软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007640] [CSTR: 32375.14.jos.007640]

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

基于大语言模型的提交日志生成*

徐圣斌[1], 贾林杰[1], 许汤智[1], 朱晓瑞[2], 余 萍[1], 徐 锋[1], 马晓星[1]

[1](计算机软件新技术国家重点实验室 (南京大学), 江苏 南京 210023)

[2](南京晓庄学院 信息工程学院, 江苏 南京 211171)

通信作者: 徐锋, E-mail: xf@nju.edu.cn

摘 要: 提交日志是一类描述代码变更的自然语言文本, 对理解代码及代码演化过程十分重要. 受限于软件开发成本等因素, 开发人员往往不会精心撰写提交日志, 导致现有软件项目的提交日志质量堪忧. 鉴于此, 提交日志自动生成任务受到了广泛关注. 现有工作主要从开源软件项目中收集提交日志数据, 并基于此训练深度学习模型完成任务. 然而, 受限于开源项目数据质量, 现有工作难以取得令人满意的效果. 大语言模型通过在大规模代码和文本数据上的预训练, 学习了丰富的语言规律和世界知识, 能够生成高质量、自然且符合上下文的文本, 为提交日志生成提供了新的思路. 提出基于大语言模型的提交日志生成方法, 通过上下文学习、模型微调等手段将大语言模型用于提交日志生成, 使用两种示例检索方法增强上下文学习方法, 并从自然性和相关性角度分析生成文本, 研究大语言模型方法的优势与不足以及如何应对不足. 实验证实了大语言模型表现超过基线方法, 且思维链及更大规模模型能应对对代码变更理解不足的情况.

关键词: 提交日志生成; 大语言模型; 上下文学习; 模型微调

中图法分类号: TP311

中文引用格式: 徐圣斌, 贾林杰, 许汤智, 朱晓瑞, 余萍, 徐锋, 马晓星. 基于大语言模型的提交日志生成. 软件学报. http://www.jos.org.cn/1000-9825/7640.htm

英文引用格式: Xu SB, Jia LJ, Xu TZ, Zhu XR, Yu P, Xu F, Ma XX. Commit Message Generation Based on Large Language Models. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7640.htm

Commit Message Generation Based on Large Language Models

XU Sheng-Bin[1], JIA Lin-Jie[1], XU Tang-Zhi[1], ZHU Xiao-Rui[2], YU Ping[1], XU Feng[1], MA Xiao-Xing[1]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2](College of Information Engineering, Nanjing Xiaozhuang University, Nanjing 211171, China)

Abstract: Commit messages are natural language text that describe code changes and are crucial for understanding code and its evolution. Constrained by software development costs and related factors, developers often fail to carefully craft commit messages, resulting in unsatisfactory quality in existing software projects. As a result, commit message generation has gained widespread attention. Current work mainly involves collecting commit message data from open-source software projects and training deep learning models on this data to accomplish the task. However, due to the quality issues of open-source project data, existing methods struggle to achieve satisfactory performance. Large language models, by pre-training on vast amounts of code and text data, learn rich linguistic patterns and world knowledge, enabling the generation of high-quality, natural, and contextually appropriate text, thus providing new directions for commit message generation. This study proposes a commit message generation method based on large language models, in which large language models are applied to commit message generation through techniques such as in-context learning and model fine-tuning. Two example retrieval methods are employed to enhance the in-context learning approach. The generated text is then analyzed in terms of naturalness and relevance, and the strengths and limitations of the large language model-based methods, as well as ways to address these limitations,

* 基金项目: 国家自然科学基金 (62025202, 62072225); 江苏省前沿技术研发计划 (BF2024059)
收稿时间: 2024-09-04; 修改时间: 2025-03-17, 2025-12-17; 采用时间: 2026-01-23; jos 在线出版时间: 2026-05-27

are examined. Experiments demonstrate that large language models outperform baseline methods. In addition, chain-of-thought reasoning and larger-scale models can address the issue of insufficient understanding of code changes.

Key words: commit message generation; large language model (LLM); in-context learning; model fine-tuning

在软件开发实践中, 代码会不断变化以适应不断变化的需求和外部环境, 为了更好地管理软件演化, 版本控制系统 (如 Git) 要求开发人员编写提交日志记录和总结每次提交的代码更改. 提交信息以自然语言的形式呈现, 可以使开发人员无需深入具体的代码变更细节即可快速理解代码更改的高层次意图. 因此, 提交日志成为软件开发中不可或缺的一部分, 在软件理解和维护中发挥着重要作用. 图 1 展示了一个真实的代码变更及其对应的提交日志. 然而, 编写有意义且准确的提交日志对于开发人员来说并非易事. 此外, 敏捷开发、快速迭代的软件开发现状使得代码变更频繁, 进一步增加了开发人员的负担. 受软件开发成本等因素影响, 提交日志经常被开发人员忽视, 导致软件项目中存在提交日志缺失、质量低下等问题[1,2].

代码变更:

```
--- a/src/main/java/org/acra/ErrorReporter.java
+++ b/src/main/java/org/acra/ErrorReporter.java
@@ -393,7 +393,7 @@ void deletePendingReports() {
     * This method looks for pending reports and does the action required
     * depending on the interaction mode set.
     */
-  private void checkReportsOnApplicationStart() {
+  public void checkReportsOnApplicationStart() {

      // Delete any old unsent reports if this is a newer version of the app
      // than when we last started.
```

提交日志: Put ErrorReporter.checkReportsOnApplicationStart() back in the public api

图 1 GitHub 中一个真实的代码变更及其对应的提交日志

为了减轻开发人员负担, 提升软件项目中提交日志的质量, 研究人员提出了各种自动生成提交日志的技术. 早期工作[3−6]基于规则和模板生成提交日志, 生成的提交日志冗长, 缺失对代码变更关键意图的描述. 随后, 一些研究人员提出检索相似代码变更的提交日志作为当前变更的提交日志[7−9], 这类方法的效果完全取决于能否检索到相似的变更以及该变更对应提交日志是否良好. 最近, 基于深度学习的提交日志生成方法[10−20]成为主流并取得了一定的效果. 这些工作使用大量提交日志数据进行监督学习, 而 Tian 等人[2]对 5 个高质量 Java 开源项目的提交日志进行了研究, 发现平均约有 44% 的提交日志需要改进, 而更多开源项目的提交日志质量还要差于这 5 个项目. 依靠从开源项目中收集的提交日志无法构造出足够的高质量提交日志数据, 这也导致已有的基于学习的提交日志生成方法无法生成令人满意的提交日志.

另一方面, 大语言模型[21]作为人工智能领域的重大突破, 除了在自然语言生成、语义理解等领域有着良好表现[22], 在软件工程领域的许多任务上也取得了优秀的效果[23−26]. 其中, Geng 等人[26]将大语言模型的上下文学习 (in-context learning) 能力用于多意图注释生成任务, 仅提供少数代码注释对作为示例, 便可以生成比现有监督学习方法更自然、更充分、更有用的注释文本. 大语言模型从海量文本和代码数据学习到的丰富语言规律和世界知识成为解决提交日志生成问题的新思路. 基于上述分析, 本文提出基于大语言模型的提交日志生成方法, 通过上下文学习和模型微调验证大语言模型在提交日志生成任务的表现并分析大语言模型方法的优势与不足. 具体而言, 本文重点关注以下研究问题.

- RQ1: 大语言模型用于提交日志生成的效果: 大语言模型在零样本、少样本以及微调下的表现如何?
- RQ2: 大语言模型方法的优势与不足: 大语言模型生成的提交日志在可读性和相关性上是否优于已有方法? 还存在什么不足?
- RQ3: 大语言模型不足的应对: 思维链或更大规模模型能否应对大语言模型方法的不足?

针对 RQ1, 本文使用零样本、单样本、五样本的方式利用 GPT-3.5 模型, 还使用了两种示例检索方法对单样本和五样本方法进行增强, 同时对 LLaMA 模型进行微调, 增强其提交日志生成能力. 对比最好的基线方法, GPT-3.5

零样本学习设置在 BLEU-4[27]、METEOR[28]、ROUGE-L[29]指标上分别相对提升 22.1%、57.3%、2.5%, 微调后的 LLaMA 在这 3 个指标上分别相对提升 92.1%、39.0%、28.5%, 表明了大语言模型方法的有效性. 此外, 实验结果证明了单样本和五样本设置比零样本设置更能发挥大语言模型能力, 且示例检索排序能在此基础上进一步提升方法效果, 采用词元相似度检索的五样本方法 (GPT-5-shot-T) 对比零样本方法在 3 个指标上分别相对提升 91.3%、11.8%、27.0%.

针对 RQ2, 本文从测试集中随机采样 500 条数据, 对参考提交日志和各方法生成提交日志的可读性和相关性进行人工打分. 结果显示, GPT-5-shot-T 方法生成的提交日志在可读性和相关性上不仅超过了基线方法生成的提交日志, 还超过了参考提交日志. 具体分析 GPT-5-shot-T 方法生成提交日志不如参考提交日志的样本, 发现大语言模型难以生成需要领域知识的内容, 存在对变更整体理解有误、对变更理解不够深刻的问题, 偶尔还会出现“幻觉”.

针对 RQ3, 本文设计了采用思维链的提示模型, 并将 GPT-4 模型用于 RQ2 中 GPT-5-shot-T 方法表现不佳的样本. 通过实验验证了思维链提示和更大规模模型能应对对变更整体理解有误、对变更理解不够深刻的问题.

总而言之, 本文的主要贡献如下.

● 提出基于大语言模型的提交日志生成方法并设计实验验证方法的有效性, 方法同时考虑上下文学习和模型微调两种大语言模型利用方式.

● 深入分析大语言模型用于提交日志生成任务的优势与不足, 并验证思维链及更大规模模型能否应对大语言模型方法的不足.

本文第 1 节介绍大语言模型相关基础知识. 第 2 节介绍基于大语言模型的提交日志生成方法. 第 3 节介绍实验设置和实验结果. 第 4 节讨论其他相关问题以及本文研究的有效性威胁. 第 5 节介绍提交日志生成相关工作. 最后第 6 节进行总结.

1 基础知识

1.1 大语言模型

大语言模型由预训练语言模型发展而来. 早期的预训练语言模型 ELMo[30]通过双向 LSTM 学习上下文感知的单词表示, 这种单词表示用于下游任务时可以大幅提升任务效果. 后来, 基于能高度并行计算的 Transformer 结构[31], 研究人员提出了几种结构不同的预训练语言模型, 如 BERT[32]、GPT[33]和 BART[34]. 其中, BERT 采用纯编码器结构, 专注于理解文本内容; GPT 模型使用纯解码器结构, 擅长生成连贯文本序列; BART 采用编码器-解码器结构, 能够有效处理包括文本生成和理解在内的多种任务.

研究人员发现简单的扩展预训练语言模型的规模 (增加模型参数或增加训练数据) 可以有效提升模型在下游任务的效果[35], 许多工作试图通过训练更大规模的语言模型来探索模型的性能极限. 这一过程中, 研究人员发现在模型结构类似甚至相同的情况下, 更大规模的预训练语言模型相比小型模型表现出了不同的行为, 并能解决一系列复杂的任务. 如此, “大语言模型”一词逐渐被用来特指参数规模巨大、具有强大能力的预训练语言模型. 大语言模型的一个典型应用是 OpenAI 推出的对话模型 ChatGPT. ChatGPT 基于 GPT-3.5 和 GPT-4 系列模型, 在使用大量文本数据 (包括自然语言文本和代码) 预训练的基础上, 从人类反馈中进行强化学习 (reinforcement learning from human feedback, RLHF)[36]以进一步优化其生成文本质量和交互自然性, 实现和人类价值观的对齐. 本文中选择 ChatGPT 作为闭源大语言模型代表用于提交日志生成.

LLaMA[37]是 Meta AI 发布的开源大语言模型, 包含 4 个不同规模参数的模型 (7B、13B、30B 和 65B). 自发布以来, LLaMA 引起了学术界和产业界的广泛关注, 在各种基准测试中取得了良好的表现, 迅速成为最受欢迎的开源大语言模型之一. 本文选择 LLaMA 作为开源大语言模型代表用于提交日志生成.

1.2 模型微调

模型微调 (fine-tuning) 是深度学习领域使用预训练模型的一种常见方式, 在自然语言处理和计算机视觉任务

中广泛应用. 通过在特定任务的数据集上继续训练, 调整和优化预训练模型的权重使其适应于特定任务. 这种方法可以利用预训练模型已经学习到的丰富知识, 通过相对较少的数据和计算资源, 显著提升模型在特定任务的性能.

当前主流微调方式可分为两类: 全参微调 (full fine-tuning) 和参数高效微调 (parameter-efficient fine-tuning, PEFT). 全参微调是指模型全部参数权重参与更新以适配领域数据, 效果好, 在模型参数规模不大时被广泛使用. 随着预训练语言模型参数规模增大, 对大语言模型进行全参微调的成本变得十分高昂, 参数高效微调被提出以应对此问题. 参数高效微调原理是仅微调少量或额外的模型参数, 固定大部分预训练参数, 从而降低计算和存储成本. 先进的参数高效微调方法已经能达到与全参微调相近的效果.

根据可微调参数的位置划分, 参数高效微调可大致分为 3 类: Prefix-Tuning[38,39]、Adapter-Tuning[40,41]和 LoRA[42]. 其中, Prefix-Tuning 方法会在模型输入的前面添加一小段连续的可训练的参数 (前缀 Prefix), 在训练时通过调整这些额外的前缀参数来适应特定任务. Adapter-Tuning 则是在预训练模型内部的网络层之间添加小型的可训练模块 (适配器 Adapter) 来适配下游任务, 微调过程中只有适配器参数被更新. LoRA 全称是低秩适应 (low-rank adaptation), 它通过对预训练模型中的权重矩阵进行低秩分解更新来适配特定任务. 相比其他参数高效微调方法, LoRA 完全保留了模型结构, 在推理时可以做到无额外延时. 本文将 LLaMA 用于提交日志生成前会使用 LoRA 进行模型微调.

1.3 上下文学习

上下文学习是大语言模型的一种使用范式, 最早在 GPT-3[22]中被提出, 之后迅速成为研究热点. 上下文学习允许模型根据自然语言指令和少数任务示例 (即上下文) 来适应新任务, 无需进行参数更新, 大大减少了大语言模型的使用成本. 上下文学习需要设计格式化的自然语言提示 (prompt), 提示一般会包括一段任务描述、包含少数任务样本的示例和最后的待测样本. 将提示输入到大语言模型, 大语言模型会根据输入的上下文将提示补充完整, 大语言模型输出的部分可以看作其针对待测样本给出的答案. 形式化地说明, 使用 I 表示特定任务的任务描述, $\{(x_1,y_1),\ldots,(x_k,y_k)\}$ 表示该任务的 k 个样本, 其中 x_i 和 y_i 分别对应样本的输入和输出部分, 使用大语言模型对该任务的一个新样本 y_i 进行预测可以表示成:

$$LLM(I,f(x_1,y_1),\ldots,f(x_k,y_k),f(x_{k+1},\_))\rightarrow y'_{k+1} \tag{1}$$

其中, LLM 为大语言模型, $f(\cdot,\cdot)$ 为将任务样本转为自然语言提示的函数, "_"表示目标端内容的占位符, 即仅提供输入 x_{k+1}, 对应输出由大语言模型自动生成, y'_{k+1} 对应 LLM 针对 y_{k+1} 的输出. 将待测样本转为提示时只转化输入部分, 输出部分由大语言模型补全. 上下文学习的性能不仅依赖于大语言模型的理解和泛化能力, 还受输入到大语言模型的提示的影响. 研究表明, 任务描述、示例个数、示例表示、示例在提示中的排列顺序等都会影响上下文学习的效果[43]. 根据提示中使用的示例个数不同, 上下文学习可以具体分为零样本 (zero-shot) 和少样本 (few-shot) 设置. 零样本指提示中不包含示例, 只有任务描述和待测样本: 少样本则是提供少数样本作为示例, 根据提供样本数目不同, 可以细分成 k-shot 设置. 对于 ChatGPT 模型, 本文使用上下文学习方式将其用于提交日志生成.

2 基于大语言模型的提交日志生成方法

在应用大语言模型时, 本文考虑了上下文学习和模型微调两种方式, 接下来具体介绍如何应用大语言模型, 包括上下文学习中的提示模板设计、示例检索排序、模型微调时的微调数据构造.

2.1 提示模板设计

已有的提交日志生成工作[10,13,18,26]将提交日志生成问题建模为以代码变更为输入、提交日志为输出的"翻译"或"总结"问题. 本文采用同样的建模方式, 直接以 diff 形式的代码变更为输入, 提交日志为输出. 图 2 展示了上下文学习所用提示模板. 提示模板设计遵循了当前的最佳实践, 提示第 1 行包括了详细的任务描述, 还指定了输出文本长度为一行; 提示模板中间为 k 个示例, 示例中的不同内容被明确分隔以区分; 提示最后是待测样本, 其与示例格式相同但提交日志部分留空待大语言模型补全.

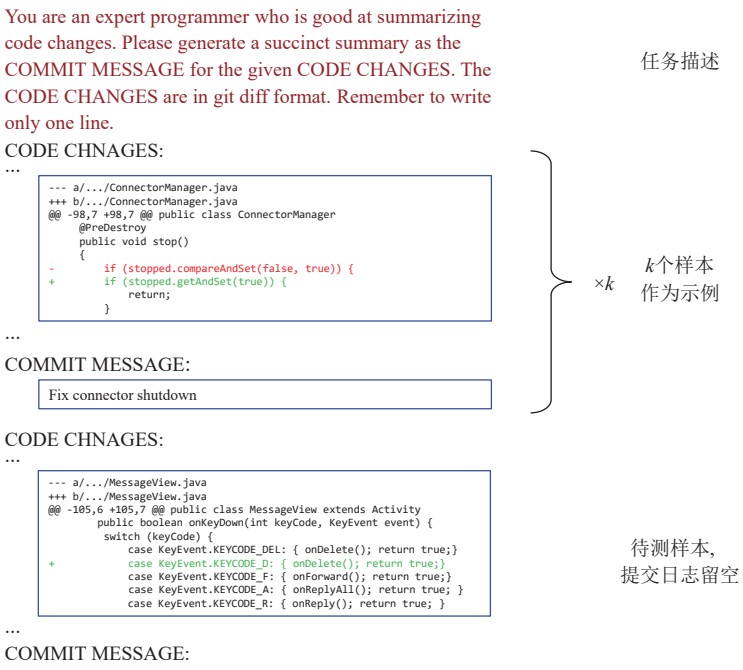


图 2 提交日志生成任务所用提示模板

2.2 示例检索排序

对不同待测样本使用固定的示例 (随机从语料库中选择) 是最简单的提示生成方法, 但这种提示往往难以完全发挥大语言模型的上下文学习能力, 选择与待测样本相似的样本作为示例能帮助大语言模型更好理解任务[26,44]. 图 3 展示了结合示例检索排序的上下文学习框架.

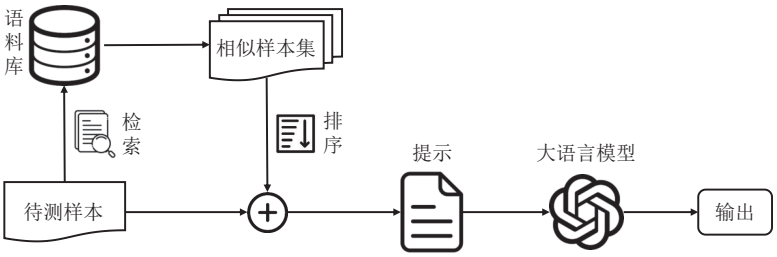


图 3 结合示例检索排序的上下文学习框架

该框架下, 大语言模型生成提交日志的过程如下: 1) 从语料库中检索与待测样本相似的代码变更, 得到相似样本集合, 每个样本包含代码变更和对应提交日志; 2) 将相似样本集中样本排序后和待测样本一起填充到图 2 所示提示模板得到最终提示; 3) 将提示输入到大语言模型获取其输出结果. 本文使用了两种不同的示例检索方法, 分别基于代码变更的词元相似度和语义相似度. 接下来对两种方法做详细介绍.

● 基于词元相似度的方法. 针对代码变更的检索, Liu 等人[9]提出了一种基于词元相似度的方法 NNGen, 取得了很好的效果. 本文使用的基于词元相似度的检索方法采用了和 NNGen 一样的步骤. 具体而言, 先将目标代码变更 (待测样本) 和语料库中的代码变更使用词袋模型表示, 这一步会忽略代码变更中词元的顺序, 将每个代码变更表示成一个向量, 向量中每个维度的值对应特定词在当前代码变更中的词频. 基于代码变更向量, 可以计算语料库中代码变更和目标代码变更的余弦相似度, 然后根据相似度大小排序, 选择语料库中最相似的 n 个代码变更对应

的样本作为目标代码变更的相似样本集. 和 NNGen 方法一样, 代码变更中的文件头 (以“– – –”和“+++”为行首的行)、代码行的变更标记 (行首的“+”“–”和空格符, 分别对应添加、删除和不变) 以及代码中的元素 (如关键字、标识符、常量、运算符等) 都会参与词频统计, 这里不会对标识符进一步分词, 而是保留标识符的原始形式.

● 基于语义相似度的方法. 不同项目的代码变更在词元层面往往存在较大差异导致词元相似度在跨项目设置下难以检索到相似的代码变更, 这时理解代码变更语义对检索相似代码变更十分重要. 本文采用 Xu 等人[14]工作中使用的代码变更建模方法 CoDiSum 获取代码变更的语义向量表示并基于此进行语义相似度的检索. 具体而言, 先利用语料库中的数据训练一个 CoDiSum 模型, 然后使用 CoDiSum 模型的编码器部分将目标代码变更和语料库中的代码变更编码成向量. 考虑到 CoDiSum 编码器会将一条代码变更数据编码成向量序列, 本文中使用平均的方式聚合向量序列中的信息, 得到语义向量. 基于语义向量表示, 计算向量的余弦相似度作为代码变更的语义相似度, 选出 n 个样本作为目标代码变更的相似样本集.

本文对每个目标代码变更会检索 10 个相似样本 (即 $n = 10$). 填充样本到提示模板时会根据少样本学习的具体设置进一步对相似样本进行筛选和排序, 依据是目标代码变更和相似样本代码变更的 BLEU-4 值 (值越大表明代码变更文本越相似). 以 5-Shot 设置为例, 此时需要向提示模板填充 5 个样本作为示例, 会选择 BLEU-4 值最大的 5 个样本填充. 考虑大语言模型更有可能预测提示末尾出现的答案[45], BLEU-4 值越大的样本会填充到离待测样本更近的位置.

2.3 微调数据构造

微调设置下同样需要以提示模板将训练样本及待测样本转为文本形式. 相比上下文学习方法, 微调使用的提示模板更加简单, 因为模型可以从数据中学习任务的具体要求, 不需要显式说明. 图 4 展示了将⟨代码变更, 提交日志⟩ 二元组转成适用于微调的形式的提示模板. 该模板的任务描述部分只简短的说明需要为代码变更生成提交日志, 代码变更和提交日志前分别使用“CODE CHANGES”和“COMMIT MESSAGE”指示. 微调时使用下一词元预测任务, 每条数据被完整地输入到模型. 微调完成之后, 待测样本输入到模型, 此时将提交日志部分留空待模型补全.

Please generate a commit message for the given code changes.
CODE CHANGES:

```
--- a/.../ConnectorManager.java
+++ b/.../ConnectorManager.java
@@ -98,7 +98,7 @@ public class ConnectorManager
     @PreDestroy
     public void stop()
     {
-        if (stopped.compareAndSet(false, true)) {
+        if (stopped.getAndSet(true)) {
             return;
         }
```

COMMIT MESSAGE:

Fix connector shutdown

图 4 微调设置下提交日志生成任务所用提示模板

3 实验设置与结果

3.1 实验设置

数据集: 提交日志生成任务使用 Xu 等人[14]所用的包含 90 661 个⟨代码变更, 提交日志⟩ 二元组的数据集. 该数据集来源于 GitHub 上 1 000 个流行的 Java 项目, 经过严格的过滤和去重, 具有较高的质量. 数据集中每个代码变更保留 3 行上下文, 这是 git diff 的默认上下文行数, 在可读性和简洁性中取得了较好平衡, 本文复用这一设置. 数据集已经被随机划分成训练集, 验证集和测试集, 其中训练集大小为 75 000, 验证集大小为 8 000, 测试集大小为 7 661. 本次实验中沿用了已有的数据划分, 上下文学习设置下, 测试集中样本为待测样本, 训练集作为供检索的语料库; 微调设置下, 使用完整训练集进行微调, 验证集评估微调效果, 最后在测试集上测试. 表 1 展示了数据集中代码变更文件行数的统计信息, 包括平均值、最大最小值、四分位数. 从中可以看出数据集所包含代码变更文件的

行数主要在 10–27 行之间, 还有少部分代码变更文件行数超过 27 行. 图 5 展示了测试集中代码变更文件的具体行数分布.

表 1　数据集中代码变更文件行数的统计结果

数据集	样本数量	代码变更文件行数统计 (行)					
		平均值	最小值	最大值	下四分位数	中位数	上四分位数
训练集	75 000	22.0	10.0	97.0	14.0	19.5	27.0
验证集	8 000	21.9	10.0	91.0	14.0	19.0	27.0
测试集	7 661	21.8	10.0	85.0	14.0	19.0	27.0

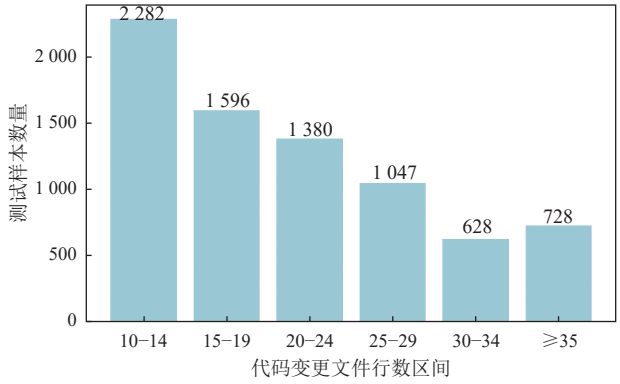


图 5　测试集中代码变更文件行数分布

基线方法: 本次实验选用 NNGen 方法[9]、CoDiSum 方法[14]和 FIRA 方法[18]作为基线方法. 其中, NNGen 是一种基于检索的方法, 使用词袋模型将代码变更表示为向量, 基于向量间的余弦相似度为输入寻找相似的代码变更然后重用其提交日志; CoDiSum 和 FIRA 是现有最先进的基于学习的提交日志生成方法, 都采用编码器-解码器架构, CoDiSum 会同时编码代码变更结构和代码自然语义并结合, 其解码器部分引入拷贝网络[46]以支持从输入序列拷贝, FIRA 方法使用细粒度图表示代码变更, 然后使用图神经网络进行编码, 解码器部分同样引入拷贝网络.

实现细节: 实验选用了 GPT-3.5 和 LLaMA 作为大语言模型代表, 前者用于测试上下文学习能力, 后者用于测试模型微调后的效果. GPT-3.5 是闭源模型, 通过调用 API 的方式使用, 调用时选用的模型名称为“gpt-3.5-turbo-0613”, 设置 temperature 为 0 以减少生成文本的不确定性. 测试时除了使用零样本 (0-shot) 和单样本 (1-shot) 设置, 还使用了五样本 (5-shot) 设置, 受输入序列长度限制, 没有进一步增加示例样本个数. 微调设置下, 受算力限制, 只对“llama-7b”进行了 LoRA 微调[42], 微调参数类型设置为 Q 矩阵和 V 矩阵, 低秩矩阵的秩设置为 8, 缩放因子 alpha 的值设置为 16. 微调的批大小设置为 128, epoch 设置为 3, 优化器使用 AdamW[47], 学习率为 0.000 3, 前 100 步用于热身, 每训练 200 步会在验证集上验证, 最后使用在验证集上表现最好的模型用于测试.

3.2　RQ1: 大语言模型的效果

表 2 展示了 CoDiSum 及不同设置下大语言模型用于提交日志生成任务的结果. 对大语言模型, 使用零样本学习设置的 GPT-0-shot 在 BLEU-4、METEOR、ROUGE-L 这 3 个指标上都超过了最好的基线方法 FIRA. 其中, BLEU-4 值相对提升了 22.1%; METEOR 值相对提升了 57.3%; ROUGE-L 值相对提升了 2.5%. 这表明大语言模型在仅给出详细任务描述的情况下已经能很好地完成提交日志生成任务.

GPT-1-shot 和 GPT-5-shot 对应使用固定示例时大语言模型少样本学习取得的效果. 与 GPT-0-shot 相比, GPT-1-shot 在 BLEU-4、METEOR、ROUGE-L 指标上分别相对提升 17.7%、3.5%、2.5%, 表明单个示例样本也能显

著提升大语言模型性能. 另外, 对比 GPT-5-shot 和 GPT-1-shot 的结果, GPT-5-shot 在 3 个指标上都只有微小的相对提升 (小于 1.0%), 表明简单增加示例样本不是提升大语言模型效果的有效手段.

GPT-1-shot-T 和 GPT-5-shot-T 对应使用基于词元相似度进行示例检索时大语言模型少样本学习取得的效果. 与 GPT-1-shot 对比, GPT-1-shot-T 在 BLEU-4、METEOR、ROUGE-L 指标上分别相对提升 43.1%、0.7%、15.5%. 与 GPT-5-shot 相比, GPT-5-shot-T 在 BLEU-4、METEOR、ROUGE-L 指标上分别相对提升 61.0%、7.8%、22.9%. 上述结果表明在提示中示例个数相同的情况下, 基于词元相似度的检索方法有效提升了大语言模型的效果. 此外, GPT-5-shot-T 比 GPT-1-shot-T 在 3 个指标上分别相对提升 13.6%、7.1%、7.4%, 表明增加相似样本数量能有效提升模型效果.

表 2　大语言模型用于提交日志生成任务的有效性结果 (%)

方法	BLEU-4	METEOR	ROUGE-L
NNGen	2.16	4.34	9.44
CoDiSum	2.19	7.46	20.02
FIRA	4.26	9.18	20.59
GPT-0-shot	5.20	14.44	21.89
GPT-1-shot	6.12	14.95	22.43
GPT-5-shot	6.18	14.97	22.62
GPT-1-shot-T	8.76	15.06	25.90
GPT-5-shot-T	**9.95**	**16.14**	**27.81**
GPT-1-shot-S	7.28	13.70	23.79
GPT-5-shot-S	8.49	14.82	25.70
LLaMA	8.18	12.76	26.45

GPT-1-shot-S 和 GPT-5-shot-S 使用基于语义相似度的示例检索方法. 与 GPT-1-shot 和 GPT-5-shot 相比, 在示例数量相同的情况下, GPT-1-shot-S 和 GPT-5-shot-S 在 BLEU-4 和 ROUGE-L 指标上有明显提升, 再次证明相似样本检索的有效性. GPT-5-shot-S 比 GPT-1-shot-S 的相对提升再次证明增加相似样本数量能提升模型效果. 与 GPT-1-shot-T 和 GPT-5-shot-T 相比, 示例数量相同情况下, GPT-1-shot-S 和 GPT-5-shot-S 在全部指标上都更差且有明显差距, 表明在提交日志生成任务上, 词元上更相似的示例比语义上更相似的示例更有效. 后文图 6 展示了测试集中一个待测样本和根据词元相似度和语义相似度检索所得相似样本. 从文件头可以看出词元相似样本与待测样本来自相同项目, 而语义相似样本则不是. 从检索原理分析, 基于词元相似度的检索方法倾向于选择相同项目的样本作为相似样本. 同项目的样本能给大语言模型提供更多信息 (如语言风格、用词偏好等), 这是导致两种示例检索方法性能差异的主要原因. 此外, 大模型预训练时往往也是同项目的数据一起输入, 语义相似度检索形成的提示输入与训练输入分布偏差更大, 因此效果更差.

微调后的 LLaMA 模型在 BLEU-4、METEOR、ROUGE-L 指标上对比最好的基线方法 FIRA 分别相对提升了 92.1%、39.0%、28.5%, 再次表明大模型相对基线方法的优势. 另一方面, 微调 LLaMA 并没有在这些指标上超过最好的上下文学习方法 GPT-5-shot-T, 这可能和 LLaMA 本身参数量偏小且没有进行全量微调等因素有关, 此外, 微调过程仅使用提交日志生成任务数据, 导致模型遗忘原有知识, 影响最终生成提交日志效果.

综上所述, 在提交日志生成任务上, 大语言模型在上下文学习和微调设置下表现均优于基线方法, 表明了大语言模型方法的有效性. 上下文学习设置下, 示例检索排序能有效提升模型效果且基于词元相似度的检索方法明显优于基于语义相似度的检索方法, 提示中增加相似样本数量也能提升大语言模型效果. 然而, 表现最好的 GPT-5-shot-T 方法在 BLEU-4、METEOR、ROUGE-L 指标上的绝对值分别只有 9.95%、16.14%、27.81%, 还有很大的提升空间, 需要对大模型的优势与不足进行深入分析.

3.3 RQ2: 大语言模型的优势与不足

为深入分析大语言模型的优势与不足, 本节在 RQ1 结果的基础上进一步进行定量分析, 先从测试集中随机选

取 500 条数据, 然后从提交日志的可读性和提交日志与代码变更的相关性两个维度对参考文本和生成文本进行标注, 最后基于标注结果进行分析. 可读性是指提交日志内容的易读和易理解程度, 主要考虑文本语法上是否正确, 语义上是否明确无歧义. 相关性是指提交日志与代码变更情况的匹配程度, 主要考虑文本是否缺失必要信息, 是否包含错误信息.

标注过程十分严格, 每条数据会对应一个网页, 页面包含 1 个代码变更和 4 条提交日志 (分别是参考提交日志以及 CoDiSum、GPT-5-shot-T、LLaMA 生成的提交日志), 页面中不会包含提交日志文本的具体来源信息. 标注人员会根据页面信息对每条提交日志的可读性和相关性打分, 两者的分数范围都为 1–5, 分别对应很差、较差、中等、较好、很好. 为减少标注人员个人偏好对标注结果的影响, 每条数据会由两个标注人员独立打分. 对不同标注人员针对同一条数据的打分计算 Kendall 相关系数[48], 可读性分数和相关性分数的相关系数平均值分别为 0.54 和 0.59, 显示出较强的正相关性, 说明标注人员对相同数据的打分具有较高的一致性. 本节最终将两个标注人员的打分求均值 (保留整数位) 之后作为每条提交日志的最终得分. 整个标注任务由 5 个有相关编程经验的计算机专业硕士研究生完成, 总共花费约 50 人时.

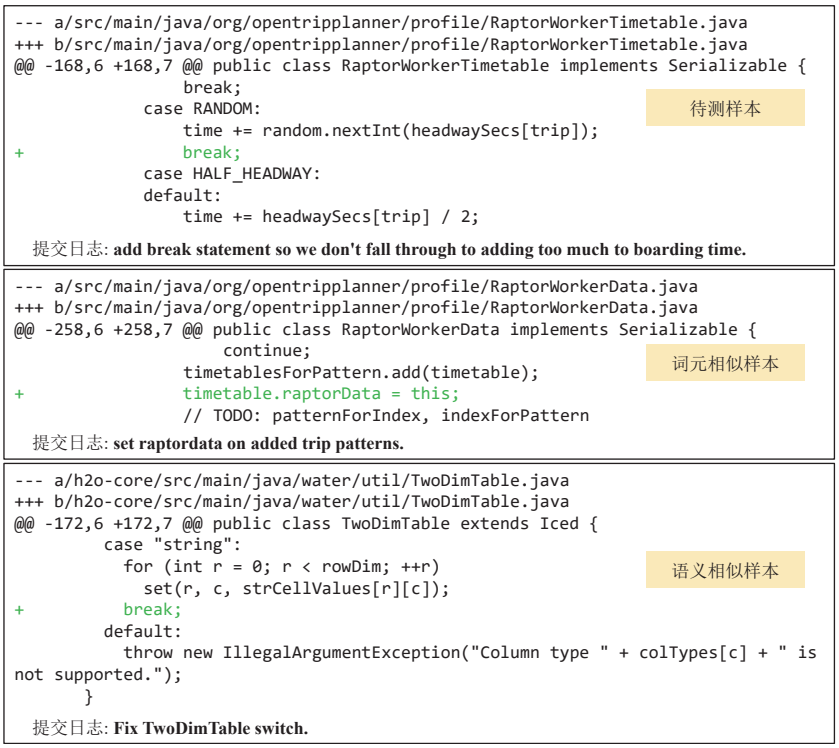


图 6 提交日志生成任务样本的词元相似样本和语义相似样本

图 7(a) 展示了提交日志可读性打分的结果分布. 总体上, 绝大部分提交日志的得分大于等于 3 分, 可读性在中等及以上, 只有 CoDiSum 方法会生成相对较多 (43/500) 可读性差的提交日志. 继续检查这些可读性差的提交日志, 其中 37 条是因为包含未知符号“UNK”, 6 条是因为包含连续重复单词. 基于大语言模型的方法生成的提交日志不存在上述两类问题, 说明大语言模型生成的文本在可读性上比监督学习方法有明显优势. 分析可读性中等及以上的提交日志, CoDiSum 方法各得分的出现次数全面小于参考提交日志对应得分的出现次数, LLaMA 微调方法生成的提交日志分数 4 出现次数最高, GPT-5-shot-T 生成的提交日志分数 5 出现次数最高. 与参考提交日志相

比, LLaMA 微调方法生成的提交日志没有实现全面超越, 可能原因是微调过程使用的提交日志数据质量不够高导致大语言模型文本生成能力下降. GPT-5-shot-T 方法生成文本的可读性全面超越了参考提交日志及 LLaMA 微调生成的提交日志, 说明上下文学习能更好地发挥大语言模型生成流畅文本的能力.

图 7(b) 是对提交日志相关性打分后所得结果分布. 以参考提交日志的得分分布为基准, CoDiSum 生成的提交日志相关性得分更多分布在较差和很差, 说明该方法生成的提交日志在相关性上与参考提交日志差距较大. LLaMA 微调后生成的提交日志比 CoDiSum 生成的提交日志更好, 但其得分也较多分布在中等、较差和很差, 表明其在相关性上与参考提交日志还存在一定差距. 微调后的 LLaMA 也没能很好理解代码变更, 原因可能在于 LLaMA 模型参数规模不够大、微调数据质量不够好. GPT-5-shot-T 方法生成的提交日志相比参考提交日志更多分布在较好和很好两类, 表明大语言模型通过上下文学习能很好生成与代码变更一致的提交日志.

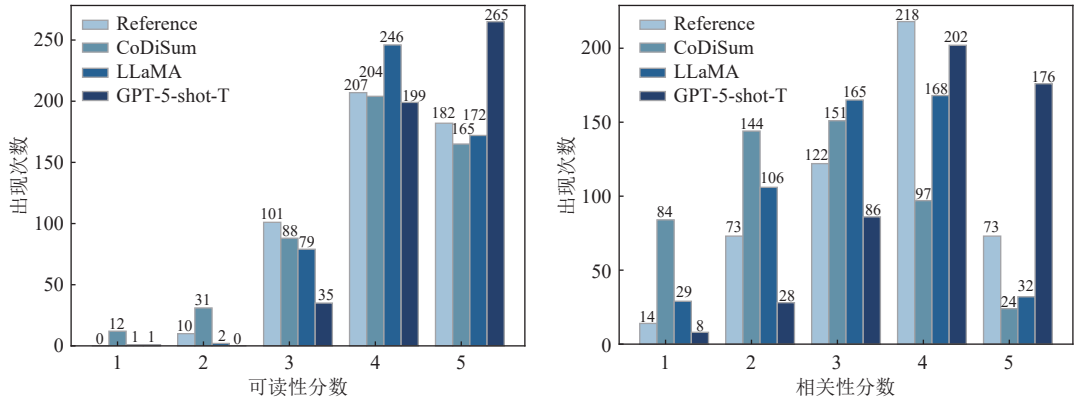


图 7 提交日志的可读性和相关性打分结果

深入对比 GPT-5-shot-T 方法生成的提交日志和参考提交日志的相关性得分, GPT-5-shot-T 生成提交日志得分高于对应参考提交日志的数据有 249 条, 其中有 102 条得分差距在两分及以上, 低于对应参考提交日志的数据有 100 条, 其中有 32 条差距在两分及以上. 分析这些得分差距在两分及以上的数据, 参考提交日志相关性得分更低的主要原因有 3 个.

● 参考提交日志包含特定领域知识 (58 条). 标注人员缺乏所标注数据对应软件项目相关知识, 只能查看部分代码变更内容. 当参考提交日志中包含项目开发人员才能理解的领域知识时 (例如“Fix bad merge”“Fixed a regression in View.cancelLongPress”), 标注人员难以理解或难以判断是否准确. 而 GPT-5-shot-T 生成的提交日志主要描述代码如何变更, 很少生成这些领域知识相关内容, 因此相关性得分会更高.

● 参考提交日志缺乏具体信息 (34 条). 部分参考提交日志仅描述了代码变更中的部分信息, 或者使用宽泛的描述 (例如“Resolved some findbugs issues”“Fix swing graphic bug”), 与更具体的代码变更描述 (例如“Suppressed findbugs warning in GarbageCollectionMetricSet”“Fix incorrect yPoints assignment in WrapGraphics”) 相比, 标注人员会给这种宽泛描述打出较低分数.

● 参考提交日志包含错误信息 (10 条). 少部分参考提交日志包含错误信息甚至内容完全错误, 例如代码变更内容为“Refactor code to use InvUtils.getInventory instead of Utils.getInventory”, 对应参考提交日志却是“Missed a line”.

GPT-5-shot-T 方法生成的提交日志相关性得分更低的原因大致有 4 类.

● 对变更整体理解有误 (14 条). 生成的提交日志没有正确反映整体代码变更情况. 例如对“TaskContainer-

move meaningful toString to the base class”这一提交, 生成“add toString to TaskContainer for better debugging”. 方法只理解了变更中添加“toString”部分, 没有考虑其中还有删除“toString”的部分.

● 对变更理解不够深刻 (13 条). 生成的提交日志正确反映了整体代码变更情况, 但没有变更细节内容或变更细节描述错误. 例如对“Improve javadoc formatting”这一提交, 方法生成“Updated class javadoc to reflect recent changes to FilterToBeanProxy”这一提交日志, 此次提交确实更新了 FilterToBeanProxy 类的 Javadoc 注释. 但是并没有改变注释内容, 仅仅调整了其格式.

● 缺少领域知识 (4 条). 生成的提交日志正确反映了当前代码变更, 但参考提交日志中还包含一些代码变更外的知识, 对比之下, GPT-5-shot-T 生成的提交日志相关性得分较低. 例如: 参考提交日志为“Add a bogus method in otherwise empty test, to prevent build breakage”, 生成了“Add a test for JsonPointer”.

● 生成“幻觉” (1 条). 生成的提交日志中包含没有事实支撑的内容. 例如, 为参考提交日志为“For unrecoverable projects, rename them with a suffix so the next time we won’t try to recover them again”的提交生 成“Rename corrupted project files instead of deleting them”, 实际上代码变更中没有任何有关删除文件的代码.

综上所述, 基于大语言模型的提交日志生成方法在生成文本的可读性和相关性上均优于 CoDiSum 方法. 可读性方面, 基于大模型的方法解决了 CoDiSum 方法容易生成未知词和重复单词的问题, 其生成的文本与参考文本质量类似甚至有所超越. 相关性方面, LLaMA 微调方法生成的文本不如参考文本, 但 GPT-3.5 结合示例检索排序生成的文本已经能超越已有的参考文本. 深入研究发现, 大语言模型生成的文本更多描述代码如何变更, 难以生成参考提交日志中存在的有关领域知识的内容. 此外, 大语言模型在变更理解上仍存在一定不足, 且偶尔出现“幻觉”.

3.4 RQ3: 大语言模型不足的应对策略

要应对大语言模型难以生成包含领域知识的内容这一问题, 思路是从软件项目中检索变更相关的缺陷报告、需求文档等内容并提供给大语言模型, 然而本文所使用数据集并不包含相关内容, 因此无法进行验证. 针对对变更整体理解有误、对变更理解不够深刻和生成“幻觉”的问题, 本文采用两种思路进行应对: 思维链[49]和更大参数模型 (如 GPT-4).

具体而言, 针对提交日志生成任务, 本文设计了如图 8 所示的采用了思维链的提示模板. 该模板在指令部分要求大模型从上到下描述所有的代码变更情况后进行总结以形成提交日志, 提示模板中还提供了一个采用该思路进行提交日志生成的例子, 模板中“待测样本”部分为需要填充的部分. 然后将其用于 RQ2 中 GPT-5-shot-T 表现不佳的 32 条数据. 使用更大参数模型则是采用与 GPT-5-shot-T 方法相同的方案, 区别在于调用时选用 GPT-4 系列的“gpt-4-0613”模型. 本文还实验了将思维链与更大参数模型结合取得的效果, 表 3 展示了实验结果, 表格中的分母为对应类别的样本数量, 分子为使用的策略能应对的样本数量 (“应对”指该策略生成的提交日志接近甚至超过参考提交日志).

从表 3 中看到, 仅使用思维链 (调用 GPT-3.5) 能应对 12/14 (85.7%) 的对变更整体理解有误的情况, 但只能应对 5/13 (38.5%) 的对变更理解不够深刻的情况. 这说明思维链中先列出所有变更然后进行总结的方式可以很好地帮助大模型理解变更整体, 但没法帮助 GPT-3.5 深入理解代码变更. 将模型从 GPT-3.5 换成 GPT-4 可以全方面提升大语言模型的变更理解能力, 其能应对 13/14 (92.9%) 的对变更整体理解有误的情况和 10/13 (76.9%) 的对变更理解不够深刻的情况. 此外, 使用思维链提示并调用 GPT-4 模型 (组合方法) 取得最好的效果, 能应对 14/14 (100%) 的对变更整体理解有误的情况和 12/13 (92.3%) 的对变更理解不够深刻的情况. 而应对缺少领域知识的情况, 上述策略并没有效, 这也符合预期. 此外, 上述策略都能应对生成“幻觉”的情况, 但是考虑到只有一个“幻觉”样本, 结论的有效性还待进一步研究确认.

综上所述, 针对大语言模型对代码变更理解不足的情况, 采用思维链或选用参数更大能力更强的模型能有效应对. 而如何应对无法生成领域知识内容和生成“幻觉”的情况还有待深入研究.

You are an expert programmer who is good at summarizing code changes. Please generate a succinct summary as the COMMIT MESSAGE for the given CODE CHANGES. The CODE CHANGES are in git diff format. To generate a good commit message, first describe the changes in the code from top to bottom, then summarize these changes to form the commit message. The final commit message should contain only one statement and in the following format:
COMMIT MESSAGE: {message}
{message} is the generate commit message.
CODE CHANGES:

```
--- a/hazelcast/src/main/java/com/hazelcast/map/impl/LazyEntryView.java
+++ b/hazelcast/src/main/java/com/hazelcast/map/impl/LazyEntryView.java
@@ -1,6 +1,8 @@
 package com.hazelcast.map.impl;

 import com.hazelcast.core.EntryView;
+import com.hazelcast.map.merge.HigherHitsMapMergePolicy;
+import com.hazelcast.map.merge.LatestUpdateMapMergePolicy;
 import com.hazelcast.map.merge.MapMergePolicy;
 import com.hazelcast.map.merge.PassThroughMergePolicy;
 import com.hazelcast.map.merge.PutIfAbsentMapMergePolicy;
@@ -51,16 +53,18 @@ class LazyEntryView<K, V> implements EntryView<K, V> {
     }

     public V getValue() {
-        if (validMergePolicyForValueNullCheck(mergePolicy)) {
+        if (returnRawData(mergePolicy)) {
             return value;
         }
         value = serializationService.toObject(value);
         return value;
     }

-    private boolean validMergePolicyForValueNullCheck(MapMergePolicy mergePolicy) {
+    private boolean returnRawData(MapMergePolicy mergePolicy) {
         return mergePolicy instanceof PutIfAbsentMapMergePolicy
-                || mergePolicy instanceof PassThroughMergePolicy;
+                || mergePolicy instanceof PassThroughMergePolicy
+                || mergePolicy instanceof HigherHitsMapMergePolicy
+                || mergePolicy instanceof LatestUpdateMapMergePolicy;
     }

     public void setValue(V value) {
```

The changes in the code include:
- added imports for HigherHitsMapMergePolicy and LatestUpdateMapMergePolicy.
- changed the method call from validMergePolocyForValueNullCheck to returnRawData in getValue.
- renamed method validMergePolicyForValueNullCheck to returnRawData.
- included checks for HigherHitsMapMergePolicy and LatestUpdateMapMergePolicy in returnRawData.

The key changes include method renaming and checks including. In summary:
COMMIT MESSAGE: update method name to returnRawData and add more checks in it.

CODE CHANGES:

```
{待测样本}
```

The changes in the code include:

图 8 采用了思维链的提示模板

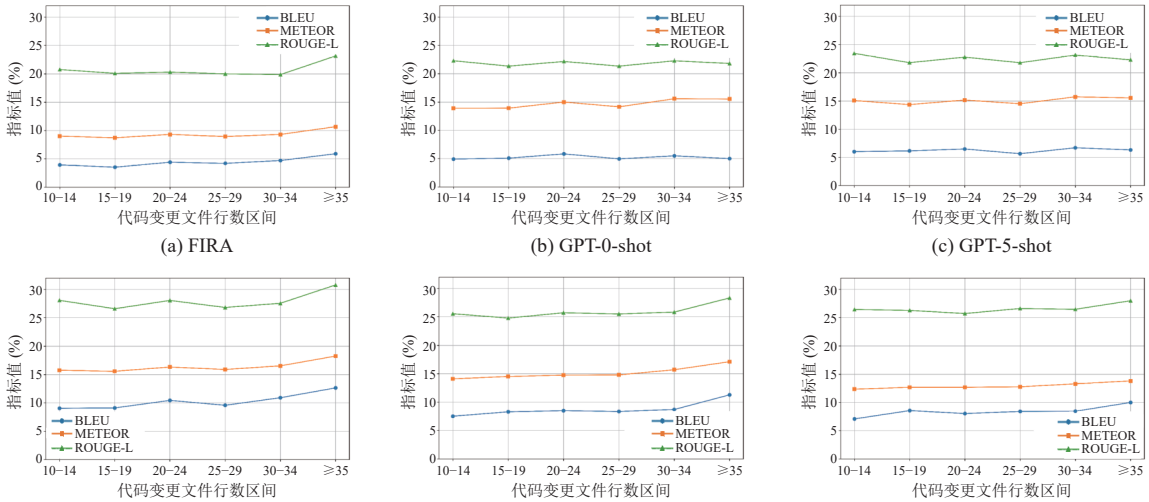
表 3 本文采用的应对大语言模型不足的策略取得的效果

类别	思维链	GPT-4	结合方法
对变更整体理解有误	12/14	13/14	14/14
对变更理解不够深刻	5/13	10/13	12/13
缺少领域知识	0/4	0/4	0/4
生成“幻觉”	1/1	1/1	1/1

4 讨 论

4.1 不同代码变更文件行数下各方法的效果

本节探讨代码变更文件行数是否会影响各方法生成提交日志的质量, 图 9 展示了 FIRA、GPT-0-shot、GPT-5-shot、GPT-5-shot-T、GPT-5-shot-S、LLaMA 这 6 种方法对不同行数的代码变更文件生成提交日志时, BLEU、METEOR 和 ROUGE-L 指标值的变化情况. 从图 9 中可以看出, 对 GPT-0-shot 和 GPT-5-shot 方法, 代码变更文件行数和指标值没有明显的相关关系; 而对 FIRA、GPT-5-shot-T、GPT-5-shot-S、LLaMA 这 4 种方法, 代码变更文件行数在 10–34 行之间时, 生成的提交日志指标值比较稳定, 在行数大于等于 35 时, 生成的提交日志在指标上有明显的提升. 一种解释是对行数大于等于 35 的这种稍复杂的代码变更, 开发人员愿意进行规范的描述, 使得收集的数据质量较高, 对特别简单的代码变更, 开发人员可以直接理解变更内容, 因此编写提交日志会更随意. 注意, 根据表 1, 本文测试集中代码变更文件的行数最大值为 85, 因此图 9 中最后“≥35”一列是指行数在 35–85 这一区间的代码变更文件, 本文方法在更复杂的代码变更 (行数更多) 上生成提交日志的结果还有待深入研究.



(a) FIRA (b) GPT-0-shot (c) GPT-5-shot

(d) GPT-5-shot-T (e) GPT-5-shot-S (f) LLaMA

图 9 各种方法对不同行数的代码变更文件生成提交日志时指标值的变化情况

4.2 其他大语言模型的效果

随着大语言模型相关技术的发展, 越来越多的大语言模型被发布出来, 本节考察了一些常见大语言模型用于提交日志生成任务的效果并进行讨论. 具体来说, 本节实验使用了 GPT-5-shot-T 设置, 提示中都使用 5 个通过词元相似度检索得到的示例, 仅将 gpt-3.5-turbo-0613 模型替换为其他模型, 具体考察的模型包括 DeepSeek-V2-Chat-0628、DeepSeek-Coder-V2-Instruct-0724、Qwen2-72B-Instruct 和 Yi-Medium. 实验结果如表 4 所示, 加粗表示该列中的最优结果.

表 4 其他大语言模型用于提交日志生成的结果 (%)

方法	BLEU-4	METEOR	ROUGE-L
gpt-3.5-turbo-0613	**9.95**	16.14	**27.81**
DeepSeek-V2-Chat-0628	8.92	**16.19**	27.41
DeepSeek-Coder-V2-Instruct-0724	8.12	15.92	26.58
Qwen2-72B-Instruct	5.48	15.06	23.66
Yi-Medium	6.88	14.80	24.02

对比上述大语言模型结果, 可以看到 GPT-3.5 模型仍处于领先地位, 其中 DeepSeek-V2-Chat-0628 模型效果已

经接近 GPT-3.5, 其他 3 个大语言模型与 GPT-3.5 还存在一定差距. 另一方面, 与表 2 中基准方法取得的结果相比, 最差的大语言模型方法也能在 3 个指标上全面超越最好的基准方法, 再次说明大语言模型与基准方法在提交日志生成任务上存在显著能力差距, 基于大语言模型的提交日志生成方法有很大优势.

4.3 直接使用 LLaMA 的效果

本节研究不微调 LLaMA, 而是将其直接用于提交日志生成任务时的效果. 具体来说, 本节中采用零样本 (0-shot)、单样本 (1-shot) 和五样本 (5-shot) 设置, 每条测试数据的提示和前文所述 GPT-0-shot、GPT-1-shot 和 GPT-5-shot 方法的提示一致. 表 5 展示了方法的结果.

表 5　直接将 LLaMA 用于提交日志生成的结果 (%)

方法	BLEU-4	METEOR	ROUGE-L
LLaMA (微调)	8.18	12.76	26.45
LLaMA-0-shot	1.37	4.37	7.00
LLaMA-1-shot	3.34	7.74	12.19
LLaMA-5-shot	0.61	1.65	3.34

具体而言, 零样本、单样本和五样本设置下 LLaMA 的结果远远比不上微调后的 LLaMA, 也不如表 2 中 GPT-0-shot、GPT-1-shot 和 GPT-5-shot 的结果, 主要原因有两个: 一是 LLaMA 的参数量只有 7B, 远小于 GPT-3.5 的参数量, 基础能力不如 GPT-3.5, 难以从多个示例中有效提取并内化提交日志模式; 二是本文使用的 LLaMA 并未经过指令微调, 无法理解提示中的指令, 导致生成内容有偏差.

4.4 内部有效性威胁

GPT-3.5 和 LLaMA 模型训练过程中均使用了大量开源软件项目数据, 可能存在数据泄露问题, 即大语言模型在预训练时已经遇到过本文所用测试集相关的代码和文档数据, 这是本文工作的内部有效性威胁之一. 但是本文中使用的代码数据主要是 diff 形式的代码变更, 并非预训练数据的主流表示形式, 数据泄露风险相对更小. 此外, 大语言模型方法生成的提交日志在客观指标上并不突出, 说明大语言模型并不是直接输出预训练时的“记忆”.

GPT-3.5 生成文本的不确定性也是研究的内部有效性威胁之一. 为减少这一不确定性, 本文实验中调用模型时会设置 temperature 为 0, 在 20 条提交日志生成数据上进行 3 次重复零样本学习下的生成, 3 次结果完全一致, 表明设置 temperature 为 0 确实减少了 GPT-3.5 生成文本的不确定性.

少样本学习时, 示例检索排序过程的随机性也会威胁研究有效性, 因此本文还提出了基于相似度的示例检索排序方法, 一方面减少随机性, 一方面和随机选择示例的方法对比. 一个可能的内部有效性威胁来源于人工标注和分析时样本选择的随机性, 但相信这种随机性并不会影响本文分析出的有关大语言模型方法的优势与不足的总体结论.

人工标注过程中人的主观性是本文研究的一个潜在有效性威胁, 为应对这一威胁, 本文需要标注的样本都由两人独立标注, 且标注人员事先不清楚样本来源, 对两份标注结果进行检验也发现结果有较高的一致性.

4.5 外部有效性威胁

本文只在 Java 语言数据集上进行实验, 所得研究结论是否适用于其他编程语言仍为未知, 这一点是本文的外部有效性威胁. 但 Java 语言本身是使用最广泛的编程语言之一, 具有一定的代表性. 此外, 本文研究主要基于大语言模型, 而大语言模型训练过程中使用的代码数据涉及了许多种编程语言, 相信在大语言模型熟悉的编程语言上能得到相似的研究结论. 对大语言模型不熟悉的编程语言则需要进行更多研究.

5 相关工作

5.1 提交日志生成

过去的 10 多年中, 提交日志生成工作主要经历了 3 个阶段. 早期工作基于规则和模板[3−6], 如 Buse 等人[3]

分析代码语句及其路径条件的变化并将其转为自然语言作为提交日志; Cortés-Coy 等人[4]识别变更的版型 (stereotype) 信息并提取重要的变更实体, 然后填充到模板形成提交日志. 这类方法生成的提交日志冗长, 还无法反映代码变更背后的关键意图.

之后, 研究人员使用基于检索的方法生成提交日志[7−9], 如 Liu 等人[9]使用向量空间模型检索最相似的代码变更并复用其提交日志; Huang 等人[8]基于变更前后代码的句法和语义检索最相似代码变更. 基于检索的方法生成的提交日志更加自然, 但是其效果完全取决于能否检索到足够相似的代码变更.

后来, 基于深度学习的方法[10−20]成为主流并取得了不错的效果. 最初, 研究人员将代码变更当成文本处理, 使用神经机器翻译模型将代码变更``翻译"成提交日志[10−12]. Liu 等人[13]和 Xu 等人[14]引入了拷贝机制[46]以应对未登录词问题, Xu 等人[14]还同时建模代码变更结构和代码自然语义以更好理解代码变更. Liu 等人[15]使用路径表示[50]从语法树层面理解代码变更. Dong 等人[18]将代码变更表示为细粒度图然后使用图神经网络建模. Shi 等人[17]提出的 RACE 方法将信息检索技术融合到神经网络中, 帮助更好生成提交日志. He 等人[19]提出的 COME 方法同时使用生成和检索的方式生成提交日志, 通过决策模块选择更合适的结果. Tao 等人[20]提出一种知识感知去噪训练方法可以更好地利用包含噪声的数据, 使得训练出的模型能更好地生成提交日志. 上述工作很少考虑已有提交日志数据的质量问题, Tian 等人[2]对 5 个高质量 Java 开源项目的提交日志进行了研究, 发现平均约有 44% 的提交日志需要改进, 而更多开源项目的提交日志质量还不如这 5 个项目. 仅依靠从开源项目收集的提交日志训练生成模型难以生成准确、自然的提交日志, 大语言模型包含的丰富语言规律和世界知识正是应对上述难点的关键, 这也是本文工作的动机.

最近, 部分工作[51,52]同样探索了大语言模型用于提交日志生成的效果. 与本文工作不同, Zhang 等人[51]主要探索大语言模型零样本学习的效果, 研究现有提交日志评价指标的不足; Li 等人[52]则探索如何检索更多的领域知识, 以帮助大语言模型生成更真实更全面的提交日志.

5.2 代码摘要生成

代码摘要生成同样是本文的相关工作. 代码摘要是指代码的一段自然语言描述, 主要表现形式为注释. 类似提交日志生成, 代码摘要生成工作主要可分为 3 类.

基于规则和模板的方法先从代码中提取关键信息, 然后填充到模板形成代码摘要. Sridhara 等人[53−55]通过对代码语法结构信息和标识符语言学信息的深入分析, 从代码中提取动作、主题和参数信息并按照规则组合形成文本. Moreno 等人[56]先识别 Java 类的版型 (stereotype), 然后根据版型选择代码的重要信息, 最后结合模板生成自然语言文本.

基于信息检索的方法会从代码内部检索关键词或从外部信息源检索最相关的文本作为代码摘要. 例如, Haiduc 等人[57]受文本摘要方法的启发, 使用潜在语义检索 (latent semantic indexing, LSI) 技术从代码中提取最重要的 N 个单词作为摘要; McBurney 等人[58]使用主题模型检索多个关键单词并将其层次化组织以便于理解; Wong 等人[59,60]分别从 Stack Overflow 和开源软件项目中收集代码和文本对, 然后使用克隆检测技术检索相似代码获取其对应文本作为注释.

基于深度学习的方法利用神经网络强大的拟合和泛化能力, 从大量匹配的代码-文本数据自动学习二者的映射关系并以此完成代码摘要生成. 例如: Iyer 等人[61]使用带注意力机制的编码器解码器模型为 C#和 SQL 代码生成摘要; Alon 等人[62]建模了代码语法树路径与代码摘要的关系并以此生成代码摘要; Haque 等人[63]在生成代码摘要时加入对代码文件级上下文的建模以提升摘要质量.

随着预训练模型和大语言模型的成功, CodeBERT[64]、CodeT5[65]等代码预训练模型在代码摘要生成任务上取得了很好的效果. Geng 等人[26]利用大语言模型的上下文学习能力应对高质量训练数据稀少的问题, 在多意图代码注释生成问题上表现优异. 本文同样利用了大语言模型的上下文学习能力, 与 Geng 等人工作[26]的不同在于: 1) 本文将大语言模型用于提交日志生成, 其需要对代码变更而不是方法层面的代码进行摘要总结; 2) 本文还研究了微调后的大语言模型在提交日志生成任务的表现.

5.3 其他提交日志相关工作

Li 等人[66]针对提交日志质量问题进行了实证研究, 发现了两个重要结论: 1) 提交日志质量能影响软件缺陷倾向 (proneness); 2) 软件项目中的提交日志质量随时间推移逐渐下降, 而开发人员则认为自身撰写的提交日志质量越来越高.

Irsan 等人[67]和 Zhang 等人[68]使用文档摘要方法从 PR (pull request) 描述、提交日志、相关问题描述中选择重要内容并生成高质量的 PR 标题以帮助代码审查过程.

6 总 结

本文提出基于大语言模型的提交日志生成方法, 使用上下文学习和模型微调方式利用大语言模型优秀的文本生成能力, 通过实验探索大语言模型性能以及示例检索排序技巧能否进一步提升大语言模型性能并对实验结果进行深入分析, 研究大语言模型的优势与不足.

实验结果验证了大语言模型优秀的文本生成能力, 使用 GPT-3.5 进行零样本学习就能超越当前最好的小模型方法, 少样本学习相对零样本学习能更好地发挥大语言模型的能力. 此外, 使用基于相似度的示例检索排序能进一步提高大语言模型效果. 然而, 大语言模型也存在一些不足, 主要体现在: 1) 难以生成已有提交日志中包含的有关特定领域知识的内容, 如代码变更的原因和目的; 2) 对变更的理解不总是正确, 导致生成错误内容; 3) 偶尔出现“幻觉”. 通过思维链或采用更大规模模型可以应对变更理解不正确的情况, 而如何生成包含领域知识的内容和如何应对“幻觉”还值得深入研究.

References

[1] Dyer R, Nguyen HA, Rajan H, Nguyen TN. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In: Proc. of the 35th Int’l Conf. on Software Engineering (ICSE). San Francisco: IEEE, 2013. 422–431. [doi: 10.1109/ICSE.2013.6606588]

[2] Tian YC, Zhang YX, Stol KJ, Jiang L, Liu H. What makes a good commit message? In: Proc. of the 44th Int’l Conf. on Software Engineering. Pittsburgh: ACM, 2022. 2389–2401. [doi: 10.1145/3510003.3510205]

[3] Buse RPL, Weimer WR. Automatically documenting program changes. In: Proc. of the 25th IEEE/ACM Int’l Conf. on Automated Software Engineering. Antwerp: ACM, 2010. 33–42. [doi: 10.1145/1858996.1859005]

[4] Cortés-Coy LF, Linares-Vásquez M, Aponte J, Poshyvanyk D. On automatically generating commit messages via summarization of source code changes. In: Proc. of the 14th IEEE Int’l Working Conf. on Source Code Analysis and Manipulation. Victoria: IEEE, 2014. 275–284. [doi: 10.1109/SCAM.2014.14]

[5] Shen JF, Sun XB, Li B, Yang H, Hu JJ. On automatic summarization of what and why information in source code changes. In: Proc. of the 40th IEEE Annual Computer Software and Applications Conf. (COMPSAC). Atlanta: IEEE, 2016. 103–112. [doi: 10.1109/COMPSAC.2016.162]

[6] Linares-Vásquez M, Cortés-Coy LF, Aponte J, Poshyvanyk D. ChangeScribe: A tool for automatically generating commit messages. In: Proc. of the 37th IEEE/ACM IEEE Int’l Conf. on Software Engineering. Florence: IEEE, 2015. 709–712. [doi: 10.1109/ICSE.2015.229]

[7] Huang Y, Zheng QY, Chen XP, Xiong YF, Liu ZY, Luo XN. Mining version control system for automatically generating commit comment. In: Proc. of the 2017 ACM/IEEE Int’l Symp. on Empirical Software Engineering and Measurement (ESEM). Toronto: IEEE, 2017. 414–423. [doi: 10.1109/ESEM.2017.56]

[8] Huang Y, Jia N, Zhou HJ, Chen XP, Zheng ZB, Tang MD. Learning human-written commit messages to document code changes. Journal of Computer Science and Technology, 2020, 35(6): 1258–1277. [doi: 10.1007/s11390-020-0496-0]

[9] Liu ZX, Xia X, Hassan AE, Lo D, Xing ZC, Wang XY. Neural-machine-translation-based commit message generation: How far are we? In: Proc. of the 33rd ACM/IEEE Int’l Conf. on Automated Software Engineering. Montpellier: ACM, 2018. 373–384. [doi: 10.1145/3238147.3238190]

[10] Jiang SY, Armaly A, McMillan C. Automatically generating commit messages from diffs using neural machine translation. In: Proc. of the 32nd IEEE/ACM Int’l Conf. on Automated Software Engineering (ASE). Urbana: IEEE, 2017. 135–146. [doi: 10.1109/ASE.2017.8115626]

[11] Loyola P, Marrese-Taylor E, Matsuo Y. A neural architecture for generating natural language descriptions from source code changes. In: Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 2: Short Papers). Vancouver: ACL, 2017.

287–292. [doi: 10.18653/v1/P17-2045]

[12] Loyola P, Marrese-Taylor E, Balazs J, Matsuo Y, Satoh F. Content aware source code change description generation. In: Proc. of the 11th Int'l Conf. on Natural Language Generation. Tilburg: ACL, 2018. 119–128. [doi: 10.18653/v1/W18-6513]

[13] Liu Q, Liu ZH, Zhu HM, Fan HF, Du BW, Qian Y. Generating commit messages from diffs using pointer-generator network. In: Proc. of the 16th IEEE/ACM Int'l Conf. on Mining Software Repositories (MSR). Montreal: IEEE, 2019. 299–309. [doi: 10.1109/MSR.2019.00056]

[14] Xu SB, Yao Y, Xu F, Gu TX, Tong HH, Lu J. Commit message generation for source code changes. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. Macao: IJCAI.org, 2019. 3975–3981.

[15] Liu SQ, Gao CY, Chen S, Nie LY, Liu Y. ATOM: Commit message generation based on abstract syntax tree and hybrid ranking. IEEE Trans. on Software Engineering, 2022, 48(5): 1800–1817. [doi: 10.1109/tse.2020.3038681]

[16] Nie LY, Gao CY, Zhong ZC, Lam W, Liu Y, Xu ZL. CoreGen: Contextualized code representation learning for commit message generation. Neurocomputing, 2021, 459: 97–107. [doi: 10.1016/j.neucom.2021.05.039]

[17] Shi ES, Wang YL, Tao W, Du L, Zhang HY, Han S, Zhang DM, Sun HB. RACE: Retrieval-augmented commit message generation. In: Proc. of the 2022 Conf. on Empirical Methods in Natural Language Processing. Abu Dhabi: ACL, 2022. 5520–5530. [doi: 10.18653/v1/2022.emnlp-main.372]

[18] Dong JH, Lou YL, Zhu QH, Sun ZY, Li ZL, Zhang WJ, Hao D. FIRA: Fine-grained graph-based code change representation for automated commit message generation. In: Proc. of the 44th Int'l Conf. on Software Engineering. Pittsburgh: ACM, 2022. 970–981. [doi: 10.1145/3510003.3510069]

[19] He YC, Wang LR, Wang KY, Zhang YP, Zhang H, Li ZJ. COME: Commit message generation with modification embedding. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Seattle: ACM, 2023. 792–803. [doi: 10.1145/3597926.3598096]

[20] Tao W, Zhou YC, Wang YL, Zhang HY, Wang HF, Zhang WQ. KADEL: Knowledge-aware denoising learning for commit message generation. ACM Trans. on Software Engineering and Methodology, 2024, 33(5): 133. [doi: 10.1145/3643675]

[21] 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, 2025.

[22] 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: Curran Associates Inc., 2020. 1877–1901.

[23] Guo Q, Cao JM, Xie XF, Liu SQ, Li XH, Chen BH, Peng X. Exploring the potential of ChatGPT in automated code refinement: An empirical study. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 34. [doi: 10.1145/3597503.3623306]

[24] Sridhara G, Ranjani HG, Mazumdar S. ChatGPT: A study on its utility for ubiquitous software engineering tasks. arXiv:2305.16837, 2023.

[25] Pearce H, Tan B, Ahmad B, Karri R, Dolan-Gavitt B. Examining zero-shot vulnerability repair with large language models. In: Proc. of the 2023 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2023. 2339–2356. [doi: 10.1109/SP46215.2023.10179324]

[26] Geng MY, Wang SW, Dong DZ, Wang HT, Li G, Jin Z, Mao XG, Liao XK. Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning. In: Proc. of the 46th IEEE/ACM Int'l Conf. on Software Engineering. Lisbon: ACM, 2024. 39. [doi: 10.1145/3597503.3608134]

[27] Papineni K, Roukos S, Ward T, Zhu WJ. Bleu: A method for automatic evaluation of machine translation. In: Proc. of the 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia: ACL, 2002. 311–318. [doi: 10.3115/1073083.1073135]

[28] Denkowski M, Lavie A. Meteor universal: Language specific translation evaluation for any target language. In: Proc. of the 9th Workshop on Statistical Machine Translation. Baltimore: ACL, 2014. 376–380. [doi: 10.3115/v1/W14-3348]

[29] Lin CY. ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. Barcelona: ACL, 2004. 74–81.

[30] Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L. Deep contextualized word representations. In: Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long Papers). New Orleans: ACL, 2018. 2227–2237. [doi: 10.18653/v1/N18-1202]

[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: Curran Associates Inc., 2017. 6000–6010.

[32] 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 (Vol. 1:

Long and Short Papers). Minneapolis: ACL, 2019. 4171–4186. [doi: 10.18653/v1/N19-1423]

[33] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding by generative pre-training. 2018. https://www.cs.princeton.edu/courses/archive/spring20/cos598C/lectures/lec4-pretraining.pdf

[34] Lewis M, Liu YH, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. ACL, 2020. 7871–7880. [doi: 10.18653/v1/2020.acl-main.703]

[35] Kaplan J, McCandlish S, Henighan T, Brown TB, Chess B, Child R, Gray S, Radford A, Wu J, Amodei D. Scaling laws for neural language models. arXiv:2001.08361, 2020.

[36] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright CL, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiano P, Leike J, Lowe R. Training language models to follow instructions with human feedback. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 27730–27744.

[37] Touvron H, Lavril T, Izacard G, Martinet X, Lachaux MA, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, Rodriguez A, Joulin A, Grave E, Lample G. LLaMA: Open and efficient foundation language models. arXiv:2302.13971, 2023.

[38] Li XL, Liang P. Prefix-tuning: Optimizing continuous prompts for generation. In: Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int'l Joint Conf. on Natural Language Processing (Vol. 1: Long Papers). ACL, 2021. 4582–4597. [doi: 10.18653/v1/2021.acl-long.353]

[39] Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning. In: Proc. of the 2021 Conf. on Empirical Methods in Natural Language Processing. Punta Cana: ACL, 2021. 3045–3059. [doi: 10.18653/v1/2021.emnlp-main.243]

[40] Houlsby N, Giurgiu A, Jastrzebski S, Morrone B, de Laroussilhe Q, Gesmundo A, Attariyan M, Gelly S. Parameter-efficient transfer learning for NLP. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach, 2019. 2790–2799.

[41] He JX, Zhou CT, Ma XZ, Berg-Kirkpatrick T, Neubig G. Towards a unified view of parameter-efficient transfer learning. arXiv:2110.04366, 2022.

[42] Hu EJ, Shen YL, Wallis P, Allen-Zhu Z, Li YZ, Wang SA, Wang L, Chen WZ. LoRA: Low-rank adaptation of large language models. arXiv:2106.09685, 2021.

[43] Dong QX, Li L, Dai DM, Zheng C, Ma JY, Li R, Xia HM, Xu JJ, Wu ZY, Liu TY, Chang BB, Sun X, Li L, Sui ZF. A survey on in-context learning. arXiv:2301.00234, 2024.

[44] Nashid N, Sintaha M, Mesbah A. Retrieval-based prompt selection for code-related few-shot learning. In: Proc. of the 45th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 2450–2462. [doi: 10.1109/ICSE48619.2023.00205]

[45] Zhao TZ, Wallace E, Feng S, Klein D, Singh S. Calibrate before use: Improving few-shot performance of language models. In: Proc. of the 38th Int'l Conf. on Machine Learning. PMLR, 2021. 12697–12706.

[46] See A, Liu PJ, Manning CD. Get to the point: Summarization with pointer-generator networks. In: Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers). Vancouver: ACL, 2017. 1073–1083. [doi: 10.18653/v1/P17-1099]

[47] Loshchilov I, Hutter F. Decoupled weight decay regularization. In: Proc. of the 7th Int'l Conf. on Learning Representations. New Orleans: OpenReview.net, 2019.

[48] Kendall MG. A new measure of rank correlation. Biometrika, 1938, 30(1–2): 81–93. [doi: 10.2307/2332226]

[49] Wei J, Wang XZ, Schuurmans D, Bosma M, Ichter B, Xia F, Chi EH, Le QV, Zhou D. Chain-of-thought prompting elicits reasoning in large language models. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 24824–24837.

[50] Alon U, Zilberstein M, Levy O, Yahav E. code2vec: Learning distributed representations of code. Proc. of the ACM on Programming Languages, 2019, 3(POPL): 40. [doi: 10.1145/3290353]

[51] Zhang YX, Qiu ZQ, Stol KJ, Zhu WH, Zhu JX, Tian YC, Liu H. Automatic commit message generation: A critical review and directions for future work. IEEE Trans. on Software Engineering, 2024, 50(4): 816–835. [doi: 10.1109/TSE.2024.3364675]

[52] Li JW, Faragó D, Petrov C, Ahmed I. Only diff is not enough: Generating commit messages leveraging reasoning and action of large language model. Proc. of the ACM on Software Engineering, 2024, 1(FSE): 745–766. [doi: 10.1145/3643760]

[53] Sridhara G, Hill E, Muppaneni D, Pollock L, Vijay-Shanker K. Towards automatically generating summary comments for java methods. In: Proc. of the 25th IEEE/ACM Int'l Conf. on Automated Software Engineering. Antwerp: ACM, 2010. 43–52. [doi: 10.1145/1858996.1859006]

[54] Sridhara G, Pollock L, Vijay-Shanker K. Automatically detecting and describing high level actions within methods. In: Proc. of the 33rd Int'l Conf. on Software Engineering. Waikiki: ACM, 2011. 101–110. [doi: 10.1145/1985793.1985808]

[55] Sridhara G, Pollock L, Vijay-Shanker K. Generating parameter comments and integrating with method summaries. In: Proc. of the 19th IEEE Int'l Conf. on Program Comprehension. Kingston, 2011. 71–80. [doi: 10.1109/ICPC.2011.28]

[56] Moreno L, Aponte J, Sridhara G, Marcus A, Pollock L, Vijay-Shanker K. Automatic generation of natural language summaries for Java classes. In: Proc. of the 21st Int'l Conf. on Program Comprehension (ICPC). San Francisco: IEEE, 2013. 23–32. [doi: 10.1109/ICPC.2013.6613830]

[57] Haiduc S, Aponte J, Moreno L, Marcus A. On the use of automated text summarization techniques for summarizing source code. In: Proc. of the 17th Working Conf. on Reverse Engineering. Beverly: IEEE, 2010. 35–44. [doi: 10.1109/WCRE.2010.13]

[58] McBurney PW, Liu C, McMillan C, Weninger T. Improving topic model source code summarization. In: Proc. of the 22nd Int'l Conf. on Program Comprehension. Hyderabad: ACM, 2014. 291–294. [doi: 10.1145/2597008.2597793]

[59] Wong E, Yang JQ, Tan L. AutoComment: Mining question and answer sites for automatic comment generation. In: Proc. of the 28th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Silicon Valley: IEEE, 2013. 562–567. [doi: 10.1109/ASE.2013.6693113]

[60] Wong E, Liu TY, Tan L. Clocom: Mining existing source code for automatic comment generation. In: Proc. of the 22nd IEEE Int'l Conf. on Software Analysis, Evolution, and Reengineering (SANER). Montreal: IEEE, 2015. 380–389. [doi: 10.1109/SANER.2015.7081848]

[61] Iyer S, Konstas I, Cheung A, Zettlemoyer L. Summarizing source code using a neural attention model. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers). Berlin: ACL, 2016. 2073–2083. [doi: 10.18653/v1/P16-1195]

[62] Alon U, Brody S, Levy O, Yahav E. code2seq: Generating sequences from structured representations of code. In: Proc. of the 7th Int'l Conf. on Learning Representations. New Orleans: OpenReview.net, 2019.

[63] Haque S, LeClair A, Wu LF, McMillan C. Improved automatic summarization of subroutines via attention to file context. In: Proc. of the 17th Int'l Conf. on Mining Software Repositories. Seoul: ACM, 2020. 300–310. [doi: 10.1145/3379597.3387449]

[64] Feng ZY, Guo DY, Tang DY, Duan N, Feng XC, Gong M, Shou LJ, Qin B, Liu T, Jiang DX, Zhou M. CodeBERT: A pre-trained model for programming and natural languages. In: Findings of the Association for Computational Linguistics: EMNLP 2020. ACL, 2020. 1536–1547. [doi: 10.18653/v1/2020.findings-emnlp.139]

[65] Wang Y, Wang WS, Joty S, Hoi SCH. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In: Proc. of the 2021 Conf. on Empirical Methods in Natural Language Processing. Punta Cana: ACL, 2021. 8696–8708. [doi: 10.18653/v1/2021.emnlp-main.685]

[66] Li JW, Ahmed I. Commit message matters: Investigating impact and evolution of commit message quality. In: Proc. of the 45th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 806–817. [doi: 10.1109/ICSE48619.2023.00076]

[67] Irsan IC, Zhang T, Thung F, Lo D, Jiang LX. AutoPRTitle: A tool for automatic pull request title generation. In: Proc. of the 2022 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Limassol: IEEE, 2022. 454–458. [doi: 10.1109/ICSME55016.2022.00058]

[68] Zhang T, Irsan IC, Thung F, Han D, Lo D, Jiang LX. Automatic pull request title generation. In: Proc. of the 2022 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME). Limassol: IEEE, 2022. 71–81. [doi: 10.1109/ICSME55016.2022.00015]

作者简介

徐圣斌, 博士, CCF 专业会员, 主要研究领域为智能化软件工程, 代码翻译, 代码生成.

贾林杰, 硕士生, 主要研究领域为智能化软件工程, 代码翻译.

许汤智, 博士生, CCF 学生会员, 主要研究领域为智能化软件工程, 代码测试, 代码修复.

朱晓瑞, 博士, 主要研究领域为软件工程.

余萍, 博士, 副教授, CCF 专业会员, 主要研究领域为智能化软件工程, 云计算, 大数据技术.

徐锋, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为软件缺陷定位, 数据挖掘, 推荐系统, 信任管理.

马晓星, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为智能软件工程, 自适应软件系统, 软件质量保障, 人机物融合软件系统.