Context Conquers Parameters: Outperforming Proprietary LLM in Commit Message Generation

Aaron Imani, Iftekhar Ahmed, Mohammad Moshirpour

University of California, Irvine

背景需求：提交信息的重要性

- **提交信息 (Commit Message, CM)**：记录代码变更的自然语言描述。
- **核心作用**：
  - 软件维护与演进的关键文档。
  - 帮助开发者理解代码历史、追踪问题、进行重构。
- **挑战**：高质量的提交信息对软件质量至关重要，但人工编写耗时且易出错。
- **目标**：自动化的提交信息生成 (CMG)。

传统方法 vs. 大模型时代

早期CMG方法

- 基于代码差异 (diff)、抽象语法树 (AST)、修改嵌入等。
- 依赖于训练数据的质量，易受低质量或机器人生成的提交信息影响。
- 未能完全满足开发者对"好"提交信息的期望。

大语言模型 (LLM) 的兴起

- 强大的推理能力，特别是经过人类反馈强化学习 (RLHF) 训练的模型 (如 GPT-4)。
- 显著提升了CMG的效果。
- Omniscient Message Generator (OMG)：**利用GPT-4，结合六种上下文信息，实现了当时最先进的CMG效果。**

OMG的局限性：隐私与可持续性

依赖专有模型 (GPT-4)

- **隐私风险**：需要将敏感代码（如方法体、类体）发送给第三方API。
  - 案例：三星因员工误将内部代码分享给ChatGPT而禁用所有专有聊天机器人。

环境风险 (碳排放)

- OMG生成一个提交信息需要多次调用LLM (约12次)。
- 大规模使用可能导致巨大的碳排放，超过模型训练本身。
- 例如：1000名开发者一年可能产生超过1200万次调用。

开源大模型（OLLM）的潜力

开源大模型 (Open-Source LLM, OLLM)

优势：

✓ 成本效益高。

✓ 隐私友好：可在本地部署，无需将敏感数据上传。

✓ 更可持续：参数量小，运行在本地GPU上，减少碳排放。

研究空白：尚未有研究探讨OLLM能否在CMG任务上达到与OMG（基于GPT-4）相当的水平。

核心问题：能否用开源模型替代专有模型，实现高质量、安全、可持续的CMG？

研究问题

开源大模型能否生成与顶尖大模型 (GPT-4) 相当的提交信息？

直接用OLLM替换OMG中的GPT-4，效果如何？

如何弥补开源模型在全面性上的不足？

如果OLLM生成的提交信息不够全面，怎么办？

更小的开源大模型 (SLM) 能否生成与顶尖大模型 (GPT-4) 相当的提交信息？

能否用更小、更轻量级的OLLM (SLM) 实现同样的效果？

方法论：基础提交上下文

借鉴OMG：采用OMG提出的六种上下文信息。

关键点：后三项 (4-6) 是由LLM生成的，称为 **LLM-derived commit context**。

1. 关联的Issue/PR
2. 文件相对重要性
3. 软件维护活动类型 (SMA)
4. 修改方法的多意图摘要 (MMS)
5. 修改类的摘要
6. 代码差异 (Diff)

MMS (Multi-Intent Method Summarization)：

- OMG采用的方法摘要技术。
- 从五个方面总结方法：What, Why, How-to-use, How-it-is-done, Property。
- 为每个修改前后的方法分别生成摘要。

实验设置

硬件

NVIDIA A6000 GPU (48GB VRAM)

推理引擎

VLLM (高效内存管理)

温度 (Temperature)

设为 0，保证输出一致性。

模型

OLLM	SLM
量化后的 Llama3 70B (AWQ)	量化后的 Llama3 8B (AWQ)

AWQ (Activation-aware Weight Quantization)：一种量化技术，尽量减少对模型性能的影响。

提示方法 (Prompting)

- RQ1: 初试 ReAct (失败)，后改用 Zero-shot Prompting。
- RQ3: Zero-shot Prompting。

评估指标

自动化指标 (Automated Metrics)

- BLEU, METEOR, ROUGE-L：衡量生成的提交信息与参考提交信息 (OMG生成的)的相似度。
- 作为初步质量保证，确保模型输出与OMG接近。

人工评估指标 (Human Metrics)

Rationality（合理性):

是否解释了变更原因，识别了SMA类型。

Comprehensiveness(全面性):

是否总结了所有变更，包含重要细节。

Rationality（合理性):

是否简洁明了。

Rationality（合理性):

语法是否正确，是否流畅。

通过实践者调查进行评估。

RQ1：开源大模型能否媲美GPT-4？

实验

- 使用 Llama3 70B (AWQ) 替换 OMG 中的 GPT-4。
- 采用 Zero-shot Prompting。
- 对比 OMG 生成的提交信息。

结果

- **自动化指标**：Zero-shot Prompting 显著优于 ReAct (BLEU 提升 118%)。与 OMG 的相似度很高 (BLEU > 30)。
- 人工评估 (Survey 1)：
  - OLLM 生成的提交信息在大部分方面 (理性、简洁性、表达性) 与 OMG 相当。
  - 但在 **全面性 (Comprehensiveness)** 上，OMG 仍被实践者认为更好 (33% vs 25%)。

RQ2：如何弥补全面性差距？

问题：为什么 OLLM 在全面性上不如 OMG？

深入分析：聚焦于 LLM-derived commit context

假设1：文档干扰：代码中的注释 (Comments, Javadocs) 可能误导模型。

假设2：MMS 方法：OMG 为修改前后的代码分别生成 MMS，可能无法捕捉到变更带来的影响。

改进措施：

1. **移除文档：**在生成类/方法摘要前，先去除代码中的注释。
2. **改进MMS (CMMS)：**提出 **Change-based Multi-Intent Method Summarization (CMMS)**。
   - 先生成修改前的 MMS。
   - 分析代码变更 (diff)。
   - 让 OLLM 解释这些变更如何影响修改前的 MMS 的各个方面。

RQ2：CMMS 的有效性

实验

对比不同上下文组合的效果：

- 原始 OMG 上下文
- 移除文档后
- 使用 CMMS 后
- 移除文档 + 使用 CMMS

结果

自动化指标

两项改进均显著提升与 OMG 的相似度 (BLEU 分别提升 34% 和 35%)。

人工评估 (Survey 2)

71% 的参与者认为，使用改进后的上下文 (移除文档 + CMMS) 生成的提交信息 **更全面**。

结论：通过 移除文档 和 采用 CMMS，可以有效弥补 OLLM 在全面性上的差距。

RQ3：更小的模型 (SLM) 能否胜任？

目标：探索使用更小的开源模型 (SLM) 是否能达到与 GPT-4 或大 OLLM 相当的效果。

大 OLLM
Llama3 70B (AWQ)

VS

小模型 (SLM)
Llama3 8B (AWQ)

初始尝试

- 使用改进后的上下文 (移除文档 + CMMS)。
- 采用 Zero-shot Prompting。
- 结果：自动化指标远低于大 OLLM (Llama3 70B)。

问题根源：即使有了好的上下文，小模型似乎也难以准确理解代码 **diff** 中的变更。

RQ3：小模型的困境与 FIDEX

分析

小模型在理解 diff 变更时存在困难。

- 随机抽样 38 个 commit，让 SLM 解释 diff。
- 结果：**82%** 的情况下，SLM 的解释不准确。

需求

需要一种 **diff 增强 (Diff Augmentation)** 技术，帮助 SLM 正确理解 diff。

FIDEX (Fine-grained Interactive Diff EXplainer)

目标：提供清晰、准确、细粒度的 diff 解释。

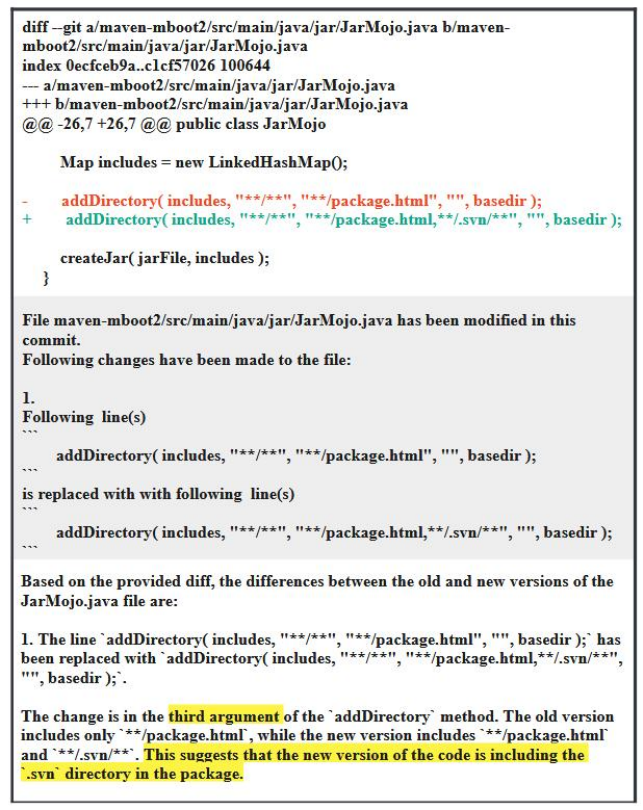
确定性部分

使用 Diff Narrator 脚本，精确解析 diff，生成 Diff Narrative (变更项列表)。

LLM部分

- 明确区分不同语句类型 (Javadoc, 方法调用等)。
- 注意代码顺序变化。
- 区分风格/格式变化与实质内容变化。

FIDEX 示例



```
diff --git a/maven-mboot2/src/main/java/jar/JarMojo.java b/maven-
mboot2/src/main/java/jar/JarMojo.java
index 0ecfceb9a..c1cf57026 100644
--- a/maven-mboot2/src/main/java/jar/JarMojo.java
+++ b/maven-mboot2/src/main/java/jar/JarMojo.java
@@ -26,7 +26,7 @@ public class JarMojo

        Map includes = new LinkedHashMap();

-       addDirectory( includes, "**/**", "**/package.html", "", basedir );
+       addDirectory( includes, "**/**", "**/package.html,**/.svn/**", "", basedir );

        createJar( jarFile, includes );
    }
```

File maven-mboot2/src/main/java/jar/JarMojo.java has been modified in this commit.
Following changes have been made to the file:

1.
Following line(s)

```
    addDirectory( includes, "**/**", "**/package.html", "", basedir );
```

is replaced with with following line(s)

```
    addDirectory( includes, "**/**", "**/package.html,**/.svn/**", "", basedir );
```

Based on the provided diff, the differences between the old and new versions of the JarMojo.java file are:

1. The line `addDirectory( includes, "**/**", "**/package.html", "", basedir );` has been replaced with `addDirectory( includes, "**/**", "**/package.html,**/.svn/**", "", basedir );`.

The change is in the third argument of the `addDirectory` method. The old version includes only `**/package.html`, while the new version includes `**/package.html` and `**/.svn/**`. This suggests that the new version of the code is including the `.svn` directory in the package.

效果说明

- **左上**：原始 Diff (代码变更)。
- **中间**：Diff Narrative (由 Diff Narrator 生成，灰色部分)。
- **右下**：FIDEX 生成的 Diff 解释 (使用 Llama3 8B)。
- **效果**：FIDEX 能够清晰地指出 `addDirectory` 方法的第三个参数从 `""` 变成了 `"**/svn/**"`，即新增了对 `svn` 目录的处理。

RQ3：FIDEX 的效果

实验

使用 FIDEX 增强后的 Diff 作为输入，再次尝试用 SLM (Llama3 8B) 生成提交信息。

自动化指标

FIDEX 显著提升了 SLM 的性能，使其与大 OLLM (Llama3 70B) 的差距大幅缩小 (BLEU 提升 **21%**)。

人工评估 (Survey 3)

22 名参与者比较了 OMEGA (使用 SLM + FIDEX) 和 OMG (使用 GPT-4) 生成的提交信息。

- **总体偏好**：OMEGA 生成的提交信息更受青睐 (**46%** vs 34%)。
- **各项指标**：OMEGA 在 **理性、表达性、全面性** 上均优于 OMG，只有在 **简洁性** 上略逊一筹。

结论：通过 **FIDEX** 增强 Diff 信息，即使是只有 8B 参数的 SLM，也能在 CMG 任务上超越 GPT-4！

OMEGA：最终的 CMG 方法

Context Conquers Parameters (上下文胜过参数量)

更小的模型 (SLM)

4-bit 量化 Llama3 8B (仅需 8GB VRAM)。

优化的上下文

- 移除文档 (Documentation Removal)。
- 改进的方法摘要 (CMMS)。

增强的 Diff 信息 (FIDEX)

提供清晰、准确的代码变更解释。

优势：

- ✓ 高质量 (超越 GPT-4)。
- ✓ 高效 (运行在普通 GPU 上)。
- ✓ 隐私友好 (本地部署)。
- ✓ 可持续 (低能耗)。

主要贡献

1 **可行性验证**

证明了用开源大模型 (OLLM) 替换 GPT-4 生成 CM 是可行的，除了全面性外，其他方面表现相当。

2 **新方法提出**

引入了 **CMMS (Change-based Multi-Intent Method Summarization)**，用于改进基于代码变更的软件工程任务中的方法摘要。

3 **Diff 增强技术**

开发了 **Diff Narrator** 和 **FIDEX**，显著提升了小模型在 CMG 任务中的性能。

结论与启示

核心发现：通过精心设计和优化 **上下文信息**，特别是针对 **diff** 的理解，可以显著提升 **小模型 (SLM)** 在复杂任务 (如 CMG) 中的性能，甚至超越大型专有模型。

实践意义

- 为工业界提供了一种 **低成本、高安全、可持续** 的高质量 CMG 解决方案。
- 降低了对昂贵专有模型的依赖。

未来方向

- 探索更多任务特定的上下文优化策略。
- 进一步研究 SLM 在其他软件工程任务中的潜力。

开源：代码和数据集已公开 [33]。