transfer phases work on subject transferring tasks. This further proves the superior transferability of AdaptKT. Second, comparing Original and baseline variants, it is easy to see that fine-tuning is an effective way to improve the knowledge tracing model's performance on target domain with subject transferring task since the Original results like a random guess. Third, comparing the result in Table 3 and Table 4 of AdaptKT, we can easily see that transferring between two schools on the same subject is easier than transferring between two different subjects. This is because the former has more common properties between both domains than the latter.

5.5 Sensitivity Analysis

AdaptKT contains two regularization factors λ and γ. The well-selected values can help the framework perform better. Therefore, we conduct empirical experiments to evaluate their sensitivity by setting different values, i.e. 0.0, 0.25, 0.5, 0.75, 1.0 of them on $A \to B$, $B \to A$, $M \to P$ and $P \to M$ tasks. The results are shown in Figure 9. We find that an appropriate λ is of great significance for AdaptKT, such as $\lambda = 0.0$ means no source domain data are used, so the corresponding transferring results are not good, and the larger of the λ, the more source domain data will be selected. The best value of λ is around 0.5, where AdaptKT performs best on AUC. Meanwhile, it is obvious that γ is also important in AdaptKT because it balances the training loss and domain discrepancy. $\gamma = 0.0$ means ignoring the domain discrepancy, that is, the adaptation layer is abandoned without doing any transferring processes, and larger value means it has the more important effect.

5.6 Practical Applications

To investigate the practical application effect of AdaptKT, under transferring and non-transferring scenarios, we visualize and compare the knowledge states on two knowledge concepts trigonometric function and dynamics, with $A \to B$ and $M \to P$ tasks respectively. As shown in Figure 8, there are two heatmaps. The top one is the visualization of a student's knowledge states on knowledge concept trigonometric function under transferring and non-transferring scenarios, transferring direction is from school A to school B. The bottom one is the same. These heatmaps show many interesting results. First, comparing the non-transferring and transferring results in each heatmap, we can easily see that the transferred result is more stable and the change is not dramatic, especially at the end, student's knowledge state has no changes since he answers these questions correctly. Second, comparing the transferring results of $A \to B$ and $M \to P$, it is obvious that transferring between schools on the same subjects is easier than transferring between different subjects, since the result of the former is more accurate. In general, it proves that our three phases adaptable method is very useful and effective.

6 CONCLUSION

In this paper, we addressed the issues of existing KT methods and studied the domain adaptation for the knowledge tracing (DAKT) problem. We proposed an adaptable knowledge tracing (AdaptKT) method with three transferring phases. Extensive experimental results demonstrated that AdaptKT was an adaptable knowledge tracing method for solving the DAKT problem. In the future, there are some directions for further studies. First, the modeling process of slipping and guessing can be well designed in the future for more interpretable. Second, we will exploit the DAKT problem during different grades, and different subjects between different schools, etc. Third, we will build a more robust model to avoid the sensitivity of hyperparameters.

7 ACKNOWLEDGEMENT

This research was partially supported by grants from the National Natural Science Foundation of China (Grants No. 61922073, U20-A20229 and 62106244) and the Fundamental Research Funds for the Central Universities (Grant No. WK2150110021).

REFERENCES

[1] Ghodai Abdelrahman and Qing Wang. 2019. Knowledge Tracing with Sequential Key-Value Memory Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 175–184.

[2] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. 2014. Engaging with massive online courses. In *Proceedings of the 23rd international conference on World wide web*. ACM, 687–698.

[3] Yoshua Bengio. 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2, 1 (2009), 1–127.

[4] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alexander J. Smola. 2006. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics* 22 14 (2006), e49–57.

[5] Hugh Burns, Carol A Luckhardt, James W Parlett, and Carol L Redfield. 2014. *Intelligent tutoring systems: Evolutions in design.* Psychology Press.

[6] Hao Cen, Kenneth R. Koedinger, and Brian Junker. 2006. Learning Factors Analysis - A General Method for Cognitive Model Evaluation and Improvement. In *Intelligent Tutoring Systems*.

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

[8] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (2009), 855–868.

[9] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. 2009. Covariate shift by kernel mean matching. *Dataset shift in machine learning* 3, 4 (2009), 5.

[10] Zhenya Huang, Xin Lin, Hao Wang, Qi Liu, Enhong Chen, Jianhui Ma, Yu Su, and Wei Tong. 2021. DisenQNet: Disentangled Representation Learning for Educational Questions. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 696–704.

[11] Z. Huang, Q. Liu, Y. Chen, L. Wu, and G. Hu. 2020. Learning or Forgetting? A Dynamic Approach for Tracking the Knowledge Proficiency of Students. *ACM Transactions on Information Systems* 38, 2 (2020), 1–33.

[12] Z. Huang, X. Wei, and Y. Kai. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *Computer Science* (2015).

[13] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. 2018. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell* 172, 5 (2018), 1122–1131.

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, May 7-9, 2015, Conference Track Proceedings*.

[15] Van Der Maaten Laurens and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 2605 (2008), 2579–2605.

[16] Charles X. Ling, Jin Huang, and Harry Zhang. 2003. AUC: a Statistically Consistent and more Discriminating Measure than Accuracy. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, Georg Gottlob and Toby Walsh (Eds.). Morgan Kaufmann, 519–526. http://ijcai.org/Proceedings/03/Papers/077.pdf

[17] Liu, Qi, Runze, Chen, Enhong, Guandong, Yu, Zhigang, and Guoping. 2018. Fuzzy Cognitive Diagnosis for Modelling Examinee Performance. *ACM transactions on intelligent systems and technology* 9, 4 (2018), 296–296.

[18] 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.

[19] Yunfei Liu, Yang Yang, Xianyu Chen, Jian Shen, Haifeng Zhang, and Yong Yu. 2020. Improving Knowledge Tracing via Pre-training Question Embeddings. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 1577–1583.

[20] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*. JMLR.org, 97–105.

[21] Zhongqi Lu, Yin Zhu, Sinno Jialin Pan, Evan Wei Xiang, Yujing Wang, and Qiang Yang. 2014. Source Free Transfer Learning for Text Classification. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (Qu&#233;bec City, Qu&#233;bec, Canada) *(AAAI'14)*. AAAI Press, 122–128. http://dl.acm.org/citation.cfm?id=2893873.2893894

[22] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013).

[23] Koki Nagatani, Qian Zhang, Masahiro Sato, Yan-Ying Chen, Francine Chen, and Tomoko Ohkuma. 2019. Augmenting Knowledge Tracing by Considering Forgetting Behavior. In *The World Wide Web Conference*. ACM, 3101–3107.

[24] Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. 2019. Graph-based Knowledge Tracing: Modeling Student Proficiency Using Graph Neural Network. In *IEEE/WIC/ACM International Conference on Web Intelligence*.

[25] Genevieve B Orr and Klaus-Robert Müller. 2003. *Neural networks: tricks of the trade.* Springer.

[26] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

[27] Shalini Pandey and Jaideep Srivastava. 2020. RKT: Relation-Aware Self-Attention for Knowledge Tracing. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, 1205–1214.

[28] Philip I. Pavlik, Hao Cen, and Kenneth R. Koedinger. 2009. Performance Factors Analysis - A New Alternative to Knowledge Tracing. 200 (2009), 531–538. https://doi.org/10.3233/978-1-60750-028-5-531

[29] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2019. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

[30] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J. Guibas, and Jascha Sohl-Dickstein. 2015. Deep Knowledge Tracing. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015*. 505–513.

[31] Shuanghong Shen, Qi Liu, Enhong Chen, Zhenya Huang, Wei Huang, Yu Yin, Yu Su, and Shijin Wang. 2021. Learning Process-consistent Knowledge Tracing. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, Feida Zhu, Beng Chin Ooi, and Chunyan Miao (Eds.). ACM, 1452–1460.

[32] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*. Springer, 443–450.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014*. 3104–3112.

[34] Ben Tan, Yu Zhang, Sinno Jialin Pan, and Qiang Yang. 2017. Distant Domain Transfer Learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017*. AAAI Press, 2604–2610.

[35] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, June 7-12, 2015*. IEEE Computer Society, 3156–3164.

[36] Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yuying Chen, Yu Yin, Zai Huang, and Shijin Wang. 2020. Neural cognitive diagnosis for intelligent education systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 6153–6161.

[37] Zhiwei Wang, Xiaoqin Feng, Jiliang Tang, Gale Yan Huang, and Zitao Liu. 2019. Deep knowledge tracing with side information. In *International conference on artificial intelligence in education*. Springer, 303–308.

[38] J. Wu, Z. Huang, Q. Liu, D. Lian, H. Wang, E. Chen, H. Ma, and S. Wang. 2021. Federated Deep Knowledge Tracing. In *WSDM '21: The Fourteenth ACM International Conference on Web Search and Data Mining*.

[39] Haiqin Yang and Lap Pong Cheung. 2018. Implicit Heterogeneous Features Embedding in Deep Knowledge Tracing. *Cogn. Comput.* 10, 1 (2018), 3–14.

[40] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014*. 3320–3328.

[41] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic Key-Value Memory Networks for Knowledge Tracing. In *Proceedings of the 26th International Conference on World Wide Web, April 3-7, 2017*. ACM, 765–774.

[42] Kai Zhang, Qi Liu, Hao Qian, Biao Xiang, Qing Cui, Jun Zhou, and Enhong Chen. 2021. EATN: An Efficient Adaptive Transfer Network for Aspect-level Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[43] Kai Zhang, Hefu Zhang, Qi Liu, Hongke Zhao, Hengshu Zhu, and Enhong Chen. 2019. Interactive attention transfer network for cross-domain sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5773–5780.

[44] Moyu Zhang, Xinning Zhu, Chunhong Zhang, Yang Ji, Feng Pan, and Changchuan Yin. 2021. Multi-Factors Aware Dual-Attentional Knowledge Tracing. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2588–2597.