arXiv:2304.03938v1 [cs.CY] 8 Apr 2023

比较学生与大语言模型生成的代码解释

Juho Leinonen 奥克兰大学 奥克兰，新西兰
juho.leinonen@auckland.ac.nz

Paul Denny 奥克兰大学 奥克兰，新西兰
paul@cs.auckland.ac.nz

Stephen MacNeil 天普大学 费城，宾夕法尼亚州，美国
stephen.macneil@temple.edu

Sami Sarsa 阿尔托大学 埃斯波，芬兰
sami.sarsa@aalto.fi

Seth Bernstein 天普大学 费城，宾夕法尼亚州，美国
seth.bernstein@temple.edu

Joanne Kim 天普大学 费城，宾夕法尼亚州，美国
joanne.kim@temple.edu

Andrew Tran 天普大学 费城，宾夕法尼亚州，美国
andrew.tran10@temple.edu

Arto Hellas 阿尔托大学 埃斯波，芬兰
arto.hellas@aalto.fi

摘要

围绕代码进行推理并解释其用途，是计算机科学家的基本技能。计算教育领域已针对学生解释代码的能力与其他技能（如代码编写和跟踪）之间的关系开展了广泛研究。具体而言，以较高抽象程度描述代码在所有可能输入下将如何运行的能力，与代码编写能力高度相关。然而，对许多学生而言，培养准确、简洁地理解和解释代码的专业能力是一项挑战。现有的教学方法能够帮助学生逐步培养代码解释能力，例如按需生成代码解释范例，但目前难以有效扩展到大规模课堂。近年来，功能强大的大语言模型（LLM）不断涌现，或许可以提供一种解决方案。在本文中，我们探讨利用大语言模型生成解释的潜力，这些解释可作为示例，帮助学生培养理解和解释代码的能力。为了评估大语言模型生成的解释，我们将其与一门大型课程（$n \approx 1000$）中学生创建的解释进行比较，比较维度包括准确性、可理解性和长度。我们发现，大语言模型生成的解释可以按需自动生成；与学生创建的解释相比，这些解释在理解难度方面显著更低，并且是对代码更准确的总结。我们讨论了这一发现的重要意义，并提出了将此类模型融入入门编程教育的方法。

CCS 概念

• **社会与专业主题** → 计算教育； • **计算方法** → 自然语言生成。

关键词

自然语言生成、代码理解、GPT-3、计算机科学导论课程、代码解释、资源生成、大型语言模型

ACM 引用格式：
Juho Leinonen、PaulDenny、Stephen MacNeil、SamiSarsa、Seth Bernstein、Joanne Kim、Andrew Tran 和 Arto Hellas。2023。《学生与大语言模型生成的代码解释比较》。载于*2023* 年计算机科学教育创新与技术会议论文集第 *1* 卷（*ITiCSE 2023*），*2023* 年 *7* 月 *8–12* 日，芬兰图尔库。ACM，纽约州纽约市，8 页。https://doi.org/10.1145/3587102.3588785

允许免费制作本作品的部分或全部内容的数字或纸质副本，供个人或课堂使用，但前提是不得出于营利或商业目的复制或分发这些副本，并且副本须包含本声明及首页上的完整引文。必须尊重本作品中第三方组件的版权。其他用途请联系所有者/作者。*ITiCSE 2023*，*2023* 年 *7* 月 *8–12* 日，芬兰图尔库© 2023 版权所有归所有者/作者所有。ACM ISBN 979-8-4007-0138-2/23/07。
https://doi.org/10.1145/3587102.3588785

1 引言

理解和解释代码是计算机科学专业学生需要培养的一项重要技能 [7, 29, 44]。以往的计算教育研究通常认为，初学者在掌握较低层次的代码跟踪技能之后，才会逐渐具备解释代码的能力，而这也是培养较高层次代码编写能力的前提 [22, 37]。毕业后，学生还需要在求职面试中向招聘经理解释自己的代码，在新团队成员入职时向同事解释代码，并在刚开始使用新的代码库时向自己解释代码。然而，学生往往难以解释自己编写的代码，而解释代码本身也是一项难以培养的技能 [20, 38]。此外，计算机科学课程并不总是将解释代码明确列为学习目标，这进一步加剧了这些挑战。

示例学习是一种有效的教学方法，常用于编程教育 [2, 48]。然而，教师为某些类型的资源（例如代码解释）编写优质示例可能会耗费大量时间。虽然可以采用学习者众包技术，让学生直接参与代码解释的编写，从而高效生成代码解释 [21, 32],，但由学生提供学习内容时，已知存在质量方面的问题 [1, 9]。为寻找解决这一问题的方法，研究人员探索了“机器人众包”（即使用基于人工智能的生成器创建内容，或为人类创建内容提供支持）的学习材料潜力 [11, 36], ，包括代码

解释 [24, 25]。在这一阶段，人们对人工智能生成的代码解释与教师或学生编写的代码解释相比质量如何，以及它们是否可以替代其中任何一种，知之甚少。

我们比较了学习者撰写的代码解释与机器人生成的代码解释的质量，以考察大语言模型（LLM）生成解释、供学生作为学习示例使用的潜力。我们使用大语言模型为三个函数生成代码解释，并请学生为相同的函数撰写解释。随后，我们测量了学生对这两种来源的解释质量的看法。为帮助解读结果，我们请学生说明他们认为最有用的代码解释具备哪些特点。本研究围绕以下两个研究问题展开：

RQ1 学生撰写的代码解释与大语言模型生成的代码解释在准确性、长度和可理解性方面有何差异？RQ2 学生看重代码解释的哪些方面？

我们的结果表明，大语言模型生成的代码解释与学生撰写的代码解释在理想长度方面相当，但大语言模型生成的解释被认为更准确、更易理解。尽管积极参与撰写自己的解释对学生有益，我们仍得出结论：在早期的示例学习情境中，大语言模型生成的解释可作为学生的优质示例，也可以成为学习者撰写的代码解释的一种可行替代方案。

2 相关工作

2.1 代码理解

代码理解技能对于帮助编程学生理解代码片段背后的逻辑和功能至关重要 [39]。程序员可以采用各种代码理解策略，从而灵活地理解编程概念 [43]。这些策略包括跟踪执行过程 [6], 、解释 [31], 和概念机器 [15]。这些策略需要时间，而且对不同学生的效果各异 [17]。无论如何，学生在尝试理解代码时都可能遇到障碍，包括逻辑错误 [12] 和语法错误 [10] 。

自顶向下学习和自底向上学习是两种学习方法，分别侧重于整体概况和具体细节 [47]。自顶向下学习从高层次概念入手，逐步深入具体细节；而自底向上学习则从细节开始，逐渐建立对高层次概念的理解 [40]。在教授复杂主题时，这两种方法都很有用，因为学习者可以通过理解各个部分来掌握整体概念。在计算机科学和编程领域，这两种方法可用于帮助学习者掌握编码和编程的基础知识 [34]。

2.2 代码解释的教学效益

解释是对学生至关重要的教学资源。解释有助于学生理解代码片段如何执行 [27], ，进而帮助学生提高编写自己代码时的推理能力 [29]。解释还能通过拆解复杂概念来减轻压力 [14]。

早期的代码解释方法，例如 BRACElet 项目，会向学生提供“用通俗英语解释”类型的问题，鼓励学生从更高的抽象程度说明其代码的用途 [46]。解释自己的代码这一过程能为学生带来短期和长期的学习收益 [29, 42]。在大型课堂中，解释代码也可以是一种协作活动，学生之间相互解释代码。这一过程也可以更加非正式，例如结对编程：学生在编写代码时，向搭档解释自己的代码和思维过程 [16]。

尽管解释代码是一项重要技能，且先前的研究已经探讨过代码解释任务，但学生很少有机会接触代码解释示例，尤其是由同伴编写的示例。随手可得的代码解释示例有助于学生接触代码解释，从而帮助他们学习解释自己的代码。由教师编写此类解释是一项耗时的工作。在大型课堂中，很难腾出时间为学生提供个性化解释 [41]。因此，研究能否借助大语言模型大规模生成此类解释，是一个值得关注的研究课题。

2.3 计算机科学教育中的大语言模型

近期，基于人工智能的代码生成模型不断涌现，在计算机教育研究领域引发了广泛关注 [3]。该领域的初步研究主要聚焦于评估这些模型解决入门课程中常见编程问题的表现。一项具有开创性的研究题为“机器人来了” [13], ，使用了 Codex 模型和一个私人编程问题库，其中的问题取自高风险总结性评估。研究结果表明，Codex 生成的解答在评估中的得分约为 80%；与历年课程数据相比，这一表现超过了四分之三的学生。另一项使用编程问题公开数据集的类似研究发现，Codex 首次尝试时约有一半的概率生成正确解答；若允许重复尝试并对输入提示进行少量调整，这一比例可提高至 80% [8]。

除了评估性能外，另有一系列研究探讨了基于人工智能的代码生成模型在生成学习资源方面的潜力。例如，Sarsa 等人探索了各种提示和方法，以使用 Codex 模型生成代码解释和编程练习，并发现该模型经常能产出新颖且高质量的资源 [36]。然而，他们的评估仅由专家进行，并未涉及学生在实际情境中使用这些资源。MacNeil 等人使用 GPT-3 模型生成简短代码片段的解释，随后将这些解释与相应代码一同呈现在在线电子书中，供学生阅读 [24]。尽管他们的评估规模较小，参与者约有 50 人，但学生在选择阅读这些解释时，认为它们很有帮助。然而，正如作者所指出的，学生参与阅读的程度低于预期，而且学生并未参与代码示例或配套解释的创作。

本研究通过直接比较学生生成的代码解释与人工智能模型生成的代码解释，作出了独特贡献。此前的研究已表明，大语言模型能够生成被专家和初学者都认为质量较高的代码解释；而本研究首次探讨学生如何评价同伴生成的代码解释，并将其与人工智能模型生成的代码解释进行比较。

3 方法

3.1 背景与数据

本研究的数据收集自奥克兰大学的一门一年级编程课程。2022年开展本研究时，约有 1000 名学生选修了这门课程。

3.1.1 数据收集。 数据是在两次独立的实验课期间收集的，每次实验课持续一周。在第一次实验课（即开始收集数据时），课程已讲授 C 编程语言中的算术、类型、函数、循环和数组等概念。数据收集遵循了该大学的伦理准则。

在第一次实验课（Lab A）中，学生看到了三个函数定义，并被要求概述和解释每个函数的预期用途。在第二次实验课（Lab B）中，也就是第一次实验课两周后，学生看到了从 Lab A 中这些函数的代码解释里随机抽取的四条样本。其中一些代码解释选自学生在 Lab A 期间撰写的解释，另一些则由大型语言模型 GPT-3 [4] 生成。学生需要从准确性、可理解性和长度等方面对这些解释进行评分。在 Lab B 结束时，学生受邀以开放式回答以下问题：“现在你已经撰写并阅读了许多代码解释，请回答以下问题，谈谈你认为优秀代码解释最有用的特征：代码解释的哪些方面对你有用？”

图 1 列出了在 Lab A 中展示给学生的三个函数。每个函数都包含一个单独的循环，用于处理作为输入传递给函数的数组元素，并且函数名称能够体现所实现的算法。对于这三个函数，学生都被要求概述并解释函数的预期用途。具体来说，他们被要求：“查看函数名称、所用变量的名称以及函数实现的算法，并简短描述你认为该函数的预期用途”。

3.1.2 数据抽样。 图 2 概述了用于抽取 Lab B 所用代码解释的流程。参与 Lab A 代码解释生成工作的学生，针对三个函数中的每一个提交了 963 条解释。对于每个函数，我们根据词数将代码解释分为三类：第 10 百分位、第 10 至第 90 百分位以及第 90 百分位。从这三类中分别随机选取三条解释，因此每个函数有九条解释。在这 27 条学生生成的解释之外，我们还加入了 GPT-3 编写的 27 条解释，即针对三个函数中的每一个生成九条解释。对于 Lab B，每位学生都会从 54 条解释中随机看到四条。他们需要根据以下三个问题对每条解释进行评分（每题采用 5 分制）：

- 这段解释很容易理解（5 项：非常不同意、不同意、中立、同意、非常同意）
- 这段解释准确概述了代码（5 项：非常不同意、不同意、中立、同意、非常同意）
- 这段解释的长度恰到好处（5 项：太短很多、有点太短、恰到好处、有点太长、太长很多）

3.1.3 分析。为回答 RQ1 并量化学生与大语言模型生成的代码解释之间的差异，我们比较了学生对两种来源的代码解释所对应李克特量表问题的回答。

由于李克特量表的回答数据属于有序数据，我们采用非参数曼-惠特尼U检验 [26] 来检验学生与大语言模型代码解释在李克特量表问题数据方面是否存在差异。我们检验了：（1）代码解释是否易于理解；（2）代码解释是否准确概括了代码；（3）代码解释的长度是否理想。此外，我们还（4）考察了代码解释的实际长度，以此作为基线，判断学生与 GPT-3 生成的代码解释在长度上是否存在差异，这有助于解读其他研究发现。

我们总共进行了四次曼-惠特尼U检验。为处理多重检验问题，我们采用经 Bonferroni 校正的 $p < 0.05/4$ 作为统计显著性的阈值。遵循 [45] 的指南以及 [35], 中更广泛的讨论，我们仅将 p 值作为一项证据来源；在呈现研究结果时，还列出包括秩二列相关（RBC） [19] 和共同语言效应量（CLES） [28] 在内的支持性统计量。

为回答RQ2，即考察学生重视代码解释的哪些方面，我们对随机选取的100份学生针对开放式问题“代码解释具备哪些特点时会对你有用？”的回答进行了主题分析。

4 结果

4.1 描述性统计

共有954名学生参与了评估代码解释质量的活动。表1展示了回答的均值和中位数，其中李克特量表回答已转换为数值；表中还列出了学生编写的代码解释和大语言模型生成的代码解释的平均长度。

图3进一步展示了回答的分布，分别用不同颜色标示各类回答，以便直观比较不同回答值；表1中的数值概览则对此作了补充。

4.2 学生生成与大语言模型生成的代码解释在质量上的差异

我们进行了曼-惠特尼U检验，以研究学生生成的代码解释与大语言模型生成的代码解释之间是否存在差异。我们采用

```
int 最大值(int values[]，int length)
{
    int i, max;

    max = values[0];
    for (i = 1; i < length; i++) {
        if (values[i] > max) {
            最大值 = values[i];
        }
    }

    返回 max;
}
```

```
int 零值计数(int values[]，int length)
{
    int i, count;

    count = 0;
    for (i = 0; i < length; i++) {
        if (values[i] == 0) {
            计数++;
        }
    }

    返回计数;
}
```

```
双精度型 负值平均值(整数 values[]，整数 length)
{
    整数 i, 总和, 计数;
    i = 0;
    总和 = 0;
    计数 = 0;

    当 (i < length) {
        如果 (values[i] < 0) {
            总和 = 总和 + values[i];
            计数++;
        }
        i++;
    }

    返回 (double)sum / count;
}
```

图1：向 Lab A 的学生展示的三个函数定义。学生需要简要描述每个函数的预期用途。

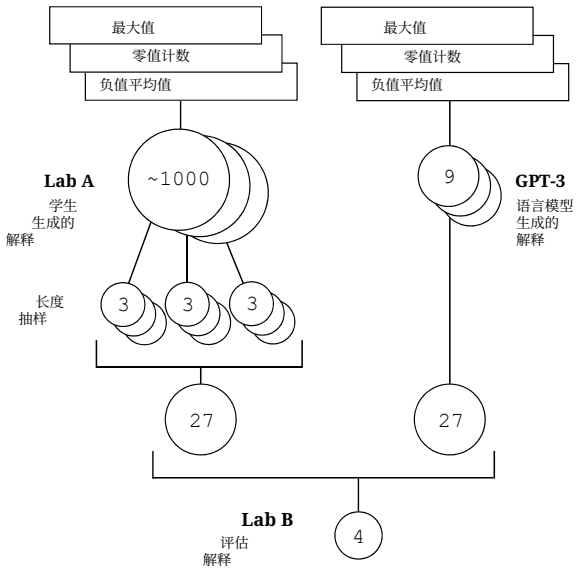


图2：代码解释的生成与抽样概览。在 Lab B 中，每位学生分配到四条代码解释进行评估，这些解释从共54条代码解释中随机选出（其中一半由 Lab A 的学生生成，另一半由 GPT-3 生成。）

双侧检验，评估代码解释在易于理解、准确概述所示代码以及长度是否恰当方面是否存在差异。此外，我们还比较了代码解释的实际长度（字符数）。

统计检验的结果汇总于表2。总体而言，学生撰写的代码解释与大语言模型生成的代码解释在易于理解程度以及是否准确概述代码方面存在统计学上的显著差异。根据邦费罗尼校正，对于代码解释是否长度适宜这一学生感知方面，差异无统计学意义；代码解释的实际长度也无统计学上的显著差异。

表1：学生对代码解释质量的回答的描述性统计。使用李克特量表给出的回答已转换为：1对应“强烈不同意”，5对应“强烈同意”。

	学生生成的 均值	学生生成的 中位数	大语言模型生成的 均值	大语言模型生成的 中位数
易于理解	3.75	4.0	4.12	4.0
准确的总结	3.78	4.0	4.0	4.0
理想长度	2.75	3.0	2.66	3.0
长度（字符数）	811	738	760	731

表2：学生与大语言模型撰写的代码解释质量双侧比较的曼-惠特尼U检验结果。$U-val$ 表示曼-惠特尼U检验的U值，p 列出回答来自该分布的概率（未校正），RBC 表示数据的秩二列相关系数，$CLES$ 给出通俗语言效应量。

	U 值	p	RBC	$CLES$
易于理解	1421596.0	$3.010 \cdot 10^{-31}$	0.206	0.397
准确的总结	1568575.5	$2.866 \cdot 10^{-12}$	0.123	0.438
理想长度	1860870.5	0.025	-0.040	0.520
长度（字符数）	1779386.0	0.764	0.006	0.497

总体而言，根据表2对共同语言效应量（CLES）的解读，在学生生成和大语言模型生成的代码解释配对中，学生生成的代码解释更容易理解的比例约为40%，而大语言模型生成的代码解释更容易理解的比例约为60%。同样，在这些配对中，学生生成的代码解释总结得更准确的比例约为44%，而大语言模型生成的代码解释总结得更准确的比例约为56%。尽管这些差异在统计学上

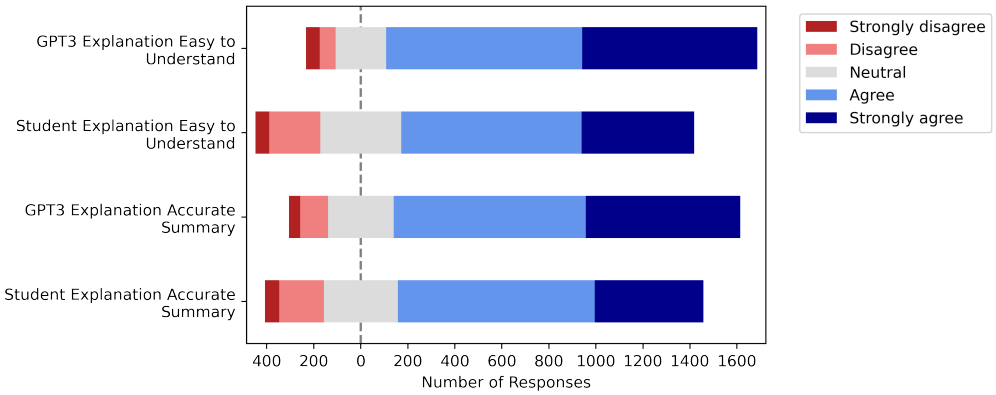


图3：学生对大语言模型生成和学生生成的代码解释易于理解且能准确概述代码的评价分布。

显著（如图3所示，这些差异也很明显），但效应量可视为较弱（$RBC < 0.3$）。

4.3 GPT-3和学生给出的优质与劣质解释示例

我们的定量结果表明，大语言模型生成的解释比学生生成的解释更准确、更易于理解。在关于学生重视代码解释哪些方面的主题分析中，我们发现，大语言模型生成的解释和学生生成的解释在不同主题特质上存在差异，包括准确性、完整性、抽象程度和所用专业词汇方面的差异。在一项开放式回答中，大多数学生将好的解释描述为“简洁地概述代码函数，说明其输入[和输出] 是什么”，展示“代码如何以编程术语运行，同时也用通俗易懂的英语解释代码试图实现什么”，并且“简洁明了，但篇幅仍然足以描述完整段代码”。

接下来，为了展示 GPT-3 和学生所写解释的类型，我们给出四个示例。这些示例在学生感知的质量方面有所不同，分别是针对 LargestValue 函数、由大语言模型和学生编写的解释中评分最高和最低的示例。示例根据准确性与可理解性的平均得分进行排序。

以下大语言模型生成的解释获得了最高评分（可理解性 = 4.2，准确性 = 4.5，长度 = 3.5）：

> “这段代码片段用于查找整数数组中的最大值。它首先假设数组中的第一个值就是最大值。接着，它遍历数组中剩余的值，并将每个值与当前最大值进行比较。如果某个值大于当前最大值，它就成为新的最大值。重复此过程，直到检查完数组中的所有值。最后返回最大值。”

以下大语言模型生成的解释评分最低（可理解性 = 4.1，准确性 = 3.7，长度 = 2.0）：

> “这段代码片段返回整数数组中的最大值。”

以下由学生撰写的解释获得了最高评价（可理解性 = 4.2，准确性 = 4.5，长度 = 3.6）：

> “函数 LargestValue 用于查找输入数组中的最大整数值，并以 int 类型输出该值。它接受两个输入：
>
> * values - 一个整数数组 * length - 该数组的长度（数组中的元素个数）
>
> 它将数组中的第一个整数值设为初始的“max”值（数组中的最大整数），然后从左到右扫描输入数组，检查是否存在更大的整数值。如果找到，就将其存入输出整数变量“max”中；函数结束时返回该值。”

以下学生生成的解释获得了最低评分（可理解性 = 3.3、准确性 = 3.3、长度 = 1.6）：

> “用于在数组中查找最大值”

学生更喜欢逐行格式的解释，这类解释既说明函数的相关信息，也描述函数的运作方式。许多学生还认同，好的解释应说明代码的输入和输出。

糟糕的解释通常会遗漏代码中的某些细节，同时篇幅过长或过短。例如，解释可能只笼统说明代码的用途，却没有详细说明使用了哪些数据结构，或传递给函数的输入是什么。

有趣的是，我们发现所有大语言模型生成的解释都以“这段代码片段”或“这段代码片段的目的”开头，而学生撰写的解释则更加多样。这在一定程度上是因为给大语言模型的提示要求它解释“以下代码片段”的用途。不过，学生和大语言模型给出的大多数解释

通常都遵循相似的结构：函数的用途、对代码的分析，最后是返回的输出。

4.4 代码解释的特征

在主题分析（n=100）中，我们发现学生在关注代码的具体方面（n=57）和一般方面（n=65）的人数上几乎各占一半，也有一些学生的回答同时涉及这两方面。关注代码具体方面时，学生认为需要逐行解释（21%）。学生还关注更细节的内容，如变量名称、输入和输出参数（36%）以及术语定义（8%）。一些学生还要求补充代码解释中很少包含的内容。例如，学生希望提供示例、模板，以及代码编写背后的思维过程。

学生详细谈到了优质解释应具备的特点。长度是一个重要方面，40%的学生明确提到了解释的长度。不过，对于理想解释的具体长度，并没有形成明确共识。相反，学生的评论往往关注效率，也就是用尽可能少的文字传达尽可能多的信息。即使简短的解释切中要点，可能也会受到教师青睐，学生似乎仍会给它较低的评价。这可能部分是因为这类解释几乎没有提供额外信息，或提供的信息并未超出函数本身已显而易见的内容，例如函数名称。学生作为初学者，可能更喜欢内容详细的解释，因为这有助于他们更好地学习和理解代码实际执行的过程。

5 讨论

5.1 学生与大语言模型生成的代码解释之间的差异

GitHub Copilot 等类似工具将关注重点从编写代码转向理解代码的用途、评估生成的代码是否合适，以及根据需要修改代码，从而使代码理解成为一项更加重要的技能。不过，大语言模型不仅可以帮助学生生成代码，也可以通过生成代码解释来帮助他们理解代码，而这些解释可以用作代码理解练习。

我们发现，与学生生成的代码解释相比，GPT-3 生成的代码解释在可理解性和准确性方面的平均评分更高。这表明，大语言模型生成的代码解释可以作为课程示例，帮助学生学习阅读代码。学生和大语言模型生成的代码解释在感知长度和实际长度上均无差异，因此评分提高并非源于大语言模型生成了更长（或更短）的解释。

我们认为，大语言模型生成的代码解释可以为处于这一阶段的学生提供有益的支架支持：他们已经能够理解大语言模型生成的代码解释，但还没有足够的能力自行编写代码解释。大语言模型生成的代码解释也可以作为示例，帮助学生撰写自己的代码解释。

先前研究提到的一个缺点是，学生可能会过度依赖大语言模型的支持 [5, 13]。减少学生过度依赖大语言模型生成的代码解释的一种方法，是监控学生对此类支持的使用情况（例如，为学生设定有限的令牌数量 [30]），学生向大语言模型请求解释时会消耗这些令牌），以限制学生对这些工具的使用或依赖。例如，学生可以先获得固定数量的令牌，并在请求解释时消耗令牌，然后通过撰写自己的代码解释来赚取令牌。

5.2 学生看重代码解释的哪些方面?

在主题分析中，我们发现学生表示更偏好逐行解释。这似乎也是大语言模型最擅长生成的解释类型 [36]。这一发现有些出乎意料，因为以往关于“用通俗英语解释”代码解释任务的研究，通常认为“关系型”回答——即对代码用途进行简短、抽象的描述——优于“多结构型”回答，也就是逐行解释。这表明，教师和学生对什么样的解释才算好的看法可能并不一致。甚至有可能，以往的一些研究“不公平地”降低了学生多结构型解释的评分，因为学生或许有能力给出更抽象的关系型解释，却认为更长、更详细的解释“更好”，因此选择了这类解释。

在主题分析中，我们还观察到，大语言模型生成的解释严格遵循一种标准格式。向学生展示大语言模型生成的解释，或许能帮助他们在自己的解释中采用标准格式，从而有助于生成更好的解释。这与以往的研究类似：这些研究表明，模板可以帮助设计师更好地界定问题 [23]，也能帮助作者写出更好的电子邮件 [18]。

5.3 局限性

我们的研究存在一些局限性，下面将逐一说明。首先，就可推广性而言，本研究中的学生都是初学者。这可能会影响他们所创作的解释类型，也可能影响他们对同伴和GPT-3所创作解释的评价。例如，先前研究发现，学生和教师对学习者创作的编程练习的评价存在差异 [33]。有可能（甚至很可能）更高年级的学生或教师能够创作出评价高于GPT-3所创作解释的代码解释。初学者重视的解释类型也可能与更有经验的学生不同：例如，随着经验增加，学生可能会开始更看重抽象程度更高、篇幅更短的解释。

关于被解释的代码，本研究只向学生提供了正确代码。未来一个值得探讨的方向是评估学生和大语言模型在解释错误代码以及检测其中程序错误方面的表现。研究中被解释的函数也相对简单。未来研究应探讨针对更多样、更复杂函数的解释。

在这项探索性研究中，我们仅考察了学生对代码解释质量的看法。未来的研究应探讨使用学生和大语言模型生成的代码解释是否会对学生学习产生不同影响。

我们承认，我们对数据进行了汇总分析，即一些学生可能只看到了大语言模型生成的解释，而另一些学生只看到了学生编写的解释。我们对

看过两个大语言模型生成的解释和两个学生编写的解释的学生数据进行了简要分析，发现其效果与本研究报告的结果相似，因此我们认为汇总所有学生的数据在方法上是合理的。

最后，我们使用了 GPT-3 的 davinci-text-002 版本。较新的版本 davinci-text-003 于 2022年11月 发布。使用较新的大语言模型，表现很可能至少相近，甚至更好。

6 结论

在这项工作中，我们开展了一项研究：学生先编写代码解释，随后评估同伴编写的代码解释以及 GPT-3 编写的代码解释。我们发现，学生对 GPT-3 编写的代码解释在准确性和可理解性两方面的评分都更高，尽管学生编写的代码解释与大语言模型编写的代码解释在感知长度和实际长度上均无差异。此外，我们还发现，学生更喜欢详尽的解释，而非简洁的高层次解释。

我们的结果表明，大语言模型生成的代码解释质量良好，因此可能有助于学生练习代码阅读和解释。我们认为，随着 GitHub Copilot 等基于大语言模型的人工智能代码生成器的出现，这些技能将变得更加重要，因为未来软件开发人员的工作将越来越多地转向评估大语言模型生成的源代码，而不是从头编写代码。

致谢

我们感谢 Ulla Tuominen 基金会向第一作者提供资助。

参考文献

[1] Solmaz Abdi、Hassan Khosravi、Shazia Sadiq 和 Gianluca Demartini。2021年。《学习资源质量评估：一种学习者众包方法》。*IEEE* 学习技术汇刊 14，1（2021年），81–92。[2] Siti-Soraya Abdul-Rahman 和 Benedict du Boulay。2014年。《通过示例讲解学习编程：学习风格与认知负荷的关系》。《计算机在人类行为中的应用》 30 2014 286 298 10 1016 2013 09 007（），–。[3] https://doi.org/。/j.chb.。。Brett A. Becker、Paul Denny、James Finnie-Ansley、Andrew Luxton-Reilly、James Prather 和 Eddie Antonio Santos。2023年。《编程很难——或者说，至少过去很难：人工智能代码生成带来的教育机遇与挑战》。载于第*54*届*ACM*计算机科学教育技术研讨会论文集，第*1*卷。ACM，500–506。[4] Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等。2020年。语言模型是少样本学习者。神经信息处理系统进展 33（2020年），1877–1901。[5] Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde de Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman 等。2021年。评估基于代码训练的大语言模型。*arXiv*预印本 *arXiv:2107.03374*（2021年）。[6] Bas Cornelissen、Andy Zaidman 和 Arie van Deursen。2011年。通过轨迹可视化开展程序理解的对照实验。*IEEE* 软件工程汇刊 37，3（2011年），341–355。[7] Kathryn Cunningham、Yike Qiao、Alex Feng 和 Eleanor O'Rourke。2022年。将"高层次"落到实处：明确对话式程序员学习目标。载于第*53*届*ACM*计算机科学教育技术研讨会论文集，第*1*卷（美国罗得岛州普罗维登斯）（*SIGCSE 2022*）。ACM，551–557。[8] Paul Denny、Viraj Kumar 和 Nasser Giacaman。2023年。与 Copilot 对话：探索使用自然语言解决计算机科学导论课程问题的提示工程。载于第*54*届*ACM*计算机科学教育技术研讨会论文集，第*1*卷。1136–1142。[9] Paul Denny、Andrew Luxton-Reilly 和 Beth Simon。2009年。使用 PeerWise 评估学生贡献问题的质量。载于第十一届澳大拉西亚计算教育会议论文集——第*95*卷（新西兰惠灵顿）（*ACE '09*）。澳大利亚计算机学会，AUS，55–63。[10] Paul Denny、Andrew Luxton-Reilly 和 Ewan Tempero。2012年。并非所有语法错误都相同。载于第*17*届*ACM*创新与技术年会论文集 A 计算机科学教育的以使用海法）（*ITiCSE '12*） . 75 80 10 1145 2325296 2325318 – 。https://doi.org/ 。/ 。[11] Paul Denny、Sami Sarsa、Arto Hellas 和 Juho Leinonen。2022年。众包教育资源——利用大语言模型开展学习者众包。arXiv 预印本 arXiv:2211.04715（2022年）。[12] Andrew Ettles、Andrew Luxton-Reilly 和 Paul Denny。2018年。初学者程序员常犯的逻辑错误。载于第20届澳大拉西亚计算教育会议论文集。83–89。[13] James Finnie-Ansley、Paul Denny、Brett A. Becker、Andrew Luxton-Reilly 和 James Prather。2022年。机器人即将到来：探索 OpenAI Codex 对入门编程的影响。载于澳大拉西亚计算教育会议。ACM，10–19。[14] Jean M. Griffin。2016年。通过拆解来学习：通过阅读、跟踪和调试来解构代码。载于第17届信息技术教育年会论文集。ACM，148–153。[15] Philip J Guo。2013年。在线 Python 导师：面向计算机科学教育的可嵌入式网络程序可视化工具。载于第44届 ACM 计算机科学教育技术研讨会论文集。579–584。[16] Brian Hanks、Sue Fitzgerald、Renée McCauley、Laurie Murphy 和 Carol Zander。2011年。教育中的结对编程：文献综述。《计算机科学教育》 21, 2 2011 135 173 10 1080 08993408 2011 579808 （），–。[17] https://doi.org/。/。。Regina Hebig、Truong Ho-Quang、Rodi Jolak、Jan Schröder、Humberto Linero、Magnus Ågren 和 Salome Honest Maro。2020年。学生如何体验和评价软件理解技术？载于第28届程序理解国际会议论文集。425–435。[18] Julie S Hui、Darren Gergle 和 Elizabeth M Gerber。2018年。Introassist：支持撰写入门求助请求的工具。载于2018年 CHI 人机交互会议论文集。1–13。[19] Dave S Kerby。2014年。简单差值公式：一种教授非参数相关的方法。《综合心理学》3（2014），11–IT。[20] Teemu Lehtinen、Aleksi Lukkarinen 和 Lassi Haaranen。2021年。学生难以解释自己的程序代码。载于第26届 ACM 创新与计算机科学教育技术会议论文集，第1卷。ACM，206–212。[21] Juho Leinonen、Nea Pirttinen 和 Arto Hellas。2020年。为 SQL 练习众包内容创作。载于2020年 ACM 创新与计算机科学教育技术会议论文集。349–355。[22] Raymond Lister、Colin Fidge 和 Donna Teague。2009年。关于入门编程中解释、跟踪与写作技能之间关系的进一步证据。SIGCSE Bull. 41, 3（2009），161–165。[23] Stephen MacNeil、Zijian Ding、Kexin Quan、Thomas j Parashos、Yajie Sun 和 Steven P Dow。2021年。创意工作的框定：通过交互式支架支持帮助初学者更好地界定问题。载于《创造力与认知》。1–10。[24] Stephen MacNeil、Andrew Tran、Arto Hellas、Joanne Kim、Sami Sarsa、Paul Denny、Seth Bernstein 和 Juho Leinonen。2023年。在网页软件开发电子书中使用大语言模型生成的代码解释的经验。载于第54届 ACM 计算机科学教育技术研讨会论文集，第1卷。931–937。[25] Stephen MacNeil、Andrew Tran、Dan Mogil、Seth Bernstein、Erin Ross 和 Ziheng Huang。2022年。使用 GPT-3 大语言模型生成多样化的代码解释。载于2022年 ACM 国际计算教育研究会议论文集，第2卷。ACM，37–39。[26] Henry B Mann 和 Donald R Whitney。1947年。关于检验两个随机变量之一是否在随机意义上大于另一个。《数学统计年鉴》（1947年），50–60。[27] Samiha Marwan、Nicholas Lytle、Joseph Jay Williams 和 Thomas Price。2019年。在初学者编程环境中为下一步提示添加文本解释的影响。载于2019年 ACM 创新与计算机科学教育技术会议论文集。ACM，520–526。[28] Kenneth O McGraw 和 Seok P Wong。1992年。一种通用语言效应量统计量。《心理学通报》111, 2（1992），361。[29] Laurie Murphy、Sue Fitzgerald、Raymond Lister 和 Renée McCauley。2012年。用简明英语解释的能力与计算机编程熟练程度相关。载于第九届国际计算教育研究会议论文集。ACM，111–118。[30] Henrik Nygren、Juho Leinonen、Nea Pirttinen、Antti Leinonen 和 Arto Hellas。2019年。将示例解答作为支持机制进行实验。载于第1届英国与爱尔兰计算教育研究会议论文集。1–7。[31] Steve Oney、Christopher Brooks 和 Paul Resnick。2018年。使用 Chat.Codes 创建引导式代码解释。《ACM人机交互会议论文集》2，CSCW，文章131（2018年11月），20页。https://doi.org/10.1145/3274400[32] Nea Pirttinen、Vilma Kangas、Irene Nikkarinen、Henrik Nygren、Juho Leinonen 和 Arto Hellas。2018年。通过 Crowd-Sorcerer 众包编程作业。载于第23届 ACM 创新与计算机科学教育技术年会论文集。326–331。[33] Nea Pirttinen 和 Juho Leinonen。2022年。学生能否评审同伴？同伴评审与教师评审的比较。载于第27届 ACM 创新与计算机科学教育技术会议论文集，第1卷。

[34] Margaret M. Reek。1995年。自顶向下方法教授编程。载于第二十六届*SIGCSE*计算机科学教育技术研讨会论文集。ACM，6–9。[35] Kate Sanders、Judy Sheard、Brett A Becker、Anna Eckerdal 和 Sally Hamouda。2019年。计算机教育研究中的推断统计：方法论综述。载于*2019*年 *ACM* 国际计算教育研究会议论文集。177–185。[36] Sami Sarsa、Paul Denny、Arto Hellas 和 Juho Leinonen。2022年。使用大语言模型自动生成编程练习和代码解释。载于*2022*年 *ACM* 国际计算教育研究会议论文集，第*1*卷。ACM，27–43。[37] Judy Sheard、Angela Carbone、Raymond Lister、Beth Simon、Errol Thompson 和 Jacqueline L. Whalley。2008年。使用 SOLO 评估初学者程序员。载于第*13*届创新与计算机科学教育技术年会论文集。ACM，209–213。[38] Simon 和 Susan Snowdon。2011年。解释程序代码：给学生答案有所帮助，但帮助有限。载于第七届计算教育研究国际研讨会论文集。ACM，93–100。[39] Leigh Ann Sudol-DeLyser、Mark Stehlik 和 Sharon Carver。2012年。代码理解问题作为学习事件。载于第*17*届 *ACM* 创新与计算机科学教育技术年会论文集。ACM，81–86。[40] Ron Sun、Edward Merrill 和 Todd Peterson。2000年。通过自底向上学习获取知识。知识型系统（2000年），249–291。[41] Zahid Ullah、Adidah Lajis、Mona Jamjoom、Abdulrahman Altalhi、Abdullah Al-Ghamdi 和 Farrukh Saleem。2018年。自动评测对初学者编程的影响：现有系统的优势与局限。计算机工程教育应用26, 6 (2018), 2328–2341. [42] Arto Vihavainen、Craig S Miller 和 Amber Settle。2015。入门编程中自我解释的益处。载于第*46*届*ACM*计算机科学教育技术研讨会论文集。284–289. [43] A. Von Mayrhauser 和 A.M. Vans。1995。软件维护与演进过程中的程序理解。计算机28, 8 (1995), 44–55. [44] Wengran Wang、Yudong Rao、Rui Zhi、Samiha Marwan、Ge Gao 和 Thomas W. Price。2020。Step Tutor：通过循序渐进的示例反馈支持学生。载于*2020*年*ACM*计算机科学教育创新与技术会议论文集。ACM，391–397. [45] Ronald L Wasserstein 和 Nicole A Lazar。2016。ASA关于p值的声明：背景、过程与目的。《美国统计学家》70, 2 (2016), 129–133. [46] Jacqueline L. Whalley、Raymond Lister、Errol Thompson、Tony Clear、Phil Robbins、P. K. Ajith Kumar 和 Christine Prasad。2006。关于初学者程序员阅读与理解技能的澳大拉西亚研究：采用布鲁姆分类法和SOLO分类法。载于第*8*届澳大拉西亚计算教育会议论文集——第*52*卷。澳大利亚计算机学会，澳大利亚，243–252. [47] Honglin Wu、Fu Zhang、Jingwei Cheng 和 Ke Wang。2019/11。采用自底向上方法确定教学内容。载于第二届人文教育与社会科学国际会议（*ICHESS 2019*）论文集。Atlantis Press，597–600. [48] Rui Zhi、Thomas W. Price、Samiha Marwan、Alexandra Milliken、Tiffany Barnes 和 Min Chi。2019。探索演示例题在初学者编程环境中的影响。载于第*50*届*ACM*计算机科学教育技术研讨会论文集。ACM，98–104.