软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(6):2536−2558 [doi: 10.13328/j.cnki.jos.007324] [CSTR: 32375.14.jos.007324]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

大模型生成代码的开源许可证违规风险洞察与分析*

王毅博[1], 王 莹[1], 余 跃[2], 许 畅[3,4], 于 海[1], 朱志良[1]

[1](东北大学 软件学院, 辽宁 沈阳 110169)

[2](国防科技大学 计算机学院, 湖南 长沙 410073)

[3](南京大学 计算机科学与技术系, 江苏 南京 210046)

[4](计算机软件新技术国家重点实验室 (南京大学), 江苏 南京 210046)

通信作者: 王莹, E-mail: wangying@swc.neu.edu.cn

摘 要: 大型语言模型的快速发展极大地影响了软件工程领域. 这些模型利用大量开源仓库代码进行预训练, 能够高效完成诸如代码生成和代码补全等任务. 然而, 开源软件仓库中存在大量受开源许可证约束的代码, 这给大模型带来了潜在的开源许可证违规风险. 聚焦于大模型生成代码与开源仓库的许可证违规风险, 基于代码克隆技术开发一个支持大模型生成代码溯源与版权违规问题的检测框架. 针对 9 个主流代码大模型生成的 135 000 个 Python 代码, 利用该框架在开源社区中溯源并检测开源许可证兼容性. 通过实践调查 3 个研究问题来探究大模型代码生成对开源软件生态的影响: (1) 大模型生成的代码多大程度克隆于开源软件仓库? (2) 大模型生成的代码是否存在开源许可证违规风险? (3) 真实开源软件中包含的大模型生成代码是否存在开源许可证违规风险? 实验结果发现在使用功能描述和方法签名所生成的 43 130 和 65 900 个大于 6 行的 Python 代码中, 分别溯源到了 68.5% 和 60.9% 的代码存在克隆的开源代码片段. 其中 CodeParrot 和 CodeGen 系列模型的克隆比例最高, GPT-3.5-Turbo 最低. 其次, 92.7% 的通过功能描述生成的代码中没有开源许可证声明. 通过与溯源代码许可证进行对比, 81.8% 的代码存在开源许可证违规风险. 此外, 在收集到的 229 个 GitHub 平台开发者使用大模型生成的代码中, 有 136 个代码溯源了到开源代码片段, 其中 38 个为 Type1 和 Type2 克隆类型, 有 30 个存在开源许可证违规风险. 以问题报告的形式提交给开发者, 到目前为止, 得到了 8 位开发者的反馈.

关键词: 大语言模型; 开源许可证; 开源许可证冲突; 代码克隆; 代码搜索

中图法分类号: TP311

中文引用格式: 王毅博, 王莹, 余跃, 许畅, 于海, 朱志良. 大模型生成代码的开源许可证违规风险洞察与分析. 软件学报, 2025, 36(6): 2536–2558. http://www.jos.org.cn/1000-9825/7324.htm

英文引用格式: Wang YB, Wang Y, Yu Y, Xu C, Yu H, Zhu ZL. Insights and Analysis of Open-source License Violation Risks in LLMs Generated Code. Ruan Jian Xue Bao/Journal of Software, 2025, 36(6): 2536–2558 (in Chinese). http://www.jos.org.cn/1000-9825/7324.htm

Insights and Analysis of Open-source License Violation Risks in LLMs Generated Code

WANG Yi-Bo[1], WANG Ying[1], YU Yue[2], XU Chang[3,4], YU Hai[1], ZHU Zhi-Liang[1]

[1](Software College, Northeastern University, Shenyang 110169, China)

[2](College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

[3](Department of Computer Science and Technology, Nanjing University, Nanjing 210046, China)

[4](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210046, China)

* 基金项目: 国家自然科学基金 (61932021, 62141210); 111 项目 (B16009)
本文由“大模型下的软件质量保障”专题特约编辑王赞教授、王莹副教授、陈碧欢副教授、姚远副教授、张敏灵教授推荐.
收稿时间: 2024-08-24; 修改时间: 2024-10-14; 采用时间: 2024-11-25; jos 在线出版时间: 2024-12-10
CNKI 网络首发时间: 2025-03-13

Abstract: The field of software engineering has been significantly influenced by the rapid development of large language models (LLMs). These models, which are pre-trained with a vast amount of code from open-source repositories, are capable of efficiently accomplishing tasks such as code generation and code completion. However, a large number of codes in the open-source software repositories are constrained by open-source licenses, bringing potential open-source license violation risks to the large models. This study focuses on the license violation risks between code generated by LLMs and open-source repositories. A detection framework that supports the tracing of the source of code generated by large models and the identification of copyright infringement issues is developed based on code clone technology. For 135 000 Python codes generated by 9 mainstream code large models, the source is traced and the open-source license compatibility is detected in the open-source community by this framework. Through practical investigation of three research questions, the impact of large model code generation on the open-source software ecosystem is explored: (1) To what extent is the code generated by large models cloned from open-source software repositories? (2) Is there a risk of open-source license violations in the code generated by large models? (3) Is there a risk of open-source license violations in the large model-generated code included in real open-source software? The experimental results indicate that among the 43 130 and 65 900 python codes longer than six lines generated by using functional descriptions and method signatures, 68.5% and 60.9% of the codes respectively are traced to have cloned open-source code segments. The CodeParrot and CodeGen series models have the highest clone ratios, while GPT-3.5-Turbo has the lowest. Besides, 92.7% of the codes generated by using functional descriptions lack license declaration. By comparing with the licenses of the traced codes, 81.8% of the codes have open-source license violation risks. Furthermore, among 229 codes generated by LLMs collected from GitHub, 136 codes are traced to have open-source code segments, among which 38 are of Type1 and Type2 clone types, and 30 have open-source license violation risks. These issues are reported to the developers in the form of problem reports. Up to now, feedback has been received from eight developers.

Key words: large language model (LLM); open-source license; open-source license violation; code clone; code search

1 引 言

近年来, 大型语言模型彻底改变了自然语言处理领域[1,2]. 诸如 BERT[3] 和 GPT[4]系列模型在文本分类、机器翻译等各种自然语言任务中取得了显著成功[5]. 随着大规模开源数据集的发展, 例如 Google BigQuery[6], The Pile[7], The-Stack[8]等, 基于大量代码数据集预训练的大模型在软件工程任务也取得了显著进步. 这些模型能够高效地完成软件工程领域的任务, 比如代码生成、代码补全、代码重构、代码审查以及漏洞分析等, 成为软件工程领域的重要组成部分[9−13]. OpenAI 推出的 Codex[14]、GitHub 推出的 Copilot[15]、亚马逊推出的 CodeWhisperer[16]、MetaAI 推出的 CodeLLaMa[17]以及 Hugging Face 推出的 StarCoder[18]等工具都旨在通过智能代码建议和自动代码生成来提高开发人员的生产力.

尽管大模型带来了诸多便利, 但由于其预训练过程使用的数据集可能包含敏感的个人信息和受版权保护的内容, 这为大模型的隐私保护和合规性应用带来了严峻的挑战[19–21]. 研究发现, GPT-2 以及 Copilot 模型所生成的内容会泄露用户的电子邮件、社交媒体账号等个人隐私信息[22,23]. 在代码层面, 它们生成的代码内容也可能来自于开源社区的代码片段[24,25]. GitHub Copilot 曾因使用公共代码仓库训练而面临程序员的集体诉讼, 要求索赔 90 亿美元[26]. 诉讼指出, Copilot 生成的代码可能与开源社区的代码相似, 而这些代码受开源许可证的约束, 违反了许可证条款. 这引发了人们对大模型生成内容的担忧. 尤其是大模型可能会记忆并生成受开源许可证保护的代码片段, 而用户并不清楚许可证条款内容, 这可能会导致的许可证冲突问题.

图 1 中 Flask 框架开发者 Armin Ronacher 在使用 Copilot 补全 fast inverse square root float 代码时, 发现 Copilot 生成了与“雷神之锤”[27]源码完全一致的代码片段, 两个代码的注释, 变量名以及变量值完全一致. 更重要的是, “雷神之锤”[27]中的此代码片段受 GPL 许可证的约束, 然而 Copilot 却生成了 BSD 类型的开源许可证声明[28]. 在大模型辅助软件开发的背景下, 这种开源许可证冲突的问题变的更加复杂. 大模型生成的代码可能来自于开源社区的代码片段, 这些代码可能由各种类型的开源许可证所约束[29,30]. 因此, 研究大模型生成代码与开源社区内代码的开源许可证违规问题, 对于确保大模型技术的合法和合规应用具有重要意义.

现有研究尚未全面探究大模型生成代码的版权合规问题. 为填补这一空白, 本文基于代码克隆技术开发了一个支持大模型生成代码溯源与版权违规问题检测框架. 该框架针对大模型生成代码, 在开源软件仓库中追本溯源, 且通过溯源代码的开源许可证兼容性分析, 判断生成代码是否存在开源许可证违规问题.

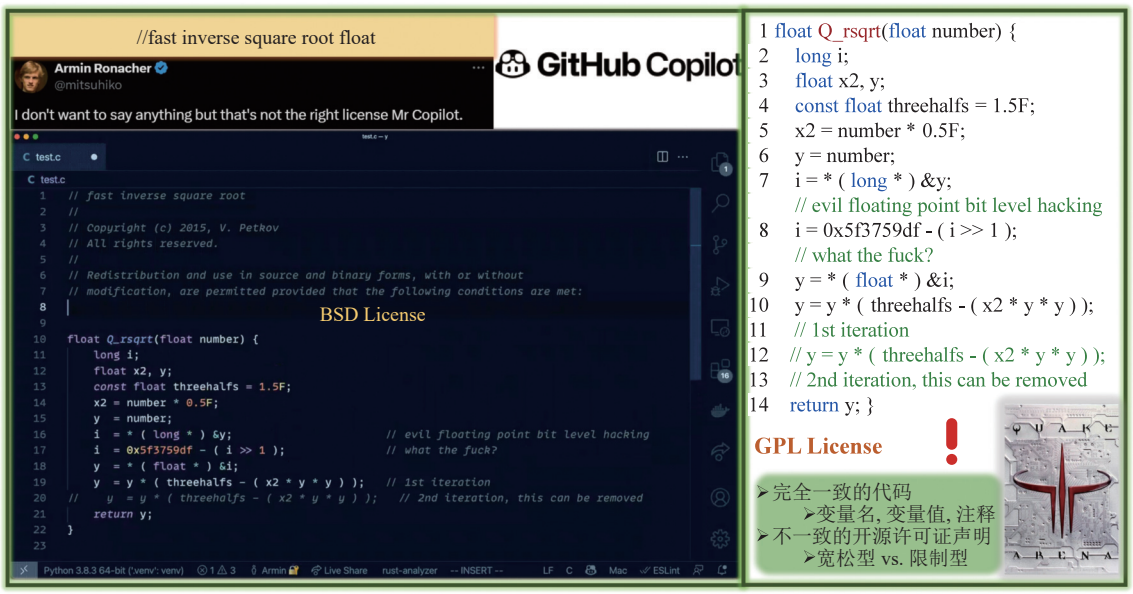


图 1 大模型生成代码克隆于开源代码示例

为了探究大模型生成代码的版权违规现象的普遍性与严重性, 我们对 9 个主流代码大模型生成的 135 000 个 Python 代码片段, 利用该框架在开源社区中追本溯源并进行开源许可证兼容性检测, 通过实践调查 3 个问题: “大模型生成的代码多大程度克隆于开源软件仓库?”“大模型生成的代码是否存在开源许可证违规风险?”“真实开源软件中包含的大模型生成代码是否存在开源许可证违规风险?”, 揭示大模型代码生成技术对开源软件生态的影响. 上述结果与发现有助于引起软件从业者对生成式 AI 在版权违规风险方面的关注.

利用代码溯源与版权违规问题检测框架对大模型生成代码溯源分析, 我们发现:

(1) 在大模型生成的 135 000 个 Python 代码中, 有 109 030 个代码行数大于 6 行, 其中 70 542 个 (64.7%) 代码可以在开源仓库内溯源到相似的代码片段. 对于通过代码生成指令生成的代码, 平均有 68.5% 的代码可以溯源到存在克隆的开源代码片段、平均 2.2% 的代码可以溯源到 Type1 克隆类型的开源代码片段、平均 17.8% 代码可以溯源到 Type2 克隆类型的开源代码片段. 此外, CodeParrot 系列模型相比于其他大模型溯源了最多的代码, GPT-3.5-Turbo 溯源到的代码最少. 对于通过代码补全指令生成的代码, 平均有 60.9% 的代码可以溯源到存在克隆的开源代码片段.

(2) 通过人工审查的方式分类克隆开源代码片段内容, 我们发现大模型最频繁克隆的代码内容为方法的定义和实现. 此外我们也发现不同的大模型所克隆的代码内容存在明显差异. 特别是 CodeParrot 系列模型, 它克隆了开源许可证声明的文本描述, 这种现象在其他模型中较为罕见. 相比于其他大模型复制文档内容 (Doc String), GPT-3.5-Turbo 未检测到文档类型的代码克隆.

(3) 81.8% 的由大模型生成的代码存在许可证违规风险. 在 43 130 个通过方法描述生成的代码中, 仅 3 127 个代码包含开源许可证声明, GPT-3.5-Turbo 模型未生成任何开源许可证声明. 通过对比大模型生成代码与溯源代码之间的开源许可证声明, 我们发现大模型生成代码未声明开源许可证与溯源代码的宽松型许可证、弱限制型许可证以及限制型许可证组合占比分别为 51.9%、1.8% 以及 21.7%, 其中限制型许可证为 GPL-3.0, GPL-2.0 以及 AGPL-3.0.

(4) GitHub 社区中由大模型生成的代码同样存在开源许可证违规风险. 在收集到的 229 个由 ChatGPT 以及 Copilot 生成的代码中, 67.7% 的代码未声明开源许可证, 24.0% 的受宽松许可证约束, 8.3% 的代码受限制型许可证约束. 此外, 有 38 个代码存在 Type1 和 Type2 以及 98 个代码存在 Type3/4 克隆类型的代码片段. 在 38 个存在 Type1 和 Type2 克隆类型代码中, 有 30 个代码存在开源许可证违规风险. 通过以问题报告的形式提交给开发者,

有 3 位开发者私有了整个代码仓库, 2 位开发者删除了大模型生成的代码, 2 位开发者未更改代码, 1 位开发者重构了大模型生成的代码.

本文第 2 节介绍大模型以及大模型记忆训练数据相关研究背景. 第 3 节介绍大模型生成代码溯源与版权违规问题检测框架. 第 4 节详细介绍本文的研究问题以及实验设置. 第 5–7 节通过对大模型生成代码的分析重点回答本文提出的 3 个研究问题. 第 8 节对比相关研究工作. 第 9 节分析有效性威胁. 第 10 节总结全文.

2 研究背景

2.1 大模型

大模型通常具有数亿到数十亿的参数, 使用自监督学习技术在大量文本数据或者代码数据上进行训练, 通过对训练数据概率分布的学习, 大模型能够根据用户的输入生成高质量的文本. 大模型的出现不仅改变了自然语言处理的范式, 也在软件开发领域引发了革命性变革. 大模型通常基于 Transformer[31]架构构建, Transformer 架构在处理自然语言和代码任务时表现出色, 它能够有效捕捉序列数据中的长距离依赖关系. 在实现中, 基于 Transformer 架构的模型可分为 Decoder-only (如 CodeParrot[32], StarCoder[33]和 GPT-3[34]等)、Encoder-only (如 CodeBERT[3]等) 、Encoder-Decoder (如 CodeT5 和 CodeT5+[35]等) 类型. 大模型通过理解用户输入提示词, 可以生成语法正确且逻辑一致的代码片段. 以提示词 $X=(x_1,x_2,\ldots,x_n)$ 为例, 大模型根据该提示词生成目标文本 y. 提示词 X 可以是一个代码功能描述或者代码片段. 根据给定的输入提示词 X, 大模型计算生成目标文本 y 的条件概率:

$$p(y|X)=\prod_{t=1}^{T}p(y_t|x_1,x_2,\ldots,x_{t-1}) \tag{1}$$

其中, $p(y|X)$ 表示生成目标文本 y 的条件概率, T 表示生成目标文本的序列长度, y_t 表示目标文本的第 t 个标记, $x_1,x_2,\ldots,x_n$ 表示提示词 X 中的标记序列. 大模型通过迭代过程生成文本, 在每一步中, 基于当前的输入提示和已经生成的部分文本, 计算下一个标记的概率分布, 并从中采样以生成下一个标记, 直到完成整个文本生成过程.

2.2 大模型记忆训练数据

大模型记忆: 大模型的预训练数据集通常通过在互联网上收集大量的文本数据构建, 这些数据来源广泛, 包括书籍、文章、网站和社交媒体帖子等[7]. 尽管这种数据收集方法能够确保数据集的全面和多样性, 但数据集内可能包含隐私信息或者缺乏原始创作者同意的内容. 大模型会记住训练数据集中的具体实例, 并在生成输出时重复这些实例. 这种现象称之为大模型的记忆[19].

开源许可证合规性挑战: 大模型记忆能力对于性能提升具有一定的积极作用, 但也带来了许多挑战. 大模型记忆并重复的数据很可能包含用户隐私数据, 甚至涉及版权内容. 图 1 中 Flask 的开发者暴露了 Copilot 生成的代码存在开源许可证违规风险. ChatGPT 也面临同样的风险, 在给定方法签名的提示下, ChatGPT 生成的代码[36]与 GitHub 平台的 wazuh 仓库的代码仅存在字符串常量值的差异, 仅将生成的代码中存在隐私泄露的字符串常量替换为了安全的常量, 但 wazuh 仓库的代码却受 GPL 许可证的约束[37]. 开源社区内的代码通常附有开源许可证的约束. 不同的开源许可证有不同的条款和约束, 确保代码的使用和再分发符合这些条款是开源软件开发的重要一环[38]. 如果大模型记住了一个特定开源许可证约束的代码片段, 并在生成代码时复现, 而开发者使用此代码时不清楚许可证条款, 这可能会导致开源许可证违规问题.

3 大模型生成代码溯源与版权违规问题检测框架

为了探究大模型生成代码与开源社区代码存在开源许可证违规问题, 我们基于代码克隆技术开发一个支持大模型生成代码溯源与版权违规问题检测的分析框架. 该框架结合代码搜索和代码克隆技术, 对大模型生成代码自动溯源与开源仓库中存在代码克隆的片段, 并进行开源许可证兼容性检测. 本文构建的开源仓库数据集内包含了上千万个代码文件, 现有的代码克隆检测方法在大规模数据集上面临检测速率慢以及内存消耗大的问题. Katzy 等人[21]实验发现 Deckard[39]在 2 万条代码的数据集上需要 7 h, Simian[40]需要 1 h. 然而 Deckard、Simian 以及

NiCad[41]在 40 万条代码数据集上均无法检测代码克隆.

图 2 展示了框架的整体架构, 包括代码搜索、代码克隆检测和开源许可证兼容性检测这 3 个阶段.

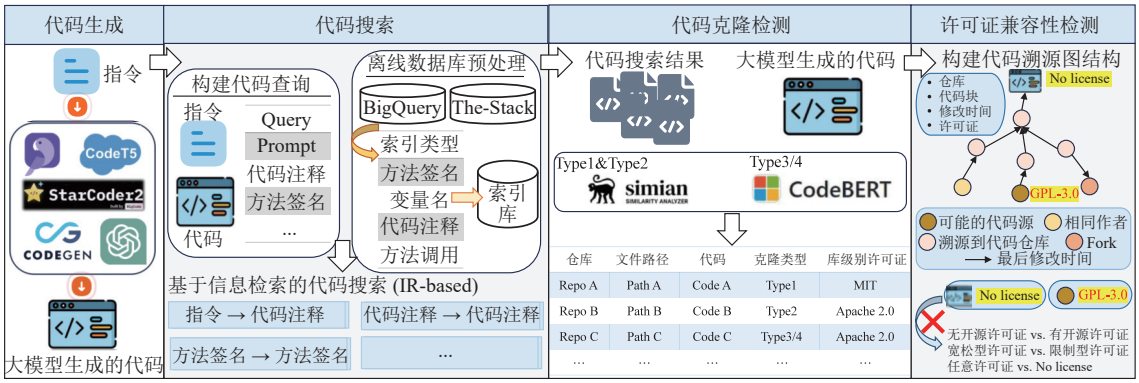


图 2 大模型生成代码的溯源以及开源许可证违规检测

3.1 数据预处理

我们使用 Apache Lucene[42]对开源仓库的代码片段构建索引, 以确保代码搜索的准确性和速度. Apache Lucene 是目前最流行的信息检索库之一, 它可以维护索引与实例之间的映射. 本文借助 Apache Lucene 维护了索引与代码片段、代码所在仓库、代码完整路径、库级别开源许可证信息之间的映射关系. 基于 Apache Lucene 的代码搜索在相关工作展现出了十分优异的性能, Kim 等人[43]通过构建的代码索引在 IJaDataset 数据集上达到了 90% 的召回率.

本文基于 The-Stack[44]和 GitHub-Code[45]开源数据集的大量代码构建代码索引, 两个数据集内包含了大量的 GitHub 平台公开代码仓库的 Python 代码. 尽管本文主要针对 Python 语言, 但本文所提出的大模型生成代码溯源与版权违规问题检测框架同样可以应用于其他编程语言, 为探究大模型生成代码与开源仓库之间许可证兼容性问题提供参考.

The-Stack 是由 Hugging Face 发布的超过 3TB 开源代码的数据集, 覆盖了 2015 年 1 月–2022 年 3 月的 GitHub 平台公开仓库代码, 主要以 Python 文件为主 (占比 96%). 不同的是, GitHub-Code 通过 Google BigQuery 所执行查询的结果构建. GitHub-Code 包含了约有 7 226 626 个 Python 文件. 两个数据集内共有 17 265 826 个 Python 代码文件, 现有代码克隆工具在该规模数据集内无法有效检测代码克隆. 数据集内都包含了代码片段、代码所在仓库、代码完整路径以及库级别开源许可证信息, 这使得预处理阶段能够更轻松地维护索引与代码的映射关系.

通过 tree-sitter[46]构建数据集内每个代码的抽象语法树 (AST), 提取索引信息作为索引列表. 这些索引信息可以定义为“索引类型: 索引内容”. 本文重点考虑了签名、代码注释、变量名和方法调用这 4 种索引类型[43]. 详细的索引类型以及对应的索引描述如表 1 所示.

表 1 代码索引示例

索引类型	索引描述
方法签名	代码内声明的方法签名
变量名	代码内声明的变量名称
代码注释	代码内的单行注释以及多行注释
方法调用	代码内所调用方法的 (非) 限定名

图 3 展示了代码片段和对应索引的示例. 通过对代码抽象语法树的构建以及解析, 预处理阶段可以提取方法签名、变量名、代码注释以及方法调用作为索引内容, 并将这些索引内容与对应的代码信息关联, 用于后续的代码搜索阶段.

```python
import tensorflow as tf
from tensorflow.keras import layers
# Define a simple feedforward neural network model
def create_model(input_shape, num_classes):
    # Creating a Sequential model
    # Adding a dense layer with ReLU activation
    model = tf.keras.Sequential([
        layers.Dense(128, activation='relu',
                     input_shape=input_shape),
        layers.Dense(num_classes,
                     activation='softmax')
    ])
    return model
```

索引创建

索引类型	索引值
方法签名	def create_model(input_shape, num_classes):
代码注释	Creating a Sequential model
变量名	input_shape, num_classes
方法调用	tensorflow.keras.Sequential

图 3 索引创建示例

3.2 代码搜索与代码克隆

基于构建完成的 The-Stack 以及 GitHub-Code 索引库, 本方法能够有效地检索与大模型生成代码相似的开源代码, 并借助代码克隆工具判断大模型生成代码与相似代码的代码克隆类型.

3.2.1 生成代码查询

我们把从大模型生成代码提取到的索引信息称之为代码查询. 针对由大模型生成的代码, 本节需要构建与第 3.1 节一致的索引, 这样能够更准确地匹配数据集预处理阶段的索引信息. 此外, 代码生成指令也将视为代码查询的一部分. 如果代码生成指令为自然语言描述的形式, 则将其视为代码注释类型的索引. 这些索引信息将以“索引类型: 索引内容”的形式用于代码搜索.

3.2.2 检索相似的开源代码片段

在代码查询创建完成之后, 每一个代码查询都会在预处理的开源代码仓库内检索相似的代码片段. 代码查询的格式和数据预处理阶段的格式都采用了“索引类型:索引内容”的形式, 这使得代码搜索可以借助 Apache Lucene 提供的全文搜索 (full-text search) 进行. 我们的方法根据衡量代码查询与索引值之间的相似性来排名相似代码. 在索引值相似性对比方面, 使用 Lucene 内已经实现的 BM25 算法来计算代码查询与索引值之间的相似程度. 在索引值相似性排序方面, 使用 Lucene 默认使用的相关性排序算法, 确保搜索结果按照代码查询与索引值之间的相似程度降序排列. 这意味着最相关的索引值将排在搜索结果的前面, 使得搜索过程更快速地找到相似的开源代码. 从检索到的相似代码列表中选择前 n 个代码作为当前代码查询的搜索结果. 在本文后续所有实验中, n 的值统一设置为 10.

3.2.3 检测大模型生成代码与相似代码的代码克隆类型

对于在代码搜索阶段检索到的相似的开源代码, 我们利用已有的代码克隆检测工具分类大模型生成代码与相似代码的克隆类型. 代码克隆通常用来衡量代码之间的相似程度, 主要包括以下 4 种类型:除了空格、注释以外完全相同的代码 (Type1); 除了变量名、类型名和函数名等之外完全相同的代码 (Type2); 两个代码之间存在语句的增删 (Type3); 两个代码具有相同的语义, 但句法语法结构不同 (Type4)[47].

我们使用 Simian[40]检测 Type1 和 Type2 类型的代码克隆. (1) Simian 工具支持 Python 语言的代码克隆检测, Ragkhitwetsagul 等人[48]实验结果表明 Simian 工具在 SOCO 数据集中达到了 97.8% 的准确度. 而 CCFinderX[49]以及 NiCad[41]等工具仅支持 Java 语言的代码克隆检测. (2) 更重要的是, 由于大模型生成时 token 数量的限制, 当要生成的代码超过 token 限制时, 模型会停止生成[50]. 这些代码称之为不完整代码, 它们无法生成抽象语法树以及无法编译[51]. Ragkhitwetsagul 等人[48]提到 Simian 能够有效检测不完整代码的克隆. 而 PMD/CPD[52]等工具不支持不完整代码的克隆检测[48]. Simian 基于文本相似性对比的代码的克隆程度, 通过将代码片段转换为抽象表示形式, 使用文本匹配技术计算相似度得分. 它允许设置相似度阈值等检测参数, 并生成详细的克隆检测报告, 包括克隆代码的位置、代码克隆类型等信息. 在本文的后续实验中, 我们采用代码克隆粒度为大于等于 6 行的代码块. 这种设置在相关工作中广泛应用, 并且表现出良好的检测效果[24,25].

我们使用 SSCD[53]检测 Type3/4 类型的代码克隆. Chochlov 等人[53]的实验结果表明 SSCD 在 Company-C 数

据集可以达到 92% 的准确率, 同时该模型权重开源在了 Hugging Face 平台. SSCD 在 CodeBERT 的基础上添加了池化层, 能够对输入代码生成 768 维的向量表示, 用于比较不同代码片段之间的相似性. 本文通过本地部署 SSCD 模型, 对代码进行代码块级别的 768 维向量生成. 通过计算不同代码块之间的余弦相似度来判断是否为 Type3/4 的克隆类型. 在本文后续实验中, 同样使用 6 行代码块的粒度, 余弦相似度阈值设定为 0.95. 当两个代码块之间的余弦相似度大于 0.95 时, 这两个代码块视为 Type3/4 的代码克隆. 这个阈值的设定参考已有利用余弦相似度检测代码克隆的工作, 并且表现出了较高的准确性[54,55].

3.3 开源许可证兼容性检测

3.3.1 提取开源许可证声明

我们首先在文件级别提取每个代码文件的开源许可证声明, 避免仅依赖库级别许可证信息导致的不准确性. 这是因为尽管数据集预处理阶段存储了开源代码的库级别开源许可证信息, 但库级别的开源许可证声明与文件级别的开源许可证仍然存在不一致的情况[56]. 如果文件级别未识别到开源许可证声明, 则使用库级别的许可证声明.

我们使用 Scancode[57]提取文件级别的开源许可证声明, 并采用 SPDX[58]所罗列的开源许可证进行统一命名. Scancode 通过解析代码文件顶部的信息, 将其与已知的许可证进行匹配并输出结果. 然而 Scancode 工具的输出与开源仓库内开源许可证存在命名的差异. 为了解决这一问题, 我们采用 SPDX 所罗列的开源许可证进行统一命名. SPDX 提供了一系列常用的开源许可证命名, 可以更规范的约束开源许可证的使用流程[58].

3.3.2 构建开源许可证兼容性检测图

我们通过提取开源代码的最后修改时间确定溯源代码[59]. 由于与大模型生成代码存在克隆的开源代码可能来源于不止一个代码仓库, 对溯源代码的提取可以更准确地检测开源许可证的违规风险. 我们利用 Git 实现这一功能. Git 中的 git blame 命令可以收集代码文件中每一行的最后修改的日期和时间. 区别于获取整个代码文件的修改时间, 这种方式可以获取在代码块级别的修改时间.

通过对克隆代码最后修改时间的提取, 本方法构建用了于开源许可证兼容性检测的图结构. 如图 2 所示, 图内的结点为存在代码克隆的代码片段, 图内的边为代码片段的最后修改时间. 每个结点都包含仓库名、代码路径、克隆代码片段以及开源许可证等信息. 通过构建的图 1 所示的结构, 本方法可以追溯最早修改的代码片段, 确定溯源代码. 后续将通过对比模型生成代码与溯源代码的开源许可证分类检测许可证兼容性问题.

3.3.3 检测开源许可证兼容性问题

与 Golubev 等人[59]方法一致, 开源许可证兼容性检测基于以下 3 种许可证分类: 宽松型许可证 (Permissive)、弱限制型许可证 (Weak Copyleft) 以及限制型许可证 (Strong Copyleft). Golubev 等人[59]通过划分开源许可证种类的方式, 探究了 GitHub 社区存在代码克隆的 Java 代码的许可证兼容性. 我们根据 DejaCode[60]提供的开源许可证分类, 判断大模型生成代码与溯源代码的开源许可证所属类别. 当大模型生成代码与溯源代码的开源许可证隶属于不同的许可证类别时, 它们更有可能存在许可证冲突问题[60]. 下列两种情况存在开源许可证违规风险.

(1) 无开源许可证与有开源许可证: 如果大模型生成的代码没有开源许可证, 但溯源代码有开源许可证, 则存在违规风险. 因为所有的开源许可证要求用到其内容的项目也存在开源许可证声明.

(2) 宽松型许可证与限制型许可证: 如果大模型生成的代码提取到了宽松型的开源许可证声明 (如 MIT), 但溯源代码提取到的为弱限制型或者限制型的开源许可证 (如 GPL), 则存在违规风险.

除了上述两种情况以外, 我们也考虑了溯源到的开源代码没有开源许可证的声明的情况. 与 Golubev 等人[59]一致, 这种情况同样视为可能的开源许可证违规风险.

4 实验设置

4.1 代码大模型

本文主要探究大模型生成的代码与开源社区代码存在开源许可证违规问题. 为了实现这一目的, 本文选择了多个不同参数规模的大模型进行实验, 包括 CodeParrot[32], CodeGen-mono[61], CodeT5+[35], StarCoder2[33]以及 GPT-

3.5-Turbo[34]系列.

(1) CodeParrot 系列模型基于 Decoder 架构, 它使用 Google BigQuery 数据集进行预训练, 有效地实现了 Python 代码的生成. 本文选择了 110M 以及 1.5B 两个参数规模的模型.

(2) CodeGen-mono 系列模型基于 Decoder 架构, 它在 CodeGen-multi 的基础上基于 Python 代码数据集进行预训练, 支持根据自然语言描述生成代码. 本文选择了 350M 和 2B 两个参数规模的模型.

(3) CodeT5+系列模型基于 Encoder-Decoder 架构, 使用了 BigQuery 数据集进行 Python 代码层面的预训练, 并支持自然语言生成代码. 本文选择了 220M, 770M 以及 2B 这 3 个不同参数规模的模型.

(4) StarCoder2 模型基于 Decoder 架构. 它不仅使用了 GitHub 公开仓库的代码, 同时也使用了 Kaggle 等平台的高质量代码数据. 以上模型在 Hugging Face 社区都公开了可本地部署的模型权重文件.

(5) GPT-3.5-Turbo 是 OpenAI 推出的一种自然语言处理模型, 基于 GPT 架构. 不同于上述模型, 它的预训练数据集未公开. 本文通过 OpenAI 官方文档推荐的 API 使用方式进行实验.

本文选择上述大模型基于以下 3 个原因. 首先是这些模型都可以实现 Python 代码的补全以及代码生成任务. 如表 2 所示, 这些模型在 HumanEval 基准测试集中表现出色. 其次, 许多大模型都基于 GitHub 公开仓库的代码进行预训练. 这使得预处理阶段可以更方便地构建本地索引数据库, 从而有效覆盖开源代码仓库. 此外, 这些大模型具有不同的参数规模, 本文也选择了开源的 StarCoder2-3B 以及模型权重闭源的 GPT-3.5-Turbo 模型进行实验.

表 2　大模型在 HumanEval 基准集 pass@k 的测试结果 (%)

Model	$k=1$	$k=10$	$k=100$
CodeParrot-110M	3.80	6.57	12.78
CodeParrot-1.5B	3.99	8.69	17.88
CodeT5+-220M	12.00	20.70	31.60
CodeT5+-770M	15.50	27.20	42.70
CodeT5+-2B	24.20	38.20	57.80
CodeGen-mono-350M	12.76	23.11	35.19
CodeGen-mono-2B	23.70	36.64	57.01
StarCoder2-3B	31.7	—	—
GPT-3.5-Turbo	64.9	—	—

4.2　代码生成指令构建

我们设计了两种类型的代码生成指令用于生成 Python 代码, 包括代码功能描述以及方法签名. 该方式模拟用户与模型的交互过程, 能够有效生成与代码逻辑有关的内容. 图 4 展示了本文构建的两种代码生成指令的示例.

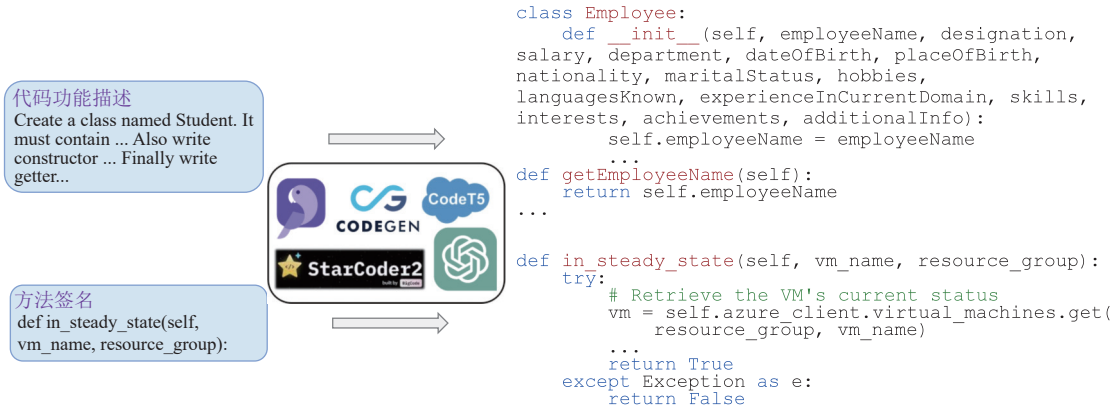


图 4　代码生成指令示例

(1) 我们选择 Self-Instruct[62]数据集作为代码功能描述数据集. Self-Instruct 数据集包含了大量复杂并且多样的指令. 这些指令基于不同的种子任务, 数据集创建者利用 ROUGE-L 相似度分布确保指令之间的差异性. 其次这些指令覆盖了广泛的主题, 包括数据转换和处理、API 使用和集成以及自动化测试等场景. Self-Instruct 创建于 2023 年 6 月, 本文后续选择的大模型 (除了 StarCoder2) 的训练日期都在此之前, 这表明这些模型并未完全训练过完全一致的数据. 该数据集包含 5 000 条由人工编写的种子指令以及高质量输出, 数据集内的 5 000 条功能描述将作为代码生成指令.

(2) 我们从 The-Stack 数据集提取了 10 000 条方法签名构建数据集. Yang 等人[25]提取了方法签名的内容作为提示词, 揭示了 CodeParrot 系列模型存在记忆代码的现象. Ciniselli 等人[24]利用不完整的 Java 代码作为 T5 模型的输入探究了其复制训练数据的问题. 我们参考此做法, 从 The-Stack 数据集内提取了 10 000 条方法签名作为指令用于代码生成. 这些方法签名提供了详细的方法定义, 包括方法名、参数名以及参数类型等.

4.3 研究问题

本文主要探究下列 3 个研究问题.

(1) 问题 1: 大模型生成的代码多大程度克隆于开源软件仓库?

(2) 问题 2: 大模型生成的代码是否存在开源许可证违规风险?

(3) 问题 3: 真实开源软件中包含的大模型生成代码是否存在开源许可证违规风险?

通过探究上述研究问题, 我们可以全面了解大模型生成代码与开源社区代码之间的许可证兼容性问题. 这些结果与发现有助于引起用户以及开发者对大模型版权违规风险方面的关注.

4.3.1 大模型生成的代码多大程度克隆于开源软件仓库

研究动机: 大模型通过大量开源仓库代码进行预训练, 这使得它们生成的代码可能与训练数据存在高度相似性, 甚至直接克隆自开源软件仓库. 量化大模型生成的代码与开源仓库代码的克隆比例, 可以评估其对现有开源项目和开发者社区的潜在影响.

实验设置: 我们利用第 4.2 节构建的两种不同场景的代码生成指令作为大模型输入, 构建每个大模型生成代码的数据集. 随后, 过滤掉了代码长度小于 6 行的代码. 对于每一个由大模型生成的代码, 利用第 3 节的代码溯源方法溯源不同代码克隆类型的开源代码. 为了更深入的理解大模型的克隆情况以及代码克隆检测结果准确率, 我们人工审查了 Type1 以及 Type2 克隆类型的代码内容. 在对每个大模型溯源到的代码去重后, 按照 95% 的置信度和 5% 的置信区间进行抽样审查, 分类出不同类别的克隆代码片段, 例如开源许可证声明、文档描述以及方法定义等[25]. 本文不局限于单个系列的模型, 还包括 CodeParrot、CodeGen-mono、CodeT5+、StarCoder2 以及 GPT-3.5-Turbo 不同参数规模下的 9 个大模型.

4.3.2 大模型生成的代码是否存在开源许可证违规风险

研究动机: 开源软件仓库的代码受到开源许可证的约束, 开源许可证规定了代码的再分发和修改条件. 如果大模型生成的代码克隆了由开源许可证约束的代码, 可能会导致许可证冲突. 深入研究这一问题, 不仅能够帮助开发者理解和识别大模型生成代码中潜在的许可证冲突, 还可以为开源社区提供指南, 以确保在使用大模型生成代码时遵循相关的许可证要求.

实验设置: 该阶段将关注研究问题 1 中大模型生成代码与克隆代码之间的开源许可证冲突问题. 与 Wolter 等人[56]针对开源代码许可证兼容性研究一致, 我们仅关注于 Type1 和 Type2 的克隆代码. Type1 克隆类型为完全相同的代码片段, 它可以证明模型生成代码是否直接克隆于开源代码. 其次 Type2 仅存在变量名方面的差别, 它证明模型可能在一定程度上受到训练数据的影响. 而 Type3/4 的代码克隆类型通常涉及复杂的结构变化和语义变化, 无法有效证明模型生成的代码来自于相似的代码片段. 利用第 3 节的开源许可证兼容性检测框架提取大模型生成代码以及克隆代码的开源许可证信息. 通过对大模型生成代码开源许可证的提取, 可以了解到本文实验的大模型是否会生成开源许可证声明的内容. 通过提取克隆代码最后修改时间构建图结构以及确定溯源代码, 可以了解到大模型生成代码与溯源代码是否存在开源许可证违规风险.

4.3.3 真实开源软件中包含的大模型生成代码是否存在开源许可证违规风险

研究动机: 随着越来越多的开发者在开源项目中使用大模型生成的代码, 开源社区内可能会出现开源许可证冲突问题. 这不仅影响到个人开发者, 也可能对整个开源生态系统的健康发展造成负面影响. GitHub 是全球最大的代码托管平台, 它拥有大量的公共代码仓库. 通过对 GitHub 平台内大模型生成代码的分析, 可以了解到当前真实开源软件中通过大模型生成的代码是否存在开源许可证违规风险.

实验设置: 为了分析开源社区中通过大模型生成的代码, 我们利用关键字在 GitHub 平台检索并收集 Python 代码. 关键字为{“Generate”, “Write” }×{“by”, “use”, “with”}×{“ChatGPT”, “Copilot”}的组合, ×代表笛卡尔积. 为了过滤当前阶段所收集到的代码内的噪声, 我们首先自动化过滤了行数小于 6 行的代码. 其次通过人工审查的方式过滤了解决常见问题的代码, 比如快速排序算法等. 过滤后的代码讲作为研究问题 3 的实验对象. 通过对收集到的代码开源许可证的提取, 可以探究开发者对大模型生成代码是否添加了开源许可证的约束. 通过对比大模型生成代码以及克隆代码的许可证违规, 可以探究开源许可证违规风险是否存在于开源社区.

4.4 配置信息

上述 9 个大模型部署于配备 NVIDIA A100 GPU (40 GB 显存) 的服务器, 模型权重均在 Hugging Face 社区公开可下载. GPT-3.5-Turbo 通过 Open AI 官方 API 访问. 在不同的实验阶段, 上述模型参数均采用默认设置. 在数据预处理阶段使用的 GitHub-Code, The-Stack 以及用于构建代码生成指令的 Self-Instruct 数据集也均从 Hugging Face 社区下载.

5 大模型生成的代码多大程度克隆于开源软件仓库

5.1 实验结果

我们利用第 4.2 节构建的两种不同场景的代码生成指令作为大模型输入, 为每个大模型构建生成代码数据集.

- 在功能描述指令下, 9 个大模型总共生成了 45 000 个代码文件, 其中代码行数大于 6 行的为 43 130 个代码文件. 我们在 8 626 个代码中溯源到了存在 Type1 和 Type2 克隆类型的开源代码片段.
- 在方法签名指令下, 9 个大模型总共生成了 90 000 个代码文件, 其中代码行数大于 6 行的为 65 900 个代码文件. 我们在 10 230 个代码中溯源到了存在 Type1 和 Type2 克隆类型的开源代码片段.

表 3 详细呈现了不同大模型在不同代码生成指令下的各代码克隆类型溯源的比例.

表 3 大模型生成代码溯源结果

模型	参数	方法功能描述指令 (%)			方法签名指令(%)		
		Type1	Type2	Type3/4	Type1	Type2	Type3/4
CodeGen-mono	350M	1.7	13.4	77.5	0.3	9.0	70.0
CodeGen-mono	2B	2.0	16.9	77.7	0.7	12.4	72.1
CodeT5+	220M	0.8	4.5	36.7	0.3	7.0	35.5
CodeT5+	770M	1.0	5.0	37.3	0.3	7.4	35.9
CodeT5+	2B	2.8	10.7	40.0	0.5	11.6	38.3
CodeParrot	110M	4.3	41.7	50.1	2.3	30.2	47.0
CodeParrot	1.5B	4.3	42.0	51.5	3.5	31.1	47.6
StarCoder2	3B	2.2	17.2	49.0	0.9	14.1	47.5
GPT-3.5-Turbo	—	1.0	8.4	16.6	0.5	7.7	14.4

我们发现, 通过方法功能描述生成的代码中, 平均有 68.5% 的代码可以溯源到存在克隆的开源代码片段. 其中, 平均 2.2% 的代码可以溯源到 Type1 克隆类型的开源代码片段. 平均 17.8% 的代码可以溯源到 Type2 克隆类型的开源代码片段. CodeParrot 系列以及 CodeGen 系列模型生成的代码中, 平均 97.0% 和 94.6% 可以溯源到存在克隆的开源代码片段. 而 GPT-3.5-Turbo 仅有 26.0% 的代码溯源到了存在克隆的开源代码片段.

通过方法签名生成的代码中, 平均有 60.9% 的代码可以溯源到存在克隆的开源代码片段. 其中, 平均 1.0% 的

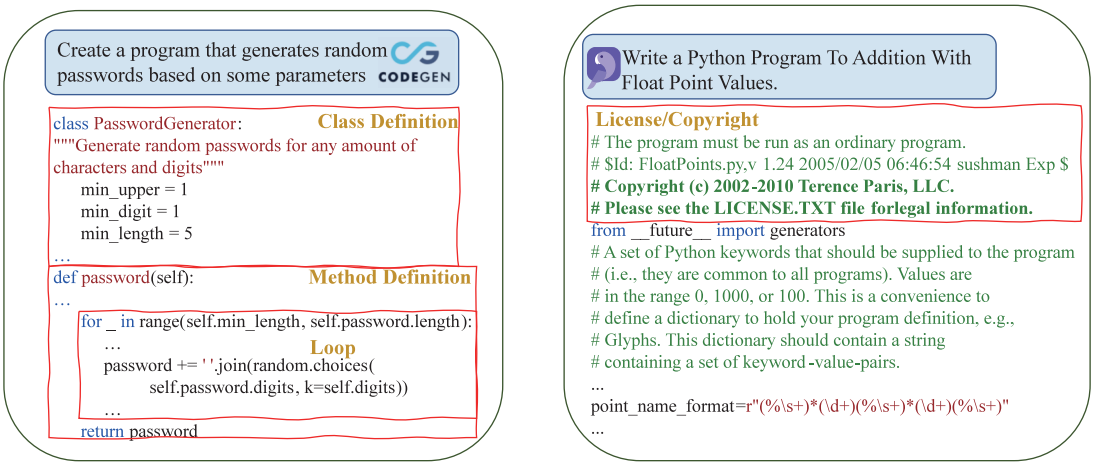
代码可以溯源到 Type1 克隆类型的开源代码片段, 平均 14.5% 的代码可以溯源到 Type2 克隆类型的开源代码片段. 同样的, CodeParrot 系列以及 CodeGen 系列模型溯源到的代码比例最高, 分别为 80.9% 和 82.3%. 而 GPT-3.5-Turbo 相比于其他大模型溯源到的代码最少, 为 22.6%. 我们发现, 通过方法签名生成的代码溯源到的比例相比于方法功能描述更低. 这可能是因为方法功能描述所提供的上下文信息更丰富.

本文框架的代码搜索阶段基于 Lucene[42]实现, 代码克隆阶段基于 Simian[40]以及 SSCD[53]工具实现. 在第 3.2.3 节中提到, 这些工具在代码搜索[43]以及代码克隆相关工作[24,25,29]中被广泛使用, 并展现了优异的性能. 为了保证代码克隆检测结果的准确性, 我们在人工分类克隆代码片段内容的同时也依然检查了 Simian 输出结果是否符合代码克隆类型的定义.

由于开源代码片段数量过多, 本节仅分类方法功能描述指令下溯源到的代码, 并且重点关注 Type1 和 Type2 克隆类型的代码片段. 通过代码片段的哈希值过滤重复的克隆代码片段, 按照 95% 的置信度和 5% 的置信区间进行抽样审查[25]. 每个模型去重后克隆代码片段共有 16 670 个, 抽样的克隆代码块数量为 2 489 个. 这些克隆代码片段均由两位作者独立检查并进行交叉验证. 我们使用 Cohen's Kappa[63]来评估两位作者之间的一致性. 如果 Kappa 值低于 0.9, 作者将进行讨论以解决分歧, 从而确保结果的准确性和一致性.

我们发现 2 489 个片段中发现有 60 个代码片段虽然符合代码克隆类型定义, 但是这些代码片段均为变量定义或者赋值语句, 不具备主要代码逻辑功能. 为了避免这一情况, 我们在分析克隆代码类型时对这些片段进行了过滤.

如图 5 所示, 我们通过克隆代码片段的特征分类其类型, 在分类时一个代码块可能包含多个不同的分类. 例如图 5(a) 中 CodeGen-2B 模型克隆代码片段内出现了类名以及详细类代码的定义, 同时还包括方法的定义以及实现, 此外在方法的定义和实现中也出现了循环代码的实现[64]. 图 5(b) 中 CodeParrot 模型克隆代码片段内出现了 Copyright 和 License 等内容, 可以证明它克隆了开源仓库内的开源许可证信息[65]. 每个模型去重后的克隆代码块数量以及详细的分类结果如表 4 所示.



(a) CodeGen-2B克隆代码　　(b) CodeParrot克隆代码

图 5　大模型生成代码克隆片段分类

通过对重复代码块的过滤, 我们发现不同的大模型所溯源到的代码数量以及代码内容存在明显差异. CodeParrot 系列模型所生成的代码相比于其他大模型溯源到了最多的代码片段, 平均溯源到了 5 274 个去重的代码块. 而 GPT-3.5-Turbo 仅溯源到了 386 个去重的代码块. 此外, CodeParrot 溯源到了开源许可证声明的代码内容, 这种现象在其他模型上并没有出现. 而且 CodeParrot 所克隆的 import 语句相比于其他大模型也更多. GPT-3.5-Turbo 则没有检测到开源许可证声明以及文档类型的内容. 导致这部分差异的原因首先可能由于不同模型所采用的预训练数据集. 其次不同大模型所采用不同的架构设计也会加剧这种问题, 不同的架构会影响它们处理和记忆

数据的方式.

相同的是, 所有大模型溯源到的最多的代码内容为整个方法的定义以及实现. 导致这种现象的原因可能是由于代码生成指令为详细的代码功能描述, 包含了更丰富的上下文信息, 大模型会定义方法并实现代码功能描述内的需求. 这种交互方式也更贴近开发者与大模型的交互, 这些模型的实验结果也表明这种交互方式会存在代码克隆的风险.

表 4 不同模型生成代码的克隆代码片段内容分类

代码片段	CodeGen-350M	CodeGen-2B	CodeT5+-220M	CodeT5+-770M	CodeT5+-2B	CodeParrot-110M	CodeParrot-1.5B	StarCoder2-3B	GPT
#去重代码块	1 063	1 271	370	601	941	5 247	5 300	1 491	386
#抽样代码块	282	295	189	175	273	358	358	306	193
Docstring	75	77	49	73	42	83	117	131	0
License/Copyright	0	0	0	0	0	45	53	0	0
Import Statement	34	49	34	33	23	136	146	24	4
Class Definition	61	62	57	55	49	48	73	90	20
Method Definition	**109**	**143**	**82**	**83**	**118**	**50**	**92**	**167**	**102**
Method Call	**172**	**205**	**99**	**111**	**190**	**142**	**151**	**200**	**145**
Conditional	53	21	23	35	78	55	48	61	63
Testing	5	0	3	5	3	49	34	3	0
Exceptions	0	2	0	0	0	10	8	0	4
Print Statement	26	33	2	7	36	4	0	19	10
Loop	27	30	15	10	48	52	36	22	37

研究问题 1 结论总结: 通过对 9 个大模型生成代码的溯源, 我们发现大模型生成的代码在很大程度上克隆于开源社区. 代码功能描述以及方法签名生成的代码中, 平均有 68.5% 和 60.9% 的代码可以溯源到开源代码片段. 此外, 不同大模型克隆的代码内容存在明显差异. CodeParrot 克隆了开源许可证声明, 这种现象在其他模型中并不常见. 总体而言, 大模型生成的代码存在较高的克隆风险, 尤其是方法定义和实现部分. 这对代码版权以及合规性的使用带来了极大的挑战.

6 大模型生成的代码是否存在开源许可证违规风险

6.1 实验结果

在研究问题 1 中, 我们在 43 130 个通过方法描述生成的代码中, 溯源到了 8 626 个 Type1 和 Type2 克隆类型的代码片段. 本节首先利用 Scancode 提取这些代码内的开源许可证. 具体的数据如表 5 所示.

表 5 代码内开源许可证数量

类型	CodeGen-350M	CodeGen-2B	CodeT5+-220M	CodeT5+-770M	CodeT5+-2B	CodeParrot-110M	CodeParrot-1.5B	StarCode2-3B	GPT
#许可证 (生成代码)	7	2	42	58	3	1 312	1 690	13	0
#许可证 (溯源代码)	**4**	0	0	0	0	**816**	**1 016**	**7**	0

我们发现大模型生成的代码大部分没有开源许可证声明. 在生成的 43 130 个代码片段中, 仅有 3 127 个代码包含开源许可证声明 (7.3%). 在溯源到的 8 626 个代码片段中, 仅有 1 843 个包含开源许可证声明 (21.4%). CodeParrot 系列模型相比于其他大模型生成并且溯源到了最多的开源许可证声明, 而 GPT-3.5-Turbo 没有生成任何开源许可证声明. 这可能是因为 CodeParrot 系列模型在生成代码时更倾向于从开源社区中克隆代码片段, 这些代码片段包含了开源许可证信息. 这与研究问题 1 中发现的 CodeParrot 系列模型相比其他大模型更倾向于克隆的代码片段内容相符合.

上述数据表明绝大多数模型生成的代码没有开源许可证, 但是由于所有的开源许可证都要求用到其内容的项

目也存在开源许可证声明, 这可能导致开源许可证冲突. 为了进一步分析开源许可证违规问题, 我们提取了溯源代码的开源许可证. 通过代码最后修改时间构建图结构提取可能的溯源代码, 进而提取溯源代码在文件级别的开源许可证. 如果不存在文件级别的许可证, 则使用仓库级别的许可证.

我们发现溯源到的开源代码中 96.6% 的代码受到开源许可证的约束. 我们在开源代码中提取到了 121 个不同的开源许可证. 图 6 呈现了溯源到的代码涉及最多的 10 个开源许可证, 可以发现 MIT 许可证的数量最多, 占整体代码文件的 32.0%. 通过对 121 个开源许可证种类的划分, 其中宽松许可证占比 65.8%, 弱限制型许可证占比 2.3%, 限制型许可证占比 27.8%. 此外, 没有许可证声明的代码占比为 3.4%, 其他类型的开源许可证占比为 0.7%. 这表明绝大部分代码在开源社区中受许可证的约束, 如果大模型生成的代码没有开源许可证信息并且开发者遗漏了对这种问题的处理, 很可能导致许可证违规风险.

通过对比大模型生成代码与溯源的开源代码之间的许可证, 我们发现 81.8% 的代码存在开源许可证违规风险. 我们共检测到了 291 种开源许可证组合. 图 7 桑基图中仅呈现了大于 5 次的开源许可证组合. 可以发现, 大模型没有生成开源许可证声明的情况占总数据的 75.9%. 其中, 大模型未声明开源许可证与溯源代码的宽松型许可证、弱限制型许可证、限制型许可证以及其他类型许可证组合占比分别为 51.9%、1.8%、21.7% 以及 0.6%. 大模型生成代码的宽松型许可证与溯源代码的弱限制型和限制型的组合比例分别为 0.2% 以及 2.9%. 此外, 大模型生成代码以及溯源代码均没有许可证声明的组合比例为 2.7%.

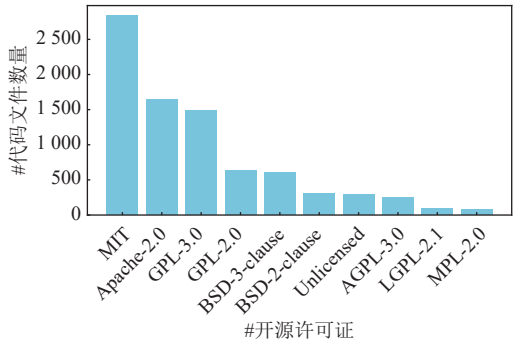


图 6 溯源到的开源代码的许可证声明 (Top10)

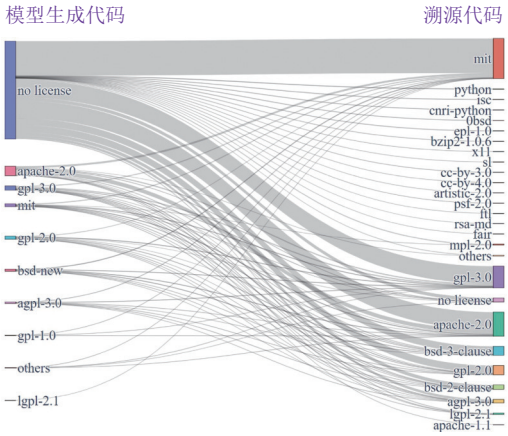


图 7 生成代码与溯源代码的开源许可证组合

如图 8 所示, 左侧为 CodeParrot 模型通过给定自然语言描述生成的代码, 我们检测到它与多个开源仓库内的代码存在克隆. 通过对比不同代码仓库内代码的最后修改时间, 我们溯源发现 ntu-dsi-dcn 仓库内最早出现此代码, 以此判断它为溯源代码. 该开源仓库内的代码受 GPL-2.0 许可证的约束[66]. 在第 3.3.3 节中提到, 当大模型生成代码没有开源许可证声明, 但是溯源代码受许可证约束时, 大模型生成代码存在开源许可证违规风险.

表 6 呈现了大模型生成代码与溯源代码之间常见的开源许可证组合及其频次. 可以发现未声明开源许可证与 MIT 开源许可证的组合出现了 2 433 次, 占比 28.2%. 除了宽松型许可证以外, 限制型许可证也十分常见. 尤其是 GPL-3.0, 未声明开源许可证与 GPL-3.0 的组合出现了 1 163 次, 占比 13.5%. 上述数据表明, 尽管考虑大模型生成的许可证声明, 其与开源代码之间的许可证冲突仍然普遍存在, 甚至会溯源到受限制型许可证约束的代码. 这使得开发者在使用大模型时, 需要仔细审查大模型生成的代码, 以确保消除许可证违规问题. 特别是在使用和发布这些代码时, 确保遵守相关开源许可证规定, 避免潜在的法律风险和版权纠纷.

研究问题 2 结论总结: 我们发现大模型生成代码存在严重的许可证违规风险. (1) 大模型生成的代码中仅 7.3% 的代码包含开源许可证声明; (2) 溯源到的开源代码有 96.6% 的代码受开源许可证的约束; (3) 通过对比大模

型生成代码与溯源代码之间的许可证, 81.8% 的代码存在开源许可证违规风险. 其中, 大模型未声明开源许可证与溯源代码的宽松型、弱限制型以及限制型许可证组合占比为 51.9%、1.8% 以及 21.7%.

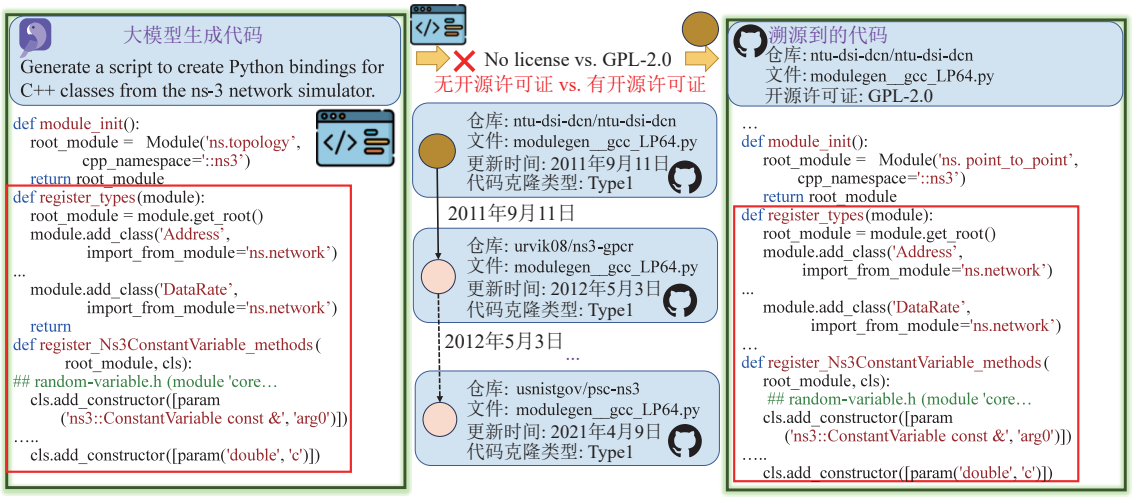


图 8 开源许可证兼容性问题示例

表 6 开源许可证组合 (Top10)

开源许可证 (生成代码 → 溯源代码)	#文件数量
No license → MIT	2 433
No license → Apache-2.0	1 198
No license → GPL-3.0	1 163
No license → GPL-2.0	471
No license → BSD-3-clause	463
Apache-2.0 → Apache-2.0	293
No license → No license	232
No license → BSD-2-clause	212
No license → AGPL-3.0	185
Apache-2.0 → MIT	136

7 真实开源软件中包含的大模型生成代码是否存在开源许可证违规风险

7.1 实验结果

我们利用第 4.3.3 节关键字在 GitHub 平台检索大模型生成的代码, 收集到了 647 个由 ChatGPT 和 Copilot 生成的 Python 代码片段. 为了确保收集到的代码具有较高的质量. 我们对数据进行了严格的筛选, 自动排除了 240 个代码行数少于 6 行的代码. 此外, 还排除了 178 个用于解决常见简单问题 (如快速排序算法等) 的代码. 过滤的代码通常在实际应用中并不具备足够的复杂性和代表性. 最终, 我们在 204 个仓库中保留下了 229 个高质量的代码文件, 其中 189 个由 ChatGPT 生成, 40 个由 Copilot 生成. 表 7 展示了这些仓库的详细数据.

表 7 代码仓库数据

统计维度	Stars			Forks			Commits			Issues			LOC		
统计指标	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
统计数据	3 760	0	44	1 973	0	16	30	1	23	30	0	7	591	6	54

我们发现, 开源社区中由大模型生成的代码大部分没有开源许可证声明, 这可能会导致许可证违规风险. 表 8 呈现了 229 个代码中所有的开源许可证声明以及出现的频次. 数据显示, 其中有 67.7% 的代码没有声明任何开源

许可证. 此外, 24.0% 的代码受宽松型许可证约束, 如 MIT、Apache-2.0、CC0-1.0、CC-BY-4.0 和 BSD-3-Clause. 余下的 8.3% 的代码受限制型许可证约束.

通过代码溯源方法, 我们发现在 229 个代码中有 38 个 (16.7%) 代码可以溯源到 Type1 和 Type2 类型的代码片段, 98 个 (43.0%) 代码可以溯源到 Type3/4 类型的代码片段. 我们人工审查了 38 个溯源到 Type1 和 Type2 的代码克隆片段, 均符合代码克隆类型的定义. 通过对比大模型生成代码与 Type1 和 Type2 溯源代码的开源许可证, 我们提取了 14 种不同的开源许可证组合. 表 9 展示了对 14 种开源许可证分类组合的种类和数量, 其中部分组合存在开源许可证违规风险, 这可能会对大模型生成代码的使用和分发带来潜在的法律风险.

表 8 开源许可证

开源许可证	#文件数量
No license	155
MIT	40
GPL-3.0	14
Apache-2.0	9
GPL-2.0	3
CC0-1.0	3
AGPL-3.0	2
CC-BY-4.0	2
BSD-3-Clause	1

表 9 开源许可证组合

开源许可证 (生成代码 → 溯源代码)	#文件数量
No license → No license	16
Permissive → Permissive	7
No license → Permissive	6
No license → Strong Copyleft	2
Permissive → No license	2
Strong Copyleft → No license	2
Permissive → Strong Copyleft	2
Strong Copyleft → Strong Copyleft	1

通过对检测到的 38 个克隆代码开源许可证分类, 我们发现有 30 个大模型生成代码与克隆代码存在开源许可证违规风险. 存在违规风险的组合分别是 No license 与宽松型、No license 与限制型、宽松型与限制型以及其他许可证与 No license. 其中有 20 个由大模型生成的代码复制了受 No license 约束的代码片段. 而根据 GitHub 平台的规定, 开发者在使用没有许可证的仓库代码时必须获得仓库开发者的授权. 为了确认开发者是否得到了许可, 我们同样将这些代码以问题报告的形式提交给了开发者. 另外我们观察到, 由于开发者可以指定仓库级别以及文件级别的开源许可证. 4 个代码的开发者指定了许可证, 但是溯源到的代码内没有开源许可证声明, 比如宽松型与 No license 的组合以及限制型与 No license 的组合.

我们将存在开源许可证违规风险的代码以问题报告的形式提交给了开发者, 详细数据如表 10 所示. 问题报告包括存在克隆的代码片段, 违规的开源许可证条款等. 目前共得到了 8 位开发者的反馈. 有 3 位开发者在收到问题报告后选择不公开代码仓库或者删除了代码仓库. 3 个仓库均未声明任何开源许可证, 3 个仓库存在克隆的代码行数平均为 39 行. 为了确保数据的真实性, 我们开源了这 3 个仓库提交的问题报告、大模型生成的代码以及溯源到的代码[67].

表 10 问题报告

代码仓库	许可证声明	代码克隆类型	许可证声明 (溯源代码)	状态
Omarzanji[68]	No license	Type2	Apache-2.0	删除代码
Genuinemerit[69]	MIT	Type2	GPL-2.0	删除代码
Caarmen[70]	MIT	Type2	AGPL-3.0	重构代码
Hwan340[67]	No license	Type2	MIT	私有/删除库
Innominata88[67]	No license	Type1	MIT	私有/删除库
Villesalmela[71]	No license	Type1	GPL-2.0	私有/删除库
fiware-orion[72]	AGPL-3.0	Type2	No license	已回复
Sandesh[73]	No license	Type2	No license	已回复
ClosedAI469[74]	No license	Type2	MIT	待回复
Rcmadden[75]	No license	Type2	GPL-3.0	待回复
Sarahhoogstraten[76]	No license	Type2	MIT	待回复

表 10 问题报告 (续)

代码仓库	许可证声明	代码克隆类型	许可证声明 (溯源代码)	状态
Mariafaraj[77]	No license	Type2	MIT	待回复
Interactive[78]	No license	Type2	No license	待回复
Bcapps-1[79]	No license	Type2	No license	待回复
Bcapps-2[79]	No license	Type2	No license	待回复
Alexacompetitor[80]	No license	Type2	No license	待回复
Cal-Rex[81]	No license	Type1	No license	待回复
Abivishaq[82]	No license	Type2	No license	待回复
Johsieders[83]	No license	Type2	No license	待回复
Neil-r[84]	CC0-1.0	Type2	No license	待回复
Martinsndifon[85]	No license	Type2	No license	待回复
TheDiscord[86]	AGPL-3.0	Type2	No license	待回复
Sensor[87]	No license	Type2	No license	待回复
MondoGao[88]	No license	Type1	No license	待回复
BlackFriday[89]	MIT	Type1	No license	待回复
ATCS[90]	No license	Type2	No license	待回复
Tryhackme[91]	No license	Type2	No license	待回复
open-world-vf[92]	No license	Type1	No license	待回复
Attention[93]	No license	Type1	No license	待回复
Benjdevries[94]	No license	Type2	No license	待回复

两位开发者在查看问题报告后, 选择删除由大模型生成的代码. 具体情况为 Omarzanji[68]开发者使用 ChatGPT 生成的代码中有 35 行代码与已有开源代码相似. 然而开发者并未声明任何类型的开源许可证, 克隆代码的开源许可证为 Apache-2.0. 开发者审查完问题报告后回复到“It was generated as a test script to collect activation samples. I have deleted the code in that file”. 主要原因是因为开发者使用 ChatGPT 生成的代码仅用作测试脚本, 不影响代码的主要逻辑[68]. 另一位 Genuinemerit[69]开发者同样选择移除了大模型生成的代码, 但是开发者提到“If I choose at a later time to include it as is, I will include with it an appropriate copyright notice”. 并且开发者认为在引入大模型生成的代码时也需要关注许可证冲突问题.

一位开发者在查看问题报告后迅速重构了大模型生成代码. 开发者声明了仓库级别的开源许可证, 但克隆代码的开源许可证为 AGPL-3.0. 尽管开发者认为存在相似的代码, 但是认为不一定存在开源许可证问题, 开发者回复到“Not very interesting IP here to protect”[70]. 但是开发者仍迅速对大模型生成代码进行了修改并合并到项目中. 同时表示“It gave me an opportunity to improve this part of the code!”.

两位开发者在查看问题报告后, 未对大模型生成代码做出变更[72,73]. 尽管如此, 他们都认为在使用大模型的代码时应该关注开源许可证风险. 一位开发者回复到“I will be more cautious about using code that lacks explicit licensing and ensure I comply with all necessary guidelines”[73]. Fiware-orion[72] 引用了 OpenAI[95]的条款“(b) own the Output. We hereby assign to you all our right, title, and interest, if any, in and to Output.”, 并指出确定代码片段的开源许可证所有权不同于文件级别, 模块级别以及库级别. 他添加“Generated by ChatGPT”注释的目的为了避免代码片段级别引入的许可证风险. 此外, 他认为测试以及集成 GPT 生成的代码与从 StackOverflow 平台获取代码片段没有区别.

研究问题 3 结论总结: 我们通过关键字检索的方式在 GitHub 平台收集到了 229 个由大模型生成的代码. 我们发现在 229 个代码文件中有 155 个代码未声明开源许可证, 55 个代码声明为宽松型许可证, 19 个代码受限制型许可证的约束. 其中有 136 个代码溯源到了存在代码克隆的开源代码片段. 我们在 38 个 Type1 和 Type2 克隆类型的代码片段内检测到了 30 个存在开源许可证违规风险. 在以问题报告形式提交给开发者后, 得到了 8 位开发者的反馈.

8 相关研究

8.1 代码搜索与代码克隆

代码搜索和代码克隆都旨在通过不同的技术来识别和处理源代码中的相似性[96−103]. 代码搜索关注于在大规模代码库中快速定位和检索与查询条件相匹配的代码片段, 主要分为基于信息检索 (IR-based) 以及基于深度学习 (DL-based) 的技术[97−100]. 基于信息检索的研究主要致力于扩展和优化代码查询来提高代码搜索的准确性[98,99]. 基于深度学习的相关研究主要利用神经网络捕捉代码查询与代码片段之间的语义关系[101−103]. 通过对大规模代码库的有效处理, 代码搜索可以实现快速的相似代码检索. 代码克隆则侧重于识别和分析源代码中的重复或相似代码段, 判断代码之间的相似程度[51]. 一部分研究关注于代码的语法和语义层面的相似性, 通过对比文本或者语法的相似性判断代码克隆类型. 这一技术可以有效地检测 Type1 和 Type2 的克隆类型, 但无法有效地处理 Type3 和 Type4 类型[41,47,48]. 另一部分研究则利用深度学习模型学习代码的语义, 可以有效的处理 Type3 和 Type4 的克隆类型[104−106]. 然而, 随着代码库的数据量以及数据复杂度增加, 代码克隆相关技术难以在大规模代码库下即时检测[41].

我们利用信息检索的方法在大规模的数据集内检索相似的代码片段, 并通过不同克隆类型的代码克隆检测工具分析相似代码的克隆类型. 通过结合代码搜索以及代码克隆检测技术, 我们的方法可以更迅速并且有效地检测到存在不同克隆类型的开源代码片段.

8.2 大模型代码溯源

随着大模型的迅速发展以及相关大型数据集的开源, 越来越多的研究关注大模型生成内容带来的隐私以及版权问题. Somepalli 等人[20]利用图像检索方法发现扩散模型会频繁复制训练数据. Carlini 等人[22]在 Stable Diffusion 和 Imagen 等大型扩散模型中提取了大量几乎完全复制的训练样本. Carlini 等人[19]从 GPT-2 模型中提取出大量包含个人身份信息、源代码、UUIDs 等敏感数据的训练样本. 然而上述工作均未考虑在代码层面的复制情况. Niu 等人[23]从 GitHub Copilot 模型生成的内容中提取到了敏感的个人信息. Ciniselli 等人[24]研究发现 T5 模型生成的代码内 10% 是与训练集中完全一致的代码, 80% 是与训练集中代码结构相同但变量名等不同的代码. Yang 等人[25]利用代码克隆检测技术发现 CodeParrot 模型记忆了大量训练数据内的代码片段. Yang 等人[25]虽然讨论了大模型生成代码与训练集会存在重复内容, 但是并未考虑生成代码与开源社区之间开源许可证的违规风险. 其次 Yang 等人[25]仅利用 Simian 工具检测代码克隆, 这种方式在本文 9 个大模型以及 15 000 条代码生成指令下无法有效的开展.

8.3 开源许可证合规性

开源许可证版权冲突的检测多年来一直是工业界和学术界的研究热点. OpenChain[107]以及 SPDX[58]对开源许可证的倡议以及 FOSSology[108]等许可证冲突检测工具, 使用户更容易识别和遵守开源许可证的约束. 开源社区的代码通常受开源许可证的约束, Almeida 等人[109]发现开发者甚至不熟悉开源许可证约束条款. Duan 等人[110]在开源软件级别识别许可证冲突问题, 它们发现超过 4 万个应用可能违反 GPL/AGPL 许可证条款. Golubev 等人[59]通过对 GitHub 平台内 Java 项目收集与分析发现有 29.6% 的项目之间存在潜在的代码复制情况, 这其中有 9.4% 存在许可证冲突问题. Wolter 等人[56]通过对 1 000 个 GitHub 代码仓库以及仓库内文件分析发现大约一半的代码仓库没有许可证声明, 并且 10% 的代码存在许可证违规的情况. Moraes 等人[111]在抽样的 1 552 个 JavaScript 项目中发现大约 62% 的代码库声明了多种开源许可证. 此外, 他们提到许多开发者并不了解开源许可证之间的关系以及使用多种许可证后果.

大模型的出现很可能加剧开源许可证冲突. 开发者并不知情大模型生成代码是否受许可证约束, 这很可能导致开源许可证违规. 本文所设计的代码溯源以及开源许可证检测方法可以有效地分析大模型生成代码与开源社区存在克隆的情况, 并且检测生成代码与克隆代码之间的开源许可证违规.

9 有效性威胁

9.1 内部有效性

大模型生成代码溯源与版权违规问题检测框架: 本文的实验代码由 4 部分组成: 数据预处理、代码搜索、代码克隆以及开源许可证兼容性检测阶段. 每一部分实现的代码和依赖的工具可能会影响方法的准确性. 为了降低内部有效性威胁, 我们在数据预处理和代码搜索阶段基于 Lucene[42]实现, 该方法通过合理的索引构建在相关工作中展现了有效的代码搜索性能, 在 IJaDataset 数据集上可以达到 90% 的召回率[43]. 尽管代码搜索阶段可能遗漏存在克隆的代码片段, 我们在两种代码指令下仍溯源到了 68.5% 和 60.9% 的代码. 在代码克隆阶段, 我们依赖于 Simian[40]以及 SSCD[53]工具, 这两个工具在相关工作中被广泛使用, 并展现了优异的性能. Simian 工具能够有效地检测不完整 Python 代码片段的 Type1 以及 Type2 克隆, SSCD 在 Company-C 数据集内可以达到 92% 的准确率[48,53]. 在开源许可证兼容性检测阶段, 我们通过 DejaCode[60]以及 SPDX[58]获取开源许可证类型并仔细审查, 确保开源许可证对比阶段的准确性. 这一系列措施旨在最大限度地降低内部威胁, 提升实验结果的可信度.

人工分析: 我们在研究问题 1 中对代码克隆内容的分析涉及手动操作, 这可能会引入主观性和偏见. 为了减轻这一威胁, 我们采用了开放编码方案, 由两位作者独立检查并交叉验证所有结果. 我们使用 Cohen's Kappa[63]来衡量两位作者之间的一致性. 如果 Kappa 值小于 0.9, 作者需要讨论以解决分歧, 以确保结果的准确性和一致性. 此外, 我们公开了本文的数据集, 以便审查我们的实验结果.

9.2 外部有效性

编程语言的普适性: 本文基于 Python 编程语言开展实验, 这使得本文的结论可能不适用于其他编程语言. 然而 Ciniselli 等人[24]发现 CodeT5 模型生成的 Java 代码存在大量的代码克隆的现象. 这表明代码克隆现象在其他语言同样存在. 而开源社区的代码通常受到开源许可证的约束, 这种代码克隆现象也表明模型生成代码可能与开源仓库内的代码存在开源许可证兼容性风险. 尽管本文主要针对 Python 语言, 但本文提出的框架和发现也可以为其他编程语言的许可证兼容性问题提供参考和启示.

代码生成指令的复杂性: 在真实开发场景下, 开发者使用大模型生成代码的场景更为复杂. 为了减轻这一威胁, 我们首先构造了不同场景的代码生成指令, 以确保结果能够适用于不同的实际开发场景, 其中包括代码功能描述以及方法签名. 其次, 我们选择了多种不同架构和参数规模的大模型进行实验. 这些大模型包括 CodeParrot、CodeGen、CodeT5+、StarCoder2 和 GPT-3.5-Turbo, 确保研究结果具有广泛的适用性. 实验结果表明, 通过方法功能描述生成的代码平均溯源到了 68.5% 的代码, 通过方法签名生成的代码平均溯源到了 60.9%.

10 结论与展望

本文旨在探究大模型生成代码与开源仓库代码之间的许可证违规问题. 我们首先设计了一个支持大模型生成代码溯源与版权违规问题检测框架. 利用该框架, 我们通过实践调查 3 个问题, 揭示大模型代码生成技术对开源软件生态的影响. (1) 通过溯源 9 个大模型在功能描述和方法签名生成的 45 000 和 90 000 个代码, 我们发现分别有 65.8% 和 60.9% 的代码可以溯源到存在克隆的开源代码片段. (2) 通过对比大模型生成代码与溯源代码的许可证, 我们发现 Type1 和 Type2 克隆类型的代码片段存在显著的开源许可证违规风险, 其中 81.8% 的代码存在开源许可证冲突. (3) 这些问题在真实开源软件同样存在, 在收集到的 229 个 GitHub 平台开发者使用大模型生成的代码中, 我们溯源到了 38 个 Type1 和 Type2 克隆类型的开源代码, 其中有 30 个存在开源许可证违规风险.

对于开源许可证研究人员: 尽管本文所提出的方法可以有效地检测代码大模型生成代码与开源社区的许可证兼容性问题, 但是仍然存在提升的空间. 一方面, 研究人员可以针对开源许可证的条款, 开发更精确的许可证违规检测技术. 另一方面, 在大模型训练过程中, 设计自动化工具以有效去除敏感信息和受版权保护的内容也是一个重要方向.

对于代码大模型开发者: 代码大模型开发者需要在模型设计和训练阶段加强对开源许可证的重视, 确保使用

的数据集来源合法且许可证清晰. 随着大模型相关许可证的发展, 开发者应在大模型的数据使用和应用范围中添加合适的条款, 以确保大模型的使用符合最新的法律和伦理标准.

对于代码大模型用户: 用户在使用代码大模型生成代码时, 首先应具备基本的开源许可证知识, 以识别和避免可能的法律风险. 其次, 应仔细审查代码的来源和许可证信息, 确保遵循相关开源许可证要求. 特别是随着大模型专用许可证的发展, 用户在项目中使用大模型生成的代码前, 应进行许可证合规性检查.

在未来的工作中, 我们计划扩充更丰富的本地索引库, 包括代码仓库的数量以及索引的种类. 我们也将探究更多的大模型以及其他的编程语言是否存在类似问题. 此外, 探究提词工程以及预训练数据集的有效清洗等技术消除开源许可证问题也将是我们未来研究的重要方向.

我们在 GitHub 平台公开了本文的数据[67]. 其中包括: (1) 大模型生成代码溯源与版权违规问题检测框架源码; (2) 大模型生成的 135 000 个代码以及溯源到的开源代码信息, 包括代码片段以及开源许可证信息; (3) 229 个在 GitHub 平台通过关键字检索到的由 ChatGPT 以及 Copilot 生成的 Python 代码; (4) 开源社区检测到的存在许可证违规的仓库代码以及问题报告.

References:

[1] Li G, Peng X, Wang QX, Xie T, Jin Z, Wang J, Ma XX, Li XD. Challenges from LLMs as a natural language based human-machine collaborative tool for software development and evolution. Ruan Jian Xue Bao/Journal of Software, 2023, 34(10): 4601–4606 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7008.htm [doi: 10.13328/j.cnki.jos.007008]

[2] Zhao WX, Zhou K, Li JY, Tang TY, Wang XL, Hou YP, Min YQ, Zhang BC, Zhang JJ, Dong ZC, Du YF, Yang C, Chen YS, Chen ZP, Jiang JH, Ren RY, Li YF, Tang XY, Liu ZK, Liu PY, Nie JY, Wen JR. A survey of large language models. arXiv:2303.18223, 2023.

[3] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis: ACL, 2019. 4171–4186. [doi: 10.18653/v1/N19-1423]

[4] Brown TB, Mann B, Ryder N, *et al*. Language models are few-shot learners. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: ACM, 2020. 159. [doi: 10.5555/3495724.3495883]

[5] Chang YP, Wang X, Wang JD, Wu Y, Yang LY, Zhu KJ, Chen J, Yi XY, Wang CX, Wang YD, Ye W, Zhang Y, Chang Y, Yu PS, Yang Q, Xie X. A survey on evaluation of large language models. ACM Trans. on Intelligent Systems and Technology, 2024, 15(3): 39. [doi: 10.1145/3641289]

[6] googleapis. Python client for Google BigQuery. 2025. https://github.com/googleapis/python-bigquery

[7] Gao L, Biderman S, Black S, Golding L, Hoppe T, Foster C, Phang J, He H, Thite A, Nabeshima N, Presser S, Leahy C. The Pile: An 800 GB dataset of diverse text for language modeling. arXiv:2101.00027, 2020.

[8] Kocetkov D, Li R, Allal LB, Li J, Mou CH, Ferrandis CM, Jernite Y, Mitchell M, Hughes S, Wolf T, Bahdanau D, von Werra L, De Vries H. The Stack: 3 TB of permissively licensed source code. arXiv:2211.15533, 2022.

[9] Xu FF, Alon U, Neubig G, Hellendoorn VJ. A systematic evaluation of large language models of code. In: Proc. of the 6th ACM SIGPLAN Int'l Symp. on Machine Programming. San Diego: ACM, 2022. 1–10. [doi: 10.1145/3520312.3534862]

[10] Zhang S, Chen ZF, Shen YK, Ding MY, Tenenbaum JB, Gan C. Planning with large language models for code generation. In: Proc. of the 11th Int'l Conf. on Learning Representations. Kigali: ICLR, 2023.

[11] Jiang JY, Wang F, Shen JS, Kim S, Kim S. A survey on large language models for code generation. arXiv:2406.00515, 2024.

[12] Fan A, Gokkaya B, Harman M, Lyubarskiy M, Sengupta S, Yoo S. Large language models for software engineering: Survey and open problems. In: Proc. of the 2023 IEEE/ACM Int'l Conf. on Software Engineering: Future of Software Engineering. Melbourne: IEEE, 2023. 31–53. [doi: 10.1109/ICSE-FoSE59343.2023.00008]

[13] Zan DG, Chen B, Zhang FJ, Lu DJ, Wu BC, Guan B, Wang YJ, Lou JG. Large language models meet NL2Code: A survey. In: Proc. of the 61st Annual Meeting of the Association for Computational Linguistics. Toronto: ACL, 2023. 7443–7464. [doi: 10.18653/v1/2023.acl-long.411]

[14] Chen M, Tworek J, Jun H, *et al*. Evaluating large language models trained on code. arXiv:2107.03374, 2021.

[15] GitHub. GitHub Copilot—Your AI pair programmer. 2025. https://github.com/features/copilot

[16] Amazon, AWS. The most capable generative AI-powered assistant for software development—Amazon Q developer. 2025 (in Chinese). https://aws.amazon.com/cn/codewhisperer/

[17] Roziere B, Gehring J, Gloeckle F, *et al*. Code Llama: Open foundation models for code. arXiv:2308.12950, 2023.

[18] Li R, Allal LB, Zi YT, *et al*. StarCoder: May the source be with you! arXiv:2305.06161, 2023.

[19] Carlini N, Tramèr F, Wallace E, Jagielski M, Herbert-Voss A, Lee K, Roberts A, Brown TB, Song D, Erlingsson Ú, Oprea A, Raffel C. Extracting training data from large language models. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 2633–2650.

[20] Somepalli G, Singla V, Goldblum M, Geiping J, Goldstein T. Diffusion art or digital forgery? Investigating data replication in diffusion models. In: Proc. of the 2023 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Vancouver: IEEE, 2023. 6048–6058. [doi: 10.1109/CVPR52729.2023.00586]

[21] Katzy J, Popescu R, van Deursen A, Izadi M. An exploratory investigation into code license infringements in large language model training datasets. In: Proc. of the 1st IEEE/ACM Int'l Conf. on AI Foundation Models and Software Engineering. Lisbon: ACM, 2024. 74–85. [doi: 10.1145/3650105.3652298]

[22] Carlini N, Hayes J, Nasr M, Jagielski M, Sehwag V, Tramèr F, Balle B, Ippolito D, Wallace E. Extracting training data from diffusion models. In: Proc. of the 32nd USENIX Security Symp. Anaheim: USENIX, 2023. 5253–5270.

[23] Niu L, Mirza MS, Maradni Z, Pöpper C. CodexLeaks: Privacy leaks from code generation language models in GitHub Copilot. In: Proc. of the 32nd USENIX Security Symp. Anaheim: USENIX, 2023. 2133–2150.

[24] Ciniselli M, Pascarella L, Bavota G. To what extent do deep learning-based code recommenders generate predictions by cloning code from the training set? In: Proc. of the 19th Int'l Conf. on Mining Software Repositories. Pittsburgh: ACM, 2022. 167–178. [doi: 10.1145/3524842.3528440]

[25] Yang Z, Zhao ZP, Wang CY, Shi CY, Kim D, Han DY, Lo D. Unveiling memorization in code models. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 72. [doi: 10.1145/3597503.3639074]

[26] Gershgorn D, Altman S, *et al*. GitHub Copilot. 2021. https://en.wikipedia.org/wiki/GitHub_Copilot

[27] Carmack J, Romero J, *et al*. Fast inverse square root. 2025. https://en.wikipedia.org/wiki/Fast_inverse_square_root

[28] Ronacher A. Armin Ronacher on X. 2021. https://x.com/mitsuhiko/status/1410886329924194309

[29] Wu X, Wu JY, Zhou MH, Wang ZQ, Yang LY. Selection of open source license: Challenges and influencing factors. Ruan Jian Xue Bao/Journal of Software, 2022, 33(1): 1–25 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6279.htm [doi: 10.13328/j.cnki.jos.006279]

[30] Wang ZQ, Wu S, Xiao GQ, Zhang ZL, Liu ZY, Peng J. Study of open-source software license compliance. Ruan Jian Xue Bao/Journal of Software, 2022, 33(8): 3035–3058 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6374.htm [doi: 10.13328/j.cnki.jos.006374]

[31] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proc. of the 31st Int'l Conf. on Neural Information Processing Systems. Long Beach: ACM, 2017. 6000−6010. [doi: 10.5555/3295222.3295349]

[32] CodeParrot, Hugging Face, *et al*. CodeParrot. 2022. https://huggingface.co/CodeParrot/CodeParrot

[33] Lozhkov A, Li R, Allal LB, *et al*. StarCoder 2 and The-Stack v2: The next generation. arXiv:2402.19173, 2024.

[34] OpenAI. OpenAI platform. 2025. https://platform.openai.com/docs/models

[35] Wang Y, Le H, Gotmare AD, Bui N, Li JN, Hoi S. Codet5+: Open code large language models for code understanding and generation. In: Proc. of the 2023 Conf. on Empirical Methods in Natural Language Processing. Singapore: ACL, 2023. 1069−1088. [doi: 10.18653/v1/2023.emnlp-main.68]

[36] ChatGPT. Generate self-signed certificate. 2024. https://chatgpt.com/share/f4fee93f-9abc-4b77-85d8-d28ccdf99e41

[37] wazuh.configuration.py. 2020. https://github.com/wazuh/wazuh/blob/master/api/api/configuration.py

[38] Vendome C, Bavota G, Penta MD, Linares-Vásquez M, German D, Poshyvanyk D. License usage and changes: A large-scale study on GitHub. Empirical Software Engineering, 2017, 22(3): 1537–1577. [doi: 10.1007/s10664-016-9438-4]

[39] Jiang LX, Misherghi G, Su ZD, Glondu S. Deckard: Scalable and accurate tree-based detection of code clones. In: Proc. of the 29th Int'l Conf. on Software Engineering. Minneapolis: IEEE, 2007. 96–105. [doi: 10.1109/ICSE.2007.30]

[40] Simian. Simian identifies duplication in source code. 2022. https://simian.quandarypeak.com/

[41] Cordy JR, Roy CK. The NiCad clone detector. In: Proc. of the 19th IEEE Int'l Conf. on Program Comprehension. Kingston: IEEE, 2011. 219–220. [doi: 10.1109/ICPC.2011.26]

[42] Apache Software Foundation. Welcome to Apache Lucene. 2024. https://lucene.apache.org/

[43] Kim K, Kim D, Bissyandé TF, Choi E, Li L, Klein J, Traon YL. FaCoY: A code-to-code search engine. In: Proc. of the 40th Int'l Conf. on Software Engineering. Gothenburg: ACM, 2018. 946–957. [doi: 10.1145/3180155.3180187]

[44] BigCode/The-Stack. 2023. https://huggingface.co/datasets/bigcode/the-stack

[45] CodeParrot/GitHub-code. 2023. https://huggingface.co/datasets/CodeParrot/github-code
[46] Smith J, Johnson MB, *et al*. Introduction—Tree-sitter. 2024. https://tree-sitter.github.io/
[47] Sajnani H, Saini V, Svajlenko J, Roy CK, Lopes CV. SourcererCC: Scaling code clone detection to big-code. In: Proc. of the 38th Int'l Conf. on Software Engineering. Austin: IEEE, 2016. 1157–1168. [doi: 10.1145/2884781.2884877]
[48] Ragkhitwetsagul C, Krinke J. Siamese: Scalable and incremental code clone search via multiple code representations. Empirical Software Engineering, 2019, 24(4): 2236–2284. [doi: 10.1007/s10664-019-09697-7]
[49] Kamiya T. CCFinderX: An interactive code clone analysis environment. In: Inoue K, Roy CK, eds. Code Clone Analysis: Research, Tools, and Practices. Singapore: Springer, 2021. 31–44. [doi: 10.1007/978-981-16-1927-4_2]
[50] Tambon F, Dakhel AM, Nikanjam A, Khomh F, Desmarais MC, Antoniol G. Bugs in large language models generated code: An empirical study. arXiv:2403.08937, 2024.
[51] Zakeri-Nasrabadi M, Parsa S, Ramezani M, Roy C, Ekhtiarzadeh M. A systematic literature review on source code similarity measurement and clone detection: Techniques, applications, and challenges. Journal of Systems and Software, 2023, 204: 111796. [doi: 10.1016/j.jss.2023.111796]
[52] PMD. PMD: An extensible cross-language static code analyzer. 2024. https://pmd.github.io/
[53] Chochlov M, Ahmed GA, Patten JV, Lu GX, Hou W, Gregg D, Buckley J. Using a nearest-neighbour, BERT-based approach for scalable clone detection. In: Proc. of the 2022 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Limassol: IEEE, 2022. 582–591. [doi: 10.1109/ICSME55016.2022.00080]
[54] Numata S, Yoshida N, Choi E, Inoue K. On the effectiveness of vector-based approach for supporting simultaneous editing of software clones. In: Proc. of the 17th Int'l Conf. on Product-focused Software Process Improvement. Trondheim: Springer, 2016. 560–567. [doi: 10.1007/978-3-319-49094-6_41]
[55] Yokoi K, Choi E, Yoshida N, Inoue K. Investigating vector-based detection of code clones using BigCloneBench. In: Proc. of the 25th Asia-Pacific Software Engineering Conf. (APSEC). Nara: IEEE, 2018. 699–700. [doi: 10.1109/APSEC.2018.00095]
[56] Wolter T, Barcomb A, Riehle D, Harutyunyan N. Open source license inconsistencies on GitHub. ACM Trans. on Software Engineering and Methodology, 2023, 32(5): 110. [doi: 10.1145/3571852]
[57] aboutcode-org. ScanCode toolkit. 2024. https://github.com/nexB/scancode-toolkit
[58] Kapitsaki GM, Kramer F, Tselikas ND. Automating the license compatibility process in open source software with SPDX. Journal of Systems and Software, 2017, 131: 386–401. [doi: 10.1016/j.jss.2016.06.064]
[59] Golubev Y, Eliseeva M, Povarov N, Bryksin T. A study of potential code borrowing and license violations in Java projects on GitHub. In: Proc. of the 17th Int'l Conf. on Mining Software Repositories. Seoul: ACM, 2020. 54–64. [doi: 10.1145/3379597.3387455]
[60] DejaCode. DejaCode License. 2024. https://public.dejacode.com/licenses
[61] Nijkamp E, Pang B, Hayashi H, Tu LF, Wang H, Zhou YB, Savarese S, Xiong CM. CodeGen: An open large language model for code with multi-turn program synthesis. In: Proc. of the 11th Int'l Conf. on Learning Representations. Kigali: ICLR, 2023.
[62] CodeParrot/Self-Instruct-StarCoder. https://huggingface.co/datasets/CodeParrot/Self-Instruct-starcoder
[63] McHugh ML. Interrater reliability: The Kappa statistic. Biochemia Medica, 2012, 22(3): 276–282. [doi: 10.11613/BM.2012.031]
[64] austindlawless. Cudas.org an AgularJS APP. 2016. https://github.com/austindlawless/cudas
[65] Google. Rekall Memory Forensic Framework. 2020. https://github.com/google/recall
[66] ntu-dsi-dcn. The official open source NS-3 simulation framework for datacenter network architectures. 2015. https://github.com/ntu-dsi-dcn/ntu-dsi-dcn
[67] Thompson-cyber. llmsource. 2024. https://github.com/Thompson-cyber/llmsource
[68] omarzanji. Potential license violation in the sample-recoder.py#5. 2024. https://github.com/omarzanji/ditto_activation/issues/5
[69] genuinemerit. an open question: ChatGPT generated code may clone from another repository. 2024. https://github.com/genuinemerit/saskan-app/issues/8
[70] caarmen. Potential GPL license violation in createadminuser.py generated by ChatGPT. 2024. https://github.com/caarmen/iou/issues/28
[71] llm-source. villesalmelaot-harjoitustyo. 2024. https://github.com/villesalmela/ot-harjoitustyo/issues/1
[72] telefonicaid. A question about the code generated by ChatGPT. 2024. https://github.com/telefonicaid/fiware-orion/issues/4629
[73] sandesh-z. Class solution in avg_binary_tree. 2024. https://github.com/sandesh-z/leetcode_problems/issues/1
[74] ClosedAI469. Potential license violation in classification.py. 2024. https://github.com/ClosedAI469/Visual-Stimuli-Reconstruction-using-Multi-subject-fMRI-data/issues/19
[75] rcmadden. Potential license violation: ChatGPT-generated code and GPL-3.0 cloned code in OOP_exercise.py#1. 2024. https://github.com/rcmadden/courses/issues/1

[76] sarahhoogstraten. Potential license violation with LLM-generated code. 2024. https://github.com/sarahhoogstraten/CMSE202-f23-turnin/issues/1
[77] mariafaraj. Potential license violation with ChatGPT generated code. 2024. https://github.com/mariafaraj/CMSE202-f23-turnin/issues/2
[78] Artemii-Shlychkov. Potential license violation with ChatGPT generated code. 2024. https://github.com/Artemii-Shlychkov/Interactive-Conway-s-Game/issues/1
[79] barrycarter. Potential license violation with ChatGPT generated code. 2024. https://github.com/barrycarter/bcapps/issues/2
[80] llm-source. Observations on a potential license violation with ChatGPT generated code. 2024. https://github.com/arashbioinfo/alexa-competitor/issues/1
[81] Cal-Rex. Potential license violation with ChatGPT generated code. 2024. https://github.com/Cal-Rex/milestone-project-4-fly-movement-library/issues/34
[82] Abivishaq. Potential license violation with ChatGPT generated code. 2024. https://github.com/Abivishaq/Intro_to_robotics_7785_lab4/issues/1
[83] johsieders, Thompson-cyber. Potential license violation with GPT generated code. 2024. https://github.com/johsieders/potpourri/issues/1
[84] llm-source. Neil-rimgt license violation. 2024. https://github.com/neil-r/imgt_657_examples/issues/1
[85] martinsndifon. Potential license violation with GPT generated code. 2024. https://github.com/martinsndifon/python/issues/1
[86] rf20008. An open question of the ChatGPT generated code. 2024. https://github.com/rf20008/TheDiscordMathProblemBotRepo/issues/20
[87] faludi. Potential license violation with GPT generated code. 2024. https://github.com/faludi/xbee-sensor-lab/issues/1
[88] MondoGao. Potential license violation with ChatGPT generated code. 2024. https://github.com/MondoGao/uwm-cs760-hw4/issues/1
[89] friuns2. An open question of AI generated code. 2024. https://github.com/friuns2/BlackFriday-GPTs-Prompts/issues/87
[90] Echandler123. License question in AI generate code. 2024. https://github.com/Echandler123/ATCS-2023/issues/1
[91] djalilayed. An open question for AI generated codes. 2024. https://github.com/djalilayed/tryhackme/issues/1
[92] timwalsh300. Potential license violation with ChatGPT generated code. 2024. https://github.com/timwalsh300/open-world-vf/issues/1
[93] sethcoast. A question in model.py. 2024. https://github.com/sethcoast/attention/issues/1
[94] benjdevries. A question of AI generated code. 2024. https://github.com/benjdevries/advent-of-code-2023/issues/1
[95] OpenAI. Trems of use. 2025. https://openai.com/policies/terms-of-use/
[96] Di Grazia L, Pradel M. Code search: A survey of techniques for finding code. ACM Computing Surveys, 2023, 55(11): 220. [doi: 10.1145/3565971]
[97] Lee MW, Roh JW, Hwang SW, Kim S. Instant code clone search. In: Proc. of the 18th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. Santa Fe: ACM, 2010. 167–176. [doi: 10.1145/1882291.1882317]
[98] Bajracharya S, Lopes C. Mining search topics from a code search engine usage log. In: Proc. of the 6th IEEE Int'l Working Conf. on Mining Software Repositories. Vancouver: IEEE, 2009. 111–120. [doi: 10.1109/MSR.2009.5069489]
[99] Balachandran V. Query by example in large-scale code repositories. In: Proc. of the 2015 IEEE Int'l Conf. on Software Maintenance and Evolution. Bremen: IEEE, 2015. 467–476. [doi: 10.1109/ICSM.2015.7332498]
[100] Xie YT, Lin JY, Dong HD, Zhang L, Wu ZH. Survey of code search based on deep learning. ACM Trans. on Software Engineering and Methodology, 2024, 33(2): 54. [doi: 10.1145/3628161]
[101] Husain H, Wu HH, Gazit T, Allamanis M, Brockschmidt M. CodeSearchNet challenge: Evaluating the state of semantic code search. arXiv:1909.09436, 2019.
[102] Salza P, Schwizer C, Gu J, Gall HC. On the effectiveness of transfer learning for code search. IEEE Trans. on Software Engineering, 2023, 49(4): 1804–1822. [doi: 10.1109/TSE.2022.3192755]
[103] Sisman B, Kak AC. Assisting code search with automatic query reformulation for bug localization. In: Proc. of the 10th Working Conf. on Mining Software Repositories (MSR). San Francisco: IEEE, 2013. 309–318. [doi: 10.1109/MSR.2013.6624044]
[104] Hu YT, Zou DQ, Peng JR, Wu YM, Shan JJ, Jin H. TreeCen: Building tree graph for scalable semantic code clone detection. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Automated Software Engineering. Rochester: ACM, 2022. 109. [doi: 10.1145/3551349.3556927]
[105] Li LQ, Feng H, Zhuang WJ, Meng N, Ryder B. CCLearner: A deep learning-based clone detection approach. In: Proc. of the 2017 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Shanghai: IEEE, 2017. 249–260. [doi: 10.1109/ICSME.2017.46]
[106] Tao CN, Zhan Q, Hu X, Xia X. C4: Contrastive cross-language code clone detection. In: Proc. of the 30th IEEE/ACM Int'l Conf. on Program Comprehension. ACM, 2022. 413–424. [doi: 10.1145/3524610.3527911]
[107] Azhakesan A, Paulisch F. Sharing at scale: An open-source-software-based license compliance ecosystem. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering: Software Engineering in Practice. Seoul: ACM, 2020. 130–131. [doi: 10.1145/

3377813.3381351]
[108] Gobeille R. The FOSSology project. In: Proc. of the 2008 Int'l Working Conf. on Mining Software Repositories. Leipzig: ACM, 2008. 47–50. [doi: 10.1145/1370750.1370763]
[109] Almeida DA, Murphy GC, Wilson G, Hoye M. Do software developers understand open source licenses? In: Proc. of the 25th IEEE/ACM Int'l Conf. on Program Comprehension (ICPC). Buenos Aires: IEEE, 2017. 1–11. [doi: 10.1109/ICPC.2017.7]
[110] Duan RA, Bijlani A, Xu M, Kim T, Lee W. Identifying open-source license violation and 1-day security risk at large scale. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 2169–2185. [doi: 10.1145/3133956.3134048]
[111] Moraes JP, Polato I, Wiese I, Saraiva F, Pinto G. From one to hundreds: Multi-licensing in the JavaScript ecosystem. Empirical Software Engineering, 2021, 26(3): 39. [doi: 10.1007/s10664-020-09936-2]

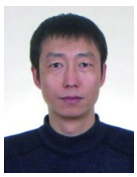
附中文参考文献:

[1] 李戈, 彭鑫, 王千祥, 谢涛, 金芝, 王戟, 马晓星, 李宣东. 大模型: 基于自然交互的人机协同软件开发与演化工具带来的挑战. 软件学报, 2023, 34(10): 4601–4606. http://www.jos.org.cn/1000-9825/7008.htm [doi: 10.13328/j.cnki.jos.007008]
[16] Amazon, AWS. 适用于软件开发的生成式人工智能助手—Amazon Q 开发者版. 2025. https://aws.amazon.com/cn/codewhisperer/
[29] 吴欣, 武健宇, 周明辉, 王志强, 杨丽蕴. 开源许可证的选择: 挑战和影响因素. 软件学报, 2022, 33(1): 1–25. http://www.jos.org.cn/1000-9825/6279.htm [doi: 10.13328/j.cnki.jos.006279]
[30] 王志强, 伍胜, 肖国强, 张自力, 刘志有, 彭景. 开源许可证合规性研究. 软件学报, 2022, 33(8): 3035–3058. http://www.jos.org.cn/1000-9825/6374.htm [doi: 10.13328/j.cnki.jos.006374]

王毅博(1997－), 男, 博士生, CCF 学生会员, 主要研究领域为智能化软件开发, AI 大模型, 开源软件大数据分析.

许畅(1977－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为大数据软件工程, 智能软件测试与分析, 自适应和自控软件系统.

王莹(1987－), 女, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为开源治理技术, 软件测试及分析学.



于海(1971－), 男, 博士, 副教授, 主要研究领域为软件测试, 软件重构及软件测试技术, 软件体系结构, 复杂网络理论, 混沌加密技术.

余跃(1988－), 男, 博士, 副研究员, 主要研究领域为实证软件工程, 群体化开发, 开源软件生态.

朱志良(1962－), 男, 博士, 教授, 博士生导师, 主要研究领域为混沌加密技术, 复杂网络理论, 软件测试技术.