(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114092288 B
(45)授权公告日 2024.07.30

(21)申请号 202111395652.6

(22)申请日 2021.11.23

(65)同一申请的已公布的文献号
申请公布号 CN 114092288 A

(43)申请公布日 2022.02.25

(66)本国优先权数据
202011333752.1 2020.11.24 CN

(73)专利权人 北京航空航天大学
地址 100191 北京市海淀区学院路37号

(72)发明人 吴文峻 梁堉 武丽莎 韩勇

(74)专利代理机构 北京永创新实专利事务所 11121
专利代理师 易卜

(51)Int.Cl.
G06Q 50/20(2012.01)
G09B 19/00(2006.01)

(56)对比文件
CN 106095663 A,2016.11.09
CN 109408114 A,2019.03.01

审查员 李鑫宇

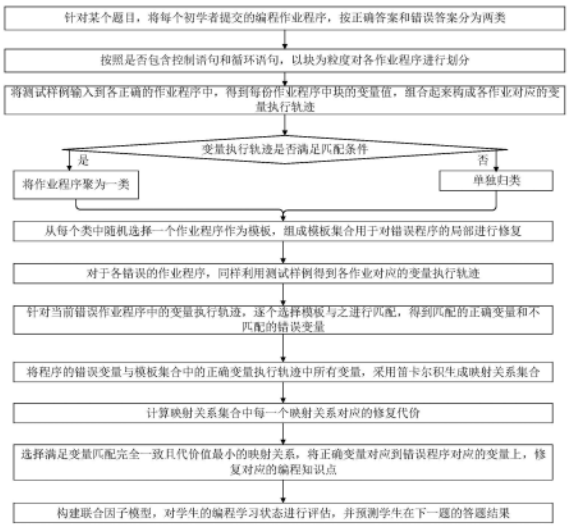
权利要求书2页 说明书12页 附图3页

(54)发明名称

一种面向编程初学者的个性化智能辅导方法

(57)摘要

本发明公开了一种面向编程初学者的个性化智能辅导方法,涉及教育智能化领域;首先,针对某个题目的各编程作业,以块为粒度进行划分;然后,将测试样例输入到各作业程序中,得到各自的变量执行轨迹;根据匹配条件,将正确的作业程序分为聚类,从每个类中随机选择一个模板;针对当前错误作业程序,逐个选择模板与之进行变量执行轨迹的匹配,采用笛卡尔积生成映射关系并计算每一个映射关系对应的修复代价;选择满足变量匹配完全一致且代价值最小的映射关系,将正确变量对应到错误程序对应的变量上,修复对应的知识点,完成最终的修复反馈生成。最后,构建联合因子模型,实现对学生的编程学习状态进行评估;本发明提高了编程作业的修复率。



CN 114092288 B

1.一种面向编程初学者的个性化智能辅导方法,其特征在于,具体步骤为;

首先,针对某个题目,将每个初学者提交的编程作业程序,按正确答案和错误答案分为两类;将每个作业程序按照是否包含控制语句和循环语句,以块为粒度进行划分;

然后,将题目对应的测试样例输入到各作业程序中,得到每份作业程序中块的变量值,组合起来构成各作业对应的变量执行轨迹;

针对正确答案,逐个选择各自对应的变量执行轨迹,判断是否满足匹配条件,如果是,则将所有匹配的作业程序聚为一类,反之,不匹配的作业程序单独归类;从每个类中随机选择一个作业程序作为模板,组成模板集合用于对错误程序的局部进行修复;

针对错误答案,将当前错误作业程序S中的变量执行轨迹,与从模板集合中逐个选择出的正确的变量执行轨迹进行匹配,得到程序S中匹配的正确变量和不匹配的错误变量;将程序S的错误变量与模板集合中当前正确变量执行轨迹中的所有变量,采用笛卡尔积生成映射关系集合;计算每一个映射关系对应的修复代价;从所有修复代价中选择满足变量匹配完全一致且代价值最小的映射关系,将正确变量执行轨迹中的正确变量对应到错误程序S对应的变量上,使用正则表达式将修复操作对应到编程知识点上,完成最终的修复反馈生成;

所述修复代价是指:将错误代码片段a修改成b所需要执行的动作大小;

采用将程序片段转化为抽象语法树,基于树的结构计算树编辑距离Distance(a,b)作为修复代价,计算公式为:

$$Distance(a,b)=S_q+I_q+D_r$$

S_q为两个抽象语法树的节点修改次数,I_q为两个抽象语法树的节点插入操作次数,D_r为两个抽象语法树的节点删除操作次数;

最后,构建联合因子模型,实现对学生的编程学习状态进行评估,并预测学生在下一题的答题结果。

2.如权利要求1所述的一种面向编程初学者的个性化智能辅导方法,其特征在于,所述的以块为粒度对各作业程序进行划分的规则具体如下:

i.不存在循环语句和选择语句,为整体的一块,不做划分;存在选择语句,但不嵌套循环语句,选择语句作为整体的一块,不做划分,每个块表示为{L:0};

ii.单层循环语句,作为单独的块,表示为{L:1};

iii.如果选择语句中嵌套循环语句,整体作为单独的块,表示为{L:1};

iv.多层嵌套循环,每一层都是单独的块;循环块L1中嵌套循环块L2,表示为{L1:{L2:1}}。

3.如权利要求1所述的一种面向编程初学者的个性化智能辅导方法,其特征在于,所述的两份变量执行轨迹的匹配条件,需同时满足以下:

i.两份变量执行轨迹中块的数量是否相同,以及控制语句和循环结构语句是否相同;

ii.两份变量执行轨迹中变量的数量是否相同;

iii.两份变量执行轨迹是否一一对应。

4.如权利要求1所述的一种面向编程初学者的个性化智能辅导方法,其特征在于,所述的对正确答案进行聚类的过程如下:

首先、初始随机选择两个变量执行轨迹,判断是否满足匹配条件,如果是,将这两个匹

配的作业程序归为一类;否则,这两个对应的作业程序分别各自为一类;

然后,继续选择第三个变量执行轨迹,针对匹配的两个作业程序,将第三个变量执行轨迹与任选一个进行匹配,判断是否满足匹配条件,如果是,归为该类中,否则,第三个变量执行轨迹单独归为一类;

针对不匹配的两个作业程序,将第三个变量执行轨迹逐个与该两个进行匹配,重复判断是否满足匹配条件,将满足匹配的归为一类,否则,三个都各自成为一类;

依次选择下一个变量执行轨迹,重复上述过程,直至将所有作业的变量执行轨迹匹配完毕。

5.如权利要求1所述的一种面向编程初学者的个性化智能辅导方法,其特征在于,所述的联合因子模型是依据学生的答题表现,将所有知识点的掌握情况和难度线性相加,来预测学生答题概率;答题正确的概率p_{ij}的计算公式如下:

$$p_{ij}=P_r(Y_{ij}=1|\theta_i,\beta,\gamma)=\prod_{k=1}^{K}\left(\frac{e^{\theta_i+\beta_k+\gamma_k T_{ik}}}{1+e^{\theta_i+\beta_k+\gamma_k T_{ik}}}\right)^{q_{jk}c_{ik}}$$

Y_{ij}表示学生i答对题目j的概率;θ_i表示学生i在答题前对题目的掌握程度;β_k表示知识点k的轻松程度,γ_k表示知识点k的学习率,q_{jk}表示题目j是否包含知识点k,包含为1,不包含为0;c_{ik}表示学生i在知识点k上的实际表现,T_{ik}表示学生i在知识点k上的尝试次数,γ_k和T_{ik}相乘表示每尝试一次,学生对知识点k的掌握程度更高;K表示编程题目中总计的知识点个数;

通过对联合因子模型的参数θ_i,β,γ进行优化,学生的下一题输入优化后的联合因子模型中,进行该题的结果预测。

一种面向编程初学者的个性化智能辅导方法

技术领域

[0001] 本发明涉及教育智能化领域,具体是一种面向编程初学者的个性化智能辅导方法。

背景技术

[0002] 编程智能辅导是智慧教育中的重要内容,目的是帮助学生修复提交的错误程序,估计对编程知识点的掌握情况。

[0003] 当前,编程智能辅导领域存在以下问题:对初学者编写的小规模程序的自动修复支持不足,编程错误修复时间冗长且修复结果晦涩难懂,缺乏对编程学习状态预测的模型,现有的编程课程线上实践系统不足以提供个性化的反馈;上述问题导致编程智能辅导系统还有待完善。

[0004] 在传统的在线测评系统里,学生只能得到代码是否正确的结果,而错误代码的进一步修复工作需要学生独立完成或者老师来协助完成,人工修复需要大量的时间,效率极低;且编程学习过程中学生对自己的知识点掌握情况也缺乏基本认识,很难得到个性化的指导,亟需一种针对编程初学者的智能辅导系统,辅助学生对编程课程的学习,同时,能实时评测学生提交的代码,对于错误程序可以为学生提供修复意见以及错误相关知识点,学生阶段性学习后可以估计学生对编程知识点的学习程度。

[0005] 如果采用人工修复,学生自己修复错误会浪费大量时间,也会打击其编程学习积极性,而教师和学生的数量差异难以满足教师对学生错误的逐一修复。因此,实现针对特定课程的智能辅导系统已经成为现阶段智慧学习的重要内容。

发明内容

[0006] 本发明针对编程初学者的个性化智能辅导问题,为了克服现有技术的不足,搭建了一种面向编程初学者的个性化智能辅导方法,从收集编程课程的学习行为数据入手,对编程实践的正确提交进行匹配聚类,得到每个类的模板,对错误提交则使用自动修复工具返回反馈意见,同时,通过对学生的错误提交,建立学习状态预测模型对该学生新提交的结果进行评估预测。

[0007] 所述的面向编程初学者的个性化智能辅导方法,包括如下步骤:

[0008] 步骤一、针对某个题目,将每个编程初学者提交的编程作业程序,按正确答案和错误答案分为两类;

[0009] 步骤二、按照是否包含控制语句和循环语句,以块为粒度对各作业程序分别进行划分;

[0010] 划分的规则如下:

[0011] i.不存在循环语句和选择语句,为整体的一块,不做划分;存在选择语句,但不嵌套循环语句,选择语句作为整体的一块,不做划分,每个块表示为{L:0};

[0012] ii.单层循环语句,作为单独的块,表示为{L:1};

[0013] iii.如果选择语句中嵌套循环语句,整体作为单独的块,表示为{L:1};

[0014] iv.多层嵌套循环,每一层都是单独的块;循环块L1中嵌套循环块L2,表示为{L1:{L2:1}}。

[0015] 步骤三、将该题目对应的测试样例输入到各正确的作业程序中,得到每份作业程序中块的变量值,组合起来构成各作业对应的变量执行轨迹;

[0016] 测试样例中包括该题目正确的输入和输出;

[0017] 步骤四、逐个选择变量执行轨迹,判断是否满足匹配条件,如果是,则将所有匹配的作业程序聚为一类,反之,不匹配的作业程序单独归类;

[0018] 两份变量执行轨迹的匹配需同时满足以下条件:

[0019] i.两份变量执行轨迹中块的数量是否相同,以及控制语句和循环结构语句是否相同;

[0020] ii.两份变量执行轨迹中变量的数量是否相同;

[0021] iii.两份变量执行轨迹是否一一对应。

[0022] 具体为:

[0023] 首先、初始随机选择两个变量执行轨迹,判断是否满足匹配条件,如果是,将这两个匹配的作业程序归为一类;否则,这两个对应的作业程序分别各自为一类;

[0024] 然后,继续选择第三个变量执行轨迹,针对匹配的两个作业程序,将第三个变量执行轨迹与任选一个进行匹配,判断是否满足匹配条件,如果是,归为该类中,否则,第三个变量执行轨迹单独归为一类;

[0025] 针对不匹配的两个作业程序,将第三个变量执行轨迹逐个与该两个进行匹配,重复判断是否满足匹配条件,将满足匹配的归为一类,否则,三个都各自成为一类;

[0026] 依次选择下一个变量执行轨迹,重复上述过程,直至将所有作业的变量执行轨迹匹配完毕;

[0027] 步骤五、从每个类中随机选择一个作业程序作为模板,组成模板集合用于对错误程序的局部进行修复;

[0028] 步骤六、对于各错误的作业程序,同样利用测试样例得到各作业对应的变量执行轨迹;

[0029] 步骤七、针对当前错误作业程序S中的变量执行轨迹,与从模板集合中逐个选择出的正确的变量执行轨迹进行匹配,得到程序S中匹配的正确变量和不匹配的错误变量;

[0030] 正确变量指的是错误作业程序的变量执行轨迹中,与正确变量执行轨迹中匹配的变量,即错误作业程序中没有出错的变量;

[0031] 步骤八、将程序S的错误变量与模板集合中的正确变量执行轨迹中所有变量,采用笛卡尔积生成映射关系集合;

[0032] 步骤九、计算映射关系集合中每一个映射关系对应的修复代价。

[0033] 修复代价是指:将错误代码片段a修改成b所需要执行的动作大小;

[0034] 本发明采用将程序片段转化为抽象语法树,基于树的结构计算树编辑距离Distance(a,b)作为修复代价,计算公式为:

[0035] $Distance(a,b)=S_q+I_q+D_r$

[0036] S_q为两个抽象语法树的节点修改次数,I_q为两个抽象语法树的节点插入操作次数,

D_r为两个抽象语法树的节点删除操作次数；

[0037] 步骤十、从所有修复代价中选择满足变量匹配完全一致且代价值最小的映射关系，将正确变量执行轨迹中的正确变量对应到错误程序S对应的变量上，使用正则表达式将修复操作对应到编程知识点上，完成最终的修复反馈生成。

[0038] 步骤十一、构建联合因子模型，实现对学生的编程学习状态进行评估，并预测学生在下一题的答题结果；

[0039] 联合因子模型是依据学生的答题表现，将所有知识点的掌握情况和难度线性相加，来预测学生答题概率。

[0040] 答题正确的概率p_{ij}的计算公式如下：

[0041] $$p_{ij}=P_r(Y_{ij}=1|\theta_i,\beta,\gamma)=\prod_{k=1}^{K}\left(\frac{e^{\theta_i+\beta_k+\gamma_k T_{ik}}}{1+e^{\theta_i+\beta_k+\gamma_k T_{ik}}}\right)^{q_{jk}c_{ik}}$$

[0042] Y_{ij}表示学生i答对题目j的概率；θ_i表示学生i在答题前对题目的掌握程度；β_k表示知识点k的轻松程度，γ_k表示知识点k的学习率，q_{jk}表示题目j是否包含知识点k，包含为1，不包含为0；c_{ik}表示学生i在知识点k上的实际表现，T_{ik}表示学生i在知识点k上的尝试次数，γ_k和T_{ik}相乘表示每尝试一次，学生对知识点k的掌握程度更高；K表示编程题目中总计的知识点个数。

[0043] 通过对联合因子模型的参数θ_i，β，γ进行优化，学生的下一题输入优化后的联合因子模型中，进行该题的结果预测；

[0044] 本发明与现有技术相比，具有以下优势：

[0045] (1)一种面向编程初学者的个性化智能辅导方法，相比现有软件修复工具，本发明针对学生编程作业进行修复，可以生成对正确程序的聚类结果和错误程序的修复操作；该方法将程序分析粒度定位到块，利用变量的执行轨迹来进行匹配和修复，基于语义来修复，将修复重点定为算法而不是表达形式，提高了编程作业的修复率。

[0046] (2)一种面向编程初学者的个性化智能辅导方法，优化了生成局部修复的算法，减少得到修复代价的时间；使用树编辑距离计算修复代价，为不同场景的修复提供了多种选择；

[0047] (3)一种面向编程初学者的个性化智能辅导方法，现有的学习状态预测模型中在分析学生答题状态时，默认每个题目只包含一个知识点，而在编程实践中，每一个编程题目比较复杂，可能包含多个知识点，本发明引入对每个知识点的掌握情况参数，将学生答题情况反应到具体知识点上；

[0048] (4)一种面向编程初学者的个性化智能辅导方法，学生提交的代码比传统客观题目要包含更多地信息，而现有模型的观测变量只能是二元变量(答对，答错)，本发明中对编程实践的学习状态评估，引入了多知识点和代码信息。

附图说明

[0049] 图1为本发明一种面向编程初学者的个性化智能辅导方法的原理图；

[0050] 图2为本发明一种面向编程初学者的个性化智能辅导方法的流程图；

[0051] 图3为本发明一种面向编程初学者的个性化智能辅导方法的模块图；

[0052] 图4为本发明采用树编辑距离修复和向量距离修复的时间对比；

[0053] 图5为本发明相对修复代价与错误程序数量的直方图。

具体实施方式

[0054] 为了便于本领域普通技术人员理解和实施本发明,下面结合附图对本发明作进一步的详细和深入描述。

[0055] 智能辅导系统ITS是借助人工智能技术,让计算机扮演虚拟导师向学习者传授知识、提供学习指导的适应性学习支持系统。作为现代远程教育向智能化发展的产物,ITS结合了人工智能、计算机科学、教育学、行为科学和心理学等诸多学科的科学理论和技术方法,旨在通过人机交互的方式为学习者提供学习引导和帮助。智能辅导系统可以根据学生答题效果来判断他们对相应知识点的掌握水平,从而帮助学生生成个性化的学习路线,提出针对性的辅导意见。

[0056] 现有的智能辅导系统主要是应用在基础学科领域,对于编程课程的智能辅导系统仍然有所欠缺,更多的是简单的线上评测系统,存在以下几个缺陷:(1)不能对学生错误进行修复指导;(2)没有对学生学习过程的总结,学生无法得到自己对编程知识点的掌握情况。针对上述问题,本发明提出了一种面向编程初学者的个性化智能辅导方法,实现了对编程初学者的智能辅导;

[0057] 所述的面向编程初学者的个性化智能辅导方法,如图1所示,采集学生的编程数据,并进行校验后,存储;通过对学生提交的程序中正确的作业程序进行聚类,用于对错误的作业程序进行修复;通过学习状态预测模型进行训练,推断学生对编程知识点的能力,完成最终的反馈生成。

[0058] 如图2所示,包括如下步骤:

[0059] 步骤一、针对某个题目,将每个编程初学者提交的编程作业程序,按正确答案和错误答案分为两类;

[0060] 步骤二、按照是否包含控制语句和循环语句,以块为粒度对各作业程序分别进行划分;

[0061] 划分的规则如下:

[0062] i.不存在循环语句和选择语句,为整体的一块,不做划分;存在选择语句,但不嵌套循环语句,选择语句作为整体的一块,不做划分,每个块表示为{L:0};

[0063] ii.单层循环语句,作为单独的块,表示为{L:1};

[0064] iii.如果选择语句中嵌套循环语句,整体作为单独的块,表示为{L:1};

[0065] iv.多层嵌套循环,每一层都是单独的块;循环块L1中嵌套循环块L2,表示为{L1:{L2:1}}。

[0066] 步骤三、将该题目对应的测试样例输入到各正确的作业程序中,得到每份作业程序中块的变量值,组合起来构成各作业对应的变量执行轨迹;

[0067] 测试样例中包括该题目正确的输入和输出;

[0068] 步骤四、逐个选择变量执行轨迹,判断是否满足匹配条件,如果是,则将所有匹配的作业程序聚为一类,反之,不匹配的作业程序单独归类;

[0069] 两份变量执行轨迹的匹配需同时满足以下条件:

[0070] i.两份变量执行轨迹中块的数量是否相同,以及控制语句和循环结构语句是否相

同;

[0071] ii.两份变量执行轨迹中变量的数量是否相同;

[0072] iii.两份变量执行轨迹是否一一对应;即在变量运行轨迹集合中,对于程序中的每一个变量,在另一个程序中都有一个变量与其以同样的顺序采用相等的值。

[0073] 具体为:

[0074] 首先、初始随机选择两个变量执行轨迹,判断是否满足匹配条件,如果是,将这两个匹配的作业程序归为一类;否则,这两个对应的作业程序分别各自为一类;

[0075] 然后,继续选择第三个变量执行轨迹,针对匹配的两个作业程序,将第三个变量执行轨迹与任选一个进行匹配,判断是否满足匹配条件,如果是,归为该类中,否则,第三个变量执行轨迹单独归为一类;

[0076] 针对不匹配的两个作业程序,将第三个变量执行轨迹逐个与该两个进行匹配,重复判断是否满足匹配条件,将满足匹配的归为一类,否则,三个都各自成为一类;

[0077] 依次选择下一个变量执行轨迹,重复上述过程,直至将所有作业的变量执行轨迹匹配完毕;

[0078] 步骤五、从每个类中随机选择一个作业程序作为模板,组成模板集合用于对错误程序的局部进行修复;

[0079] 步骤六、对于各错误的作业程序,同样利用测试样例得到各作业对应的变量执行轨迹;

[0080] 步骤七、针对当前错误作业程序S中的变量执行轨迹,与从模板集合中逐个选择出的正确的变量执行轨迹进行匹配,得到程序S中匹配的正确变量和不匹配的错误变量;

[0081] 正确变量指的是错误作业程序的变量执行轨迹中,与正确变量执行轨迹中匹配的变量,即错误作业程序中没有出错的变量;

[0082] 步骤八、将程序S的错误变量与模板集合中的正确变量执行轨迹中所有变量,采用笛卡尔积生成映射关系集合;

[0083] 利用修复映射关系将变量对应到错误程序的变量上,得到需要修复的行号和具体操作。

[0084] 步骤九、计算映射关系集合中每一个映射关系对应的修复代价。

[0085] 修复代价是指:将错误代码片段a修改成b所需要执行的动作大小;用抽象语法树(Abstract Syntax Tree)得到,AST的变化用两种方式来计算修复代价,一种是基于树的结构计算树编辑距离Distance(a,b)作为代价,计算公式为:

[0086] $Distance(a,b)=S_q+I_q+D_r$

[0087] S_q为两个抽象语法树的节点修改次数,I_q为两个抽象语法树的节点插入操作次数,D_r为两个抽象语法树的节点删除操作次数;

[0088] 另一种考虑将AST转化为向量,用两个向量的差异性来表示修复代价:

[0089] $$\left[h_1^1,\ldots h_1^{b_1}\right],\left[h_2^1,\ldots h_2^{b_2}\right],\left[h_{L^{2^{L}-1}}^{1},\ldots h_{L^{2^{q}-1}}^{b_{L^{2^{q}-1}}}\right]$$

[0090] 步骤十、从所有修复代价中选择满足变量匹配完全一致且代价值最小的映射关系,将正确变量执行轨迹中的正确变量对应到错误程序S对应的变量上,使用正则表达式将修复操作对应到编程知识点上,完成最终的修复反馈生成。

[0091] 为了求得最终修复,除了满足变量匹配一致性的条件以外,要做到修复代价最小。因此考虑使用约束优化技术,以最小的修复成本选择修复集合的一个子集;首先,需要将这个问题转化为线性约束条件下的最优化问题,然后将其交给现成的ILP解算器得到最优结果。

[0092] 步骤十一、构建联合因子模型,实现对学生的编程学习状态进行评估,并预测学生在下一题的答题结果;

[0093] 修复的错误知识点可以视为学生没有掌握的知识点,作为学生学习过程的刻画,可以引入联合因子模型,提高模型的预测效果。

[0094] 待求参数与答题正确的概率p_{ij}的计算公式如下:

[0095] $$p_{ij}=P_r(Y_{ij}=1|\theta_i,\beta,\gamma)=\prod_{k=1}^{K}\left(\frac{e^{\theta_i+\beta_k+\gamma_k T_{ik}}}{1+e^{\theta_i+\beta_k+\gamma_k T_{ik}}}\right)^{q_{jk}c_{ik}}$$

[0096] Y_{ij}表示学生i答对题目j的概率;θ_i表示学生i的能力,即在答题前对题目的掌握程度,学生的能力和答对概率呈正相关,能力越强,答对题目的概率越高;β_k表示知识点k的轻松程度,与q_{jk}相乘来影响最后答题的概率,只有题目包含该知识点,知识点的轻松程度才会对答对结果有影响;γ_k表示知识点k的学习率,即学生在第j道题目没有掌握知识点k,但是通过学习,在第j+1道题目掌握了该知识点的概率;q_{jk}表示题目j是否包含知识点k,该项从Q-matrix中得到,是一个二元变量,题目包含知识点k,q_{jk}为1,不包含为0;T_{ik}表示学生i在知识点k上的尝试次数,γ_k和T_{ik}相乘表示每尝试一次,学生对知识点k的掌握程度更高;K表示Q-matrix中总计的知识点个数,最后的结果是将所有知识点的掌握情况和难度线性相加来预测学生答题概率。

[0097] Q-matrix表示了题目与知识点之间的关系,第m行第n列值为1,则表示题目m中包含知识点n,值为0则表示题目m中不包含知识点n。参数c_{ik}是针对编程教学引入的新变量,由编程工具直接给出,表示学生i在知识点k上的实际表现,如果编程工具已经给出对知识点的表现,出错则取值为1,用来强调对结果的影响,反之则取值为0,如果编程工具还未给结果则认为还未掌握知识点,相应地取值为1。

[0098] 通过对联合因子模型的参数θ_i,β,γ进行优化,学生的下一题输入优化后的联合因子模型中,进行该题的结果预测;具体优化过程为:

[0099] (1)参数初始化:初始化θ_i^0,β_k^0,γ_k^0,(i=1,2,…,N;k=1,2,…,K)令m=0,确定最大迭代次数G,初始化迭代器g=0,计算$L(\theta_i^0,\beta_k^0,\gamma_k^0)$;

[0100] (2)令$W=L(\theta_i^0,\beta_k^0,\gamma_k^0)$,计算似然函数$L(\theta_i^g,\beta_k^g,\gamma_k^g)$,若满足g>G或$\left\|W-L(\theta_i^g,\beta_k^g,\gamma_k^g)\right\|<\varepsilon$且g≠0,停止运算,否则令$W=L(\theta_i^g,\beta_k^g,\gamma_k^g)$;

[0101] 在M个学生回答N个题目的情景下,上述情景的对数似然函数为:

[0102] $$\begin{aligned}L(\theta_i,\beta_k,\gamma_k)&=\sum_{j=1}^{M}\sum_{i=1}^{N}Y_{ij}\log\frac{\pi_{ij}}{1-\pi_{ij}}+\log(1-\pi_{ij})\\&=\sum_{j=1}^{M}\sum_{i=1}^{N}Y_{ij}\log w_{ij}-\log(1+w_{ij})\end{aligned}$$

[0103] 其中,

[0104] $$\exp\left\{\theta_i+\sum_{k=1}^{K}q_{jk}\beta_k+\sum_{k}^{K}q_{jk}\gamma_k T_{ik}\right\}=w_{ij}$$

[0105] (3)令m＝0，$\gamma_k^m=\gamma_k^g$，保持γ_k^m不变，计算L对θ_i^m，β_k^m的梯度$\nabla L(\theta_i^m)$，$\nabla L(\beta_k^m)$；

[0106] 如果满足$\|\nabla L(\theta_i^m)\|<\varepsilon$且$\|\nabla L(\beta_k^m)\|<\varepsilon$，停止迭代，$\theta_i^*=\theta_i^m$，$\beta_k^*=\beta_k^m$，；否则取步长$\lambda^m$，使得：

[0107] $$L(\theta_i^m+\lambda^m(-\nabla L(\theta_i^m)),\beta_k^m+\lambda^m(-\nabla L(\beta_k^m)),\gamma_k^m)=\min_{\lambda\geq 0}L(\theta_i^m+\lambda^m(-\nabla L(\theta_i^m)),\beta_k^m+\lambda^m(-\nabla L(\beta_k^m)),\gamma_k^m)$$

[0108] (4)令$\theta_i^{m+1}=\theta^m+\lambda^m\left(-\nabla L(\theta_i^m)\right),\beta_k^{m+1}=\beta_i^m+\lambda^m\left(-\nabla L(\beta_k^m)\right)$，计算似然函数$L(\theta_i^{m+1},\beta_k^{m+1},\gamma_k^m)$；

[0109] 如果满足$\left\|J(\theta_i^{m+1},\beta_k^{m+1},\gamma_k^m)-J(\theta_i^m,\beta_k^m,\gamma_k^m)\right\|<\varepsilon$或$\|\theta_i^{m+1}-\theta_i^m\|<\varepsilon$且$\|\beta_k^{m+1}-\beta_k^m\|<\varepsilon$，停止迭代，令$\theta_i^*=\theta_i^{m+1}$，$\beta_k^*=\beta_k^{m+1}$，更新m＝m+1，转(3)；

[0110] (5)保持θ_i^{m+1}，β_k^{m+1}不变，$\theta_i^g=\theta_i^{m+1}$，$\beta_k^g=\beta_k^{m+1}$，m＝0，计算L对$\gamma_k^m$的梯度$\nabla L(\gamma_k^m)$，如果$\|\nabla L(\gamma_k^m)\|<\varepsilon$，停止迭代，令$\gamma_k^*=\gamma_k^m$；否则选取步长$\lambda^m$，使

[0111] $$L(\theta_i^g+\lambda^m(-\nabla L(\theta_i^m)),\beta_k^g+\lambda^m(-\nabla L(\beta_k^g)),\gamma_k^m)=\min_{\lambda\geq 0}L(\theta_i^g+\lambda^m(-\nabla L(\theta_i^g)),\beta_k^g+\lambda^m(-\nabla L(\beta_k^g)),\gamma_k^m)$$

$$\gamma_k^{m+1}=\gamma_k^m+\lambda^m(-\nabla L(\gamma_k^m))$$

[0112] 计算$L(\theta_i^g,\beta_k^g,\gamma_k^m)$；

[0113] (6)如果$\left\|L(\theta_i^g,\beta_k^g,\gamma_k^{m+1})-L(\theta_i^g,\beta_k^g,\gamma_k^m)\right\|<\varepsilon$或$\|\gamma_k^{m+1}-\gamma_k^m\|<\varepsilon$，停止迭代；令$\gamma_k^*=\gamma_k^{m+1}$，g＝g+1，转(2)；否则m＝g+1，转(3)；

[0114] 直至实现梯度下降后，停止迭代；

[0115] 所述的面向编程初学者的个性化智能辅导方法，基于如图3所示的三个模块，包括编程数据收集存储模块，错误自动修复模块和学习效果评估模块；

[0116] 数据通过两个渠道收集，一部分通过学生注册登录测评模块，在线提交作业程序发送数据到存储模块，该测评模块基于青岛大学开源在线评测平台建立，另一部分来自头歌实践教学平台(提供编程课程在线实训的网站)，通过与头歌平台对接，以特定的数据格式定期接收学生回答客观题目以及完成实训的相关数据，确认过数据格式正确后存储到模块中。

[0117] 错误自动修复模块从数据收集存储模块中获取作业程序的数据，分为正确回答和错误回答。对于正确作业的程序进行聚类并选出每一类的模板，以树状图的形式将多种解题方法提示给学生；错误作业的程序则根据模板来进行修复，最终将修改提示和错误知识点反馈给学生。

[0118] 学习效果评估模块主要对客观题目和实训题目的答题数据进行分析，最终将评估结果用可视化图表来表示。

[0119] 其中，错误自动修复模块中程序匹配需要满足对多样性表达的支持，如果算法目

标一致但是在语法表达有差异,倾向于认为是一类,这样会得到最少的类别,在后期修复时可以与较少的模板比较。因此,需要首先定义程序是否属于一类的标准;一般来说,程序分析中的粒度有函数、语句和变量;如果用函数作为编程作业的分析粒度,算法的循环、控制结构难以掌握,会忽视过程中的一些信息,从而丢失程序实现的细节。而如果用语句和变量作为分析粒度,循环结构的执行会大大增加分析的复杂度,难以满足修复的时效性。因此考虑将程序划分为块,从而以块作为粒度来进行程序分析。程序匹配修复的具体步骤如下:

[0120] 步骤1.首先对程序进行匹配。对于给定程序P、Q和多组输入I,先来对两个程序来进行分块,观察两个程序的块是否一一对应,不能达到一一对应就直接退出;否则,按照刚刚划分的块来得到P、Q的变量执行轨迹;同时取P和Q变量的笛卡儿积为H;对于P、Q变量笛卡儿积中每个变量对(v2,v1),查看这一对变量的执行轨迹是否相同,如果不相同,就将其从H中剔除。

[0121] 这个算法得到了一系列轨迹匹配的变量对(v2,v1),但并不能判断两个程序的变量是否一一对应,因此可以借助图的算法来进行判断,将变量看成节点,轨迹一样的变量之间连成边。也就是说构成的图中,所有点被分为两组,一组的点只能连接另一组的点,探寻变量组成的二分图上是否存在完美匹配,用匈牙利算法即可求解程序P和Q是否匹配。

[0122] 这种对轨迹对应关系的判断方法,更加看重程序的动态执行,能够对代码风格或者表达不同但是算法思想一致的代码进行聚类,为程序修复奠定基础。

[0123] 步骤2.程序聚类要满足可扩展性,正确样本是动态增加的,如果每次有新的正确提交都需要将历史正确样本进行重新聚类,会大大降低算法效率。考虑将聚类算法设置为增量式,不需要每次都重新启动算法。

[0124] 首先输入正确代码组成的程序集合$P=\{p_1,p_2,p_3,\cdots,p_n\}$,初始化聚类结果$C=\{\}$,模板类个数$k=0$,循环遍历代码集合中P中每个程序$p_i$,执行以下过程:

[0125] i.如果聚类结果C为空,令$c_1=p_i$,$C=\{c_1:[p_i]\}$,$k=k+1$;

[0126] ii.反之则循环遍历聚类结果C中模板程序c_i,执行以下过程:

[0127] a)如果c_i和p_i匹配,则$C[c_i]+=[p_i]$,

[0128] b)反之则新建模板$c_{k+1}=p_i$,令$C[c_{k+1}]=[p_i]$,$k=k+1$;

[0129] iii.返回聚类结果C;

[0130] 步骤3.程序修复需要参考的是聚类后的结果,由于聚类后属于同一类的程序算法一致,因此修复时只需要将错误程序与每个类的模板比较,这样可以减少算法的复杂度,提升算法效率。

[0131] 考虑到在程序匹配中已经确定了程序分析的粒度为块,因此在修复中采用从局部到整体的方法,先将错误代码进行分块,为块中的表达式生成一组局部修复,所有块的结果组成修复集合。

[0132] 3.1确定修复代价。

[0133] diff工具通过添加、删除代码行这两个基本变化操作,反映出代码行的差异,但是其不能反映出代码语法结构的差异。为了能够量化修复操作的大小,将程序表达式转化为抽象语法树的形式,树上的每个节点表示了程序语言的某个结构,这样表示便于隐藏语法细节。因此,对于表达式的增加、删除、修改可以用相应的抽象语法树AST(Abstract Syntax Tree)变化得到。

[0134] 而AST的变化可以用两种方式来计算代价：

[0135] 树编辑距离类似于字符串编辑距离，主要是求从树a变到树b所需要的最小操作代价。从a到b可能需要修改节点、插入节点或者删除节点，因此需要为每一个单位操作赋予权值1，如果将修改次数记作S_q，删除操作次数记作D_r，插入操作次数记作I_q；则树编辑距离就是这些操作次数的总和，即Distance(a,b)＝S_q+I_q+D_r；

[0136] 对AST向量化，要尽可能多地保留各个节点的位置信息。抽象语法树本质上是一棵二叉树，为了转化成向量，考虑拆分成了一系列原子树来捕获树的结构信息。具体来说，先得到树的节点种类数L和树的高度q，那么相应的原子树最多有L^{2^q-1}种。用b_i来计算第i个原子树的数量，那么表示树的形态的向量为$[b_1,\dots,b_{L^{2^q-1}}]$；为了增加位置信息，将形态向量中的每一个元素扩展为b_i维，分别表示这个原子树所在AST中的高度，AST的位置形态向量为：

[0137] $$\left[h_1^1,\dots h_1^{b_1}\right],\left[h_2^1,\dots h_2^{b_2}\right],\left[h_{L^{2^{L}-1}}^1,\dots h_{L^{2^q-1}}^{b_{L^{2^q-1}}}\right]$$

[0138] 这样的表示方法可以既保留AST的形态，又保留它的位置信息。将AST转化为向量后，最终修复代价J为：

[0139] $$J=\sqrt{\sum_{i=1}^{|L|^q-1}\left\|sort(h_{b_i},\phi)-sort(h_{b_i}^{'},\phi)\right\|_2^2}$$

[0140] 3.2局部修复。

[0141] i.输入错误程序p，模板程序c和需要测试的输入I；

[0142] ii.如果p和c的分块结果不同，则返回“修复失败”并退出；

[0143] iii.输入I，得到c的轨迹集合ΓC＝{[P_C](ρ)|ρ∈I}；

[0144] iv.输入I，得到p的轨迹集合ΓP＝{[P_p](ρ)|ρ∈I}；

[0145] v.循环遍历要修复的程序p中所有块中的所有变量(l,v2)；

[0146] vi.初始化局部修复集合$LR(\ell,v_2)=\emptyset$；

[0147] vii.得到v2的赋值表达式$e_{impl}=\mathcal{U}_{impl}(\ell,v_2)$；

[0148] viii.循环遍历模板程序c中所有变量v1，得到v1的赋值表达式$e_C=\mathcal{U}_{P_C}(\ell,v_1)$；

[0149] ix.构造映射关系集合W，每个映射关系w，w∈W都将e_{impl}的所有变量映射到e_C的所有变量，保证v1经过映射得到v2；

[0150] x.遍历集合W中的w:；

[0151] a)e_C中的变量经过w替换成为e_{repair}；

[0152] b)如果e_{repair}和e_{impl}匹配，则将(w,0)加入LR中；

[0153] c)否则，计算e_{repair}到e_{impl}修复代价cost，将(w,cost)加入到LR中；

[0154] 3.3求解最终修复。

[0155] 整个程序修复有很多选择，完整的修复就是找到局部修复的一个具有一致性的子集。也就是说，在这个集合中，所有的变量映射之间不产生冲突，错误程序中的每个变量仅对应于模板程序中的唯一变量且不存在两个变量对应同一个的关系。

[0156] i.首先确定决策变量，用V_{P_C}来表示正确程序的变量集合，V_{impl}来表示错误程序，LR

(l,v)表示局部修复集合,则决策变量的定义域D为:

[0157] $$D=\{x_{v_1v_2}|v_1\in V_{P_C},v_2\in V_{impl}\}\cup\{x_r|r\in LR(\ell,v),(\ell,v)\in L\times V_{impl}\}$$

[0158] ii.而根据上面的四个条件可以列出要满足的可行域:

[0159] $$\left(\sum_{v_2\in V_{impl}}x_{v_1v_2}\right)=1,\quad v_1\in V_{P_C}$$

$$\left(\sum_{v_1\in V_{P_C}}x_{v_1v_2}\right)=1,\quad v_2\in V_{impl}$$

[0160] $$\left(\sum_{r\in LR(\ell,v)}x_r\right)=1,\quad (\ell,v)\in L\times V_{impl}$$

$$-x_r+x_{u_1u_2}\geq 0,\quad r=(\omega,-)\in LR,\quad \omega(u_2)=u_1$$

[0161] iii.最后要得到的目标函数如下:

[0162] $$O=min\left(\sum_{r\in LR(\ell,v)}cost(r)x_r\right)$$

[0163] iv.求解最终修复结果的问题转化为0-1线性约束条件下的最优化问题,用lpsolve即可得到最优解。

[0164] 步骤4.针对得到的修复结果,利用修复映射关系将变量对应到错误程序的变量上,并得到需要修复的行号和具体操作。根据操作的语句,使用正则表达式的方法基于一定规则将修复操作对应到知识点上,完成最终的反馈生成。

[0165] 编程学习预测模型步骤如下,利用针对编程的联合因子模型来得到学生对编程知识点的掌握情况:

[0166] (1)影响答题表现的因素主要有两类,一类是题目相关,包括题目难度,包含的知识点以及每个知识点的易学程度;另一类是学生特性,包括学生答题前的技能掌握程度、学习速率以及一些误差因素,例如猜对概率和误答概率等等。

[0167] 而题目中每个知识点的难易程度有所不同,学习速率也有所不同,设置知识点的难度和学习率作为参数。

[0168] 反馈模块的具体实现步骤如下所示:

[0169] (1)使用django模块搭建编程错误修复系统,展示学生正确题目的多种解法和错误题目的修复方法提示;

[0170] (2)使用Echarts等JavaScript图表库展示学生能力的变化;

[0171] (3)将可视化结果反馈给教师或学生。

[0172] 结果表示:

[0173] 下面给出一组收集的学生编程实例对本发明的方法作进一步详述。某课程有九个章节的题目,表1描述了这部分数据的基本情况。

[0174] 表1:课程各章节标题及学生和相关知识点

[0175]

题目	标题	学生	提交	相关知识点
C-1	字符串拼接	513	807	VA, OP, ST, EX, PR
C-2	挑战列表	289	1002	CO, VA, OP, EX, LI, PR
C-3	排序	318	391	VA, EX, TU, PR

[0176]

C-4	改变字典元素	315	379	CO, VA, ST, EX, DI, PR
C-5	计算数值	392	602	CO, VA, OP, EX, PR
C-6	数据分类	118	207	CO, VA, OP, EX, PD
C-7	计算阶乘	207	520	CO, VA, OP, ST, EX, CD, PR
C-8	数据遍历	693	1293	CO, VA, OP, ST, EX, CD, LO, PR
C-9	综合挑战	645	1684	CO, VA, OP, ST, EX, LI, CD, LO, PR

[0177] 在表1中,该课程包含6个小节,每个小节有不同数量的题目,最后加起来总共是9个题目,有1030个学生参与答题,每个同学做的题目不是完全一致,每道题目的提交次数也各有差异,这些提交加起来是6885次,其中针对同一个练习最多的提交次数是23次。所有的题目涉及到11个知识点,包括常量(CO),变量(VA),运算符(OP),字符串(ST),表达式(EX),列表(LI),元组(TU),字典(DI),条件(CD),循环(LO)和打印语句(PR)。每个题目包含的知识点数目也不同,C-9是一个综合问题,包含了所有的知识点。

[0178] 在该数据集上进行本发明的实验步骤如下:

[0179] 1、进行程序匹配实验。

[0180] 使用编程自动修复工具对同一个题目的正确提交使用匹配算法后,对匹配的两个程序进一步分析,得到了同一个算法的不同表达方式,如表2所示,表达式集合1和表达式集合2中的表达式针对同一输入,变量的执行轨迹都一致。

[0181] 表2:匹配的表达式

[0182]

表达式集合 1	表达式集合 2
$1.result \mathrel{+}= [float(poly[e] * e),]$ $2.if(e == 0): result.append(0.0)$ $else: result.append(float(poly[e] * e))$ $3.result.append(1.0 * poly[e] * e)$ $4.result.append(float(e * poly[e]))$ $5.result \mathrel{+}= [e * poly[e]]$	$1.if(len(result) == 0):$ $return\ [0.0] else: return\ result$ $2.if(len(result) > 0):$ $return\ result$ $else: return\ [0.0]$ $3.return\ result\ or\ [0.0]$

[0183] 对实验课程中9个题目的正确提交进行聚类,最终得到结果如表3所示,可以看出来这9个题目聚类的个数在2-5之间,题目越复杂,聚类的数目越多。题目C-9的聚类结果是5个。

[0184] 表3:聚类结果

[0185]

题目	标题	平均代码行数	聚类个数
C-1	字符串拼接	5	2

[0186]

C-2	挑战列表	11	3
C-3	排序	11	3
C-4	改变字典元素	12	4
C-5	计算数值	18	5
C-6	数据分类	14	3
C-7	计算阶乘	12	3
C-8	数据遍历	12	4
C-9	综合挑战	19	5

[0187] 2、进行程序自动修复实验

[0188] 在该课程数据集上测试两种不同修复代价表示方式的修复率和修复时间,即树编辑距离和向量距离表示法的差异性,结果表明发现修复率没有区别,如图4所示,展示了使用树编辑距离作为修复代价和使用向量距离作为修复代价的修复算法所用时间的对比,可以看出在题目简单时,例如C-1,C-2中,两者的修复时间差异不明显,而随着题目复杂度逐渐增加,用向量距离作为修复代价用的时间要明显低于树编辑距离作为修复代价的修复算法。

[0189] 为了衡量修复操作的复杂性,以向量距离作为修复代价,将修复代价除以错误程序的总向量值得到相对修复代价,对其进行归一化处理,结果用直方图来表示。如图5所示,在直方图中,横轴表示相对修复代价值,纵轴表示错误程序数目,如果错误程序为空,AST数量为0,则相对修复代价大于1.0,甚至可能无穷大。在直方图中,68%的修复相对修复代价小于0.2,25%的修复相对修复代价小于0.1。表明修复工具完成的修复大多都是简单修复,实用性较强。

[0190] 3、编程学习状态预测

[0191] 在本发明收集的数据集上采取十折交叉验证完成了实验,并计算多个模型预测下的AUC和ACC,结果如表4所示,本发明所提出的模型CFMP在准确率和曲线下面积(AUC)方面表现最佳,CFMP模型的AUC比CFM模型提高了13.2%,表明了扩展的学习状态评估模型在编程场景下预测结果更加准确。

[0192] 表4预测结果

[0193]

指标	Accuracy	AUC
CFM	0.675	0.736
CFMP	0.770	0.833
DKT	0.608	0.672
DKTP	0.682	0.715

[0194] 本发明未详细阐述部分属于本领域公知技术。

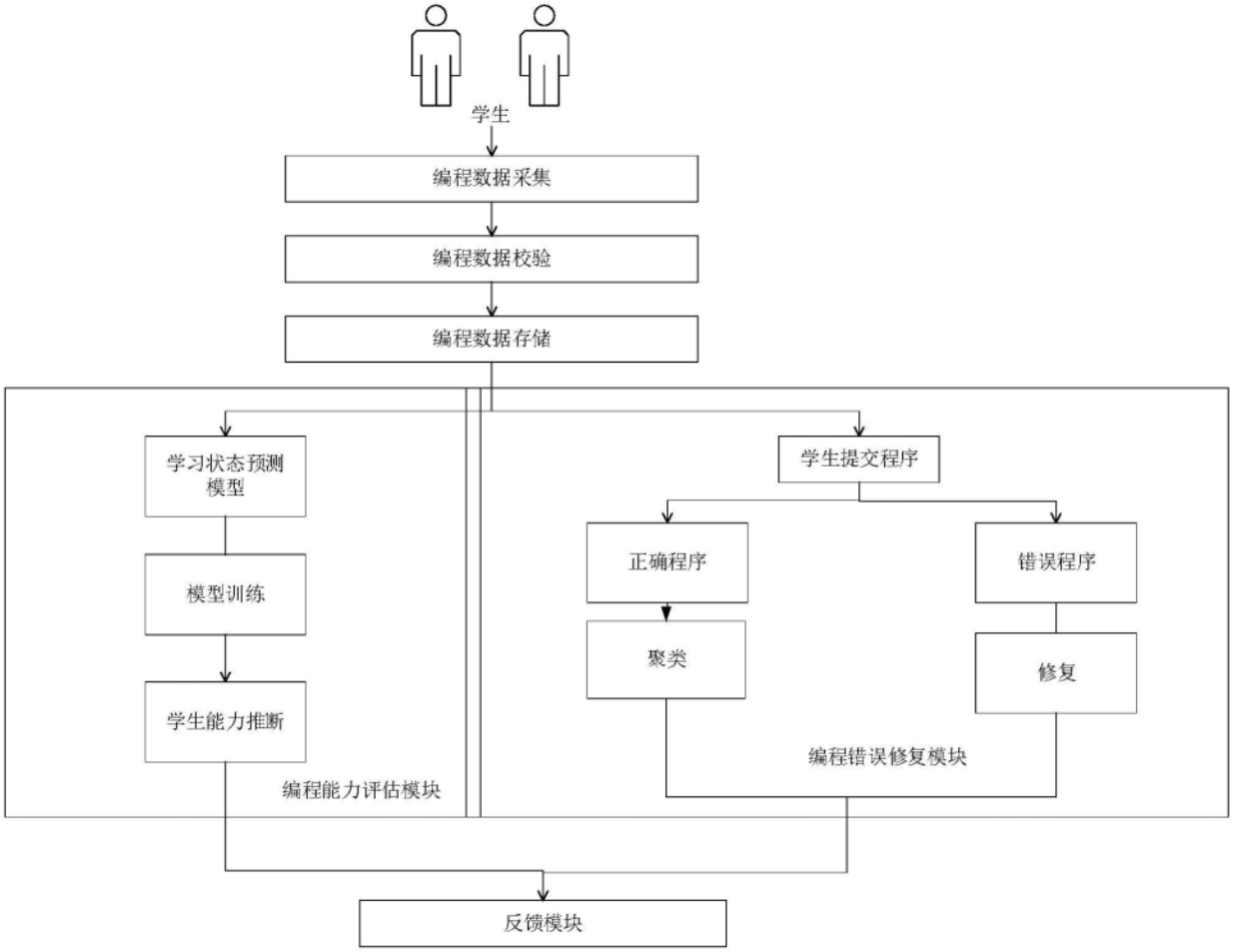

学生
编程数据采集
编程数据校验
编程数据存储
学习状态预测模型
模型训练
学生能力推断
编程能力评估模块
学生提交程序
正确程序
聚类
错误程序
修复
编程错误修复模块
反馈模块

图1

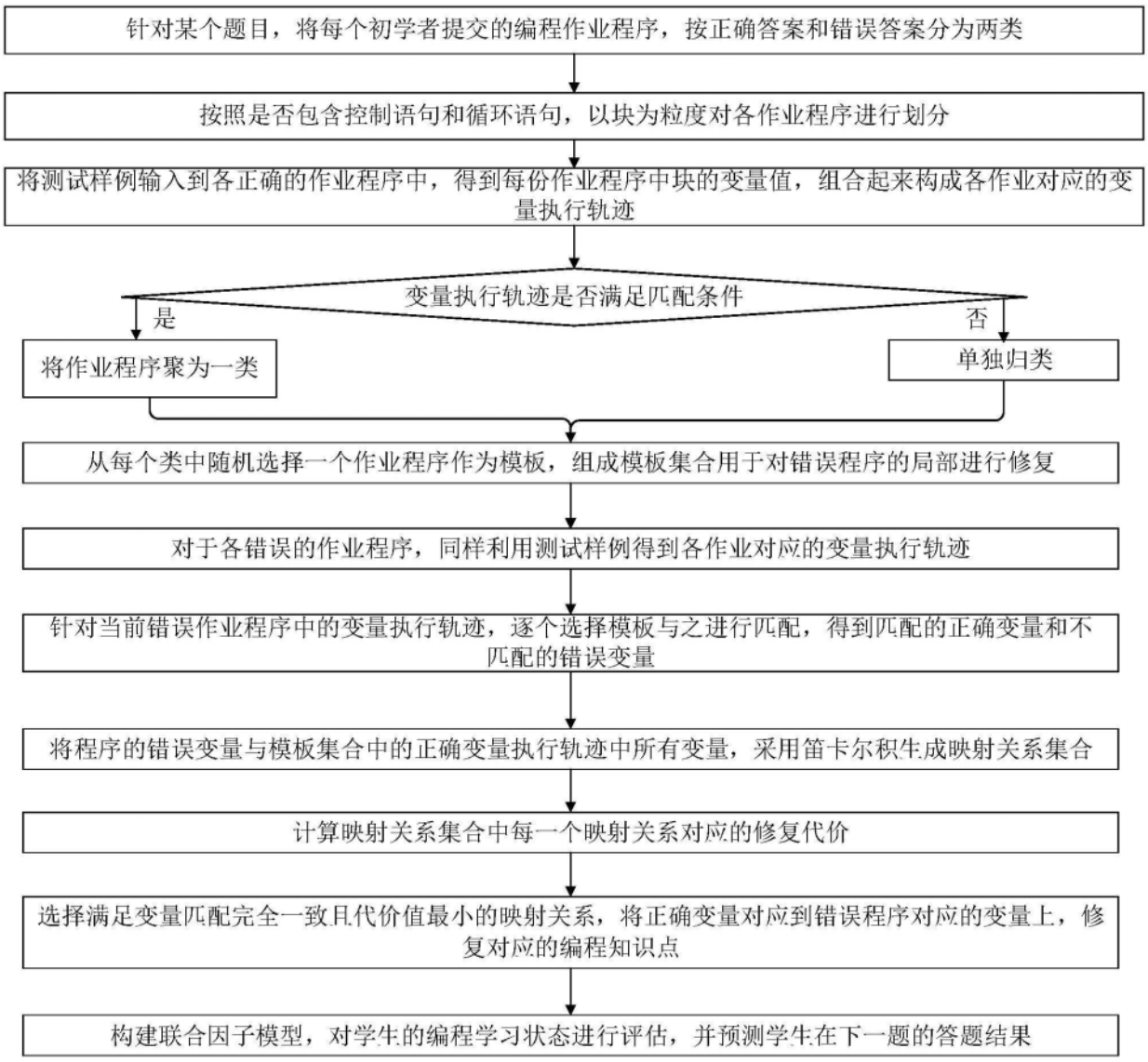

针对某个题目，将每个初学者提交的编程作业程序，按正确答案和错误答案分为两类
按照是否包含控制语句和循环语句，以块为粒度对各作业程序进行划分
将测试样例输入到各正确的作业程序中，得到每份作业程序中块的变量值，组合起来构成各作业对应的变量执行轨迹
变量执行轨迹是否满足匹配条件
是
否
将作业程序聚为一类
单独归类
从每个类中随机选择一个作业程序作为模板，组成模板集合用于对错误程序的局部进行修复
对于各错误的作业程序，同样利用测试样例得到各作业对应的变量执行轨迹
针对当前错误作业程序中的变量执行轨迹，逐个选择模板与之进行匹配，得到匹配的正确变量和不匹配的错误变量
将程序的错误变量与模板集合中的正确变量执行轨迹中所有变量，采用笛卡尔积生成映射关系集合
计算映射关系集合中每一个映射关系对应的修复代价
选择满足变量匹配完全一致且代价值最小的映射关系，将正确变量对应到错误程序对应的变量上，修复对应的编程知识点
构建联合因子模型，对学生的编程学习状态进行评估，并预测学生在下一题的答题结果

图2

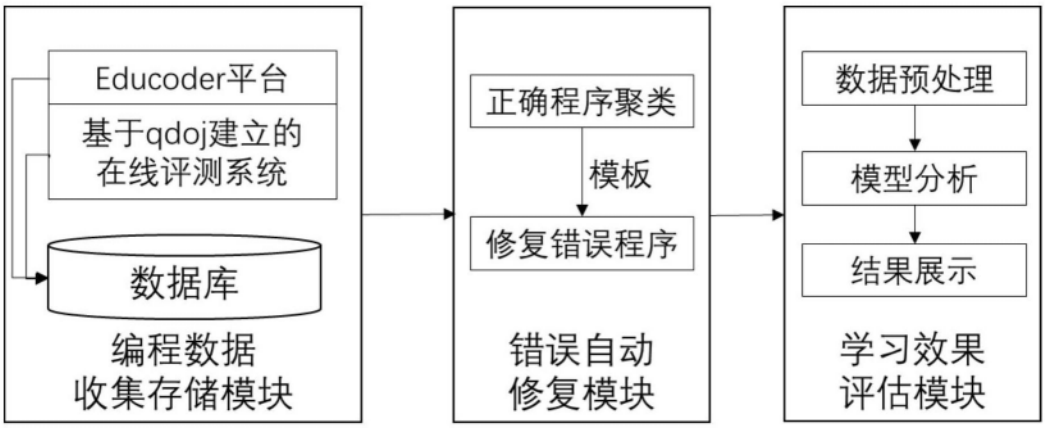

Educoder平台
基于qdoj建立的
在线评测系统
数据库
编程数据
收集存储模块
正确程序聚类
模板
修复错误程序
错误自动
修复模块
数据预处理
模型分析
结果展示
学习效果
评估模块

图3

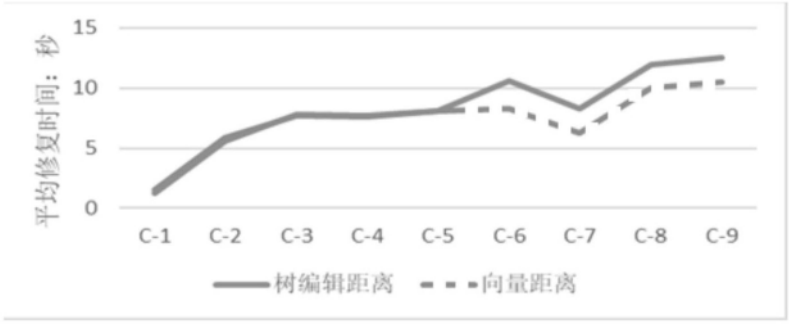

平均修复时间：秒
15
10
5
0
C-1
C-2
C-3
C-4
C-5
C-6
C-7
C-8
C-9
树编辑距离
向量距离

图4

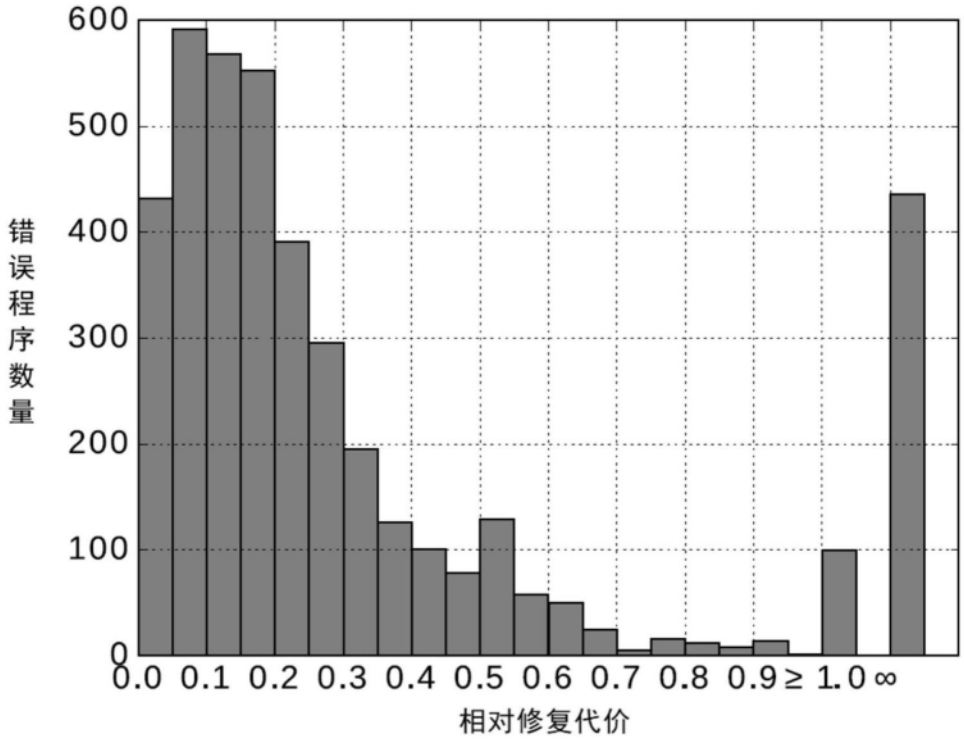

错误程序数量
600
500
400
300
200
100
0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 ≥ 1.0 ∞
相对修复代价

图5