大模型工具在编程教育中的纠错与优化效能的评价研究——以北邮码上与 OpenAI 的 Code Copilot 为例

Evaluation of Error Correction and Optimization Efficiency of Large Model Tools in Programming Education -- a Case Study of Code Copilot with OpenAI

牛佳慧[1]，黄月[2*]
[12] 北京邮电大学人文学院
[*] 2024111687@bupt.cn, huangy@bupt.edu.cn

【摘要】 数字技术高速发展的当代，编程已成为学习者的基础技能。然而，初学者在学习与探索时常因频繁代码错误受到过多挫折影响学习兴趣与动力。基于人工智能的大模型工具可以辅助学习者进行自主学习，但是不同工具在教育上的效能如何，尚未有具体研究。本研究以两个编程问题的 18 份 Python 错误代码作为测试数据源对两个主流工具 MashOn 和 Code Copilot 进行测试。结果表明，MashOn 和 Code Copilot 在覆盖率上可达到 100%，在精确率上均可达到 80%，能够生成完整的反馈和优化建议。在优化代码方面，MashOn 逊于 Code Copilot。但是 MashOn 的五轮启发式辅导对初学者更具指导意义，教育效能更强。
【关键词】 编程教育；人工智能；代码纠错；优化效能

Abstract: *With the rapid development of digital technology, programming has become a basic skill for learners. However, beginners are often frustrated by frequent code errors that affect their interest. Large model tools based on artificial intelligence can help learners learn, but their effectiveness in education has not been specifically studied. In this study, 18 Python error codes for two programming problems were used as data sources to test two mainstream tools, MashOn and Code Copilot. The results show that MashOn and Code Copilot can achieve 100% coverage and 80% accuracy, generating complete feedback and optimization recommendations. MashOn is inferior to Code Copilot when it comes to optimizing code. But MashOn's five rounds of heuristic tutoring are more instructive for beginners.*
Keywords: Programming education, Artificial intelligence, Code error correction, Optimization efficiency

1. 引言

数字技术高速发展的当代，编程已成为学习者的基础技能。编程的学习对于发展问题解决、创造性思维、算法思维、反思性思维、批判性思维和计算思维等 21 世纪技能具有重要作用(Erümit et al., 2020)。研究表明，初学者在面对频繁的程序报错提示时反复感受的挫折感可能是阻碍其持续学习的主要障碍(傅骞 et al., 2021)。

人工智能的快速发展在教育领域中引发了编程教学模式的优化和创新。AI-STEAM 课程通过体验式学习能够显著提升学生在编程逻辑等方面的知识掌握程度(Hsu et al., 2021)。CGAI（Conversational and Generative Artificial Intelligence，对话式和生成式人工智能）的突出应用包括提升人机交互体验、计算机程序/代码生成及系统创建(Akpan et al., 2024)。生成式人工智能工具的即时对话性可以在学习者因过度受挫而陷入沮丧时，为其提供有效的指导，从而保护学习者的学习兴趣、维持其学习动力。但是，繁多的人工智能工具在编程学习中的效用如何？编程教育在人工智能的支撑下应该做出哪些适应性的改变与革新？本研究将选择主流大模型工具，设置核心指标，分析与比较其对公共平台上学习者群体错误代码的矫错与优化能力，探讨其在编程教育中的效能，以及其在编程教育中的发展方向。

2. 国内外已有研究

2.1. 基于人工智能的多重工具被用于编程教育

国内外开发了多个集成人工智能技术的编程工具用于探究学生的学习效果。Wu 等基于建构主义学习理论和认知负荷理论，构建了一个基于 AIGC 的智能编程学习平台能够提供更加个性化的学习体验和更及时的学习互动。同时，AI 可以预测学生的学习表现以增加学生对当前自身的学习水平和状态的了解，便于提升学习效率(Wu et al., 2023)。Maranga 等使用多层感知器来动态调整编程问题的难度，以适应每个学生的技能水平，实现自适应评估功能，并促进编程教育个性化(Maranga et al., 2024)。

Sarsa 等探究使用 OpenAI Codex 生成编程练习（包括示例解决方案和测试用例）和代码解释的能力。大多数由 Codex 自动生成的编程练习都是合理且新颖的，并且包含了适当的样例解决方案(Sarsa et al., 2022)。教师还可以根据人工智能提供学生学习进展的数据支持来调整教学策略，完成数据驱动的教学改进与优化。Lee 等开发了一个基于精准教育的及时干预系统，通过利用深度学习和图像处理技术来即时识别学生在计算机编程课程中遇到的学习障碍，便于教师提供及时和适当的指导(Lee et al., 2023)。可见，人工智能技术支持下的编程工具进一步提升学习实时个性化、学习过程全监督和教学策略及时正确调整的水平。

2.2. 基于人工智能的多重理念与方法被用于编程教学

一些新型理念与思维训练也用于编程教学中，包括 CT（Computational Thinking，计算思维）方法被用于设计智能学习解决方案，以增强学生的编程学习体验(Agbo et al., 2019)。逆向工程思维等与 GAI（Generative Artificial Intelligence，生成式人工智能）进行融合而形成的教学模式应用于人机协作的编程教育中，以优化未来人机协作模式(翟雪松 et al., 2024)。

在学习过程中，学生们强烈偏好那些使他们在学习过程中保持自主性的人工智能辅助学习工具，相对于简单地展示直接的解决方案，更喜欢指导他们解决问题的步骤的脚手架(Denny et al., 2024)。同时，教师认为编程教育需要适应 AI 工具的出现，可能需要重新考虑课程学习目标，更多地关注编程过程而非最终提交的作品(Sheard et al., 2024)。可见，师生双方均在思考人工智能赋能下的编程教学模式未来发展方向。

综上，当前研究研究主要探讨人工智能赋能教育的学习范式转变、人工智能对师生学习行为的影响等研究主题，但是缺乏基于国内自主研发的大模型智能编程平台为学生提供的编程辅导质量的实践研究。

3. 研究过程

3.1. 研究对象

本研究选择北京邮电大学的码上（MashOn）平台和 OpenAI 中的 Code Copilot 作为被检测的大模型工具。码上（MashOn）是北京邮电大学于 2023 年 10 月 27 日上线，用于本科生编程教学的一款大模型工具。它于 2024 年入选了教育部高等教育司首批“人工智能+高等教育”应用场景典型案例，是国内高等教育领域率先开启的大模型赋能教育的探索。Code Copilot 是一种由 OpenAI 与 GitHub 联合推出的人工智能驱动的代码生成工具，StackOverflow 正式发布的 2023 年度开发者调研报告显示其是最受欢迎的 AI 开发工具。本研究将对这两款工具就代码解释、纠错和优化进行功能测试和分析。

3.2. 研究设计

本研究测试代码的数据源与测试工具均来源于 Geeksforgeeks 平台（简称 GF 平台，https://www.geeksforgeeks.org）。GF 平台是一个专为程序员设计的学习平台，易获得大量真实的源代码，其代码测试模块也受到广泛认可。研究选择 Fibonacci Sum 问题（斐波那契数列求和，https://www.geeksforgeeks.org/problems/fibonacci-sum1423/1）与 GCD Array 问题（最大公因数求解，https://www.geeksforgeeks.org/problems/gcd-array--170645/1）作为实验问题，它们均属于学习者必学的数据结构与算法基础问题。两个问题的代码源分别位于 GF 平台的 Pr

actice DSA 模块下，程序语言为 python。GF 平台的在线测评结果主要分为四类：(1)Wrong Answer（简称 WA），即答案错误；(2)Runtime Error（简称 RE），即运行时错误；(3)Time Limit Exceeded（简称 TLE），即时间超限；(4)Accept（简称 AC），即正确。本研究将在两个问题的 Submissions 界面下，在三种代码错误类型中分别随机选择三份代码，一共得到 2*3*3=18 份代码源文件进行测试。

研究首先检查并汇总每一份代码的错误情况，然后将修改完错误后的代码重新上传至 GF 平台进行测评并且保证通过测试，再将错误代码集分别上传至北邮码上平台 Tutoring 界面下 Code Correction 模块（https://ezcoding.bupt.edu.cn/ai/CODE_CORRECTION/create），和 OpenAI 中的 Code Copilot 模块（https://chatgpt.com/g/g-2DQzU5UZl-code-copilot）。然后，将大模型提示的代码错误情况进行分析、统计和汇总，对大模型提供的优化代码重新上传至网站进行评测，并记录代码运行结果。

3.3. 数据分析

本研究选择覆盖率和精确率作为核心指标来衡量两个编程工具在代码解释、纠错和优化方面的性能。将覆盖率定义为编程工具是否会根据每次的提交代码生成反馈，即解释上传的代码、提示代码中的错误和优化代码。因为代码要求的理解水平有可能会超出大模型的能力范围，进而导致大模型无法解答并生成相应的反馈。将精确率定义为编程工具反馈的代码问题在代码本身存在的所有错误中的占比。因为大模型回答的代码错误越全面细致，则表示该工具的代码纠错能力越强，性能越高。

(1) Fibonacci Sum 题目

Fibonacci Sum 题目提问方式和内容如图 1 所示。其中，Problem description 是题目含义，Example 是符合题目要求的测试用例，Buggy program 是用户提交但无法通过在线评测的代码。使用同一文本分别在两个平台进行测试。

Prompt to Generate Feedback

I'm working on a Python programming problem. The current program below is not working well. Can you help by explaining this program, describing the bug(s) in this program and fixing it? Below I first provide the problem description and then the current buggy program.

Problem description:
Given a number positive number N, find the value of f0 + f1 + f2 +... + fN where fi indicates ith Fibonacci number.
Remember that f0 = 0, f1 = 1, f2 = 1, f3 = 2, f4 = 3, f5 = 5,
Since the answer can be very large, the answer takes modulo with 1000000007 and returned.
You don't need to read input or print anything. Your task is to complete the function fibSum() which takes an integer N as input parameter and returns the sum of all the Fibonacci numbers from F0 to FN.
Expected Time Complexity: O(Log N)
Expected Space Complexity: O(1)
Constraints:1 <= N <= 100000

Example:
Input1:N = 3 Output1:4 Explanation1: 0 + 1 + 1 + 2 = 4
Input2:N = 4 Output2:7 Explanation2: 0 + 1 + 1 + 2 + 3 = 7

Buggy program:
{buggy_program}

图 1 关于 Fibonacci Sum 题目的提问方式及内容

该题在 MashOn 平台的生成反馈如图 2 所示。MashOn 平台采用五轮启发式辅导步骤：第一步 Intelligent Question Review 从理解斐波那契数列、矩阵幂运算的思路、矩阵幂运算的实现、计算总和和边界情况五个方面来拆解该问题的思考过程，第二步 Code Analysis 从类定义、方法定义、内部方法 fib_sum、内存检查、递归计算和返回结果六个方面拆解具体的代码实现过程，第三步 Key Point Guidance 提出三条关键代码提示，第四步 Detailed Guidance 提出四条详细代码提示，以及第五步 Correct Code 给出优化后的代码，以及 Code Comparison 提供优化前后的代码对比。

图 2 关于 Fibonacci Sum 题目在 MashOn 平台的生成反馈

该题在 Code Copilot 的部分生成反馈如图 3 所示。Code Copilot 从五个方面进行回答：首先从错误的斐波那契函数、模运算缺失、低效率和求和误解四个方面提示当前的代码错误 Explanation of the Buggy Program，其次利用斐波那契数的性质并采取矩阵幂运算指出正确实现方法 Correct Approach，然后提供优化后的代码 Fixed Code 和优化后的代码关键点 Key Points，最后对代码的建议 Suggestions。由于回复过长， Key Points 和 Suggestions 部分没有在图 3 中展示出来。

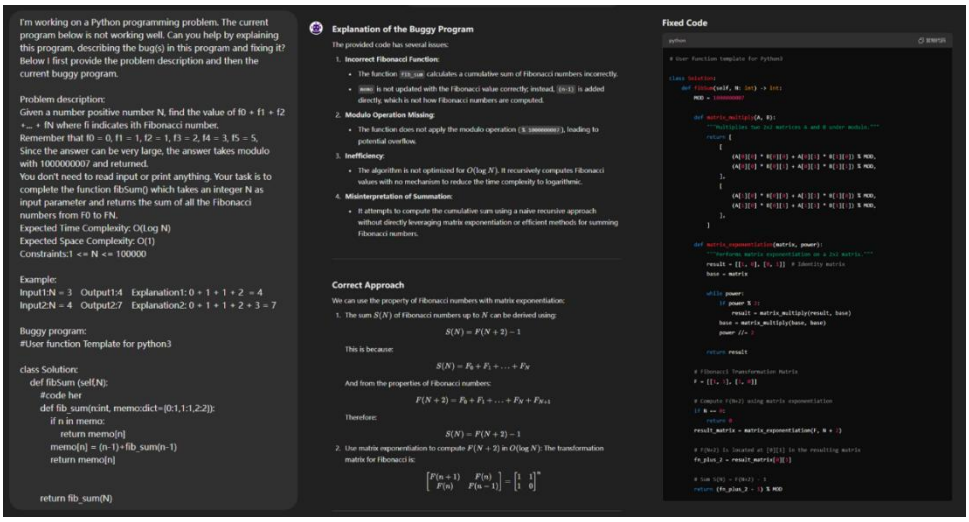

图 3 关于 Fibonacci Sum 题目在 Copilot 编程工具的生成部分反馈

Fibonacci Sum 问题的检测结果如由表 1 所示，两个编程工具均可以达到 100%的覆盖率，即可以对用户提供的代码进行解读、错误提示和优化。在精确率方面，MashOn 平台的平均精确率可达到 80%左右，Code Copilot 的平均精确率可达到 79%左右。因此，二者对于该问题的代码纠错能力几乎无差别。但是，经 Code Copilot 优化后的代码均可以通过网站的评测，而 MashOn 平台的代码优化能力稍差。

表 1 Fibonacci Sum 程序集检测结果

代码	错误类型	MashOn			Code Copilot		
		覆盖率	精确率	测评结果	覆盖率	精确率	测评结果

test1.py	WA	100%	60%	TLE	100%	75%	AC
test2.py	WA	100%	66.67%	AC	100%	100%	AC
test3.py	WA	100%	75%	AC	100%	25%	AC
test4.py	RE	100%	100%	WA	100%	100%	AC
test5.py	RE	100%	80%	AC	100%	20%	AC
test6.py	RE	100%	75%	AC	100%	100%	AC
test7.py	TLE	100%	66.67%	WA	100%	33.33%	AC
test8.py	TLE	100%	100%	AC	100%	50%	AC
test9.py	TLE	100%	100%	AC	100%	66.67%	AC

(2) GCD Array 题目

关于 GCD Array 题目提问方式和内容如图 4 所示，结构与 Fibonacci Sum 问题类似。

Prompt to Generate Feedback

I'm working on a Python programming problem. The current program below is not working well. Can you help by explaining this program, describing the bug(s) in this program and fixing it? Below I first provide the problem description and then the current buggy program.

Problem description:
You are given an array, arr of length N, and also a single integer K . Your task is to split the array arr into K non-overlapping, non-empty subarrays. For each of the subarrays, you calculate the sum of the elements in it. Let us denote these sums as S1, S2, S3, ..., SK. Where Si denotes the sum of the elements in the ith subarray from left. Let G = GCD(S1, S2 ,S3 , ..., SK). Find the maximum value of G that can be obtained. The array may contain duplicate elements. You don't need to read input or print anything. Your task is to complete the function solve() which takes the array arr[] and its size N and an integer K as input parameters and returns the required maximum GCD value.
Expected Time Complexity: O(N * x)
Expected Space Complexity: O(x), x is the number of factors of the sum of all elements.
Constraints:1 <= N <= 104 1 <= K <= N 1 <= arr[i] <= 105
Example:
Input1:N = 5 K = 4 arr[] = {6, 7, 5, 27, 3} Output1: 3 Explanation1: Since K = 4, you have to split the array into 4 subarrays.For optimal splitting, split the array into4 subarrays as follows: [[6], [7, 5], [27], [3]] Therefore, S1 = 6, S2 = 7 + 5 = 12, S3 = 27, S4 = 3 Hence, G = GCD(S1, S2, S3, S4) = GCD(6, 12, 27, 3) = 3 It can be shown that 3 is the maximum value of G that can be obtained. Thus, the answer is 3.
Input2:N = 3 K = 2 arr[] = {1, 4, 5} Output2: 5 Explanation2: Since K = 2, you have to split the array into 2 subarrays. For optimal splitting, split the array into 2 subarrays as follows: [[1, 4], [5]] Therefore, S1 = 1 + 4 = 5, S2 = 5 Hence, G = GCD(S1, S2) = GCD(5,5) = 5 It can be shown that 5 is the maximum value of G that can be obtained. Thus, the answer is 5.

Buggy program:
{buggy_program}

图 4 关于 GCD Array 题目的提问方式及内容

该题在 MashOn 平台的生成反馈如图 5 所示，包含五步：第一步 Intelligent Question Review 从理解问题、识别潜在的程序错误、修复程序的方法和详细步骤解释四个方面来拆解该问题的思考过程，第二步 Code Analysis 从初始化、可行性检查、GCD 计算、子数组求和与 GCD 更新、以及返回结果五个方面拆解具体的代码实现过程，第三步 Key Point Guidance 提出五条关键代码提示，第四步 Detailed Guidance 提出五条详细代码提示，以及第五步 Correct Code 给出优化后的代码，以及 Code Comparison 提供优化前后的代码对比。

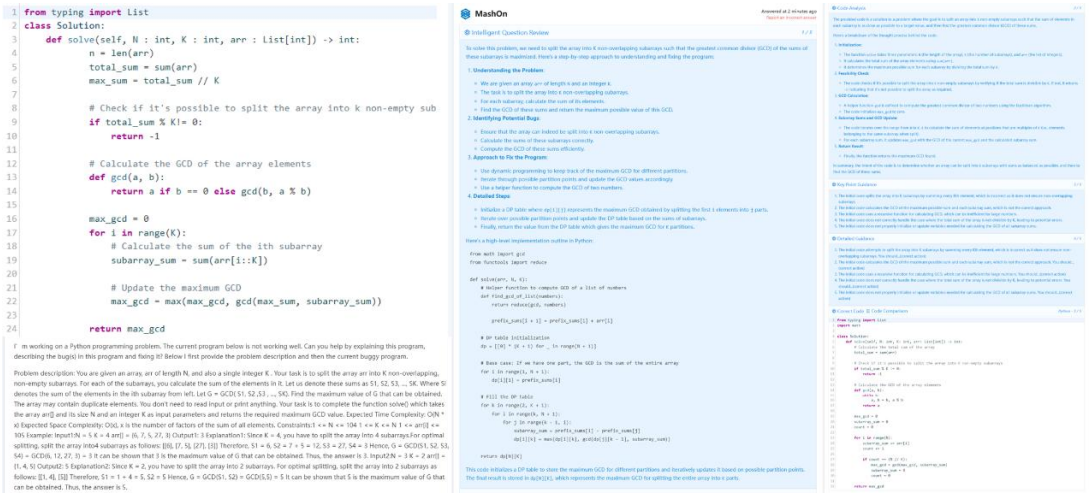

图 5 关于 GCD Array 题目在 MashOn 平台的生成反馈

该题在 Code Copilot 的生成反馈如图 6 所示，也包含五步：首先从子数组求和计算错误、max_sum 的误用、返回情况、GCD 计算逻辑和没有考虑 GCD 的全部计算情况提示当前的代码错误 Explanation of the Buggy Program，其次从数组 total_sum 求和、total_sum 的因子求解、检查每个因子和返回最大 GCD 四步提出正确的代码实现方法 Correct Approach，然后提供优化后的代码 Fixed Code 和优化后的代码关键点 Key Changes and Features（与关于 Fibonacci Sum 的回答中的 Key Points 相同），最后还有代码复杂度分析 Complexity Analysis、测试用例 Example Run 和对代码的建议 Suggestions。

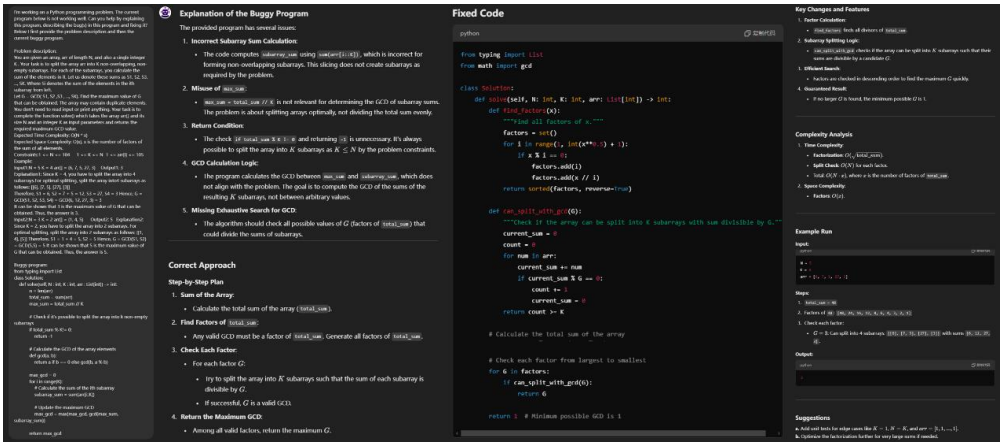

图 6 关于 GCD Array 题目在 Code Copilot 编程工具的生成反馈

GCD Array 的程序集检测结果如表 2 所示，两个平台的覆盖率均可达到 100%，即该题目处于大模型的理解范围内。在精确率方面，MashOn 平台的平均精确率约为 65%，Code Copilot 的平均精确率约为 66%。因此，二者对于该问题的代码纠错能力也几乎无差别。但是经 Code Copilot 优化后的代码的评测通过率略高于 MashOn 平台。

表 2 GCD Array 程序集检测结果

代码	错误类型	MashOn			Code Copilot		
		覆盖率	精确率	测评结果	覆盖率	精确率	测评结果
test1.py	WA	100%	66.67%	WA	100%	33.33%	WA
test2.py	WA	100%	100%	WA	100%	66.67%	WA
test3.py	WA	100%	50%	WA	100%	100%	WA
test4.py	RE	100%	33.33%	AC	100%	66.67%	WA
test5.py	RE	100%	100%	WA	100%	100%	AC
test6.py	RE	100%	66.67%	TLE	100%	33.33%	AC
test7.py	TLE	100%	100%	TLE	100%	100%	AC
test8.py	TLE	100%	0%	WA	100%	0%	AC
test9.py	TLE	100%	66.67%	TLE	100%	100%	AC

4. 结果与讨论

本研究选取了北邮的码上 MashOn 平台和 OpenAI 的 Code Copilot 两个主流工具，基于两个编程问题的 18 份 Python 错误代码作为测试数据源，比较分析两个工具在代码解释、纠错和优化方面的表现，获得以下结论。

(1) 人工智能工具在编程学习中可以基本满足初学者的需求

研究表明，MashOn 和 Code Copilot 在覆盖率上均达到了 100%，在精确率上均可达到 80%，能够生成完整的反馈和优化建议。在优化代码方面，MashOn 稍逊于 Code Copilot。而 MashOn 的五轮启发式辅导以学生为中心逐步引导和培养程序设计和算法思维，体现了建构主义学习

观。且初学者可以决定停止到哪一步，即“点到为止”，可见 MashOn 充当合理的脚手架，符合认知负荷理论，对初学者更具指导意义。

但是大模型工具会提供一些不算错误的冗余错误提示，即无法理解功能实现的多种代码形式。例如，对于第 1 个问题，有部分用户单独讨论 N==1，因此将循环条件表示为 for i in range(2, N)。但是码上平台认为该实现方式有误，建议改为 for i in range(1, N)的方式。可见当前大模型还不够“智能”，指出的代码错误也有待辨别和商榷。

(2) 人工智能不能代替教师成为编程教育的主导角色,充当辅助角色更为合适

在代码优化方面，虽然第 2 个问题的经典性不如第 1 个问题，但是第 2 个问题的算法逻辑相较第 1 个问题更为复杂一些，导致第 2 个问题的代码错误类型更加多样化。而两个工具似乎都无法有效处理程序的算法逻辑部分，使得代码纠错和优化水平均有所下降。因此需要教师在教学过程中引导学生合理正确使用大模型工具。

此外，研究存在一些局限性，包括：(1)目前研究收集的错误代码集存在知识点类型有些局限和数量过少的问题。而且，输入代码除了包括实现具体的数据结构，还应该包含功能实现和数据分析等应用类型。(2)而且当今大模型迭代速度迅速，可能会导致本次研究结果的时效性不长。但是增加对当前大模型工具的效能了解，有助于人工智能赋能编程教育的具体实现路径立足于当前的实际现状。

5. 未来展望

在未来，研究将收集更多数量、知识点和类型的错误代码集以涵盖基础的编程概念，具体的功能实现，数据分析与挖掘等编程语言应用方面，进而开展关于智能编程平台的代码纠错和优化功能更为全面检测的研究。

6. 致谢

本文系 2020 年度国家社会科学基金教育学一般课题“基于用户画像的高校教师混合教学建模及提升机制研究”(批准号:BCA200086)的阶段性研究成果。

参考文献

傅骞, 张力文, 马昊天, & 郑娅峰. (2021). 大学生编程韧性水平调查及其影响因素研究. 电化教育研究, 42(4), 29–36. https://doi.org/10.13811/j.cnki.eer.2021.04.005

翟雪松, 张丽洁, 夏亮亮, 徐鑫, & 朱强. (2024). 基于 GAI 的逆向工程教学思维在人机协作中的应用研究——以编程教育为例. 电化教育研究, 45(9), 61–68. https://doi.org/10.13811/j.cnki.eer.2024.09.008

Agbo, F. J., Oyelere, S. S., Suhonen, J., & Adewumi, S. (2019). A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. Proceedings of the 19th Koli Calling International Conference on Computing Education Research, 1–10. https://doi.org/10.1145/3364510.3364521

Akpan, I. J., Kobara, Y. M., Owolabi, J., Akpan, A. A., & Offodile, O. F. (2024). Conversational and generative artificial intelligence and human-chatbot interaction in education and research. INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH. https://doi.org/10.1111/itor.13522

Denny, P., MacNeil, S., Savelka, J., Porter, L., & Luxton-Reilly, A. (2024). Desirable Characteristics for AI Teaching Assistants in Programming Education. Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, 408–414. https://doi.org/10.1145/3649217.3653574

Erümit, A. K., Benzer, A. İ., & Şahin, G. (2020). A Framework for Studying Programming Teaching in Secondary Education. Croatian Journal of Education - Hrvatski Časopis Za Odgoj i Obrazovanje, 22(2). https://doi.org/10.15516/cje.v22i2.3560

Hsu, T.-C., Abelson, H., Lao, N., & Chen, S.-C. (2021). Is It Possible for Young Students to Learn the AI-STEAM Application with Experiential Learning? Sustainability, 13(19), 11114. https://doi.org/10.3390/su131911114

Lee, H.-Y., Lin, C.-J., Wang, W.-S., Chang, W.-C., & Huang, Y.-M. (2023). Precision education via timely intervention in K-12 computer programming course to enhance programming skill and affective-domain learning objectives. International Journal of STEM Education, 10(1), 52. https://doi.org/10.1186/s40594-023-00444-5

Maranga, X. D., Pausanos, M. V., Sinoy, S., Sta Romana, C. L., & Arellano, C. (2024). Integrating Adaptive Assessment in CodeChum for Personalized Programming Education. Proceedings of the 2024 10th International Conference on Education and Training Technologies, 89–95. https://doi.org/10.1145/3661904.3661910

Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. (2022). Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1, 27–43. https://doi.org/10.1145/3501385.3543957

Sheard, J., Denny, P., Hellas, A., Leinonen, J., Malmi, L., & Simon. (2024). Instructor Perceptions of AI Code Generation Tools—A Multi-Institutional Interview Study. Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, 1223–1229. https://doi.org/10.1145/3626252.3630880

Wu, H., Fa, D., Wu, X., Tan, W., Chang, X., Gao, Y., & Weng, J. (2023). Research on the Construction of Intelligent Programming Platform Based on AI-generated Content. The 15th International Conference on Education Technology and Computers, 9–15. https://doi.org/10.1145/3629296.3629298