借助人工智能辅助提升调试技能：用于调试支持的实时工具

Elizaveta Artser JetBrains Research 慕尼黑，德国
elizaveta.artser@jetbrains.com

Daniil Karol
JetBrains Research
柏林，德国
daniil.karol@jetbrains.com

Anna Potriasaeva JetBrains Research 贝尔格莱德，塞尔维亚
anna.potriasaeva@jetbrains.com

Aleksei Rostovskii JetBrains Research 柏林，德国
aleksei.rostovskii@jetbrains.com

Katsiaryna Dzialets
JetBrains Research
慕尼黑，德国
katsiaryna.dzialets@jetbrains.com

Ekaterina Koshchenko JetBrains Research 阿姆斯特丹，荷兰
ekaterina.koshchenko@jetbrains.com

Xiaotian Su 苏黎世联邦理工学院 苏黎世，瑞士
xiaotian.su@inf.ethz.ch

April Yi Wang
苏黎世联邦理工学院
苏黎世，瑞士
april.wang@inf.ethz.ch

Anastasiia Birillo
JetBrains Research
贝尔格莱德，塞尔维亚
anastasia.birillo@jetbrains.com

arXiv:2601.02504v1 [cs.SE] 5 Jan 2026

摘要

调试是编程教育和软件开发中的一项关键技能，但在计算机科学课程体系中却常常被忽视。为此，我们推出了一款集成于集成开发环境中的人工智能驱动的调试助手。它通过分析代码、建议断点并提供上下文提示，实现实时支持。该助手结合检索增强生成、大语言模型、程序切片和自定义启发式方法，在减少大语言模型调用次数的同时提高准确性，从而提升效率。三级评估——技术分析、用户体验研究和课堂测试——展现了其在调试教学中的潜力。

CCS 概念

• **应用计算** → **计算机辅助教学**；•**社会与专业主题** →**软件工程教育**。

关键词

智能辅导、调试、IDE 内学习、生成式人工智能

ACM 参考格式：
Elizaveta Artser、Daniil Karol、Anna Potriasaeva、Aleksei Rostovskii、Kat- siaryna Dzialets、Ekaterina Koshchenko、Xiaotian Su、April Yi Wang 和 Anas- tasiia Birillo。2026。《借助人工智能提升调试技能：用于调试支持的实时工具》。载于*2026 IEEE/ACM 第 48 届国际软件工程会议（ICSE-SEET '26）*，*2026*年*4*月*12*日至*18*日，巴西里约热内卢。ACM，纽约，美国纽约州，6 页。https://doi.org/10.1145/3786580.3786976

1 引言

调试是编程教育 [13]和专业软件开发 [28], 中一项必不可少的技能，但却很少得到教授在计算机科学课程体系 [2]中得到明确教授。大多数研究聚焦于预先设计的练习，用来讲授调试概念 [16, 23, 27, 30, 34] ，但未能帮助学生将这些技能应用于实际编码任务 [35]。与此同时，提供实时反馈能显著提升学生的表现和学习热情 [24, 25, 33]。

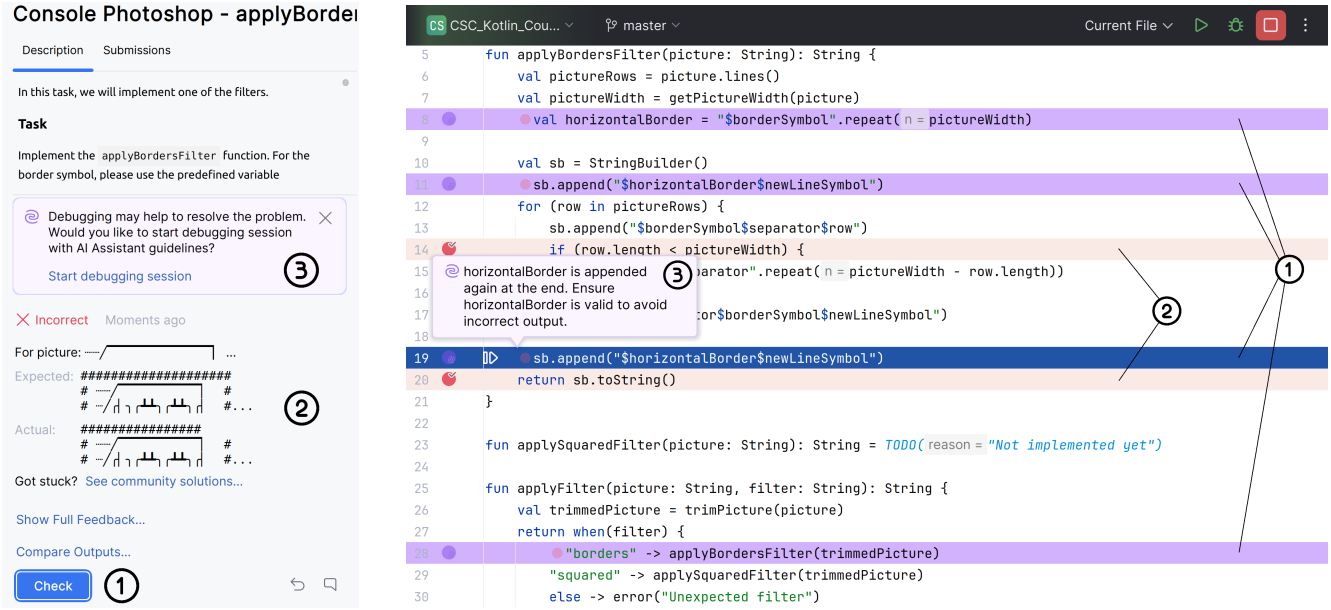
大语言模型（LLM）的最新进展为在编程练习中提供实时调试辅助带来了潜在解决方案。大语言模型已广泛用于程序修复 [18] 以及在教育中生成个性化反馈 [24]。然而，大语言模型可能会生成不准确或不可靠的回答，因此难以完全依赖它们 [10, 20, 31, 36]。近期一项研究提出了一种有前景的方法：将大语言模型与集成开发环境的内部机制（如静态分析和代码质量检查器）相结合，以提升最先进方法所提供的下一步提示的质量 [5]。类似的方法也广泛应用于其他领域，以提升模型的输出质量 [8, 17]。由于调试通常在集成开发环境中进行，这种方法有望与大语言模型结合，帮助学生进行调试。

本文介绍了一款人工智能驱动的调试助手，该助手集成于开源的 JetBrains Academy 插件中，用于 IDE 内学习 [6]。该工具通过分析学生的代码、为存在程序缺陷的程序推荐断点并提供上下文解释，提供实时调试支持。它将检索增强生成 (RAG) [15]与大语言模型相结合，以生成解决方案，并利用程序切片和自定义启发式方法高效识别断点位置。该工具的源代码可在补充材料中找到 [3]。一项包含技术分析、用户体验评估和课堂试点测试的三级评估显示了令人鼓舞的结果，凸显了该工具在有效协助学生调试方面的潜力。

2 相关工作

一种常见的调试教学方式是通过调试练习 [30, 32, 35], ，让学生找出并（有时）修复程序缺陷

本作品采用知识共享署名 4.0 国际许可协议授权
ICSE-SEET '26，里约热内卢，巴西© 2026 版权归所有者/作者所有。ACM ISBN 979-8-4007-2423-7/2026/04
https://doi.org/10.1145/3786580.3786976



(a) 任务面板，含 (1) 检查按钮，(2) 失败的测试，(3) 会话开始消息。

(b) 引导式调试会话中的代码编辑器，含 (1) 人工智能调试助手建议的断点，(2) 学生设置的断点，以及 (3) 断点提示。

图 1：人工智能调试助手工具的主要组件。

（这些程序缺陷）出现在有问题的代码中。以往的研究对此进行了拓展，提出了多种方法，例如在练习中加入调试打印语句 [12] 或日志 [11]、将调试视为假设驱动过程 [26, 32]，或采用故障排查框架 [21]。策略还包括学习材料，如手册 [16] 或速查表 [4]，以及结构化教学，例如调试课程或交互式演示 [9, 34]。这些方法效果显著，但通常需要额外材料，也需要教育者投入大量精力，才能及时提供支持。

大语言模型的兴起为支持学生调试和练习策略带来了新的机遇。聊天式助手 [18, 22] 通过概念性解释、伪代码和针对性注释提供有约束的反馈，帮助学生查找错误。像 *BugSpotter* [30] 这样的工具可以生成调试任务，而*HypoCompass* [24] 则专注于通过诊断含错误代码来进行假设构建。尽管这些工具潜力巨大，但基于大语言模型的工具在可靠性和无缝集成到集成开发环境方面仍面临挑战。

为了解决这一问题，一些研究将调试支持嵌入集成开发环境中。例如，*Ladebug* [23] 提供了一个基于浏览器、类似集成开发环境的环境，让学生在预设练习中进行实践。最近，Noller 等人将故障定位技术与大语言模型相结合，通过自动设置断点和聊天机器人界面，引导学生在集成开发环境中完成调试过程 [29]，。尽管我们的工作与之密切相关，但我们采用了不同的断点建议方法，重点关注解答的性能、质量和成本。此外，我们直接在编辑器中提供断点的上下文提示，减少了对聊天机器人界面的依赖。这一区别表明，人工智能驱动的调试支持可以通过不同方式集成到集成开发环境中，因此研究多种方法具有重要价值。

我们的工作将调试支持集成到集成开发环境中，与学生的工作流程相契合，并减少上下文切换。不同于使用预设任务的既有方法，我们的工具针对学生自己的代码提供个性化支持。通过将静态分析等集成开发环境内部机制与大语言模型相结合，它能够提供可靠且可扩展的辅助，同时不会增加教育工作者的工作负担。

3 人工智能调试工具

3.1 使用流程

人工智能调试助手集成在 JetBrains Academy 插件中，用于 IDE 内学习 [6]。学生可以查看任务说明，并点击检查按钮（图 1a-1）检查进度，该按钮会在本地运行测试。如果发生错误（图 1a-2），该工具会为学生提供启动引导式调试会话 的选项（图 1a-3）。

在调试会话期间，工具会用紫色标出推荐的断点（图 1b-1），以便与学生设置的断点（图 1b-2）区分开来。学生按照这些断点进行调试，并查看解释提示（图 1b-3）。会话结束后，AI 推荐的断点会转换为常规断点，让学生能够继续常规调试流程。

3.2 工具管线

该方法的总体思路是生成一个正确版本的学生的解答，使其通过失败的测试。在此基础上，AI 调试助手会生成并解释应向学生展示的断点。图 2 展示了该工具的内部管线。该工具由三个主要组件构成：

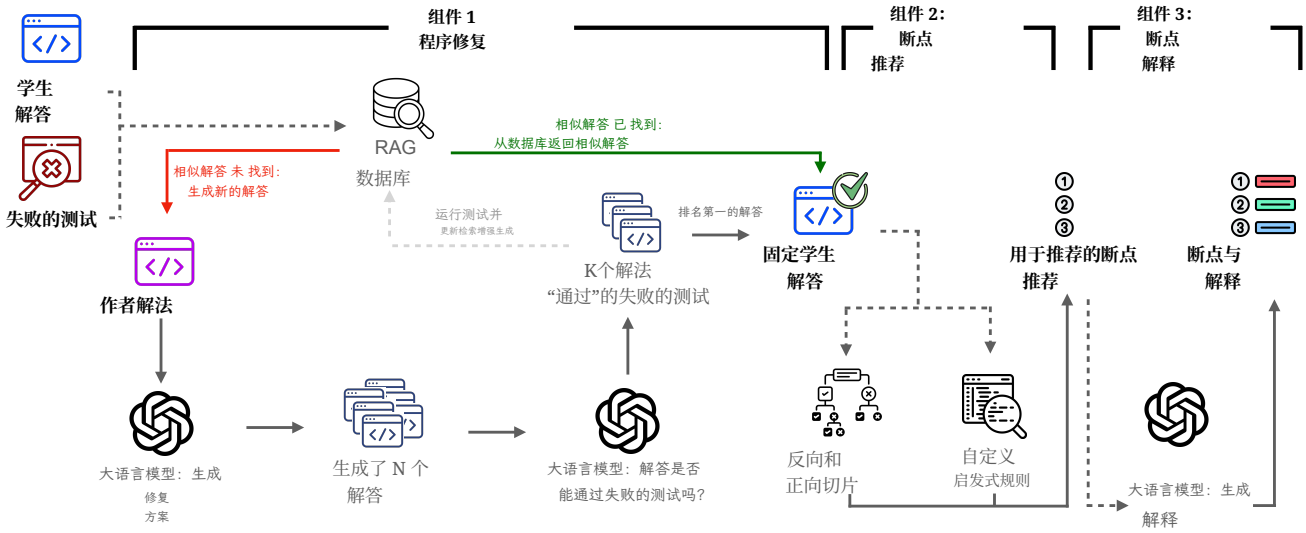


图 2：工具管线：(1) 程序修复；(2) 断点推荐；(3) 断点解释组件。程序修复组件接收学生解答和一次失败的测试，并在检索增强生成数据库中搜索相似的正确解答。如果找到匹配项，流程便结束；如果没有，则会在作者的初始解答之外生成一个新的修正后的解答。断点推荐组件会分析修正后的解答，断点解释组件则会为每个推荐的断点生成解释。

程序修复（图 2-1）、断点推荐（图 2-2）和断点解释（图 2-3）。目前，该工具仅支持 Kotlin，但未来可以扩展至其他语言。

3.2.1 程序修复。第一个关键组件是程序修复（图 2-1），它以学生的解答和失败的测试为输入，并返回一个能够通过该测试的修正解答。为生成修复方案，我们采用组合方法：使用检索增强生成系统 [15] 从现有解答中检索正确解答，或使用大语言模型生成新解答。这可以尽量减少不必要的大语言模型调用并避免幻觉 [36], ，同时也适用于没有既往学生提交记录的新课程。

检索增强生成系统。 检索增强生成系统使用现有的解答以及有关测试的信息，这些信息以 LaBSE模型生成的嵌入形式存储[1]。数据库可以为空（适用于新任务），也可以预先填充之前的提交内容。当请求调试会话时，系统会为学生的解答和失败的测试生成嵌入。系统使用 ≥ 0.8的余弦相似度阈值，搜索能够通过同一测试的相似解答。如果找到，则返回该解答。否则，大语言模型会生成新的解答。

大语言模型解答生成。 我们使用 gpt-4o 模型生成修正后的解答。由于大语言模型容易产生幻觉 [10,20, 31, 36], ，因此会并行生成五个解答，并从中选出最佳解答。我们不运行实际测试，而是让 o3-mini 模型充当二元分类器，预测每个解答能否通过失败的测试。对于预测为通过 的修正后解答，我们使用嵌入进行排序，优先选择与学生原始代码最接近的解答。最终返回排名最高的解答。完整提示词见补充材料 [3]。

将新解答上传至检索增强生成系统。 大语言模型解答生成会创建新的提交，并将其添加到检索增强生成系统中，以扩充数据库并减少今后调用大语言模型的次数。起初，我们依赖模型的预测来节省时间（图 2-2）。为确保仅添加有效解答，我们之后会运行实际测试。这一步不仅能扩充数据库，还能让我们验证所采用的方法，并找出大语言模型对测试失败情况的预测不准确之处。这一步独立于 AI 调试助手工具的其他部分执行，不会影响其性能。

3.2.2 断点推荐。 生成正确版本的解答后，下一步是向学生建议一组可能的断点位置 （图2-2）。在我们的工作中，我们考虑两种类型的断点：(1) 位于程序缺陷所在的行上 、需要直接修复的断点；(2) 位于可能受程序缺陷影响的行上的断点，用于引导学生找到需要修复的区域。第一类断点通过计算学生当前解答与正确解答之间的差异自动确定——这些正是需要修改才能使解答正确的确切位置。更有意义的部分在于利用这些信息，找出学生应检查的其他位置，从而完成这些修复。

我们采用组合方法来识别可能受程序缺陷影响的代码行。其中包括编程反向和正向切片 [1, 19] 以及自定义启发式方法 ，以限制推荐断点的数量。为了向学生展示最终的断点列表，我们取这些断点的交集。我们认为这种方法有效，是因为它解决了两个关键问题：1）保留最重要的断点，即最有可能影响程序缺陷的断点；2）限制断点数量。这样可以确保我们的系统不会创建过多的断点

[1]LaBSE模型：https://huggingface .co/sentence - transformers/LaB SE

并让学生能够专注于调试程序中最相关的部分。

反向和正向切片。代码切片可识别影响（反向切片）或受（正向切片）特定程序元素影响的代码部分。它利用静态信息定位问题区域，广泛应用于调试工具 [1, 19]。

我们基于 IntelliJ IDEA API 中的程序结构接口（PSI）[2] 构建了数据流图和控制流图，以提取数据依赖和控制依赖。数据依赖指一个变量的值取决于另一个变量的情况，例如属性或函数调用中的依赖。例如，在 var c = a + b 中，c 对 a 和 b 存在反向依赖，而 a 和 b 对 c 存在正向依赖。控制依赖指一条语句的执行取决于另一条语句的情况，例如if 语句或循环。例如，在 if 语句中，两个条件分支都对该 if 语句存在反向控制依赖。因此，此步骤会生成指向学生代码中错误部分的正向和反向依赖。然而，仅靠切片可能会建议过多代码行，因为它会包含代码中受影响的所有行。

启发式规则。基于我们的编程经验和以往研究 [14]，，我们为可能设置断点的位置制定了一份自定义启发式方法列表。为确保方法具有确定性、避免模型幻觉并降低成本，我们没有使用大语言模型。

(1) **条件语句启发式规则**。当修复涉及更改条件语句时，例如 if、when 或 while，应在所有依据这些条件执行的代码块开头设置断点，以便监控逻辑在不同场景下的执行流程。(2) **变量修改启发式规则**。如果修复涉及修改变量的代码行，例如属性更改、二元表达式、for 表达式中的循环参数，或 while 条件中的属性，应设置断点，以观察变量修改前后的状态。(3) **函数作用域启发式规则**。当修复发生在函数定义内时，应在函数开头以及代码中所有调用该函数的位置设置断点，以跟踪函数的执行情况。这有助于确定更改后是否调用了该函数，以及它的行为是否正确。

3.2.3 断点解释。流水线的最后一步是生成断点解释（图 2-3）。我们使用 GPT-4o 为每个断点生成解释，帮助学生聚焦于特定代码行。该模型会识别正在修改的变量或对象，并提供详细说明，帮助理解这些变化。对于位于可能受程序缺陷影响的代码行上的断点，我们要求模型解释这些代码行与最终错误之间的关系。对于位于程序缺陷所在位置的代码行上的断点，我们要求模型解释错误发生的原因 [3]。

4 评估

4.1 研究问题1：断点推荐质量

本节介绍对所提出方法的技术评估，以回答研究问题*1*：人工智能驱动的调试系统在推荐断点方面有多精准？。评估重点关注程序修复模块（图 2-1）和断点推荐模块（图 2-2）。

表 1：候选模型的执行时间对比，其中 P 表示精确率，R 表示召回率。灰色表示最佳模型。

模型	平均值（秒）	最大值（秒）	P	R	F1
gpt-4o-mini(2048)	9.11	12.18	-	-	-
gpt-4o-mini(1024)	**8.24**	**10.68**	0.68	**0.88**	0.76
o3-mini（低）	**6.50**	**10.23**	**0.70**	0.87	**0.78**
o3-mini（中）	17.85	44.92	-	-	-

4.1.1 程序修复评估。开发的系统依赖于经过修正且必须通过失败测试的学生解答。我们使用基于大语言模型的模型来预测该解答能否通过失败的测试。为探究此类分类器能否有效替代实际测试执行，我们评估了它们作为二元预测器的可行性，以判断生成的修复能否通过失败的测试。

执行时间。我们在为*Kotlin* 入门介绍[3]生成的 150 份解答样本上，对四种模型进行了基准测试。我们使用 MacBook Pro（配备 Apple M3 Max 芯片和 64 GB 内存）对每份解答运行分类器。根据测量结果（见表 1），我们选取了 o3-mini（low）和 gpt-4o-mini（1024）模型进行后续评估，因为它们速度最快。

质量评估。为了评估模型预测生成的修复是否正确的可靠性，我们使用 1,796 份错误的 Kotlin 学生提交进行了评估。针对每份提交，先由 GPT-4o 生成修复（见图 2-1）。随后，我们使用前一项实验中速度最快的两种模型，对每份修复后的提交进行预测，判断修复后的代码是否确实修复了学生解答，并通过对每份修复后的提交运行测试来验证结果。分类器的性能汇总见表 1。综合考虑效率和质量后，我们选择了 o3-mini。

4.1.2 断点推荐评估。 所开发系统的另一个关键组件是断点推荐模块（图 2-2）。我们假设学生的错误解答及相应的修正后的解答均可获取。我们从 Kotlin 课程[3]中抽取了 32 组学生解答及其修正版本，借此评估所提出的方法。本文的两位作者均拥有两年以上的研究经验和四年以上的 Kotlin 编程经验，他们分别标注数据，找出潜在的断点位置。完成标注后，两人讨论结果并达成共识。我们使用标注后的数据评估了系统的质量。系统的精确率为 0.9，召回率为 0.7，F1 分数为 0.79。这些结果表明系统表现出色，因此我们决定保留当前方法，以便进一步对学生进行评估。

4.2 研究问题2：工具的可用性

所提出的系统嵌入在集成开发环境中，学生可能难以使用它 [7]。为了解决这一问题并回答研究问题*2*：学生如何看待该工具的用户体验，以及如何改进？，我们与学生开展了一项可用性研究。

[2]PSI：https://plugins.jetbrains.com/docs/intellij/psi.html

[3]Kotlin 入门介绍课程：https://plugins.jetbrains.com/plugin/21067-kotlin-onboarding-introduction

参与者。我们邀请了八名具备基础 Kotlin 知识的计算机科学专业学生参加为时 15 分钟的半结构化访谈。所有参与者的调试经验都有限，也就是说，他们曾尝试使用调试器，但并未经常使用。

实验设置。实验分为两个阶段：（1）三次试点访谈；（2）五次用于分析的访谈。访谈以英语进行，使用功能有限的交互式 Figma 原型，使参与者能够与人工智能调试助手互动，但无法编辑代码。参与完全自愿。我们向学生说明了研究目的、数据收集方式及数据用途。参与者使用该原型解决一道入门级 Kotlin 题目，该题目提供的解答含有程序缺陷；他们需要找到程序缺陷并说明如何修复。完整的访谈脚本和任务见补充材料 [3]。

结果。大多数参与者在初次使用后便觉得该工具直观易用。他们认可预先配置的断点，认为这些断点鼓励了更深入的探索。总体而言，该原型易于使用。许多参与者喜欢提示这一设计，但认为提示内容不够充分。五位参与者指出，诸如“检查这个值是否正确”之类的通用提示并无助益。最终版本已更新，加入了更详细的解释。学生还指出，断点解释会与代码重叠。在初始原型中，提示默认显示；最终版本则改为仅在将鼠标悬停于断点上时显示提示。

4.3 研究问题3：学生视角

评估的最后一步是在课堂上进行的一项试点研究，旨在回答研究问题*3*：新手学生对该工具在课堂环境中的初步印象如何？

参与者。我们邀请了20名工程科学专业的一、二年级本科生参加。所有参与者都具备基本的编程经验，至少完成过一个学期的编程课程。为确保实验过程清晰明了，我们选择了调试经验有限的学生；我们根据他们使用调试工具的频率以及其对调试过程的自我报告信心来评估这一点。

实验环境。学生在受控实验室环境中参加了一场为时90分钟的课程。参与完全自愿，学生事先获知了研究目的、数据收集方式及数据用途。研究未收集任何个人数据。学生使用预装了该工具的笔记本电脑，并获得了纸质说明，其中介绍了该工具的概念和主要组成部分。在课程期间，参与者完成了两类编程练习：若干常规编程任务，要求实现一些函数；以及两项调试任务，一项在前、一项在后，每项任务均包含多个程序错误 [3]。完成任务后，学生填写了一份问卷。

结果。 实验显示，在调试会话中，20 名学生中只有 8 名使用了人工智能调试助手。最可能的原因是学生对 Kotlin 知之甚少，加上会话时间有限，导致出现语法错误，妨碍学生充分使用该助手。然而，学生总体上对这一工具的理念持积极态度。

根据 1-5 李克特量表问题的结果，使用所提出的系统的学生给出了较为正面的反馈，对所提供断点位置的正确性平均评分为 3.13，对断点解释清晰度的平均评分为 3.25。此外，三名学生指出，该工具能够定位并总结程序缺陷来源，这对调试很有帮助。同时，两名学生提到，自动建议的断点很有用，也有助于简化调试过程。

学生提出了三个主要的工具改进方向。首先，四名学生建议将该工具与聊天式系统集成，让学生能够提出澄清问题。未来的更新中可以轻松实现这一想法。其次，可以改善用户体验，因为学生不清楚需要将鼠标悬停在断点上才能查看说明信息。可以通过入门引导工具在最初几次使用时演示这一功能，从而解决这一问题。最后，一些学生提到可以添加代码生成功能，以提供修正后的解答。不过，这还需要进一步研究，因为该工具的主要目标是教学并协助调试，而不是直接提供正确解答。

5 有效性威胁

所提出的方法存在若干主要局限。首先，技术评估的规模有限。增加任务的多样性有助于通过发现更多案例来完善自定义启发式规则集。这仍是未来需要研究的方向。其次，课堂试点评估的参与者人数较少。尽管邀请了 20 名学生，但只有 8 人在课堂上使用了该工具。虽然我们收集到初步证据，表明学生认可这种方法，但无法评估该工具对学习的实际影响，结果也可能受到新奇效应的影响。未来计划开展实验时间更长、参与者人数更多的研究，以解决这一局限。

6 结论

本研究针对计算机科学课程中调试技能教学的不足，提出了一种集成到集成开发环境中的人工智能驱动的调试助手。该工具通过识别断点、提供解释，并运用检索增强生成和程序切片等技术，提供实时、个性化的调试支持，从而确保准确性和效率。我们的三级评估证明了这一方法的可行性，并为未来评估其对学生调试实践和学习成果的影响指明了方向。本研究为将实时调试工具融入编程教育奠定了基础，并鼓励进一步研究可扩展的调试教学解决方案。

致谢

我们要感谢 Evgenii Moiseenko 在本项目中指导团队，尤其是在程序切片、代码审查和算法讨论方面。特别感谢 Yaroslav Zharov 和Rauf Kurbanov 在机器学习方面的指导，并协助我们构建系统。我们还要感谢 JetBrains Research 的 Human AI Experience 团队为学生设计并开展用户体验研究，并提供宝贵见解以改进系统。最后，感谢苏黎世联邦理工学院决策科学实验室协调学生实验室研究。

参考文献

[1] Hiralal Agrawal 和 Joseph R Horgan。1990年。动态程序切片。*ACM SIGPlan Notices* 25，6（1990），246–256。[2] Marzieh Ahmadzadeh、Dave Elliman 和 Colin Higgins。2005年。对计算机科学初学者调试模式的分析。*ACM SIGCSE Bulletin* 37，3（2005），84–88。doi:10.1145/1151954.1067472[3] Elizaveta Artser、Daniil Karol、Anna Potriasaeva、Aleksei Rostovskii、Katsiaryna Dzialets、Ekaterina Koshchenko、Xiaotian Su、April Yi Wang 和 Anastasiia Birillo。2025 年。补充材料。检索于 2026 年 1 月 4 日，网址：https: //zenodo.org/records/18045875[4] Andrew Ash 和 John Hu。2025年。WIP：探索调试速查表和微型讲座在提升本科生调试技能与思维方式方面的价值。*arXiv* 预印本 *arXiv:2506.11339*（2025年）。[5] Anastasiia Birillo、Elizaveta Artser、Anna Potriasaeva、Ilya Vlasov、Katsiaryna Dzialets、Yaroslav Golubev、Igor Gerasimov、Hieke Keuning 和 Timofey Bryksin。2024年。一步一步来：结合大语言模型与静态分析，为编程任务生成下一步提示。载于第*24*届 *Koli Calling* 国际计算教育研究会议论文集。1–12。[6] Anastasiia Birillo、Mariia Tigina、Zarina Kurbatova、Anna Potriasaeva、Ilya Vlasov、Valerii Ovchinnikov 和 Igor Gerasimov。2024年。连接教育与开发：将集成开发环境打造为交互式学习平台。载于第一届 *ACM/IEEE* 集成开发环境研讨会论文集。53–58。[7] Anastasiia Birillo、Ilya Vlasov、Katsiaryna Dzialets、Hieke Keuning 和 Timofey Bryksin。2025年。IDE 内编程课程：在真实环境中学习软件开发。载于*2025 IEEE/ACM* 第二届 *IDE* 研讨会（*IDE*）。IEEE，1–6。[8] Scott Blyth、Sherlock A Licorish、Christoph Treude 和 Markus Wagner。2025年。将静态分析作为反馈循环：让大语言模型生成的代码超越正确性。（2025年），100–109。[9] Liia Butler、Charlotte Kiesel、Dipayan Mukherjee、Mohammed Hassan、Mattox Beckman 和 Geoffrey Herman。2025年。ILDBug：一种调试教学的新方法。载于第*56*届 *ACM* 计算机科学教育技术研讨会论文集（第*2*卷）。1731–1731。[10] Yupeng Chang、Xu Wang、Jindong Wang、Yuan Wu、Linyi Yang、Kaijie Zhu、Hao Chen、Xiaoyuan Yi、Cunxiang Wang、Yidong Wang 等。2024年。大语言模型评估综述。*ACM* 智能系统与技术汇刊15，3（2024），1–45。[11] Ryan Chmiel 和 Michael C Loui。2004年。调试：从初学者到专家。*Acm Sigcse Bulletin* 36，1（2004），17–21。[12] Joel Fenwick 和 Peter Sutton。2012年。利用 Quicksand 改进初学阶段之后学生的调试实践。载于第十四届澳大拉西亚计算教育会议论文集，第*123*卷。141–146。[13] Sue Fitzgerald、Renée McCauley、Brian Hanks、Laurie Murphy、Beth Simon 和 Carol Zander。2009年。从学生视角看调试。*IEEE* 教育汇刊53，3（2009），390–396。[14] Eduardo Andreetta Fontana 和 Fabio Petrillo。2021年。断点类型映射：一项探索性研究。载于*2021*年 *IEEE* 第*21*届软件质量、可靠性与安全国际会议（*QRS*）。IEEE，1014–1023。[15] Yunfan Gao、Yun Xiong、Xinyu Gao、Kangxiang Jia、Jinliu Pan、Yuxi Bi、Yixin Dai、Jiawei Sun、Haofen Wang 和 Haofen Wang。[无日期]。大语言模型的检索增强生成：综述。*arXiv* 预印本 *arXiv:2312.10997*（[无日期]）。[16] Rita Garcia、Chieh-Ju Liao 和 Ariane Pearce。2022年。阅读调试手册：面向计算机科学入门课程学生的调试手册。载于*2022*年 *IEEE* 教育前沿会议（*FIE*）。IEEE，1–7。[17] Imen Jaoua、Oussama Ben Sghaier 和 Houari Sahraoui。2025年。结合大语言模型与静态分析器以生成代码审查。载于*2025*年 *IEEE/ACM* 第*22*届软件仓库挖掘国际会议（*MSR*）。IEEE，174–186。[18] Majeed Kazemitabaar、Runlong Ye、Xiaoning Wang、Austin Zachary Henley、Paul Denny、Michelle Craig 和 Tovi Grossman。2024年。Codeaid：评估一种兼顾学生与教育者需求的基于大语言模型的编程助手在课堂中的部署。载于*2024*年 *CHI* 计算机系统中的人为因素会议论文集。1–20。[19] Bogdan Korel 和 Janusz Laski。1988年。动态程序切片。《信息处理快报》29, 3 (1988), 155–163。[20] Charles Koutcheme、Nicola Dainese、Sami Sarsa、Arto Hellas、Juho Leinonen、Syed Ashraf 和 Paul Denny。2025年。评估用于生成和评判编程反馈的语言模型。载于第*56*届*ACM*计算机科学教育技术研讨会论文集 *V. 1*。624–630。[21] Chen Li、Emily Chan、Paul Denny、Andrew Luxton-Reilly 和 Ewan Tempero。2019年。迈向调试教学框架。载于第*21*届澳大拉西亚计算教育会议论文集。79–86。[22] Mark Liffiton、Brad E Sheese、Jaromir Savelka 和 Paul Denny。2023年。Codehelp：在编程课堂中使用带有护栏机制的大语言模型提供可扩展支持。载于第*23*届 *Koli Calling* 国际计算教育研究会议论文集。1–11。[23] Andrew Luxton-Reilly、Emma McMillan、Elizabeth Stevenson、Ewan Tempero 和 Paul Denny。2018年。Ladebug：帮助编程初学者提升调试技能的在线工具。载于第*23*届*ACM*计算机科学教育创新与技术会议论文集。159–164。[24] Qianou Ma、Hua Shen、Kenneth R. Koedinger 和 Sherry Tongshuang Wu。2024年。如何在人工智能时代教授编程？将大语言模型用作调试的可教代理。载于*2024*年*ACM*国际计算教育研究会议（*ICER*）论文集。ACM。[25] Samiha Marwan、Ge Gao、Susan Fisk、Thomas W Price 和 Tiffany Barnes。2020年。自适应即时反馈可以提升编程初学者的编程参与度和在计算机科学领域坚持下去的意愿。载于*2020*年*ACM*国际计算教育研究会议论文集。194–203。[26] Tilman Michaeli 和 Ralf Romeike。2019年。提升课堂中的调试技能：教授系统化调试流程的效果。载于第*14*届基础与中等计算教育研讨会论文集。1–7。[27] Michael A Miljanovic 和 Jeremy S Bradbury。2017年。Robobug：一款用于学习调试技术的严肃游戏。载于*2017*年*ACM*国际计算教育研究会议论文集。93–100。[28] Monika AF Müllerburg。1983年。调试在软件工程环境中的作用。载于高级调试研讨会论文集。81–90。[29] Yannic Noller、Erick Chandra、Srinidhi Chandrashekar、Kenny Choo、Cyrille Jegourel、Oka Kurniawan 和 Christopher M Poskitt。2025年。模拟交互式调试。*arXiv* 预印本 *arXiv:2501.09694* (2025)。[30] Victor-Alexandru Padurean 、 Paul Denny 和 Adish Singla。2025年。BugSpotter：自动生成代码调试练习。载于第*56*届*ACM*计算机科学教育技术研讨会论文集 *V. 1*。896–902。[31] Samantha Boatright Smith、Heather Wei、Abby O'Neill、Aneesh Durai、John DeNero、JD Zamfirescu-Pereira 和 Narges Norouzi。2025年。识别人工智能失误：学生应对计算机科学导论中的大语言模型错误。载于第*56*届*ACM*计算机科学教育技术研讨会论文集 *V. 2*。1627–1628。[32] Jacqueline Whalley、Amber Settle 和 Andrew Luxton-Reilly。2021年。入门调试流程分析。载于第*23*届澳大拉西亚计算教育会议论文集。11–20。[33] Joseph B Wiggins、Fahmid M Fahid、Andrew Emerson、Madeline Hinckle、Andy Smith、Kristy Elizabeth Boyer、Bradford Mott、Eric Wiebe 和 James Lester。2021年。探索编程初学者在智能积木式编程环境中的提示请求。载于第*52*届*ACM*计算机科学教育技术研讨会论文集。52–58。[34] G Aaron Wilkin。2025年。《调试：从艺术到科学》：一项关于调试课程及其对学生表现和信心影响的案例研究。载于第*56*届*ACM*计算机科学教育技术研讨会论文集 *V. 1*。1225–1231。[35] Stephanie Yang、Miles Baird、Eleanor O'Rourke、Karen Brennan 和 Bertrand Schneider。2025年。解读调试教学：调试干预的系统性文献综述。《*ACM*计算教育汇刊》24, 4 (2025), 1–44。[36] Ziyao Zhang、Chong Wang、Yanlin Wang、Ensheng Shi、Yuchi Ma、Wanjun Zhong、Jiachi Chen、Mingzhi Mao 和 Zibin Zheng。2025年。实际代码生成中的大语言模型幻觉：现象、机制与缓解方法。《*ACM*软件工程论文集》2, ISSTA (2025), 481–503。