Programming Knowledge Tracing based on Problem Solution Embedding

[1st] Yongfeng Huang[1], [2nd] Rongfang Wang[2]

{yfhuang@dhu.edu.cn[1], 2212647@mail.dhu.edu.cn[2]}

Donghua University, Shanghai, China

Abstract. With the development of the internet and information technology, using online learning systems for programming practice has increasingly become a new trend. In this process, continuously tracking students' proficiency in programming skills is also particularly important. However, current research on students' programming practice mainly focuses on their submitted code, neglecting the importance of official exercise solutions in programming competitions for assessing students' proficiency in programming skills. Therefore, we propose a new improved model for graph-based knowledge tracing (CTGKT). Specifically, by embedding official exercise solutions, combined with students' submitted codes and the exercise text contents, it assesses the proficiency of students in programming skills. Experiments on our programming competition practice dataset demonstrate that CTGKT model achieves state-of-the-art performance compared to existing methods.

Keywords: knowledge tracing; performance prediction; feature fusion; CodeBERT

1 Introduction

Knowledge tracing(KT) utilizes students exercise records to monitor their knowledge states and predict future exercise performance. Existing methods for tracking programming skill proficiency mainly rely on KT, specifically divided into two methods: deep knowledge tracing (DKT) and graph-based knowledge tracing(GKT).However, the two methods mentioned above overlook the relationship between official exercise solutions and skills. For programming competition practice, using a brute-force solution versus using a high-level official solution to the same problem indicates different levels of proficiency in programming skills.

Regarding this issue, we have conducted relevant research on the above issues and proposed a new improved model based on graph-based knowledge tracing(CTGKT). CTGKT utilizes RoBERTa to extract feature vectors from the official exercise solutions and exercise text contents of complex programming problems, while CodeBERT is employed to capture the feature representations of students' submitted code. Based on the graph-based knowledge tracing model, we introduce an attention mechanism for feature fusion and updating students' knowledge states. This enables us to better predict the accuracy of students feature performance, thus indicating their proficiency in programming skills. In addition, we conducted experiments on our dataset to validate the effectiveness of CTGKT, and the results indicate that CTGKT outperforms existing methods in terms of performance.

EIMT 2024, March 29-31, Wuhan, People's Republic of China

DOI 10.4108/eai.29-3-2024.2347724

2 Related Work

2.1 Knowledge Tracking

Inspired by the success of deep learning, recent knowledge tracing research has employed deep learning techniques. The main idea of DKT utilizes recurrent neural networks as the fundamental architecture to establish interactions between programming skills and corresponding student exercises, effectively modeling the learning process of students[1][2].In recent years, GKT has gradually been introduced into the field of programming education, graph structures can effectively represent the relationships between multiple questions and multiple skills. The main idea of GKT utilizes graph representation techniques to represent the characteristics of students' exercises. By modeling the relationship between problems and skills, it effectively measures students' proficiency in programming skills during the exercise process[3][4].

2.2 Feature embedding

In addition, the BERT model can also be embedded into the DKT model to track students' proficiency in programming skills. The main idea is as follows: The BERT model, based on the Transformer architecture, adopts a bidirectional contextual encoding approach, making it excellent at capturing contextual information from text[5]. RoBERTa builds upon BERT, improving model performance by using a larger dataset, a larger batch size, and more training iterations[6]. To better understand the structure and semantic information of source code, Microsoft Research proposed the CodeBERT model, which can deeply analyze the inherent logic and context of source code[7].

3 Problem Statement

Assuming a dataset for students' competition training, it is composed of all the competition students, namely $U=\{u_1,u_2,...,u_m\}$, $i \in [1,m]$. Among them, ui is the overall exercise record of a student, namely $u_i=\{u_{i1},u_{i2},...,u_{in}\}$, $j \in [1,n]$. u_{ij} is a exercise record of the student, namely $u_{ij}=\{q_i,s_i,t_i,c_i,a_i\}$. q_i is the exercise question of the record,s_i is one or more skills corresponding to the question, t_i is the official exercise solution to the question, c_i is the submission code of the student's answer to the question, and a_i is the student's response result($\{0,1\}$) to the question.

Problem Formalization: Given the students' exercise records $U=\{u_1,u_2,...,u_m\}$ and a new question q_{t+1}, the knowledge tracing model is used to update the students' knowledge state and predict the accuracy of their results to the new question, namely $p_{t+1}(a_{t+1}=1|U,q_{t+1})$.

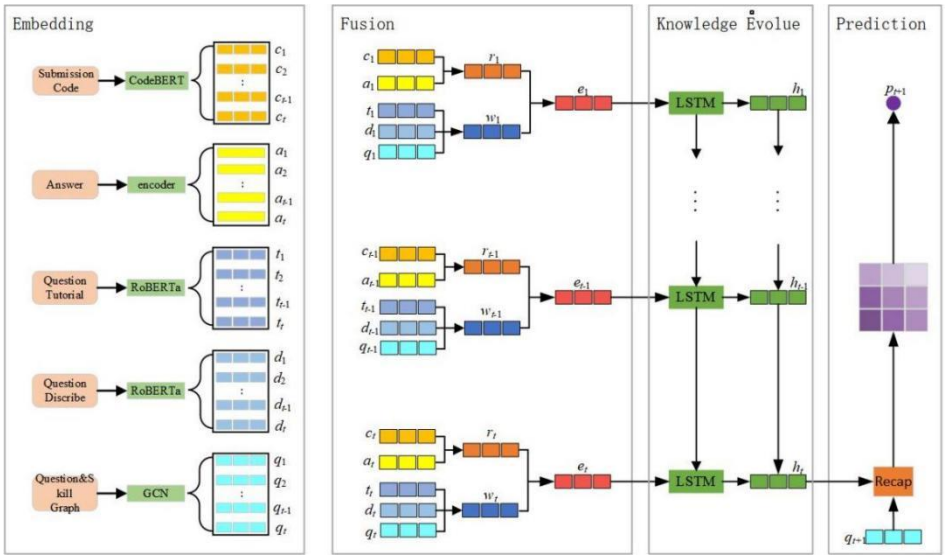

Figure.1. TCGKT

4 The Proposed Model CTGKT

The architecture of CTGKT is depicted in Figure 1, primarily consists of a feature embedding module and a knowledge tracing module.

4.1 Embedding Module

This section will provide a detailed introduction on how the CTGKT model captures feature embedding in students' programming competition practices through the embedding module.

For students participating in programming competitions, their submitted code during daily practice contains information about their proficiency in programming skills and their ability to comprehensively apply competitive skills. We use CodeBERT to transform students' submitted code into a universal and comparable vector representation that comprehensively reflects the features and semantic information of the code.

For programming competition practices, the focus is not limited to basic syntax knowledge. Even though different solutions to a single problem can yield correct results, they may vary significantly in terms of time complexity and space complexity. The official exercise solutions, as an essential component of competition exercises, provide us with rich problem-solving approaches. Additionally, considering that programming competition problems often contain long textual descriptions information, we concatenate these two embedding to form a complete embedding representation of the problem information. We utilize RoBERTa to generate embedding for exercise texts and official exercise solutions, and connect these two embedding to form a complete representation of the problem information embedding.

In the context of programming competition practice, each problem typically involves one or

more skills, and a skill may also appear in multiple problems. To gain a deeper understanding of the relationship between problems and skills, we adopt a graph structure to establish connections between questions and skills. And we use graph convolutional network(GCN) to conduct a thorough analysis of this relationship, and obtain the aggregated embedding of questions and skills.

4.2 Fusion Module

In the fusion module, we organically integrate the submission code embedding **c**, office exercise solution embedding **t**,the exercise text embedding **d**, question embedding **q**, and answer embedding **a** obtained from the embedding module to acquire a more comprehensive exercise embedding **e**. Firstly, we utilize the answer fusion module to fuse **c** with **a**, resulting in an aggregated embedding **r** that combines the code representation of the students' problem-solving process with the semantic information of the final answer. Subsequently, the question fusion module combines **t**, **d**, and **q** to generate an aggregated embedding **w**, which aims to integrate different types of information related to the question, including the in-depth explanation of the official solution, text description, and the semantic meaning of the question itself. Finally, we input the two aggregated embedding **r** and **w** into a nonlinear neural network for further integration and processing, thereby generating the desired final exercise embedding **e**. This embedding represents the overall representation of the students' programming competition exercise, encompassing their submission code representation, the semantic information of the answer, and the comprehensive information related to the question. The entire process can be expressed as formulas (1), (2), and (3).

$$\boldsymbol{r}_t = \mathrm{Relu}(\boldsymbol{W}_1[\boldsymbol{c}_t, \boldsymbol{a}_t] + \boldsymbol{b}_1) \tag{1}$$

$$\boldsymbol{w}_t = \mathrm{Relu}(\boldsymbol{W}_2[\boldsymbol{t}_t, \boldsymbol{d}_t, \boldsymbol{q}_t] + \boldsymbol{b}_2) \tag{2}$$

$$\boldsymbol{e}_t = \mathrm{Relu}(\boldsymbol{W}_3[\boldsymbol{w}_t, \boldsymbol{r}_t] + \boldsymbol{b}_3) \tag{3}$$

W_1、W_2、W_3、b_1、b_2、b_3 are trainable matrices and parameters.

4.3 Assessment and Prediction Module

The model generates students' knowledge states through the aforementioned exercise embedding **e**, and based on these knowledge states, the model can further make predictions using this information. To ensure effective training of the model, we have chosen the Adam optimization method. This method continuously updates the parameters in the model by minimizing the loss function, thereby enhancing the model's prediction accuracy and performance. The loss function is expressed as formula (4).

$$\mathbb{L}(\theta) = -\sum \left(r_{t+1} \log p_{t+1} + (1 - r_{t+1}) \log \left(1 - p_{t+1}\right)\right) + \lambda_\theta ||\theta||^2 \tag{4}$$

5 Experiments

In this section, we first introduced the real dataset used in the experiment, and then conducted comparison and ablation experiments to verify the effectiveness of the model.

5.1 Dataset

We conducted a statistical analysis on the daily exercise records of all participants from our

university who took part in programming contests. After necessary data cleaning procedures such as removing null values and duplicate submissions, we organized 8,714 submission records in total, which correspond to 1,089 programming contest problems with official exercise solutions. These submission records contain detailed information such as problem IDs, required programming skills, official solutions, the exercise text, students' submitted code, and the results of students' results to the problems. The total number of annotated skills in the datasets is 37, with example skills including greedy algorithms, recursion, and constructive algorithms. In terms of datasets partitioning, we randomly divided it into a training set and a test set at a ratio of 4:1. To further improve the generalization ability of the model, we further divided the training set into five parts for cross-validation.

Table 1. Comparison experiment

Model	AUC	Model	AUC
DKT	71.36%	SAKT	76.98%
DKVMN	74.75%	SAINT	77.22%
DKT+	75.21%	AKT	77.33%
KQN	75.56%	IEKT	78.39%
SKVMN	75.13%	ATKT	78.86%
GKT	76.04%	CTGKT	83.42%

5.2 Comparison Experiment

To evaluate the effectiveness of the CTGKT model in predicting students' performance, we conducted a comparative experiment. After preprocessing our own dataset, we trained and compared it with existing Knowledge Tracing (KT) models. All models were trained using the Pytorch framework on a CPU platform.

As shown in Table 1, through the comparison experiment on our datasets, we found that the CTGKT model outperformed other comparative models in terms of the AUC metric. This indicates that the CTGKT model is better able to capture students' knowledge states and changes, providing more accurate prediction results.

5.3 Ablation Experiment

To assess the impact of different embedding components on the performance of the CTGKT model, we conducted an ablation study. We designed several variants of CTGKT. Here are the detailed descriptions of these variants:

CTGKT-baseline: the baseline model, which does not include any additional information.

CTGKT-tutorial: a variant that only uses official exercise solution information as input.

CTGKT-code: a variant that only utilizes students submitted code information as input.

CTGKT-describe: a variant that only uses the exercise text information as input.

CTGKT-tutorial, code: a variant that incorporates both official exercise solution and students submitted code information as input.

CTGKT-tutorial, describe: a variant that incorporates both official exercise solution and the exercise text information as input.

CTGKT-code, describe: a variant that incorporates both students submitted code and the exercise text information as input.

CTGKT: the complete CTGKT model, through which we can understand the predictive power when all components work together.

Table 2. Ablation experiment

Model	F1-score	ACC	AUC
CTGKT-baseline	71.22%	80.62%	77.52%
CTGKT-tutorial	77.28%	83.96%	81.33%
CTGKT-code	78.06%	83.65%	81.81%
CTGKT-discribe	77.43%	83.79%	81.45%
CTGKT-tutorial,code	78.91%	84.65%	82.49%
CTGKT-tutorial,discribe	78.47%	84.03%	82.05%
CTGKT-code,discribe	78.29%	84.35%	82.24%
CTGKT	80.18%	85.25%	83.42%

As shown in Table 2, when considering the official exercise solutions, the exercise texts, and students submitted code information comprehensively, CTGKT exhibits the best performance in evaluation metrics such as AUC, ACC, and F1-score. Further comparing CTGKT with other variant models, we find that CTGKT-tutorial, CTGKT-code, and CTGKT-describe all show improvements compared to CTGKT-baseline. This indicates that official exercise solution, the exercise texts, and students submitted code all play positive roles in improving model performance. Among them, CTGKT-code performs particularly well, suggesting that students submitted code plays an indispensable role in enhancing model effectiveness. Moreover, when comparing the three variant models of CTGKT-tutorial, code, CTGKT-tutorial, describe, and CTGKT-code, describe, it can be observed that introducing two types of student information simultaneously is more effective in improving model performance than introducing only one type of information.

6 Conclusions

Addressing the shortcomings of existing knowledge tracing models in handling data from programming competition students, who overlook the relationship between solution information and skills, we propose a novel graph-based knowledge tracing model, namely CTGKT. This model innovatively incorporates official exercise solution, not only considering the impact of students submitted code and the exercise text on measuring proficiency in programming skills, but also fully incorporating the influence of solution information on complex problems and skills. Through experiments, we have validated the effectiveness of CTGKT. In the future, we will further explore the impact of different programming competition practice methods (such as daily practice and competition practice) on modeling students' proficiency in programming skills.

References

[1] Hui-Chun Hung, Ping-Han Lee:Applying Deep Knowledge Tracing Model for University Students' Programming Learning. ICOIN 2023: 574-577

[2] Yang Shi, Min Chi, Tiffany Barnes, Thomas W. Price:Code-DKT: A Code-based Knowledge Tracing Model for Programming Tasks. EDM 2022

[3] Renyu Zhu, Dongxiang Zhang, Chengcheng Han, Ming Gao, Xuesong Lu, Weining Qian, Aoying Zhou:Programming Knowledge Tracing: A Comprehensive Dataset and A New Model. ICDM (Workshops) 2022: 298-307

[4] Ruixin Li, Yu Yin, Le Dai, Shuanghong Shen, Xin Lin, Yu Su, Enhong Chen:PST: Measuring Skill Proficiency in Programming Exercise Process via Programming Skill Tracing. SIGIR 2022: 2601-2606

[5] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT (1) 2019: 4171-4186

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov:RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR abs/1907.11692 (2019)

[7] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, Ming Zhou:CodeBERT: A Pre-Trained Model for Programming and Natural Languages. EMNLP (Findings) 2020: 1536-1547