

引用格式:陶英杰,王微微,郭俊霞. 基于测试意图的 Web 测试用例修复[J]. 北京化工大学学报(自然科学版), 2026, 53(4): 105–115.

TAO YingJie, WANG WeiWei, GUO JunXia. A web test case repair approach based on test intent[J]. Journal of Beijing University of Chemical Technology (Natural Science), 2026, 53(4): 105–115.

基于测试意图的 Web 测试用例修复

陶英杰¹ 王微微² 郭俊霞^{1*}

(1. 北京化工大学 信息科学与技术学院, 北京 100029; 2. 北京石油化工学院 信息工程学院, 北京 102617)

摘 要: Web 测试用例自动修复技术在 Web 应用程序测试套件维护过程中扮演着至关重要的角色。由于 Web 应用程序的版本不断演化, 新版本 Web 应用页面元素可能会发生变化, 从而导致测试用例失效。现有技术存在的一个主要问题是缺乏对 Web 测试用例的测试意图的有效表征以及失效原因的分析, 未能实现从一个宏观的角度引导和辅助测试用例修复, 使得由于测试 workflow 变更和传递失效导致的测试用例失效无法被有效修复。通过实证分析, 发现测试意图和导致测试失效的原因对测试用例修复有积极作用, 为此提出一种名为 LetTe (guiding LLM to fix Web UI tests based on test intent) 的意图驱动 Web 测试用例修复方法, 首先解析失效的 Web 测试用例的测试意图和失效原因, 然后引导大语言模型 (LLM) 通过设计提示词和微调来修复它们。LetTe 的修复逻辑是模拟人类专家的思考及修复过程, 根据测试意图和失效原因“思考”可能的修复方案, 然后通过相应的思维链生成修复候选。在开源网站和公开数据集中收集的 7 个 Web 应用程序上对 LetTe 进行了评估, 结果表明, 本文方法修复正确率达到 75%, 优于所有基线方法。

关键词: 测试意图; Web 应用测试; 测试用例修复; 大语言模型

中图分类号: TP311.5 **DOI:** 10.13543/j.bhxbzr.2026.04.012

引 言

如今的 Web 应用程序功能丰富、交互友好, 同时还在不断发展和更新迭代。随着版本的更迭, 其测试用例也需要与时俱进。在 Web 应用程序的更新过程中, 一些变化可能会导致旧版本的 Web 测试用例失效, 而直接丢弃失效测试用例的做法会降低测试套件本身的质量。根据 Pinto 等^[1]的研究统计, 在软件演化过程中, 发生变化的测试用例有 29% 是通过修复获得的, 而手动修复失效测试用例的成本又很高昂。因此, 如何实现对失效 Web 测试用例的自动化修复成为一个研究热点。

Hammoudi 等^[2]指出, 73.62% 的测试中断都是由于定位器失效造成的, 即测试执行过程中无法定

位到对应的页面元素。因此, 在对失效测试用例进行修复时, 最直接的解决方案就是修复定位器, 根据页面元素的属性或层次结构的变化修复测试脚本中对应的表达式。这类解决方案通常通过匹配新版本页面元素来修复测试用例^[3–6], 然而单纯地依靠修复失效定位器很难实现较好的修复效果, 尤其是当测试用例失效是由测试 workflow 改变、传递失效造成时。这些方法由于缺乏对 Web 测试用例的测试意图的有效表征以及对测试用例失效原因的深入分析, 仅通过匹配查找候选元素提供修复线索, 使得其只能关注到局部失效点, 而忽视了全局的修复辅助信息, 从而限制了方法对于更复杂原因如测试 workflow 改变、传递失效等导致的失效测试用例的修复能力。为了解决上述问题, 本文提出一种测试意图驱动的 Web 测试用例修复技术 LetTe (guiding LLM to fix Web UI tests based on test intent), 模拟人类专家遇到测试用例中断时的修复逻辑。具体来说, 通过检测新版本 Web 应用上的变化来分析可能的失效原因, 然后提取和总结旧版

收稿日期: 2025–03–03

第一作者: 男, 1999 年生, 硕士生

* 通信联系人

E-mail: gjxia@mail.buct.edu.cn

本 Web 应用程序的测试行为,从而分析出测试意图,并在这二者的指导下进行测试用例修复。借助大语言模型的能力,本文设计了一套模型微调 and 提示方法的新范式,并设计了相关实验进行实证研究。实验结果表明,本文提出的 LetTe 方法的修复效果优于所有基线方法,与此同时,也证明了测试意图对失效 Web 测试用例而言是一个有效的高层修复指导。

1 Web UI 测试用例修复及 LLMs 在软件测试中的应用

1.1 Web UI 测试用例修复

以往的许多研究都侧重于候选元素的匹配,Choudhary 等^[3]利用 document object model (DOM) 信息进行查找和匹配;Stocco 等^[5]利用视觉信息匹配候选元素;Qi 等^[7]将语义的概念引入 Web 测试用例修复领域,利用文本和图像中包含的语义信息进行匹配;Brisset 等^[8]在网页元素匹配方面进行了进一步探索,提出一种名为 similarity-based flexible tree matching (SFTM) 的方法,该方法将树形匹配视为一个优化问题;Wen 等^[9]在修复失效的 Web 测试用例时考虑了 DOM 和 UI 中的信息,并通过分类进一步优化了对这些信息的利用以提高修复效果。但是这些研究均未考虑测试用例的测试意图,也很少考虑测试用例的失效原因,只从低维的角度去查找匹配项,这为本文提供了一个绝佳的研究思路。

1.2 LLMs 在软件测试中的应用

目前在软件测试领域,从自动化测试、错误重现到单元测试领域的测试用例修复,大语言模型 (LLMs) 已得到广泛应用^[10-11]。这也在研究之初给予了本文启发,即是否可以不依赖任何传统的 Web 测试用例修复方法,仅依靠大语言模型来完成更好的测试用例修复。同时在这个过程中,模拟人类专家的修复过程,合理划分修复工作流程。由于 LetTe 利用了 LLM 的功能,本文在设计提示词时参考了 OpenAI 的“如何获得更好结果的六大策略”等官方文件^[12]。表 1 总结了在本文方法中使用的提示技巧,并用序号对它们进行了编号,以方便在下文解释每个子任务的技术细节。此类提示技巧能系统化提升提示词质量,通过清晰指令、参考文本、任务拆解、思考时间等策略,显著增强模型输出的准确性与相关性。

表 1 提示技巧

Table 1 Prompting techniques

序号	OpenAI 官方文档中的技巧
R1	要求模型采用特定的角色或风格
R2	明确指定完成任务所需的步骤
R3	指导模型在急于得出结论之前先自行找出解决方案
R4	使用分隔符明确指出输入的不同部分
R5	使用内部独白或一系列查询来隐藏模型的推理过程

2 LetTe 方法

2.1 总体框架

本文提出了一种测试意图驱动的 Web 应用程序测试用例修复方法 LetTe。图 1 简要介绍了本方法,主要包括两个阶段:1) 失效测试用例分析,通过对比新旧版本页面的主要元素,分析原测试用例的测试意图,以及新版本出现测试中断的原因,为下一阶段的修复工作提供指导;2) 失效测试用例修复,通过构建包含测试意图和失效原因指导下的修复逻辑的思维链推理,生成失效测试用例的候选修复。考虑到不可修复等特殊情况,LetTe 在其生成的候选修复不正确时,进一步判断是否可以重现测试用例的意图,如果通过判断认为仍然可以修复,那么将再次尝试,否则认为该页面在应用更新过程中发生了重大变化,已经无法修复。

2.2 失效测试用例分析

在人工修复测试用例情况下,人类专家首先通过查看测试用例名称、观察原测试用例的执行情况、比较新旧版本的页面间差异,进而分析测试意图和失效原因。因此,为了尽可能模拟这一系列思考分析过程,失效测试用例分析阶段由 4 个模块组成:主要元素提取,元素检查,测试意图分析和失效原因分析。本文方法从失效测试用例和报错信息中提取必要信息,分析测试意图和可能的失效原因,为后续的 Web 测试用例修复提供参考。

借助表 1 中的提示技巧,本文为失效测试用例分析阶段的每个模块制定了如下提示规则(具体的提示词示例如图 2 所示):

主要元素提取提示词 技巧【R1, R2】+ 输入【HTML 源代码】;

元素检查提示词 技巧【R1, R3】+ 输入【HTML 源代码 + 主要元素及其 Xpath】;

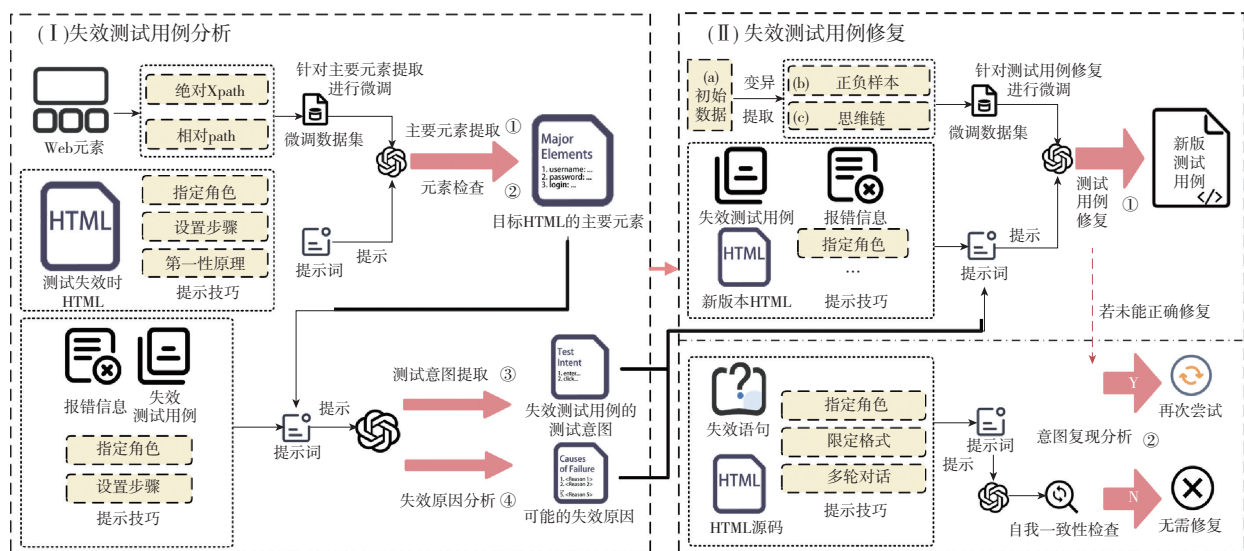


图1 基于测试意图的Web测试用例修复方法框架

Fig. 1 Framework for test intent-based web test repair

测试意图提取提示词技巧【R1】+ 输入【文件名 + 原始测试用例 + 已检查的主要元素和Xpath】;

失效原因分析提示词技巧【R1,R2】+ 输入【旧版本主要元素 + 新版本主要元素 + 报错信息】。

2.2.1 主要元素提取及检查

在主要元素提取模块中,LetTe从简化的HTML源代码中提取主要的可交互元素(如按钮、输入框)以及这些元素对应的Xpath表达式。本文将主要元素定义为导航栏元素除外的位于Web应用程序界面主要交互区域的表单元素或表格元素。为了自动提取页面的主要元素,本文根据提示规则构建了相应的提示模板,并在该模块中对LLM进行微调。

自然语言大模型通常是面向通用任务来设计和训练的,其答案具有随机性,即在相同情况下,同一任务可能会得到不一致的回答。此外,Web应用程序有多种交互界面风格,这也导致了在页面主要元素提取类的任务时,回复格式可能千差万别,仅通过零样本提示或少样本提示皆是如此。因此,在提取主要元素及其Xpath表达式时,本文采用微调技术来解决这一问题。具体来说,从6个不同的Web应用程序的60个页面中提取120条由主要元素及其对应Xpath组成的数据作为本模块的微调数据,每个Web应用程序中各分别选择10个页面。关于这6个Web应用程序的具体信息,其中5个与下文第3节实验设计过程中收集的5个开源Web应用程序一致,还有一个是名为PrestaShop的Web应

用程序,这些应用程序囊括了后台管理、消息展示、电子商务等不同类型的页面风格。通过构建具有大量页面风格差异的微调数据,可以提高本文方法的鲁棒性。与此同时,需要注意的是在从5个(也被用于后续实验的)Web应用程序中提取微调数据时,刻意避开了在新版本中属性或结构发生变化的页面,以保障不对最终的实验结果造成影响。考虑到Xpath表达式的灵活性,LetTe同时使用了相对Xpath以及绝对Xpath来作为微调数据,示例如表2所示。微调数据设计并收集完成后,将数据上传到OpenAI官方平台进行微调。在本模块的模型微调中,依托于OpenAI平台采用监督微调(SFT)的微调技术。在具体实现中,超参数设定为3个训练轮次(epochs)与单样本批量(batch size为1)。该配置针对小规模数据集特性,通过多次遍历数据增强模型对任务模式的捕获能力,同时以逐样本梯度更新提升训练稳定性,避免大批次随机性对稀疏标签场景的干扰。此外,对于微调数据量大小的选择,依据OpenAI平台官方文档中“微调示例计数建议”,同时为了平衡微调数据集的构造成本和微调效果,选定了120条的数据量大小。

在元素检查模块中,为了使主要元素的提取更加准确,本方法还会检查主要元素提取模块生成的答案。利用第一性原理,该模块会检查之前答案的准确性,并在必要时进行修正。由于该模块需要首先自行生成答案(重新提取页面主要元素和Xpath),然后再作对比检查,因此,此处使用的微调

System Prompt: You are a web testing expert. Given an html, first analyze which are the main elements, then list the main elements and their xpath expressions (absolute xpath or relative xpath). Note: The xpath expression must follow the xpath syntax and be able to find the corresponding element.

User Prompt: Given an html: *<HTML Source Code>* Please list all major interactable elements and their corresponding correct xpaths.



用户



LLM

Assistant: *<List of Major Elements and Xpath>*

主要元素提取

System Prompt: You are a validator of the web's major interactable elements and their xpaths. Given an html, the possible main elements, and the xpath corresponding to the elements, you need to check if you can find the corresponding elements in the html according to the syntax rules of the xpath and whether there are any major elements that are not listed. You need to firstly re-extract the main elements and their corresponding xpaths, and then compare your own answer with the provided one. Consider all the factors together and double-check if xpath finds the corresponding element accurately and then get the correct answer.

User Prompt: Given an html: *<HTML Source Code>* The possible main elements and the xpath that corresponds to the elements: *<List of Major Elements and Xpath>* Please check whether these main elements and their corresponding xpaths are correct in this html page. Please leave them unchanged if they are correct, or correct them if they are not. List all major elements and their corresponding correct xpath expressions. No further explanations are required.



用户



LLM

Assistant: *<List of Checked Major Elements and Xpath>*

元素检查

System Prompt: You are a web application testing expert. Given a test case and its filename and the main elements of the page, try to understand what the test case is testing and give its test intent.

User Prompt: The test case filename is *<Filename>* Test case: *<Original Test Case>* The main elements of the currently tested page: *<List of Checked Major Elements and Xpath>* Please give the test case operation intent.



用户



LLM

Assistant: Test Intent: *<The Test Steps>* + *<Overall Intent>*

测试意图分析

System Prompt: You are a web application testing expert. When the web application was updated, the test cases failed. Given the sets of major elements of the old and new versions and the error reporting information, analyze the reasons for the failure of the test case.

User Prompt: The major elements of the old version: *<Major Elements List of Old Version>* The major elements of the new version: *<Major Elements List of New Version>* The error message: *<errorMessage>* Please compare the main elements of the two versions to find out what changed and analyze the root cause and other reasons for test case failure. *<Note: The first element removed is most likely the root cause.>*



用户



LLM

Assistant: *<The Root Cause>* + *<Other Possible Causes>*

× 5

失效原因分析

图2 失效测试用例分析阶段提示词示例

Fig. 2 Examples of prompts for the broken test analysis phase

表2 微调数据抽取示例

Table 2 Example of extracted fine-tuning data

表达式类型	表达式内容
相对 Xpath	1. "Username" xpath="//*[@id="login name"]"
	2. "Password" xpath="//*[@id="usr password"]"
	3. "Sign in" xpath="//*[@id="next"]"
绝对 Xpath	1. "Username" xpath="/html/body/div/form/div[1]/input"
	2. "Password" xpath="/html/body/div/form/div[2]/input"
	3. "Sign in" xpath="/html/body/div/form/input"

模型与主要元素提取模块中的微调模型保持一致。

2.2.2 测试意图提取

测试意图是设计和执行测试用例的核心指导

原则。当人类专家试图修复测试用例时,他们会严格遵循测试用例的原始意图来进行修复。研究表明,对意图的利用对于单元测试的修复、移动应用自动测试和代码注释生成等任务具有积极影响^[13-16]。此外,本文认为,一个正确的测试用例修复应该是被修复测试用例的意图与原始测试用例的意图保持一致,通过利用测试意图,可以明确修复目标,并据此指导修复策略。测试意图可以通过总体测试目标和具体测试路径来表示。

在本模块中,LetTe 分析失效测试用例的测试意图,其中测试用例的文件名、原始测试用例和旧版本中的主要元素信息作为本模块的输入信息。测试用例一般使用驼峰命名法命名,用近似自然语言

的方式简要说明测试用例需要测试的总体内容或目的,而原始测试用例中每条被执行的测试代码则表示了测试用例的具体测试路径,旧版本中的页面主要元素信息则是进一步映射了测试用例执行的每一步具体的作用。

2.2.3 失效原因分析

人类专家在面对测试用例失效时,一般首先会查看并分析是什么错误导致了测试失效,因此本文构建了这样一个模块来模拟该原因分析过程。为了在测试用例修复过程中准确定位问题所在,利用LetTe在该模块中充分分析Web测试用例失效的各种潜在原因。该模块的输入是新旧版本页面中提取的主要元素及其相应的Xpath,以及测试用例失效中断时报告的错误信息。本文将该模块所分析出的失效原因规范为3类情况,即测试用例失效的直接原因、根本原因和其他可能的可能原因。

这一过程具体来说,是首先对新旧版本中主要元素及其Xpath之间的差异进行对比和分析;然后,本模块进一步将差异对比和分析的结果用于对测试用例失效原因的具体分析。

2.3 失效测试用例修复

Web应用程序更新时,开发人员可能会删除某个特性或将原始版本中的某些功能集成到其他功能中,因此基于或涵盖这些功能特性的测试用例就会在运行时失效中断,而这类失效测试用例是无法修复或无需修复的。出于对此类情况的考虑,本文在第二阶段设计了两个模块:测试用例修复模块和意图复现分析模块。

当LetTe遇到失效的测试用例时,会开始修复过程。在正常情况下,LetTe会使用测试用例修复模块进行修复,但是当生成的修复候选中不存在正确的修复时,它就会利用意图复现分析模块来分析该测试用例的意图在新版本中能否被再次复现,即是否还有可能被正确修复。根据判断结果,LetTe会提供下一步的修复建议。如果判断结果为“无需修复”,则表明页面差异较大,说明在新版本中的上一步操作可能与原始测试意图不一致。这种不一致可能导致迁移到错误的页面或根本无法进一步交互,使其无法修复。本阶段的提示规则如下(具体的提示词示例如图3所示)。

测试用例修复提示词:技巧【R1】+ 输入【原始

测试用例+测试意图+报错信息+失效原因+新版本HTML源代码】。

意图复现分析提示词:技巧【R1,R4,R5】+ 输入【文件名+失效语句+失效语句意图+旧版本HTML源代码+新版本HTML源代码】。

2.3.1 测试用例修复

通过第一阶段分析出测试意图和失效原因后,本模块接收这些修复指导作为输入。此外,本模块还使用新版HTML源码、报错信息和原始测试用例作为输入。在这项工作中,本文在LetTe的设计中融入了思想树(tree-of-thought, ToT)推理^[17]的理念,它允许LetTe针对每种不同的可能失效原因生成不同的修复方案。具体地,LetTe针对每一个失效测试用例生成5种可能的失效原因后,依据每种原因进行修复尝试,最后输出5个候选修复。在每一个修复尝试的过程中,LetTe会根据测试意图和失效原因来指导修复,同时利用新版本中页面信息所提供的线索来修复失效测试用例。

由于目前市场主流的LLM都具有最大Token数量的限制,能作为模型输入而提供的案例样本就非常有限,不可能涵盖所有的Web测试用例失效类型,因此Web测试用例修复任务并不适合应用少样本学习的提示方法。本文对该模块中使用LLM采用微调技术,从而适应Web测试用例修复任务。受深度学习训练数据中经常采用的正负样本的启发,本文对该模块的微调数据做进一步的修改,同时为测试用例修复的每一个样本设计了思维链推理过程,如图1中的(a)~(c)所示,具体的微调数据构建步骤如下。

1)构建初始微调数据。本模块利用与主要元素提取模块中相同的6个Web应用程序构建了30个初始微调数据。对每个Web应用程序中的新旧版本提取同一个页面信息,并在此基础上手动构建了5个失效测试用例和HTML源代码,然后利用测试意图分析模块和失效原因分析模块获取相应的测试意图和可能的失效原因。

2)构建正负样本微调数据。在构建完成初始微调数据后,针对失效原因对这30个微调数据进行人工检查和变异,对原始的可能失效原因进行语言风格保留的语义篡改生成负样本(即如果原始失效原因分析经检查是正确的原因分析,则添加语言风格相似但原因分析不正确的变体,反之,则添加语

System Prompt: You are a web test case fixer now. Given the test intent, a failing test case, error reporting information, possible causes of failure, and new version element information, you are asked to update and fix the test case without changing the test intent(Tested elements still need to be tested, untested elements should not be tested in the new version, and deleted elements should not be tested.). Test input format requirement means in send_keys, based on the input box type, such as email, birthday, give input data that satisfies the format, for example, that the input in the email input box should be in email format, the input in the birthday input box should be in date format, etc. <Note: Before outputting the new test case, make sure that the xpath in the test case that also appears in the error message is not consistent with the original (consistent means that the fix was not successful).>

User Prompt: The web application was updated from version A to version B, and the test cases failed. Broken test case: <Original Test Case> The possible test intent of this test case: <Test Intent> The error message: <ErrorMessage> The possible cause of failure: <Possible Cause> new version html: <New Html Source Code> Please update and fix the test cases according to the information provided and finally return the entire updated test case. <Note: Consider whether the test input after the test fix meets potential format requirements.>



用户



LLM

Assistant: <Candidate Repairs>

× 5

测试用例修复

System Prompt: You are a web testing expert. A test case fails after a web application update, and now you need to analyze whether there is a possibility of fixing it (add steps, remove steps, change xpath, change the operating logic, etc.). Based on the test case name, you can analyze the overall test intent and if it will ultimately meet the overall intent, it can be fixed.

User Prompt: Given the test name, the broken statement and old-version's html, you will find out what the broken statement is doing, then describe the operation in as much detail as possible <what's the purpose of the statement>. The test name: <Filename> The broken statement: <Broken Statement> The old version HTML source code: <Old Html Source Code>.



用户



LLM

Assistant: <Intent of the Broken Statement>

*** Second round of dialogues ***

User Prompt: Given the test name, a detailed description of the broken statement, and the new version HTML, determine if the test case can still be fixed correctly for the new version. The test name: <Filename> The description of the broken statement: <Intent of the Broken Statement> the new version HTML source code: <New Html Source Code> Please answer with the format: <yes/no>.



用户



LLM

Assistant: <Yes> / <No>

意图复现分析

图3 失效测试用例修复阶段提示词示例

Fig. 3 Examples of prompts for the broken test repair phase

言风格相似但原因分析正确的变体),这相当于将微调数据的样本数量增加了一倍,负样本模拟模型推理中的常见错误类型(如过度泛化、因果误判),通过对抗训练增强决策边界清晰度。

3)构建思维链(CoT)推理数据。本模块的微调数据中既有正确的失效原因分析,也有错误的失效原因分析。当人类专家面对两种截然不同的失效原因分析时,首先会根据逻辑判断原因分析是否正确,并根据所提供的信息找出测试用例失效的根本原因,然后构思出针对该根本原因和所有其他原因的修复方案,如增加一个新步骤、删除一个步骤、修改定位器中的一个属性等,最终给出修复后的正确测试用例。为了使LLM通过微调技术能够更好地学习到人类专家在修复过程中的精髓,进而模拟人类专家的修复逻辑,本模块还特意为每一组微调数据设计了相应的思维链推理过程,显式建模人类专家的多步推理逻辑,其中包括上述具体的修复思维过程和建议的修复方案,然后将思维链推理与正确

修复的测试用例结合起来,形成最终的解决方案,即模型输出。

在进行微调时,本文仍然使用OpenAI平台进行微调。需要说明的是,本工作中的微调并非是基于主要元素提取模块中的微调模型进行的,而是基于原始的模型。此外,微调数据量大小的选择依据以及微调过程中的参数设置与失效测试用例分析阶段相同。

2.3.2 意图复现分析

当生成的修复候选均为不正确的修复时,LetTe将使用意图复现分析模块来判断是否还有可能重现目标失效测试用例的测试意图。由于测试用例修复模块每次生成的修复候选有多个,而多个修复候选的无效使得LetTe需要先行考虑测试用例是否还能被正确修复,而不是一味地消耗资源进行尝试。当测试用例失效时,人类专家通常会根据错误报告的信息,参考网页中的定位器来查找相应的元素是否仍然存在,或者是否存在类似的元素,这其

实也是人工检验失效测试用例是否还能被修复的过程。所以为了模拟这一过程,LetTe提取测试用例的文件名,并根据报错信息提取中断语句,然后结合旧版本HTML源码分析出中断语句本身的语句级意图。最终,根据文件名、中断语句的意图和新版本HTML源码,LetTe分析并确定该失效测试用例是否仍有被正确修复的可能。

受Wang等^[18]的研究启发,为了得到值得信赖的答案,本文方法还在该模块中添加了自我一致性检查,即对同一问题进行多次提问来获得最统一的答案(本文中采用连续提问5次)。LetTe从所有生成的二进制结果(yes/no)中选出答案最一致的一个作为最终答案,也即少数服从多数。

3 实验验证

3.1 实验设置

3.1.1 实验环境及数据集

本文在LetTe的实施中采用GPT-3.5-Turbo(gpt-3.5-turbo-0125),其也是LetTe方法中作为微调模型的基础模型。在LetTe的第一阶段的所有模块和第二阶段的测试用例修复模块中,本文将温度参数设置为0.8,与之前的研究工作^[19]保持一致。

本文采用所有过往研究数据集^[4]中的开源Web应用程序以及相应的测试用例(以Java编写),并为这些Web应用程序的特定版本配置了相应的运行

环境。Web应用程序的详细信息如表3所示,其中, ΔV 表示Web应用程序新旧版本之间的版本代差,即新版本和旧版本之间存在几个中间版本。

表3 过往研究数据集中的应用程序

Table 3 Applications from previous research dataset

应用程序	ΔV	旧版本号	新版本号
AddressBook	30	4.0	6.1
Claroline	7	1.10.7	1.11.5
Collabtive	4	0.65	1.0
MantisBT	1	1.1.8	1.2.0
MRBS	12	1.2.6.1	1.4.9

鉴于该数据集中使用的Web应用程序大多数版本均为10年前提交的更新,本文重新选择了部分Web应用程序,并更新了其余应用程序的版本。在Source Forge和GitHub网站上重新收集了5个开源Web应用程序,确保它们在过去两年中保持了更新记录,同时保证每个Web应用程序都有两个可运行版本且中间至少间隔两个迭代版本。使用Python为这些Web应用程序编写测试用例,最终为这5个Web应用程序创建了共336个测试用例。本文手动统计了每个Web应用程序的功能数量,并要求测试用例能够覆盖每个应用程序60%以上的功能。经过对所有的测试用例进行测试,共获得107个失效测试用例,统计结果如表4所示。

表4 自行收集数据集数据统计

Table 4 Statistics of self-collected dataset

应用程序	ΔV	旧版本号	新版本号	测试用例总数	失效个数
Admidio	12	4.2.0	4.2.13	63	5
Collabtive	2	3.0.0	3.1.0	43	5
MantisBT	11	2.24.3	2.26.0	40	23
MRBS	2	1.11.1	1.11.4	30	5
PHP-Fusion	5	9.03.90	9.10.30	160	69

3.1.2 指标与基线

将修复正确率(correct repair rate)作为评价指标。修复正确率用于评估对失效测试用例的正确修复的能力,本文将能通过测试运行并且修复后符合原始失效测试用例的测试意图的修复视为一个正确修复。

以前人研究的实验结果^[4]作为参考,从对比实

验中选择所有参与对比的过往方法作为本文的基线,用于同LetTe进行比较,包括WATER(Web Application TEst Repair)^[3]、VISTA(Visual Web Test Repair)^[5]、WEBEVO^[20]、SFTM^[8]和UITESTFIX^[4]。

本文还以LetTe方法为基础,修改了测试用例修复模块的输入部分,即更改了大语言模型的提示词。具体来说,此处将原来提示词中针对测试意图

的描述内容清空,即将“测试意图:[内容]”改为“测试意图:Null”。为了方便后续描述,将该变体命名为 LetTe_{ni}。

3.2 实验结果

3.2.1 不同方法对比

为了探究与现有方法相比 LetTe 在开源数据集上的有效性,本文在开源数据集^[4]上进行了实验,实验结果如表 5 所示。修复正确率的比较结果显示,本文提出方法 LetTe 优于所有基线方法,可以达到 75% 的平均修复率。

此外,为了深入理解 LetTe 优于其他基线方法的原因,通过研究修复过程中的日志文件,对一些仅由 LetTe 修复的独特的失效测试和断言进行总结,得出以下发现。

1) LetTe 在查找候选修复时的搜索空间更为集中; LetTe 是先通过比较差异来确定基础的搜索空间,然后利用 LLM 在自然语言处理方面的优势搜索语义和意图层面的匹配结果,从而进一步缩小搜索空间,正如一些日志文件中所示,LLM 在一轮修复中提供的有效修复结果甚至出现了大量与有效修复内容相似或重复的内容。

2) LetTe 可以生成具有更多样化的定位表达式

的修复候选; Selenium 可使用多种类型的定位器来查找页面,且每种类型的定位器都可以有不同的表达式, LetTe 在修复过程中充分利用了这一点。基于 LLM 已有的预训练知识, LetTe 提供了不止一种定位器表达式来定位同一元素,这样做的目的是为了提提高正确修复的概率,尤其是对于那些无法定位或无法使用单一表达式进行交互的元素。

3.2.2 消融实验

为了证明本文提出方法中测试意图对失效 Web 测试用例修复的高层指导作用,针对测试意图进行消融实验,实验结果见表 6。可以发现,在清空输入给模型的测试意图之后, LetTe 的平均修复正确率为 85.05%, 而 LetTe_{ni} 的平均修复正确率为 72.90%, 总体下降了 12.15%。通过分析可知, LetTe_{ni} 正是因为缺乏测试意图作为高层指导而导致修复正确率大幅下降。通过检查日志文件,可以发现在相当多的失效测试用例修复过程中,其主要元素提取模块已经能够准确地从新版本的页面中提取出正确的元素及其 Xpath 表达式,但在测试用例修复过程中却未能被有效地利用。因为 LLM 无法判断该信息是否是一条有用的修复线索,从而导致最终修复失败。因此,本文总结出以下发现。

表 5 LetTe 与其他基线方法的修复正确率比较

Table 5 Comparison of correct repair rates for LetTe with other baseline methods

应用程序	失效测试用例个数	正确修复的失效测试用例个数					
		WATER	VISTA	WEBEVO	SFTM	UITESTFIX	LetTe
AddressBook	8	1	2	0	2	2	6
Claroline	29	0	24	0	0	24	26
Collabtive	5	3	1	2	0	3	2
MantisBT	38	18	4	19	6	24	30
MRBS	25	0	4	0	2	4	15
总数	105	22(21%)	35(33%)	21(20%)	10(10%)	57(54%)	79(75%)

加粗数字为不同修复方法在该应用程序失效测试用例中最多的修复个数;括号内数字代表不同修复方法针对所有失效测试用例的修复正确率。

表 6 针对测试意图消融实验的修复正确率比较

Table 6 Comparison of correct repair rates for test intent ablation experiment

方法	修复正确率/%					平均正确率/%
	Admidio	MantisBT	PHP-Fusion	Collabtive	MRBS	
LetTe	100	82.61	82.61	100	100	85.05
LetTe _{ni}	80	69.57	71.01	80	100	72.90

(1)测试意图是Web测试用例修复的一个有效的高层修复指导,它有助于有效利用页面信息,并指导修复过程中逻辑思维的构建;去掉测试意图导致的修复正确率大幅下降足以证明这一点,并且LetTe在失效测试用例各个修复阶段中的模型推理更具有逻辑合理性。

(2)测试意图这类高层修复指导有助于实现对测试 workflow 改变或传递错误导致的测试用例失效的修复;LetTe在其实践中均对这两种情况实现了正确修复。

3.2.3 不同大模型的影响

为了证明本文提出的LetTe方法 workflow 设计的优越性和合理性,引入DeepSeek R1(deepseek-r1-671b)大语言模型,通过采用不同的大模型来观察其

对修复效果的影响。由于DeepSeek R1的671b版本在目前实验室环境中暂时不存在满足微调的硬件要求,故设计了一个LetTe方法变体,其中的模型不经过微调但采用小样本提示的方式进行修复,将其命名为LetTe_{nfs},将直接与DeepSeek R1问答的修复方式命名为DSR1。由于LetTe_{nfs}本身以GPT-3.5-Turbo作为基础模型,本文还利用DeepSeek R1对基础模型进行了替换,命名为LetTe_{nfs_ds},实验结果如表7所示。在总共107个失效测试用例中,DSR1正确修复了37个失效测试用例(占全部失效测试用例的34.58%,下同),LetTe_{nfs}和LetTe_{nfs_ds}则分别正确修复了73个(68.22%)和74个(69.16%)失效测试用例。基于该实验结果,并结合对日志文件的分析,得出以下发现。

表7 不同大模型对本文方法修复效果的影响

Table 7 Influence of different LLMs on the repair effectiveness of our approach

方法	正确修复的失效测试用例个数					总计
	Admidio	MantisBT	PHP-Fusion	Collabtive	MRBS	
DSR1	4	4	26	3	0	37
LetTe _{nfs_ds}	5	18	43	5	3	74
LetTe _{nfs}	5	17	42	5	4	73

(1)基于测试意图的Web测试用例修复方法的工作流对于提高测试用例修复正确率具有积极意义。对比DSR1和LetTe_{nfs_ds},在同样以DeepSeek R1作为基础模型的情况下,将DeepSeek R1嵌入本方法中可以使得测试用例的修复正确率比直接使DeepSeek R1提高34.58%,这也再一次证明了本方法的有效性。

(2)本文方法框架具有充分合理性,不会受到不同模型的影响。在本实验中,将方法中原先采用的GPT-3.5-Turbo替换为DeepSeek R1,由结果可知,采用这两种不同的模型可以实现近似的修复效果,进而证明方法的设计具有合理性,即在不同大模型的基础上依然能够实现有效的Web测试用例修复。

4 结论

本文提出了一种测试意图驱动的Web应用程序测试用例自动修复方法LetTe,该方法使用LLM,利用提示工程,并基于思维链推理构建微调模型。在修复过程中,通过LLM深入分析失效测试用例,

最终在测试意图和失效原因这类高层修复指导下实现对失效测试用例的有效修复。为了评估LetTe的有效性,在7个真实Web应用程序上进行了实验,在自行构建的107个失效测试用例中,修复正确率达到85.05%,证明了测试意图的指导作用。在过往研究数据集上,LetTe的正确修复率也达到了75%,性能优于其他基线方法。

然而,本文所研究的方法无法有效地对诸如操作逻辑变更这类复杂情况下导致的测试用例失效进行修复,表明方法仍有改进空间。为了获得更好的修复效果,后续研究将针对上述问题进行改进。

参考文献:

- [1] PINTO L S, SINHA S, ORSO A. Understanding myths and realities of test-suite evolution [C] // Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. Cary, 2012: 33.
- [2] HAMMOUDI M, ROTHERMEL G, TONELLA P. Why do record/replay tests of web applications break? [C] // 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST). Chicago, 2016:

- 180–190.
- [3] CHOUDHARY S R, ZHAO D, VERSEE H, et al. WATER: web application TEST repair[C]//Proceedings of the First International Workshop on End-to-End Test Script Engineering. Toronto, 2011: 24–29.
- [4] LIN Y Z, WEN G Y, GAO X. Automated fixing of web UI tests via iterative element matching[C]//2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). Luxembourg, 2023: 1188–1199.
- [5] STOCCO A, YANDRAPALLY R, MESBAH A. Visual web test repair[C]//Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Orlando, 2018: 503–514.
- [6] HAMMOUDI M, ROTHERMEL G, STOCCO A. WATERFALL: an incremental approach for repairing record-replay tests of web applications[C]//Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Seattle, 2016: 751–762.
- [7] QI X F, QIAN X, LI Y H. Semantic test repair for web applications[C]//Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. San Francisco, 2023: 1190–1202.
- [8] BRISSET S, ROUVOY R, SEINTURIER L, et al. SFTM: fast matching of web pages using similarity-based flexible tree matching[J]. Information Systems, 2023, 112: 102126.
- [9] WEN Z Z, LU Y F, XU T T, et al. Enhancing web test script repair using integrated UI structural and visual information[C]//2024 IEEE International Conference on Software Maintenance and Evolution (ICSME). Flagstaff, 2024: 75–86.
- [10] LIU Z, CHEN C Y, WANG J J, et al. Make LLM a testing expert: bringing human-like interaction to mobile GUI testing via functionality-aware decisions[C]//Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. Lisbon, 2024: 100.
- [11] YOON J, FELDT R, YOO S. Intent-driven mobile GUI testing with autonomous large language model agents[C]//2024 IEEE Conference on Software Testing, Verification and Validation (ICST). Toronto, 2024: 129–139.
- [12] OpenAI Developer Platform. Six strategies for getting better results [EB/OL]. [2025-02-27]. <https://platform.openai.com/docs/guides/prompt-engineering/six-strategies-for-getting-better-results>.
- [13] ZIFTCI C, KRÜGER I. Test intents: enhancing the semantics of requirements traceability links in test cases[C]//Proceedings of the 28th Annual ACM Symposium on Applied Computing. Coimbra, 2013: 1272–1277.
- [14] GENG M Y, WANG S W, DONG D Z, et al. Large language models are few-shot summarizers: multi-intent comment generation via in-context learning[C]//Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. Lisbon, 2024: 453–465.
- [15] MU F W, CHEN X, SHI L, et al. Developer-intent driven code comment generation[C]//2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). Melbourne, 2023: 768–780.
- [16] LI X Y, D'AMORIM M, ORSO A. Intent-preserving test repair[C]//2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST). Xi'an, 2019: 217–227.
- [17] YAO S Y, YU D, ZHAO J, et al. Tree of thoughts: deliberate problem solving with large language models [EB/OL]. (2023-12-03) [2025-02-27]. arXiv: 2305.10601.
- [18] WANG X Z, WEI J, SCHUURMANS D, et al. Self-consistency improves chain of thought reasoning in language models [EB/OL]. (2023-03-07) [2025-02-27]. arXiv:2203.11171.
- [19] FAN Z Y, GAO X, MIRCHEV M, et al. Automated repair of programs from large language models[C]//2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). Melbourne, 2023: 1469–1481.
- [20] SHAO F, XU R, HAQUE W, et al. WebEvo: taming web application evolution via detecting semantic structure changes[C]//Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2021: 16–28.

A web test case repair approach based on test intent

TAO YingJie¹ WANG WeiWei² GUO JunXia^{1*}

(1. College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029;

2. College of Information Engineering, Beijing Institute of Petrochemical Technology, Beijing 102617, China)

Abstract: Web test case automatic repair technology plays a crucial role in the maintenance of web application test suites. As web applications are frequently updated, changes may introduce web elements in new versions, causing test cases to fail. A major issue with existing methods is the lack of effective representation of the test intent and analysis of the reasons for failure, resulting in a failure to guide and assist in test case repair at a macro level. This makes it impossible to effectively repair broken test cases caused by test flow changes or propagated breakages. Empirical analysis shows that test intent and causes of failure are very useful for test case repair. This paper proposes a test intent-driven web test repair approach named LetTe (guiding LLM to fix Web UI tests based on test intent), which first parses the test intent and failure reasons of broken web test cases, and then guides the large language model (LLM) to fix them via prompt design and fine-tuning. LetTe's repair logic simulates that of a human expert—given the test intent and reasons for failure, “think about” possible repair plans and then generates repair candidates via the corresponding chain-of-thought. This paper evaluates LetTe across seven Web applications collected from open-source websites and publicly available datasets. The experimental results show that our approach achieves a 75% correct repair rate, which is higher than that of all baseline methods.

Key words: test intent; Web application test; test case repair; large language model

(责任编辑:吴万玲)