

增强大模型实现测试用例生成

严东荣, 赵行前, 顾 钧

(中移(苏州)软件技术有限公司, 江苏 苏州 215011)

摘要: 针对基础大模型生成测试用例的可执行性、正确性、与需求对齐性、覆盖率低下的问题, 提出一种基于检索增强生成、提示语工程、大模型微调的改进方法。首先, 将检索增强生成与大模型语言相结合, 使大模型的生成结果有外部数据的补充; 其次, 建立提示词优化系统, 利用动态提示调整、提示语模板化使输出结果更准确; 最后, 进一步微调预训练语言模型, 但仅微调少量的嵌入词, 而非大规模更新模型参数来减少资源投入。实验表明, 所提方法相较于基础大模型在可执行性、正确性、需求对齐性、需求覆盖率上分别提升27.92%、33.83%、45.3%、42%。

关键词: 大模型; 检索增强; 提示语工程; 大模型微调; 测试用例生成

DOI: 10.11907/rjdk.251005

扫描二维码阅读全文: 

中图分类号: TP311.5

文献标识码: A

文章编号: 1672-7800(2025)010-0111-06

Enhancing Large Model for Test Case Generation

YAN Dongrong, ZHAO Xingqian, GU Jun

(China Mobile (Suzhou) Software Technology Co., Ltd., Suzhou 215011, China)

Abstract: Aiming at the problems of low feasibility, correctness, alignment with requirements, and coverage of test cases generated by basic large models, an improvement method based on retrieval enhanced generation, prompt language engineering, and large model fine-tuning is proposed. Firstly, combining retrieval enhanced generation with large model language to supplement the generated results of the large model with external data; Secondly, establish an optimization system for prompt words, utilizing dynamic prompt adjustment and prompt language templating to make the output results more accurate; Finally, further fine tune the pre trained language model, but only adjust a small number of embedded words, rather than updating the model parameters on a large scale to reduce resource investment. The experiment shows that the proposed method improves feasibility, correctness, requirement alignment, and requirement coverage by 27.92%, 33.83%, 45.3%, and 42%, respectively, compared to the basic large model.

Key Words: large model; retrieval-augmented generation; prompt engineering; fine-tuning large models; test case generation

0 引言

大语言模型(Large Language Model, LLM)能执行各种自然语言处理(Natural Language Processing, NLP)任务, 如文本生成、问答和分类。特别在软件工程(Software Engineering, SE)领域, LLM正越来越多地被应用于需求工程、代码生成、漏洞识别、测试生成和软件维护等任务^[1-5]。

检索增强生成(Retrieval-Augmented Generation, RAG)融合了信息检索与文本生成的优势, 引入外部知识库来丰富生成内容, 以匹配应用场景的个性化需求, 增强文本相关性和准确性, 提升专业领域知识问答能力和用户

个性化体验能力。提示语工程(Prompt Engineering, PE)旨在精心设计提示语来优化语言模型的输入, 以显著提升模型对特定任务的理解和输出准确性。微调(Fine-tuning)技术是在预训练语言模型的基础上, 针对特定应用场景进行的定制化的训练过程, 通过调整模型参数以适应新任务。

目前, 应用大语言模型从需求文档中自动化生成手工测试用例的方法十分新颖, 虽然应用场景较少, 但具有极其广泛的应用前景。为此, 本文提出一种创新的大模型手工测试用例自动化生成方法, 融合了检索增强生成技术、提示语工程及模型微调技术, 构建了一个特定的测试领域大模型来生成手工测试用例。同时, 还探讨了检索增强生

收稿日期: 2025-01-06

作者简介: 严东荣(1988-), 男, 硕士, 中移(苏州)软件技术有限公司工程师, 研究方向为软件开发、测试技术和人工智能; 赵行前(1998-), 男, 硕士, 中移(苏州)软件技术有限公司助理工程师, 研究方向为人工智能、自然语言处理、软件开发; 顾钧(1984-), 男, 中移(苏州)软件技术有限公司工程师, 研究方向为软件开发、测试技术和人工智能。本文通讯作者: 严东荣。

成、提示语工程和微调技术在提升大模型生成测试用例方面的实际效益。

1 大模型应用

当前,各类测试用例的编写主要依赖测试人员对需求的理解,结合自身测试知识与经验来设计与构建,因此存在诸多局限性:①高度依赖测试人员对需求的理解深度、测试用例设计能力及测试经验,对测试经验不足的新手而言,设计与编写测试用例颇具难度,易出现用例设计错误;②单纯依靠人工思维,可能会遗漏某些异常情况,导致设计的测试用例无法全面覆盖所有异常场景,进而造成场景缺失;③测试人员从理解需求到设计并编写测试用例,需耗费大量时间成本^[6,7]。为此,引入大模型生成测试用例的方法,以期有效解决上述问题。

1.1 测试用例生成

测试用例生成技术在软件测试领域的重要性日益凸显,显著提升了测试效率和覆盖率^[8]。研究人员已探索了多种方法自动化这一过程,包括随机化、启发式算法、基于搜索的算法、深度学习以及大语言模型^[9-11]。

随机化测试用例生成方法中,Pacheco等^[12]通过随机生成方法调用序列来创建测试用例,利用先前测试中获得的反馈优化测试过程,特别适用于单元测试。通过反馈导向的随机测试生成算法生成一系列包含未测试的调用序列,并基于执行结果创建测试用例。

基于搜索的测试用例生成方法中,Fraser等^[13]使用进化算法生成测试用例,目标是最大化代码覆盖率。首先生成随机测试用例、执行测试并测量覆盖率;其次,选择能实现最高代码覆盖率的测试用例作为下一代的父代;最后,通过不断迭代、优化生成高质量的测试用例。

近年来,深度学习和大语言模型也被应用于测试用例生成。Tufano等^[14]利用序列到序列的转换器模型,通过学习真实世界的测试用例来生成新的测试用例。该方法将测试用例生成任务视为神经机器翻译问题。Alagarsamy等^[15]、Xie等^[16]采用了类似的策略,结合断言知识和测试签名验证来提升生成测试用例的准确性和有效性。

此外,姚邦国等^[17]提出基于模型的测试(Model-Based-Testing, MBT)。该方法是一种通过被测系统的逻辑模型自动生成测试用例的技术,可缩短测试场景梳理及手工测试的耗时。MBT按照自动化程度可分为手工测试、半自动化测试和全自动化测试3个等级,整体流程包括需求、模型、用例、执行、报告和归档6个阶段。通过图形化界面对业务系统进行建模后,模型可在不同版本间复用,有助于提升测试场景的梳理效率。

1.2 基于大模型自动化生成测试用例

LLM在软件测试中应用广泛,不限于单元测试,在集成、系统及验收测试等复杂层级都能通过自然语言处理解

析需求文档,自动生成含输入、输出及步骤的测试用例,加快生成速度、提升覆盖率、减少失误,缩短产品上市周期。

2 大模型生成测试用例

2.1 检索增强生成

检索增强生成将检索系统与大语言模型相结合,基于向量的语义搜索获取相关文本信息并返回给大模型,让大模型基于相关上下文信息生成最终响应^[18]。这一过程可确保生成的内容不仅基于预训练模型所获得的知识,还能通过外部数据进行补充,使生成的响应更健壮和准确^[19]。

2.1.1 索引

本文数据集为20 000条测试用例,数据来源于测试系统中通过SQL查询导出的历史数据并通过细致整理,包含需求描述与对应测试用例。其中,需求描述覆盖多种软件功能特性,例如用户登录、文件上传、数据查询等功能需求;用例名称简洁明了地概括了用例的核心内容,例如基于正确用户名和密码的登录功能测试、超大文件上传功能测试等;用例步骤详细阐述了测试执行的具体操作流程,为后续分析提供了详实的数据基础。

本文针对需求描述与用例步骤等文本内容,采用基于语义及操作步骤的切割方法,借助自然语言处理工具,依据独立的操作步骤或语义单元对长文本进行分割。以“打开浏览器,输入网址,点击搜索按钮,查看搜索结果”这一用例步骤为例,将其切割为“打开浏览器”“输入网址”“点击搜索按钮”“查看搜索结果”4个文本块,保证每个文本块均能独立代表一个可操作的步骤或语义信息。

为了便于后续处理,对用例名称进行标准化处理、统一格式,针对登录、上传等具有特定含义的词汇,赋予其唯一标识或编码,以在向量表示中更清晰地呈现其语义。同时,将数据集按70%、15%、15%的比例划分训练集、验证集与测试集。其中,训练集用于训练大模型微调阶段嵌入词,为检索增强生成模块的索引构建提供数据基础;验证集在模型训练过程中用于评估模型性能表现,以动态调整提示策略、提示语模板等,防止模型过拟合;测试集则联合专家评估法进行最终评估。

此外,预处理后的数据需转化为向量形式并构建索引,例如用例名称可借助Sentence-Transformers生成能捕捉语义信息的向量,并运用Weaviate向量数据库构建索引来实现高效检索^[20]。具体形式如表1所示。

2.1.2 检索

准备好向量数据库后,当大语言模型接收到查询或提问(用户输入的测试需求描述)时,调用外部接口将查询和提问文本传给RAG系统进行向量化处理以生成向量。本文选用余弦相似度算法度量查询向量与索引向量之间的相似性,计算原理基于求解两个向量夹角的余弦值,如式(1)所示。

Table 1 Dataset for indexing and retrieving of test case generation
表 1 生成测试用例索引与检索的数据集

需求描述	用例名称	用例步骤	预期结果
用户在登录页面输入正确的用户名和密码,点击登录按钮后,能够成功登录系统,跳转到用户主页面,且登录过程在 3 秒内完成。	正确用户名和密码登录测试	1、打开系统登录页面。 2、在用户名输入框中输入已注册的用户名。 3、在密码输入框中输入该用户对应的正确密码。 4、点击登录按钮。	页面在 3 秒内成功跳转到用户主页面,且页面显示用户相关信息,如用户名、头像等。
用户在文件上传页面,能够选择本地大小不超过 50MB 的文件,点击上传按钮后,文件能够成功上传到服务器,且上传进度条实时显示上传进度,上传完成后给出成功提示。	正常文件上传测试	1、打开文件上传页面。 2、点击“选择文件”按钮,在本地文件系统中选择一个大小为 20 MB 的文件(例如一个普通的文档文件)。 3、点击上传按钮。 4、观察上传进度条的变化。	上传进度条随着上传过程实时更新,文件成功上传到服务器后,页面给出“文件上传成功”的提示信息。

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

(1)

式中: $\vec{A}$ 、 $\vec{B}$ 分别对应查询向量与索引向量,取值范围为 $[-1, 1]$,当计算所得数值越接近于 1 时,意味着两个向量之间的相似性越高。

针对用户“文件上传”相关测试用例的查询需求,先将查询内容转换为向量表示形式,再借助余弦相似度算法进行运算,从而在索引空间中精确定位与“文件上传”具有较高相关性的测试用例向量。

本文结合需求描述、用例名称和用例步骤 3 个维度的信息进行检索。在用户查询时,不仅根据需求描述的语义进行匹配,还根据用例名称的关键词、用例步骤中的关键操作进行筛选。例如,查询“安全登录测试”,除了在需求描述中匹配“安全登录”相关语义,还在用例名称中查找包含“安全登录”关键词的用例,以及在使用例步骤中检查是否有涉及密码加密、防暴力破解等安全操作的用例。首先,向量数据库对所有匹配的向量依据相似度进行排序,以获得最相似的文本块,并形成相关上下文;其次,将输入的测试需求向量化后与模型库中的历史需求向量库进行相似度比较,以获取最相似的测试用例上下文;最后,通过 API 接口将上下文返回给大模型作进一步处理。

随着测试用例不断更新和完善,定期重新构建索引,可确保检索结果反映了最新的测试用例数据情况,避免因数据过时导致的检索偏差。

2.1.3 内容生成

将检索所得的相关测试用例数据与用户查询进行整合,输入至大语言模型以实现内容生成。

2.1.4 RAG 流程

输入:用户查询 Q, 结构化测试用例数据集 D。

输出:生成结果 R。

1. 对数据集 D 进行预处理

- 1.1 数据清洗,去除重复、修正错误、补充缺失
- 1.2 需求描述和用例步骤进行语义和操作步骤切割
- 1.3 标准化用例名称,编码特定词汇

2. 将预处理后的数据转换为向量 V_D

2.1 用例名称通过语言模型生成向量

2.2 需求描述和用例步骤文本块结合位置等结构化特征生成向量并组合

3. 使用向量数据库构建索引 I

4. 对用户查询 Q 进行向量化,得到向量 V_Q

5. 在索引 I 中检索与 V_Q 相似度高的向量,得到相关测试用例数据 E

5.1 采用余弦相似度计算相似度

5.2 结合多维度信息筛选,避免检索偏差

6. 将 Q 与 E 拼接,得到增强查询 Q'

7. 将 Q' 输入大语言模型,得到生成结果 R

2.2 提示语工程

2.2.1 提示语设计

设计有效的提示语是提示语工程的核心,测试用例生成的核心在于构建有效的提示,为大语言模型提供必要的上下文与指引,否则模型将难以明晰输出要求^[21]。

鉴于软件工程领域中构建提示尚无既定的经验性准则,因此存在一定程度的主观性。为此,本文开展大量实验,尝试了从基础到定制的多种提示组合,以设计出相应的改进方案。基本提示是起点的简单指令,不包含额外指令、上下文或专业格式化,结构为:/*根据需求编写手工测试用例,/*需求描述*//*;改进后提示基于多步骤迭代优化,先通过未使用的数据测试提示避免数据泄露,再手动分析生成性能,并提供一系列新提示来优化生成结果。

表 2 为 LLM 设计了更具特定上下文的提示,强调了相关断言语句的重要性,遵循结构化的测试用例格式以避免重复,添加了更结构化的测试用例上下文细节、提示模板组件及预期目的。背景描述:指示模型专注于创建测试用例,强调需求背景及尽量多生成相关用例。测试用例模版:提供了需要生成测试用例的模版结构。需求主题及需求方向:提供需求的概况性主题及需求方向,例如页面、安全等。实践表明,在基础提示下每个查询生成的测试用例数量平均为 3.5 个;改进后,平均生成的测试用例数量提升至 4.8 个,增长率约为 37.14%,表明提示能引导大语言模型生成更丰富的测试用例。

2.2.2 动态提示调整

动态提示调整(Dynamic Prompt Adjustment, DPA)是

Table 2 Structure of optimized prompts and basic prompts

表2 优化后提示与基础提示的结构

基础提示	改进后提示
	根据以下需求:
	/**需求背景**/**需求主题**/**需求方向
写一个手动测试用例	**/**需求内容**/
/**需求描述**/	生成包含用例名称、用例等级、用例类型、前置条件、用例步骤、预期结果的测试用例,并尽量考虑正向、反向测试用例

指根据特定任务的需求动态调整输入提示的内容和形式,在大语言模型的应用中显著提升模型的响应质量和执行准确性^[22]。在软件测试用例生成领域,DPA可引入特定角色概念来进一步优化。从认知架构引导层面而言,依据认知架构理论,人类处理复杂任务依赖特定的认知框架。为大语言模型赋予“优秀测试人员”角色后,可构建针对软件测试任务的专属认知架构,助力模型聚焦功能逻辑测试、参数逻辑验证等核心领域,优化信息处理流程,降低认知负荷,提升测试用例生成质量与效率。

本文实验增加了一个测试人员的角色,例如“你现在是一个优秀的测试人员,可仔细考虑用户测试功能逻辑需求的方方面面,包括但不限于功能逻辑测试、参数逻辑验证等。你应该满足用户测试需求的所有功能要求和功能逻辑,考虑所有的参数组合条件。接下来,请你根据用户测试需求编写尽可能多的测试用例”。

在完整性方面,实验组生成的测试用例平均包含 5.8 个测试步骤及完整的关键要素,而对照组平均仅包含 4.0 个;在准确性方面,实验组生成的测试用例准确率达到 87%,对照组为 70%;在覆盖率方面,实验组测试用例对软件功能点和参数组合的平均覆盖率为 80%,对照组仅为 60%。综上,“优秀测试人员”角色设定在动态提示调整中,对提升大语言模型生成测试用例的质量和效果具有积极作用。

2.2.3 提示语模板化

本文为不同任务设计标准化的提示语模板,根据具体任务进行微调,以提升生成结果的一致性和可控性。其中,提示语模板包括: #用例名称、#用例类型、#用例步骤、#预期结果。回答要求:①请参考上述测试用例编写案例编写新的测试用例;②你必须仔细理解用户的所有测试需求、测试主题和测试描述,返回的测试用例必须与用户测试需求相关联,并满足用户的所有测试需求且可验证所有参数描述,包括异常值、边界值、准确值等;③你既要给出正向验证测试需要的测试用例,还要给出所有反向验证的测试用例;④用户的每个测试需求都至少对应一个正向和反向验证测试用例,给出的所有测试用例都必须以“用例名称”“用例类型”“用例步骤”和“预期结果”格式返回,测试步骤需清晰、详细,预期结果需准确,用例类型只能在功能、性能、可靠性、安全、兼容性、可运维、用户体验、计费中

选择一种作为用例类型;⑤你不用给出有关性能测试、兼容性测试、用户界面和用户体验测试、数据一致性、用户反馈和安全性测试等测试角度的测试用例,例如多设备或浏览器一致性、数据同步、响应速度测试、保存和取消等,因为如果给出将会造成数据冗余。

2.3 大模型微调

大模型微调通过对预训练语言模型进行训练,以提升其在特定任务或领域上的表现^[23]。本文选择提示语调优(Prompt Tuning, PT),在预训练语言模型输入中引入可学习嵌入向量提示的微调方法^[24]。这些可训练的提示向量在训练过程中动态更新,指导模型输出更适合特定任务的响应,思路是先固定预训练 LLM 的参数,先为每个任务额外添加一个或多个 embedding,再拼接 query 正常输入 LLM,并只训练这些 embedding。

例如,原始输入句子为“前端渲染时间不超过 3 s”。在 PT 中会在输入句子前添加一些额外的嵌入词,例如[性能测试],让输入变成为[性能测试],使前端渲染时间不超过 3 s,这些嵌入词拼接上原来的 query 后仍可训练。实验表明,通过训练这些 embedding,可让模型更好地理解这是一个关于性能测试的问题,从而提升模型在回答测试用例相关问题时的准确性,该方法的优点在于:只需微调少量的嵌入词,无需对整个模型进行大规模的参数更新,因此计算效率和资源消耗相对较低。此外,由于原始模型的参数保持不变,模型的通用性和在其他任务上的性能也不会受到影响。

3 实验结果与分析

3.1 实验环境与评估度量

本文选用通义千问 14b v1.0 作为基础模型,再配备两张 A100 的显卡开展实验,生成的测试用例和测试集通过 Beyond Compare 工具进行比较,并采用专家法进行综合评估统计^[25]。同时,本文根据以下 4 个方面评估 LLM 生成的测试用例质量:

(1)用例可执行性。手工测试用例的可执行性取决于测试人员的实操体验,前置条件、测试步骤的清晰度与合理性关乎用例执行顺序是否合理、有无冲突。为此,测试人员借助 TestRail、JIRA 等工具记录执行状态、生成报告,结合规范与指南来统计成功执行的用例在总用例中的占比情况,以衡量模型的可执行性。

(2)用例正确性。手工测试用例的正确性判定,主要基于人工对预期结果与实际结果一致性,需确保预期结果与业务需求及系统设计高度契合。对于测试集的正确性评估而言,全面覆盖边界条件和异常情况是关键的考量因素。

(3)需求对齐性。评估手工测试用例与需求的对齐性,需将用例与需求文档进行细致比较,以确保其涵盖所

有功能点、条件及约束。在测试集层面,可计算需求覆盖比例并绘制需求跟踪矩阵以直观展现覆盖情况。

(4)需求覆盖率。评估手工测试用例的需求覆盖率,需明确每个用例覆盖的需求点,对覆盖多个需求点的用例给予更高评价,并依据需求重要性进行加权计算。在测试集层面,通过精准测试管理工具进行统计并生成报告,以有效评估需求覆盖率。

3.2 实验结果

图1—图4展示了本文方法和基线模型在需求到测试用例生成任务上的结果。实验表明,本文方法生成的测试用例可执性为55.46%、正确性为68.57%、需求对齐性为80.17%,实现了77.56%需求覆盖率,相较于基础大型语言模型而言进步显著。同时,对模型各模块进行分析:

(1)使用RAG。模型的可执行性、正确性、需求对齐、需求覆盖率分别提升12.61%、15.8%、33.58%、16.93%,证实了RAG学习相关知识的有效性,为大模型完成特定任务方面起到了决定性作用,是通过大模型生成个性化用例的必要步骤。

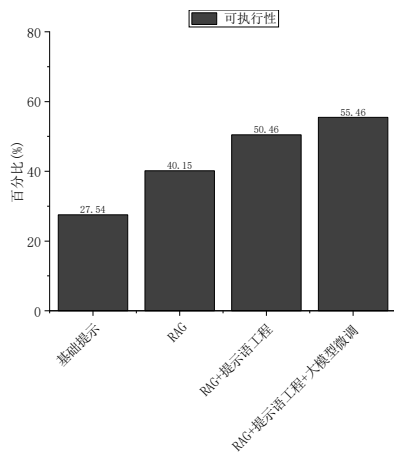


Fig. 1 Executability optimization results

图1 可执行性优化结果

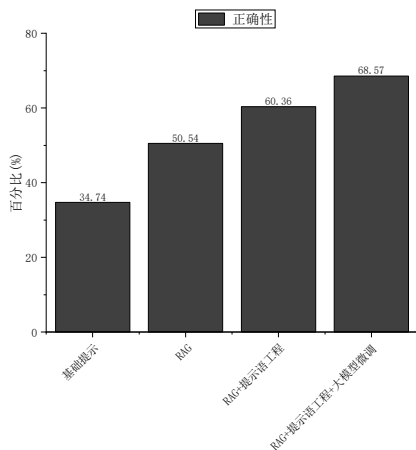


Fig. 2 Correctness optimization results

图2 正确性优化结果

(2)加入提示语工程技术。模型的可执行性、正确性、需求对齐、需求覆盖率分别提升10.31%、9.82%、8.2%、

16.76%,用例可执行性、正确性和需求覆盖率提升显著。其中,可执行性与提示语模板的改进提示有关,会使得用例更规范;需求覆盖率能让模型多考虑并生成正向、反向用例,表明改进提示语工程的设计对生成更规范、覆盖率更广的测试用例作用明显。

(3)加入大模型调优。模型的评估值虽然提升不如RAG、提示语工程,但仍有一定的作用。具体地,可执行性从50.46%提高到55.46%,正确性从60.36%提升到68.57%,需求对齐从76.65%提升到80.17%,需求覆盖率从69.25%提升到77.56%,表明大模型的微调技术只要运用得当,也能提升模型性能。

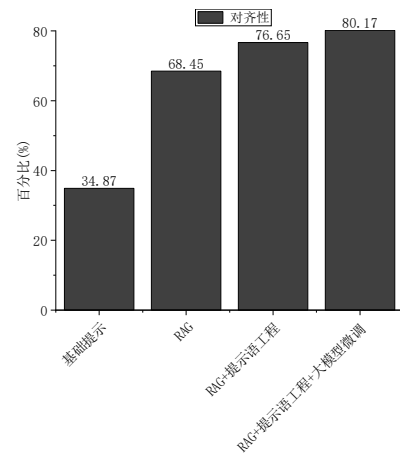


Fig. 3 Alignment optimization results

图3 对齐性优化结果

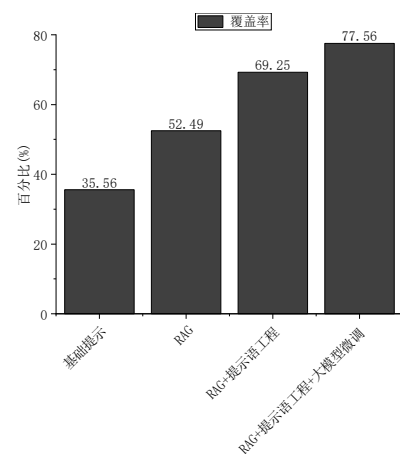


Fig. 4 Coverage optimization results

图4 覆盖率优化结果

4 结语

本文提出一种基于检索增强生成、提示语工程、大模型微调的大模型在软件需求到测试用例生成的方法,相较于基础大模型的性能提升显著,凸显了检索增强生成、提示语工程、大模型微调模块的重要性。实验表明,基础大模型可能无法在特定软件需求的测试用例生成任务中具有理想的表现;检索增强生成使基础大型语言有了强大的

背景知识,相较于微调的基础模型性能提升显著,使得生成测试用例的服务可靠性有了最基础、最有力的保障;微调提示语工程和大模型对生成内容的正确性、覆盖率等方面提升显著。综上,在使用大型语言模型生成测试用例时,应充分考虑结合检索增强生成、提示语工程、大模型微调等技术。

然而,本文研究由于高质量数据较少,会影响系统整体效果,后续将进一步提升数据质量并改良实验结果的评估方法。综上,所提方法为软件测试领域提供了新视角,通过技术创新提升了测试用例生成的效率和质量,对推动软件测试自动化和智能化发展具有重要的理论和实践意义。

参考文献:

- [1] VOGELSANG A, FISCHBACH J. Using large language models for natural language processing tasks in requirements engineering: a systematic guideline[DB/OL]. <https://arxiv.org/abs/2402.13823>.
- [2] ZHANG K, LI J, JIN Z, et al. Codeagent: enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges[DB/OL]. <https://arxiv.org/abs/2401.07339>.
- [3] CUI M J, JIANG Z W, CHEN Y R, et al. A review of the application of large language model technologies in threat intelligence[J]. *Journal of Information Security*, 2024, 9(5): 1-25.
崔孟娇,姜政伟,陈奕任,等. 面向威胁情报的大语言模型技术应用综述[J]. *信息安全学报*, 2024, 9(5): 1-25.
- [4] MOORE T. ChatGPT and software engineering: opportunities and challenges[M]. Boston: Springer, 2023.
- [5] ZHANG Q J, FANG C R, XIE Y, et al. A survey on large language models for software engineering[DB/OL]. <https://arxiv.org/abs/2312.15223>.
- [6] HUA Y L. Research on automatic generation of Web application test cases[D]. Nanjing: Southeast University, 2017.
化云龙. Web应用测试用例自动生成技术研究[D]. 南京:东南大学, 2017.
- [7] ZHANG Q R, HUANG S, SUN L L. A survey of Web function automation testing[J]. *Software Guide*, 2023, 22(3): 227-236.
张清睿,黄松,孙乐乐. Web功能自动化测试综述[J]. *软件导刊*, 2023, 22(3): 227-236.
- [8] LIU C M. The path to becoming a test architect: from test engineer to test architect(2nd edition)[M]. Beijing: Mechanical Industry Press, 2021.
刘琛梅. 测试架构师修炼之道从测试工程师到测试架构师(第2版)[M]. 北京:机械工业出版社, 2021.
- [9] XIAO X, LI S, XIE T, et al. Characteristic studies of loop problems for structural test generation via symbolic execution[C]// Silicon Valley: IEEE/ACM International Conference on Automated Software Engineering, 2013.
- [10] HARMAN M, MCMINN P. A theoretical and empirical study of search-based testing: local, global, and hybrid search[J]. *IEEE Transactions on Software Engineering*, 2010, 36(2): 226-247.
- [11] BOUKHLIF M, KHARMOUM N, HANINE M, et al. Towards an intelligent test case generation framework using LLMs and prompt engineering[C]// Saïdia: International Conference on Smart Medical, IoT & Artificial Intelligence, 2024.
- [12] PACHECO C, ERNST M D. Randoop: feedback-directed random testing for Java[C]// Montreal: 22nd ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications, 2007.
- [13] FRASER G, ARCURI A. Evosuite: automatic test suite generation for object-oriented software[C]// Szeged: 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, 2011.
- [14] TUFANO M, DRAIN D, SVYATKOVSKIY A, et al. Unit test case generation with transformers[DB/OL]. <https://arxiv.org/abs/2009.05617>.
- [15] ALAGARSAMY S, TANTITHAMTHAVORN C, ALETI A. A3test: assertion-augmented automated test case generation[DB/OL]. <https://arxiv.org/abs/2302.10352>.
- [16] CHEN Y, HU Z, ZHI C, et al. ChatUniTest: a framework for LLM-Based test generation[DB/OL]. <https://arxiv.org/abs/2305.04764>.
- [17] YAO B G. Research on the generation method of rest interface test cases based on dependence model[D]. Chengdu: Sichuan University, 2025.
姚邦国. 基于依赖模型的REST接口测试用例生成方法研究[D]. 成都:四川大学, 2025.
- [18] WANG X, YANG Q, QIU Y, et al. KnowledgeGPT: enhancing large language models with retrieval and storage access on knowledge bases[DB/OL]. <https://arxiv.org/abs/2308.11761>.
- [19] KANDPAL N, DENG H, ROBERTS A, et al. Large language models struggle to learn long-tail knowledge[C]// Honolulu: International Conference on Machine Learning, 2023.
- [20] CHANG X W, BO L. Financial causal sentence recognition based on BERT-CNN text classification[J]. *Journal of Supercomputing*, 2022, 78(5): 6503-6527.
- [21] ZHOU Y, MURESANU A I, HAN Z, et al. Large language models are human-level prompt engineers[DB/OL]. <https://arxiv.org/abs/2211.01910>.
- [22] KAMESH R. Think beyond size: dynamic prompting for more effective reasoning[DB/OL]. <https://arxiv.org/abs/2410.08130>.
- [23] ZIEGLER D M, STIENNON N, WU J, et al. Fine-tuning language models from human preferences[DB/OL]. <https://arxiv.org/abs/1909.08593>.
- [24] LESTER B, AL-FOU R, CONSTANT N. The power of scale for parameter-efficient prompt tuning[DB/OL]. <https://arxiv.org/abs/2104.08691>.
- [25] YIN J Y. Application of large language models in higher education: evaluation of the writing-assisting capabilities of large language models[J]. *e-Education Research*, 2024, 45(9): 82-83, 96.
尹珺瑶. 大模型在高等教育中的应用: 大模型写作辅助能力测评[J]. *电化教育研究*, 2024, 45(9): 82-83, 96.

(责任编辑:刘嘉文)