Adaptive and Personalized Exercise Generation for Online Language Learning

Peng Cui **Mrinmaya Sachan**
Department of Computer Science, ETH Zürich
{peng.cui, mrinmaya.sachan}@inf.ethz.ch

Abstract

Adaptive learning aims to provide customized educational activities (e.g., exercises) to address individual learning needs. However, manual construction and delivery of such activities is a laborious process. Thus, in this paper, we study a novel task of *adaptive* and *personalized* exercise generation for online language learning. To this end, we combine a knowledge tracing model that estimates each student's evolving knowledge states from their learning history and a controlled text generation model that generates exercise sentences based on the student's current estimated knowledge state and instructor requirements of desired properties (e.g., domain knowledge and difficulty). We train and evaluate our model on real-world learner interaction data from Duolingo and demonstrate that LMs guided by student states can generate superior exercises. Then, we discuss the potential use of our model in educational applications using various simulations. These simulations show that our model can adapt to students' individual abilities and can facilitate their learning efficiency by personalizing learning sequences.[1]

1 Introduction

Adaptive learning technologies which continuously monitor student progress to dynamically adjust the level or type of learning materials based on the individual's abilities are quite popular (Becker et al., 2018). Empirical studies have shown various benefits of adaptive learning, such as improved student learning outcomes (Bailey et al., 2018; Holthaus et al., 2019), lower dropout rates (Daines et al., 2016), and increased instructor satisfaction (Yarnall et al., 2016). Despite their effectiveness, designing adaptive systems is challenging as it usually involves planning a series of exercises that is personalized and adaptive to each student, which requires diverse exercise planning as well as an understanding of the student learning process.

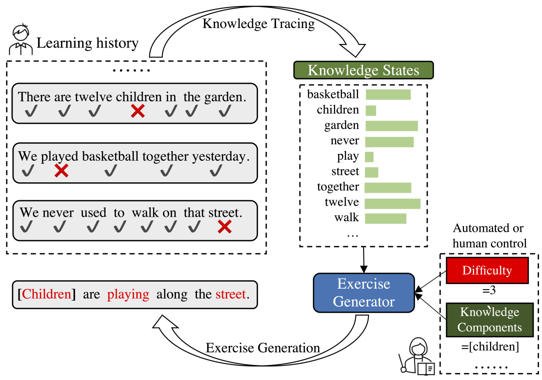

Figure 1: We first assess student knowledge states from their learning history and then generate exercises based on estimated states and instructor control of desired properties including domain knowledge (vocabulary) and difficulty levels (expected error numbers).

On the other hand, powered by advances in neural NLP, works have been done for automatically generating text-based exercises or questions for educational purposes in second language learning (Heck and Meurers, 2022; Perez and Cuadros, 2017), mathematics (Polozov et al., 2015; Zhou and Huang, 2019; Wang et al., 2021), and computer science (Susanti et al., 2017). Nevertheless, how to apply these approaches in adaptive systems remains an open question. First, existing methods largely rely on pre-defined question templates or specified information sources (e.g., a passage), thereby resulting in limited knowledge coverage and low question difficulty control, and as a consequence, do not meet each student's individual and nuanced learning needs. Besides, they are usually designed to generate standalone exercises, whereas adaptive learning systems usually require a continuous supply of exercises. Another related line of research studies exercise recommendation to customize learning content based on individual ca-

[1]Our implementation is available at https://github.com/nlpcui/AdaptiveQG.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 10184–10198
July 9-14, 2023

pabilities and goals (Wu et al., 2020; Huang et al., 2022). However, these systems are limited by the diversity of the exercise pool.

To address the above limitations, we study the task of exercise generation in the context of adaptive learning, where we hypothesize that a student's *dynamic knowledge state* holds the key to generating *adaptive* and *personalized* exercises. Specifically, we ground our study in the domain of language learning to create exercise sentences for translation, of which Figure 1 illustrates the overall process. We start with an assumption about the dynamics between exercise difficulty, vocabulary, and a student's knowledge state (§ 3). Then, we propose an approach (§ 4) that marries knowledge tracing (KT; Corbett and Anderson (1994)), a technique for estimating students' mastery states of knowledge components from their learning history, with a controlled text generation model that generates the next exercise based on instructor requirements, such as specified *domain knowledge* and *target difficulty*. We further explore various strategies to adapt the generation of exercises based on students' changing knowledge states. In doing this, our model not only supports personalized generation where the instructor (or the system) can express some desired properties of the generated exercises but is also adaptive to each student's learning progress.

We conduct extensive experiments on real-world student learning data from Duolingo[2], a popular online language learning platform that offers structured and individualized learning content. Our results (§ 5) show that pre-trained LMs can help KT assess student language knowledge while student states estimated by KT can guide LMs to generate adaptive and personalized exercises. We further discuss the potential use of our model in educational applications with simulations. The simulations show that our model can dynamically adjust exercise difficulty to match individual learning progress and facilitate their learning efficiency by customizing exercise sequences.

2 Related Work

Adaptive Learning technologies that dynamically monitor student progress and adjust the course content based on an individual's abilities have demonstrated various benefits in education (Becker et al., 2018). Such systems usually consist of three core components: (1) a *domain model* which refers to the content and structure of the topic to be taught, (2) a *learner model* which repeatedly measures and updates learner characteristics, and (3) an *adaption model* which combines information from the domain and learner model to offer adaptive instructions (Vagale and Niedrite, 2012; Imhof et al., 2020). In this study, we build the learner model based on the KT technique and combine the domain and adaption model into an LM which generates learning content adaptively based on user features captured by the learner model.

Knowledge Tracing (Corbett and Anderson, 1994) is the technique to estimate students' knowledge mastery $\mathbf{s}$ from their practiced exercises ($\mathbf{e}$) and responses ($\mathbf{r}$):

$$\mathbf{s_{t+1}} = f_{KT}((\mathrm{e}_1, \mathrm{r}_1), (\mathrm{e}_2, \mathrm{r}_2), ..., (\mathrm{e}_\mathrm{t}, \mathrm{r}_\mathrm{t})). \quad (1)$$

Early KT approaches model f_{KT} as variants of logistic regression, such as Item Response Theory (IRT) and Additive Factor Model (AFM) (Cen et al., 2008), or probabilistic models such as Bayesian Knowledge Tracing (Corbett and Anderson, 1994) and its variants (Yudelson et al., 2013; Käser et al., 2017). These approaches heavily rely on their assumptions of the learning process which are often incomplete. In recent years, neural networks have become the dominant method in this area. Piech et al. (2015) proposed the first Deep Knowledge Tracing model based on Recurrent Neural Networks. After that, various architectures have been applied to model different characteristics of learning, such as self-attention (Pandey and Karypis, 2019; Shin et al., 2021), memory networks (Abdelrahman and Wang, 2019), and graph neural networks (Tong et al., 2020).

Exercise Generation. Previous exercise generation approaches for language learning primarily retrieve and manipulate text to create fixed types of exercises, such as gap fill and multiple-choice exercises (Agarwal and Mannem, 2011; Perez and Cuadros, 2017; Heck and Meurers, 2022), which are limited by the richness of the corpus. Besides them, some Question Generation (QG) approaches have been proposed for educational purposes (Zhao et al., 2022; Wang et al., 2021). While some of them allow for user control of certain question properties, they do not consider learners' individual and dynamic learning needs and progress. Thus, they cannot achieve the goal of adaptive learning. Recently, Srivastava and Goodman (2021) proposed an adaptive question generation model that connects question difficulty with student knowledge.

[2] https://www.duolingo.com/

However, it neither models students' fine-grained knowledge states nor provides control over domain knowledge. Consequently, it is insufficient for practical use.

Controlled Text Generation (CTG) methods aim to steer text generation toward certain attributes. Existing CTG approaches can be broadly classified into three types: directly training a class-conditional language model (CCLM) (Keskar et al., 2019; Ziegler et al., 2019; Ficler and Goldberg, 2017), guiding a model via an attribute discriminator (Dathathri et al., 2020; Liu et al., 2020), or manipulating decoder's logits (also referred to as weighted decoding) (Holtzman et al., 2018; Yang and Klein, 2021). This study explores difficulty and lexical control in generating language learning exercises. Additionally, we seek to adapt the model's controllability to different users by building the dependency between control signals and individual states.

3 Problem Formalization

Let $\mathcal{H}_{\leq n} = \{(e_1, r_1), ..., (e_n, r_n)\}$ be a student's **learning history** consisting of n exercises and responses. Here, $e_i = \{w_{i,1}, ..., w_{i,|e_i|}\}$ is an **exercise sentence** for translation and $r_i \in \{0, 1\}^{|e_i|}$ is the **correctness label** for each word in e_i. We generate the next exercise e_{n+1} based on:

- C_{n+1}: **knowledge components** that should be involved in e_{n+1}. In language learning, we consider a word as a knowledge component, and therefore $C_{n+1} = \{c_1, ..., c_{|C_{n+1}|} | c_* \in \mathcal{V}\}$ is a subset of vocabulary $\mathcal{V}$ that should be included in the output. In general, the knowledge components can be user or system defined based on the current learning material.

- $\mathbf{s_{n+1}}$: a student's **knowledge state** for the knowledge components (the vocabulary) after n interactions. $\mathbf{s_{n+1}}$ can be formalized as a $|\mathcal{V}|$-dimensional vector with each entry between 0 and 1 indicating the mastery probability of that word.

- d_{n+1}: the **expected difficulty** of e_{n+1}. We use individual performance to estimate problem difficulty. For a particular student, the difficulty of an exercise is defined as the expected number of word errors the student would make in translating it.

Given the above setting, we formalize our task as:

$$e_{n+1} = \arg\max_{e} P(e|\mathbf{s_{n+1}}, d_{n+1}, C_{n+1}), \quad (2)$$

where e_{n+1} satisfies the following constraints:

$$\forall c \in C_{n+1} : \exists i, e_{n+1_{i:i+|c|}} = c, \quad (3)$$

$$d_{n+1} = \sum_{w \in e_{n+1}} (1 - \mathbf{s_{n+1}}[w]), \quad (4)$$

corresponding to *word constraint* and *difficulty constraint*, respectively. Here, $\mathbf{s_{n+1}}[w]$ represents the correct probability of translating word w; therefore, the sum of $\{1 - \mathbf{s}[w]|, w \in e\}$ is the expected number of errors in translating e, which can be seen as a measure of the difficulty of e.

Our task is distinct from previous CTG works in two aspects: 1) our control is *dynamic*; student states acting as control are also learnable; 2) there is a strong dependency among control signals (Eqs. 3 and 4), which is non-trivial to learn. Note that in this work, we measure difficulty via student performance and only consider vocabulary knowledge in defining $\mathbf{s}$ for simplicity. Other definitions of sentence difficulty (e.g., definitions that incorporate other types of linguistic knowledge such as syntax) can be explored in future work.

4 Methodology

Our model is illustrated in Figure 2. We first employ a knowledge tracer $\mathcal{T}$ (§ 4.1) to estimate a student's time-varying knowledge states. Then, we build an LM-based exercise generator $\mathcal{G}$ (§ 4.2) to create exercises based on estimated states and specified difficulty and knowledge components (words). We jointly optimize the two modules with an inconsistency loss (§ 4.3) at training and apply a constrained decoding strategy (§ 4.4) at inference. Finally, we discuss how our model can accommodate personalized learning recommendation algorithms on the fly (§ 4.5).

4.1 Knowledge Tracing

The goal of our knowledge tracing model $\mathcal{T}$ is to estimate a student's latest knowledge state $\mathbf{s_{n+1}}$ given previous interactions $\mathcal{H}_{\leq n}$. We adopt the deep knowledge tracing (DKT) model proposed by Piech et al. (2015). We concatenate past exercises as a word sequence $\mathbf{e}_{1:n} = \{w_{1,1}, ..., w_{n,|e_n|}\}$ and past responses as a label sequence $\mathbf{r}_{1:n} = \{r_{1,1}, ..., r_{n,|e_n|}\}$, where $w_{i,j}$ and $r_{i,j}$ represent the jth word or label of the ith exercise. Then we

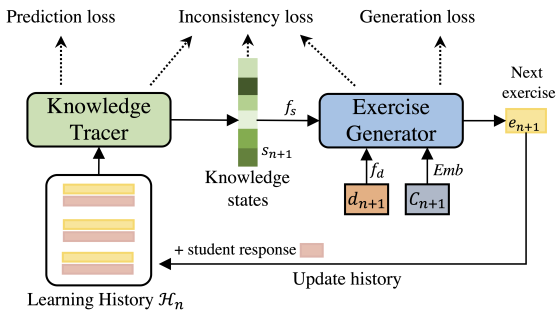

Figure 2: The framework of our proposed model. We estimate a student's latest knowledge state $\mathbf{s_{n+1}}$ from the learning history $\mathcal{H}_n$, and then combine it with user-specified difficulty d_{n+1} and knowledge components C_{n+1} to generate the next exercise e_{n+1}. The two modules are jointly trained with an inconsistency loss to penalize their disagreement.

convert the two sequences into word embeddings $\vec{\mathbf{e}}_{1:n}$ and label embeddings $\vec{\mathbf{r}}_{1:n}$ and send them to an LSTM encoder to predict the next state $\mathbf{s_{n+1}}$:

$$\mathbf{h_n} = \text{LSTM}(\vec{\mathbf{e}}_n + \vec{\mathbf{r}}_n; \mathbf{h_{n-1}}), \tag{5}$$

$$\mathbf{s_{n+1}} = sigmoid(\mathrm{W_s} * \mathbf{h_n} + \mathrm{b_s}). \tag{6}$$

The model is trained to predict the binary word labels of the next exercise using the estimated knowledge state. The cross-entropy loss for a single student's history of N interactions is computed as:

$$\mathcal{L}_{ce} = \sum_{i=1}^{|N|} \sum_{j=1}^{|e_i|} \text{CE}(r_{i,j}, \mathbf{s}_i[w_{i,j}]). \tag{7}$$

We adopt the regularization strategy proposed by Yeung and Yeung (2018) to stabilize training:

$$\mathcal{L}_{r_{\{1,2\}}} = \sum_{n=2}^{N} \sum_{i=1}^{|\mathcal{V}|} |\mathbf{s_n}^{(i)} - \mathbf{s_{-1}}^{(i)}|^{\{1,2\}}, \tag{8}$$

where $\mathcal{L}_{r_1}$ ensures that only the states of relevant knowledge components are updated, and $\mathcal{L}_{r_2}$ penalizes the vibration. The final objective of $\mathcal{T}$ is $\mathcal{L}_{\mathcal{T}} = \mathcal{L}_{ce} + \lambda_1 * \mathcal{L}_{r_1} + \lambda_2 * \mathcal{L}_{r_2}$ with λ for balance.

4.2 Controllable Exercise Generator

Our exercise generator $\mathcal{G}$ is fine-tuned from a pre-trained LM. Specifically, we generate an exercise e based on a student's current knowledge state $\mathbf{s}$, target words C, and expected difficulty d (we drop the interaction index to reduce clutter). We parameterize the inputs as follows:

$$\mathbf{x} = [f_s(\mathbf{s}); f_d(d); Emb(c_1, ..., c_{|C|})], \tag{9}$$

where knowledge state $\mathbf{s}$ and scalar difficulty d are projected to control vectors via two feedforward layers f_s and f_d, and C are mapped to word embeddings. The training objective for generating a single exercise is defined as:

$$\mathcal{L}_{\mathcal{G}} = -\sum_{t}^{|e|} logP(w_t|w_1, ..., w_{t-1}, \mathbf{x}). \tag{10}$$

During training, we sample a proportion of words from reference exercises as C and calculate difficulty d from ground-truth correctness labels, whereas states $\mathbf{s}$ are estimated by $\mathcal{T}$. At inference, d and C can be determined by instructors or the system, allowing automated and human intervention.

4.3 Joint Learning with Inconsistency Loss

We jointly optimize the knowledge tracer $\mathcal{T}$ and exercise generator $\mathcal{G}$ with an *inconsistency loss* inspired by Cui and Hu (2021), enabling the two modules to learn from each other. Concretely, after generating an exercise e, we calculate its difficulty using input state $\mathbf{s}$ via Eq. 4, which should be as close to the input difficulty d as possible:

$$\mathcal{L}_{inc} = |d - \sum_{w \in e}(1 - \mathbf{s}[w])|. \tag{11}$$

Since the second term is non-differentiable due to the $argmax$ operation involved in producing e, we replace it with "soft" tokens:

$$\mathcal{L}_{inc} = |d - \sum_{t}^{|e|}(1 - \mathbf{p}_t \odot \mathbf{s})|, \tag{12}$$

where $\mathbf{p}_t = softmax(\mathbf{o}_t/\tau)$ is the t^{th} distribution normalized from its logits $\mathbf{o}_t \in \mathbb{R}^{|\mathcal{V}|}$ with a temperature parameter τ, and $\odot$ represents dot product.

For the generator $\mathcal{G}$, this loss constrains the generation toward the target difficulty. For $\mathcal{T}$, the LM distributions p_θ provide similarity information between vocabulary words. This is analogous to the relationship of knowledge components, which has been shown helpful in knowledge tracing (Tong et al., 2020). The final objective of our model is $\mathcal{L} = \mathcal{L}_{\mathcal{T}} + \gamma_1 \mathcal{L}_{\mathcal{G}} + \gamma_2 \mathcal{L}_{inc}$.

4.4 Lexical Difficulty Constrained Decoding

We propose a beam search-based decoding algorithm to enforce the constraints introduced in § 3. At each step, we update the beam according to:

$$Y_t = \underset{\mathbf{y}_{<t} \in Y_{t-1}, y_t \in \mathcal{V}}{\text{argtopk}} \; logP(\mathbf{y}_{\leq t}|\mathbf{x}) + \sum_{F_i \in \mathcal{F}} \alpha_i F_i(\mathbf{y}_{\leq t}), \tag{13}$$

where Y_t is the set of decoded hypotheses in step t and k is the beam size. The first term is the standard objective of beam search and the second term is a weighted combination of additional scoring functions in terms of the satisfaction of different constraints. We formulate our constraints $\mathcal{F}$ in Eqs. 3 and 4 as:

$$F_c(\mathbf{y}) = \sum_{c \in \mathcal{C}} I(c, \mathbf{y}), \ \ and \ \ F_d(\mathbf{y}) = -|d - h(\mathbf{y})|,$$

corresponding to the satisfaction of word constraint and difficulty constraint, respectively. $I(c, y)$ is a Boolean predicate indicating whether word c is included in sequence $\mathbf{y}$ and $h(\mathbf{y})$ calculates its difficulty via Eq. 4.

Succinctly, the decoding algorithm works in three steps. First, we **expand** the current k hypotheses to $k \times |\mathcal{V}|$ candidates. Then, we **prune** the search space by dropping candidates that are not in the top-k_F list of any scoring functions F. Finally, we **rescore** the pruned candidates based on the full objective (Eq. 13) and select the k-best ones to update the beam.

However, we found that greedily applying F_d in the rescoring step would bias the decoder toward sequences with difficult words in the earlier steps. Drawing inspiration from Lu et al. (2022), we use lookahead heuristics that incorporate future estimates into the decoding process. Concretely, to score a subsequence $\mathbf{y}_{<t}$, we first greedily decode the next $l+1$ steps "soft" tokens (i.e., distributions): $\tilde{\mathbf{y}}_{t:t+l}$=$[\mathbf{p}_t, ..., \mathbf{p}_{t+l}]$. Then, we combine the constraint satisfaction of decoded $\mathbf{y}_{<t}$ and the estimated future $\tilde{\mathbf{y}}_{t:t+l}$:

$$\tilde{F}_c(\mathbf{y}_{<t}) = \sum_{c \in \mathcal{C}} \max(I(c, \mathbf{y}_{<t}), \max_{j \in [t, t+l]} P(y_j = c)),$$

$$\tilde{F}_d(\mathbf{y}_{<\mathbf{t}}) = -|d - h(\mathbf{y}_{<\mathbf{t}}) - \sum_{j=t}^{t+l} 1 - \mathbf{p}_j \odot \mathbf{s}|.$$

The procedure of our decoding algorithm is in Appendix A.

4.5 Plug-and-Play Personalized Generation

Our model can be flexibly plugged into an existing personalized learning recommendation algorithm to automatically generate novel and customized exercises. We showcase this functionality using the ExpectiMax curriculum planning strategy derived from DKT. Given a student's current state $\mathbf{s_n}$, we can calculate the expected knowledge state after practicing a new exercise e using our KT model $\mathcal{T}$:

$$\tilde{\mathbf{s}}_{n+1} = \sum_{r \in \{0,1\}^{|e|}} P(r) * \mathcal{T}(\mathbf{s}_n, (e, r)), \quad (14)$$

where $\mathcal{T}(\cdot)$ computes the updated knowledge state given a new interaction (e, r). The probability of label sequence r is computed from $\mathbf{s}_n$ assuming conditional independence $P(r) = \prod_{i=1}^{|e|} P(r_i)$, where $P(r_i) = \mathbf{s}_n[e_i]$. ExpectiMax scores e based on how well it can improve a student's average knowledge state, i.e., $F_k(e) = \bar{\tilde{\mathbf{s}}}_{n+1} - \bar{\mathbf{s}}_n$, where $\bar{\mathbf{s}}$ denotes mean of the vector. We incorporate F_k into the decoding objective (Eq. 13) and call it ExpectiMax-Gen.

In principle, our model can accommodate different recommendation algorithms with different ranking functions F_k. The key benefit is that our model can *generate* novel exercises, while retrieval-based systems can only *select* exercises from an existing pool.

Model	Word-level		Exercise-level	
	Seen	Unseen	Seen	Unseen
Ensemble	73.41	70.58	65.55	64.93
Standard DKT	80.46	75.54	72.32	71.54
$\text{DKT}_{\text{LM},\tau=0.5}$	80.47	75.51	72.39	71.47
$\text{DKT}_{\text{LM},\tau=1.0}$	80.49	75.54	72.38	71.49
$\text{DKT}_{\text{LM},\tau=2.0}$	**80.55**	**75.69**	**72.41**	**71.74**
$\text{DKT}_{\text{LM},\tau=3.0}$	80.54	75.48	72.33	71.52
$\text{DKT}_{\text{LM},\tau=5.0}$	80.31	75.46	72.28	71.50

Table 1: AUC ($\times$ 100) performance of knowledge tracing models on seen and unseen text examples. Exercise-level results are obtained by averaging word-level predictions.

5 Experimental Results and Analysis

We experiment on the English track of Duolingo Second Language Acquisition Modeling (SLAM) dataset (Settles et al., 2018), which contains about 1 million interactions of 2.6k learners over the first 30 days of learning a second language. For each student, we use the first 80% of interactions for training, and the subsequent and the last 10% for validation and testing, respectively. Details of the dataset and experimental setup are in Appendix B.

We first evaluate the ability of the KT model to estimate student knowledge states in § 5.1. Then, we analyze the effectiveness of the exercise generator in § 5.2. Lastly, we showcase the superiority of our model in two educational scenarios with simulation experiments in § 5.3.

Models	BLEU ↑		METEOR ↑		KC-Coverage (%) ↑		D-MAE ↓		Invalid (%) ↓
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen	
$\text{EG}_{\mathcal{H}}$	9.23	<0.01	18.79	6.05	14.26	2.49	0.396	1.500	**0.071**
$\text{AQG}_{\mathcal{H}+d}$	10.28	<0.01	20.15	7.16	15.84	2.95	0.463	0.985	1.674
EG_{C}	18.41	5.21	45.36	36.14	**99.77**	90.63	0.367	0.837	0.301
EG_{C+d}	11.84	15.94	40.89	42.10	96.23	91.62	0.564	0.679	0.385
$\text{APEG}_{\mathbf{s}+C+d}$	**22.47**	**34.60**	**56.15**	**44.01**	99.61	**95.71**	**0.246**	**0.604**	0.283
- joint learning	22.01	33.15	55.80	42.85	99.63	94.08	0.251	0.619	0.281
- constrained decoding	21.58	32.06	55.43	40.49	99.59	94.77	0.263	0.681	0.277
Upper bound	53.65	41.24	74.97	52.10	99.75	95.96	0.060	0.302	0.233

Table 2: Results of exercise generation. APEG is our proposed model, and AQG is an adaptively difficulty-controlled question generation model proposed by Srivastava and Goodman (2021). The subscripts represent whether historical interactions ($\mathcal{H}$), target words (C), difficulty (d), and student state ($\mathbf{s}$) are used to generate exercises.

5.1 Knowledge Tracing Evaluation

We use the standard **AUC (ROC)** as the metric of knowledge tracing in accordance with Settles et al. (2018). We denote our DKT model jointly trained with the LM-based exercise generator as DKT_{LM} and compare it with the following baselines: 1) Ensemble (Osika et al., 2018) which is one of the winning methods of the SLAM challenge that combines a RNN and a GBDT classifier. We reimplement this model to use texts only as input and remove other side features, such as response time. We do this because we are interested in its performance in a *general* setting where we do not assume the availability of diverse side information; 2) the standard DKT (Piech et al., 2015) which is trained only with the KT loss $\mathcal{L}_{\mathcal{T}}$. We use it to verify whether jointly learning with an LM can help predict student language knowledge.

We present the results in Table 1, where we can see that DKT outperforms the Ensemble model when only text features are used, and our best model $\text{DKT}_{\text{LM},\tau=2}$ outperforms DKT on all metrics. We hypothesize the performance gain comes from the word similarity information entailed in the output distributions p_θ of the LM. This can be regarded as the relationship between knowledge components, which is demonstrated effective in knowledge tracing (Tong et al., 2020). To verify this, we tune the temperature τ which controls the sparsity of output distributions: $\tau \to 0$ produces a sparse distribution that is too assertive and provides little relationship information, while $\tau \to \infty$ produces a uniform distribution where all words are evenly related. The results in the second section of Table 1 suggest that a medium τ improves the performance, while a small (τ=1) or large (τ=5) is harmful, particularly for predicting unseen data. The broader message from this observation is that the knowledge encoded in pre-trained LMs has the potential to improve knowledge tracing in the domain of language learning. We also conduct an analysis of the influence of regularization terms Eq. 8, detailed in Appendix C.

5.2 Exercise Generation Evaluation

The main results of exercise generation are presented in Table 2, which are split according to whether the exercises are seen in the training set. Evaluation metrics include reference-based **BLEU** (Papineni et al., 2002) and **METEOR** (Banerjee and Lavie, 2005), **KC-Coverage** which is the percentage of target knowledge components (words) that appear in the outputs, **D-MAE** which is the mean absolute error between the input difficulty and output difficulty, **Invalid** which is the percentage of exercises that have grammar errors detected using an automatic tool[3]. Since we generate exercises for language learning, we expect a valid exercise to be grammatically correct. We analyze the performance from the following aspects.

Lexical Controllability. We first examine the lexical controllability of our model, which is crucial for generating personalized exercises for language learning. We compare our model with two baselines:1) $\text{EG}_{\mathcal{H}}$ which generates the next exercise based on the student's historical interactions; and 2) $\text{AQG}_{\mathcal{H}+d}$[4] which generates the next exercise based on historical interactions and a target difficulty. The two baselines perform poorly on BLEU, METEOR, and KC-Coverage metrics, particularly

[3]https://github.com/jxmorris12/language_tool_python.

[4]We obtain its results using the code released by the authors. Note that AQG is built on a different definition of difficulty. Thus, the D-MAE result might bias toward our model. We report this metric for reference only.

	BLEU ↑	Coverage (%) ↑	D-MAE ↓
w/o lookahead	20.46	99.18	0.263
w/ lookahead	**21.20**	**99.30**	**0.257**

Table 3: Comparison of generation performance with and without lookahead on the validation set.

for unseen data. This indicates that they cannot predict the accurate content of the next exercise based on historical data or difficulty information, possibly because there is no strong connection within a sequence of exercises or such connection cannot be captured by an LM. We note that $\text{EG}_{\mathcal{H}}$ performs well on the validness metric. However, upon inspecting its results, we found the model almost only copies exercises from history, with less than 0.02% novel generations. The same issue is observed in $\text{AQG}_{\mathcal{H}+d}$ where more than 90% exercises are repetitive. We follow Srivastava and Goodman (2021) to improve its novelty using a repetition penalty during the generation, but this results in far more invalid exercises (1.7%). In comparison, our model achieves a better balance between generalization ability and fluency.

Effect of Student Modeling. To investigate whether student modeling helps exercise generation, we build two baselines without student knowledge states: 1) EG_C which conditions generation on target KCs (words) only, and 2) EG_{C+d} on both target words and difficulty. The former variant can be considered a keyword-to-text generation model, while the latter imposes additional difficulty control. Our full model $\text{APEG}_{\mathbf{s}+C+d}$ significantly outperforms both of them, which proves our aforementioned hypothesis that a student's dynamic knowledge states must be considered in generating adaptive and personalized exercises. An interesting observation is that incorporating difficulty control improves the performance on unseen data, indicating the model to some degree learns generalizable difficulty information. Nevertheless, our further analysis shows the model is not adaptive to students of different abilities, which will be discussed in § 5.3.

Ablation Study. The key challenge of our task is to learn the dependency between student knowledge, vocabulary, and exercise difficulty (Eqs. 3 and 4). To understand which parts of our model contribute to this goal, we build two ablated variants by removing the joint learning strategy (§ 4.3) and the constrained decoding algorithm (§ 4.4), respectively. As shown in the second section of Table 2, the search-based method is slightly better than the learning-based method, while combining them leads to the best performance.

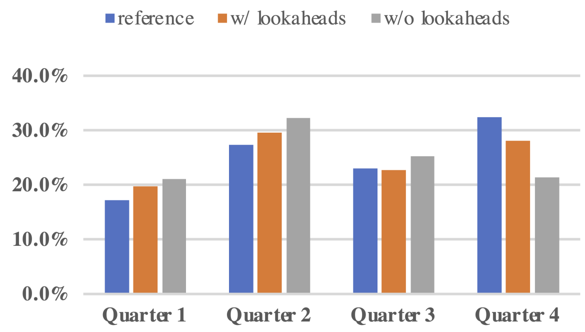

Figure 3: Distributions of accumulated word difficulty in four equally sized segments of 2000 sampled exercise sentences.

We further explore the effect of the lookahead strategy on difficulty constraints. Table 3 presents the ablation results on the validation set, where we can see lookahead strategy improves both generation quality and controllability. To understand how it works, we measure the distribution of difficulty in different regions of exercise sentences. Such distribution is computed as the accumulated word difficulty in four equally sized segments of 2000 sampled sentences. As shown in Figure 3, the difficult words of reference exercises are largely concentrated in the 2^{nd} and 4^{th} quarter. Our decoding algorithm with lookahead produces a similar result, while removing lookahead would bias the distribution toward 2^{nd} and 3^{rd} quarter. This confirms our assumption that naively applying F_d would greedily select difficult words in the early steps, which is not the distribution of reference exercises. Our decoding algorithm avoids this issue by estimating the future and therefore achieves better results.

Upper Bound Analysis. When we train our model, we use ground-truth difficulty d and target words C obtained from references; however, the student states $\mathbf{s}$ are estimated from the KT model. We conduct an upper bound analysis to understand the influence of the accuracy of $\mathbf{s}$ on the generation performance. Since a student's actual mastery of every vocabulary word is not available, we choose to replace the ground-truth difficulty levels d with those estimated from $\mathbf{s}$. As shown in the last section of Table 2, all metrics are considerably boosted when the inconsistency between states $\mathbf{s}$ and difficulty d is eliminated. This again proves the effect

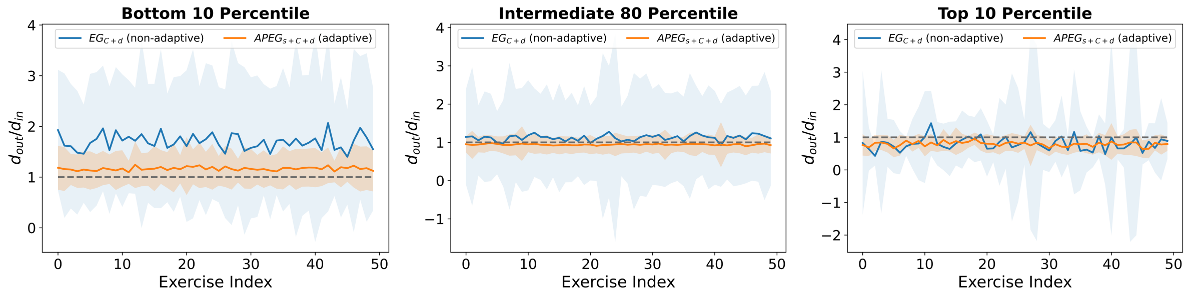

Figure 4: Generating 50 additional exercises of specified difficulty levels for different student groups using APEG$_{s+C+d}$ (adaptive) and non-adaptive EG$_{C+d}$ models. The Y-axis is the ratio of output difficulty d_{out} to input difficulty d_{in}; the closer to 1 (dotted line) the better. Solid lines are averaged results of group students at each step, and shadows represent standard deviations.

$\mathbf{d_{in}}$	**Target words**	**Generated exercises**	$\mathbf{d_{out}}$
		Avg. knowledge state $\bar{s}$ = 0.32	
1.0	{men}	Fifteen men.	1.25
2.0	{study}	I study English.	2.18
3.0	{airport}	Where is the airport?	2.73
		Avg. knowledge state $\bar{s}$ = 0.65	
1.0	{profile}	He has a famous profile.	0.94
2.0	{white, bitter}	The white mushroom is bitter.	1.75
3.0	{hit, nail}	She hit the nail on the head.	2.89

Table 4: Examples of exercises based on different controls. d_{in} is the input difficulty while d_{out} is the output difficulty estimated by our knowledge tracing model. The degree of highlight represents a student's mastery of vocabulary words (the darker the harder).

of incorporating student states and explains how such information comes to play: the knowledge states explicitly convey the dynamics between control signals d, $\mathcal{C}$, and target exercises e, which is non-trivial to learn by the model itself.

Case Study. We provide a few cases in Table 4. We can see our model can dynamically adjust the exercise content according to specified words, target difficulty, as well as students' different mastery states of the vocabulary. The exercises generated for advanced students (avg. state = 0.65) are generally more difficult than for poor students (avg. state = 0.32) under the same input difficulty.

5.3 Educational Applications

In this subsection, we showcase the potential applications of our model in two educational scenarios with simulation experiments.

5.3.1 Adaptive Difficulty Calibration

A crucial requirement for adaptive learning systems is to dynamically adjust the difficulty of learning items to match each student's learning progress (Becker et al., 2018). However, previous difficulty-controlled question generation approaches are mainly based on inherent problem difficulty, independent of individual abilities (Susanti et al., 2017; Kumar et al., 2019). Ideally, our model can achieve this goal by learning the dependency between difficulty and student knowledge states. To verify this, we generate 50 additional exercises of specified difficulties for each student after their existing interactions. At each step, we construct input by sampling a target word from the vocabulary and a difficulty level from a uniform distribution $[1, 3]$. We compare our full model APEG$_{s+C+d}$ with its variant EG$_{C+d}$ which achieves the best difficulty controllability for unseen data. This baseline can be considered a vanilla non-adaptive difficulty-controlled exercise generation model.

In this simulation, we are interested in whether the difficulty controllability of our model can adapt to students of various knowledge levels. To this end, we rank students based on their average knowledge states $\bar{s}$ and split the result accordingly. As shown in Figure 4, the difficulty controllability of the baseline is not reliable across different groups. In particular, it tends to generate harder (up to $2 \times d_{in}$) exercises for the bottom 10 percentile students but easier (up to $\frac{1}{2} \times d_{in}$) ones for the top 10 percentile students, although it performs well for the intermediate 80 percentile students. In comparison, our adaptive model is also slightly biased toward the intermediate group but much more consistent than the baseline, with less than 20% fluctuations on average. Besides, we can see from the shadows that the baseline experiences huge variances at each step, indicating it is not adaptive to different knowledge states, even though the students within a group are at a similar level.

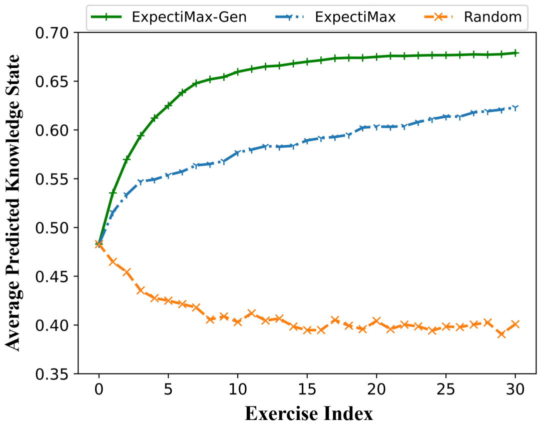

Figure 5: Simulation results over 30 exercises. The X-axis is the number of exercises, and the Y-axis is students' average predicted knowledge state $\bar{\bar{s}}$ indicating a student's overall mastery of the vocabulary.

5.3.2 Improving Learning Efficiency

We now examine whether our model can be used to improve student learning efficiency by personalizing exercise sequences. To this end, we customize 30 continuous exercises for 50 sampled students using our proposed EXPECTIMAX-GEN (§ 4.5) and the original EXPECTIMAX. Both of them aim to maximize the expected knowledge state of the next step $\bar{\bar{s}}_{n+1}$. For the former, at each step, we first find the best single word that can maximize $\bar{\bar{s}}_{n+1}$ and then generate the next exercise based on the selected word and a fixed difficulty of 1. For the latter, we directly select the best exercise from the pool. We update students' knowledge states after each practice and repeat this process until we collect 30 exercises. We compare the change in $\bar{\bar{s}}$ to measure which strategy is more efficient in improving students' knowledge.

The simulation results are shown in Figure 5. We also include a randomly selected exercise sequence as a lower bound, which turns out to harm student learning most of the time. The decrease in knowledge state is possibly caused by overly difficult exercises which would lead to wrong answers and reduce the predicted probability. Under the same practice opportunities, exercises generated by EXPECTIMAX-GEN lead to faster knowledge growth than those selected by EXPECTIMAX. Upon further inspection, we found about 70% of them are unseen in the corpus. This explains the efficiency of EXPECTIMAX-GEN as it can create novel exercises targeting individual needs on the fly while EXPECTIMAX is limited by the pool.

5.3.3 Qualitative Discussions on Simulation

Our simulations are based on the DKT model. We note that some previous studies have observed inconsistencies between DKT behaviors and the human learning process (Shen et al., 2021). Thus, we adopt a simple regularization approach (Eqs. 5 and 6) to alleviate such inconsistencies (Yeung and Yeung, 2018), which we found can reduce the variance of simulation results and improve KT performance (Appendix C).

A popular argument regarding the relationship between the difficulty of learning content and student outcomes is that the level of difficulty should be set just above the learner's current knowledge, i.e., $d \approx 0.5$ (Settles and Meeder, 2016; Gallego-Durán et al., 2018). During the simulations, we found EXPECTIMAX does not follow this heuristic but tends to generate relatively easy exercises ($d < 0.3$ mostly) repeatedly using certain words, consistent with the finding in Tschiatschek et al. (2022). One possible reason is that easier exercises are more likely to produce correct answers, which in turn increases the averaged predicted probability of DKT (i.e., estimated knowledge state).

Nevertheless, the above observations do not influence our conclusion as the superiority of our model comes from its ability to adapt to students' knowledge (§ 5.3.1) and generate customized exercises targeting individual needs (§ 5.3.2), independent of the simulation policy.

6 Conclusion

We propose an adaptive and personalized exercise generation model combining recent advances in knowledge tracing and controllable generation using pre-trained LMs. Our approach works by learning the dynamics between exercise difficulty and student vocabulary knowledge in the domain of language learning. Experimental results on real-world language learning data from Duolingo demonstrate that our model can generate adaptive and personalized exercises needed in an Educational setting. We further showcase our model's applicability in Education with simulation studies.

Ethics Statement

The learner data used in this study are anonymized by Settles et al. (2018) and, to the best of our knowledge, do not contain sensitive information. We foresee no further ethical or privacy concerns with the work.

Limitations

We state the limitations of this work from the following aspects. First, we make an initial assumption about the dynamics between exercise difficulty, vocabulary, and student knowledge. While we believe our assumption is sensible in the domain of language learning, we acknowledge that we make some simplifications for the ease of modeling. For example, we measure difficulty using individual performance, whereas a better way could be combining it with inherent problem difficulty, e.g., text complexity. Besides, we only consider vocabulary mastery in defining student knowledge and predicting their performance. Exploring more dimensions of language knowledge (e.g., syntax) might lead to a finer-grained personalization. Second, our model relies on student learning logs to estimate their real-time knowledge states. This model might face the cold start problem when dealing with insufficient history. Though it is beyond the scope of this study, techniques like computerized adaptive testing can be used to combat this problem. Lastly, due to the lack of a real learning environment, we discuss the educational promise of our model with simulation experiments. In the future, a user study can be incorporated to validate our conclusions.

References

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*, pages 175–184.

Manish Agarwal and Prashanth Mannem. 2011. Automatic gap-fill question generation from text books. In *Proceedings of the sixth workshop on innovative use of NLP for building educational applications*, pages 56–64.

Allison Bailey, Nithya Vaduganathan, Tyce Henry, Renee Laverdiere, and Lou Pugliese. 2018. Making digital learning work: Success strategies from six leading universities and community colleges. *Boston: Massachusetts: Boston Consulting Group*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Samantha Adams Becker, Malcolm Brown, Eden Dahlstrom, Annie Davis, Kristi DePaul, Veronica Diaz, and Jeffrey Pomerantz. 2018. Horizon report 2018 higher education edition brought to you by educause. Technical report, EDUCAUSE.

Hao Cen, Kenneth Koedinger, and Brian Junker. 2008. Comparing two irt models for conjunctive skills. In *International Conference on Intelligent Tutoring Systems*, pages 796–798. Springer.

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

Peng Cui and Le Hu. 2021. Topic-guided abstractive multi-document summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1463–1472, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jennifer B Daines, Tonya Troka, and John M Santiago. 2016. Improving performance in trigonometry and pre-calculus by incorporating adaptive learning technology into blended models on campus. In *2016 ASEE Annual Conference & Exposition*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Francisco J Gallego-Durán, Rafael Molina-Carmona, and Faraón Llorens-Largo. 2018. Measuring the difficulty of activities for adaptive learning. *Universal access in the information society*, 17:335–348.

Tanja Heck and Detmar Meurers. 2022. Parametrizable exercise generation from authentic texts: Effectively targeting the language means on the curriculum. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, pages 154–166.

Matthias Holthaus, Tansu Pancar, and Per Bergamin. 2019. Recommendation acceptance in a simple adaptive learning system.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1638–1649, Melbourne, Australia. Association for Computational Linguistics.

Shuyan Huang, Qiongqiong Liu, Jiahao Chen, Xiangen Hu, Zitao Liu, and Weiqi Luo. 2022. A design of a simple yet effective exercise recommendation system in k-12 online learning. In *International Conference* *on Artificial Intelligence in Education*, pages 208–212. Springer.

Christof Imhof, Per Bergamin, and Stéphanie McGarrity. 2020. Implementation of adaptive learning systems: Current state and potential. *Online teaching and learning in higher education*, pages 93–115.

Tanja Käser, Severin Klingler, Alexander G Schwing, and Markus Gross. 2017. Dynamic bayesian networks for student modeling. *IEEE Transactions on Learning Technologies*, 10(4):450–462.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In *International Semantic Web Conference*, pages 382–398. Springer.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041, Online. Association for Computational Linguistics.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. 2022. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 780–799, Seattle, United States. Association for Computational Linguistics.

Anton Osika, Susanna Nilsson, Andrii Sydorchuk, Faruk Sahin, and Anders Huss. 2018. Second language acquisition modeling: An ensemble approach. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 217–222, New Orleans, Louisiana. Association for Computational Linguistics.

Shalini Pandey and George Karypis. 2019. A self attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019, Montréal, Canada, July 2-5, 2019*. International Educational Data Mining Society (IEDMS).

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Naiara Perez and Montse Cuadros. 2017. Multilingual call framework for automatic language exercise generation from free text. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–52.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. *Advances in neural information processing systems*, 28.

Oleksandr Polozov, Eleanor O'Rourke, Adam M Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. 2015. Personalized mathematical word problem generation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Burr Settles, Chris Brust, Erin Gustafson, Masato Hagiwara, and Nitin Madnani. 2018. Second language acquisition modeling. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 56–65, New Orleans, Louisiana. Association for Computational Linguistics.

Burr Settles and Brendan Meeder. 2016. A trainable spaced repetition model for language learning. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1848–1858.

Shuanghong Shen, Qi Liu, Enhong Chen, Zhenya Huang, Wei Huang, Yu Yin, Yu Su, and Shijin Wang. 2021. Learning process-consistent knowledge tracing. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1452–1460, New York, NY, USA. Association for Computing Machinery.

Dongmin Shin, Yugeun Shim, Hangyeol Yu, Seewoo Lee, Byungsoo Kim, and Youngduck Choi. 2021. Saint+: Integrating temporal features for ednet correctness prediction. In *LAK21: 11th International Learning Analytics and Knowledge Conference*, pages 490–496.

Megha Srivastava and Noah Goodman. 2021. Question generation for adaptive education. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 692–701, Online. Association for Computational Linguistics.

Yuni Susanti, Takenobu Tokunaga, Hitoshi Nishikawa, and Hiroyuki Obari. 2017. Controlling item difficulty for automatic vocabulary question generation. *Research and practice in technology enhanced learning*, 12(1):1–16.

Shiwei Tong, Qi Liu, Wei Huang, Zhenya Hunag, Enhong Chen, Chuanren Liu, Haiping Ma, and Shijin Wang. 2020. Structure-based knowledge tracing: an influence propagation view. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 541–550. IEEE.

Sebastian Tschiatschek, Maria Knobelsdorf, and Adish Singla. 2022. Equity and fairness of bayesian knowledge tracing. *arXiv preprint arXiv:2205.02333*.

Vija Vagale and Laila Niedrite. 2012. Learner model's utilization in the e-learning environments. In *DB&Local Proceedings*, pages 162–174. Citeseer.

Zichao Wang, Andrew Lan, and Richard Baraniuk. 2021. Math word problem generation with mathematical consistency and problem context constraints. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5986–5999, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhengyang Wu, Ming Li, Yong Tang, and Qingyu Liang. 2020. Exercise recommendation based on knowledge concept prediction. *Knowledge-Based Systems*, 210:106481.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Louise Yarnall, Barbara Means, and Tallie Wetzel. 2016. Lessons learned from early implementations of adaptive courseware.

Chun-Kit Yeung and Dit-Yan Yeung. 2018. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pages 1–10.

Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. 2013. Individualized bayesian knowledge tracing models. In *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings 16*, pages 171–180. Springer.

Zhenjie Zhao, Yufang Hou, Dakuo Wang, Mo Yu, Chengzhong Liu, and Xiaojuan Ma. 2022. Educational question generation of children storybooks via question type distribution learning and event-centric summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5073–5085.

Qingyu Zhou and Danqing Huang. 2019. Towards generating math word problems from equations and topics. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 494–503, Tokyo, Japan. Association for Computational Linguistics.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Decoding Algorithm

Algorithm 1 Pseudo-code for our Lexical Difficulty Constrained Decoding

Input: Target words $\mathcal{C}$, difficulty d, a collection of score functions $\mathcal{F}$ and their weights α, max step T, beam size k

Output: k hypotheses Y_T in the last step

$Y_0 \leftarrow$ **InitBeam**() ▷ {<BOS>}

for $t = 1, t \le T, t$++ **do**

  $Y_t \leftarrow \varnothing$

  Candidates $\leftarrow$ **Generate**$(Y_{t-1}, 1)$ ▷ expand

  for $F \in \mathcal{F}$ **do** ▷ prune candidates

    $Y_t \leftarrow Y_t \cup \underset{\mathbf{y}_{\le t} \in \text{Candidates}}{\text{argtopk}} F(\mathbf{y}_{\le t})$

  end for

  for $\mathbf{y}_{\le t} \in Y_t$ **do** ▷ generate l-step lookaheads

    $\tilde{\mathbf{y}}_{t+1:t+l} =$ **Generate**$(y_{\le t}, l)$

  end for

  $Y_t \leftarrow \underset{\mathbf{y}_{\le t} \in Y_t}{\text{argtopk}} \sum_{F_i \in \mathcal{F}} \alpha_i F_i(\mathbf{y}_{\le t} \circ \tilde{\mathbf{y}}_{t+1:t+l})$

end for

return Y_T

B Experimental Setup

B.1 Dataset Details

The statistics of our dataset are summarized in Table 5. Each interaction records a target sentence, per-token correctness labels of the student's response, and meta information such as user nationality and response time. We group interactions by `user_id` (anonymous) in temporal order to obtain per-student interaction sequences. Refer to Settles et al. (2018) for more descriptions of the dataset.

Statistics	**Split**		
	Train	**Dev**	**Test**
# of students	2,593	2,593	2,593
# of interactions	824,012	115,770	114,586
# of questions	7,780	5,524	5,847
# of words (KCs)	1,967	1,839	1,879

Table 5: The statistics of SLAM English track.

B.2 Implementation Details

We implement our models using the Transformers library (Wolf et al., 2020)[5]. Our knowledge tracing model is a three-layer LSTM with a hidden size of 100. We train it for 10 epochs with the regularization weights $\lambda_1 = 0.5, \lambda_2 = 0.1$, selected on the validation set. For the exercise generator, we fine-tune a pre-trained BART-base (Lewis et al., 2020) for up to 10 epochs. An early stop strategy is applied when the loss on the validation set does not decrease for three continuous epochs. We first train the DKT and exercise generator separately until both of them converge. Then, we jointly optimized the two models with hyperameters: $\gamma_1 = 1, \gamma_2 = 0.8, \tau = 2$. During generation, we set the beam size to 4. The weights α for word and difficulty constraints are set to 0.1 and 0.5 as the word constraint is easy to achieve in our experiments. We use Nvidia Tesla A100 with 40 GB of GPU memory for training and inference. On a single GPU, one training epoch of the exercise generator takes about 30 minutes, and that of DKT takes about 7 minutes when they are separately trained. Joint training takes a longer time, about an hour for one epoch. We report the average results over three runs.

[5]https://huggingface.co/docs/transformers/index

C Influence of Regularization in KT

To inspect the influence of regularization terms (Eq. 8) on the KT performance, we conduct a grid search for λ_1 and λ_2 on the validation set. As can be seen from Table 6 and Table 7, $\mathcal{L}_{r_1}$ consistently improves exercise-level performance at the cost of sacrificing word-level performance, whereas $\mathcal{L}_{r_2}$ with a suitable weight ($\lambda_2 = 0.3$) can improve both in most cases. This suggests the students' knowledge states transit gradually over time. We choose $\lambda_1 = 0.5, \lambda_2 = 0.1$ for the best balance.

AUC \ λ_2 / λ_1	**0.0**	**0.1**	**0.3**	**0.5**
0.0	79.51	79.50	79.57	79.53
0.1	79.44	79.45	79.49	79.52
0.3	79.42	79.40	79.44	79.36
0.5	79.32	79.43	79.41	79.30

Table 6: Validation results (AUC×100) of word-level prediction under varying regularization weights.

AUC \ λ_2 / λ_1	**0.0**	**0.1**	**0.3**	**0.5**
0.0	70.89	70.98	70.85	71.15
0.1	71.04	71.02	71.06	71.23
0.3	71.41	71.31	71.43	71.31
0.5	71.41	71.48	71.45	71.45

Table 7: Validation results (AUC×100) of exercise-level prediction under varying regularization weights.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Ethical and Privacy Considerations

☑ A3. Do the abstract and introduction summarize the paper's main claims?
3

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Appendix B.1

☑ B1. Did you cite the creators of artifacts you used?
Appendix B.1

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Appendix B.1

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix B.1

C ☑ Did you run computational experiments?

Appendix B.2

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix B.2

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix B.2

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Appendix B.2

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix B.2

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.