软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007414] [CSTR: 32375.14.jos.007414]

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

基于大模型语义匹配的跨平台移动应用测试脚本录制回放*

虞圣呈[1,2], 房春荣[1,2], 钟 葉[1,2], 张犬俊[1,2], 刘 钦[1,2], 刘 嘉[1,2], 郑 滔[1,2], 陈振宇[1,2]

[1](计算机软件新技术全国重点实验室 (南京大学), 江苏 南京 210093)

[2](南京大学 软件学院, 江苏 南京 210093)

通信作者: 房春荣, E-mail: fangchunrong@nju.edu.cn

摘　要: GUI 测试是移动应用质量保障的重要手段之一. 随着移动生态的不断发展, 尤其是国产移动应用 (如鸿蒙等) 生态的强势崛起, GUI 测试脚本跨平台录制回放成为了当前 GUI 测试的主要挑战之一. 开发者需将传统平台中 GUI 测试脚本迁移至新兴环境中, 以保证应用质量可靠性与多平台用户体验一致性. 然而, 不同平台间的底层实现差异导致了移动应用测试跨平台迁移的重大障碍, 这一挑战在面向新兴国产移动生态平台的测试迁移方面尤为突出. 移动应用的跨平台测试脚本录制回放是确保应用在不同操作系统和设备上保持一致性和高质量用户体验的关键. 现有技术仅解决了“一对一”事件匹配的情况, 而由于平台间 GUI 开发实践的不一致性, 测试事件的回放并非完全一对一映射, 而存在普遍的“多对多”映射情况, 即若干测试事件所对应的业务流程在不同平台上对应数量不等的测试事件. 为解决上述问题与挑战, 提出了一种基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法 (LLMRR). LLMRR 方法结合图像匹配、文本匹配和大语言模型语义匹配技术, 在录制阶段通过图像分割算法记录用户操作信息, 并保存为录制测试脚本; 在回放阶段, 通过图像匹配和文本匹配模块在回放页面上找到对应的控件, 执行操作, 当无法匹配时, 调用大模型语义匹配模块进行语义匹配, 确保在不同平台上的高效运行. 对国产鸿蒙应用的测试进行了探索, 选择了 20 个应用共 100 个测试脚本, 在 iOS、安卓和鸿蒙平台之间进行迁移测试, 并与当前最先进跨平台测试脚本录制回放方法 LIRAT 和 MAPIT 进行有效性对比. 结果表明, LLMRR 方法在测试脚本录制回放中均表现出显著优势.

关键词: GUI 测试; 鸿蒙应用测试; 测试脚本; 录制回放; 大语言模型

中图法分类号: TP311

中文引用格式: 虞圣呈, 房春荣, 钟葉, 张犬俊, 刘钦, 刘嘉, 郑滔, 陈振宇. 基于大模型语义匹配的跨平台移动应用测试脚本录制回放. 软件学报. http://www.jos.org.cn/1000-9825/7414.htm

英文引用格式: Yu SC, Fang CR, Zhong Y, Zhang QJ, Liu Q, Liu J, Zheng T, Chen ZY. Semantic Matching-based Cross-platform Mobile App Test Script Record and Replay via Large Language Models. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7414.htm

Semantic Matching-based Cross-platform Mobile App Test Script Record and Replay via Large Language Models

YU Sheng-Cheng[1,2], FANG Chun-Rong[1,2], ZHONG Ye[1,2], ZHANG Quan-Jun[1,2], LIU Qin[1,2], LIU Jia[1,2], ZHENG Tao[1,2], CHEN Zhen-Yu[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Software Institute, Nanjing University, Nanjing 210093, China)

Abstract: GUI testing is one of the most important measures to ensure mobile application (App) quality. With the continuous development of the mobile ecosystem, especially the strong rise of the domestic mobile ecosystem, e.g., HarmonyOS, GUI test script recording and

* 基金项目: 国家自然科学基金 (62272220, 62372228); 中央高校基本科研业务费专项资金 (14380029)
收稿时间: 2024-08-30; 修改时间: 2024-11-19, 2025-01-17; 采用时间: 2025-02-18; jos 在线出版时间: 2025-08-27

replay has become one of the prominent challenges in GUI testing. GUI test scripts must be migrated from traditional mobile platforms to emerging mobile platforms to ensure the reliability of App quality and consistency in user experience across diverse platforms. However, differences in underlying implementations across platforms have created substantial obstacles to the cross-platform migration of mobile App test scripts. This challenge is particularly pronounced in the testing migration for emerging domestic mobile ecosystem platforms. Cross-platform test script recording and replay is essential for maintaining consistency and a high-quality user experience across different platforms and devices. Current state-of-the-art approaches only address the “one-to-one” test event matching situations. However, due to inconsistencies in development practices across platforms, the replay of test events does not always map “one-to-one”; instead, “multiple-to-multiple” mapping situations are common. This means that some test events need to be mapped to a different number of test events to fulfill the same business logic. To address these issues and challenges, this study proposes a cross-platform mobile App test script recording and replay method based on large language model semantic matching (LLMRR). The LLMRR method integrates image matching, text matching, and large language model semantic matching technologies. During the recording phase, user operation information is captured using image segmentation algorithms and saved as recorded test scripts. During the replay phase, corresponding widgets on the replay App page are located using image matching and text matching modules to execute operations. When matching fails, the large language model semantic matching module is invoked for semantic matching, ensuring efficient operation across different platforms. This study presents the first exploration of testing for domestic HarmonyOS Apps, using 20 Apps and a total of 100 test scripts for migration testing across iOS, Android, and HarmonyOS platforms. The effectiveness of the LLMRR method is compared with the current state-of-the-art cross-platform test script recording and replay approaches, LIRAT and MAPIT. The results demonstrate that the LLMRR method exhibits significant advantages in test script recording and replay.

Key words: GUI testing; HarmonyOS App testing; test script; record and replay; large language model (LLM)

随着移动应用的迅猛发展, 各类应用已深深融入人们的日常生活中, 并成为现代社会不可或缺的一部分. 面对用户日益增长的需求和期望, 开发者也在不断追求技术创新和用户体验的提升, 以提供更高效、更便捷、更智能的服务[1]. 在这个背景下, 各种移动操作系统不断涌现, 国产系统如鸿蒙系统 (HarmonyOS) 逐渐崭露头角, 凭借其独特的优势和创新的生态系统, 迅速成为市场中的重要一员, 受到越来越多用户和开发者的关注和青睐. 自 2019 年面世以来, 鸿蒙系统正成为继安卓 (Android) 和 iOS 之后的重要选择, 为用户和开发者提供更多元化的选择和更高质量的服务. 随着其不断发展, 鸿蒙应用生态发展及其质量保障的重要性愈发凸显[2]. 对于鸿蒙应用测试而言, 测试者需要考虑到各种可能的情况, 包括不同设备的屏幕尺寸、分辨率、性能差异等, 跨平台兼容性将成为鸿蒙应用的基本要求, 开发者需要确保应用在不同设备型号和操作系统版本下都能正常运行, 提高应用的普适性和用户体验. 此外, 还需要对应用在不同版本鸿蒙系统上的表现进行测试, 以确保应用的稳定性和兼容性. 这些工作不仅需要耗费大量的时间和精力, 还需要具备专业的测试技能和经验.

虽然鸿蒙操作系统在初期与安卓系统具有一定的兼容性, 但随着其逐渐发展, 无论是系统特性还是应用特性, 均与安卓系统有着较大差异. 在内核架构方面, 鸿蒙系统基于分布式的微内核设计, 可以根据设备的功能需要进行系统内核功能的搭配, 而安卓系统则基于 Linux 的宏内核架构, 运行在不同的设备上需要一整套内核功能; 因此鸿蒙系统也具有更强的设备兼容性, 支持智能手机、智能穿戴设备、电脑、电视等多种智能家居设备, 形成一个无缝的、统一的操作系统. 此外, 鸿蒙系统通过多重安全机制, 如隔离、加密通信和身份验证等, 确保了较高的安全性能. 在其应用生态开发过程方面, 鸿蒙系统支持统一的开发框架和 API, 开发人员可以通过一次开发, 适配多个设备, 减少开发工作量, 且可以通过方舟编译器将软件代码直接编译成机器语言, 提高了软件的运行效率, 安卓系统则是在软件运行时边运行边编译, 运行效率相对较低. 因此, 对于其图形用户界面 (graphical user interface, GUI) 测试而言, 由于不同操作系统间底层支持的差异性, 现有的安卓和 iOS 版本应用测试脚本无法直接迁移到鸿蒙系统上[3]. 不同操作系统间底层支持差异使得应用在不同平台上的实现方式和交互流程有所不同, 对应 GUI 控件的定位方式与页面结构也不尽一致, 导致测试脚本需要针对每个平台进行特定设计, 无法直接迁移. 现有的测试方法通常是为特定平台量身定制的, 缺乏跨平台的通用性. 这意味着开发者需要为每个平台分别编写和维护测试脚本, 既耗时又易出错, 且需要测试开发者熟悉不同的测试框架. 然而, 尽管不同平台的应用实现方式和交互流程不同, 但其测试的核心目的往往是相同的, 即验证应用功能的正确性和用户体验的一致性. 然而, 由于底层架构的不同导致动作序列存在显著差异, 确保所有平台上的测试结果一致性变得更加困难. 传统测试脚本录制回放仅解决了同

平台脚本迁移或跨平台脚本迁移中无测试动作序列差异的情况, 通过对测试动作“一对一”的映射分析, 完成录制回放. 而对于更复杂的跨平台测试事件迁移, 即存在测试动作序列差异的情况时, 测试动作往往无法进行“一对一”映射分析, 而是需要灵活地处理不同平台间的业务逻辑流程差异和控件差异, 解决“多对多”事件映射, 即若干个录制步骤可能对应多个不等数量的回放步骤. 为了解决上述问题, 需要开发更加智能化和通用的测试工具和方法, 以适应不同平台的特性, 自动调整测试脚本的动作序列, 减少人为干预, 提高测试效率和准确性.

随着大语言模型 (large language model, LLM) 的发展, 对 GUI 测试脚本实现深度语义理解成为可能. 大语言模型能够有效理解文本信息. 其通过学习文本数据中的模式、关系和上下文信息, 具备处理复杂语言任务的能力. 测试脚本代码及 GUI 界面所呈现相关语义信息能够通过精心设计的提示词, 被大语言模型理解, 并一直为上下文基础将测试事件在新的回放设备中进行映射. 基于大语言模型的强大文本信息理解能力, 本文提出了一种基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法, 旨在解决不同平台操作流程差异带来的挑战. 该方法主要包括录制阶段和回放阶段. 在录制阶段, 系统通过图像分割算法将页面分割成控件区, 记录用户操作信息, 包括当前页面截图、被操作控件截图、操作坐标和操作类型等. 这些信息会被整理并保存为录制测试脚本. 在回放阶段, 系统读取录制信息, 并通过图像匹配模块和文本匹配模块在回放页面上找到对应的控件, 执行相应操作. 当遇到无法匹配的情况时, 调用大模型语义匹配模块进行语义匹配, 跳过冗余步骤继续操作.

与传统方法所解决的“一对一”事件映射不同[4], 本方法着力解决“多对多”事件映射, 即若干个录制步骤可能对应多个不等数量的回放步骤. 这种多对多映射模式能够更灵活地处理不同平台间的流程差异和控件差异, 提高了测试脚本的通用性和适应性. 大语言模型在实现“多对多”映射中起到了关键作用, 特别是在语义匹配方面. 通过引入大语言模型, 系统能够理解和处理复杂的跨平台流程差异. 当录制步骤在回放页面上无法找到对应控件时, 系统会将录制信息和回放页面信息输入大语言模型. 大语言模型通过语义匹配, 分析和理解录制和回放过程中的操作意图和控件信息, 推理出可能的操作路径, 并提供跳过冗余步骤或添加必要步骤的建议. 这种基于语义匹配的推理能力极大地提高了跨平台测试的智能化程度, 确保了测试脚本在不同平台上的高效运行. 总的来说, 该方法通过引入大模型语义匹配模块成功解决了不同平台间的操作流程差异问题, 提高了测试脚本的适应性和通用性, 为移动应用的跨平台测试提供了一种高效、智能的解决方案.

本文的主要研究贡献如下.

(1) 提出了一种基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法, 可适用于不同平台间脚本迁移.

(2) 提出了一种基于大语言模型的测试时间序列多对多语义匹配方法, 实现了跨平台测试脚本录制回放时测试动作语义匹配.

(3) 针对国产鸿蒙应用生态平台进行测试脚本录制回放方法的探索, 并通过大规模实证评估, 表明了所提出方法在回放成功率上与现有最先进基线方法对比有较大提升.

本文第 1 节介绍 GUI 测试脚本录制回放相关工作及研究现状. 第 2 节介绍相关背景知识, 包括移动应用 GUI 测试和鸿蒙应用测试. 第 3 节介绍本文所开展研究的动机实例. 第 4 节介绍本文所构建的基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法. 第 5 节通过实验验证了本文所提出方法的有效性. 第 6 节展示了本文所提出方法的相关讨论, 包括优势与局限性分析、有效性威胁分析、性能开销分析以及未来工作展望. 最后对全文进行总结. 本文的代码与相关数据已在 GitLink 平台开源, 以供对本文所提出算法的复现, 也为后续研究提供了实现基础: https://gitlink.org.cn/yusc/LLMRR.

1 GUI 测试脚本录制回放相关工作

测试脚本录制回放方法作为一种经典的自动化测试技术, 因其操作简便直观, 逐渐被广泛应用于软件测试领域. 录制回放技术的基本原理是通过录制用户在软件上的实际操作生成测试脚本, 然后在需要时回放这些脚本以重现用户操作, 从而实现测试自动化[3]. 与传统的脚本编写方法相比, 录制与回放技术具有显著优势: 首先, 它降低

了测试的技术门槛, 使得非专业人员也能参与自动化测试; 其次, 它能够真实再现用户操作, 具有较高的测试覆盖率和可靠性. 此外, 这种技术在测试环境重现和回归测试中尤为有效, 有助于发现和定位软件系统中的潜在缺陷. 然而, 尽管录制与回放技术有诸多优势, 其在实际应用中也面临一些挑战, 例如脚本的可维护性问题、对动态 GUI 元素的适应性不足等. 随着研究的深入和技术的进步, 针对这些问题的解决方案不断涌现, 使得录制与回放技术在 GUI 测试中的应用前景愈加广阔.

近年来, 录制与回放技术迎来了新的发展阶段. 先进的图像识别和计算机视觉技术被引入 GUI 测试[5], 使得录制与回放工具在处理复杂界面和动态元素时表现得更加灵活和高效. 以 Sikuli[6] 为代表的现代工具利用屏幕截图和图像匹配技术, 有效提升了录制回放的准确性和适应性. 这些工具不仅能够识别并操作界面元素, 还能通过脚本自动化处理复杂的测试场景. 此外, 研究人员还提出了诸如脚本修复、自动化测试脚本生成等新方法, 以解决录制与回放技术在应用中的各种挑战. 通过这些工作, 录制回放技术在现代软件开发和测试流程中占据了不可或缺的位置, 为提升软件质量和开发效率作出了重要贡献.

随着移动设备的多样化和普及, 跨设备录制回放技术在软件测试中变得越来越重要. 现代应用需要在各种设备上运行, 包括智能手机、平板电脑、智能手表和其他物联网设备[7]. 此外, 随着以鸿蒙操作系统为代表的新一代国产化操作系统的诞生与迅猛发展, 移动应用运行环境变得越发复杂. 移动应用运行的软硬件环境具有不同的设备屏幕尺寸、分辨率、硬件性能和操作系统版本, 导致其行为和用户体验可能显著不同. 通过跨设备录制回放技术, 测试人员可以在一种设备上录制测试脚本, 并在其他设备上重放这些脚本, 以验证应用在不同设备上的兼容性和表现. Havranek 等人[8] 提出了 V2S, 依靠图像处理和基于规则的方法来检测和分类视频中捕获的用户手势, 并将这些手势转换为可重放的测试脚本. Bernal-Cárdenas 等人[9] 在此基础上加以改进, 采用了对象检测和图像分类技术, 以更高的精度检测和分类用户手势和 GUI 控件. 此外, 为了改进安卓应用的测试过程, 减少人工操作的负担, 并提高测试的效率和准确性, Fazzini 等人[10] 提出的 Barista 能够准确编码用户定义的测试用例为测试脚本, 并内置可在多个平台上运行的断言, 实现在多个设备和操作系统版本上自动运行这些脚本的目的. Halpern 等人[11] 提出通过将一组来自特定设备的触摸屏事件映射到一组可重新排序并注入另一设备的虚拟用户交互来实现跨设备的记录和重放. Hu 等人[12] 介绍了 VALERA 工具, 通过一种新颖的面向流的记录和回放方法, 在高精度和低开销之间取得平衡, 支持对安卓应用的传感器和网络输入、事件调度及跨应用通信进行记录和回放. Liu 等人[13] 介绍了一种基于捕获和回放方法的自动化测试安卓应用的方法, 通过捕获用户事件并转换为可执行的 Robotium 测试脚本重放用户操作, 还允许在捕获用户交互时插入断言以验证安卓 UI 组件的输出, 并实现了一个支持工具来展示该方法的实用性.

移动应用需要在多个操作系统 (如安卓、iOS、鸿蒙等) 上运行, 以满足不同用户的需求. 不同平台的界面元素、系统行为和兼容性各异, 导致单一平台上的测试不足以保证软件在所有平台上的一致性和可靠性[14]. 通过跨平台录制回放技术, 测试人员可以在一个平台上录制测试脚本, 然后在其他平台上重放这些脚本, 从而验证应用在不同操作系统上的功能和表现. 这不仅节省了时间和资源, 还提高了测试覆盖率和效率[15]. 传统的录制与回放方法主要依赖于小部件匹配技术, 但在不同设备和平台上, 由于视觉细节的微小差异, 这些方法的有效性受到限制. 为了减少为每个平台单独编写测试用例的时间和精力, Zhang 等人[16] 据此提出了一种改进的跨平台录制与回放方法 ReSPlay, 通过分析并匹配录制过程中捕获的 GUI 序列, 确保在回放时能够准确定位和操作目标小部件, 解决了不同平台和分辨率设备上的小部件匹配问题, 有效解决了传统方法在面对视觉细节差异时的局限性, 提高了录制与回放的有效性. Alégroth 等人[17] 制作了一款名为 JAutomate 的 VGT 工具, 通过结合图像识别与记录和回放功能, 提供了高系统级的测试自动化. Ji 等人[18] 首次开展了对基于视觉的控件映射在跨平台 GUI 测试迁移中的应用进行全面研究, 结果表明现有方法尚有改进空间, 还需更复杂算法的处理.

在现代软件开发中, 应用往往具有相似或共享的功能模块, 如登录、支付、数据输入等. 鸿蒙操作系统作为一款面向全场景智慧生活的分布式操作系统, 进一步增强了这一特点. 其独特的分布式架构和能力使得跨设备、跨应用的无缝协同和资源共享成为可能. 跨应用和跨设备的录制回放技术在鸿蒙系统中尤为重要, 因为它允许这些共性功能的测试用例在多个应用和设备之间共享和重用, 从而显著提高测试覆盖率, 确保所有相关应用的核心功

能都经过充分测试. 这样, 不仅可以提升了应用质量和用户体验, 还可以推动了整个生态系统的协同和一致性. Liang 等人[19]提出了一种跨应用的录制与回放技术 RIDA, 其通过捕获源应用的 GUI 交互并生成相应的事件序列, 然后在目标应用中寻找相似的 GUI 元素来重放这些事件, 该方法依赖于高级的图像识别和模式匹配技术, 确保在不同应用之间实现高效的交互序列重放. Li 等人[7]在研究中展示了一种基于贪婪算法的跨设备录制与回放方法 Rx 框架, 旨在解决安卓应用在不同设备上的 GUI 一致性问题. Behrang 等人[20]提出了应用迁移器, 利用不同但相关应用的 GUI 之间的相似性来重用和适应测试用例, 从而降低了应用测试的成本. Gomez 等人[21]开发了 RERAN 工具, 能够精准地捕捉和重放 GUI 和传感器事件, 无须访问应用源代码或修改安卓平台, 在多跨应用场景下具有广泛适用性.

2 背景知识

对于传统移动应用 GUI 测试、鸿蒙应用测试和应用了大语言模型的软件测试, 现代测试技术的发展极大地提高了测试的效率和准确性. 基于视觉的 GUI 测试利用计算机视觉和深度学习, 实现自动识别和操作 GUI 元素, 减少对底层代码的依赖. 鸿蒙操作系统的专用自动化测试框架确保了应用在各种终端设备上的兼容性和稳定性. 大语言模型为测试用例生成、程序修复和调试提供了强大的自然语言处理能力. 这些技术的结合解决了传统测试方法的不足, 推动了软件测试领域的进步.

2.1 传统移动应用 GUI 测试

图形用户界面 (GUI) 测试是软件测试中的一个重要领域, 对其进行有效和高效的测试至关重要. 传统的手工黑盒 GUI 测试既耗时又繁琐, 无法满足现代软件开发中快速迭代和频繁更新的需求. 因此, 自动化 GUI 测试方法逐渐成为研究的关注重点. 研究人员提出了多种自动化测试工具和方法, 但这些方法通常依赖于测试脚本, 往往对 GUI 的具体实现细节高度依赖, 当 GUI 发生变化时, 脚本需要频繁更新, 导致维护成本高.

在此背景下, 基于视觉的 GUI 测试方法应运而生. 基于视觉的 GUI 测试方法在检测和操作 GUI 元素的时候, 可以有效避免对底层代码的依赖, 具有更高的通用性和适应性. 这些方法通常使用计算机视觉和深度学习技术来识别和定位 GUI 中的不同元素, 如按钮、文本框、下拉菜单等, 从而实现自动化测试. 在基于视觉的 GUI 测试当中, 大约有 47.4% 的研究关注点在于 GUI 元素检测[22]. 传统的计算机视觉方法如 Canny 边缘检测[23]、SIFT (scale-invariant feature transform) 算法[24]和 OCR (optical character recognition) 技术[25]等, 能够在一定程度上识别 GUI 中的元素. 这些方法通过处理图像的像素级信息, 提取出有意义的特征进行分析. 然而, 传统方法在应对复杂背景和多样化元素时表现有限, 难以实现高精度的检测. 近年来, 深度学习技术在图像识别领域取得了显著进展, 也被广泛应用于 GUI 元素检测中. Sun 等人[26]利用预训练的卷积神经网络 (convolutional neural network, CNN) 自动学习从截图中分析提取的成分图像中的高级特征, 并应用于 GUI 元素分类当中. 其他常用的深度学习模型如 YOLO 系列[27], 在处理 GUI 图像时也表现出较为高精度检测结果.

许多研究者致力于开发和优化基于视觉的 GUI 测试方法, 提出了多种创新的方法和工具. 近年来, 基于图像理解的识别应用 GUI 组件的方法为自动组件定位具有挑战性的场景提供了新的解决方案和方法. 一些研究主要集中在 GUI 元素的分类上. Altinbas 等人[27]研究了基于 YOLOv5 的移动 UI 图像的 GUI 元素检测方法, 通过使用深度学习模型对 UI 图像进行分析和分类, 实现了高效的 GUI 元素识别和定位. Chen 等人[28]讨论了一种结合了常见图标类型的分类、少样本学习对图标进行分类, 以及基于像素的上下文信息 (如附近文本和图标内修饰符) 的识别来实现移动应用图标的全面标注的新的系统. Chiatti 等人[29]开发了一个结合 OpenCV 图像处理和光学字符识别的模块, 对截图进行预处理、文本提取和索引, 从而构建一个用于行为研究和即时健康干预的专用搜索引擎. Jaganeshwari 等人[30]分析并设计一种高效的 Web 应用自动化测试策略, 利用计算机视觉进行 Web 图形用户界面测试, 同时介绍用于对象检测、分类和评估的参数, 并通过机器学习算法进行图像处理以提高准确性. 其他的研究旨在通过视觉信息来识别异常的 GUI 组件. 在自动检测错误方面, Ji[31]提出了一个利用机器学习方法检测与文本和图像相关的小部件错误的自动检测平台, 能够检测更复杂的 GUI 错误. Liu[32]提出了一种名为 OwlEye 的新方

法, 基于深度学习模型分析 GUI 截图中的视觉信息, 能够检测存在显示问题的 GUI, 并精确定位问题区域, 帮助开发人员修复错误. 脚本在自动化 GUI 测试中具有重要作用. 鉴于传统的 GUI 测试工具需要在脚本中预先配置所有操作和指令, 对 GUI 环境的变化敏感且需要手动重新配置, Fang 等人[33]提出了一种基于图像识别的 GUI 测试方法, 创新性地改进了 SIFT 和 Random Fern 等算法, 以检索最有效的特征并排除无效部分. Wu 等人[34]提出了一种基于图像识别技术的自动化 GUI 测试方法, 通过结合 OCR(光学字符识别) 图像识别技术和图像比对技术, 开发了一个能够自动识别和更新测试脚本的工具. 此外, GUI 的迭代更新增加了测试的复杂性. Chen 等人[35]提出了一种将传统的 CV 方法用于非文本元素区域检测, 同时与深度学习模型相结合, 充分发挥二者的优势的方法, 取得了优异的检测效果. Xie 等人[36]也提出了一个结合了两种传统计算机视觉方法 (Xianyu 和 REMAUI) 和 3 种深度学习模型 (Faster-RCNN、YOLOv3 和 CenterNet) 的新型 GUI 元素检测工具 UIED 来实现对复杂 GUI 图像的精确检测, 获得了更加优秀的检测结果. Feiz 等人[37]训练了一个结合 UI 对象检测器和 Transformer 模型架构的屏幕相似性模型识别同一应用中截图集合中的相同屏幕实例, 还训练了一个使用孪生网络架构的屏幕转换模型识别相似性并检测交互过程中出现的特定事件, 解决了自动应用爬取、自动化宏回放和大规模应用数据集分析中的存在的挑战. Zhang 等人[38]所提出的 MigratePro 方法通过综合各来源的测试脚本以实现测试迁移的增强, 然而该方法依然忽略了脚本之间由于功能细微差异引起的流程差异, 从而在特定场景下可能会失去作用. MigratePro 作为当前代表性的 GUI 测试录制回放方法, 依然以比对-搜索的方法来完成测试脚本回放时目标控件的缺乏, 但其通过更完善的对比信息收集有效提升了录制回放的成功率. 然而, 现有应用设计间存在较多差异, 尤其是在跨平台版本间. 针对上述信息, MigratePro 可能会通过已有脚本中相关情况完成部分录制回放工作, 但其并未在方法根本上对上述情况进行处理. 本文通过大语言模型提示引导进行冗余步骤的理解与识别, 并完成测试的回放. 在处理多对多事件映射情况时, 本方法深入理解 GUI 控件所包含的丰富的与功能业务逻辑紧密关联的语义信息, 以及部分 GUI 控件的进入路径所强烈依赖的对功能业务逻辑, 从而实现多对多事件映射. 一些研究还关注于使用自然语言描述屏幕截图中描述的用户界面, 以促进理解 UI 的主要目的. Kirinuki等人[39]便利用自然语言处理技术理解网页元素的语义, 并创建启发式搜索算法探索网页和寻找有效的测试过程, 自动化 Web 应用测试. 此外, 跨平台测试长期以来一直是测试人员关注的重点. 为解决移动设备碎片化导致的自动化视觉 GUI 测试工具在不同设备上运行时结果不一致的问题, Ardito 等人[40]使用了如 SIFT、SURF 和 AKAZE 等的特征匹配算法提高了视觉图形用户界面测试的鲁棒性.

综上, 基于视觉的 GUI 分析在 GUI 测试中扮演着关键的角色, 它为测试人员提供了识别、定位和理解 GUI 元素的强大工具. 这些研究不仅提高了 GUI 测试的效率和准确性, 而且还扩大了它们的适用性, 以解决特殊场景的测试需求, 有可能为更广泛的移动应用测试任务提供强大的支持.

2.2 鸿蒙生态质量保障

鸿蒙操作系统是一款由华为开发的基于微内核的、面向全场景的开放分布式操作系统, 自问世以来便始终受到业内外广泛关注. 截至 2023 年 8 月, 鸿蒙已经成为除安卓和 iOS 外的第三大手机操作系统. 目前, 包含社交、旅游、游戏等多个领域的开发者已经展开鸿蒙原生应用的开发. 高质量的应用是建立用户信任和满意度的基石, 对于鸿蒙应用而言, 优良的应用体验直接关系到用户对鸿蒙系统整体印象的形成, 因此, 针对鸿蒙系统的软件测试具有十分重要的意义.

首先, 鸿蒙操作系统的稳定性和安全性直接关系到用户的使用体验和个人信息安全, 因此必须经过充分的软件测试来确保其质量. 其次, 鸿蒙操作系统面向全场景, 若想要合理应对多种终端设备, 则还需要经过严格的兼容性测试, 来确保在各种不同的硬件平台上都能够正常运行. 另外, 鸿蒙操作系统还需要针对不同的应用场景进行测试, 以验证其在智能家居、车载系统等领域的适用性. 在这样的背景之下, 鸿蒙开发者研发了一套全新的自动化测试框架——Hypium, 用以支持鸿蒙系统应用测试. 该框架提供了单元测试框架和 UI 测试框架两个子框架, 分别用于程序的内部功能逻辑测试和 UI 界面测试, 并针对鸿蒙应用的多端部署、多语言开发等特点做了相应的优化. 它支持以插件形式集成到 DevEco Studio 中, 集成开发环境将自动生成测试目录、测试类和测试用例模板等, 让开发者在应用开发的过程中可以快速编写和执行测试用例, 实现应用的高效验证.

除了内部测试之外, 为进一步了解用户诉求, DevEco Testing Hypium 团队发布了 DevEco Testing Hypium 基础测试框架及能力增强的 SDK, 并提供了面向鸿蒙操作系统自身应用的基础录制回放工具. 然而, 对于新手测试人员, 由于其缺乏应用自动化测试经验, 相关测试工具技术门槛高, 难以从其他平台进行直接迁移. 此外, 自动化测试效率, 包括控件定位准确性、UI 界面反复变化等问题也是自动化测试工具面临的难题.

2.3 大语言模型

大语言模型是基于深度学习技术训练的大规模神经网络, 能够理解和生成自然语言文本. 它们的核心在于使用大量文本数据进行训练, 通过学习文本数据中的模式、关系和上下文信息, 从而具备处理复杂语言任务的能力. 随着计算能力的提升和海量数据的积累, 语言模型的规模和复杂性也在不断增加, 使得它们在各种自然语言处理任务中表现出色. GPT-4o 是 OpenAI 开发的最新一代大语言模型. GPT-4o 是一个多模态大模型, 支持文本、音频和图像的任意组合输入, 并能生成文本、音频和图像的任意组合输出. 与现有模型相比, 在模型规模、训练数据量和算法优化上都有显著提升, 具备更高的语言理解和生成能力, 同时在视觉和音频理解方面尤其出色.

在软件测试领域, 大语言模型的引入带来了革命性的变化. 传统的软件测试方法往往依赖于人工编写测试用例、手动分析缺陷和日志文件, 这不仅费时费力, 还容易出现遗漏和错误. 大语言模型通过其强大的自然语言处理能力, 可以在多个方面提升软件测试的效率和效果.

针对现有的基于模型等的 GUI 测试工具面临测试覆盖率低、泛化能力不足和对训练数据依赖性强等问题, Liu 等人[41]提出了一种新颖的自动化 GUI 测试方法 GPTDroid, 该方法将大语言模型应用于移动 GUI 测试, 通过功能感知的记忆提示机制, 解决了 LLM 在多轮交互中可能面临的记忆困难和局部困境问题, 实现了类人交互, 在移动 GUI 测试中展现出优异的性能. 该方法不仅在测试覆盖率和错误检测率上显著优于现有方法, 还能够发现并报告实际应用中的错误, 为开发者提供了有价值的反馈. 在测试用例生成方面, Hu 等人[42]提出了一种结合生成式 AI 的灰盒模糊测试方法, 通过利用 LLM 生成多样化的测试输入, 能够提高模糊测试的覆盖率和错误检测能力. Chen 等人[43]介绍了一个基于 ChatGPT 的自动单元测试生成工具 ChatUniTest, 该工具能够有效生成高质量的单元测试用例, 减少了手动编写测试用例的时间和成本. Plein 等人[44]探讨了利用大型语言模型基于错误报告自动生成测试用例的可行性, 研究表明, 利用大型语言模型可以有效地从错误报告中提取信息, 并生成相应的测试用例, 从而提高软件测试的效率和覆盖率. 在程序修复方面, Fan 等人[45]提出了一种利用大型语言模型进行程序自动修复的方法, 通过对大型语言模型进行微调, 可以生成高质量的修复代码, 提高程序修复的效率和准确性. Xia 等人[46]探讨了在大规模预训练语言模型时代的实际程序修复方法, 通过对大规模预训练模型进行微调和优化, 能够显著提高程序修复的效果. 在程序调试 (包括错误定位和解释) 方面, Plein 等人[47]探讨了大型语言模型在解读错误报告中的应用发现大语言模型能够有效生成错误报告的自然语言解释, 帮助开发者更快地理解和解决问题. Taylor 等人[48]介绍了一种名为 Dcc-help 的方法, 通过大型语言模型生成上下文感知的编译器错误解释, 该方法能够生成针对特定错误的详细解释, 帮助新手程序员理解错误原因. Kang 等人[49]提出了一种基于大型语言模型驱动的科学调试方法, 通过自动生成假设、观察和得出结论的循环过程, 使得模型能够自动调试代码并生成可解释的调试信息.

3 研究动机

根据全国 App 技术检测平台统计, 截至 2023 年 6 月底, 我国国内市场上监测到活跃的 App 数量为 260 万款 (https://wap.miit.gov.cn/gxsj/tjfx/hlw/art/2023/art_1e078242dee140cb99d46ecc070e7717.html). 目前, 除了安卓系统和 iOS 系统, 基于鸿蒙系统进行开发的应用数量也逐渐增多. 面对不同平台、不同设备、甚至不同版本, 移动应用在界面布局、功能实现和用户交互流程等诸多方面上均存在一定的差异, 使得利用录制回放技术进行软件测试时会出现录制冗余和录制回放两大问题, 不便于测试的高效化进行, 这对于软件测试是一项巨大的挑战.

录制冗余指的是在录制过程中, 记录下了在实际回放时不必要或多余的操作步骤, 这些步骤在目标环境的测

试中并不需要执行, 或者在目标环境中根本不存在. 例如, 录制过程中应用针对第 1 次使用出现了临时弹窗或提示, 需要关闭后再进行下一步操作, 而在回放过程中不会出现该弹窗或提示, 便可直接进行下一步操作. 回放冗余指的是在回放过程中, 遇到了录制时没有记录的额外操作步骤, 这些步骤是由于目标环境中的差异导致的, 在录制过程中未能预见和记录. 例如, 录制步骤同意软件开发者提出的某些条款, 只需要点击“同意”控件即可进入下一步操作, 但在回放页面中, 同样的条款被分开变成了两个页面, 需要先点击“下一页”才能够到条款底部, 再点击“同意”控件才可以进行下一步操作. 相较于录制操作, 回放操作需要额外进行一次操作才可以进入和录制操作相匹配的下一页面.

图 1 给出了不同平台间测试动作序列差异动机示例. 左侧两张截图和右侧 3 张截图分别对应鸿蒙系统和 iOS 系统在“美团”应用中实现“查看感冒发烧药品”操作的动作序列. 该示例中, 鸿蒙系统在美团主界面点击“美团买药”后, 直接进入买药界面; 而 iOS 系统在点击“美团买药”后, 会弹出一个广告窗口, 需要额外点击关闭广告才能进入买药界面. 当使用鸿蒙系统的录制脚本在 iOS 系统上进行回放时, 录制的操作序列为: 点击“美团买药” → 进入买药界面 → 点击“感冒发烧”, 而回放时应该执行的操作序列为: 点击“美团买药” → 关闭广告 → 进入买药界面 → 点击“感冒发烧”, 这样就导致了回放冗余. 类似的, 当使用 iOS 系统的录制脚本在鸿蒙系统上进行回放时, 则会出现录制冗余问题.

图 1　不同平台间测试动作序列差异动机示例

录制冗余和回放冗余是在移动应用测试中常见的问题, 主要源于平台、版本、设备和用户路径的差异. 传统的跨平台测试脚本录制回放技术依赖于页面布局细节, 需要开发和维护多个测试脚本, 投入大量人力成本[18]. 为了解决这个问题, 研究人员开始探索基于图像的跨平台测试脚本录制回放技术, 以图像为基础组织测试过程[50]. 然而, 传统的基于图像的方法仍然存在一些问题, 例如操作流程差异难以匹配等. 尽管在不同平台上使用相同的 UI 设计, 但操作流程差异仍然存在, 如临时弹窗、控件隐藏和信息分页不同等. 解决这些问题需要更智能的录制回放技术, 通过理解和预测操作流程, 自动跳过冗余步骤或补充必要操作, 从而提高测试脚本适应性和有效性. 基于此, 我们提出了利用大语言模型进行语义匹配的方法, 解决移动应用测试中的跨平台测试脚本的录制冗余和回放问题, 降低测试开发成本. 该方法利用大语言模型, 结合提示信息和任务定义, 对录制回放任务和操作流程差异进行理解. 大语言模型具备海量的文本数据和推理能力, 能够推理当前操作流程差异的类型, 并给出操作建议. 通过语义匹配, 可以跳过冗余的录制步骤或给出待操作控件的坐标, 从而实现更高效的测试脚本录制回放.

具体而言, 该技术通过记录用户操作的控件信息 (位置、类型、文本内容等), 利用图像识别、语义理解和深度学习技术进行控件匹配. 当遇到难以匹配或有流程差异的步骤时, 借助大语言模型的推理能力提供操作建议. 对于录制冗余, 根据大语言模型的推断结果, 系统将自动跳过冗余步骤继续匹配, 完成后续的回放操作. 对于回放冗余, 大语言模型将提供待操作控件的具体坐标, 使得回放系统能够根据该结果继续操作, 实现正确的回放. 最终, 该方法有效完成了测试脚本的录制和回放.

4 基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法 LLMRR

为了提高跨平台测试的效率和准确性, 本文提出了基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法 (LLMRR). 该方法能利用大语言模型的优势, 实现跨平台应用测试脚本的自动录制与回放. 系统输入为用户在不同平台上的操作行为记录, 输出为测试回放的准确结果和相关详细测试报告. 图 2 展示了该方法的主要框架. 方法主要包括 4 个模块: 脚本录制模块、图像匹配模块、文本匹配模块和大模型语义匹配模块.

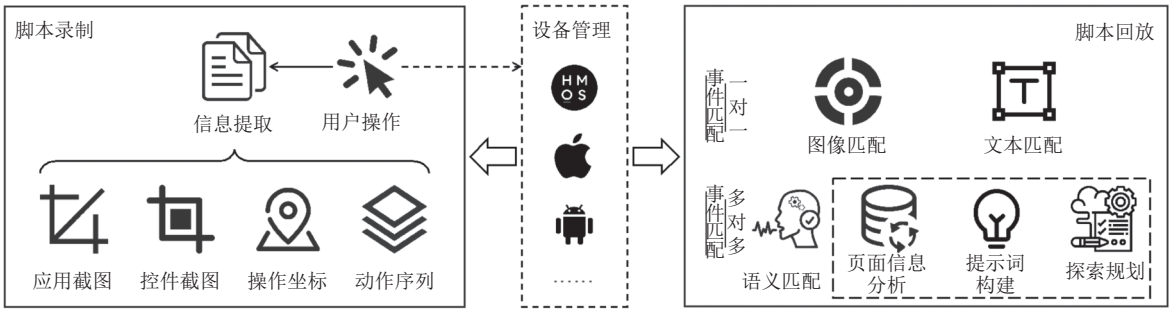

图 2 LLMRR 方法架构示意图

如图 2 所示, 该方法的整体流程包括测试脚本的录制阶段和回放阶段. 在录制阶段, 用户通过系统前端进行操作, 系统实时展示移动设备的屏幕内容. 用户每进行一次操作, 系统都会记录下操作信息, 并保存在测试步骤文件夹中, 同时截图当前页面并保存操作控件的信息. 在录制阶段, 系统所录制信息主要包括应用截图、控件截图、操作坐标、动作序列等信息. 其中, 应用截图是回放阶段信息对比的主要参照, 能够确定操作目标坐标信息; 控件截图可为分析操作类型、在回放设备查找定位目标控件提供对比依据; 操作坐标是回放动作目标的定位依据; 动作序列记录了操作的具体信息及相应参数, 如“输入”动作所附带的输入文本信息等. 在回放阶段, 系统读取录制的测试脚本, 在目标设备上逐步回放用户的操作. LLMRR 首先采用图像匹配, 如未匹配到有效结果, 则进行文本匹配. 我们对现有各平台应用进行人工调研以确定这一顺序, 发现现有各平台应用页面设计以丰富的图片信息或各类图标为主, 这一实践的原因在于图片或图标能够更直观地传达各控件意图, 且相比文字信息来说更为简洁明了. 然而在重要信息部分或图片图表不足以传达有效信息的部分, 开发者会通过文字信息以表达意图. 当图像、文本匹配均失败的情况下, LLMRR 将进入大语言模型匹配模块 (如图 1 所示), 通过与大模型的交互以自动识别是否存在步骤不一致的情况, 从而进行“录制冗余”或“回放冗余”的处理. 大语言模型通过解析当前页面和录制信息语义关联性, 生成操作建议, 处理录制和回放步骤不一致的问题, 从而确保回放过程的连续性和准确性.

4.1 脚本录制

跨平台录制回放算法的核心功能在于确定录制页面和回放页面中的控件位置, 并执行对应的操作. 在录制回放模块中, 我们使用系统接口模拟用户操作. 系统在用户启动录制后, 实时展示移动设备的屏幕内容, 记录用户的每一次操作. 具体包括用户操作的控件信息 (如控件位置、控件类型、操作类型等) 和当前页面的截图. 系统会将这些操作信息保存在一个测试步骤文件夹中. 系统将用户操作的控件信息和当前页面截图保存为单独的文件, 用于后续的匹配和回放. 录制的具体步骤如下.

(1) 初始展示: 系统展示移动设备当前页面, 用户可以根据测试意愿进行相应的操作.

(2) 操作记录: 用户执行操作后, 系统保存对应步骤信息, 并在记录操作位置坐标.

(3) 截图保存: 系统调用对应系统接口工具, 截图当前页面并保存.

(4) 控件分割: 系统基于所保存应用页面截图, 使用 Canny 算法进行控件分割, 并根据所记录测试动作信息确定用户操作控件的位置, 并保留操作控件范围.

上述步骤完成后, 系统允许用户进行下一步操作, 所有操作完成后, 系统将步骤信息汇总为脚本以供回放使

用. 在回放阶段, 系统读取录制的脚本, 逐步在回放设备上重现操作. 若图像匹配和文本匹配均成功, 则执行对应操作; 否则, 系统调用大语言模型语义匹配模块, 通过解析当前页面和录制信息, 生成操作建议并处理不一致的问题.

4.2 图像匹配

在回放阶段, 首先进行图像匹配. 本方法所使用的图像匹配算法主要由模板匹配、SIFT 匹配和图标类型匹配这 3 种算法组成. 模板匹配算法通过直接对比录制和回放页面的模板图像来进行匹配, 可以快速匹配相同或相似的图像区域; 随后, 使用 SIFT 算法对这些区域进行详细特征匹配, 以应对分辨率和光照变化等复杂情况; 最后, 对于那些无法通过前两种算法匹配的图标, 系统使用图标类型匹配算法进行分类匹配, 以确保在 GUI 设计不同的情况下仍能正确识别相同功能的控件.

通过这种多层次的图像匹配策略, 系统能够在不同平台和复杂条件下实现高精度的控件匹配, 确保跨平台测试脚本的准确回放.

4.2.1 模板匹配

模板匹配是一种在计算机视觉中常用的图像处理技术, 用于在大图像中找到与模板图像最相似的部分. 模板匹配的基本原理是通过滑动窗口在大图像中搜索与模板图像相似的区域, 并计算相似度. 在跨平台测试脚本录制回放过程中, 录制设备与回放设备常常是不同的操作系统, 这导致图像分辨率存在差异成为了一种普遍现象. 因此, 本方法仅将模板匹配算法作为辅助工具.

4.2.2 SIFT 匹配

SIFT 算法是一种广泛应用于计算机视觉领域的特征检测和描述方法, 它能够在不同尺度、旋转角度、光照和亮度条件下表现出较高的鲁棒性. 因此, SIFT 算法在本方法的应用场景中能够有效地完成匹配任务. SIFT 算法的基本流程如下.

(1) 检测尺度空间极值

为了找到图像中的潜在特征点, 以便后续的特征点检测和描述, 该算法通过构建高斯金字塔并计算高斯差分图像 (DoG) 来识别图像中可能的关键点位置.

$$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y), D(x,y,\sigma)=[G(x,y,k\sigma)-G(x,y,\sigma)]*I(x,y)=L(x,y,k\sigma)-L(x,y,\sigma),$$

其中, (x, y) 代表图像像素的位置, σ 是尺度参数, k 是尺度空间中相邻层的尺度比. L 是模糊图像, G 是高斯核函数, I 是输入图像, D 是高斯差分图, *表示卷积操作.

接下来通过在 DoG 图像中寻找极值点, 确定关键点位置. 这一步确保在不同尺度上检测到稳定的特征点, 以增强算法的尺度不变性. 对每一个像素, 与其相邻的 26 个像素进行比较 (当前尺度空间的上下两层的 3×3 邻域以及当前层的 3×3 邻域). 找到比邻域所有像素值都大或都小的点, 这些点即为潜在的关键点.

(2) 关键点的精确定位

为了提高关键点的稳定性, 利用 DoG 函数的局部极值来粗略定位关键点后, 还需要通过拟合 DoG 函数的局部极值来精确定位关键点的位置. 具体来说, 利用 Taylor 展开式对 DoG 函数进行拟合, 求解极值点的精确位置, 最后进行子像素插值以获得关键点的精确位置. DOG 函数在尺度空间的 Taylor 展开式为:

$$D(X)=D+\frac{\partial D^{\mathrm{T}}}{\partial X}X+\frac{1}{2}X^{\mathrm{T}}\frac{\partial^2 D}{\partial X^2}X,\quad X=(x,y,\sigma)^{\mathrm{T}},$$

其中, X 是关键点的位置偏移量.

(3) 关键点主方向分配

算法在关键点周围的局部区域内计算梯度方向直方图, 选取主方向作为关键点的主要方向, 使得关键点具有旋转不变性, 增强算法的鲁棒性.

$$\text{梯度幅值}: m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2},$$

$$\text{梯度方向}: \theta(x,y)=\tan^{-1}\left|\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right|,$$

其中, m 是梯度幅值, θ 是梯度方向.

(4) 关键点的特征描述

通过以上步骤, 对于每一个关键点, 都拥有位置、尺度以及方向 3 个信息. 接下来就是为每个关键点生成稳定且具有区分度的特征描述符, 并转换为响应的高维特征向量, 使其不随各种变化而改变.

图 3 展示了一个 SIFT 图像匹配算法在移动应用“微信”中的比对实例. 对于目标控件和待匹配页面截图, 我们分别对其进行灰度化等图像预处理. 由于 SIFT 算法的尺度不变特征, 无须对控件及待匹配页面截图进行尺寸大小的处理. 在预处理完成后, 我们直接调用 SIFT 算法进行特征检测、特征描述生成以及特征点集的构建. 进一步, 将两者的特征点进行有效匹配, 从而计算目标控件在待匹配页面中所匹配控件的坐标.

图 3 SIFT 控件比对实例

基于以上介绍, 结合本方法实际需求, 采用 SIFT 算法用于图像匹配模块的核心算法.

4.2.3 图标类型匹配

模板匹配和 SIFT 算法匹配主要是从几何形状上进行严格匹配. 但是, 有时由于 GUI 设计的不同, 表示同样含义的图标可能在几何形状上并不相同. 本文使用图像分类模型 VGG16, 对移动应用中常见的图标进行分类. 作为一个强大且轻量级的经典图像分类模型, VGG16 模型在诸多任务中表现出色, 且已被相关工作初步应用于控件图标类型的分类, 表现出优异的性能[51]. 考虑到效果与效率之间的平衡, 我们未选取更为先进但训练推理开销都更大的最新模型, 而是以 VGG16 模型作为我们图标类型匹配部分的图像识别模型. VGG16 的基本架构由 16 层网络组成, 包括 13 层卷积层和 3 层全连接层, 具体结构如下.

(1) 卷积层: 使用大小为 3×3 的卷积核, 步长为 1, 填充为 1, 使卷积层的输出尺寸与输入尺寸保持一致.

(2) 池化层: 使用 2×2 的最大池化, 步长为 2, 减少特征图的尺寸, 增加网络的平移不变性.

(3) 全连接层: 使用两个具有 4 096 个神经元的隐藏层和一个具有 1 000 个神经元的输出层进行分类.

为了使模型能够更准确地进行图标分类, 提高了实际应用的效果和精度. 我们在 VGG16 模型应用于移动应用图标分类任务时, 对其全连接层进行了修改, 移除了原有的全连接层, 根据具体的分类任务 (如表 1 所示) 重新进行了设计. 我们所构建的分类集是基于对各平台移动应用中所涉及的图标类型的充分调研所确定, 且我们的数据集尽可能地包含了所列类型的常见样式. 然而, 也存在极少数情况可能会存在超出数据集定义的图标类型或样式, 在这一情况下, LLMRR 将依赖图像模版匹配、文字匹配等模块进一步进行目标控件的检索. 如果确实图像匹配与文本匹配均无法在应用截图中找到目标控件, LLMRR 将会转入大模型语义匹配, 利用大语言模型的理解能力判断是否存在“录制冗余”或“回放冗余”. 然而, 经过对实验结果的充分人工审查, 我们并未在实验中发现此类情况, 也充分说明了各模块的有效性.

模型的全连接层共包含两层. 第 1 层的输入维度为 25 088, 对应于之前预处理的数据大小, 输出维度为 40, 激活函数为 ReLU; 第 2 层的输入维度即第 1 层的输出维度, 输出维度为我们的分类数 14. 由于是多分类问题, 我们使用 categorical_crossentropy 作为损失函数, 并以准确率作为评估指标. 对于单个样本 i 和类别 j, 分类交叉熵损失函数的公式如下:

$$L_i = -\sum_{j=1}^{C} y_{i,j} \log(p_{i,j})$$

其中, L_i 是样本 i 的损失值, C 是类别总数. $y_{i,j}$ 是样本 i 的真实标签, 使用 one-hot 编码表示. 如果样本 i 属于类别 j, 则 $y_{i,j}=1$, 否则 $y_{i,j}=0$. $p_{i,j}$ 是模型对样本 i 预测为类别 j 的概率.

表 1 图标类型匹配模型数据集概况

序号	数据标签	图标样式描述	图标样式示例	数据集规模	可能代表的含义或用途
1	add	加号		856	增加新页面、展开隐藏内容、增加数量等
2	arrow_down	向下箭头		2 155	关注或点击下方内容
3	arrow_left	向左箭头		1 576	返回上一页面、退出当前页面、关注或点击左方内容
4	check_mark	对勾		1 740	选中某项内容、同意某个请求
5	close	叉		2 376	关闭当前控件或页面、不认可某个内容
6	delete	大部分为垃圾桶		1 406	删除
7	menu	多为三点、三横线或者纸张图标		2 566	菜单、展开隐藏内容
8	settings	多为齿轮图标		2 510	设置
9	share	多为转弯箭头和端点明显的三折线		1 812	分享
10	tag	标签		1 388	标签
11	ticket	票、证、券		1 666	优惠券、证件、钱包、票据、钞票
12	search	大部分为放大镜		1 622	搜索框、搜索操作、查找
13	play	多为顶角向右的等腰三角形		1 100	播放、演唱、演奏、继续放映
14	question	问号		1 360	帮助、求助、显示详细信息

针对模型训练, 我们将所收集到的数据按 7:2:1 的比例分割为训练集、验证集以及测试集. 对于模型中所涉及的一些超参数信息, 设置如下: 使用 AdaDelta 算法作为优化器, 初始学习率设置为 0. 001, 激活函数为 ReLU 函数, Dropout 率为 0. 5, 批大小 (Batch Size) 设置为 64. 以上超参数的设置均遵循 VGG16 模型默认设置或参考已有文献 [51] 进行设置. 完成模型搭建后, 我们进行了 40 次迭代的训练. 训练准确率达到了 92.44%, 测试集准确率为 79.25%. 每次预测仅耗时 0.39 s, 具备可用性.

基于以上介绍, 结合本方法实际需求, 采用基于 VGG16 框架的预训练模型用于图标分类算法.

4.3 文本匹配

文本匹配模块依次通过 OCR 技术和 Sentence Transformers 框架实现文本内容的识别与匹配. OCR 技术提取图像中的文本信息, Sentence Transformers 生成的文本向量用于语义匹配, 提高了系统在跨平台和跨语言场景中的适应性.

4.3.1 字符识别

光学字符识别是一种通过计算机技术识别图像中的文字并将其转换成可编辑文本的技术. OCR 技术的应用范围广泛, 能够理解和处理印刷体或手写体文字, 从而实现自动化的文本识别和处理, 为数字化信息的获取和管理提供了便利.

OCR 技术的实现通常包括预处理、文本定位、文本分割、特征提取和分类识别步骤. 预处理阶段对输入的

图像进行预处理, 包括灰度化、二值化、去噪等操作, 以提高后续处理的准确性和效率. 文本定位阶段识别出图像中可能存在的文本区域, 本方法中采用了 Canny 边缘检测的方法. 在文本分割阶段, 采用垂直投影法, 根据投影值分割字符, 得到单个字符或单词的图像. 特征提取阶段, 通过提取图像中字符的特征, 如形状、轮廓、像素分布等, 将图像数据转换成可用于分类的向量形式. 在分类识别阶段, 利用机器学习算法或深度学习模型对提取的特征进行分类识别, 将字符映射成相应的文字.

基于以上介绍, 结合本方法实际需求, 主要对录制控件、回放页面进行 OCR 技术处理, 完成文字匹配和处理的工作. 对于回放界面, OCR 技术不仅保存识别出的文字文本, 还将保存文本所在的位置.

4.3.2 语义匹配

OCR 技术能够对字符串进行严格的匹配. 但是, 由于不同的开发团队以及不同的平台可能会对同一个控件做措辞不同但语义相同的展示. 在文本匹配阶段本方法还使用了基于 Sentence Transformers 框架的 sbert-base-chinese-nli 预训练模型进行语义匹配, 可用于包括中文和英文在内的 13 种语言.

Sentence Transformers 是一个用于生成文本向量表示的 Python 框架, 其主要目的是将句子或段落转换为高维向量空间中的向量, 从而支持各种自然语言处理任务. 该框架建立在 Hugging Face 的 Transformers 库之上, 通过使用预训练的 Transformer 模型, 可以将文本输入映射到具有语义丰富表示的向量空间中. 这些模型在大规模语料库上进行了无监督训练, 学习了丰富的语义信息. 通过微调这些预训练模型, Sentence Transformers 能够根据具体任务和数据集的特点, 生成适用于不同应用场景的文本向量表示.

OCR 技术可以帮助实现严格的文本匹配. 而语义模型匹配则可用来实现字词不同、但语义相同的文本匹配. 如图 4 所示, “分享新鲜事”和“分享一下你的新鲜事吧”这两段文本由于措辞并非完全相同, 导致其无法被 OCR 技术匹配到, 但却因其语义相同, 故可在语义匹配中被匹配到.

图 4 不同平台间同一对象不同描述措辞示例 (左: 鸿蒙; 右: iOS)

4.4 大模型语义匹配

在回放过程中, 当出现与录制过程不一致的情况时, 图像匹配和文本匹配算法往往难以有效应对. 多对多事件映射情况在跨平台测试脚本回放中普遍存在. 这一现象主要是由于不同平台 (包括安卓、iOS、鸿蒙等移动操作系统及其不同版本) 间用户交互指南所不同引起的. 例如, 在 iOS 系统中, 部分应用的菜单栏往往在“用户信息”标

签栏下的菜单中进入, 而在安卓系统中, 菜单栏往往通过主页面右滑呼出或直接在主页面右上角进入. 此外, 除了系统间差异外, 应用本身的设计也会存在测试脚本差异, 如弹出框信息或弹出广告等, 均会造成回放已录制测试脚本时出现流程差异. 因此, 我们通过大语言模型对待测应用对应功能业务逻辑进行充分理解后完成差异化测试步骤的冗余识别, 包括录制冗余与回放冗余. 在本方法中, 我们并未对多对多事件匹配场景进行特定的分类, 而是直接采用通用的大语言模型提示引导进行冗余步骤的识别, 并完成测试的回放. 在处理多对多事件映射情况时, 部分现有方法通过启发式搜索策略或类似技术进行目标控件的查找, 但由于 GUI 控件包含着丰富的与功能业务逻辑紧密关联的语义信息, 且部分 GUI 控件的进入路径也强烈依赖于对功能业务逻辑的充分理解, 因此启发式方法在这样的场景下往往无法取得较好的效果.

为了克服上述难题, 本方法提出了大模型语义匹配模块. 该模块的设计旨在处理两种主要的流程差异: 一种是“多对少”的匹配, 即录制时包含了额外的步骤, 产生录制冗余; 另一种是“少对多”的匹配, 即回放时出现了额外的步骤, 形成回放冗余. 大模型语义匹配模块通过分析当前的录制步骤和回放页面信息, 判断冗余类型, 并生成下一步操作建议. 通过对录制和回放冗余的理解, 该模块能够提供精准的操作指导, 确保回放过程的顺利进行.

首先, 需要分别从录制步骤文件和回放步骤文件中读取当前控件的类型和文本, 组合成提示保存. 提示内容应包括录制步骤和回放页面的信息, 描述当前的操作场景和流程差异, 并请求大语言模型提供操作建议. 录制冗余的情况下, 提示模板为“当前录制步骤无法在回放页面中找到对应的控件, 以下是录制步骤和回放页面的信息. 请根据信息判断是否存在录制冗余, 并给出下一步操作建议”, 同时给出对应的录制和回放页面的信息. 类似地, 在回放冗余的情况下, 提示模板则为“当前回放步骤无法找到录制步骤对应的控件, 以下是录制步骤和回放页面的信息. 请根据信息判断是否存在回放冗余, 并给出下一步操作建议”.

按 ChatGPT 服务器的规范发送请求后, 等待大模型响应, 保存并解析响应内容, 使用正则表达式提取关键信息, 例如“录制冗余”或“回放冗余”, 并根据提取结果决定下一步的操作策略. 当从大模型获取的响应中没有“录制冗余”或“回放冗余”时, 表明未按照请求进行回答, 需重新请求并明确要求回答准确和规范. 若响应中同时包含“录制冗余”和“回放冗余”, 则按照大模型的回答特性选择位置最靠后的关键词作为匹配结果. 若响应中包含“录制冗余”或“回放冗余”, 则根据冗余类型采取相应措施. 若为录制冗余, 跳过当前录制步骤, 直接匹配下一步; 若为回放冗余, 则需要在当前回放页面进行额外操作, 使页面转到匹配的录制页面, ChatGPT 将提供操作控件的位置.

最后, 将从 ChatGPT 回答中解析出的结果传递给后续模块. 传递结果使用具有固定格式的字符串, 例如“数字+空格+数字”, 如“175 354”“−1 −1”等. 其中, “−1 −1”代表录制冗余, 其他则表示回放冗余. 若为回放冗余, 则第 1 个数字代表待操作控件的横坐标, 第 2 个数字代表待操作控件的纵坐标. 通过这种方式, 系统能够有效处理跨平台测试中的流程差异问题, 确保测试脚本的适应性和准确性.

提示词设计是保证大模型有效理解应用状态及待回放脚本语义的关键. 以下是本方法的提示词组织, 如图 5 所示.

(1) 身份设定: 将 ChatGPT 设定为软件测试工程师, 并对其任务进行简要介绍.

(2) 概念介绍: 说明录制冗余和回放冗余的基本概念.

(3) 实例详解: 提供录制冗余和回放冗余的实例, 详细说明常见情境和特征.

(4) 描述差异: 给出当前回放过程中流程差异的具体信息, 并请求 ChatGPT 提供操作建议.

(5) 格式要求: 对 ChatGPT 的回答格式做出明确要求, 确保包含录制冗余和回放冗余等关键词.

上述提示词各部分分别促进了大语言模型对于测试场景及测试脚本的理解. 身份设定部分通过“角色扮演”的方式提升了大模型对与软件测试场景上下文的理解能力; 概念介绍部分以概要的方式告知大语言模型测试脚本录制回放的基本任务概念与流程; 实例详解通过“one-shot”的方式充分调动大语言模型预训练知识与上下文理解能力, 使得其对于任务模式有更清晰的认知; 描述差异部分引导大语言模型识别在具体回放过程中出现的流程差异具体信息, 从而引导完成面向差异的测试回放; 格式要求部分规定了大语言模型的输入规范, 以便后续转化为操作待测应用的可执行指令.

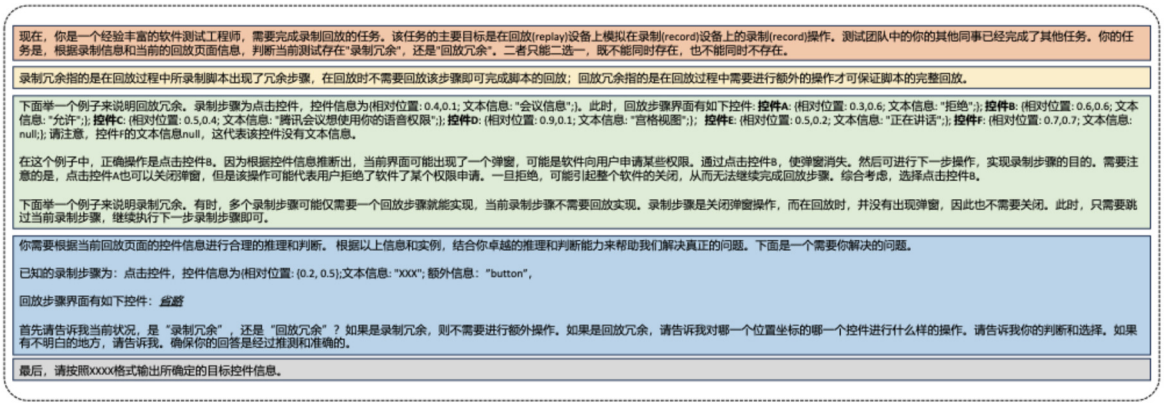

图 5　提示词样例

大模型语义匹配模块在录制和回放步骤不对应的情况下, 利用 ChatGPT-4 模型提供智能操作建议. 通过预处理录制和回放信息, 生成提示输入模型, 获得操作建议并执行相应的回放操作. 该模块能够处理复杂的流程差异, 提高系统的鲁棒性和智能化水平.

5　实验分析

为了评估 LLMRR 方法的有效性, 我们主要针对以下 3 个研究问题开展实验评估.

- RQ1: LLMRR 在整体回放效果上与现有最先进基线方法相比表现如何?
- RQ2: LLMRR 的图文匹配模块在回放效果上与现有最先进基线方法相比表现如何?
- RQ3: LLMRR 的大模型语义匹配模块在回放效果上表现如何?

5.1　实验设置

为评估 LLMRR 有效性, 我们构建了包含覆盖 20 个应用共 100 个测试脚本 (每个应用 5 个测试脚本) 的数据集来完成实证实验, 这些测试脚本均在安卓、iOS、鸿蒙这 3 个平台中进行录制, 以供后续在另外两个平台上进行回放实验. 在待测应用选择的过程中, 我们主要参考了现有 GUI 测试脚本录制回放的工作, 如 LIRAT[51] 等. 在现有工作所使用的待测应用中, 我们进一步验证其可用性, 以剔除因缺乏维护而无法安装或运行的应用. 最终, 我们确定了如表 2 所示的 20 个待测应用. 这 20 个待测应用覆盖了系统、购物、工具、财务、媒体、运动、开发等多种类别, 以体现本方法在评估中效果上的泛化性. 测试脚本的开发由 3 名经验丰富的高年级软件工程专业研究生完成, 所有参与学生均有着至少 3 年的实际工业环境移动应用测试脚本开发与维护工作经验, 足以保证本实验中所涉及测试脚本与实际工业环境中测试脚本一致. 此外, 我们还要求所有测试脚本必须针对待测应用核心功能业务进行开发, 而不是边缘功能点, 且每个应用对应的 5 个测试脚本需覆盖待测应用的不同功能, 从而保证实验数据集的广泛性. 所编写的测试脚本在 iOS、安卓、鸿蒙平台中共分别包含 1 380、1 383、1 397 个测试动作, 测试脚本中动作序列长度为 10–17 不等, 平均每个脚本步骤数为 13.87. iOS 与安卓、iOS 与鸿蒙、安卓与鸿蒙这 3 个平台对中分别有 76、75、74 个脚本包含多对多事件映射的情况.

对于消融实验, 我们主要对传统图文匹配方法与大语言模型语义匹配方法进行消融. 传统图文匹配方法与传统测试脚本录制回放中所采用技术类似, 但我们分别在图像、文本等内容的匹配上进行了创新的设计, 从而提升无步骤差异情况下的回放成功率. 大语言模型语义匹配是本方法的核心创新点, 有效应对了脚本跨平台回放时存在的步骤差异情况. 因此, 实验中 RQ2 与 RQ3 分别对这两部分的效果与基线方法进行对比分析.

在实验评估中, 主要选择两种最先进录制回放技术作为所介绍方法的基线方法进行有效性对比: LIRAT[51]、MAPIT[15] 以及 AppTestMigrator[52] 这 3 种方法均是当前最先进跨平台测试脚本录制回放方法, 其代表性已得到认可. LIRAT 主要流程与 LLMRR 相似, 即将用户的操作录制下来, 并保存为特定的测试脚本, 然后在不同的平台上

进行回放. MAPIT 的基本思路是在一个平台上通过规范的语言编写对应的测试脚本, 然后利用它提供的工具, 在源平台上运行一遍脚本, 在运行的过程中, 生成中间测试文件, 主要存储操作的组件相关信息以及屏幕内其他组件的信息. 所生成的中间测试文件与平台无关, 所以通过 MAPIT 和中间测试文件, 可以在任意平台上进行回放, 达到跨平台回放的目的. AppTestMigrator 是一种具有典型代表意义的跨应用测试迁移工具, 其可在相似应用中迁移面向待测应用共享功能的测试脚本. 实验中所涉及 4 种方法均运行于同样的脚本数据集中, 以保证实验结果的可比较性. 因本方法考虑到跨平台脚本回放步骤的不一致性, 所涉及实验脚本与 LIRAT、MAPIT 及 AppTestMigrator 方法中评估数据集不相同; 由于上述 3 种基线方法未考虑跨平台回放不一致性, 实验结果所展示的回放成功率也与原论文存在波动.

表 2　实验评估应用

序号	名称	类别	序号	名称	类别
1	AdGuard	系统	11	Linphone	通信
2	BingSearch	工具	12	MatomoMob	开发
3	Booking	购物	13	McDonald	购物
4	Evernote	工具	14	Monkey	开发
5	Investing	财务	15	OpenHAB	网络
6	Jamendo	媒体	16	OsmAnd	地图
7	Keep	运动	17	QQ Music	媒体
8	KFC	购物	18	Taobao	购物
9	Kindle	工具	19	VLC	媒体
10	Kiwix	网络	20	WikiPedia	网络

对于实验过程中所使用的大模型, 我们采用了 GPT-4o 模型, 并将参数 temperature 设置为 0, 以保证大模型产生确定且一致的输出. 在实验中, 我们也统计了大模型 Token 开销. 对于方法 LLMRR 而言, 平均回放一个脚本 Token 的开销为 39 154 个. 我们也将使用的提示词模版公开在 GitLink 的在线仓库中, 以供后续研究参考 (https://gitlink.org.cn/yusc/LLMRR). 为保障在实验过程中顺利调用大模型能力, 我们遵循 OpenAI 所设置的调用限制, 利用多账号协作完成了本实验.

5.2 整体结果与分析

在实验结果中, 表 3 展示了 LLMRR、LIRAT、 MAPIT 和 AppTestMigrator 这 4 种方法在不同平台间录制回放的成功和失败情况. 注意在本文后续的表格中, I 表示 iOS, A 表示安卓, H 表示鸿蒙, 例如 I2A 表示从 iOS 系统迁移至安卓系统.

表 3　整体录制回放实验结果

评估指标	对比方法	I2A	A2I	I2H	H2I	A2H	H2A
成功脚本数	LLMRR	**65**	**68**	**62**	**61**	**65**	**56**
	LIRAT	5	6	1	6	8	11
	MAPIT	3	5	1	1	5	7
	AppTestMigrator	18	19	18	14	16	18
失败脚本数	LLMRR	**35**	**32**	**38**	**39**	**35**	**44**
	LIRAT	95	94	99	94	92	89
	MAPIT	97	95	99	99	95	93
	AppTestMigrator	82	81	82	86	84	82
回放成功率 (%)	LLMRR	**65.0**	**68.0**	**62.0**	**61.0**	**65.0**	**56.0**
	LIRAT	5.0	6.0	1.0	6.0	8.0	11.0
	MAPIT	3.0	5.0	1.0	1.0	5.0	7.0
	AppTestMigrator	18.0	19.0	18.0	14.0	16.0	18.0

从表中可以看出, LLMRR 方法在所有跨平台迁移路径中均表现出明显优势. 例如, 在 iOS 到安卓 (I2A) 的迁

移中, LLMRR 成功完整回放了 65 个脚本, 而 LIRAT 和 MAPIT 分别仅成功回放 5 个和 3 个脚本, AppTestMigrator 成功回放了 18 个脚本. 这一趋势在其他迁移路径上也得到了验证. 在鸿蒙到 iOS (H2I) 和鸿蒙到安卓 (H2A) 的迁移中, LLMRR 方法成功回放了 61 和 56 个脚本, 而基线方法在这些路径中的成功回放数远远不及, 最高仅为 11 个脚本. LLMRR 方法在各平台间迁移中虽仍有一定数量的失败脚本, 但相较于基线方法, 其数量明显较少. 例如, 在 I2A 迁移中, LLMRR 的失败脚本数为 35, 而 LIRAT 和 MAPIT 分别为 95 和 97. AppTestMigrator 相比起 LIRAT 和 MAPIT 成功率较高, 其可能原因在于其设计目标是在不同应用之间迁移测试脚本, 因此对于多对多事件映射有一定的处理, 但面对较复杂的多对多事件映射, LLMRR 依然具有较明显的优势.

在回放成功率方面, LLMRR 方法在 I2A、A2I、I2H、H2I 等迁移路径上, 回放成功率均在 60% 以上, 最高达到 68%. 相比之下, 基线方法的成功率显著较低, 最高仅为 19%. 尤其是在鸿蒙相关的迁移路径 (如 H2A、A2H) 中, LLMRR 的成功率分别为 56.0% 和 65.0%, 而基线方法在这些路径上的成功率更是低至 5.0%–18.0%. 对于图文匹配的消融结果, 我们方法的成功率均超过了 90%, 而大语言模型语义匹配消融结果中, 成功率也均超过了 80%. 然而, 上述结果的实验结果呈现粒度分别为测试事件与多对多事件组, 因此呈现出较好的效果, 也相比基线方法有着较好的改进. 然而, 针对整体实验结果中, 我们方法的回放成功率均大约为 60%. 值得注意的是, 这一研究问题下的统计视角为整体脚本的回放成功率. 也就是说, 在一个脚本的回放过程中, 只要存在一个步骤的失败, 我们即认为该回放失败, 因此, 从最终结果来看, 完整成功回放的脚本次数可能相较步骤粒度的回放成功率略低.

特别地, 可以发现 LLMRR 方法在处理鸿蒙系统与其他平台之间的迁移时, 表现尤为突出. 例如, 在鸿蒙到 iOS (H2I) 和鸿蒙到安卓 (H2A) 的迁移路径中, LLMRR 方法的成功脚本数和回放成功率均显著高于基线方法, 显示出 LLMRR 方法在应对鸿蒙系统独特的分布式架构特性方面具有较强的适应能力.

本文的核心贡献在于利用大语言模型的待测应用业务场景逻辑理解能力, 实现了测试脚本跨平台录制回放中的多对多事件映射问题, 并取得了较好的效果. 现有方法, 如 AppTestMigrator, 虽然在多对多事件映射问题上作了初步的尝试, 但由于其依然缺乏 GUI 控件所包含的丰富的与功能业务逻辑紧密关联的语义信息的理解, 也缺乏领域知识以触发特定路径中的目标 GUI 控件, 从而导致其在大部分多对多事件映射情况下的回放失败. 因此, 我们认为, 本方法所提出的多对多事件映射问题结局方案与传统方法有着本质上的区别, 我们重点关注了多对多事件映射情况中所包含的业务逻辑理解, 从而实现了对应的录制回放, 与现有方法相比, 有着本质上的方法性提升.

总结来看, LLMRR 方法在不同平台间的录制回放实验中表现出明显的优势, 特别是在回放成功率和脚本兼容性方面. 通过引入大模型语义匹配模块, LLMRR 能够更好地处理不同平台间的操作流程和控件差异, 提高了测试脚本的适应性和通用性. 这一结果表明, LLMRR 方法在跨平台移动应用测试中具有广阔的应用前景, 能够有效提升测试效率和准确性, 为移动应用的开发和质量保障提供了有力支持.

5.3 图文匹配消融结果与分析

为进一步评估 LLMRR 各模块方法有效性, 我们在实验评估中还开展了消融实验评估. 表 4 展示了图文匹配录制回放实验结果, 针对不同平台之间的迁移, 通过对比 LLMRR (我们用 LLMRR-it, 即 LLMRR-image-and-text 来表示 LLMRR 的图像文本匹配部分) 方法与基线方法的对比表现, 分析了图文匹配模块的效果. 在本节的评估中, 我们仅关注测试脚本中可进行一对一映射的测试行为 (对于可进行一对一映射的测试行为数, iOS 与安卓平台间为 1 209, iOS 与鸿蒙平台间为 1 229, 安卓与鸿蒙平台间为 1 232), 从而展示 LLMRR 图文匹配模块 (LLMRR-it) 的有效性. 即, 我们将所有脚本中不满足一对一映射的事件人为剔除以避免多对多事件映射情况对图文匹配消融实验的影响.

在成功步骤数方面, LLMRR-it 在所有迁移路径中均具有一定的优势. 例如, 在鸿蒙到安卓 (H2A) 的迁移路径中, LLMRR-it 方法成功回放了 1 180 个步骤. 这表明, LLMRR-it 方法在处理鸿蒙系统的跨平台迁移时, 能够更准确地识别和匹配不同平台的控件, 从而实现更高的成功步骤数. 与此同时, 在失败步骤数方面, LLMRR-it 方法在鸿蒙相关的迁移路径中也表现出较低的失败率. 例如, 在鸿蒙到安卓 (H2A) 的迁移路径中, LLMRR-it 方法仅有 52 个失败步骤, 而基线方法分别为 95 个、153 个和 72 个失败步骤. 这显示出 LLMRR-it 方法在鸿蒙系统的跨平台操

作流程中的稳定性和准确性, 能够有效减少失败步骤数, 提高整体回放效果.

表 4 图文匹配录制回放实验结果

评估指标	对比方法	I2A	A2I	I2H	H2I	A2H	H2A
成功步骤数	LLMRR-it	**1 171**	**1 170**	**1 193**	**1 181**	**1 188**	**1 180**
	LIRAT	1 108	1 119	1 126	1 133	1 141	1 137
	MAPIT	1 058	1 058	1 048	1 090	1 065	1 079
	AppTestMigrator	1 123	1 134	1 150	1 144	1 151	1 160
失败步骤数	LLMRR-it	**38**	**39**	**36**	**48**	**44**	**52**
	LIRAT	101	90	103	96	91	95
	MAPIT	151	151	181	139	167	153
	AppTestMigrator	86	75	79	85	81	72
回放成功率 (%)	LLMRR-it	**96.9**	**96.8**	**97.1**	**96.1**	**96.4**	**95.8**
	LIRAT	91.6	92.6	91.6	92.2	92.6	92.3
	MAPIT	87.5	87.5	85.3	88.7	86.4	87.6
	AppTestMigrator	92.9	93.8	93.6	93.1	93.4	94.2
完整脚本回放数 (仅包含一对一回放)	LLMRR-it	**75**	**74**	**72**	**65**	**72**	**56**
	LIRAT	22	23	5	41	25	40
	MAPIT	9	20	5	19	11	17
	AppTestMigrator	31	42	40	39	35	42

回放成功率是评估跨平台录制回放方法的重要指标之一. 表 4 中数据显示, LLMRR-it 方法在各个迁移路径中的回放成功率均高于基线方法. 例如, 在 I2A 迁移路径中, LLMRR-it 方法的回放成功率为 96.9%, 而基线方法分别为 91.6%、87.5% 和 92.9%.

完整脚本回放数也是评估跨平台测试方法的重要指标. LLMRR-it 方法在各个迁移路径中的完整脚本回放数均明显高于基线方法. 例如, 在 I2A 迁移路径中, LLMRR-it 方法成功回放了 75 个完整脚本, 而基线方法分别为 22 个、9 个和 31 个 在 H2I 迁移路径中, LLMRR-it 方法成功回放了 65 个完整脚本, 而基线方法分别为 41 个、19 个和 39 个.

对于所有成功回放的 7 083 个步骤, 其中有 5 281 个步骤 (74.56%) 是通过图像匹配得到正确结果, 其余则是通过文本匹配得到正确结果. 对于失败的步骤, 则是其图像匹配与文本匹配均未得到正确结果, 从而导致回放失败. 我们进一步分析了 LLMRR-it 相对基线方法中所采用图像或文本匹配部分的优势. 针对图像匹配部分, 基线方法往往主要采用基于图像特征提取的比对, 即 LLMRR-it 中所采用的 SIFT 算法或类似技术. 然而, 此类技术要求能够在目标控件中采集到足够的特征点, 而部分图标控件中由于其简约的设计, 无法采集到足够用于匹配的特征点集, 从而导致匹配失败. 在 LLMRR-it 中, 我们进一步采用图标类型进行匹配, 从而提升匹配成功率. 对于文本匹配部分, 现有方法基本采用文本特征提取以匹配对应文本信息, 但此类算法往往忽略了语义信息, 尤其是在移动应用场景下的相似语义. 例如, 在某些场景中“提交”与“完成”具有类似的语义. 即完成表单信息的填写后将其由前端页面发送至服务端处理. 而此类情况的语义相似性无法被基线方法中传统文本匹配算法识别. LLMRR-it 创新地采用了文本语义匹配模型, 以增强在此类场景下的文本匹配效果. 简单总结来说, LLMRR-it 所采用的图像文本匹配方法设计弥补了基线方法中相应设计的不足, 有效提升成功率.

综上所述, LLMRR-it 方法在跨平台移动应用测试中具有显著优势, LLMRR-it 方法能够更准确地识别和匹配不同平台的控件, 显著提高了成功步骤数和回放成功率, 同时减少了失败步骤数, 提升了完整脚本回放数. 这些结果表明, LLMRR-it 方法在跨平台移动应用测试中具有广阔的应用前景, 能够为开发者提供高效、准确的测试工具, 从而提升移动应用的质量和用户体验.

5.4 大模型语义匹配消融结果与分析

大模型语义匹配是 LLMRR 方法的核心创新点, 也是 LLMRR 实现多对多事件映射的重要方法设计 (我们用

LLMRR-llm 来表示 LLMRR 的大语言模型语义匹配部分). 表 5 展示了语义匹配录制回放实验的具体结果, 通过对比 3 种不同方法对于所有测试脚本中所有步骤在不同迁移途径中的表现, 进一步分析了语义匹配模块在跨平台录制回放中的有效性. 其中, “不适用匹配数”指采用图文匹配完成匹配的步骤数量.

表 5　大模型语义匹配录制回放实验结果

评估指标	对比方法	I2A	A2I	I2H	H2I	A2H	H2A
成功匹配数	LLMRR-llm	**63**	**64**	**63**	**66**	**64**	**60**
	LIRAT	0	0	0	0	0	0
	MAPIT	0	0	0	0	0	0
	AppTestMigrator	13	15	13	10	12	10
失败匹配数	LLMRR-llm	**13**	**12**	**12**	**9**	**10**	**14**
	LIRAT	76	76	75	75	74	74
	MAPIT	76	76	75	75	74	74
	AppTestMigrator	63	61	62	65	62	64
不适用匹配数	ALL	**24**	**24**	**25**	**25**	**26**	**26**
回放成功率 (%)	LLMRR-llm	**82.9**	**84.2**	**84.0**	**88.0**	**86.5**	**81.1**
	LIRAT	—	—	—	—	—	—
	MAPIT	—	—	—	—	—	—
	AppTestMigrator	17.1	19.7	17.3	13.3	16.2	13.5

在语义匹配方面, 相比其他两种方法, LLMRR-llm 方法在所有迁移路径中均表现出压倒性优势. 例如, 在鸿蒙到 iOS(H2I) 的迁移路径中, 在 76 个步骤中 LLMRR-llm 方法一共成功匹配了 66 个步骤, 回放成功率达到 88%, 而 LIRAT 和 MAPIT 均为 0 个, 完全无法进行有效的语义匹配. 通过大语言模型语义匹配的多对多事件映射, 其失败案例均由大模型的幻觉引起, 大模型往往会产生一个自认为正确但实际回放无法执行的结果. 这表明 LLMRR-llm 方法在处理测试脚本的跨平台迁移时, 能够更有效地进行语义匹配, 确保有效的匹配. 而 AppTestMigrator 在多对多事件映射中相比较 LIRAT 和 MAPIT 表现出了一定的优势, 其主要设计目的为在不同待测应用中迁移相似功能的测试脚本, 因此能够一定程度上具有多对多事件映射的能力. 但由于其未针对多对多事件映射问题涉及特定策略, 依然在测试脚本录制回放中稍逊于 LLMRR-llm 的回放准确率.

上述分析表明, LLMRR-llm 方法在跨平台移动应用测试中展现出卓越的性能. 通过集成大模型语义匹配模块, LLMRR-llm 方法能够更精确地进行语义匹配, 这一点是 LIRAT 和 MAPIT 方法所无法实现的. LLMRR-llm 方法的大模型语义匹配模块显著提高了成功匹配数和回放成功率, 同时大幅减少了失败匹配数和不适用匹配数. LLMRR 方法为开发者提供了一种高效且精准的测试工具, 能够有效地完成语义匹配的任务, 在未来的跨平台移动应用测试中具有巨大的应用潜力.

6 讨　论

6.1 优势分析

LLMRR 通过引入大模型语义匹配方法进行跨平台移动应用测试脚本录制回放, 在应对不同平台的操作流程差异时展现了显著优势. 首先, LLMRR 方法结合了模板匹配、SIFT 匹配和图标类型匹配等多种算法, 这种多层次的图像匹配策略能够在不同平台和复杂条件下实现高精度的控件匹配, 从而提高了跨平台测试脚本的准确回放. 其次, LLMRR 方法利用了大语言模型的强大语义理解和推理能力, 可以准确分析测试脚本业务逻辑, 通过解析录制步骤和回放页面的信息来生成操作建议, 有效应对录制冗余和回放冗余问题, 确保回放过程的连续性和准确性. 另外, 通过大语言模型可以与应用回放时实时状态进行有效对比, 实现冗余测试行为丢弃与隐藏测试行为补足, 从而提高测试脚本的适应性和有效性. 第三, 与传统的“一对一”映射不同, 引入大模型后的 LLMRR 方法实现了多对多事件映射, 即若干录制步骤可能对应多个不等数量的回放步骤. 这种多对多映射模式能够更灵活地处理不同平台间的流程差异和控件差异, 显著提高了测试脚本的通用性和适应性.

6.2 局限性分析

尽管 LLMRR 方法在跨平台移动应用测试中展现了诸多优势, 但它也存在一定的局限性和不足之处. 首先, 图像文本匹配时存在算法的不精准情况. 尽管 LLMRR 方法结合了多种图像匹配算法, 但在实际应用中仍可能存在匹配不精准的情况, 尤其是在图像质量较低或界面设计差异较大时, 匹配效果可能会受到影响. OCR 技术在文本识别过程中可能会出现识别错误或遗漏, 而 Sentence Transformers 模型在进行文本匹配时, 也可能因为训练数据的局限性导致匹配不准确. 其次, 大语言模型在生成操作建议时, 可能会出现幻觉问题, 即生成的内容与实际情况不符. 这可能导致业务逻辑判断不准确, 影响测试结果的可靠性. 大语言模型的判断依赖于训练数据和提示信息, 如果提示信息不准确或不完整, 模型可能会产生错误的判断, 影响回放效果. 在实验过程中, 为保证实验结果可靠性, 我们未进行人工干预, 而是直接记录最真实的实验结果. 第三, 大模型对于测试脚本文本的分析能力足够鲁棒, 但对于应用页面图像信息的捕获存在障碍. 不同平台和应用的界面的设计差异、应用页面的动态变化 (如动画效果、弹出窗口等)、部分控件的视觉细节差异微小难以识别等图像信息的多样性和复杂性, 增加了图像匹配的难度, 使得大模型无法准确捕获和处理这些信息, 进而影响回放的准确性和完整性.

6.3 有效性威胁分析

在软件测试的录制回放过程中, 特别是大模型语义匹配的跨平台移动应用测试脚本录制回放中, 为了保证系统能够在不同平台和不同测试条件下都能够稳定、准确地运行, 本研究对可能出现的有效性威胁进行处理, 尽可能降低其对于本研究试验结果的影响. 本节将详细探讨所提出方法的有效性威胁分析.

首先, 外部有效性的主要威胁来源于实验应用的收集. 在本研究中, 我们收集了有限的应用以供实验验证. 在应用收集过程中, 首先参考了已有相关研究的实验中所选取应用, 并排除其中已过时的应用. 其次, 我们所收集的应用包含开源应用和商业应用, 并覆盖了不同的应用类型, 从而保证了方法泛化能力的验证, 因此, 我们认为上述做法已尽可能消除了实验的外部有效性威胁.

其次, 内部有效性的主要威胁来源于图像匹配算法和文本匹配算法中相关超参数的设置. 为缓解这一威胁, 我们将所有超参数设置为其默认值, 或参考已有相关研究中相同场景下对应模型超参数设置. 对于没有可供参考的超参数设置, 我们进行小规模实验以选择最优的超参数设置. 通过以上做法, 我们认为已经最大程度地消除了实验的内部有效性威胁.

第三, 我们所选取的基线方法可能会对实验结果造成有效性威胁, 但所选取的基线研究均为当前最先进且具有代表性的移动应用测试脚本录制回放方法, 与其相比能够充分表现本文所提出方法的有效性.

6.4 性能开销分析

由于 LLMRR 方法采用了图文匹配和大模型语义匹配的多层次策略, 所以在测试时间和效率上相较于现有的先进方法表现出了突出优势. 通过实验得到, LLMRR 在图文匹配上的平均每个步骤处理时间为 2.5 s, 相比之下, LIRAT 方法为 4.3 s, MAPIT 方法为 5.6 s, LLMRR 明显更为高效. 图文匹配的快速响应不仅提高了整体测试流程的流畅性, 还减少了测试的等待时间. 此外, 在多对多事件映射时, LLMRR 每个多对多事件映射匹配对的匹配时间为 7.5 s (包含所有步骤, 如 API 调用等), 若按最终回放步骤数计算平均时间, 整体平均每步仅需 3.9 s. 由此可见, LLMRR 方法在保持高效和准确的同时, 缩短了每个测试步骤的处理时间, 提高了整体测试效率. 我们相信, 随着 LLMRR 方法的进一步优化和完善, 其在未来将继续保持在跨平台移动应用测试中的领先地位, 提供更高效和智能的测试解决方案.

6.5 未来工作展望

本文相关实验有力证明了基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法在解决跨平台应用测试中流程差异问题方面的显著优势. 然而, 通过上文局限性分析, 我们认为本方法还可以进行改进.

首先, 本文使用的大模型为 ChatGPT-4, 是目前全球领先的大语言模型. 未来, 随着人工智能的持续发展, 很可能会有更多更先进的大模型问世. 本方法可以通过使用更先进的模型, 更准确地理解用户的意图, 并生成更为精确

和一致的响应. 此外, 由于大模型的局限性, 我们还可以进一步融合大模型的场景理解能力和知识图谱等技术的知识归纳能力, 不断提升方法对于应用状态及测试脚本业务逻辑的理解.

其次, 可引入多模态融合技术. 目前的方法主要依赖于图像和文本信息进行匹配和判断. 未来, 多模态融合技术的发展将使得系统能够结合更多类型的数据, 如语音、手势等, 以实现更全面的跨平台测试. 通过整合多种模态的数据, 系统将能够更好地理解和模拟用户的操作, 提高测试的覆盖面和准确性.

第三, 本文采用了若干图像匹配与文本匹配算法. 这些算法在我们的任务中表现出了较好的效果, 也在效率开销方面与效果达到了一定的平衡. 在未来工作中, 我们也将尝试更先进的方法与技术, 例如深度学习模型等, 以处理本文中涉及文本与图像匹配的部分, 从而提升本文匹配的有效性.

7 总 结

复杂的操作流程和多样的设备差异使得移动应用测试脚本跨平台录制回放测试面临巨大的挑战. 本文提出了一种基于大模型语义匹配的跨平台移动应用测试脚本录制回放方法 (LLMRR), 提供了针对不同平台操作流程差异带来的问题的解决方案. 该方法提供了一个基于大模型语义匹配的跨平台测试脚本录制回放框架, 能够适用于不同平台间的脚本迁移. 通过综合使用图像匹配、文本匹配和大模型语义匹配, LLMRR 实现了高效、准确的跨平台测试脚本录制和回放. 与此同时, 本文针对国产鸿蒙应用生态平台进行了测试脚本录制回放方法的探索, 并通过大规模实证评估, 表明了所提出方法在回放成功率上与现有最先进基线方法对比有较大提升. 实验结果显示, LLMRR 方法在鸿蒙系统的跨平台测试中, 成功匹配数和回放成功率显著高于传统的 LIRAT 和 MAPIT 方法, 失败匹配数和不适用匹配数明显减少. 未来, 通过进一步优化大模型语义匹配模块, LLMRR 方法有望成为跨平台移动应用测试中更强大、更高效的工具, 帮助开发者提升应用质量和用户体验.

References:

[1] Mei H, Cao DG, Xie T. Ubiquitous operating system: Toward the blue ocean of human-cyber-physical ternary ubiquitous computing. Bulletin of Chinese Academy of Sciences, 2022, 37(1): 30–37 (in Chinese with English abstract). [doi: 10.16418/j.issn.1000-3045.20211117009]

[2] Li XD, Wang J, Zhan NJ. Greeting the era of human-cyber-physical ternary ubiquitous computing and boosting innovation and supply capacity of key software and technologies. Science and Technology Foresight, 2023, 2(1): 5–6 (in Chinese with English abstract).

[3] Li C, Jiang YY, Xu C. GUI event-based record and replay technologies for Android Apps: A survey. Ruan Jian Xue Bao/Journal of Software, 2022, 33(5): 1612–1634 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6551.htm [doi: 10.13328/j.cnki.jos.006551]

[4] Zhang GL, Wan Y. Overview of mobile application GUI testing technology. Modern Computer, 2019(10): 44–48 (in Chinese with English abstract). [doi: 10.3969/j.issn.1007-1423.2019.10.010]

[5] Zhang SK, Li YC, Lei HW, Jiang P, Li D, Guo Y, Chen XQ. GUI fuzzing framework for mobile Apps based on multi-modal representation. Ruan Jian Xue Bao/Journal of Software, 2024, 35(7): 3162–3179 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7106.htm [doi: 10.13328/j.cnki.jos.007106]

[6] Yeh T, Chang TH, Miller RC. Sikuli: Using GUI screenshots for search and automation. In: Proc. of the 22nd Annual ACM Symp. on User Interface Software and Technology. Victoria: ACM, 2009. 183–192. [doi: 10.1145/1622176.1622213]

[7] Li C, Jiang YY, Xu C. Cross-device record and replay for Android Apps. In: Proc. of the 30th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Singapore: ACM, 2022. 395–407. [doi: 10.1145/3540250.3549083]

[8] Havranek M, Bernal-Cárdenas C, Cooper N, Chaparro O, Poshyvanyk D, Moran K. V2S: A tool for translating video recordings of mobile App usages into replayable scenarios. In: Proc. of the 43rd Int'l Conf. on Software Engineering: Companion Proc. (ICSE-Companion). Madrid: IEEE, 2021. 65–68. [doi: 10.1109/ICSE-Companion52605.2021.00037]

[9] Bernal-Cárdenas C, Cooper N, Havranek M, Moran K, Chaparro O, Poshyvanyk D, Marcus A. Translating video recordings of complex mobile App UI gestures into replayable scenarios. IEEE Trans. on Software Engineering, 2023, 49(4): 1782–1803. [doi: 10.1109/TSE.2022.3192279]

[10] Fazzini M, Freitas ENDA, Choudhary SR, Orso A. Barista: A technique for recording, encoding, and running platform independent Android tests. In: Proc. of the 2017 IEEE Int'l Conf. on Software Testing, Verification and Validation (ICST). Tokyo: IEEE, 2017.

149–160. [doi: 10.1109/ICST.2017.21]

[11] Halpern M, Zhu YH, Peri R, Reddi VJ. Mosaic: Cross-platform user-interaction record and replay for the fragmented Android ecosystem. In: Proc. of the 2015 IEEE Int'l Symp. on Performance Analysis of Systems and Software. Philadelphia: IEEE, 2015. 215–224. [doi: 10.1109/ISPASS.2015.7095807]

[12] Hu YJ, Azim T, Neamtiu I. Versatile yet lightweight record-and-replay for Android. In: Proc. of the 2015 ACM SIGPLAN Int'l Conf. on Object-Oriented Programming, Systems, Languages, and Applications. Pittsburgh: ACM, 2015. 349–366. [doi: 10.1145/2814270.2814320]

[13] Liu CH, Lu CY, Cheng SJ, Chang KY, Hsiao YC, Chu WM. Capture-replay testing for Android applications. In: Proc. of the 2014 Int'l Symp. on Computer, Consumer and Control. Taichung: IEEE, 2014. 1129–1132. [doi: 10.1109/IS3C.2014.293]

[14] Zheng W, Tang H, Chen X, Zhang MQ, Xia X. State-of-the-art survey of compatibility test for Android mobile application. Journal of Computer Research and Development, 2022, 59(6): 1370–1387 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.20210105]

[15] Talebipour S, Zhao YX, Dojcilović L, Li CG, Medvidović N. UI test migration across mobile platforms. In: Proc. of the 36th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Melbourne: IEEE, 2021. 756–767. [doi: 10.1109/ASE51524.2021.9678643]

[16] Zhang SK, Wu LN, Li YC, Zhang ZQ, Lei HW, Li D, Guo Y, Chen XQ. ReSPlay: Improving cross-platform record-and-replay with GUI sequence matching. In: Proc. of the 34th Int'l Symp. on Software Reliability Engineering (ISSRE). Florence: IEEE, 2023. 439–450. [doi: 10.1109/ISSRE59848.2023.00056]

[17] Alégroth E, Nass M, Olsson HH. JAutomate: A tool for system- and acceptance-test automation. In: Proc. of the 6th Int'l Conf. on Software Testing, Verification and Validation. Luxembourg: IEEE, 2013. 439–446. [doi: 10.1109/ICST.2013.61]

[18] Ji RH, Zhu TW, Zhu XQ, Chen CY, Pan MX, Zhang T. Vision-based widget mapping for test migration across mobile platforms: Are we there yet? In: Proc. of the 38th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Luxembourg: IEEE, 2023. 1416–1428. [doi: 10.1109/ASE56229.2023.00068]

[19] Liang JY, Wang SN, Deng XB, Liu YP. RIDA: Cross-App record and replay for Android. In: Proc. of the 2023 IEEE Conf. on Software Testing, Verification and Validation (ICST). Dublin: IEEE, 2023. 246–257. [doi: 10.1109/ICST57152.2023.00031]

[20] Behrang F, Orso A. Automated test migration for mobile Apps. In: Proc. of the 40th Int'l Conf. on Software Engineering: Companion Proc. Gothenburg: ACM, 2018. 384–385. [doi: 10.1145/3183440.3195019]

[21] Gomez L, Neamtiu I, Azim T, Millstein T. RERAN: Timing- and touch-sensitive record and replay for Android. In: Proc. of the 35th Int'l Conf. on Software Engineering (ICSE). San Francisco: IEEE, 2013. 72–81. [doi: 10.1109/ICSE.2013.6606553]

[22] Yu SC, Fang CR, Tuo ZY, Zhang QJ, Chen CY, Chen ZY, Su ZD. Vision-based mobile App GUI testing: A survey. arXiv:2310.13518, 2024.

[23] Zhao X, Xu BJ, Wu GX. Canny edge detection based on Open CV. In: Proc. of the 13th IEEE Int'l Conf. on Electronic Measurement & Instruments (ICEMI). Yangzhou: IEEE, 2017. 53–56. [doi: 10.1109/ICEMI.2017.8265710]

[24] Wu J, Cui ZM, Sheng VS, Zhao PP, Su DL, Gong SR. A comparative study of SIFT and its variants. Measurement Science Review, 2013, 13(3): 122–131. [doi: 10.2478/msr-2013-0021]

[25] Memon J, Sami M, Khan RA, Uddin M. Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR). IEEE Access, 2020, 8: 142642–142668. [doi: 10.1109/ACCESS.2020.3012542]

[26] Sun XL, Li TY, Xu JF. UI components recognition system based on image understanding. In: Proc. of the 20th Int'l Conf. on Software Quality, Reliability and Security Companion (QRS-C). Macau: IEEE, 2020. 65–71. [doi: 10.1109/QRS-C51114.2020.00022]

[27] Altinbas MD, Serif T. GUI element detection from mobile UI images using YOLOv5. In: Proc. of the 18th Int'l Conf. on Mobile Web and Intelligent Information Systems. Rome: Springer, 2022. 32–45. [doi: 10.1007/978-3-031-14391-5_3]

[28] Chen JS, Swearngin A, Wu J, Barik T, Nichols J, Zhang XY. Towards complete icon labeling in mobile applications. In: Proc. of the 2022 CHI Conf. on Human Factors in Computing Systems. New Orleans: ACM, 2022. 387. [doi: 10.1145/3491102.3502073]

[29] Chiatti A, Cho MJ, Gagneja A, Yang X, Brinberg M, Roehrick K, Choudhury SR, Ram N, Reeves B, Giles CL. Text extraction and retrieval from smartphone screenshots: Building a repository for life in media. In: Proc. of the 33rd Annual ACM Symp. on Applied Computing. Pau: ACM, 2018. 948–955. [doi: 10.1145/3167132.3167236]

[30] Jaganeshwari K, Djodilatchoumy S. A novel approach of GUI mapping with image based widget detection and classification. In: Proc. of the 2nd Int'l Conf. on Intelligent Engineering and Management (ICIEM). London: IEEE, 2021. 342–346. [doi: 10.1109/ICIEM51511.2021.9445281]

[31] Ji MC. UIChecker: An automatic detection platform for Android GUI errors. In: Proc. of the 9th Int'l Conf. on Software Engineering and Service Science (ICSESS). Beijing: IEEE, 2018. 957–961. [doi: 10.1109/ICSESS.2018.8663923]

[32] Liu Z. Discovering UI display issues with visual understanding. In: Proc. of the 35th IEEE/ACM Int'l Conf. on Automated Software Engineering. Virtual Event: ACM, 2020. 1373–1375. [doi: 10.1145/3324884.3418917]

[33] Fang XX, Sheng B, Li P, Wu D, Wu EH. Automatic GUI test by using SIFT matching. China Communications, 2016, 13(9): 227–236. [doi: 10.1109/CC.2016.7582314]

[34] Wu H, Zhou Z. Using convolution neural network for defective image classification of industrial components. Mobile Information Systems, 2021, 2021(1): 9092589. [doi: 10.1155/2021/9092589]

[35] Chen JS, Xie ML, Xing ZC, Chen CY, Xu XW, Zhu LM, Li GQ. Object detection for graphical user interface: Old fashioned or deep learning or a combination? In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Virtual Event: ACM, 2020. 1202–1214. [doi: 10.1145/3368089.3409691]

[36] Xie ML, Feng SD, Xing ZC, Chen JS, Chen CY. UIED: A hybrid tool for GUI element detection. In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Virtual Event: ACM, 2020. 1655–1659. [doi: 10.1145/3368089.3417940]

[37] Feiz S, Wu J, Zhang XY, Swearngin A, Barik T, Nichols J. Understanding screen relationships from screenshots of smartphone applications. In: Proc. of the 27th Int'l Conf. on Intelligent User Interfaces. Helsinki: ACM, 2022. 447–458. [doi: 10.1145/3490099.3511109]

[38] Zhang YK, Zhu QH, Yan JW, Liu C, Zhang WJ, Zhao YF, Hao D, Zhang L. Synthesis-based enhancement for GUI test case migration. In: Proc. of the 33rd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Vienna: ACM, 2024. 869–881. [doi: 10.1145/3650212.3680327]

[39] Kirinuki H, Matsumoto S, Higo Y, Kusumoto S. Web element identification by combining NLP and heuristic search for web testing. In: Proc. of the 2022 IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER). Honolulu: IEEE, 2022. 1055–1065. [doi: 10.1109/SANER53432.2022.00123]

[40] Ardito L, Bottino A, Coppola R, Lamberti F, Manigrasso F, Morra L, Torchiano M. Feature matching-based approaches to improve the robustness of Android visual GUI testing. ACM Trans. on Software Engineering and Methodology (TOSEM), 2021, 31(2): 21. [doi: 10.1145/3477427]

[41] Liu Z, Chen CY, Wang JJ, Chen MZ, Wu BY, Che X, Wang DD, Wang Q. Make LLM a testing expert: Bringing human-like interaction to mobile GUI testing via functionality-aware decisions. arXiv:2310.15780, 2023.

[42] Hu J, Zhang Q, Yin H. Augmenting greybox fuzzing with generative AI. arXiv:2306.06782, 2023.

[43] Chen Y, Hu Z, Zhi C. ChatUniTest: A framework for LLM-based test generation. arXiv:2305.04764, 2024.

[44] Plein L, Ouédraogo WC, Klein J, Bissyandé TF. Automatic generation of test cases based on bug reports: A feasibility study with large language models. In: Proc. of the 46th Int'l Conf. on Software Engineering: Companion Proc. Lisbon: ACM, 2024. 360–361. [doi: 10.1145/3639478.3643119]

[45] Fan ZY, Gao X, Mirchev M, Roychoudhury A, Tan SH. Automated repair of programs from large language models. In: Proc. of the 45th Int'l Conf. on Software Engineering (ICSE). Melbourne: IEEE, 2023. 1469–1481. [doi: 10.1109/ICSE48619.2023.00128]

[46] Xia CS, Wei YX, Zhang LM. Practical program repair in the era of large pre-trained language models. arXiv:2210.14179, 2024.

[47] Plein L, Bissyandé TF. Can LLMs demystify bug reports? arXiv:2310.06310, 2023.

[48] Taylor A, Vassar A, Renzella J, Pearce H. Dcc--help: Transforming the role of the compiler by generating context-aware error explanations with large language models. In: Proc. of the 55th ACM Technical Symp. on Computer Science Education V.1. Portland: ACM, 2024. 1314–1320. [doi: 10.1145/3626252.3630822]

[49] Kang S, Chen B, Yoo S, Lou JG. Explainable automated debugging via large language model-driven scientific debugging. arXiv:2304.02195, 2023.

[50] Zhang C, Xue YZ, Chen JC. Design and application of Android platform-based GUI capture-replay testing tool. Computer Applications and Software, 2012, 29(12): 6–9, 68 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-386x.2012.12.002]

[51] Yu SC, Fang CR, Yun YX, Feng Y. Layout and image recognition driving cross-platform automated mobile testing. In: 2021 IEEE/ACM 43rd Int'l Conf. on Software Engineering (ICSE). Madrid: IEEE, 2021. 1561–1571. [doi: 10.1109/ICSE43902.2021.00139]

[52] Behrang F, Orso A. Test migration between mobile Apps with similar functionality. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). San Diego: IEEE, 2019. 54–65. [doi: 10.1109/ASE.2019.00016]

附中文参考文献:

[1] 梅宏, 曹东刚, 谢涛. 泛在操作系统: 面向人机物融合泛在计算的新蓝海. 中国科学院院刊, 2022, 37(1): 30–37. [doi: 10.16418/

j.issn.1000-3045.20211117009]
[2] 李宣东, 王戟, 詹乃军. 迎接人机物融合泛在计算时代, 提升关键软件技术创新与供给能力. 前瞻科技, 2023, 2(1): 5–6.
[3] 李聪, 蒋炎岩, 许畅. 基于 GUI 事件的安卓应用录制重放关键技术综述. 软件学报, 2022, 33(5): 1612–1634. http://www.jos.org.cn/1000-9825/6551.htm [doi: 10.13328/j.cnki.jos.006551]
[4] 张光兰, 万莹. 移动应用 GUI 测试技术综述. 现代计算机, 2019(10): 44–48. [doi: 10.3969/j.issn.1007-1423.2019.10.010]
[5] 张少坤, 李元春, 雷瀚文, 蒋鹏, 李锭, 郭耀, 陈向群. 基于多模态表征的移动应用 GUI 模糊测试框架. 软件学报, 2024, 35(7): 3162–3179. http://www.jos.org.cn/1000-9825/7106.htm [doi: 10.13328/j.cnki.jos.007106]
[14] 郑炜, 唐辉, 陈翔, 张满青, 夏鑫. 安卓移动应用兼容性测试综述. 计算机研究与发展, 2022, 59(6): 1370–1387. [doi: 10.7544/issn1000-1239.20210105]
[50] 张灿, 薛云志, 陈军成. 一种基于 Android 平台 GUI 录制回放工具的设计与实现. 计算机应用与软件, 2012, 29(12): 6–9, 68. [doi: 10.3969/j.issn.1000-386x.2012.12.002]

虞圣呈(1998－), 男, 博士, CCF 学生会员, 主要研究领域为 GUI 测试, 众包测试.

刘钦(1982－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为软件工程.

房春荣(1986－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为智能软件工程, BigCode, AITesting.

刘嘉(1976－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为人工智能测试与优化, 复杂系统质量保障, 数据分析应用, 大数据质量度量, 开发者社交网络.

钟棻(2004－), 女, 本科生, 主要研究领域为软件测试.

郑滔(1966－), 男, 博士, 教授, 主要研究领域为形式化方法, 编程语言, 自然语言处理, 知识图谱.

张犬俊(1994－), 男, 博士, CCF 专业会员, 主要研究领域为基于深度学习的自动程序修复, 智能软件测试, 软件安全漏洞检测和修复.

陈振宇(1978－), 男, 博士, 教授, 主要研究领域为群智测试, 深度学习测试与优化, 大数据质量, 移动应用测试.