AiDD

2024 AI+研发数字峰会

AI+ Development Digital summit

AI驱动研发变革 促进企业降本增效

北京站 08/16-17

基于多模态大模型的用户界面交互和测试

王俊杰 中国科学院软件研究所

演讲嘉宾

王俊杰

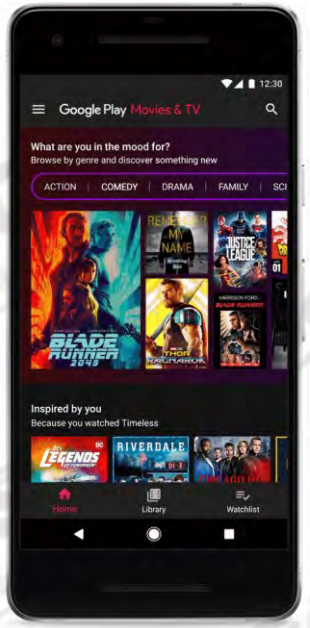
中国科学院软件研究所研究员，博士生导师

中国科学院软件研究所研究员，博士生导师，中国科学院特聘研究岗位、青年创新促进会会员，主要从事智能化软件工程、软件质量等方面的研究，近年来主要关注智能软件测试、大模型驱动的软件测试等。在国际著名学术期刊/会议发表60余篇高水平学术论文，四次荣获ACM/IEEE杰出论文奖。主持和参与了多项国家自然科学基金项目、科技部重点研发计划、CCF-华为胡杨林基金等。担任CCF A类期刊TSE的Associate Editor，ICSE、FSE、ISSRE等的PC member， TOSEM、EMSE、AUSE、软件学报等期刊的审稿人。

目录
CONTENTS

1. 用户界面测试现状和挑战
2. 测试输入生成技术
3. 面向测试路径规划的自动化GUI测试技术
4. 基于多模态大模型的自动化GUI测试技术
5. 针对文本输入的模糊测试技术
6. 面向文本输入组件的交互提升技术
7. 总结和展望

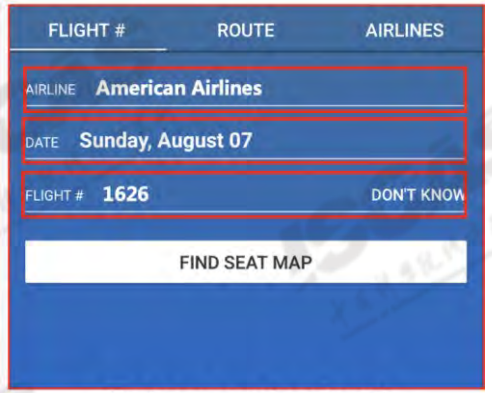
用户界面测试现状和挑战



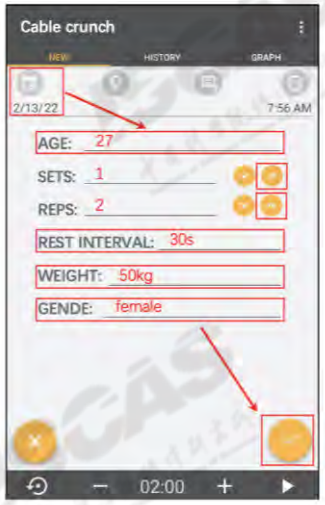
- Monkey
- Fastbot2
- Droidbot
- Ape
- WCTester
- Stoat
- TimeMachine
- ComboDroid
- Humanoid
- Q-testing

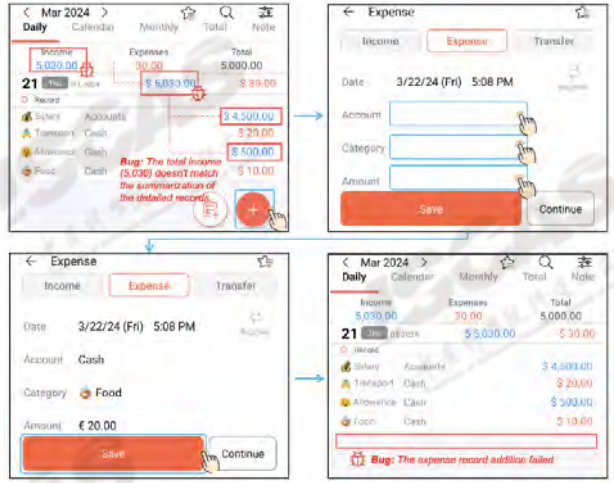
面临挑战

- 合适文本输入
- 连续长串操作
- 复合操作
- 页面功能理解
- 逻辑错误的发现



Find seat map (text input)

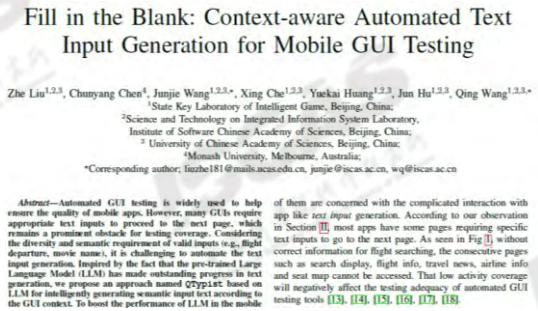




相关成果

- 多篇论文发表在软件工程和人机交互领域旗舰会议/期刊ICSE、TSE、CHI等
- 贝壳找房app、抖音app、华为鸿蒙生态、新能源汽车车载系统进行了应用或对接中

ICSE 2023



Fill in the Blank: Context-aware Automated Text Input Generation for Mobile GUI Testing

Zhe Liu[1,2,3], Chunyang Chen[4], Junjie Wang[1,2,3,*], Xing Che[1,2,3], Yuekai Huang[1,2,3], Jun Hu[1,2,3], Qing Wang[1,2,3,*]
[1]State Key Laboratory of Intelligent Game, Beijing, China;
[2]Science and Technology on Integrated Information System Laboratory,
Institute of Software Chinese Academy of Sciences, Beijing, China;
[3] University of Chinese Academy of Sciences, Beijing, China;
[4]Monash University, Melbourne, Australia;
*Corresponding author; liuzhe181@mails.ucas.edu.cn, junjie@iscas.ac.cn, wq@iscas.ac.cn

Abstract—Automated GUI testing is widely used to help ensure the quality of mobile apps. However, many GUIs require appropriate text inputs to proceed to the next page, which remains a prominent obstacle for testing coverage. Considering the diversity and semantic requirement of valid inputs (e.g., flight departure, movie name), it is challenging to automate the text input generation. Inspired by the fact that the pre-trained Large Language Model (LLM) has made outstanding progress in text generation, we propose an approach named QTypist based on LLM for intelligently generating semantic input text according to the GUI context. To boost the performance of LLM in the mobile

of them are concerned with the complicated interaction with app like *text input* generation. According to our observation in Section II, most apps have some pages requiring specific text inputs to go to the next page. As seen in Fig 1, without correct information for flight searching, the consecutive pages such as search display, flight info, travel news, airline info and seat map cannot be accessed. That low activity coverage will negatively affect the testing adequacy of automated GUI testing tools [13], [14], [15], [16], [17], [18].

ICSE 2024

Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions

Zhe Liu[1],Chunyang Chen[2], Junjie Wang[1,*], Mengzhuo Chen[1], Boyu Wu[1], Xing Che[1],
Dandan Wang[1], Qing Wang[1,*]
[1]State Key Laboratory of Intelligent Game, Beijing, China
Institute of Software Chinese Academy of Sciences, Beijing, China;
University of Chinese Academy of Sciences, Beijing, China; *Corresponding author
[2]Monash University, Melbourne, Australia;
liuzhe181@mails.ucas.ac.cn,Chunyang.chen@monash.edu,junjie@iscas.ac.cn,wq@iscas.ac.cn

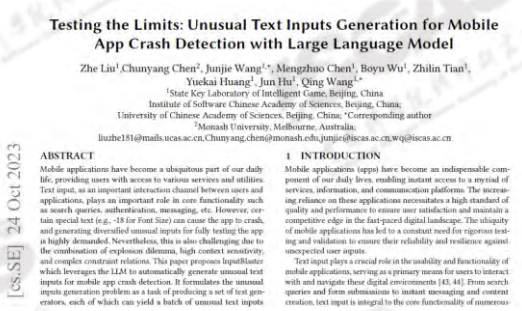
ABSTRACT

Automated Graphical User Interface (GUI) testing plays a crucial role in ensuring app quality, especially as mobile applications have become an integral part of our daily lives. Despite the growing popularity of learning-based techniques in automated GUI testing due to their ability to generate human-like interactions, they still suffer from several limitations, such as low testing coverage, inadequate generalization capabilities, and heavy reliance on training data. Inspired by the success of Large Language Models (LLMs) like ChatGPT in natural language understanding and question answering, we formulate the mobile GUI testing problem as a Q&A task. We propose GPTDroid, asking LLM to chat with the mobile apps by passing the GUI page information to LLM to elicit testing scripts, and executing them to keep passing the app feedback to LLM, iterating the whole process. Within this framework, we have also introduced a functionality-aware memory prompting mechanism that equips the LLM with the ability to retain testing knowledge

This is because of the complex and dynamic nature of modern mobile apps [18, 22, 31, 48, 51, 52], which can have hundreds or even thousands of different screens, each with its own unique set of interactions and possible user actions and logic. In addition, test inputs generated by these methods are significantly different from real users' interaction traces [49], resulting in low testing coverage. To address these limitations, there has been a growing interest in using deep learning (DL) [34, 75] and reinforcement learning (RL) [46, 50] techniques for automated mobile GUI testing. By learning from human testers' behavior, these methods aim to generate human-like actions and interactions that can be used to test the app's GUI more thoroughly and effectively. These approaches are based on the idea that the more closely the actions performed by the testing algorithm mimic those of a human user, the more comprehensive and effective the testing will be.

Nevertheless, there are still some limitations with those DL or RL-based GUI testing methods. First, learning algorithms require

ICSE 2024



Testing the Limits: Unusual Text Inputs Generation for Mobile App Crash Detection with Large Language Model

Zhe Liu[1],Chunyang Chen[2], Junjie Wang[1,*], Mengzhuo Chen[1], Boyu Wu[1], Zhilin Tian[1],
Yuekai Huang[1], Jun Hu[1], Qing Wang[1,*]
[1]State Key Laboratory of Intelligent Game, Beijing, China
Institute of Software Chinese Academy of Sciences, Beijing, China;
University of Chinese Academy of Sciences, Beijing, China; *Corresponding author
[2]Monash University, Melbourne, Australia;
liuzhe181@mails.ucas.ac.cn,Chunyang.chen@monash.edu,junjie@iscas.ac.cn,wq@iscas.ac.cn

ABSTRACT

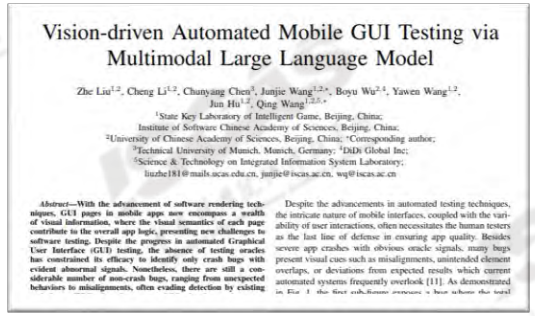
Mobile applications have become a ubiquitous part of our daily life, providing users with access to various services and utilities. Text input, as an important interaction channel between users and applications, plays an important role in core functionality such as search queries, authentication, messaging, etc. However, certain special text (e.g., ~18 for Font Size) can cause the app to crash, and generating diversified unusual inputs for fully testing the app is highly demanded. Nevertheless, this is also challenging due to the combination of explosion dilemma, high context sensitivity, and complex constraint relations. This paper proposes InputBlaster which leverages the LLM to automatically generate unusual text inputs for mobile app crash detection. It formulates the unusual inputs generation problem as a task of producing a set of text generators, each of which can yield a batch of unusual text inputs

1 INTRODUCTION

Mobile applications (apps) have become an indispensable component of our daily lives, enabling instant access to a myriad of services, information, and communication platforms. The increasing reliance on these applications necessitates a high standard of quality and performance to ensure user satisfaction and maintain a competitive edge in the fast-paced digital landscape. The ubiquity of mobile applications has led to a constant need for rigorous testing and validation to ensure their reliability and resilience against unexpected user inputs.

Text input plays a crucial role in the usability and functionality of mobile applications, serving as a primary means for users to interact with and navigate these digital environments [43, 44]. From search queries and form submissions to instant messaging and content creation, text input is integral to the core functionality of numerous

Under submission



Vision-driven Automated Mobile GUI Testing via Multimodal Large Language Model

Zhe Liu[1,2], Cheng Li[1,2], Chunyang Chen[3], Junjie Wang[1,2,*], Boyu Wu[2,4], Yawen Wang[1,2],
Jun Hu[1,2], Qing Wang[1,2,5,*]
[1]State Key Laboratory of Intelligent Game, Beijing, China;
Institute of Software Chinese Academy of Sciences, Beijing, China;
[2]University of Chinese Academy of Sciences, Beijing, China; *Corresponding author;
[3]Technical University of Munich, Munich, Germany; [4]DiDi Global Inc;
[5]Science & Technology on Integrated Information System Laboratory;
liuzhe181@mails.ucas.edu.cn, junjie@iscas.ac.cn, wq@iscas.ac.cn

Abstract—With the advancement of software rendering techniques, GUI pages in mobile apps now encompass a wealth of visual information, where the visual semantics of each page contribute to the overall app logic, presenting new challenges to software testing. Despite the progress in automated Graphical User Interface (GUI) testing, the absence of testing oracles has constrained its efficacy to identify only crash bugs with evident abnormal signals. Nonetheless, there are still a considerable number of non-crash bugs, ranging from unexpected behaviors to misalignments, often evading detection by existing

Despite the advancements in automated testing techniques, the intricate nature of mobile interfaces, coupled with the variability of user interactions, often necessitates the human testers as the last line of defense in ensuring app quality. Besides severe app crashes with obvious oracle signals, many bugs present visual cues such as misalignments, unintended element overlaps, or deviations from expected results which current automated systems frequently overlook [11]. As demonstrated

CHI 2024 最佳论文提名奖

Unblind Text Inputs: Predicting Hint-text of Text Input in Mobile Apps via LLM

Zhe Liu[1,2],Chunyang Chen[3], Junjie Wang[1,2,*], Mengzhuo Chen[1,2], Boyu Wu[2,4], Yuekai Huang[1,2],
Jun Hu[1,2], Qing Wang[1,2,5,*]
[1]State Key Laboratory of Intelligent Game, Beijing, China
Institute of Software Chinese Academy of Sciences, Beijing, China;
[2]University of Chinese Academy of Sciences, Beijing, China; *Corresponding author;
[3]Technical University of Munich, Munich, Germany; [4]DiDi Global Inc;
[5]Science & Technology on Integrated Information System Laboratory, Beijing, China
liuzhe181@mails.ucas.ac.cn,Chunyang.chen@monash.edu,junjie@iscas.ac.cn,wq@iscas.ac.cn

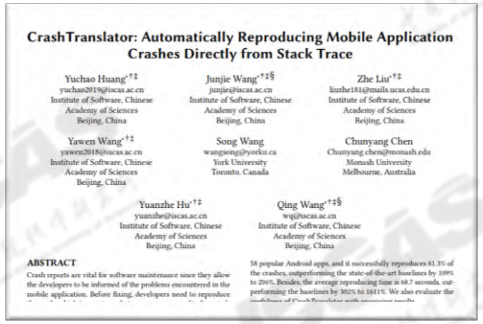
ABSTRACT

Mobile apps have become indispensable for accessing and participating in various environments, especially for low-vision users. Users with visual impairments can use screen readers to read the content of each screen and understand the content that needs to be

1 INTRODUCTION

In the era of burgeoning mobile device development, mobile applications have evolved into indispensable tools for accessing an engaging within diverse environments, offering unparalleled convenience, particularly for individuals with low vision. According

ICSE 2024



CrashTranslator: Automatically Reproducing Mobile Application Crashes Directly from Stack Trace

Yuchao Huang*†‡
yuchao2019@iscas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China

Junjie Wang*†‡§
junjie@iscas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China

Zhe Liu*†‡
liuzhe1814@mails.ucas.edu.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China

Yawen Wang*†‡
yawen2018@iscas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China

Song Wang
wangsong@yorku.ca
York University
Toronto, Canada

Chunyang Chen
Chunyang.chen@monash.edu
Monash University
Melbourne, Australia

Yuanzhe Hu*†‡
yuanzhe@iscas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China

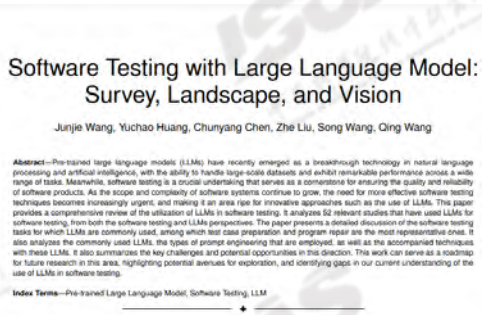
Qing Wang*†‡§
wq@iscas.ac.cn
Institute of Software, Chinese Academy of Sciences
Beijing, China

ABSTRACT

Crash reports are vital for software maintenance since they allow the developers to be informed of the problems encountered in the mobile application. Before fixing, developers need to reproduce

58 popular Android apps, and it successfully reproduces 61.3% of the crashes, outperforming the state-of-the-art baselines by 109% to 206%. Besides, the average reproducing time is 68.7 seconds, outperforming the baselines by 302% to 1611%. We also evaluate the

TSE 2024



Software Testing with Large Language Model: Survey, Landscape, and Vision

Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, Qing Wang

Abstract—Pre-trained large language models (LLMs) have recently emerged as a breakthrough technology in natural language processing and artificial intelligence, with the ability to handle large-scale datasets and exhibit remarkable performance across a wide range of tasks. Meanwhile, software testing is a crucial undertaking that serves as a cornerstone for ensuring the quality and reliability of software products. As the scope and complexity of software systems continue to grow, the need for more effective software testing techniques becomes increasingly urgent, and making it an area ripe for innovative approaches such as the use of LLMs. This paper provides a comprehensive review of the utilization of LLMs in software testing. It analyzes 52 relevant studies that have used LLMs for software testing, from both the software testing and LLMs perspectives. The paper presents a detailed discussion of the software testing tasks for which LLMs are commonly used, among which test case preparation and program repair are the most representative ones. It also analyzes the commonly used LLMs, the types of prompt engineering that are employed, as well as the accompanied techniques with these LLMs. It also summarizes the key challenges and potential opportunities in this direction. This work can serve as a roadmap for future research in this area, highlighting potential avenues for exploration, and identifying gaps in our current understanding of the use of LLMs in software testing.

Index Terms—Pre-trained Large Language Model, Software Testing, LLM

FSE2024 -SE2030

A Roadmap for Software Testing in Open-Sourced and AI-Powered Era

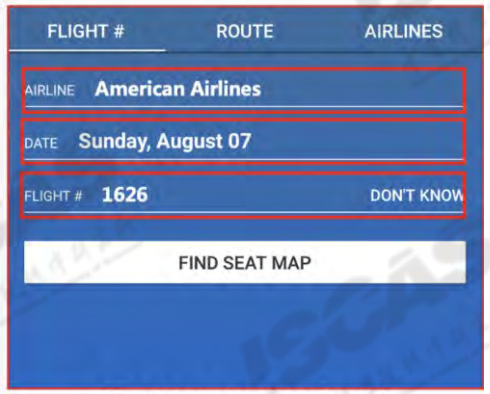
Qing Wang*, Junjie Wang*, Mingyang Li, Yawen Wang, Zhe Liu
{wq,junjie,mingyang2017,yawen2018,liuzhe2020}@iscas.ac.cn;
State Key Laboratory of Intelligent Game, Institute of Software Chinese Academy of Sciences,
University of Chinese Academy of Sciences, Beijing, China; *corresponding authors

ABSTRACT

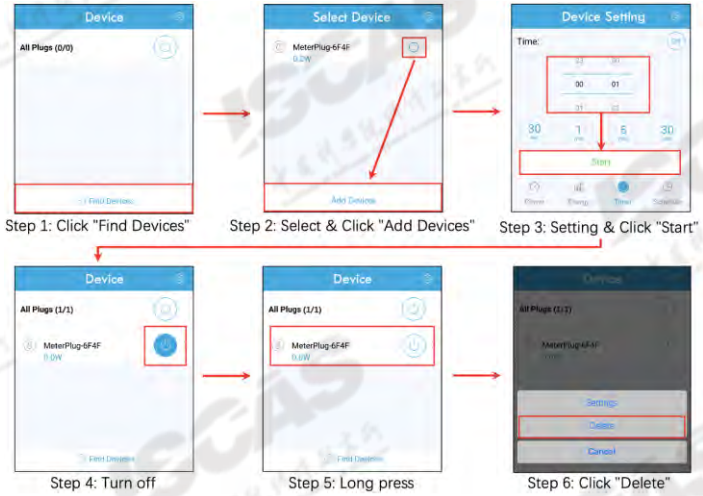
midst the ever-expanding digital sphere, the evolution of the ternet has not only fostered an atmosphere of information transrency and sharing but has also sparked a revolution in software evelopment practices. The distributed nature of open-source deelopment, along with its diverse contributors and rapid iterations, resents new challenges for ensuring software quality. Meanwhile,

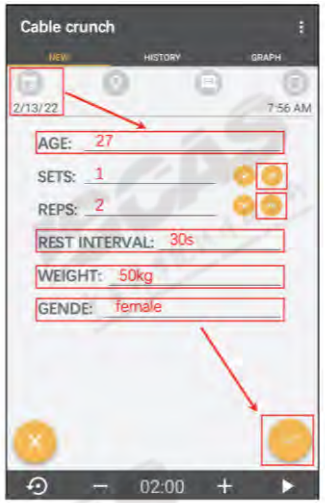
has led to the transformative open-source software developer paradigm [11]. Platforms like GitHub have flourished, providin developers from diverse backgrounds with the opportunity to co tribute code, discuss technical matters, and address software issue regardless of their geographic location [5, 63]. As of now, GitHu hosts a vast array of projects spanning various domains, inclu ing operating systems, programming languages, and applicatio

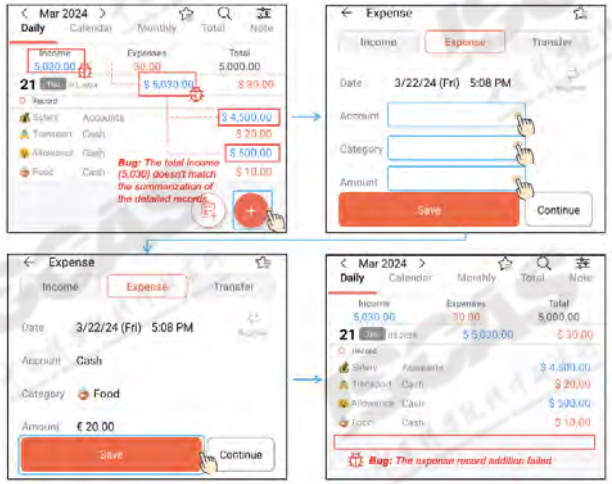
提纲



Find seat map (text input)



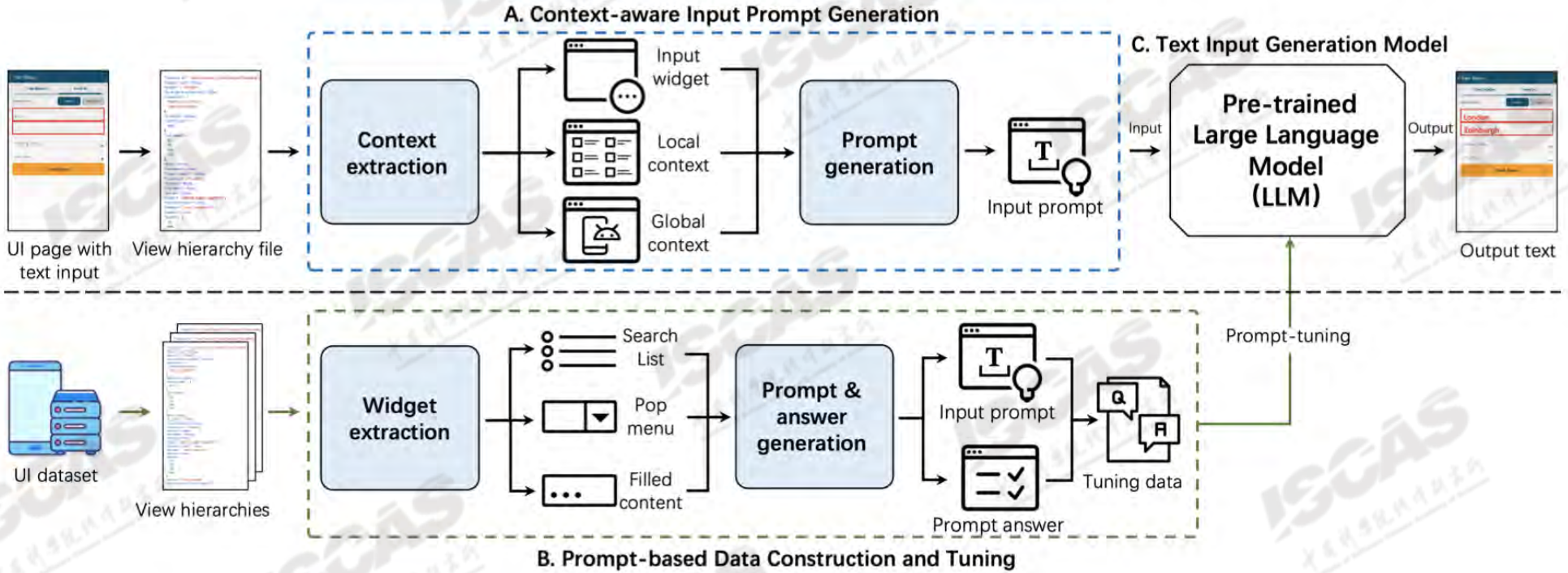




- 用户界面测试现状和挑战
- 测试输入生成技术
- 面向测试路径规划的自动化GUI测试技术
- 基于多模态大模型的自动化GUI测试技术
- 针对文本输入的模糊测试技术
- 面向文本输入组件的交互提升技术

Text Input Generation

Ask LLM to fill in the blank according to the generated prompts



Fill in the Blank: Context-aware Automated Text Input Generation for Mobile GUI. ICSE 2023

Text input generation

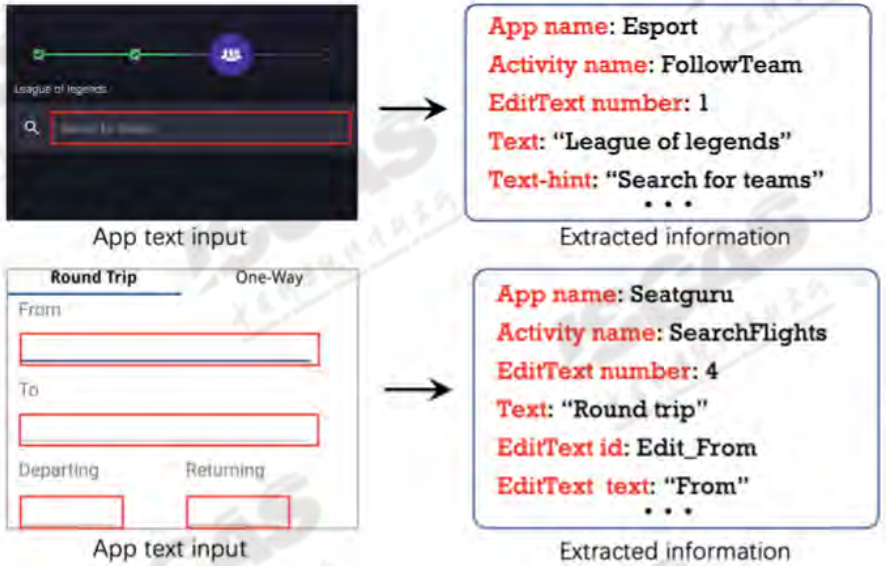
- Set up linguistic patterns to generate prompts based on the current page

Id	Sample of linguistic patterns/rules	Examples of linguistic patterns/rules instantiation
	Patterns related to input widget: $\langle IWPtn \rangle$	
1	Please input $\langle widget[n] \rangle$, the $\langle widget[n] \rangle$ is	Please input game name, the game name is
2	Please $\langle widget[v+n] \rangle$, the $\langle widget[n] \rangle$ is	Please search the food, the food is
3	$\langle widget[n] \rangle$ + $[MASK]$ + $\langle widget[n] \rangle$	Your weight is [MASK] kg
4	$\langle widget[prep] \rangle$ + $[MASK]$	From [MASK]
	Patterns related to local context: $\langle LCPtn \rangle$	
5	This input is about $\langle local[n] \rangle$	This input is about the NBA team.
6	This input is about $\langle local[n] \rangle$, we need to $\langle local[v+n] \rangle$	This input is about one-way flight, we need to search the flight information.
	Patterns related to global context: $\langle GCPtn \rangle$	
7	This is $\langle app\ name \rangle$ app, in its $\langle activity\ name \rangle$ page, the input category is $\langle input\ category \rangle$.	This is a NBA sport app, in its search the NBA team page, the input category is query category.
	Prompt generation rules	
1	$\langle GCPtn \rangle$ + $\langle LCPtn \rangle$ + $\langle IWPtn \rangle$	This is a my movie app, in its search movie page, the input category is query category. This input is about your favorite move in this year. Please search the movie, the movie is
2	$\langle GCPtn \rangle$ + [$\langle LCPtn \rangle$ + $\langle IWPtn \rangle$]{n}	This is a money wallet app, in its personal income page, the input category is numeric category. This input is about your monthly income. Income is [MASK] dollar. This input is about your expenses. Expenses is [MASK] dollar.

Notes: "[n]", "[v]", "[v+n]" and "[prep]" means noun, verb, verb+noun and preposition in the related information.

Text input generation for mobile app testing

- Examples



Evaluation

- Passing rate: 0.87
- Significant activity boost and 122% (51 vs 23) more bugs by added to GUI testing tools

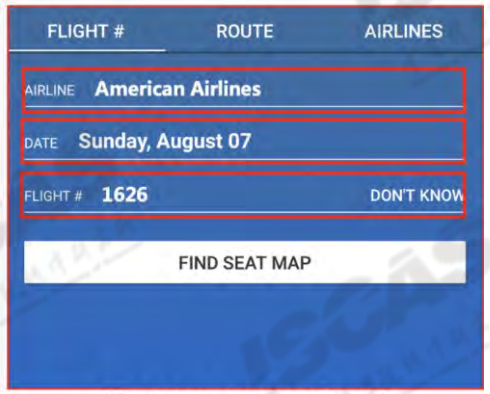
TABLE III: Result of passing rate. (RQ1)

Method	Ident	Geo	Num	Query	Comm	All
Random-/rule-based method						
Stoat	0.10	0.10	0.05	0.15	0.60	0.20
Droidbot	0.10	0.05	0.00	0.15	0.60	0.18
Ape	0.20	0.15	0.10	0.12	0.65	0.24
Fastbot	0.15	0.10	0.05	0.12	0.65	0.21
ComboDroid	0.15	0.05	0.10	0.19	0.60	0.22
TimeMachine	0.10	0.15	0.05	0.15	0.65	0.22
Humanoid	0.15	0.10	0.05	0.15	0.60	0.21
Q-testing	0.10	0.15	0.10	0.15	0.65	0.23
Constraint-based method						
Mobolic	0.25	0.15	0.25	0.15	0.65	0.28
TextExerciser	0.45	0.15	0.40	0.23	0.70	0.38
Learning-based method						
RNNInput	0.35	0.40	0.25	0.50	0.75	0.45
QTypist (-T)	0.55	0.65	0.50	0.58	0.80	0.61
QTypist	**0.85**	**0.90**	**0.85**	**0.85**	**0.90**	**0.87**

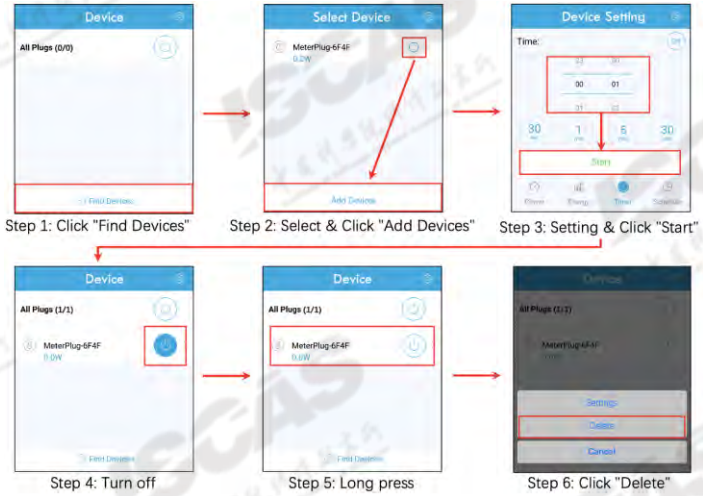
TABLE IV: Result of activity and page number compare with automated GUI testing tool with QTypist.

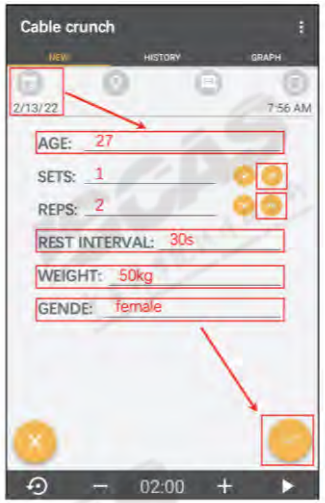
Id	App name	Categ	Down	Triggered activity number						Triggered page number					
				M	M+QT	D	D+QT	A	A+QT	M	M+QT	D	D+QT	A	A+QT
1	Eskimi	Social	1M+	3	4 ↑33%	4	5 ↑25%	4	6 ↑50%	5	8 ↑60%	7	10 ↑43%	8	14 ↑75%
2	Recharge	Shop	1M+	5	5 0%	8	13 ↑63%	9	15 ↑67%	6	8 ↑33%	10	16 ↑60%	11	18 ↑64%
3	Scout24	Vehicle	10M+	4	7 ↑75%	10	17 ↑70%	12	19 ↑58%	5	9 ↑80%	11	19 ↑73%	14	24 ↑71%
4	Stock	Finance	1M+	5	9 ↑80%	12	18 ↑50%	10	18 ↑80%	7	13 ↑86%	15	23 ↑53%	16	26 ↑63%
5	Lite	Photo	1M+	4	6 ↑50%	6	11 ↑83%	9	12 ↑33%	5	8 ↑60%	12	19 ↑58%	14	23 ↑64%
6	Speedom	Vehicle	1M+	4	5 ↑25%	8	8 0%	8	13 ↑63%	5	6 ↑20%	9	9 0%	9	15 ↑67%
7	HeartRate	Health	1M+	5	5 0%	6	7 ↑17%	6	9 ↑50%	6	6 0%	7	9 ↑29%	8	14 ↑75%
8	Currency	Tools	1M+	3	3 0%	5	7 ↑40%	6	7 ↑17%	4	4 0%	6	8 ↑33%	7	11 ↑57%
9	Yandex	Shop	10M+	8	11 ↑38%	14	20 ↑43%	15	22 ↑47%	13	16 ↑23%	17	23 ↑35%	21	31 ↑48%
10	Atom	Finance	1M+	2	2 0%	3	4 ↑33%	3	5 ↑67%	4	4 0%	5	7 ↑40%	5	8 ↑60%
11	Coco	Comm	10M+	2	3 ↑50%	8	13 ↑63%	7	13 ↑86%	3	5 ↑67%	10	15 ↑50%	9	15 ↑67%
12	Saviry	Shop	100K+	4	7 ↑75%	8	12 ↑50%	9	13 ↑44%	8	12 ↑50%	15	23 ↑53%	18	29 ↑61%
13	WhiteMeet	Dating	100K+	7	8 ↑14%	10	13 ↑30%	10	16 ↑60%	10	12 ↑20%	15	19 ↑27%	16	25 ↑56%
14	Healthplus	Health	1M+	5	5 0%	7	7 0%	8	13 ↑63%	9	10 ↑11%	11	13 ↑18%	13	19 ↑46%
15	LINEC	Photo	10M+	5	7 ↑40%	12	17 ↑42%	11	18 ↑64%	6	9 ↑50%	11	19 ↑73%	15	27 ↑80%
16	FloorPlan	Art	5M+	4	5 ↑25%	7	7 0%	9	10 ↑11%	5	8 ↑60%	10	11 ↑10%	13	18 ↑38%
17	MoneyTK	Finance	10M+	4	6 ↑50%	8	9 ↑13%	9	13 ↑44%	7	8 ↑14%	15	20 ↑33%	17	26 ↑53%
18	Schoolca	Educat	1M+	2	2 0%	4	6 ↑50%	8	17 ↑50%	4	4 0%	8	10 ↑25%	8	13 ↑63%
19	Flipboa	News	50M+	3	4 ↑33%	4	5 ↑25%	7	7 0%	5	8 ↑60%	7	11 ↑57%	8	12 ↑50%
20	Healthplus	Fitness	1M+	4	4 0%	8	9 ↑13%	11	12 ↑ 9%	6	6 0%	13	19 ↑46%	15	20 ↑33%
21	BlaWM	Travel	1M+	5	6 ↑20%	12	14 ↑17%	15	17 ↑13%	5	6 ↑20%	18	25 ↑39%	21	27 ↑29%
22	Wldetector	Product	1M+	3	3 0%	8	9 ↑13%	12	16 ↑33%	5	6 ↑20%	15	16 ↑ 7%	17	18 ↑ 6%
23	CrAm	Educat	5M+	2	3 ↑50%	4	7 ↑75%	9	10 ↑11%	7	7 0%	8	15 ↑88%	11	16 ↑45%
24	InsTEAD	Game	100K+	4	5 ↑25%	7	8 ↑14%	9	12 ↑33%	6	9 ↑50%	11	19 ↑73%	10	19 ↑90%
25	FitNot	Connect	1M+	7	8 ↑14%	12	15 ↑25%	15	18 ↑20%	8	10 ↑25%	17	25 ↑47%	19	29 ↑53%
26	GPST	Navig	1M+	4	4 0%	6	8 ↑33%	8	11 ↑38%	6	6 0%	10	14 ↑40%	9	13 ↑44%
27	DMCRA	News	100K+	5	7 ↑40%	6	8 ↑33%	8	10 ↑25%	10	13 ↑30%	11	14 ↑27%	10	14 ↑40%
28	Metro	Navig	1M+	3	3 0%	4	4 0%	6	8 ↑33%	5	5 0%	6	6 0%	7	9 ↑29%
29	WallE	Finance	5M+	4	6 ↑50%	5	8 ↑60%	7	10 ↑43%	4	6 ↑50%	6	9 ↑50%	9	12 ↑33%
30	PocketM	Travel	100K+	4	5 ↑25%	6	7 ↑17%	9	12 ↑33%	7	8 ↑14%	9	11 ↑22%	12	16 ↑33%
Average boost					**↑ 28%**		**↑ 33%**		**↑ 42%**		**↑ 30%**		**↑ 41%**		**↑ 52%**

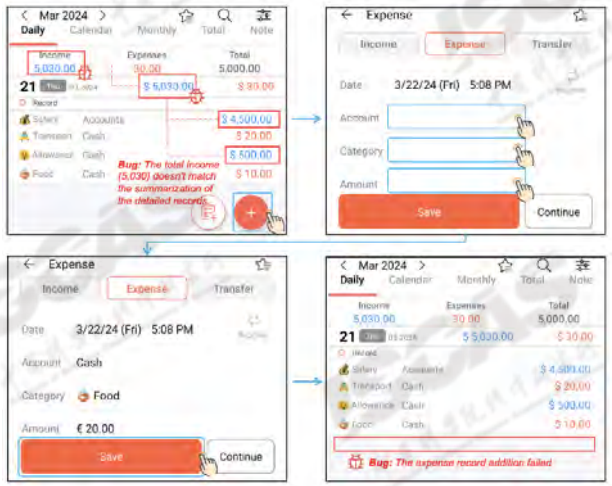
提纲



Find seat map (text input)

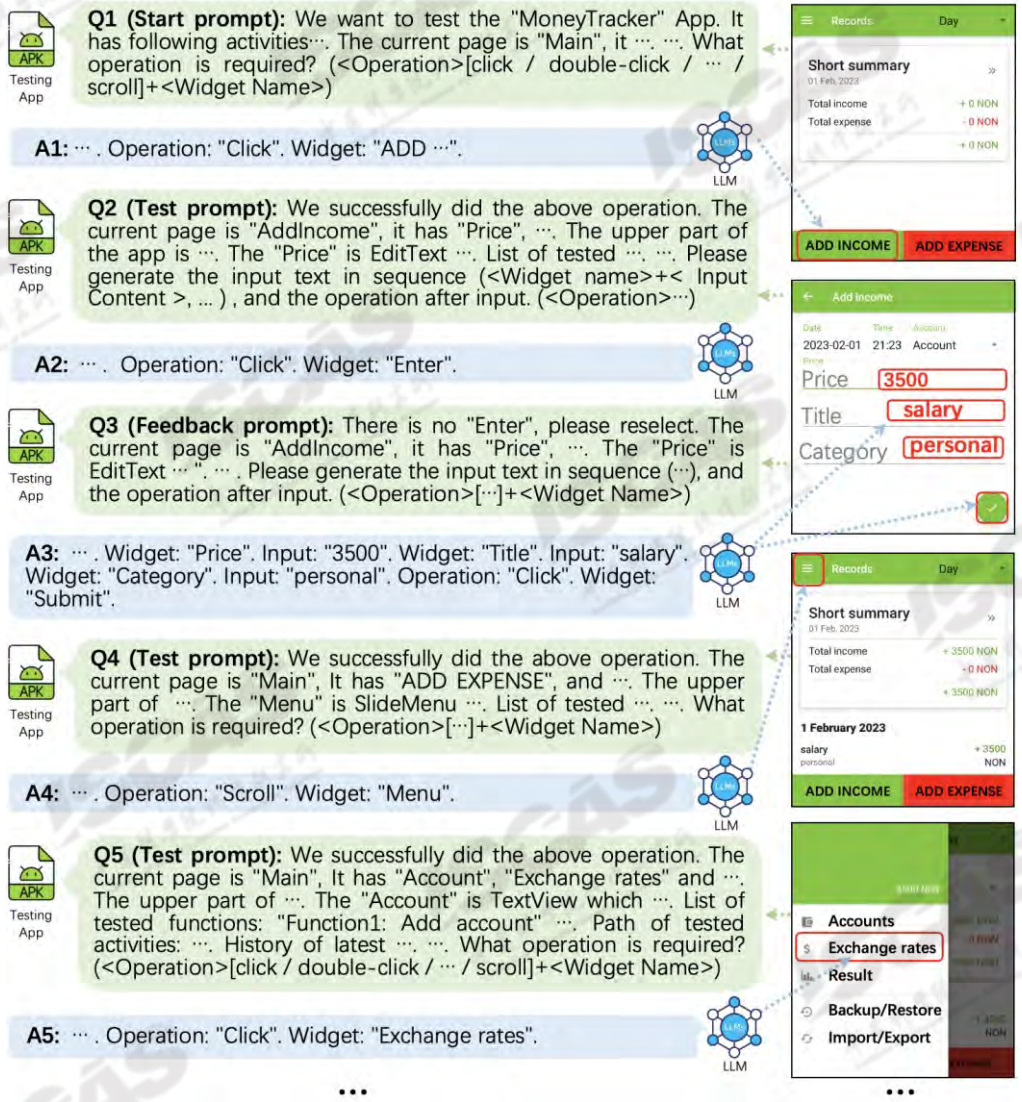






- 用户界面测试现状和挑战
- 测试输入生成技术
- 面向测试路径规划的自动化GUI测试技术
- 基于多模态大模型的自动化GUI测试技术
- 针对文本输入的模糊测试技术
- 面向文本输入组件的交互提升技术

GPTDroid: Function-aware Automatic GUI testing

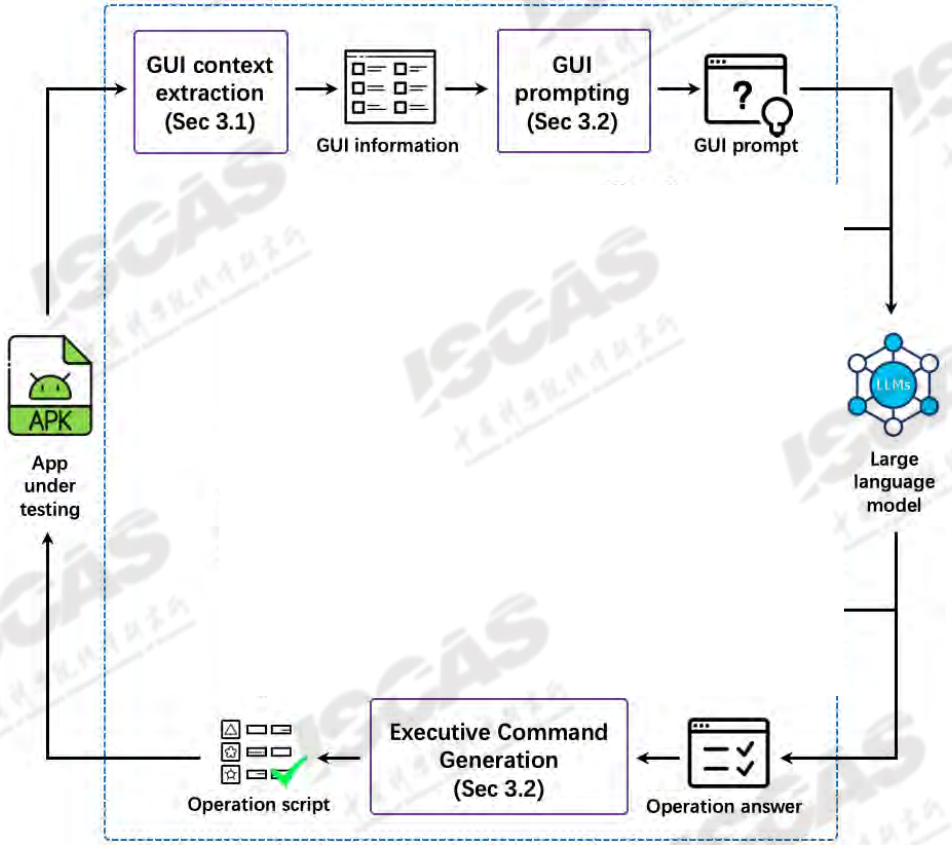


Auto GUI testing with LLM

- Formulate the automatic GUI testing problem to an interactive question & answering task
- to let the LLM conduct the whole app testing by understanding the GUI semantic information and automatically inferring possible operation steps

GPTDroid: Function-aware Automatic GUI testing

- **GUI context extraction**
- **GUI prompting and executive command generation**
- **Functionality-aware memory prompting**
  - Testing sequence memorizer to record all the detailed interactive testing information, e.g., the explored activities and widgets



Make LLM a Testing Expert: Bringing Human-like Interaction to Mobile GUI Testing via Functionality-aware Decisions. ICSE 2024

GPTDroid: Function-aware Automatic GUI testing

GUI context extraction

- Accurately depict the GUI page currently under test, as well as its contained widgets information from a more micro perspective, and the app information from a more macro perspective.

Table 1: Extracted GUI information and examples.

Id	Attribute	Description	Examples
		GUI context - App information	
1	AppName	Name of the app under testing	AppName = "Money Tracker"
2	Activities	List of names for all activities of the app, obtained from *AndroidManifest.xml* file	Activities = ["Main", "AddAccount", "Import", "Income", ...]
		GUI context - page GUI information	
3	ActivityName	Activity name of the current GUI page	ActivityName = "AddPersonalInformation"
4	Widgets	List of all widgets in current page, represented with text/id	Widgets = ["Edit Account", "btn_income", ...]
5	Position	Relative position of widgets, obtained through their coordinates	Upper = ["Welcome", ...], Lower = ["Add Income", ...]
		GUI context - widget information	
6	WidgetText	Widget text, obtained by field 'text' or 'hint-text'	WidgetText = "Welcome to the Money Tracker!"
7	WidgetID	Widget ID, obtained by field 'resource-id'.	WidgetID = "add_account"
8	WidgetCategory	Category: TextView, EditText, ImageView, etc, obtained by field 'class'	WidgetCategory = "TextView"
9	WidgetAction	Widget action, obtained by field 'clickable', such as click, input, etc.	WidgetAction = "Click"
10	NearbyWidget	Nearby widgets, obtained by the text of parent widgets and sibling widgets	NearbyWidget = "your income: [SEP] $ "

GPTDroid: Function-aware Automatic GUI testing

◆ GUI prompting and executive command generation

- Feedback prompt: inform the LLM error occurred and re-try for querying next operation
- Command generation: provide LLM with the output template, including available operations and operation primitives

Table 2: The example of linguistic patterns of GUI prompts and generation rules.

Id	Pattern type	Sample of linguistic patterns/rules	Instantiation
		GUI context patterns: *GUIContext*	
1	App information	We want to test the <*AppName*> App. It has the following activities, including <*Activities*>.	We want to test "Money tracker" App. It has the following activities, including "Main", "AddAccount", "Import", "Setting", ...
2	Page GUI information	The current page is <*ActivityName*>, it has <*Widgets*>. The upper part of the app is <*Position*>, the lower part is <*Position*>.	The current page is "Main", it has "Income", "Add", "Delete", ... The upper part of the app is "Welcome to ..., Delete, ...", the lower part of the app is "Income, ...".
3	Widget information	The widgets which can be operated are <*WidgetText / WidgetID*>. <*WidgetText/WidgetID*> is <*WidgetCategory*> which can <*WidgetAction*> and its nearby widget is <*NearbyWidget*>.	The widgets which can be operated are "Add", "Delete", "Edit Account",... . "Add" is Button which can be clicked and its nearby widget is "Add account, ..." , "Delete" is TextView which can be clicked and its nearby widget is ... .
		Operation & feedback question patterns: *OperationQuestion*	
4	Querying general action	Action operation question + <*Output Template*>	What operation is required? (<*Operation*>[click / double-click / long press / scroll]+<*Widget Name*>)
5	Querying text input	Input operation question + <*Output Template*>	Please generate the input text in sequence, and the operation after input. (<*Widget name*>+<*Input Content*>, ...) and provided (<*Operation*>[click]>+<*Widget name*>)
6	Testing feedback	Feedback question	There is no <*WidgetText / WidgetID*> on the current page, please reselect.
		Prompt generation rules	
1	**Start Prompt:** *GUIContext*[1,2,3] + *OperationQuestion*[4/5]		
2	**Test Prompt:** We successfully did the above operation. *GUIContext*[2,3] + *OperationQuestion*[4/5]		
3	**Feedback Prompt:** Sorry, <*OperationQuestion*>[6] + <*GUIContext*>[3] + <*OperationQuestion*>[4/5]		

Notes: "[1,2, ..., 6]" means the id of each pattern. If there is "EditText" on the page, GPTDroid selects "Querying text input" pattern. For the other widgets, GPTDroid selects "Querying general action" pattern.

GPTDroid: Function-aware Automatic GUI testing

◆ Functionality-aware memory prompting

- Build a testing sequence memorizer to record all detailed testing information
- Query the LLM about the function-level progress of the testing
- Functionality-aware memory prompt

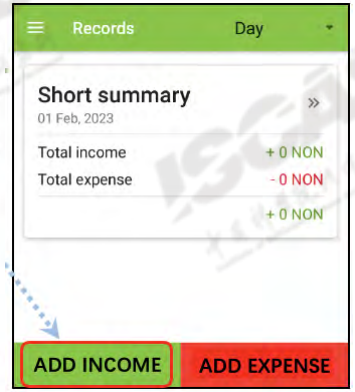
Table 3: The example of linguistic patterns of functionality-aware memory prompts and generation rules.

Id	Type	Sample of linguistic patterns/rules	Instantiation
		Long-term functionality-aware memory patterns: *FunctionMemory*	
1	Explored functionalities	**List of tested functions:** <*Function Name*> + <*Visits Time*> + <*Status*>, <*Function Name*> + <*Visits Time*> + <*Status*> ...	**List of tested functions:** "Function1: Add your income. Visits: 3. Status: Finished", "Function2: Delete information. Visits: 2. Status: Finished", ...
2	Covered activities.	**Path of tested activities:** <*Activity Name*> + <*Visits Time*>, <*Activity Name*> + <*Visits Time*>, ...	**Path of tested activities:** "Activity: Main. Visits: 3", "Activity: Account. Visits: 4", "Activity: AddAccount. Visits: 3", ...
3	Recently tested operations	**History of latest tested pages and operations: Latest 5th step** tested the <*Activity Name*> page. The following widgets with visits time of this page have been tested: <*Widget Name*> + <*Visits Time*>, ... The following executive command achieve the page transition: <*Operation*> + <*Widget Name*>. **Latest 4th step** tested the <*Activity Name*> page. The following ... ... **Latest 1st step** tested the <*Activity Name*> page. The following ...	**History of latest tested pages and operations: Latest 5th step** tested the "Exchange" page. The following widgets with visits time of this page have been tested: "Widget: Add exchange, Visits:2", "Widget: Submit exchange, Visits:1", 'Widget: Cancel, Visits:1" ... . The following executive command achieve the page transition: "Click" the "Exchange". **Latest 4th step** tested the "main" page. The following widgets ... ... **Latest 1st step** tested the "Add Account" page. The following widgets ...
		Function question patterns: *FunctionQuestion*	
4	Functionality inquiry	Functionality inquiry + <*Output Template*>	What is the functions currently being tested? Are we testing a new function? (<*FunctionName*> + <*Status*>)
		Functionality-aware memory prompt generation rules	
1	**Functionality-aware memory Prompt:** *FunctionMemory*[1,2,3] + *FunctionQuestion*[4]		

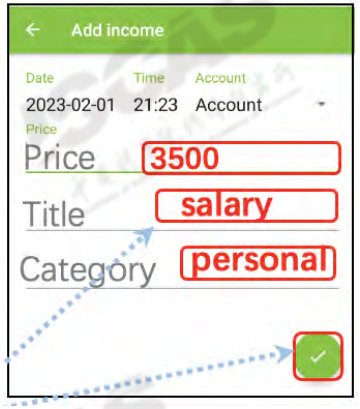
Example

Table 4: The example of how prompts work in GPTDroid.

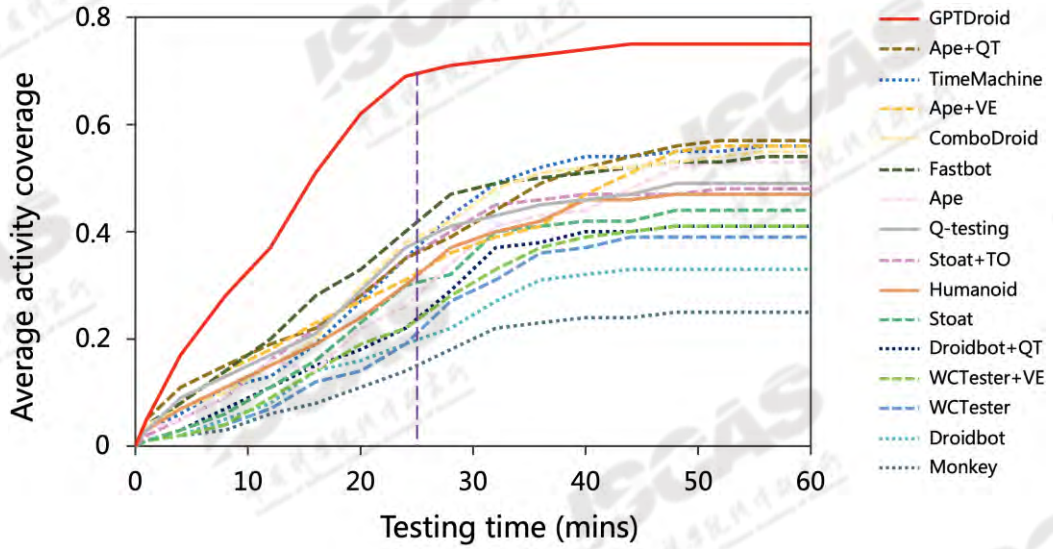
Prompt type	Instantiation
	Example 1: For general action operation
GUI context	**Start Prompt:**[We want to test "Money tracker" App, It has the following activities, including ... .] **or Test Prompt:** [We successfully did the above operation.] **or Feedback prompt:** [There is no "Exchange" on the current page, please reselect.] [The current page is "AddAccount", it has ... The upper part of the app is ... , the lower part ... The "Exchange" is Button ...]
Functionality-aware memory	**[List of tested functions:** "Function1 Add your income. ...", ...] **[Path of tested activities:** "Activity: Main Visits: 3", ...] **[History of latest tested pages and operations: Latest 5th step** tested the "Exchange" page. The following widgets ... **Latest 1st step** tested the "AddAccount" page. ...]
Function question	What is the functions currently being tested? Are we testing a new function? (<Function name> + <Status>)
Action question	What operation is required? (<*Operation*>[click / double-click / long press / scroll]+<*Widget Name*>)
LLM Answer	Function: "Add income". Status: Yes. Operation: "Click". Widget: "ADD INCOME".

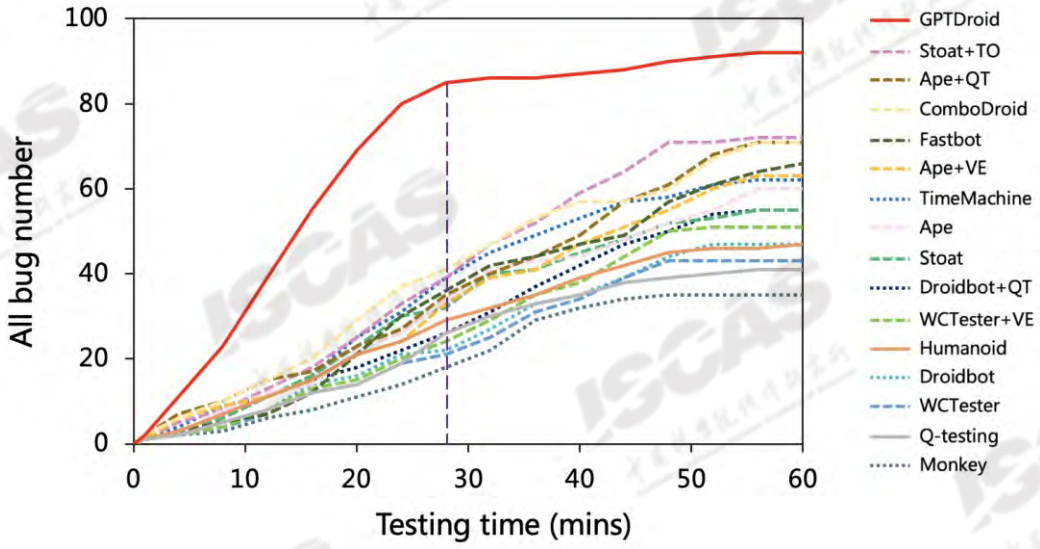


Example 2: For text input operation	
GUI context	**Start Prompt:**[...] **or Test Prompt:**. **or Feedback prompt:** ..
functionality-aware memory	**[List of tested functions:** "Function1: Add ...", ...] **[Path of tested activities:** ...] **[History of latest tested pages ...]**
Function question	What is the functions currently being tested? Are we testing a new function? (<Function name> + <Status>)
Input question	Please generate the input text in sequence, and the operation after input. (<*Widget name*>+ <*Input Content*>, ... and provided <*Operation*>+ <*Widget name*>)
LLM Answer	Function: "Add income". Status: No. Widget: "Price". Input: "3500". Widget: "Title". Input: "salary". Widget: "Category". Input: "personal". Operation: "Click". Widget: "Submit".



Evaluation



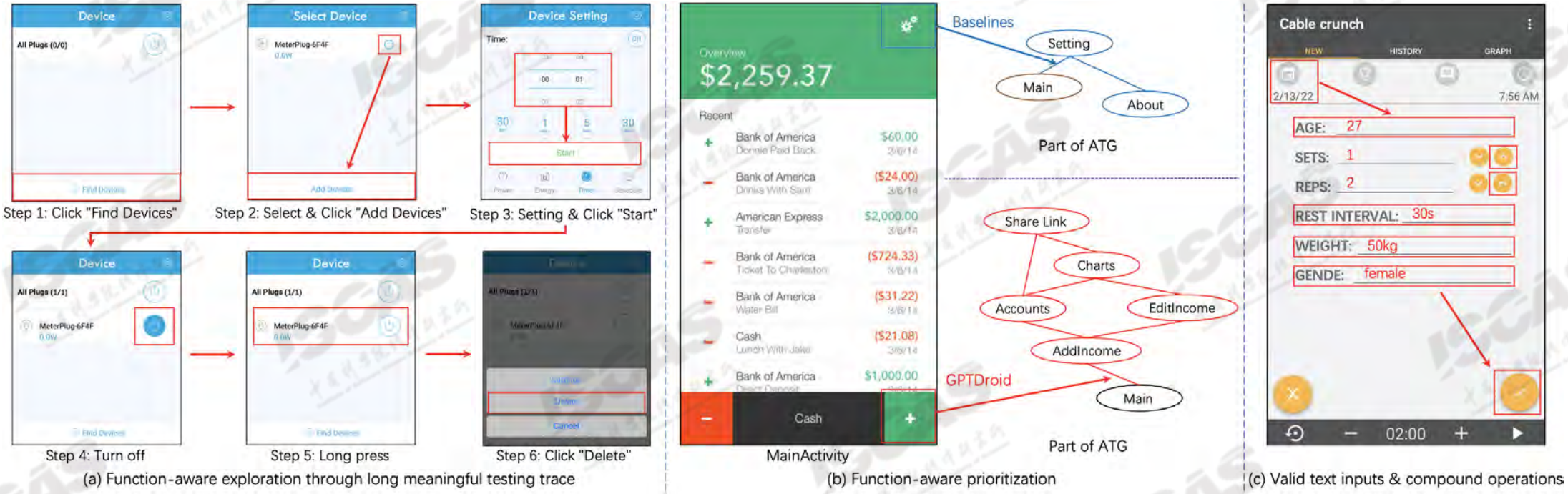


- 75% average activity coverage, 32% higher than the best baselines
- detects 95 bugs for the 93 apps, 31% higher than the best baselines

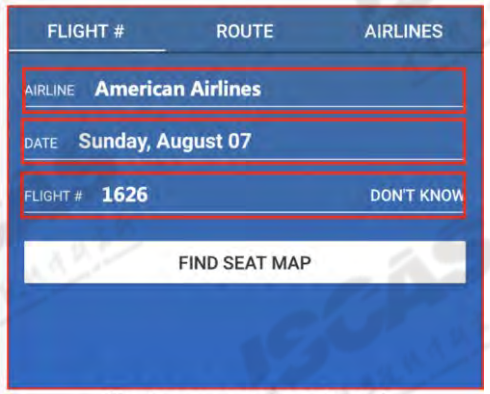
Evaluation

◆ Capability of GPTDroid

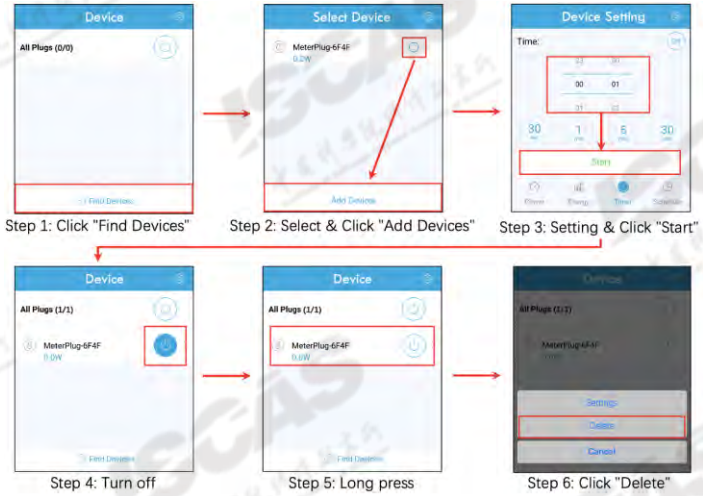
- Function-aware exploration through long meaningful testing trace
- Prioritization
- Valid text inputs & compound operations

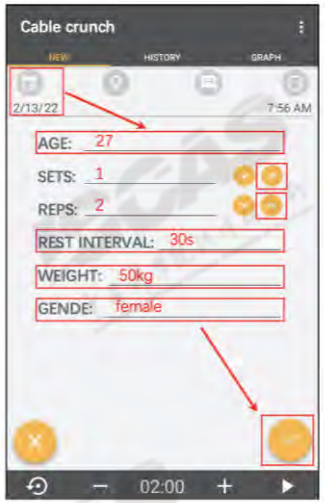


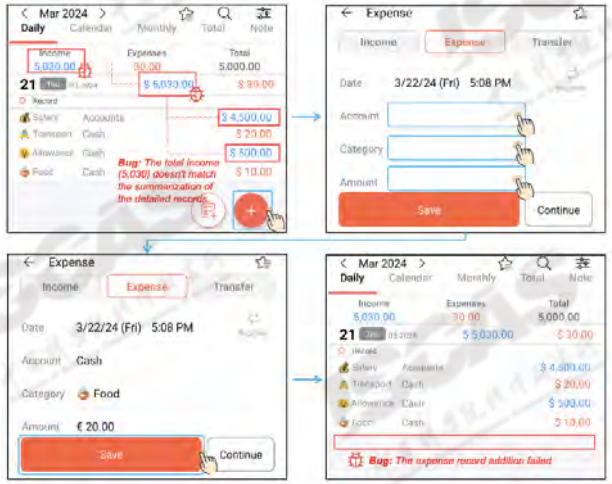
提纲



Find seat map (text input)



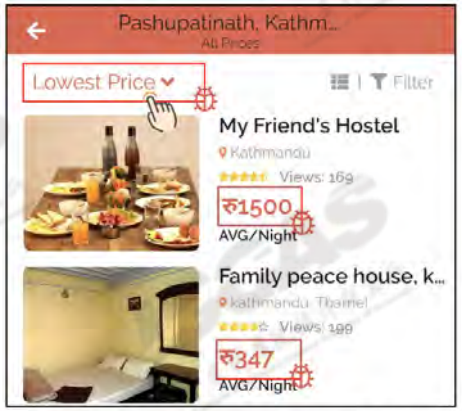




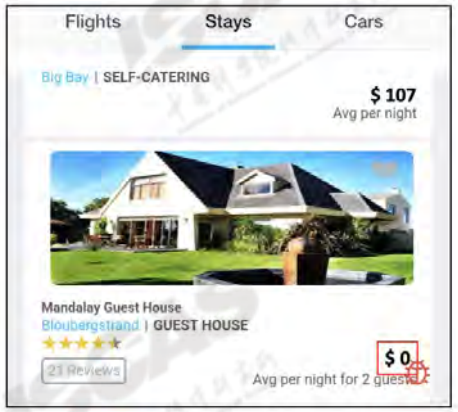
- 用户界面测试现状和挑战
- 测试输入生成技术
- 面向测试路径规划的自动化GUI测试技术
- 基于多模态大模型的自动化GUI测试技术
- 针对文本输入的模糊测试技术
- 面向文本输入组件的交互提升技术

VisionDroid：Vision-driven Automated Mobile GUI Testing

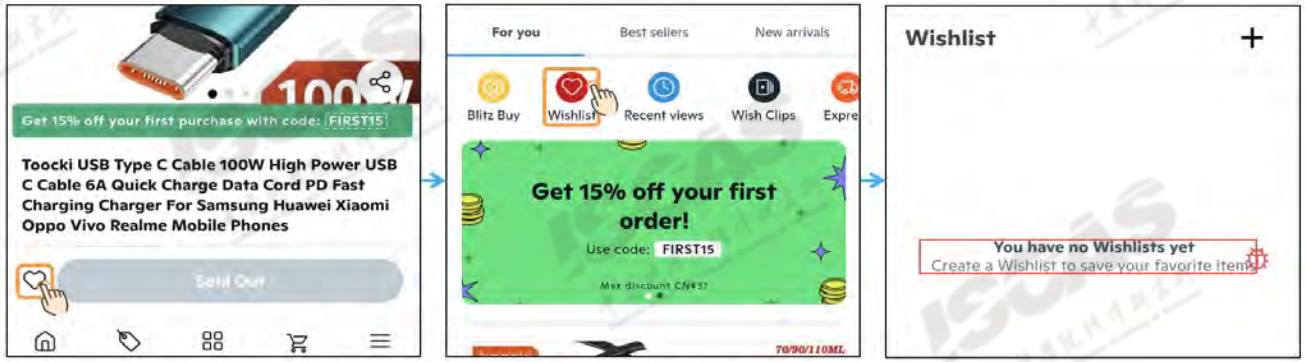
- Crash-bugs vs. Non-crash functional bugs



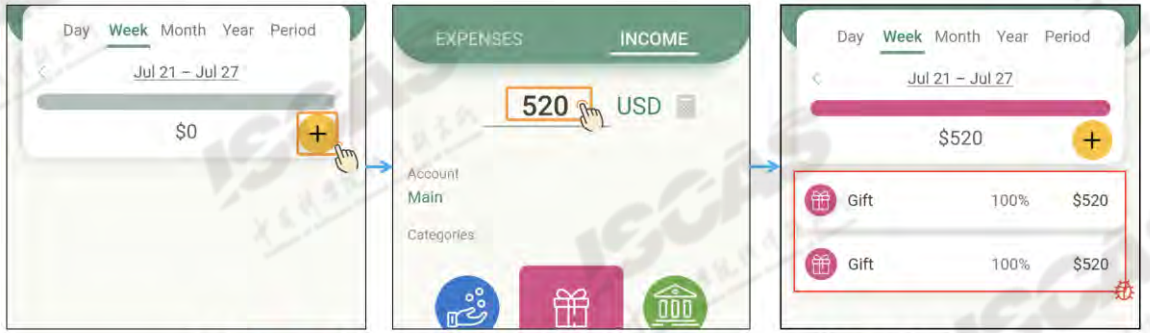
Bug: After sorting the list by the lowest price, the list isn't correctly sorted.



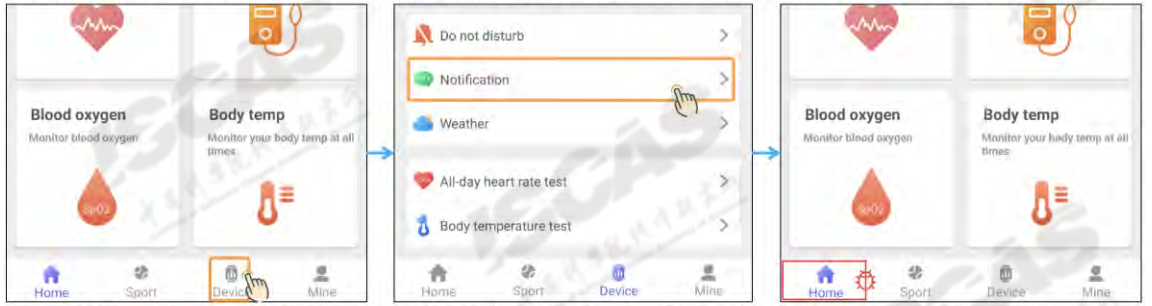
Bug: A listing shows an incorrect price of $0 for a stay.



Bug: After clicking 'Wish', the wishlist is not displayed.

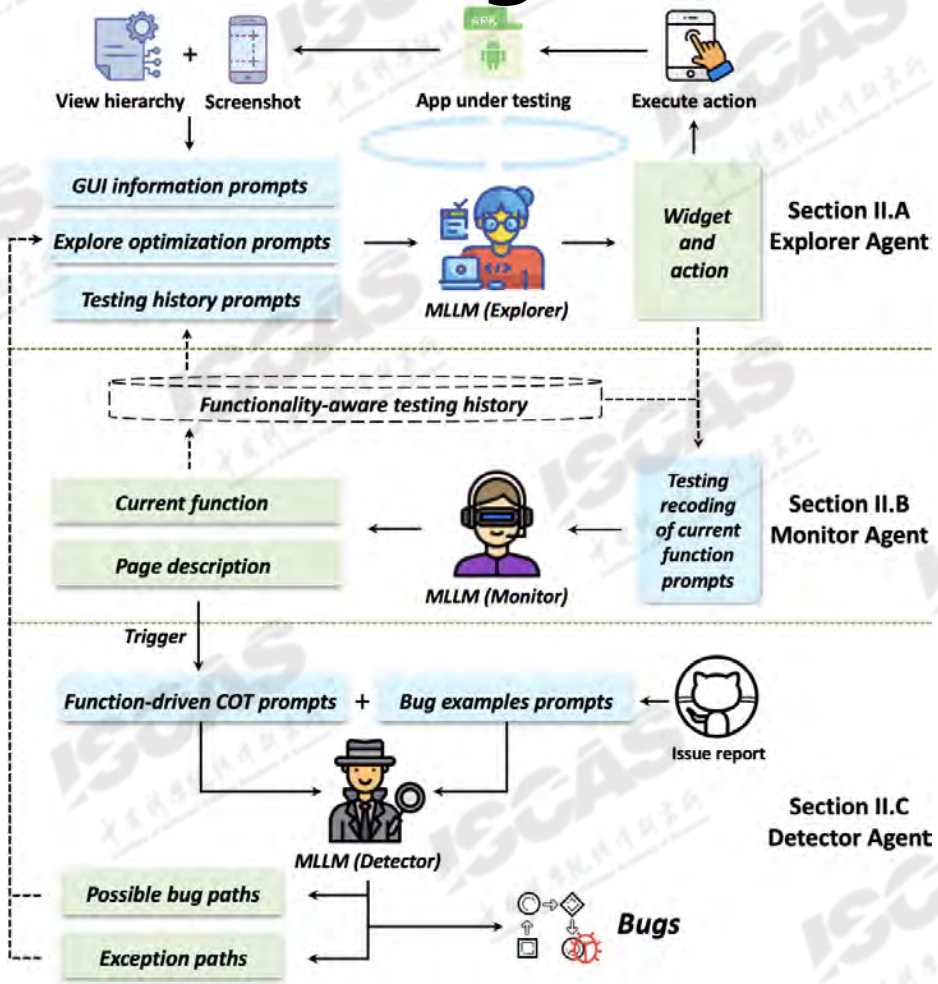


(a) Bug: The "Income" has been repeatedly added



(b) Bug: The link provided by the "Notification" button is incorrect

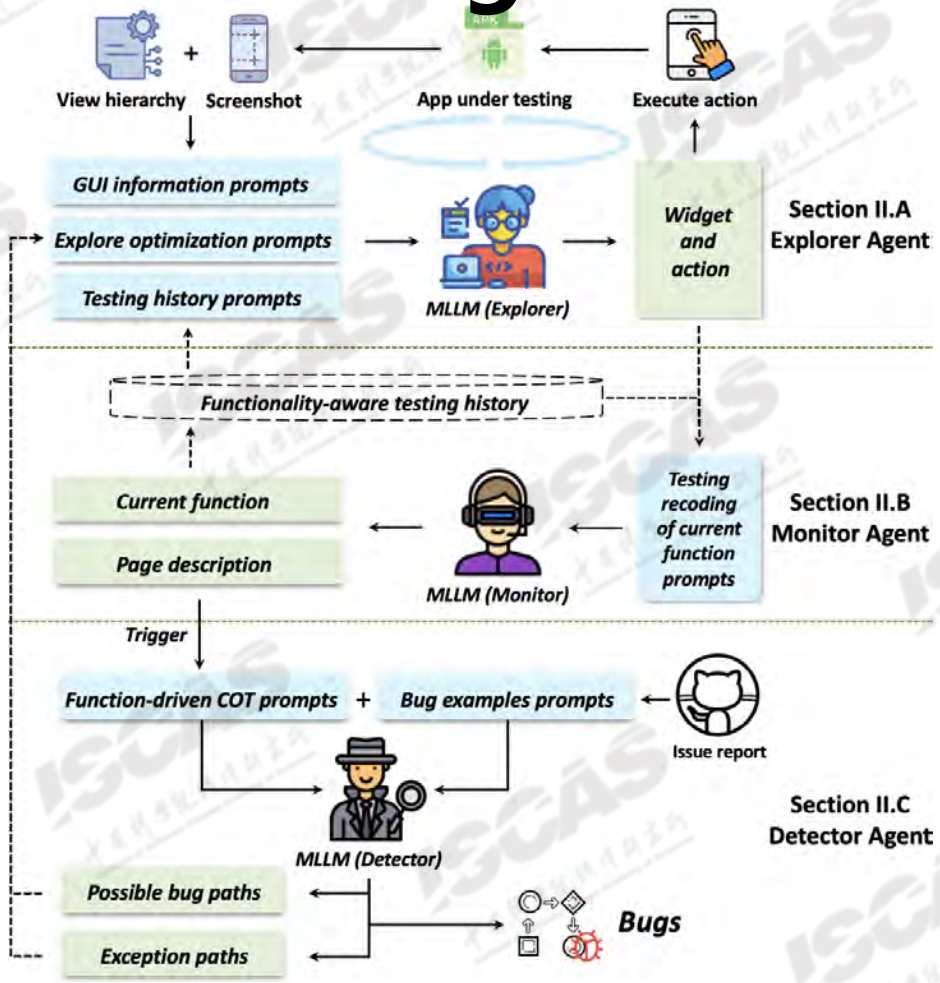
VisionDroid：Vision-driven Automated Mobile GUI Testing



- Vision-driven, multi-agent collaborative automated GUI testing approach for detecting non-crash functional bugs
- Explorer Agent: navigates through the app, captures view hierarchies and screenshots, and guides the exploration towards diverse GUI pages while focusing on the app’s functionalities.
- Monitor Agent: supervises the testing process, records the exploration history, and triggers the detector agent at the appropriate time.
- Detector Agent: identifies potential functional bugs by examining whether there are any issues in the logical transitions that occur during GUI page changes

Vision-driven Automated Mobile GUI Testing via Multimodal Large Language Model, arxiv

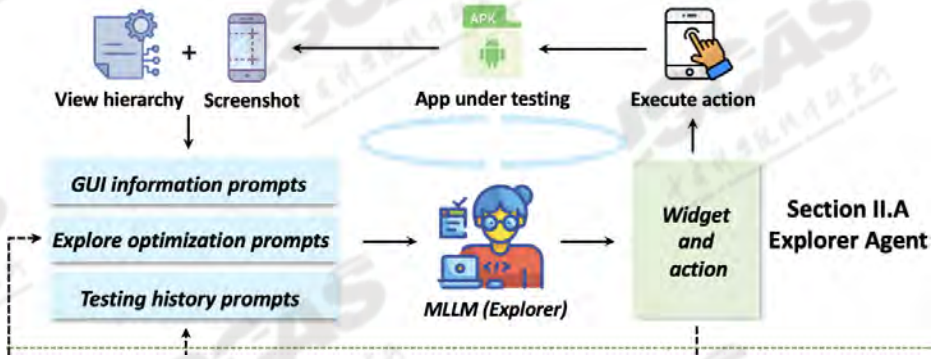
VisionDroid：Vision-driven Automated Mobile GUI Testing



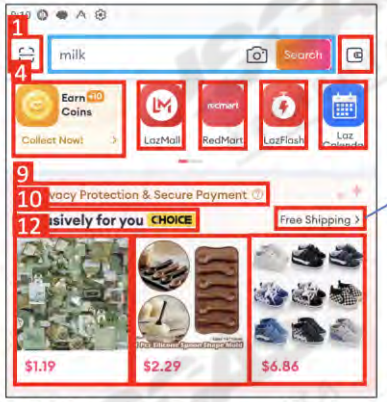
- Challenge 1: Aligning visual and text for MLLM input.
  - Alignment method that integrates text properties with visual context;
  - Screenshot annotation method, pay attention to different types of actionable widgets, resolve issue of overlapping
- Challenge 2: Functionality-oriented exploration.
  - Infer and abstract the current functionality from detailed exploration sequences, avoids exceeding token limits when interacting with the LLM, enable exploration more focusing on the functionality aspect
- Challenge 3: Inferring test oracle.
  - Let the Monitor Agent trigger the Detector Agent at the end of each functionality exploration
  - Functionality-aware Chain-of-Thought (COT) to enable the MLLM to first explicitly infer oracles and then detect functional bugs based on these inference

Vision-driven Automated Mobile GUI Testing via Multimodal Large Language Model, arxiv

Explorer Agent



(a) Direct annotation methods



(b) Our annotation methods

App information
App name: Lazada
Activities: Main, Product, ...

Page information
Acitvity name: Main

Widgets information
Text: Free shipping
Resource-id: id/btn_freeship
Hint-text: null
Clickable: true
Long-clickable: false
Checkable: false
...

(c) Text information

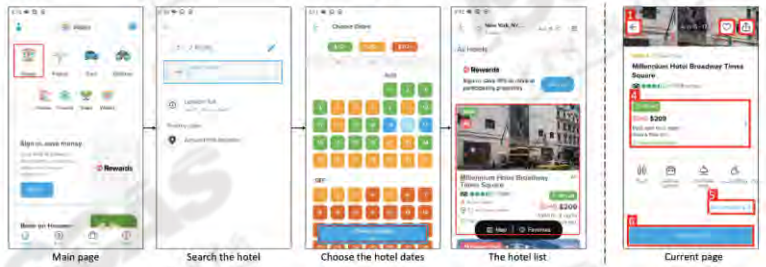
Explorer's Text Prompt Template

Below are instructions for an App exploration task. [Task Description]
[GUI information]:
The App is {App name}. Its activities are {Activities}. The current page is {Activity name}. ***From Section III A***
[Testing history]:
The tested functions and the visit number to each function are as follows:
{Function mame} + {Visit number} … ***From Section III B***
[Explore optimization]:
According to the feedback from the testing process,
{Possible bug path} (e.g., Click the … widget in … page, then click …)
{Exception path} (e.g., The … action is wrong. Click …) ***From Section III C***

[Legend of image]:
{Legend of current GUI screenshots} (e.g., We label the actionable widgets in the first column of each row in order from left to right, … . We also label the types of actionable widgets according to the rules, with red as "click", …) ***From Section III A***
{Legend of recently tested screenshots:} On the left side of the image are the screenshots from the last 4 steps with the labeled action widget. Each screenshot is labeled with the page description of the tested page. … ***From Section III B***

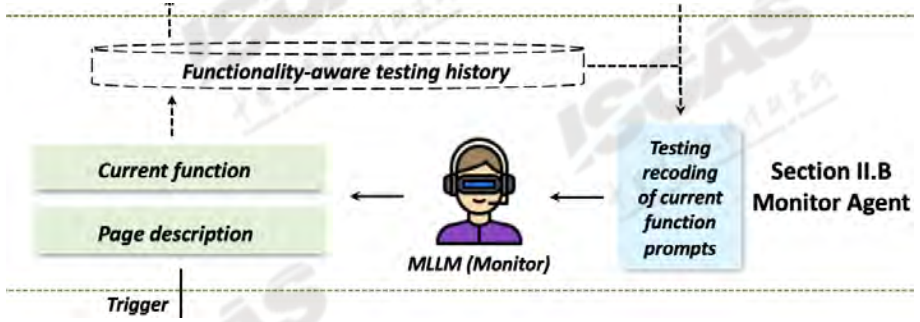
[Query]:
{Querying general action} (e.g., Which numbered widget? What actions are needed? What is the textual description of the widget? ([Widget number] + [Action][click / long-click / check / scroll] + [Widget text]).)
{Querying text input} (e.g., Please give the widget number, generate the input text, and the action after input. ([Widget number]+[Input Content]) provided ([Widget number] + [Action[click]] + [Widget text]).)

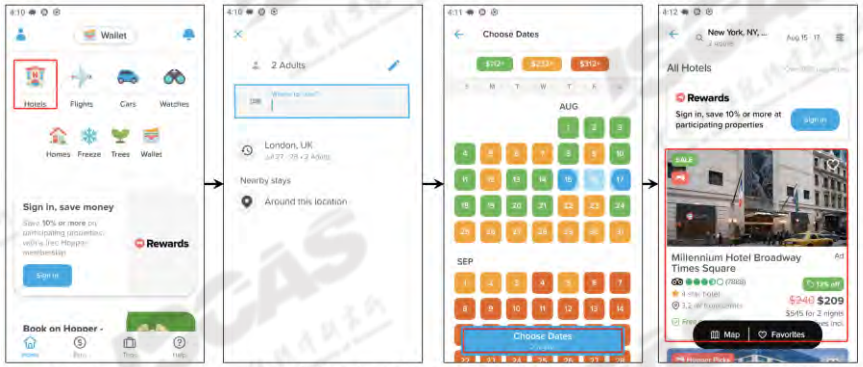
+



Explorer's Image Prompt

Monitor Agent





Explored functions

Funtion-1: Main page
Visits-1: 13
Funtion-2: Search the flight
Visits-2: 7
Funtion-3: Search the hotel
Visits-3: 5
Funtion-4: App settings
Visits-4: 4

Funtion-n: ...
Visits-n: ...
...

(b) Test view

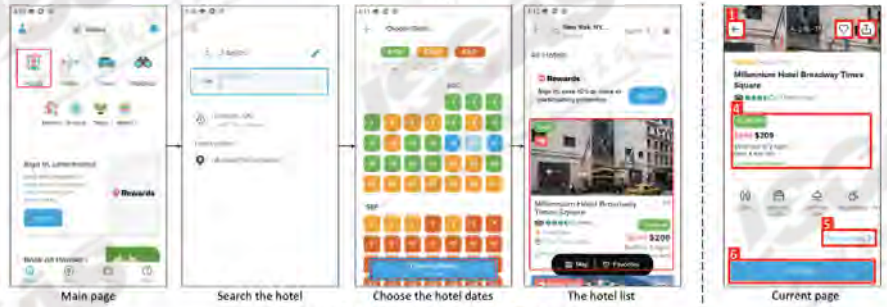
Monitor's Text Prompt Template

Below are instructions for a testing monitoring task. [Task Description]
[Testing recording of current function]:
The function being tested in the previous step is {Function name}, and the action is {Action}.
[Testing history]:
The tested functions and the visit number to each function are as follows:
{Function mame} + {Visit number} ...

[Legend of image]:
{Legend of recently tested screenshots:} On the left side of the image are the screenshots from the last 4 steps with the labeled action widget. Each screenshot is labeled with the function name of the tested page. ...

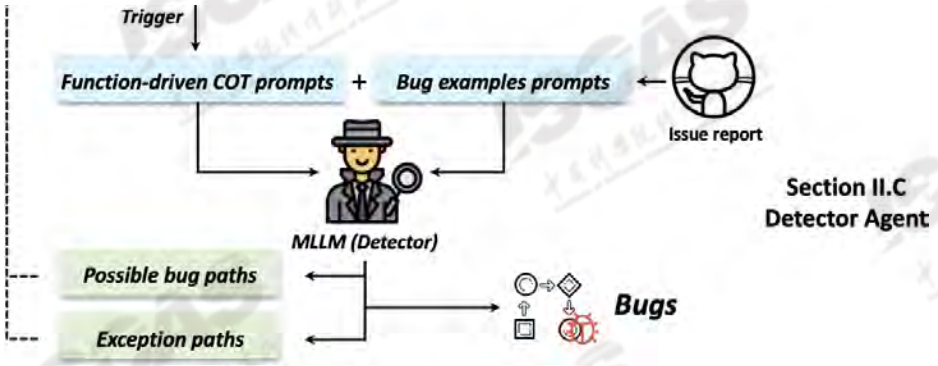
[Query]:
{Querying page description} (e.g., What is the page description? ([Page description])
{Querying function} (e.g., Please make a judgment based on the GUI information on the page. What is the function currently being tested? Are we testing a new function? ([Function name] + [Status]).)

+



Monitor's Image Prompt

Detector Agent



Section II.C
Detector Agent

Detector's Text Prompt Template

Below are instructions for the no-crash bug detection task. [Task Description]
[Bug examples]:
The categories of bugs include {Bug category}. We provide the bug examples:
(1){Bug description} + {Bug replay path} …
[Function-driven COT]:
Let's think step by step.
1. Function Identification: The tested function {Function name} … . Its description is ...
2. Expected Path: Please predict the exploration steps for this fucntion based on above information.
3. Actual Path: The tested function sequence is as follows:
(1) The 1st page is {Page description} + {action}
(2) The 2st page is {Page description} + {action}
(3) ….

[Legend of image]:
The image shows a test sequence arranged in order from left to right and from top to bottom. Each screenshot is labeled with a different colored bounding box indicating the action of the operation (red as "click", blue as "input",...). Below the screenshot is the corresponding page description.

[Query]:
{Querying bug detection} (e.g., Please analyze each step in the test sequence in the image based on the description and examples of the bugs, predict whether the page that transits after each step meets your expectation, use this to determine whether there are any bugs, and point out the bug page.)
{Querying possible bug path} (e.g., Is there any page or operation that could potentially trigger some bugs? Please provide the corresponding page and action widget.)
{Querying exception path} (e.g., By analyzing the test path, if there are any pages or operations in the path that have abnormalities? Please provide the corresponding page and action widget.)

+

Detector's Image Prompt

- Enriching Detector Prompt with Example
  - bug description, bug screenshot and natural language described bug reproduction path which facilitates the MLLM understanding of what the non-crash functional bugs are

Evaluation

TABLE I: Result of coverage and bug detection (RQ1)

Method	Baseline data						GitHub data						Bug injection data					
	Act	Cod	Rep	Bug	P	R	Act	Cod	Rep	Bug	P	R	Act	Cod	Rep	Bug	P	R
OwlEyes+M	0.21	0.18	77	10	0.13	0.08	0.33	0.28	140	62	0.44	0.16	0.35	0.31	96	13	0.14	0.18
NightHawk+M	0.21	0.18	63	13	0.21	0.10	0.33	0.28	136	62	0.46	0.16	0.35	0.31	60	15	0.25	0.21
DiffDroid+M	0.21	0.18	61	15	0.25	0.11	0.33	0.28	133	55	0.41	0.14	0.35	0.31	46	12	0.26	0.17
dVermin+M	0.21	0.18	53	4	0.08	0.03	0.33	0.28	98	28	0.29	0.07	0.35	0.31	33	11	0.33	0.15
OwlEyes+A	0.29	0.27	92	18	0.20	0.14	0.43	0.37	161	85	0.53	0.22	0.42	0.4	93	10	0.11	0.14
NightHawk+A	0.29	0.27	86	21	0.24	0.16	0.43	0.37	157	91	0.58	0.24	0.42	0.4	67	16	0.24	0.22
DiffDroid+A	0.29	0.27	87	20	0.23	0.15	0.43	0.37	147	68	0.46	0.18	0.42	0.4	44	7	0.16	0.10
dVermin+A	0.29	0.27	67	6	0.09	0.05	0.43	0.37	101	31	0.31	0.08	0.42	0.4	35	9	0.26	0.13
OwlEyes+F	0.39	0.37	107	17	0.16	0.13	0.47	0.42	180	79	0.44	0.20	0.43	0.39	94	12	0.13	0.17
NightHawk+F	0.39	0.37	100	20	0.20	0.15	0.47	0.42	161	92	0.57	0.24	0.43	0.39	59	15	0.25	0.21
DiffDroid+F	0.39	0.37	94	20	0.21	0.15	0.47	0.42	131	72	0.55	0.19	0.43	0.39	53	12	0.23	0.17
dVermin+F	0.39	0.37	41	8	0.20	0.06	0.47	0.42	90	35	0.39	0.09	0.43	0.39	67	8	0.12	0.11
OwlEyes+H	0.38	0.35	97	12	0.12	0.09	0.48	0.44	157	79	0.50	0.20	0.43	0.39	93	14	0.15	0.19
NightHawk+H	0.38	0.35	88	16	0.18	0.12	0.48	0.44	146	79	0.54	0.20	0.43	0.39	88	13	0.15	0.18
DiffDroid+H	0.38	0.35	75	17	0.23	0.13	0.48	0.44	143	57	0.40	0.15	0.43	0.39	57	15	0.26	0.21
dVermin+H	0.38	0.35	33	5	0.15	0.04	0.48	0.44	82	40	0.49	0.10	0.43	0.39	41	8	0.20	0.11
OwlEyes+G	0.47	0.43	99	17	0.17	0.13	0.51	0.47	162	79	0.49	0.20	0.47	0.41	95	14	0.15	0.19
NightHawk+G	0.47	0.43	95	18	0.19	0.14	0.51	0.47	157	96	0.61	0.25	0.47	0.41	68	12	0.18	0.17
DiffDroid+G	0.47	0.43	85	16	0.19	0.12	0.51	0.47	142	76	0.54	0.20	0.47	0.41	47	9	0.19	0.13
dVermin+G	0.47	0.43	37	6	0.16	0.05	0.51	0.47	75	47	0.63	0.12	0.47	0.41	42	9	0.21	0.13
iFixdataloss	0.33	0.3	21	8	0.38	0.06	0.48	0.43	101	45	0.45	0.12	0.43	0.39	41	10	0.24	0.14
SetDroid	0.31	0.27	22	8	0.36	0.06	0.48	0.43	164	64	0.39	0.17	0.43	0.39	47	11	0.23	0.15
Genie	0.32	0.31	55	12	0.22	0.09	0.48	0.43	123	46	0.37	0.12	0.43	0.39	43	8	0.19	0.11
Odin	0.33	0.3	57	22	0.39	0.17	0.48	0.43	107	59	0.55	0.15	0.43	0.39	57	11	0.19	0.15
Lint	-	-	1	1	1.00	0.01	-	-	54	30	0.56	0.08	-	-	2	2	1.00	0.03
LiveDroid	-	-	1	1	1.00	0.01	-	-	58	24	0.41	0.06	-	-	1	1	1.00	0.01
GPT-4	0.39	0.31	55	12	0.22	0.09	0.47	0.45	123	48	0.39	0.12	0.43	0.39	35	8	0.23	0.11
AppAgent	0.41	0.38	47	6	0.13	0.05	0.44	0.43	113	12	0.11	0.03	0.39	0.37	31	4	0.13	0.06
Trident	**0.57**	**0.55**	**109**	**55**	**0.50**	**0.42**	**0.65**	**0.62**	**278**	**201**	**0.72**	**0.52**	**0.61**	**0.59**	**67**	**47**	**0.70**	**0.65**

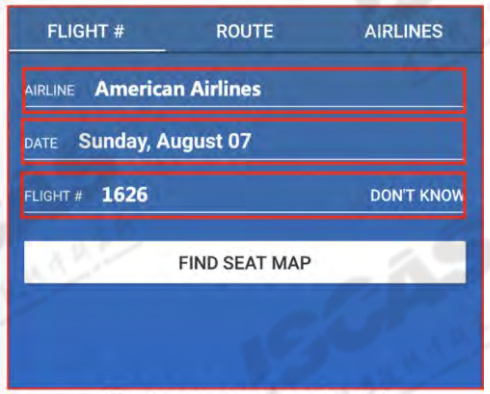
Notes: "Act" is activity coverage, "Cod" is Code coverage. "Rep" is number of bugs reported by tools, "Bug" is true bug in the reported bug, "P" is precision and "R" is the recall. "+M" means it with Monkey, "+A" means it with APE, "+F" means it with Fastbot, "+H" means it with Hunmanoid, and "+G" means it with GPTDroid.

- 50%-72% precision and 42%-65% recall
- more than 14%-112% and 108%-147% boost in average recall and precision compared with the best baseline

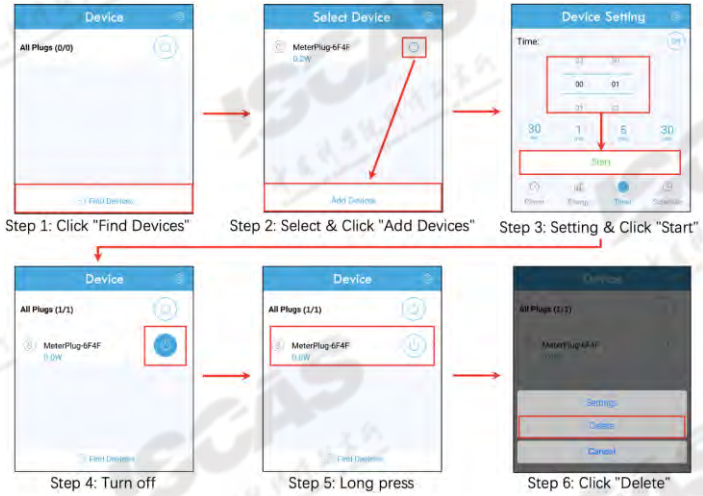
面向自然语言描述的测试用例迁移

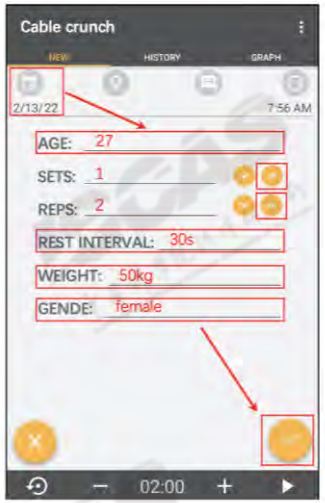
- **测试需求：**我家-添加资产流程

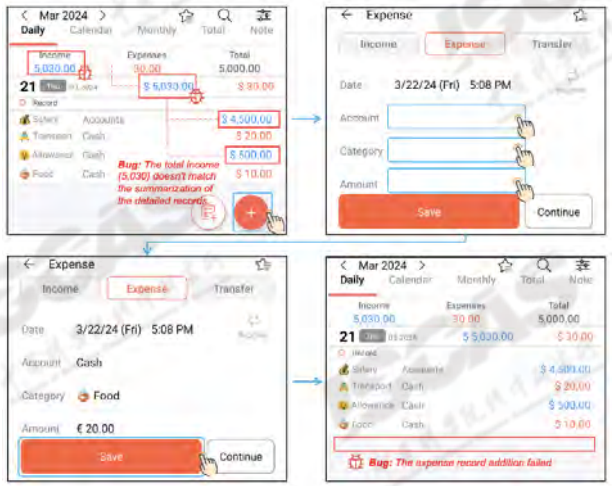
提纲



Find seat map (text input)

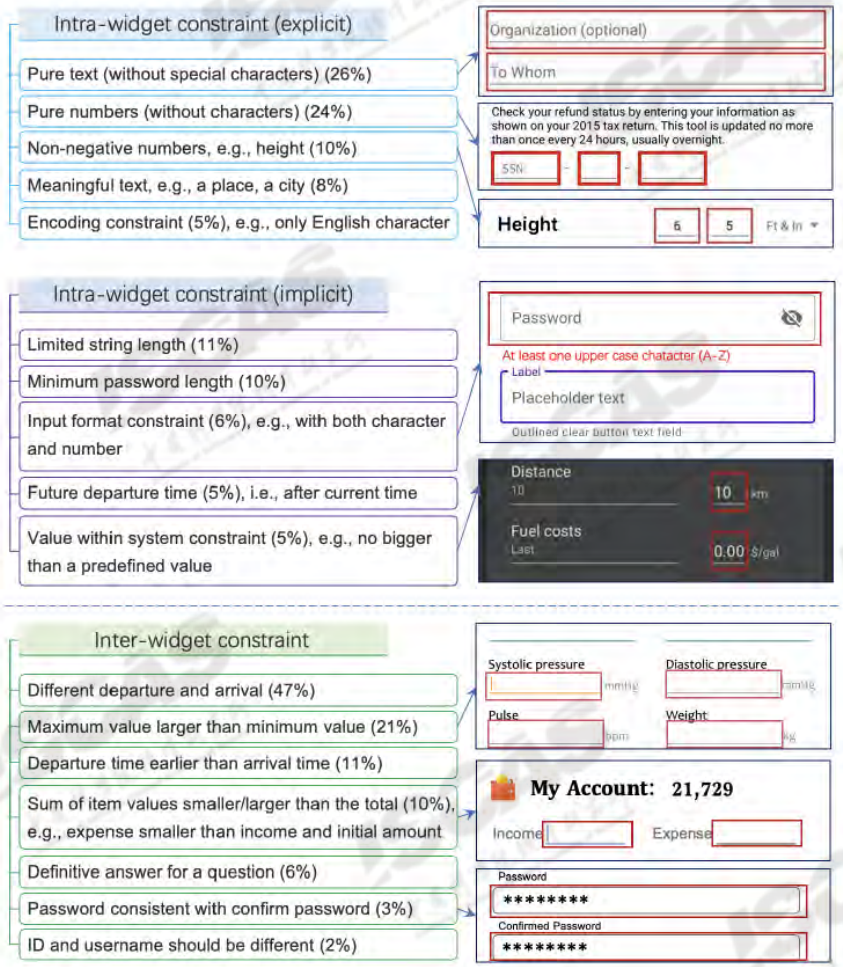






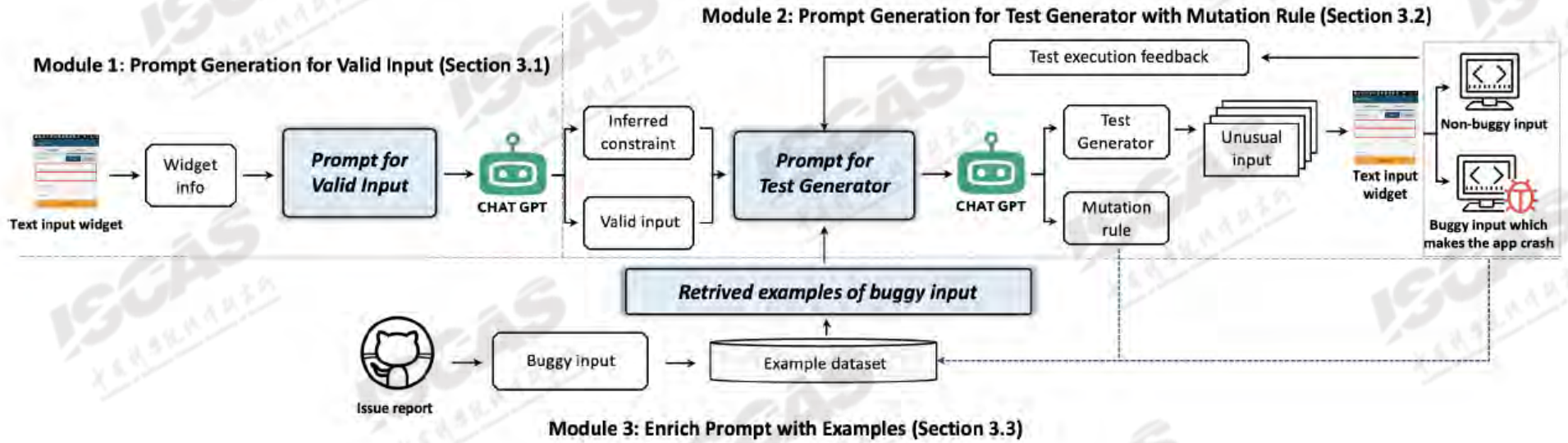
- 用户界面测试现状和挑战
- 测试输入生成技术
- 面向测试路径规划的自动化GUI测试技术
- 基于多模态大模型的自动化GUI测试技术
- 针对文本输入的模糊测试技术
- 面向文本输入组件的交互提升技术

面向文本输入的模糊测试



- **Intra-widget constraint:** requirements of a single text input, e.g., a widget for a human’ s height can only input the non-negative number.
- **Inter-widget constraint:** requirements among multiple text input widgets on a GUI page, for example, the diastolic pressure should be less than systolic pressure.

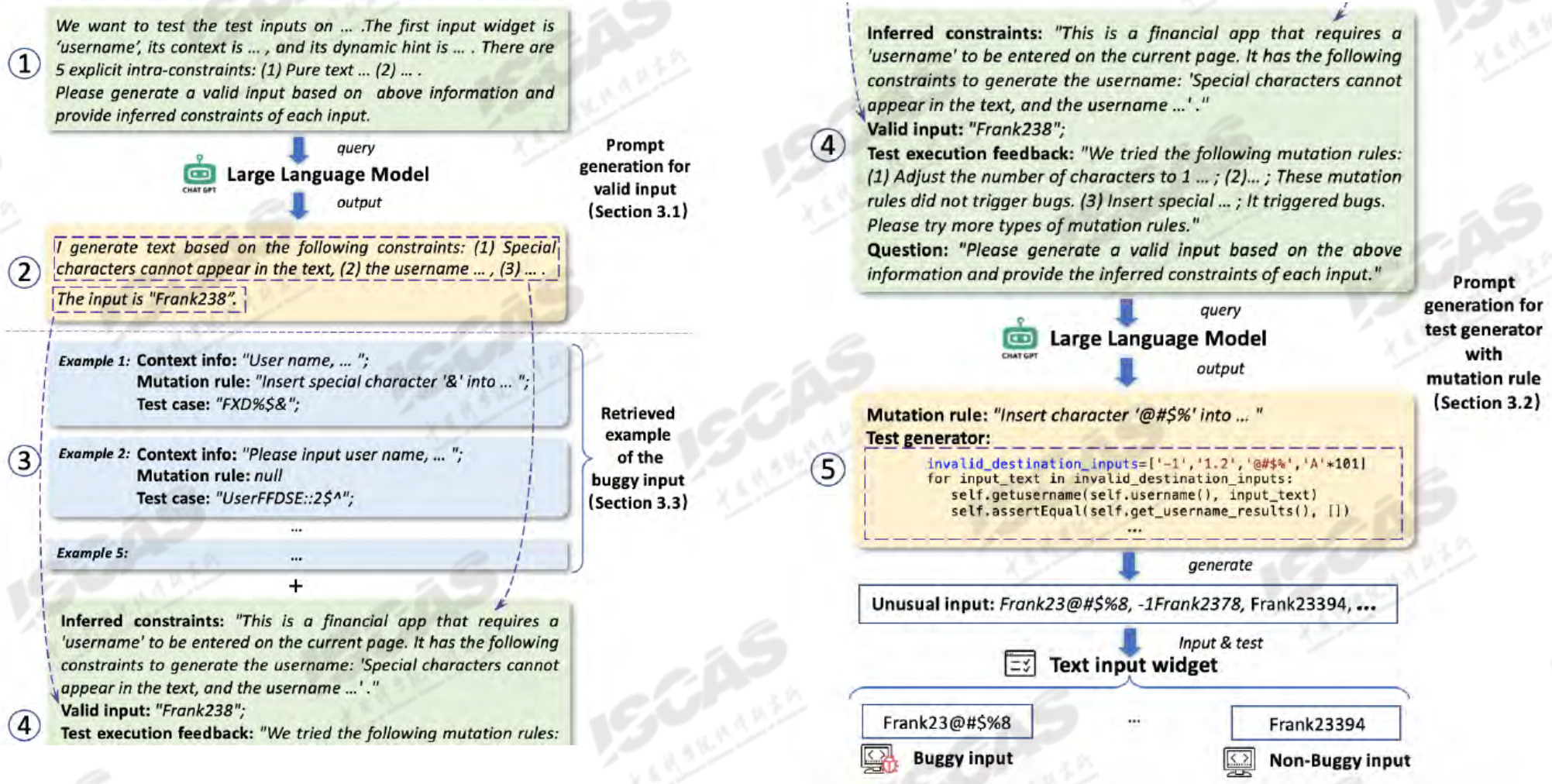
Unusual Text Inputs Generation



- produce the test generators (a code snippet) with LLM
- Each can generate a batch of unusual text inputs under the same mutation rule (e.g., insert special characters into a string)

Testing the Limits: Unusual Text Inputs Generation for Mobile App Crash Detection with Large Language Model, ICSE 2024

Example prompt

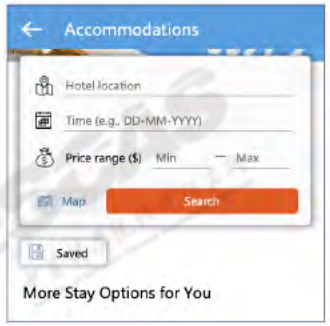


Evaluations

Method	Setting 1 (30 attempts) Bug(%)	Setting 1 (30 attempts) Attempt(#)	Setting 2 (30 minutes) Bug(%)	Setting 2 (30 minutes) Min(#)
InputBlaster	**0.72**	**13.52**	**0.78**	**9.64**
ChatGPT	0.25	25.91	0.28	23.28
Mutation or fuzzing methods				
GoldTest	0.08	29.22	0.08	28.73
PDinvalid	0.19	28.65	0.19	22.73
RandomFuzz	0.25	22.31	0.25	21.55
ruleMutator	0.28	21.42	0.28	20.53
String analysis methods				
Sloth	0.25	23.61	0.25	22.61
OSTRICH	0.22	24.14	0.22	23.41
Constraint-based methods				
Mobolic	0.17	25.83	0.17	25.09
TextExerciser	0.31	22.11	0.33	20.18
Valid input generation methods				
RNNInput	0.06	28.67	0.06	28.64
QTypist	0.08	27.78	0.11	27.31
Automated GUI testing methods				
Ape	0.08	28.11	0.11	26.88
DroidBot	0.06	28.39	0.06	28.34
Stoat	0.08	27.94	0.08	27.58
TimeMachine	0.11	26.92	0.11	26.69
ComboDroid	0.14	26.11	0.14	25.85
Q-testing	0.11	27.06	0.11	26.70
Humanoid	0.11	26.92	0.14	25.85

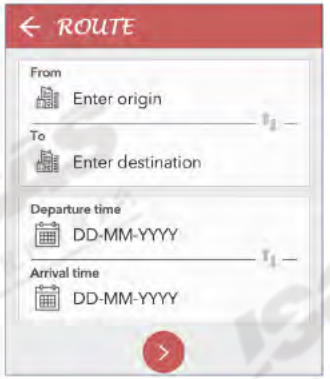
Notes: "Bug (%)" is the average bug detecting rate, "Attempt (#)" is the average number of unusual inputs before triggering the crash, "Min (#)" is the average running time (minutes) before triggering the crash.

- 72% - 78% found bugs, significant higher than baselines
- Significant few attempt times



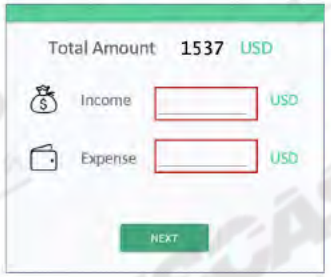
Valid input: Melbourne Skydeck, 01-02-2023, 110, 260

Unsual input 1: Melbourne-1234, 01*-0@2-21, 1100, 2609
Unsual input 2: Sydney, 30-33-84, abc1, 1394
Unsual input 3: Perth16, 193-%4, -1, 44
Unsual input 4: New York, 1991-03, 2.2.2, 193$
...
Buggy input: ^&###?>Dmep-d3, 03-07-2024, 308, 22



Valid input: New York, Washington D.C., 01-04-2023, 09-04-2023

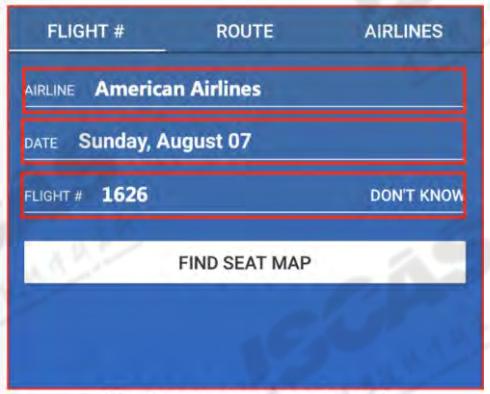
Unsual input 1: New York, Pairs, 011-04-201223, 0$9-0%4-20-23
Unsual input 2: NewYorkss, Pairs&S%, 00-88-MC23, 19-34-211
Unsual input 3: New York, New York, -3.1-12-1933, 2531-041-1911
...
Buggy input: Washington, Houston, 09-04-4099, 01-04-1121



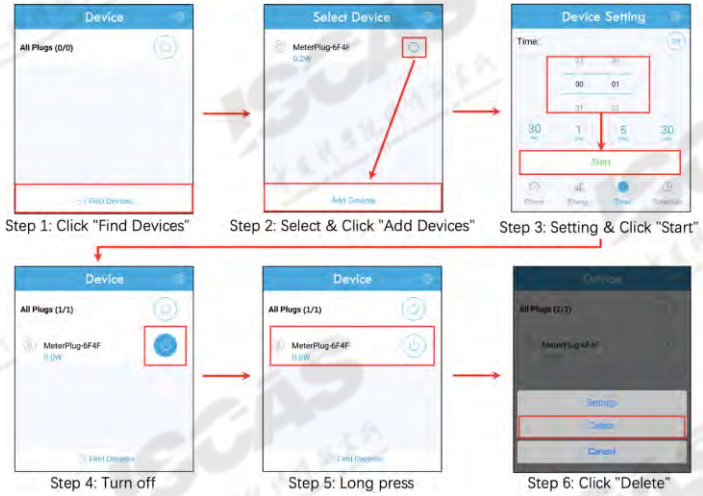
Valid input: 130, 70

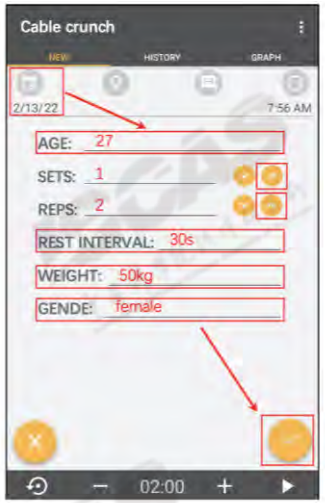
Unsual input 1: 13.1, 40.7
Unsual input 2: 9000, 9000
Unsual input 3: 130$, 137*&^
Unsual input 4: aasec, franm#
...
Buggy input: 30, 1878

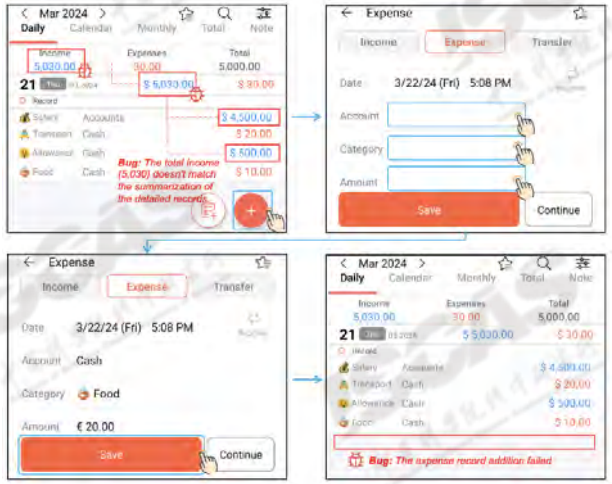
提纲



Find seat map (text input)







- 用户界面测试现状和挑战
- 测试输入生成技术
- 面向测试路径规划的自动化GUI测试技术
- 基于多模态大模型的自动化GUI测试技术
- 针对文本输入的模糊测试技术
- 面向文本输入组件的交互提升技术

Good Examples of Text Input

Examples of differences

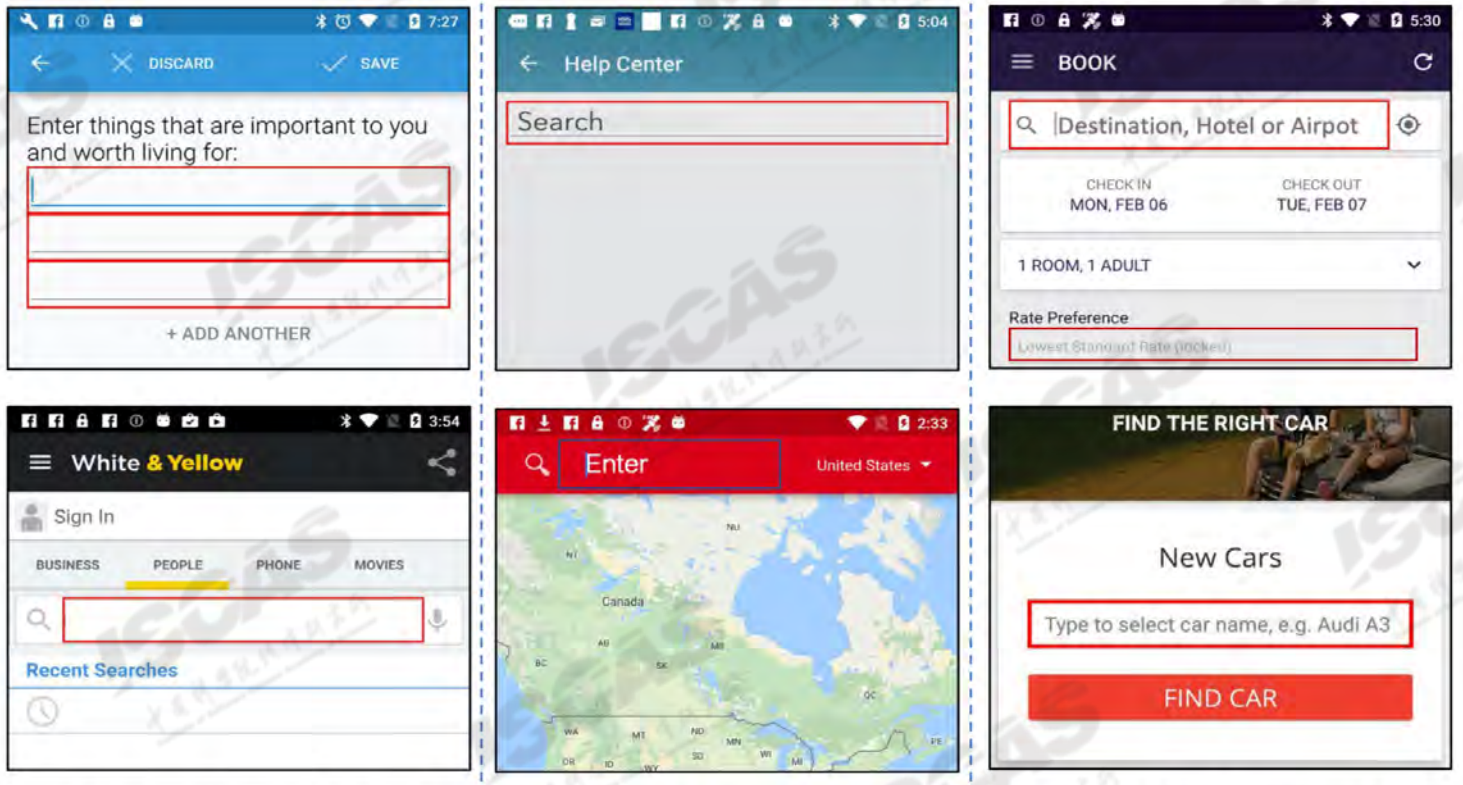
- ✓ hint-text
- ✓ label
- ✓ context description

(a) **Label of image component**: *Send* (b) **Content description**: *Lease Rate* (c) **Hint-text**: *Please provide product ...*

Bad Examples of Text Input

Without hint-text

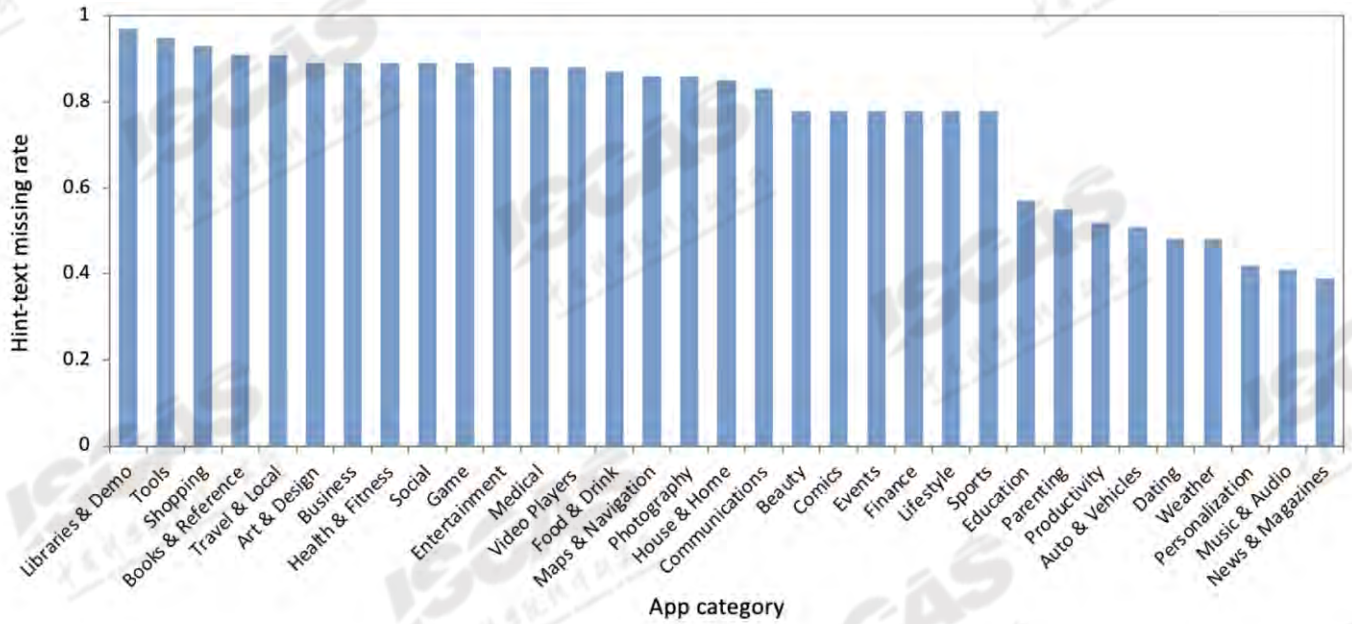
- ✓ screen reader cannot obtain information
- ✓ simple and lacks meaning



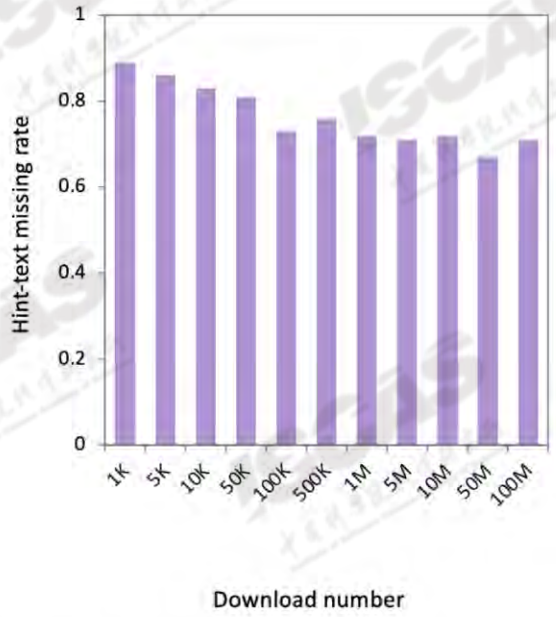
(a) Text input without *hint-text* (b) Text input without meaningful *hint-text* (c) Text input with meaningful *hint-text*

Motivation Study

- ✓ **Dataset:** 4,950 apps from 33 categories from Google Play
- ✓ 3,398 (76%) of them are without hint-text
- ✓ 30,226 (66%) had at least one text input without explicit hint-text content



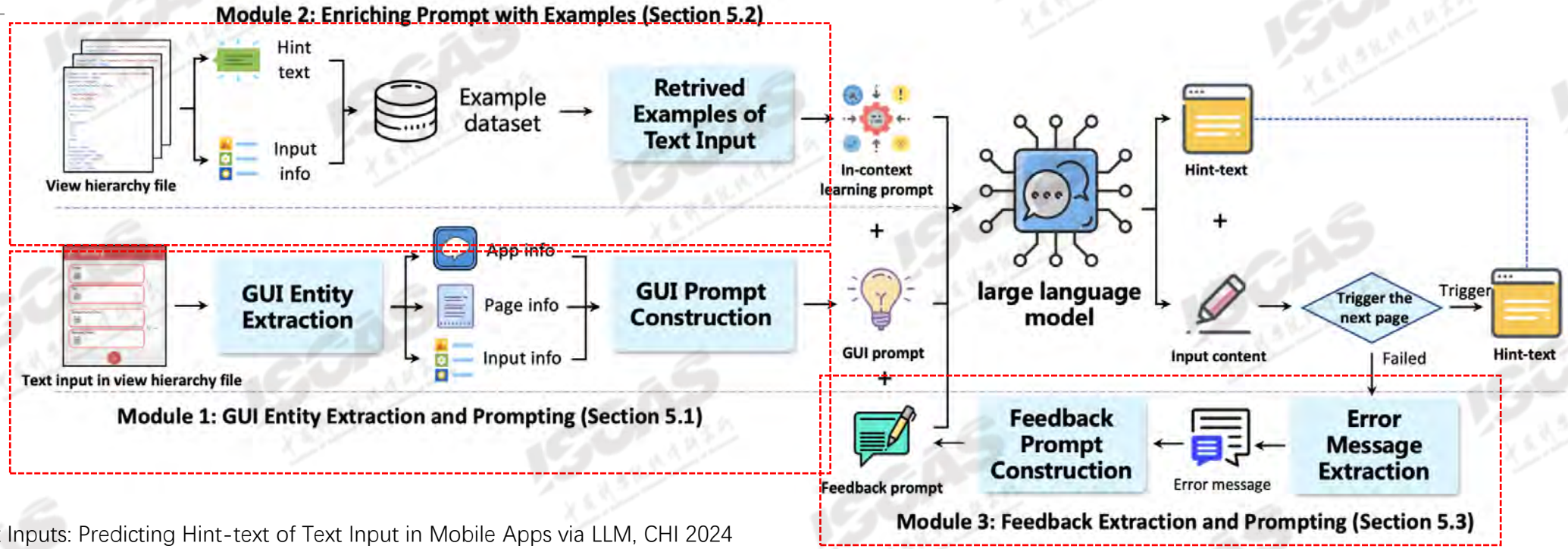
(a) Hint-text missing rate with different category of apps



(b) Hint-text missing rate with download number

HintDroid: Predicting Hint-text of Text Input

- **HintDroid: Predicting Hint-text of Text Input**
  - GUI Entity Extraction and Prompting
  - Enriching Prompt with Examples
  - Feedback Extraction and Prompting



Unblind Text Inputs: Predicting Hint-text of Text Input in Mobile Apps via LLM, CHI 2024

Module 1: GUI Entity Extraction and Prompting

- **GUI Entity Extraction and Prompting**
  - ✓ App Entity Information + Page GUI Entity Information
  - ✓ Input Component Entity information

Type	Entity	Description	Instantiation
App information	[AppName]	Name of the app under testing	[AppName]:"Flight"
	[Activities]	List of names for all activities of the app, obtained from *AndroidManifest.xml* file	[Activities]: "Main, OneWay, RoundTrip, ..."
Page GUI information	[ActivityName]	Activity name of the current GUI page	[ActivityName]: "RoundTrip"
	[Component]	List of all widgets in current page, represented with text/id	[Component]: "Depart, Arrive, Departure time, ..."
	[Position]	Relative position of widgets, obtained through their coordinates	[Position]: Upper: "Flight Search, ...", Lower: "Departure time, ... "
Input component information	[TextInput]	The text input denoted with the textual related fields	[TextInput]:"Departure time"
	[NearbyInput]	Nearby widgets denoted with their textual related fields	[NearbyInput] : "Flight Search, ... "

Module 1: GUI Entity Extraction and Prompting

- **Example of the GUI prompt construction**

Id	Prompt Type	Instantiation	Examples
		In-context learning prompt	
1	<Hint-text examples>	We will provide you with 6 examples: 1. [TextInput], [NearbyInput], [Hint-text] 2. [TextInput], [NearbyInput], [Hint-text] ... 6. [TextInput], [NearbyInput], [Hint-text]	We will provide you with 6 examples: **1st** text input is "From", its nearby components are "...", its hint-text is ... **2nd** text input is "To", its nearby components are ..., its hint-text is ... ... **6th** text input is "Flight", its nearby ..., its hint-text is "Enter the city".
		GUI prompt	
2	<App info>	[AppName], [Activities]	The app name is "Flight", it has following activities: "Main, ..."
3	<Page GUI info>	[ActivityName], [Component], [Position]	The current GUI page is "SearchFlight", it has following components:"Search, ...", the upper part of the page is "...", the lower part ....
4	<Input component info>	[TextInput], [NearbyInput]	The text input of this page is "Depart", its nearby components are ... .
		Feedback prompt	
5	<Feedback>	[Feedback], [ErrorMessage]	The input content "train" doesn't pass the page, the error message of the input component is: "Please enter the correct city name".
		Query & feedback query	
6	<Query>	Please generate a hint-text for the input component based on the above information, and generate corresponding input content based on the generated hint-text.	
7	<Feedback Query>	Please regenerate the hint text and its corresponding input content based on the feedback information above.	
		Example output	
8	<Example output>	[Hint-text], [InputContent]	Please output according to the following example: the hint-text is "xxx", the input content is "xxx".

Module 2: Enriching Prompt with Examples

- **Enriching Prompt with Examples**
  - ✓ Example Dataset Construction
  - ✓ Retrieval-based Example Selection and In-context Learning

Text input in view hierarchy file

In-context learning prompt: *We'll provide you with 6 example:*
1st text input is "From", its nearby components are "Search the Train, Departure time, Submit" , its hint-text is "Enter the Depart".
2nd text input is "Depart", its nearby components are "Search trains, Search data, Show Trains" , its hint-text is "Departure city".
3rd text input is "Depart Airport", its nearby components are "Departing Airport, Depart time, OK" , its hint-text is "Enter the city".
...
5th text input is "Stop/station", its nearby components are "Train, Train num/-name, Date, Time, Search", its hint-text is "City".
6th text input is "Arrive", its nearby components are "Filter, Flight" , its hint-text is "City name".

GUI prompt: *The app name is "Flight", it has following activities: "Main, OneWay, RoundTrip, Setting, Help, News, About, ... ".*
The current GUI page is "RoundTrip", it has following components "Flight Search, txtDepart, txtArrival, Airline, Depart Date ... ",
the upper part of the page is "Flight Search, txtDepart, txtArrival, Airline" , the lower part of the page is "Depart Date ... " .
The text input of this page is "Depart", its nearby components are "Flight Search, txtArrival, Airline" ...

Feedback prompt: *The input content doesn't pass the page, the error message is "Please enter the correct city name" .*

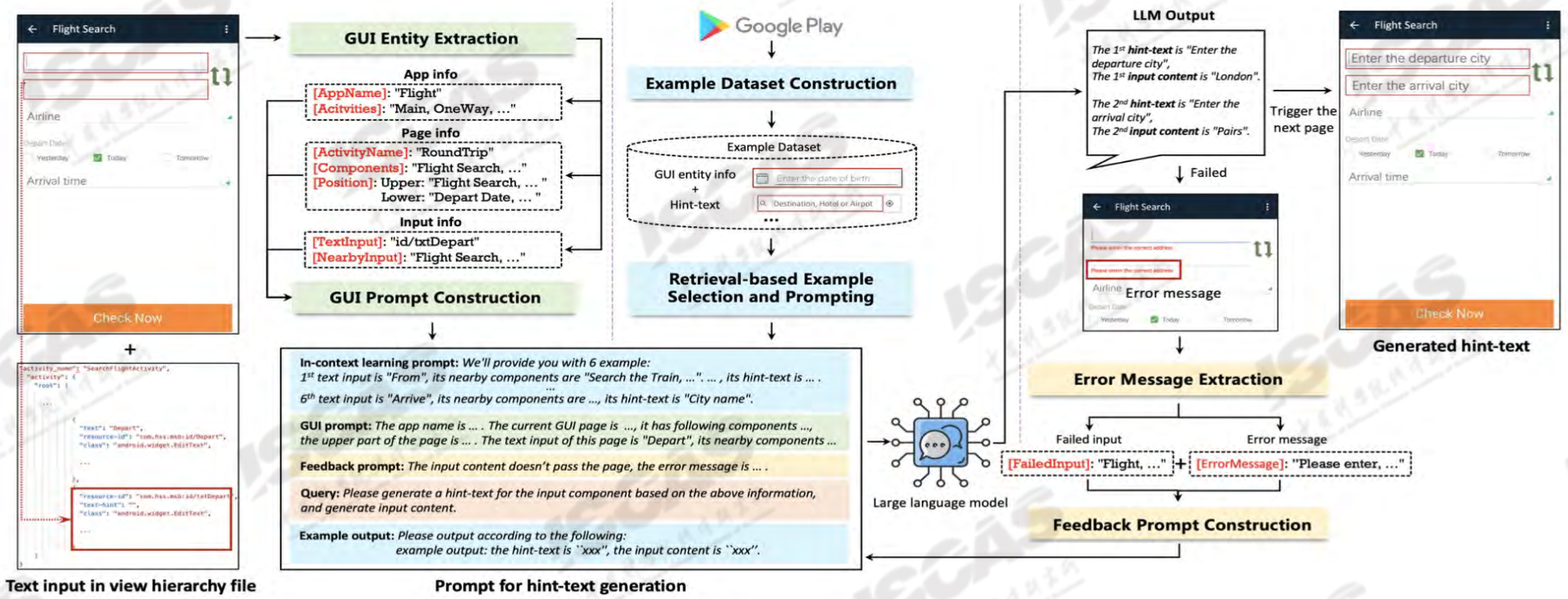
Query: *Please generate a hint-text for the input component based on the above information, and generate input content.*

Example output: *Please output according to the following:*
example output: the hint-text is ``xxx'', the input content is ``xxx''.

Prompt for hint-text generation

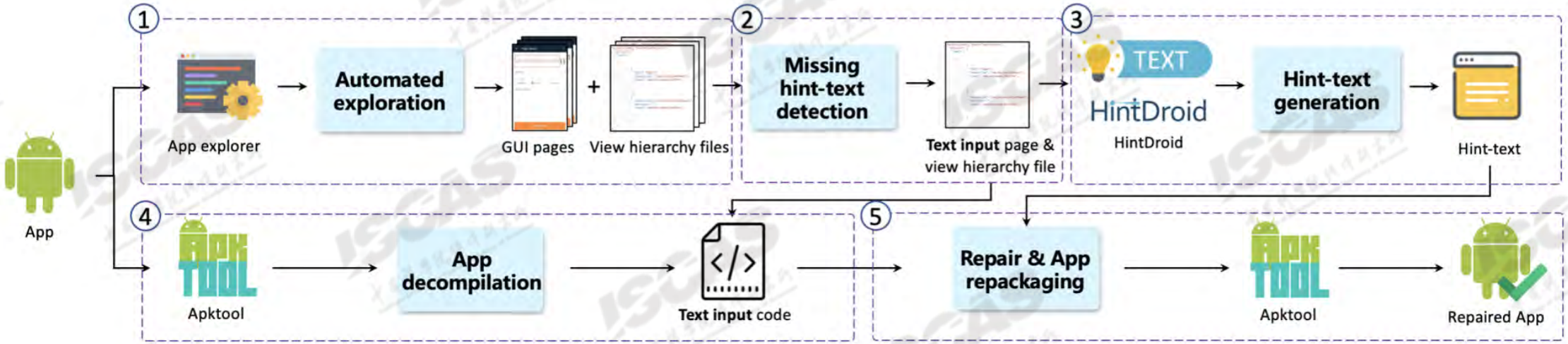
Module 3: Feedback Extraction and Prompting

- **Feedback Extraction and Prompting**
  - ✓ Automated Input Content Checking
  - ✓ Error Message Extraction



Implementation

- ✓ Extracting GUI pages
- ✓ Detecting GUI pages with missing hint-text
- ✓ Predicting hint-text based on GUI information
- ✓ Decompiling APK to obtain code
- ✓ Repackaging APK after code modification



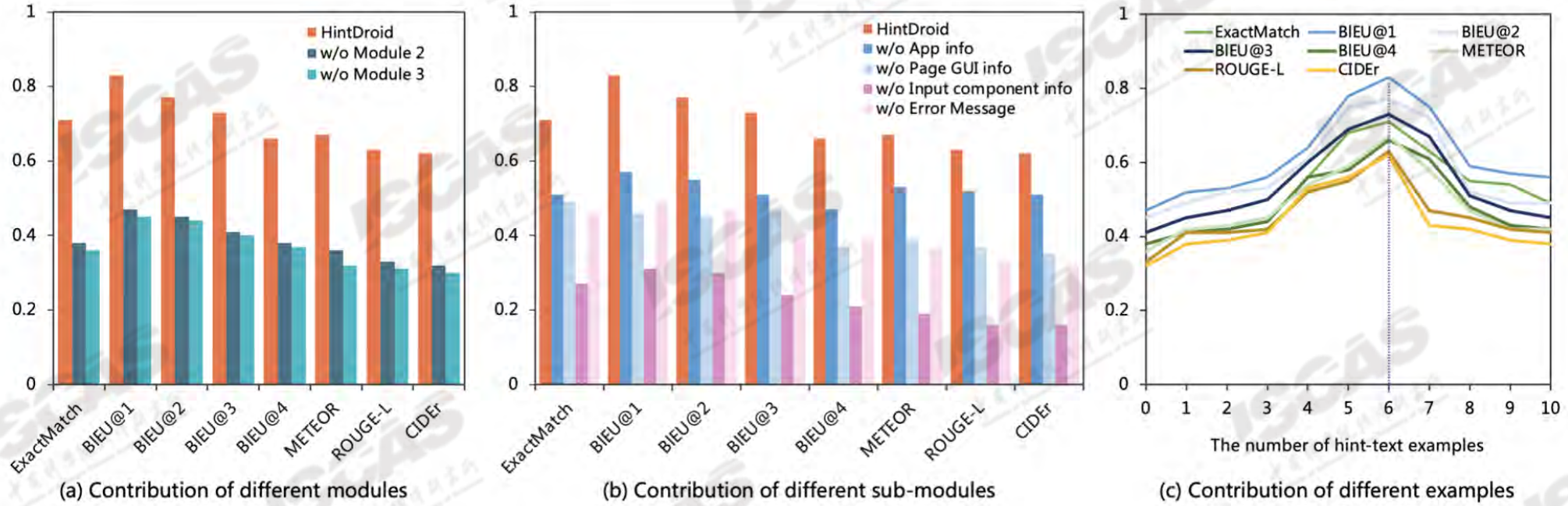
Evaluation - Effectiveness

We evaluate the effectiveness of HintDroid from the point of view of the hint-text generation accuracy.

Method	Exact match	BLEU@1	BLEU@2	BLEU@3	BLEU@4	METEOR	ROUGE-L	CIDEr
Learning-based method								
RNN	0.29	0.37	0.35	0.31	0.29	0.26	0.24	0.21
LSTM	0.28	0.33	0.31	0.25	0.22	0.19	0.17	0.13
Seq2Seq	0.30	0.37	0.32	0.29	0.27	0.25	0.21	0.18
Transformer	0.39	0.53	0.47	0.43	0.40	0.37	0.36	0.35
RNNInput	0.28	0.35	0.33	0.32	0.27	0.25	0.23	0.19
LableDroid	0.34	0.47	0.45	0.39	0.36	0.35	0.32	0.31
CNN+LSTM	0.26	0.29	0.24	0.19	0.17	0.15	0.09	0.08
Matching-based method								
Retrieval based	0.21	0.27	0.24	0.22	0.18	0.20	0.18	0.15
Random based	0.11	0.16	0.13	0.10	0.07	0.08	0.09	0.07
Rule based	0.32	0.45	0.39	0.32	0.27	0.29	0.31	0.27
LLM-based method								
GhatGPT	0.35	0.49	0.43	0.39	0.36	0.38	0.33	0.31
QTypist	0.31	0.47	0.41	0.35	0.33	0.37	0.34	0.33
HintDroid	**0.71**	**0.83**	**0.77**	**0.73**	**0.66**	**0.67**	**0.63**	**0.62**

Evaluation - Ablation Study

We can see that HintDroid's hint-text generation performance is much higher than all other variants.



Evaluation - Usefulness

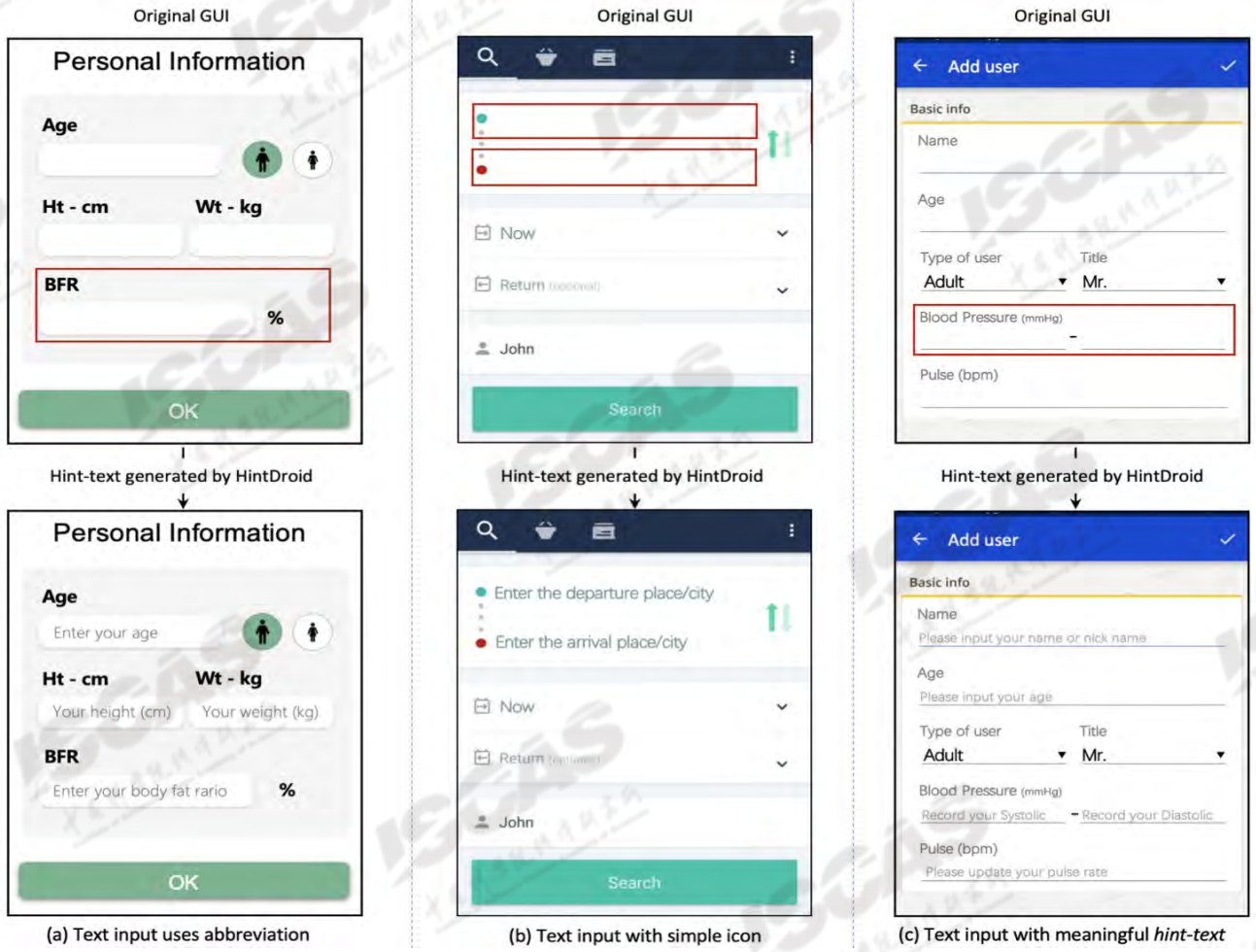
Whether HintDroid can help visually impaired users fill in the input.

- **Data:** 33 apps with 237 text input
- **Participants:**
  - P1 to P18 use HintDroid
  - P18 to P36 without
- **Metrics:**
  - Input accuracy
  - Activity coverage
  - State coverage
  - Filling time

Basic information			Input accuracy		Activity coverage		State coverage		Filling time (min)	
id	App	Category	control	experiment	control	experiment	control	experiment	control	experiment
1	HealthHB	Health	0.50	0.81	0.54	0.72	0.51	0.64	2.38	1.18
2	WeatherL&W	Weather	0.40	0.88	0.24	0.66	0.62	0.78	1.93	1.02
3	MessagerWA	Communi	0.31	0.95	0.29	0.73	0.24	0.68	2.30	1.10
4	MoneyTK	Finance	0.53	0.72	0.50	0.69	0.36	0.74	2.74	0.70
5	FoodFacts	Food	0.44	0.90	0.52	0.65	0.47	0.63	1.49	1.37
6	MPAS+	Maps	0.50	0.80	0.28	0.67	0.35	0.74	1.79	1.46
7	PSStore	Product	0.34	0.80	0.28	0.63	0.52	0.76	1.50	1.43
8	NewAudio	Music	0.42	0.84	0.20	0.64	0.51	0.82	2.89	0.64
9	WallETH	Personal	0.47	0.90	0.21	0.74	0.26	0.81	1.72	0.93
10	PicGall	Photo	0.45	0.71	0.27	0.59	0.37	0.81	2.92	0.53
11	SmartNew	News	0.36	0.75	0.52	0.68	0.25	0.64	1.60	1.49
12	MyHM	House	0.18	0.87	0.47	0.75	0.43	0.71	2.56	0.83
13	INSTEAD	Life	0.15	0.77	0.53	0.69	0.30	0.75	1.80	0.54
14	GameSpe	Game	0.55	0.73	0.25	0.75	0.50	0.76	1.71	0.51
15	BusinessEX	Business	0.12	0.72	0.21	0.65	0.35	0.79	1.50	0.55
16	PocketMaps	Travel	0.14	0.80	0.41	0.71	0.46	0.78	2.84	0.58
17	EventOR	Events	0.12	0.95	0.52	0.73	0.57	0.75	1.43	0.23
18	FitTAP	Comics	0.16	0.93	0.46	0.69	0.31	0.82	2.13	0.46
19	SkyTube	Video	0.17	0.96	0.45	0.72	0.36	0.82	2.19	0.95
20	LibReader	Books	0.26	0.71	0.33	0.69	0.25	0.69	2.50	0.32
21	NoxSecu	Tool	0.53	0.96	0.47	0.68	0.63	0.64	2.03	1.29
22	EarnMon	Social	0.13	0.91	0.39	0.73	0.63	0.66	2.29	0.32
23	WalkTra	Sports	0.35	0.70	0.42	0.72	0.56	0.65	2.71	0.68
24	ParentLA	Parenting	0.36	0.72	0.24	0.69	0.59	0.80	1.70	1.55
25	ISAY	Medical	0.53	0.96	0.48	0.60	0.29	0.72	2.18	0.43
26	Ipsos	Commun	0.09	0.72	0.54	0.79	0.55	0.79	2.71	1.25
27	FIRR	Libraries	0.53	0.79	0.52	0.65	0.34	0.67	1.33	1.23
28	DRBUs	Shopping	0.28	0.72	0.53	0.74	0.63	0.71	2.19	1.44
29	Learning	Education	0.49	0.89	0.33	0.69	0.40	0.63	2.17	0.95
30	MMDR	Dating	0.06	0.74	0.21	0.75	0.31	0.77	1.37	0.73
31	Pretty	Beauty	0.19	0.94	0.58	0.70	0.60	0.73	1.99	0.36
32	Fair	Auto	0.33	0.82	0.46	0.74	0.51	0.65	1.77	1.25
33	ArtPIX	Art	0.49	0.85	0.24	0.57	0.41	0.67	2.88	0.86
Average			0.33	**0.83**	0.39	**0.69**	0.44	**0.73**	2.10	**0.88**

Evaluation - Usefulness

Example of different good cases generated by HintDroid



总结和展望

- 用户界面测试现状和挑战
- 测试输入生成技术
- 面向测试路径规划的自动化GUI测试技术
- 基于多模态大模型的自动化GUI测试技术
- 针对文本输入的模糊测试技术
- 面向文本输入组件的交互提升技术

挑战

- 测试覆盖率达不到100%
- 页面元素远超开源应用，单个页面元素常常超过100个
- 图标类组件，缺失label等元素
- 文本描述中的模糊描述，左上，右上
- 一些特殊表述，例如：选择地区
- 大模型输出里面标号和描述对不上

AiDD

THANKS