计算机科学
COMPUTER SCIENCE

基于深度强化学习与程序分析的OJ习题推荐模型

金天成, 窦亮, 张伟, 肖春芸, 刘峰, 周爱民

引用本文

金天成, 窦亮, 张伟, 肖春芸, 刘峰, 周爱民基于深度强化学习与程序分析的OJ习题推荐模型[J]. 计算机科学, 2023, 50(8): 58-67.

JIN Tiancheng, DOU Liang, ZHANG Wei, XIAO Chunyun, LIU Feng, ZHOU Aimin. OJ Exercise Recommendation Model Based on Deep Reinforcement Learning and Program Analysis [J]. Computer Science, 2023, 50(8): 58-67.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220600260

基于深度强化学习与程序分析的OJ习题推荐模型

金天成[1,2]　窦　亮[2]　张　伟[1,2]　肖春芸[2]　刘　峰[1,2]　周爱民[1,2]
1 华东师范大学上海智能教育研究院　上海 200062
2 华东师范大学计算机科学与技术学院　上海 200062
(52205901026@stu.ecnu.edu.cn)

摘　要　当前Online Judge系统(简称OJ)上存有大量习题,导致学生很难根据自己的知识水平和学习需求快速地找到合适的习题,因此需要设计模型向学生推荐习题。然而,由于OJ的独特性以及程序设计能力评价的复杂性,现有推荐模型不能较好地完成OJ习题推荐任务,主要问题包括:OJ习题知识点标签不足与特有的命题风格使模型难以挖掘习题之间的相关性;学生所提交程序的实际正确性与OJ判定结果存在不一致的情况,使得模型对学生知识状态的评估产生偏差;现有模型较难提供可使学生程序设计能力得到显著增长的习题。据此,提出了一种基于深度强化学习与程序分析的OJ习题推荐模型。首先,分析习题的最优解来挖掘习题之间的相关性;然后,比较学生所提交程序与习题最优解的相似性来检验学生所提交程序的实际正确性,使模型能够更准确地估计学生的知识状态;最后,利用深度强化学习技术并使用知识追踪模型作为学生模拟器,以学生模拟器在解答习题推荐模型所提供的习题前后在所有习题上的表现差异作为奖励,使模型学习到怎样的习题才能够最大程度地提升学生程序设计能力,并将这样的习题推荐给学生。在业界知名OJ系统CodeForces和Libre数据集上进行实验,结果表明该模型相比目前常见的推荐模型具有更优的性能。

关键词:推荐系统;深度强化学习;程序分析;知识追踪;在线判题
中图法分类号　TP301

OJ Exercise Recommendation Model Based on Deep Reinforcement Learning and Program Analysis

JIN Tiancheng[1,2],DOU Liang[2],ZHANG Wei[1,2],XIAO Chunyun[2],LIU Feng[1,2] and ZHOU Aimin[1,2]
1 Shanghai Institute of AI for Education,East China Normal University,Shanghai 200062,China
2 School of Computer Science and Technology,East China Normal University,Shanghai 200062,China

Abstract　At present,there are a large number of exercises on the existing programming Online Judge systems(OJ),which makes it difficult for students to quickly find suitable exercises according to their own knowledge level and learning demand. Therefore, it is necessary to design a model to recommend suitable exercises to students. However,due to uniqueness of OJ and complexity of programming ability evaluation,existing recommendation model can not complete OJ exercise recommendation task well,the main problems include:OJ exercises' lack of knowledge label and unique proposition style make it difficult for existing models to mine correlation between exercises; actual correctness of the program submitted by student is inconsistent with OJ judgement result, which leads to deviation of students' knowledge state estimated by models; existing models are difficult to provide exercises that increase students' programming ability most significantly. Based on this,this paper proposes an OJ exercise recommendation model based on deep reinforcement learning and program analysis. Firstly,analyzing optimal solution of exercises to mine correlations between exercises. Then,comparing the similarity between programs submitted by students and optimal solution of exercises to check actual correctness of the programs submitted by students,so that knowledge state of students can be estimated more accurately. Finally,using deep reinforcement learning technology,taking knowledge tracking model as student simulator and treating student simulator's performance difference on all the exercises before and after answering exercises provided by exercise recommendation model as reward,so that exercise recommendation model can learn which exercise is able to improve the students' programming ability to the greatest extent,and recommend such exercises to students. This paper conducts extensive experiments on two datasets CodeForces and Libre of the well-known OJ system,and experimental results show that the proposed model can

到稿日期:2022-06-28　返修日期:2022-11-05
基金项目:国家自然科学基金青年科学基金(61907015);上海市科学技术委员会高新技术领域项目(20511102502)
This work was supported by the Young Scientists Fund of the National Natural Science Foundation of China(61907015) and Shanghai Committee of Science and Technology,China(20511102502).
通信作者:窦亮(ldou@cs.ecnu.edu.cn)

achieve higher performance than the state-of-the-art recommendation models.

Keywords Recommender system，Deep reinforcement learning，Program analysis，Knowledge tracing，Online judge

1 引言

Online Judge 系统（简称 OJ）是一种在线判题系统，通过预先设计的测试数据来检验用户提交的程序源代码的正确性，已被广泛用于计算机程序设计教学、训练和竞赛等。各个 OJ 提供的习题数量少则几千，多则上万，其中有编程基础题，也有国际程序设计竞赛中的真题，难度梯度跨度很大，题目类型多样。当前绝大部分 OJ 仅用于程序测评，并不提供个性化的习题推荐服务，这导致学习者很难根据自己的学习需求和知识水平快速地找到合适的习题，出现“信息过载”，降低了学习效率，甚至影响了学习兴趣。尽管近年来智能教育已成为教育的发展趋势，强调利用智能技术辅助实现个性化学习，然而当前的研究多数是针对 ASSISTments、Santa、学堂在线和智学网等在线教育系统的，对 OJ 的个性化推荐研究较为缺乏。探索 OJ 习题推荐方法，有助于促进学习者程序设计能力的培养，也符合现今教育手段与学习方式深度变革的时代要求。

近年来，研究者提出了许多习题推荐方法，包括基于协同过滤的方法、基于认知诊断的方法和基于知识追踪的方法等。基于协同过滤的推荐方法[1-2]利用传统推荐算法将学习资源（如习题、课程）推荐给学生。这类方法使用相似学生的学习偏好对目标学生进行偏好预测，进而根据预测的偏好得分进行学习资源推荐。基于认知诊断或知识追踪的推荐方法[3-9]则通过对学生的知识状态进行建模来实现习题推荐。这类方法首先使用认知诊断[10-12]或知识追踪[13-16]技术对学生的历史答题情况进行分析，得到学生对于每道习题的答对概率，然后将“未掌握”的习题（即答对概率不高的习题）推荐给学生，以帮助学生克服弱点。

尽管现有的习题推荐研究取得了较好成果，然而由于 OJ 的独特性以及程序设计能力评价的复杂性，现有方法仍不足以较好地完成 OJ 习题推荐任务，具体分析如下：

（1）因习题知识点标签不足而影响推荐性能。现有习题推荐方法往往通过习题所涉及的知识点来挖掘习题之间的相关性，但 OJ 中往往有较多习题没有被标注知识点标签，如业界知名 OJ 系统 Libre[17]和洛谷[18]均有超过 30％的习题没有标签。大量缺少知识点标签的习题会干扰推荐模型的性能。虽然有研究[19]通过数学习题描述中的文本和图像来挖掘习题之间的相关性，可以帮助解决知识点标签缺失的问题，但 OJ 习题多为计算机实践场景描述，习题描述中基本不会出现习题对应的知识点，并且知识点相同的习题描述也完全不同，因此较难从 OJ 习题描述中挖掘出习题之间的相关性。如下所示为 Libre 中的一道习题“信息传递”的描述，该习题所考查的知识点为“图论”和“并查集”，但在习题描述中完全没有出现与知识点相关的名词。

有 n 个同学（编号为 1 到 n）正在玩一个信息传递的游戏。在游戏里每人都有一个固定的信息传递对象，其中，编号为 i 的同学的信息传递对象是编号为 T_i 的同学。游戏开始时，每人都只知道自己的生日。之后每一轮中，所有人会同时将自己当前所知的生日信息告诉各自的信息传递对象（注意：可能有人可以从若干人那里获取信息，但是每人只会把信息告诉一个人，即自己的信息传递对象）。当有人从别人口中得知自己的生日时，游戏结束。请问该游戏一共可以进行几轮？

（2）OJ 的判题方式可能导致对学生知识状态的评估存在偏差。OJ 通过预先设计的若干测试用例来判定所提交程序的正确性（见图 1），但有时学生所提交程序的实际正确性与 OJ 判定结果会出现不一致的情况。出现这种情况的原因至少有两个：1）不同编程语言实现的相同算法对应的时间复杂度不同，如 C＋＋程序的运行时间一般比 Java 和 Python 程序短，若习题测试用例的时限设置过小，则会使正确的 Java，Python 程序被判为错误；2）某些知识点所涉及的 OJ 习题存在简单暴力的解决方法，如 KMP 算法的相关习题，如果测试用例设计得不合适，学生虽然不会 KMP 算法，但提交使用 Brute-Force 算法的程序也能通过。现有习题推荐工作仅使用在线教育系统的判定结果来评估学生的知识状态，导致所得到的知识状态产生偏差。

（3）OJ 习题推荐的主要挑战是如何设计最优的推荐策略，来提高学生正确解答习题的概率，从而达到高效提升程序设计能力的目的。基于协同过滤的方法往往将具有相同学习偏好的学生所做的习题推荐给目标学生，并不直接对应于高效提升程序设计能力的目的。基于认知诊断或知识追踪的方法向学生推荐“未掌握”（正确概率不是很高）的习题，帮助学生克服弱点，但确定最合适的正确概率值并非易事。如果概率值过高，则会推荐给学生过于简单的习题，无法保证知识水平的有效提升；如果概率值过低，则会向学生推荐过于困难的习题，导致学生学习动力下降，出现畏难心理。

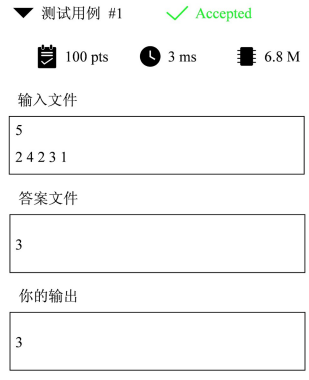

图 1 测试用例

Fig. 1 Test case

为解决上述问题，本文针对 OJ 习题提出了一种基于深度强化学习与程序分析的习题推荐模型，称为 ERDP（Exercise Recommendation Model Based on Deep Reinforcement Learning and Program Analysis）模型，主要创新包括：

（1）由于具有相同知识点的程序设计习题的代码实现往往也相似[20]，因此本文通过程序分析来挖掘习题最优解

(定义为习题对应的提交记录中正确且运行时间最短的程序)中所蕴含的知识点信息,并加入习题推荐模型来解决“习题知识点标签不足”的问题,提升模型的性能。

(2)本文首先将学生提交的 Java 和 Python 程序转为 C++程序来排除编程语言不同所导致的语法差异性;然后使用程序分析方法计算所提交程序与习题最优解的语法结构相似性,以此检验提交程序的实际正确性并加入习题推荐模型,从而解决“OJ 的判题方式可能导致对学生知识状态的评估存在偏差”的问题,使模型能够更准确地估计学生的知识状态。

(3)本文利用深度强化学习技术并使用知识追踪模型作为学生模拟器,以学生模拟器在解答习题推荐模型所提供的习题前后的所有习题正确解答概率差的平均值作为即时奖励,该值越大,说明学生程序设计能力的增长程度也越大,习题推荐模型也能获得更大的奖励值,从而让模型学习到怎样的习题才能够最大程度地提升学生正确解答习题的概率。将这样的习题推荐给学生,达到令学生程序设计能力增长程度最大的目的。

2 相关工作

2.1 学习资源推荐

2.1.1 协同过滤

目前已有不少基于协同过滤的学习资源推荐方法,如 Gong 等[1]以学生、课程、视频、教师和知识概念作为实体构建异构信息网络,利用图卷积网络对学生和知识点进行表示学习,再使用矩阵分解的方法实现知识点推荐;Shao 等[2]将学生预选的感兴趣课程引入基于 BERT 的课程推荐模型中,使用课程和学生的信息构成课程和学生的表征,取代通过随机方式生成的表征,提升模型的精度。

2.1.2 认知诊断

教育心理学中的认知诊断技术假设学生的知识状态在一定时间内固定不变,利用学生的历史答题情况和学生作答的每道习题所包含的知识点来评估学生目前的知识水平并预测学生在未来学习中的表现。Fan[10]提出了 IRT 模型,通过学生的答题情况和习题难度推断出学生的能力值;Torre[11]提出了 DINA 模型,引入 Q 矩阵,将学生建模成一个在多维知识点上的掌握向量,其中向量的每一维二进制变量表示学生是否掌握了对应维度上的知识点;为了解决认知诊断模型没有考虑相似学生的共性特征的问题,文献[7-9]将协同过滤方法与认知诊断模型进行了结合;Gao 等[12]提出了 RCD 模型,引入知识点关系图,使模型能够考虑知识点之间的依赖关系。

2.1.3 知识追踪

在真实环境下,学习活动是一个长期的过程,学生对知识的学习也是循序渐进的,因此,如何动态捕捉学生知识掌握程度的变化情况对于学生学习行为的建模具有重大意义。基于这样的考虑,Corbett 等[13]提出了知识追踪技术,旨在通过学生的历史学习数据,跟踪学生知识水平的变化情况,并提出了贝叶斯知识追踪(BKT)模型,用于预测学生成绩;Piech 等[14]提出了深度知识追踪(DKT)模型,通过训练门控制循环单元(GRU)网络对学生的知识掌握状态进行建模;Zhang 等[15]在设计知识追踪模型时,考虑了遗忘和学习行为,使学生知识状态的评估更准确;Choi 等[16]提出了 SAINT 模型,首次将编码器-解码器应用于知识追踪任务,分离了练习序列和学生反应序列,使模型能够通过深层的自注意力机制捕捉练习和学生反应之间的复杂关系。

2.1.4 强化学习在学习资源推荐中的应用

有学者将强化学习技术引入学习资源推荐模型。Huang 等[21]提出了 DRER 模型,将“复习与探索之间的权衡”“难度的平滑性”和“学生的参与度”3 个策略量化为 3 个不同的奖励机制并组成目标函数,使用这个目标函数来训练 Deep Q-Network,实现了习题推荐的多元化。Zhang 等[22]提出了 HRL 模型,并将其应用于课程推荐任务。该工作指出了基于注意力机制的序列推荐模型中存在的一个问题:当一个学生的历史学习兴趣很复杂时,对目标推荐课程有贡献的课程的影响效果会被不同种类的其他历史学习课程所稀释,导致注意力机制表现变差。HRL 使用基于强化学习的用户画像修改器将与目标课程相关度很小的课程去除,从而获得更准确的课程推荐结果。Lin 等[23]提出了 HELAR 模型,将动态循环机制引入 HRL,令用户画像修改器在利用当前最优策略时能够感知上下文,并实现小规模探索,提升 HRL 的推荐性能。Liu 等[24]提出了 CSEAL 模型,利用学生在学习一系列课程前的考试成绩以及学完课程后的考试成绩之差来估计学生的知识水平增长程度,并将其作为基于 Actor-Critic 的课程推荐模型奖励值来训练网络,使得学生在学习所推荐的课程后能够提高下一次考试成绩。受此工作启发,本文方法将学生模拟器在解答习题推荐模型所提供的习题前后在所有习题上的表现差异作为奖励,可以在不需要学生考试数据的情况下衡量学生知识水平增长的程度。

2.2 程序分析

早期的程序分析工作受自然语言处理工作的影响,将编程语言看作自然语言,从而将自然语言处理模型应用于程序分析任务[25-26],但编程语言与自然语言不同,编程语言的语法规范更为严谨。因此,部分工作提出了基于抽象语法树(Abstract Syntax Tree,AST)的程序分析模型,将程序转换为 AST 进行分析。Mou 等[27]提出了 TBCNN 模型,在 AST 上做卷积处理,将 AST 映射成蕴含语法结构信息的向量;Li 等[28]认为 AST 是弱化的图,并提出了 GGNN 模型,使用门控图神经网络处理 AST;Tan 等[20]提出了 GGANN 模型,将程序的数据流和函数调用信息加入 AST 生成 EAST(Extended Abstract Syntax Tree),并用门控图注意力神经网络学习 EAST 的拓扑结构,得到程序的表征向量。

2.3 OJ 习题推荐发展现状

CodeForces[29]是由俄罗斯的萨拉托夫国立大学开发的 OJ,对习题按知识点进行了分类,方便学生学习自己所需的知识点。美国计算奥林匹克的官方网站 USACO[30]是美国著名的 OJ,其中的训练页面呈现课程的形式,由易到难递进,每个章节有若干习题并搭配相应的讲解,构成了一条较为完整的学习路径。Libre[17]是中国人民大学创建的开源 OJ,习题按照难度、知识点、习题来源等多个维度对题目进行了归类,学生可以根据自己的学习需求选择习题进行训练。国内著名的 OJ 系统还有北京大学的 POJ[31]、浙江大学的 ZOJ[32]和

杭州电子科技大学的 HDUOJ[33]。上述 OJ 均暂未有对学生进行个性化推荐的功能，而 Kattis[34] 和洛谷[18] 分别具有不同的习题推荐机制：Kattis 使用习题错误率来估计习题的难度，以学生作答的习题数和答题正确率来估计学生能力，向学生推荐难度与能力匹配的习题；洛谷则推荐与学生当前所答习题具有相近知识点的习题。此外，Zheng 等[35] 通过 C 语言知识结构树和学生表现计算学生的认知水平，将难度与学生认知水平最匹配的习题推荐给学生。文献[36-39]使用协同过滤实现了 OJ 习题推荐，其中 Yera 等[36] 在推荐方法中引入了模糊逻辑，Mao[39] 考虑了时间的影响。

3 问题定义

在一个 OJ 系统中，假设有 $|U|$ 个学生、$|E|$ 道习题和 $|K|$ 个知识点。学生 $u(u\in U)$ 的历史答题序列为 $i^u=\{(e_1,l_1),(e_2,l_2),\cdots,(e_T,l_T)\}$，其中，$e_t(e_t\in E)$ 是学生在 t 时刻解答的习题，l_t 为系统判定的结果，$l_t=1$ 表示判定结果为正确，$l_t=0$ 表示判定结果为错误。

对于每道习题 $e\in E$，用四元组 $e=(g,c,k,d)$ 表示，其中，g 代表习题对应实现程序的 EAST，c 代表习题描述的内容文本，表示为一组词语序列 $c=\{w_1,w_2,\cdots,w_M\}$，$k\subseteq K$ 代表习题所考查的知识点集合，d 代表习题的难度，习题的难度取决于学生作答错误率、提交作答错误率和区分度。

本文将习题推荐的过程形式化为马尔可夫决策过程(Markov Decision Process，MDP)，对马尔可夫决策过程中的 4 种要素，即状态集合 S、行动集合 A、奖励函数 R 和状态转移函数 Γ 进行定义。

(1)状态集合 S。S 代表学生的状态空间。状态 $s_t^u\in S$ 由学生 u 在 t 时刻及之前的历史答题序列所构成，即 $s_t^u=i_{1:t}^u$。

(2)行动集合 A。A 代表行动空间，包含 OJ 中所有的习题。OJ 习题推荐系统在状态 s_t^u 下采取了行动 $a_t^u\in A$，相当于给学生 u 推荐了习题 $\hat{e}_{t+1}$。

(3)奖励函数 R。R 是奖励函数，$r_t^u=R(s_t^u,a_t^u)$ 代表的是 OJ 习题推荐系统在状态 s_t^u 下采取行动 a_t^u 所获得的及时奖励，奖励值的大小由学生程序设计能力的增长程度决定。

(4)状态转移函数 Γ。当 OJ 习题推荐系统在状态 s_t^u 下采取行动 a_t^u 之后，状态转移函数 Γ 能将当前的状态从 s_t^u 转移为 s_{t+1}^u，其中，$s_{t+1}^u=i_{1:t+1}^u$。

本文的目标就是要找到一个向学生推荐习题的最优策略 $\pi:S\to A$。当学生状态为 s_t^u 时，OJ 习题推荐系统能够根据策略 π 从习题集合 E 中选取一个习题 $\hat{e}_{t+1}$ 推荐给学生 u，使得整个推荐过程所累积的奖励值最大化。

4 方法描述

本文针对 OJ 习题推荐任务提出了 ERDP 模型，整体架构如图 2 所示。首先，在习题编码的过程中加入习题最优解来帮助模型挖掘习题之间的相关性；然后，将习题表征输入 GRU 网络得到仅考虑了系统判定结果的学生知识状态，并通过学生所提交程序与习题最优解的相似性来矫正这些知识状态；最后，利用深度强化学习技术并以知识追踪模型在解答习题推荐模型提供的习题前后的所有习题正确解答概率差的平均值作为奖励训练推荐模型。为了便于阅读，后文从单个学生 u 的视角来描述推荐方法，将第 3 节中出现的上标 u 从公式中去除。

4.1 习题编码

为减小部分习题知识点标签缺失对推荐模型性能的影响，本文在习题描述、知识点和难度的基础上，加入习题最优解来完善习题的表征向量。

对于习题 e_t 的最优解，本文将其转换为 EAST，表示为 g_t。将 g_t 输入门控图注意力神经网络 GGANN，得到程序的表征向量 $\boldsymbol{g}_t\in\mathbb{R}^{d_g}$，如式(1)所示：

$$\boldsymbol{g}_t=GGANN(g_t;\Phi_{GGANN}) \tag{1}$$

其中，Φ_{GGANN} 为 GGANN 网络的相关参数。

对于描述习题 e_t 的词语序列 $c_t=\{w_1,w_2,\cdots,w_M\}$，本文使用 Word2Vec 预训练获得每个词语 w_i 的词向量 $\boldsymbol{w}_i\in\mathbb{R}^{d_w}$，将词向量输入 LSTM 网络，得到习题描述的表征向量 $\boldsymbol{c}_t\in\mathbb{R}^{d_c}$，如式(2)所示：

$$\boldsymbol{c}_t=LSTM(\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_M;\Phi_{LSTM}) \tag{2}$$

其中，Φ_{LSTM} 为 LSTM 网络的相关参数。

对于习题 e_t 所考查的知识点集合 k_t，本文先获取其 multi-hot 表征 $\boldsymbol{v}_t\in\{0,1\}^{|K|}$，再使用参数矩阵 $\boldsymbol{W}_k\in\mathbb{R}^{d_k\times|K|}$ 将 $\boldsymbol{v}_t$ 转换为低维稠密的知识点表征向量 $\boldsymbol{k}_t\in\mathbb{R}^{d_k}$，如式(3)所示：

$$\boldsymbol{k}_t=\frac{\boldsymbol{W}_k\boldsymbol{v}_t}{|\boldsymbol{k}_t|} \tag{3}$$

对于习题的难度，本文选用该习题的学生作答错误率 r_t^a、提交作答错误率 r_t^s 和区分度 q_t 作为特征来进行评价。习题的学生作答错误率指答错该习题的学生在尝试解答该习题的学生中的占比；习题的提交作答错误率指判定结果为错误的提交记录在该习题的提交记录中的占比；习题的区分度表示一道习题对于学生的区分能力，具有良好区分度的习题，能够将不同水平的学生区分开。在具有良好区分度的习题上，能力强的学生能够得高分，而能力弱的学生只能得到低分。根据教育心理学理论，习题 e_t 的区分度 q_t 等于在解答过习题 e_t 的学生中，总体习题作答正确率(正确解答的习题在尝试解答的习题中的占比)最高的 27%学生与最低的 27%学生在习题 e_t 上的正确率之差。

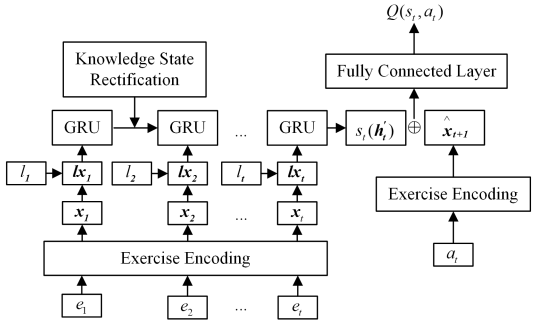

图 2 ERDP 的整体架构

Fig. 2 Overall architecture of ERDP

综上，本文使用上述向量与数值对习题进行编码，构建习题 e_t 的表征向量 x_t，如式(4)所示：

$$\boldsymbol{x}_t=\boldsymbol{g}_t\oplus\boldsymbol{c}_t\oplus\boldsymbol{k}_t\oplus[r_t^a,r_t^s,q_t] \tag{4}$$

其中，$\oplus$ 表示向量拼接操作。

4.2 知识状态矫正

为了解决 OJ 的判题方式可能导致对学生知识状态的评估存在偏差所带来的问题，本文设计了知识状态矫正模块，计算学生所提交程序与习题最优解的语法结构相似性，相似性越高则说明提交程序的实际正确性越高，根据实际正确性来矫正模型所估计的学生知识状态。

在获得习题 e_t 的表征向量 $\boldsymbol{x}_t$ 后，根据 OJ 给出的习题 e_t 的判定结果 l_t，使用式(5)处理 $\boldsymbol{x}_t$，得到考虑了 OJ 判定结果的习题向量 $\boldsymbol{lx}_t$。

$$\boldsymbol{lx}_t=\begin{cases}\boldsymbol{x}_t\oplus\boldsymbol{0}, & \text{if } l_t=1\\ \boldsymbol{0}\oplus\boldsymbol{x}_t, & \text{if } l_t=0\end{cases} \tag{5}$$

其中，$\boldsymbol{0}=[0,0,\cdots,0]$ 是与 $\boldsymbol{x}_t$ 同维度的“零”向量，$\oplus$ 表示向量拼接操作。

ERDP 引入 GRU 网络，通过学生的历史答题序列来建模学生在每一时刻的状态，具体步骤如式(6)、式(7)所示：

$$\boldsymbol{h}_t=GRU(\boldsymbol{h}'_{t-1},\boldsymbol{lx}_t;\Phi_{\text{GRU}}) \tag{6}$$

$$\boldsymbol{h}_t'=\text{sigmoid}[(\boldsymbol{g}_t)^{\mathrm{T}}\boldsymbol{M}\boldsymbol{g}_t^s+b_g]\cdot\boldsymbol{h}_t \tag{7}$$

其中，$\boldsymbol{h}_t'\in\mathbb{R}^{d_h}$ 和 $\boldsymbol{h}'_{t-1}\in\mathbb{R}^{d_h}$ 分别为 t 和 $t-1$ 时刻经过矫正后的学生知识状态向量，在本文中以 $\boldsymbol{h}_t'$ 表征 t 时刻的学生状态 s_t。式(6)中，在将 $\boldsymbol{h}'_{t-1}$ 和 $\boldsymbol{lx}_t$ 输入 GRU 网络后得到 t 时刻仅考虑了系统判定结果的学生知识状态向量 $\boldsymbol{h}_t\in\mathbb{R}^{d_h}$。式(7)则通过比较学生所提交程序与习题最优解的语法结构相似性对学生知识状态进行矫正，其中，$\boldsymbol{g}_t\in\mathbb{R}^{d_g}$ 和 $\boldsymbol{g}_t^s\in\mathbb{R}^{d_g}$ 分别为习题 e_t 的最优解和学生所提交程序的 EAST，并将其输入到 GGANN 中所得到的程序表征向量；$\boldsymbol{M}\in\mathbb{R}^{d_g\times d_g}$ 为可训练的参数矩阵；$b_g\in\mathbb{R}$ 为可训练的偏置项参数；$(\boldsymbol{g}_t)^{\mathrm{T}}\boldsymbol{M}\boldsymbol{g}_t^s+b_g$ 表示计算习题最优解与学生所提交程序的语法结构的相似性系数；sigmoid 函数则对相似性系数进行归一化，并使用归一化后的系数作为学生所提交程序的实际正确性系数来矫正仅考虑了系统判定结果的知识状态向量 $\boldsymbol{h}_t$，从而得到更为准确的学生知识状态向量 $\boldsymbol{h}_t'$。

需要说明的是，程序所使用的编程语言类型不同会导致具有相同功能的程序的 AST 存在差异性，在图 3 所示的分别用 Python 和 C++实现斐波那契数列求和的例子中，同一句代码“return $fib(n-1)+fib(n-2)$”在 Python 和 C++环境下生成的 AST 在结构上不相同，并且 Python 和 C++程序所生成 AST 的节点名称也不一致，如常量在 Python 和 C++的 AST 中的节点名称分别为“Literal”和“Constant”。若学生所提交程序与习题最优解所使用的编程语言类型不同，则 AST 也会不同，进而使模型估计的语法结构相似性和学生所提交程序的实际正确性不准确。因此，本文在训练习题推荐模型之前，使用 KalkiCode[40] 和 py14[41] 将学生提交记录中所有的 Java 和 Python 程序转为 C++程序，避免编程语言类型不同对估计语法结构相似性的影响。

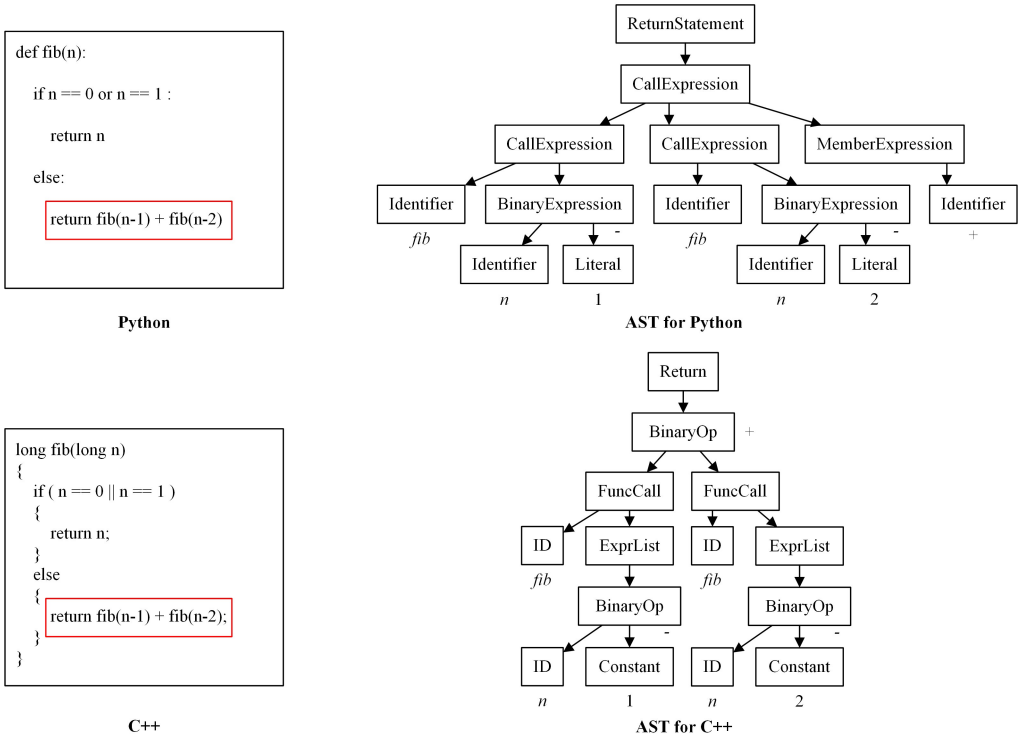

图 3 不同编程语言的 AST 示例

Fig. 3 AST examples of different programming languages

4.3 能力最大化导向奖励函数

为了使 ERDP 能够提供使学生程序设计能力增长程度最大的习题，本文利用强化学习技术并使用知识追踪模型作为学生模拟器，将学生模拟器模拟学生在状态 s_t 下，解答习题推荐模型所提供的习题 $\hat{e}_{t+1}$(即采取行动 a_t)前后的所有习题正确解答概率差的平均值作为奖励函数值，奖励函数 $R(s_t,a_t)$ 如式(8)所示：

$$R(s_t,a_t)=\frac{\sum_{e_m\in E}[p(l_m=1|s_{t+1})-p(l_m=1|s_t)]}{|E|} \tag{8}$$

其中，$p(l_m=1|s_{t+1})$ 和 $p(l_m=1|s_t)$ 分别为学生模拟器在 $t+1$ 时刻解答完 $\hat{e}_{t+1}$ 后和 t 时刻能够正确解答习题 e_m 的概率。若 $t+1$ 和 t 时刻的学生模拟器在所有习题上的正确解答概率差

越大，则说明学生在状态 s_t 下解答完习题推荐模型所推荐的习题 $\hat{e}_{t+1}$ 后，程序设计能力的增长程度也就越大。同时，习题推荐模型也能获得更大的奖励值，从而让模型学习到怎样的习题才能够最大程度地提升学生正确解答习题的概率，并将这样的习题推荐给学生，达到使学生程序设计能力增长最显著的目的。

4.4 模型训练

本文的目标为寻找向学生推荐习题的最优策略 π，使习题推荐模型基于任意状态 s_t 采取行动 a_t 所获得的累积推荐奖励值期望 $Q(s_t,a_t)$ 最大化，满足式(9)：

$$Q(s_t,a_t)=E_\pi\{R(s_t,a_t)+\gamma\max_{a_{t+1}}[Q(s_{t+1},a_{t+1})]\} \tag{9}$$

其中，$\gamma\in(0,1)$ 为惩罚因子，$\max_{a_{t+1}}[Q(s_{t+1},a_{t+1})]$ 代表的是在状态 s_{t+1} 下采取所有可能的行动后所获得的 Q 值中的最大值。

由于 OJ 中有大量的习题并且时常有新的习题被加入，计算并存储所有状态-行动对的 Q 值会消耗大量资源。为了解决这个问题，受 Deep Q-Network[42] 的启发，本文采用深度强化学习方案，使用 ERDP 模型的相关参数 θ 拟合 Q 值，如式(10)所示：

$$Q(s_t,a_t)\approx Q(s_t,a_t;\theta) \tag{10}$$

通过最小化如下损失函数来更新模型参数：

$$L(\theta)=\sum_u\sum_{t=1}^{T}\left[\frac{1}{2}(y-Q(s_t,a_t;\theta))^2\right] \tag{11}$$

$$y=R(s_t,a_t)+\gamma\max_{a_{t+1}}[Q(s_{t+1},a_{t+1};\theta')] \tag{12}$$

其中，y 是 t 时刻的目标；θ' 是先前目标网络的相关参数。$L(\theta)$ 的偏导数为：

$$\nabla L(\theta)=\sum_u\sum_{t=1}^{T}\{[Q(s_t,a_t;\theta)-y]\nabla Q(s_t,a_t;\theta)\} \tag{13}$$

当学生状态为 s_t 时，本文以 $Q(s_t,a_t;\theta)$ 值作为 ERDP 对于习题的 $\hat{e}_{t+1}$ 推荐程度，将候选习题按照推荐程度从高到低进行排序，从而生成推荐列表。$Q(s_t,a_t;\theta)$ 的计算方法如式(14)所示：

$$Q(s_t,a_t;\theta)=\mathrm{sigmoid}[\boldsymbol{W}_q^{\mathrm{T}}(\boldsymbol{h}_t'\oplus\hat{\boldsymbol{x}}_{t+1})+b_q] \tag{14}$$

其中，$\boldsymbol{h}_t'\in\mathbb{R}^{d_h}$ 为学生状态 s_t 的表征向量；$\hat{x}_{t+1}\in\mathbb{R}^{d_x}$ 为习题 $\hat{e}_{t+1}$ 的表征向量；$\boldsymbol{W}_q\in\mathbb{R}^{d_h+d_x}$ 为可训练参数向量；$b_q\in\mathbb{R}$ 为可训练的偏置项参数。

模型参数学习的步骤如算法 1 所示。

算法 1 ERDP 模型参数学习算法

输入：奖励函数 R，学习率 lr

输出：训练好的 ERDP 模型参数 θ

1. 随机初始化 ERDP 模型参数 θ
2. 训练知识追踪模型作为学生模拟器
3. $\theta'\leftarrow\theta$
4. for u in U do
5. for t←1 to T do
6. 获取 s_t，a_t 和 s_{t+1}
7. 通过学生模拟器计算得到 $R(s_t,a_t)$
8. if t<T do
9. $y\leftarrow R(s_t,a_t)+\gamma\max_{a_{t+1}}[Q(s_{t+1},a_{t+1};\theta')]$
10. else
11. $y\leftarrow R(s_t,a_t)$
12. end if
13. $\theta\leftarrow\theta-\mathrm{lr}[Q(s_t,a_t;\theta)-y]\nabla Q(s_t,a_t;\theta)$
14. $\theta'\leftarrow\theta$
15. end for
16. end for

5 实验

5.1 数据集

本文选用 CodeForces[29] 和 Libre[17] 两个业界知名 OJ 的数据作为实验数据集。两个数据集都包括了习题信息和用户提交记录，其中习题信息中的每条数据代表一道习题的信息，包含字段有：习题 id、标题、习题描述、知识点标签(可能缺失)；用户提交记录中的每条数据包含的字段有：提交 id、用户 id、习题 id、系统判定结果(正确/错误)、程序、编程语言类型、运行时间和提交时间戳。CodeForces 和 Libre 数据集的基本情况如表 1 所列，分别从每个数据集中随机抽取 80% 的学生用于训练推荐模型和学生模拟器，10% 的学生用于验证推荐模型和学生模拟器，10% 的学生用于测试推荐模型。

表 1 数据集的基本情况

Table 1 Basic information of datasets

数据集名称	习题数	用户数	提交记录数
CodeForces	7008	26787	1048575
Libre	2538	24657	1382200

5.2 评价指标

本文采用 Recall@K(Recall Rate of Top-K items)和 MRR@K(Mean Reciprocal Rank of Top-K items)作为评价指标来分析 OJ 习题推荐模型在下一交互习题预测任务上的性能，检验模型是否能满足学生的学习需求。

Recall@K 的详细定义如下：

$$Recall@K=\frac{1}{|U|}\sum_{u\in U}\frac{|R_u\bigcap T_u|}{|R_u|} \tag{15}$$

其中，$|U|$ 是学生的数量；R_u 表示测试集中与学生 u 相关的习题集；T_u 表示推荐给学生的前 K 个习题所构成的集合。

MRR@K 的详细定义如下：

$$MRR@K=\frac{1}{|U|}\sum_{u\in U}\frac{1}{rank_u} \tag{16}$$

其中，$rank_u$ 是学生的推荐列表的前 K 道习题中第一道正确推荐的习题所处的位置。

本文中，K 取为 10，对于每个真实输出随机采样了 100 个负例，真实输出和负例一同排序。

5.3 实验设置

本文使用 Tensorflow 实现所提方法，优化器选择 Adam，学生模拟器选用 SAINT 模型[16]，并且 SAINT 模型在 CodeForces 和 Libre 验证集上的准确率分别为 87.8% 和 84.6%，准确率较高，这说明 SAINT 模型可以准确地预测学生的答题结果，从而使 ERDP 能够精确地估计学生的程序设计能力

增长程度。在 ERDP 中，设置批处理大小 $bs=128$，学习率 $lr=0.001$，程序表征向量维度 $d_g=256$，词向量维度 $d_w=64$，习题描述表征向量维度 $d_c=128$，知识点表征向量维度 $d_k=16$，知识状态向量维度 $d_h=128$，惩罚因子 $\gamma=0.9$。

5.4 实验结果分析

5.4.1 基准方法对比实验结果

为了评估 ERDP 模型的性能，本文选用基于知识追踪的方法 DKT[14]，DKVMN[15]，DKT-CF[6]，SAINT[16]，基于协同过滤的方法 GRU4REC[43]，SASRec[44]，TiSASRec[45]，Locker[46] 和基于强化学习的方法 DRER[21]，HRL[22]，HELAR[23]作为基准方法，与 ERDP 进行对比实验，实验结果如表 2 所列。其中，DKT-CF，SAINT，TiSASRec，Locker，HELAR 为近两年提出的新方法，并且对于 DKT，DKVMN，DKT-CF 和 SAINT，本文向学生推荐正确解答概率最接近 0.5 的习题，以保证推荐给学生的习题既不会太难也不会太简单。

根据表 2 所列的实验结果可知，由于 ERDP 使用了习题的代码实现来挖掘习题之间的相关性，通过比较学生所提交程序和最优解来检验学生所提交程序的实际正确性，并以最大程度提升学生程序设计能力为推荐目的，因此其在 OJ 习题推荐任务上的性能要优于基准方法。其中，HRL 在基于注意力机制的序列推荐模型中引入了基于强化学习的用户画像修改器，去除了与目标习题相关度很小的噪音习题，因此其性能比仅考虑了自注意力机制（注意力机制的变体）的 SASRec 模型更佳，并且 HELAR 在 HRL 的基础上，使用动态循环机制解决了探索与利用的权衡问题，因此 HELAR 的推荐准确性高于 HRL。

表 2 不同模型的实验结果

Table 2 Experimental results of different models

模型	*Recall*@10		*MRR*@10	
	CodeForces	Libre	CodeForces	Libre
DKT	0.2173	0.1540	0.1721	0.1129
DKVMN	0.2208	0.1685	0.1787	0.1201
DKT-CF	0.2227	0.1723	0.1809	0.1303
SAINT	0.2598	0.2041	0.2120	0.1595
GRU4REC	0.2156	0.1554	0.1708	0.1138
SASRec	0.2712	0.2083	0.2236	0.1679
TiSASRec	0.3229	0.2571	0.2684	0.2161
Locker	0.3156	0.2498	0.2613	0.2085
DRER	0.2804	0.2209	0.2309	0.1724
HRL	0.2937	0.2330	0.2415	0.1806
HELAR	0.3018	0.2407	0.2476	0.1859
ERDP-s	0.3564	0.2937	0.3078	0.2576
ERDP-c	0.3713	0.3361	0.3211	0.2957
ERDP	**0.4176**	**0.3915**	**0.3602**	**0.3488**

本文也进行了消融实验，ERDP-s 和 ERDP-c 分别为未将习题最优解作为习题特征的 ERDP 和未矫正知识状态的 ERDP。通过观察表 2 可知，ERDP-s 在 CodeForces 和 Libre 数据集上进行测试时，*Recall*@10 相比 ERDP 分别降低了 0.0612 和 0.0978，*MRR*@10 降低了 0.0524 和 0.0912，这说明分析习题的代码实现能够辅助习题推荐模型挖掘习题之间的相关性，提升模型的性能。此外，模型在 Libre 上的性能下降程度大于 CodeForces。据分析，出现这种现象的原因为：在 Libre 数据集中，有 34.08% 的习题没有知识点标签，若习题推荐模型没有使用习题最优解作为特征，则较难挖掘这些没有知识点标签的习题之间的相关性，从而会降低习题推荐模型的性能，而在 CodeForces 数据集中，仅有 4.62% 的习题没有知识点标签，使得未使用习题最优解所造成的影响较小。ERDP-c 也在 CodeForces 和 Libres 上进行了测试，*Recall*@10 相比 ERDP 分别下降了 0.0463 和 0.0554，*MRR*@10 分别下降了 0.0391 和 0.0531，说明比较学生所提交程序和最优解相似性，有助于检验学生所提交程序的实际正确性，帮助模型更准确地评估学生的知识状态。

5.4.2 奖励机制有效性验证

为了验证在本文所设计的奖励机制驱动下，模型能取得更佳的推荐结果。本文修改 ERDP 模型为知识追踪模型 ERDP-KT，将式(14)中的 $Q(s_t,a_t;\theta)$ 值视为模型所预测的学生在状态 s_t 下能够正确解答习题 $\hat{e}_{t+1}$ 的概率 $p(l_{t+1}=1|s_t)$，简化表示为 p_{t+1}，如式(17)所示：

$$p_{t+1}=\text{sigmoid}[\boldsymbol{W}_q^{\mathrm{T}}(\boldsymbol{h}_t'\oplus\hat{\boldsymbol{x}}_{t+1})+b_q] \tag{17}$$

通过最小化预测概率与系统判定结果 l_t 的交叉熵损失 L 的方式训练模型参数，如式(18)所示：

$$L=\sum_u\sum_{t=1}^{T}[l_t\log p_t+(1-l_t)\log(1-p_t)] \tag{18}$$

ERDP-KT 在两个验证集上的准确率分别为 89.7% 和 88.2%，证明 ERDP-KT 模型可以准确地预测学生的答题结果。然后，使用 ERDP-KT 分别将正确解答概率最接近 0，0.1，0.2，0.3，0.4，0.5，0.6，0.7，0.8，0.9 和 1 的习题推荐给学生，在测试集上的实验结果如图 4 所示。

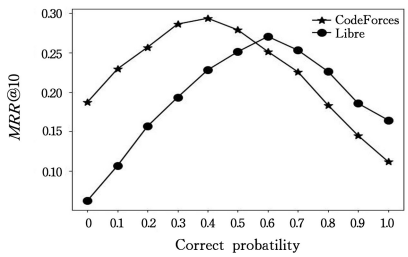

图 4 设置不同正确概率的实验结果

Fig. 4 Experimental results with different correct probability

根据图 4 的实验结果可知，向学生分别推荐正确解答概率最接近 0.4 和 0.6 的习题时，ERDP-KT 在 CodeForces 和 Libre 上的习题推荐性能达到最优，*MRR*@10 分别为 0.2935 和 0.2704，而 ERDP 所取得的 *MRR*@10 为 0.3602 和 0.3488，证明了本文所提出的奖励机制比向学生推荐指定正确解答概率的习题的方法更有效。

5.4.3 重要超参数影响分析

本文分析了 ERDP 涉及的两个重要超参数对习题推荐性能的影响，这两个超参数分别为惩罚因子 γ 和程序表征向量维度 d_g。

(1)惩罚因子。惩罚因子是习题推荐模型计算累积推荐

奖励值大小的重要参数。实验研究了不同的惩罚因子对模型性能的影响,该超参数的取值集合为{0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1}。图 5 给出了不同的惩罚因子下模型在 CodeForces 和 Libre 测试集上 MRR@10 指标的变化曲线,在实验时其他超参数设置不变。观察图 5 可知,惩罚因子对模型效果影响较大,CodeForces 和 Libre 在 0.9 处取得最佳表现。当惩罚因子过小时,模型会因忽略学生未来的程序设计能力增长程度而降低推荐效果,而当惩罚因子为 1 时,模型就会过分关注先前目标网络所估计的学生未来程序设计能力增长程度,而该估计值通常具有误差,从而降低了模型的性能。

(2)程序表征向量维度。实验研究了不同的程序表征向量维度对模型性能的影响,该超参数的取值集合为{16,32,64,128,256,512}。图 6 给出了模型在 CodeForces 和 Libre 测试集上 MRR@10 的变化曲线,可以看出,随着维度的增加,模型效果也提高,在 256 处达到最佳。当继续增加到 512 时,模型效果均有下降,原因是更大的维度反而会导致过拟合,降低了模型的泛化能力。

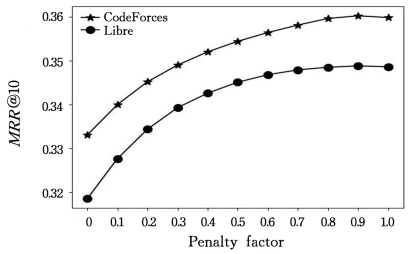

图 5 设置不同惩罚因子的实验结果

Fig. 5 Experimental results with different penalty factors

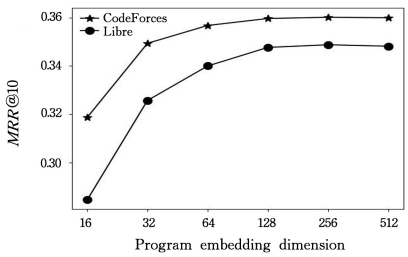

图 6 设置不同程序表征向量维度的实验结果

Fig. 6 Experimental results with different program embedding vector dimensions

结束语 本文充分关注现有习题推荐模型应用于 OJ 习题推荐任务时所面临的 3 个问题,即 OJ 习题知识点标签缺失与特有的命题风格使推荐模型难以挖掘习题之间的相关性;学生所提交程序的实际正确性与 OJ 系统判定结果不一致,导致模型所估计的学生知识状态产生偏差;现有模型较难提供可使学生程序设计能力得到最大增长的习题。

为了解决这些问题,本文提出了基于深度强化学习与程序分析的 OJ 习题推荐模型 ERDP,ERDP 分别通过如下方式解决这些问题:分析习题的代码实现来挖掘习题之间的相关性;比较学生所提交程序与习题最优解的相似性来检验学生所提交程序的实际正确性,使模型能够更准确地估计学生的知识状态;利用深度强化学习技术并使用知识追踪模型作为学生模拟器,将学生模拟器在解答习题推荐模型所提供的习题前后的所有习题正确解答概率差的平均值作为即时奖励,使模型学习到怎样的习题才能使学生程序设计能力增长程度最大,并将这样的习题推荐给学生。因为考虑并解决了这些问题,所以 ERDP 在 OJ 习题推荐任务上的性能要优于现有模型。

本文模型分析程序时使用了基于 AST 的方法,没有考虑程序的实际运行情况,无法强化程序中调用最频繁的核心代码块和弱化程序中不会被调用的代码块对程序分析结果的影响,分析结果可能不够准确;同时,本文模型忽略了 OJ 社交网络中包含的学生信息,如学生发表的习题解题思路和学生关注其他学生的解题思路,会使推荐模型忽略阅读解题思路对学生知识水平所产生的变化。在未来的工作中,会考虑将习题的测试用例输入程序,通过代码覆盖率统计工具获取程序的实际运行信息并加入程序分析模型来提升模型的性能;将 OJ 社交网络融入推荐模型[47]中,使模型能够更准确地估计学生的知识水平。此外,现有 OJ 中的解题数据量庞大,数据库的索引较大,容易造成空间浪费,因此,会考虑使用高效的学习型数据库索引推荐技术[48]来支持数据库的应用。

参 考 文 献

[1] GONG J,WANG S,WANG J,et al. Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view[C]// Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. Xi'an:ACM,2020:79-88.

[2] SHAO E,GUO S,PARDOS Z A. Degree Planning with PLANBERT:Multi-Semester Recommendation Using Future Courses of Interest[C]// Proceedings of the AAAI Conference on Artificial Intelligence. Vancouver:AAAI,2021:14920-14929.

[3] HUO Y,WONG D F,NI L M,et al. Knowledge modeling via contextualized representations for LSTM-based personalized exercise recommendation[J]. Information Sciences,2020,523:266-278.

[4] ZHAO J,BHATT S,THILLE C,et al. Interpretable personalized knowledge tracing and next learning activity recommendation[C]// Proceedings of the Seventh ACM Conference on Learning@ Scale. New York:ACM,2020:325-328.

[5] KANG W,ZHANG L,LI B,et al. Personalized exercise recommendation via implicit skills[C]// Proceedings of the ACM Turing Celebration Conference—China. Chengdu:ACM,2019:1-6.

[6] MA X R,XU Y,ZHU Q X. Personalized Exercises Recommendation Method Based on Deep Knowledge Tracing[J]. Journal of Chinese Computer Systems,2020,41(5):990-995.

[7] XIONG H J,SONG Y F,ZHANG P,et al. Personalized Question Recommendation Based on Autoencoder and Two-step Collaborative Filtering[J]. Computer Science,2019,46(11A):172-177.

[8] QI B, ZOU H X, WANG Y, et al. Questions Recommendation Based on Collaborative Filtering and Cognitive Diagnosis[J]. Computer Science, 2019, 46(11): 235-240.

[9] ZHU T Y, HUANG Z Y, CHEN E H, et al. Cognitive Diagnosis Based Personalized Question Recommendation[J]. Chinese Journal of Computers, 2017, 41(1): 176-191.

[10] FAN X. Item response theory and classical test theory: An empirical comparison of their item/person statistics[J]. Educational and Psychological Measurement, 1998, 58(3): 357-381.

[11] DE LA TORRE J. DINA model and parameter estimation: A didactic[J]. Journal of Educational and Behavioral Statistics, 2009, 34(1): 115-130.

[12] GAO W, LIU Q, HUANG Z, et al. Rcd: Relation map driven cognitive diagnosis for intelligent education systems[C]// Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. Montreal: ACM, 2021: 501-510.

[13] CORBETT A T, ANDERSON J R. Knowledge tracing: Modeling the acquisition of procedural knowledge[J]. User Modeling and User-adapted Interaction, 1994, 4(4): 253-278.

[14] PIECH C, BASSEN J, HUANG J, et al. Deep knowledge tracing [C]// Proceedings of the Advances in Neural Information Processing Systems. Quebec: MIT Press, 2015: 505-513.

[15] ZHANG J, SHI X, KING I, et al. Dynamic key-value memory networks for knowledge tracing[C]// Proceedings of the 26th International Conference on World Wide Web. Perth: ACM, 2017: 765-774.

[16] CHOI Y, LEE Y, CHO J, et al. Towards an appropriate query, key, and value computation for knowledge tracing[C]// Proceedings of the Seventh ACM Conference on Learning@ Scale. New York: ACM, 2020: 341-344.

[17] Renmin University of China. Libre OJ[EB/OL]. (2022-03-25) [2022-05-11]. https://loj.ac.

[18] WANG C Q. Luogu OJ[EB/OL]. (2022-05-11) [2022-05-11]. https://www.luogu.com.cn.

[19] LIU Q, HUANG Z, HUANG Z Y, et al. Finding similar exercises in online education systems[C]// Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. London: ACM, 2018: 1821-1830.

[20] TAN D W, ZHANG K F, LIU Y, et al. Program Classification Using Gated Graph Attention Neural Network[J]. Computer Engineering and Applications, 2020, 56(7): 176-183.

[21] HUANG Z, LIU Q, ZHAI C, et al. Exploring multi-objective exercise recommendations in online education systems[C]// Proceedings of the 28th ACM International Conference on Information and Knowledge Management. Beijing: ACM, 2019: 1261-1270.

[22] ZHANG J, HAO B, CHEN B, et al. Hierarchical reinforcement learning for course recommendation in MOOCs[C]// Proceedings of the AAAI Conference on Artificial Intelligence. Hawaii: AAAI, 2019: 435-442.

[23] LIN Y, LIN F, ZENG W, et al. Hierarchical reinforcement learning with dynamic recurrent mechanism for course recommendation[J]. Knowledge-Based Systems, 2022, 244: 108546.

[24] LIU Q, TONG S, LIU C, et al. Exploiting cognitive structure for adaptive learning[C]// Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Alaska: ACM, 2019: 627-635.

[25] ALLAMANIS M, PENG H, SUTTON C. A convolutional attention network for extreme summarization of source code [C]// International Conference on Machine Learning. New York: JMLR, 2016: 2091-2100.

[26] LU Y, LI G, ZHAO Z, et al. Learning to infer API mappings from API documents[C]// International Conference on Knowledge Science, Engineering and Management. Melbourne: Springer, 2017: 237-248.

[27] MOU L, LI G, ZHANG L, et al. Convolutional neural networks over tree structures for programming language processing[C]// Thirtieth AAAI Conference on Artificial Intelligence. Arizona: AAAI, 2016: 1287-1293.

[28] LI Y, TARLOW D, BROCKSCHMIDT M, et al. Gated graph sequence neural networks[C]// 4th International Conference on Learning Representations. Puerto Rico: OpenReview, 2016.

[29] Saratov State University. CodeForces[EB/OL]. (2022-05-11) [2022-05-11]. https://codeforces.com.

[30] Clemson University. United States of America Computing Olympiad[EB/OL]. (2022-05-11) [2022-05-11]. http://www.usaco.org.

[31] Peking University. Peking University OJ [EB/OL]. (2022-05-11) [2022-05-11]. http://poj.org.

[32] Zhejiang University. Zhejiang University OJ[EB/OL]. (2022-05-11) [2022-05-11]. http://acm.zju.edu.cn/onlinejudge.

[33] Hangzhou Dianzi University. Hangzhou Dianzi University OJ [EB/OL]. (2022-05-11) [2022-05-11]. http://acm.hdu.edu.cn.

[34] 10XRECRUIT. Kattis Problem Archive[EB/OL]. (2022-05-11) [2022-05-11]. https://open.kattis.com.

[35] ZHENG W, DU Q, FAN Y, et al. A personalized programming exercise recommendation algorithm based on knowledge structure tree[J]. Journal of Intelligent & Fuzzy Systems, 2022, 42(3): 2169-2180.

[36] YERA TOLEDO R, CABALLERO MOTA Y, MARTINEZ L. A recommender system for programming online judges using fuzzy information modeling[J]. Informatics, 2018, 5(2): 1-17.

[37] YERA R, MARTINEZ L. A recommendation approach for programming online judges supported by data preprocessing techniques[J]. Applied Intelligence, 2017, 47(2): 277-290.

[38] TOLEDO R Y, MOTA Y C. An e-learning collaborative filtering approach to suggest problems to solve in programming online judges[J]. International Journal of Distance Education Technologies(IJDET), 2014, 12(2): 51-65.

[39] MAO H Y. A Study on Knowledge Recommender Algorithm

Based on Time Transition[D]. Tianjin: Tianjin University, 2017.

[40] NANDPAWAN D. KalkiCode [EB/OL]. (2022-05-11) [2022-05-11]. https://kalkicode.com/ai/java-to-cplusplus-converter-online.

[41] LUKAS M, TIMOTHY C. Python to C++ 14 transpiler[EB/OL]. (2018-06-03) [2022-05-11]. https://github.com/lukasmartinelli/py14.

[42] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[J]. arXiv:1312.5602, 2013.

[43] HIDASI B, KARATZOGLOU A, BALTRUNAS L, et al. Session-based recommendations with recurrent neural networks [C]// Proceedings of 4th International Conference on Learning Representations. Puerto Rico: OpenReview, 2016: 1-10.

[44] KANG W C, MCAULEY J. Self-attentive sequential recommendation[C]// 2018 IEEE International Conference on Data Mining (ICDM). Singapore: IEEE Computer Society, 2018: 197-206.

[45] LI J, WANG Y, MCAULEY J. Time interval aware self-attention for sequential recommendation[C]// Proceedings of the 13th International Conference on Web Search and Data Mining. Texas: ACM, 2020: 322-330.

[46] HE Z, ZHAO H, LIN Z, et al. Locker: Locally Constrained Self-Attentive Sequential Recommendation[C]// Proceedings of the 30th ACM International Conference on Information & Knowledge Management. Queensland: ACM, 2021: 3088-3092.

[47] TANG J, CHEN Y, ZHOU M, et al. A Survey of Studies on Deep Learning Applications in POI Recommendation[J]. Computer Engineering, 2022, 48(1): 12-23, 42.

[48] YANG G, QIAO S, QU L. A survey of learning based database index advisor technology[J]. Journal of Chongqing University of Technology (Natural Science), 2022, 36(6): 189-199.

JIN Tiancheng, born in 1995, Ph.D candidate. His main research interests include educational data mining, recommender system and program analysis.

DOU Liang, born in 1980, Ph.D, associate professor, is a member of China Computer Federation. Her main research interests include software methods and AI for education.

(责任编辑:喻藜)