AiDD AI+研发数字峰会
AI+ Development Digital summit
第5届

当软件测试遇上AI算法

刘烨庞 | 南方科技大学

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 敦煌站
K+ 思考周®研习社
时间：2025.08.29-30

K+峰会 上海站
K+ 金融专场
时间：2025.10.17-18

K+峰会 香港站
K+ 思考周®研习社
时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 上海站
AI+研发数字峰会
时间：2025.05.17-18

AiDD峰会 北京站
AI+研发数字峰会
时间：2025.08.08-09

AiDD峰会 深圳站
AI+研发数字峰会
时间：2025.11.28-29

AiDD峰会详情

演讲嘉宾

AiDD AI+研发数字峰会 AI+ Development Digital summit

刘烨庞

南方科技大学计算机科学与工程系长聘副教授 (博导、研究员)

南科大软件质量实验室负责人，斯发基斯可信自主系统研究院可信软件中心负责人。2010年本科毕业于南京大学（校优秀毕业生），2015年博士毕业于香港科技大学，师从软件测试与分析专家IEEE Fellow张成志讲席教授。研究方向包括软件测试与分析，智能化软件工程，可信人工智能等，近五年主持国家、省部级以及企业项目14项。在国际期刊与会议上发表论文95篇，获ACM SIGSOFT杰出论文奖三次、杰出软件制品奖一次，被国际同行评为“全球前10最活跃软件工程青年学者”。开发的工具在开源项目（如TensorFlow，PyTorch）中检测到2000余个未知缺陷，多项成果在国内龙头企业部署应用。常年参与国际会议及期刊组织与审稿工作，多次担任领域顶会程序委员会委员，获ACM SIGSOFT杰出审稿人奖和服务奖。更多信息请见个人主页https://yepangliu.github.io/。

目录
CONTENTS

PART 01

背景

▶ 软件是人工智能时代的基础设施

人们日常使用各种App来完成通讯、社交、工作、学习、支付、理财、出行等任务

软件无处不在

深圳2022年上线汽车尾气监管平台，实时监控黑烟车，助推智慧城市建设

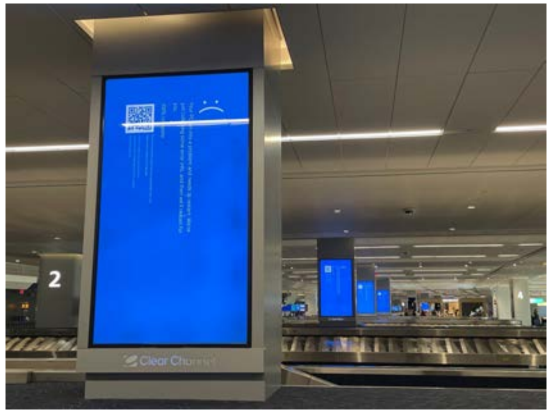
智能汽车中的动力系统、娱乐系统、驾驶辅助系统等都由软件控制

软件定义一切

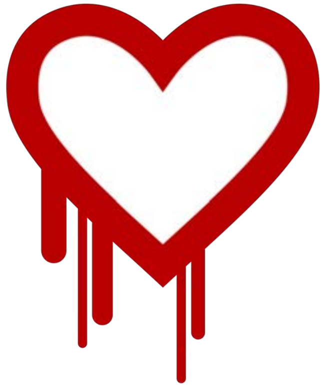
天智一号卫星采用开放系统架构，支持应用软件按需加载，系统功能按需重构

▶ 软件质量正变得前所未有的重要

- 软件事故可能导致严重后果，包括经济损失、数据泄露、人身伤害甚至死亡。



2024年7月，CrowdStrike安全**软件更新中的漏洞**导致全球近850万台Windows设备蓝屏，冲击航空、酒店、医院等众多行业，造成**数十亿美元损失**。



2014年，Heartbleed漏洞被曝光，一个存在于OpenSSL软件中的**内存管理错误**，导致攻击者**窃取云服务器中保存的海量敏感信息**。

2024年8月，一辆特斯拉Model S在西雅图附近**撞死摩托骑手**，汽车在事故发生时正处于“完全自动驾驶” 模式，**FSD软件未能正确识别道路状况并及时做出反应**。

测试是保障软件质量的重要手段

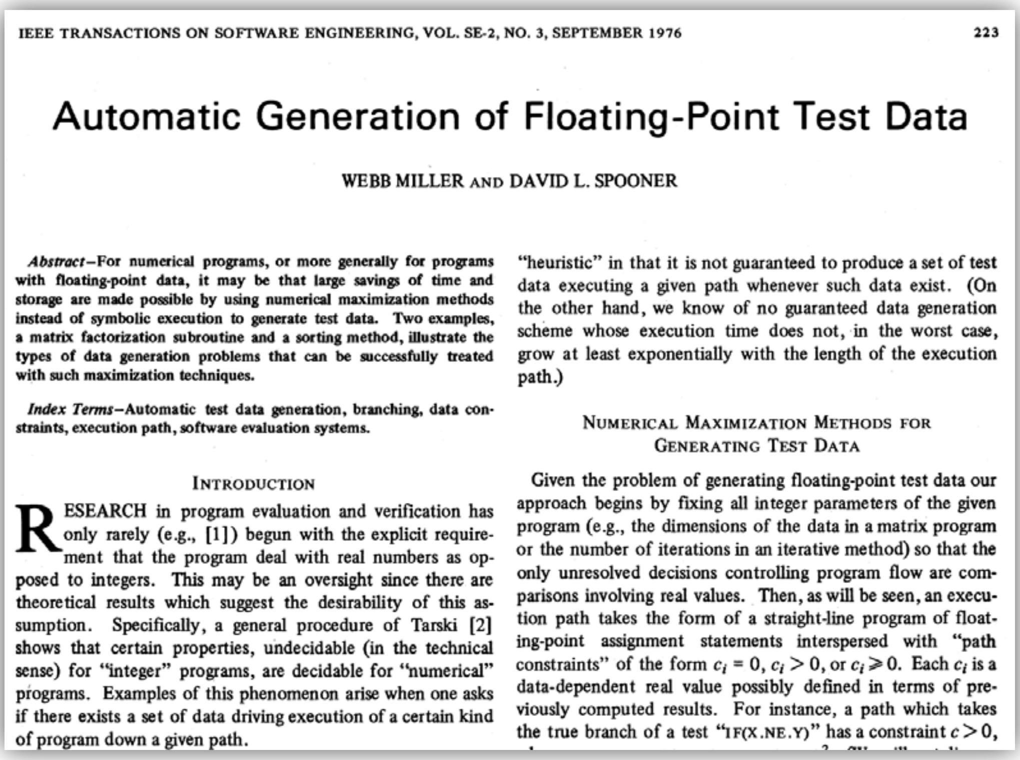
- **软件测试可以被看作是一个搜索和优化问题**
  - **搜索**：如何从极大的程序状态空间中找到错误的状态?
  - **优化**：如何在QA资源有限的情况下尽可能多地发现bugs?

- **人工测试效率低下，近五十年来研究人员设计了大量自动化测试生成技术**
  - **传统算法**：随机算法 (unguided & guided)、符号执行 (包括DSE)、基于模型的测试等
  - **AI算法**：启发式算法、多目标优化算法、深度学习算法，强化学习算法等

PART 02
智能化软件测试技术介绍

AI算法在软件测试中一直扮演重要角色

早在上世纪70年代就有学者提出基于启发式算法的测试技术 (Miller and Spooner, TSE 1976)，开创了后来被叫做Search-Based Software Testing的研究领域



IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-2, NO. 3, SEPTEMBER 1976 223

Automatic Generation of Floating-Point Test Data

WEBB MILLER AND DAVID L. SPOONER

Abstract–For numerical programs, or more generally for programs with floating-point data, it may be that large savings of time and storage are made possible by using numerical maximization methods instead of symbolic execution to generate test data. Two examples, a matrix factorization subroutine and a sorting method, illustrate the types of data generation problems that can be successfully treated with such maximization techniques.

Index Terms–Automatic test data generation, branching, data constraints, execution path, software evaluation systems.

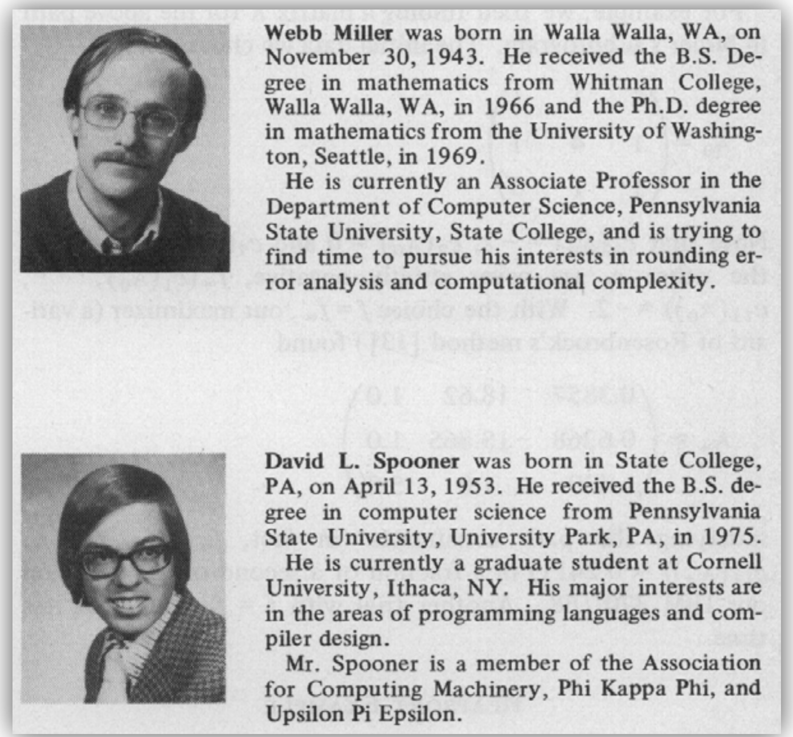
INTRODUCTION

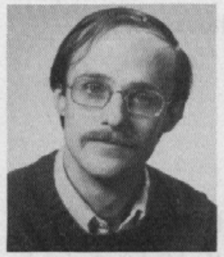
RESEARCH in program evaluation and verification has only rarely (e.g., [1]) begun with the explicit requirement that the program deal with real numbers as opposed to integers. This may be an oversight since there are theoretical results which suggest the desirability of this assumption. Specifically, a general procedure of Tarski [2] shows that certain properties, undecidable (in the technical sense) for "integer" programs, are decidable for "numerical" programs. Examples of this phenomenon arise when one asks if there exists a set of data driving execution of a certain kind of program down a given path.

"heuristic" in that it is not guaranteed to produce a set of test data executing a given path whenever such data exist. (On the other hand, we know of no guaranteed data generation scheme whose execution time does not, in the worst case, grow at least exponentially with the length of the execution path.)

NUMERICAL MAXIMIZATION METHODS FOR GENERATING TEST DATA

Given the problem of generating floating-point test data our approach begins by fixing all integer parameters of the given program (e.g., the dimensions of the data in a matrix program or the number of iterations in an iterative method) so that the only unresolved decisions controlling program flow are comparisons involving real values. Then, as will be seen, an execution path takes the form of a straight-line program of floating-point assignment statements interspersed with "path constraints" of the form $c_i = 0$, $c_i > 0$, or $c_i \geq 0$. Each c_i is a data-dependent real value possibly defined in terms of previously computed results. For instance, a path which takes the true branch of a test "IF(X.NE.Y)" has a constraint $c > 0$,





Webb Miller was born in Walla Walla, WA, on November 30, 1943. He received the B.S. Degree in mathematics from Whitman College, Walla Walla, WA, in 1966 and the Ph.D. degree in mathematics from the University of Washington, Seattle, in 1969.

He is currently an Associate Professor in the Department of Computer Science, Pennsylvania State University, State College, and is trying to find time to pursue his interests in rounding error analysis and computational complexity.

David L. Spooner was born in State College, PA, on April 13, 1953. He received the B.S. degree in computer science from Pennsylvania State University, University Park, PA, in 1975.

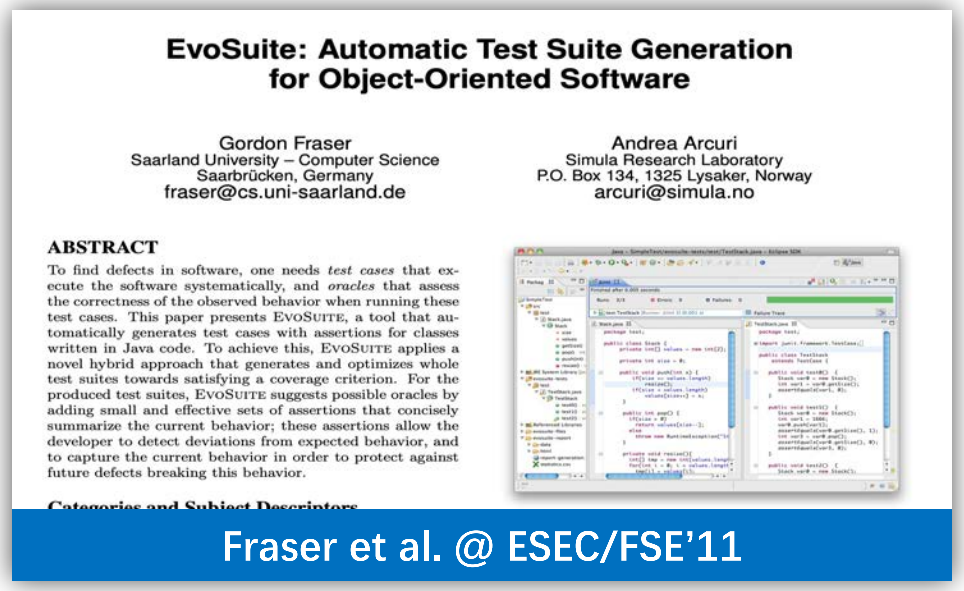
He is currently a graduate student at Cornell University, Ithaca, NY. His major interests are in the areas of programming languages and compiler design.

Mr. Spooner is a member of the Association for Computing Machinery, Phi Kappa Phi, and Upsilon Pi Epsilon.

▶ 基于启发式算法的软件自动化测试

EVOSUITE

AFL (American Fuzzy Lop)



EvoSuite: Automatic Test Suite Generation for Object-Oriented Software

Gordon Fraser
Saarland University – Computer Science
Saarbrücken, Germany
fraser@cs.uni-saarland.de

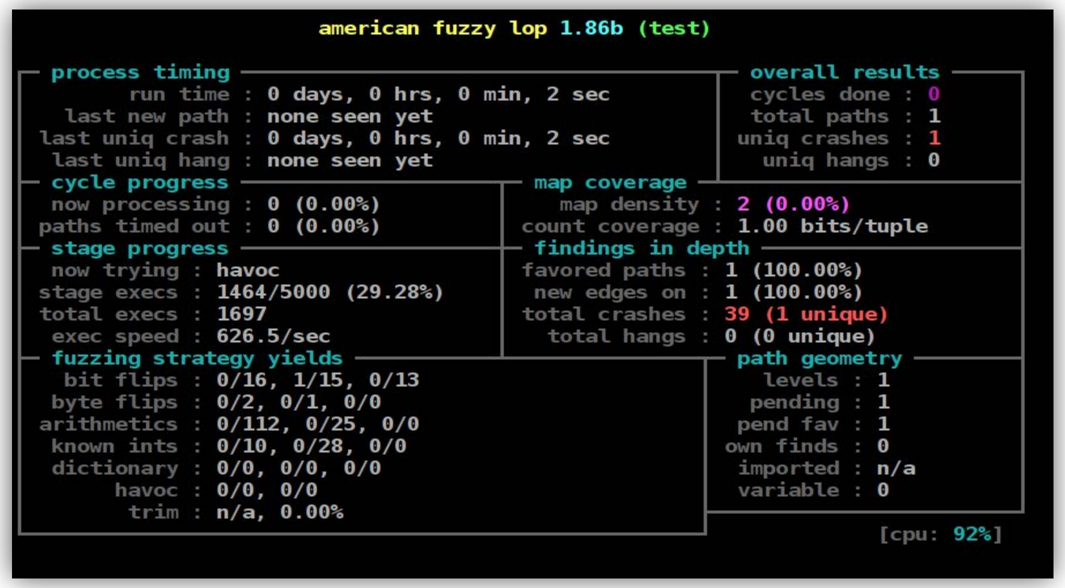
Andrea Arcuri
Simula Research Laboratory
P.O. Box 134, 1325 Lysaker, Norway
arcuri@simula.no

ABSTRACT

To find defects in software, one needs *test cases* that execute the software systematically, and *oracles* that assess the correctness of the observed behavior when running these test cases. This paper presents EvoSuite, a tool that automatically generates test cases with assertions for classes written in Java code. To achieve this, EvoSuite applies a novel hybrid approach that generates and optimizes whole test suites towards satisfying a coverage criterion. For the produced test suites, EvoSuite suggests possible oracles by adding small and effective sets of assertions that concisely summarize the current behavior; these assertions allow the developer to detect deviations from expected behavior, and to capture the current behavior in order to protect against future defects breaking this behavior.

Categories and Subject Descriptors

Fraser et al. @ ESEC/FSE’11



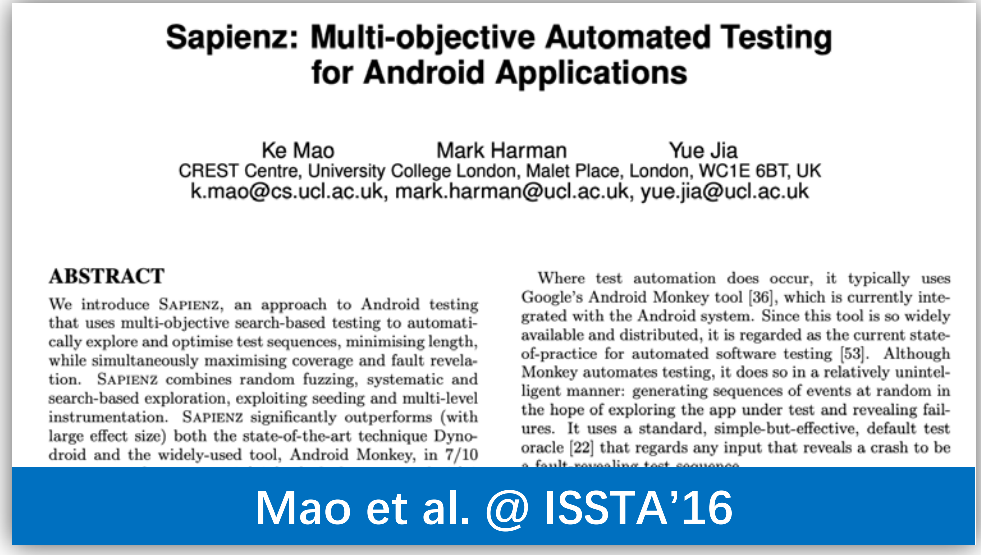
```
                        american fuzzy lop 1.86b (test)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 0 min, 2 sec        │  cycles done : 0      │
│   last new path : none seen yet                       │  total paths : 1      │
│ last uniq crash : 0 days, 0 hrs, 0 min, 2 sec        │ uniq crashes : 1      │
│  last uniq hang : none seen yet                       │   uniq hangs : 0      │
├─ cycle progress ──────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 0 (0.00%)        │    map density : 2 (0.00%)             │
│ paths timed out : 0 (0.00%)        │ count coverage : 1.00 bits/tuple       │
├─ stage progress ──────────────────┼─ findings in depth ─────────────────────┤
│  now trying : havoc                │ favored paths : 1 (100.00%)            │
│ stage execs : 1464/5000 (29.28%)   │  new edges on : 1 (100.00%)            │
│ total execs : 1697                 │ total crashes : 39 (1 unique)          │
│  exec speed : 626.5/sec            │   total hangs : 0 (0 unique)           │
├─ fuzzing strategy yields ─────────┴─────────┬─ path geometry ──────────────┤
│   bit flips : 0/16, 1/15, 0/13               │    levels : 1                 │
│  byte flips : 0/2, 0/1, 0/0                  │   pending : 1                 │
│ arithmetics : 0/112, 0/25, 0/0               │  pend fav : 1                 │
│  known ints : 0/10, 0/28, 0/0                │ own finds : 0                 │
│  dictionary : 0/0, 0/0, 0/0                  │  imported : n/a               │
│       havoc : 0/0, 0/0                       │  variable : 0                 │
│        trim : n/a, 0.00%                     ├──────────────────────────────┘
└──────────────────────────────────────────────┘              [cpu: 92%]
```

基于**遗传算法**生成高覆盖率JUnit测试用例集
(论文**1300+引用**)

Google前员工Michał Zalewski开发的具有广泛影响力的开源模糊测试工具（使用**遗传算法**提升CFG的边覆盖率）

▶ 基于多目标优化算法的软件自动化测试



Sapienz: Multi-objective Automated Testing for Android Applications

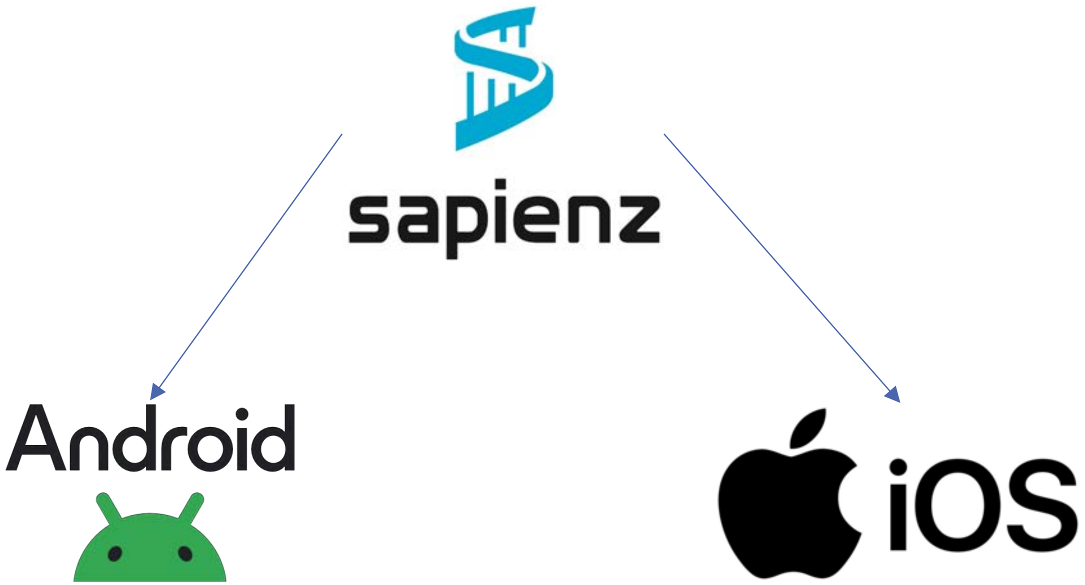
Ke Mao Mark Harman Yue Jia
CREST Centre, University College London, Malet Place, London, WC1E 6BT, UK
k.mao@cs.ucl.ac.uk, mark.harman@ucl.ac.uk, yue.jia@ucl.ac.uk

ABSTRACT

We introduce SAPIENZ, an approach to Android testing that uses multi-objective search-based testing to automatically explore and optimise test sequences, minimising length, while simultaneously maximising coverage and fault revelation. SAPIENZ combines random fuzzing, systematic and search-based exploration, exploiting seeding and multi-level instrumentation. SAPIENZ significantly outperforms (with large effect size) both the state-of-the-art technique Dynodroid and the widely-used tool, Android Monkey, in 7/10

Where test automation does occur, it typically uses Google's Android Monkey tool [36], which is currently integrated with the Android system. Since this tool is so widely available and distributed, it is regarded as the current state-of-practice for automated software testing [53]. Although Monkey automates testing, it does so in a relatively unintelligent manner: generating sequences of events at random in the hope of exploring the app under test and revealing failures. It uses a standard, simple-but-effective, default test oracle [22] that regards any input that reveals a crash to be

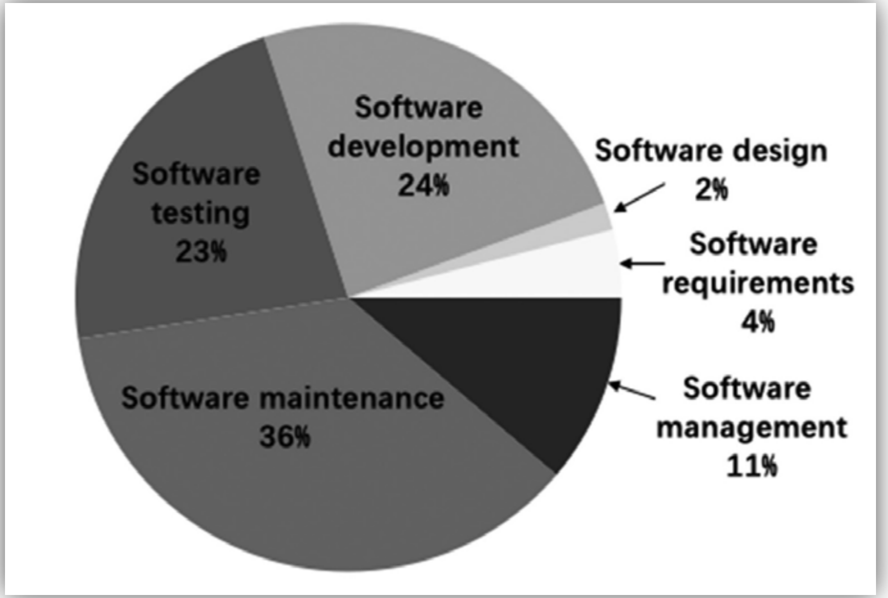
Mao et al. @ ISSTA’16

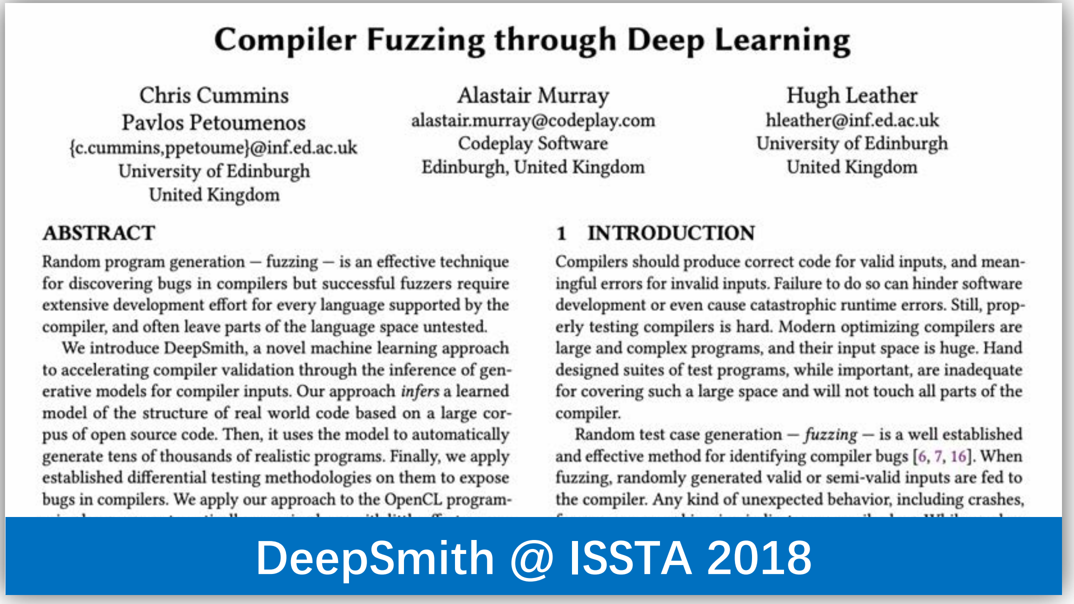


移动应用自动测试工具，**优化目标：**代码覆盖率、动作序列长度、触发的应用crash数量

(论文**650+引用**，技术在**Facebook**公司成功转化)

▶ 基于深度学习算法的软件自动化测试





Compiler Fuzzing through Deep Learning

Chris Cummins
Pavlos Petoumenos
{c.cummins,ppetoume}@inf.ed.ac.uk
University of Edinburgh
United Kingdom

Alastair Murray
alastair.murray@codeplay.com
Codeplay Software
Edinburgh, United Kingdom

Hugh Leather
hleather@inf.ed.ac.uk
University of Edinburgh
United Kingdom

ABSTRACT

Random program generation — fuzzing — is an effective technique for discovering bugs in compilers but successful fuzzers require extensive development effort for every language supported by the compiler, and often leave parts of the language space untested.

We introduce DeepSmith, a novel machine learning approach to accelerating compiler validation through the inference of generative models for compiler inputs. Our approach *infers* a learned model of the structure of real world code based on a large corpus of open source code. Then, it uses the model to automatically generate tens of thousands of realistic programs. Finally, we apply established differential testing methodologies on them to expose bugs in compilers. We apply our approach to the OpenCL program-

1 INTRODUCTION

Compilers should produce correct code for valid inputs, and meaningful errors for invalid inputs. Failure to do so can hinder software development or even cause catastrophic runtime errors. Still, properly testing compilers is hard. Modern optimizing compilers are large and complex programs, and their input space is huge. Hand designed suites of test programs, while important, are inadequate for covering such a large space and will not touch all parts of the compiler.

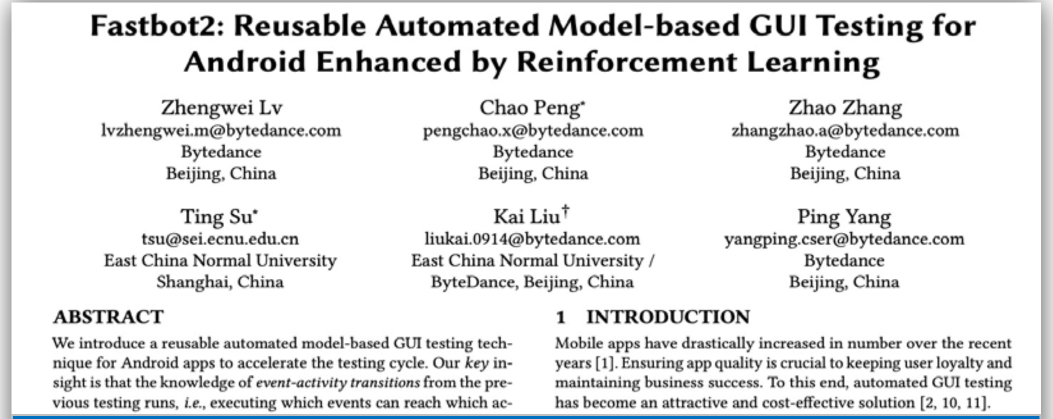
Random test case generation — *fuzzing* — is a well established and effective method for identifying compiler bugs [6, 7, 16]. When fuzzing, randomly generated valid or semi-valid inputs are fed to the compiler. Any kind of unexpected behavior, including crashes,

DeepSmith @ ISSTA 2018

DL算法广泛用于测试，**相关论文占所有DL4SE论文的23%** (Yang et al, CSUR'22)

爱丁堡大学研究人员**训练深度学习模型 (language model) 自动生成程序**， 用于编译器测试

基于强化学习算法的软件自动化测试



Fastbot2: Reusable Automated Model-based GUI Testing for Android Enhanced by Reinforcement Learning

Zhengwei Lv
lvzhengwei.m@bytedance.com
Bytedance
Beijing, China

Chao Peng*
pengchao.x@bytedance.com
Bytedance
Beijing, China

Zhao Zhang
zhangzhao.a@bytedance.com
Bytedance
Beijing, China

Ting Su*
tsu@sei.ecnu.edu.cn
East China Normal University
Shanghai, China

Kai Liu†
liukai.0914@bytedance.com
East China Normal University /
ByteDance, Beijing, China

Ping Yang
yangping.cser@bytedance.com
Bytedance
Beijing, China

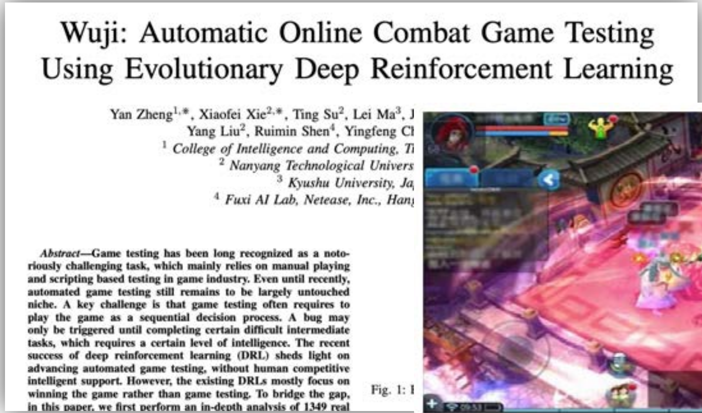
ABSTRACT

We introduce a reusable automated model-based GUI testing technique for Android apps to accelerate the testing cycle. Our *key* insight is that the knowledge of *event-activity transitions* from the previous testing runs, *i.e.*, executing which events can reach which ac-

1 INTRODUCTION

Mobile apps have drastically increased in number over the recent years [1]. Ensuring app quality is crucial to keeping user loyalty and maintaining business success. To this end, automated GUI testing has become an attractive and cost-effective solution [2, 10, 11].

Fastbot2 @ ASE’22 (Industry Track)



Wuji: Automatic Online Combat Game Testing Using Evolutionary Deep Reinforcement Learning

Yan Zheng[1,*], Xiaofei Xie[2,*], Ting Su[2], Lei Ma[3], J
Yang Liu[2], Ruimin Shen[4], Yingfeng Cl
[1] *College of Intelligence and Computing, Ti*
[2] *Nanyang Technological Univers*
[3] *Kyushu University, Ja*
[4] *Fuxi AI Lab, Netease, Inc., Han*

Abstract—Game testing has been long recognized as a notoriously challenging task, which mainly relies on manual playing and scripting based testing in game industry. Even until recently, automated game testing still remains to be largely untouched niche. A key challenge is that game testing often requires to play the game as a sequential decision process. A bug may only be triggered until completing certain difficult intermediate tasks, which requires a certain level of intelligence. The recent success of deep reinforcement learning (DRL) sheds light on advancing automated game testing, without human competitive intelligent support. However, the existing DRLs mostly focus on winning the game rather than game testing. To bridge the gap, in this paper, we first perform an in-depth analysis of 1349 real

Fig. 1: I

Wuji @ ASE’19

字节跳动质量实验室推出的app稳定性测试工具

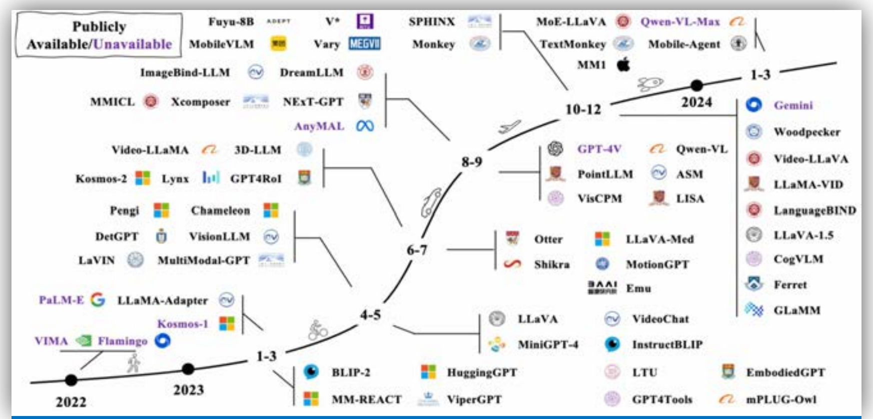
(利用**Q-learning算法**提升GUI测试效果)

天津大学Yan Zheng等学者与**网易伏羲实验室**合作

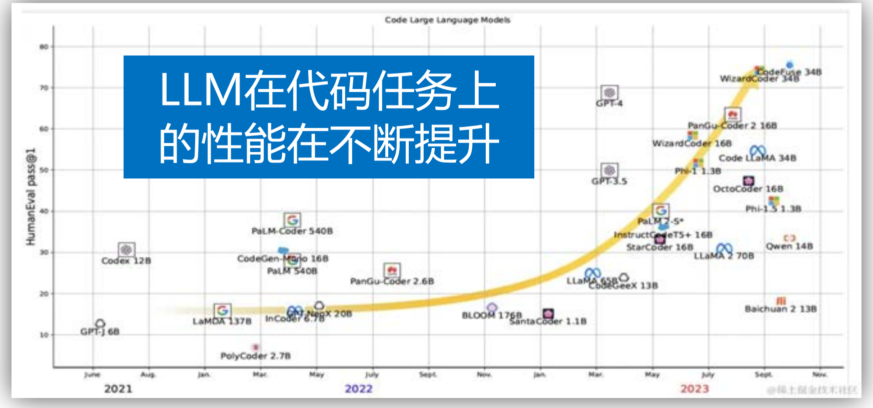
将**A2C算法**应用于在线格斗游戏测试

▶ 大模型出现，软件工程3.0时代到来！

以下图片均来自互联网：



据信通院发布的2024年《全球数字经济白皮书》，全球大模型共1328个（美国44%，中国36%）



LLM在代码任务上的性能在不断提升

大模型具备的下述强大能力决定了它在软件测试任务上有广阔的应用前景：

- 自然语言处理 (理解、生成等)
- 代码理解和生成
- 复杂模式识别与预测
- 多模态支持 (能理解软件界面) …

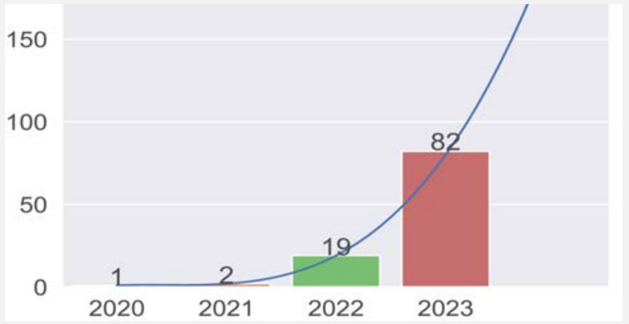
这些问题自然而然引发了很多研究和工程人员的兴趣：

- 如何正确借助大模型来提升软件测试效率?
- 如何有效利用大模型来帮助提升测试覆盖率?
- 如何在大模型的辅助下，提升缺陷检测的成功率?

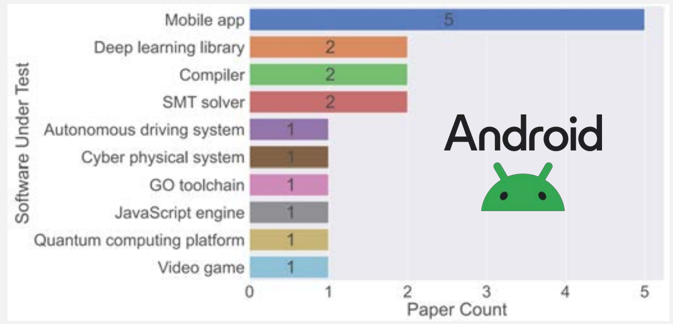
LLM4Testing 研究现状

数据和图片来自**中科院软件所王青教授团队**最近在TSE期刊上发表的综述《Software Testing With Large Language Models: Survey, Landscape, and Vision》（**基于2023年6月前发表的102篇文献**）

近年来论文数量

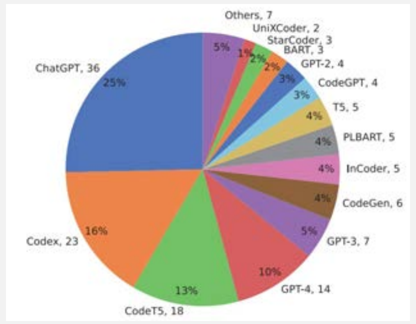


测试对象



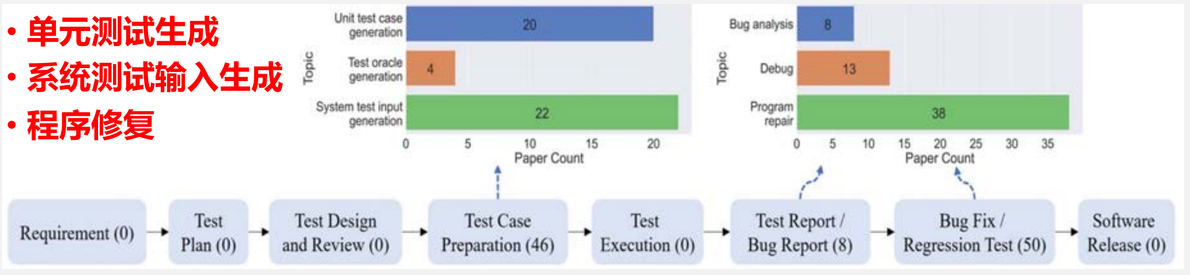
论文中使用到的大模型

- **ChatGPT (25%)**
- **Codex (16%)**
- **CodeT5 (13%)**
- **GPT-4 (10%)**

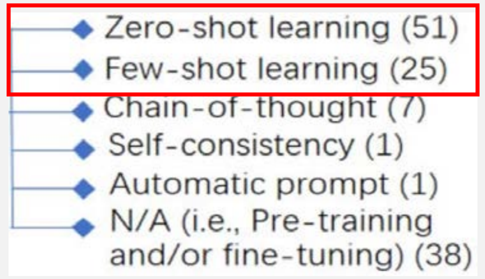


涉及的软件测试任务 (LLM的使用场景)

- **单元测试生成**
- **系统测试输入生成**
- **程序修复**

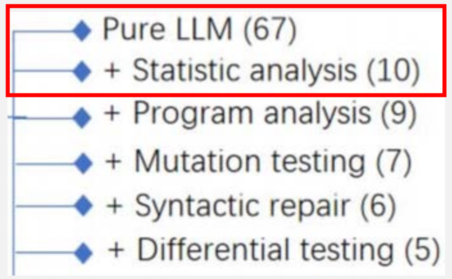


提示词工程方法



- Zero-shot learning (51)
- Few-shot learning (25)
- Chain-of-thought (7)
- Self-consistency (1)
- Automatic prompt (1)
- N/A (i.e., Pre-training and/or fine-tuning) (38)

是否与其他方法结合?



- Pure LLM (67)
- + Statistic analysis (10)
- + Program analysis (9)
- + Mutation testing (7)
- + Syntactic repair (6)
- + Differential testing (5)

PART 03
AI4Testing方向的探索与实践

▶ 我们团队近期的一点探索与实践

ISSTA 24　　如何利用LLM辅助检测**特定领域**的**“小众”bug**?

ICSME 2024　　如何利用**视觉语言模型**对**网页界面**进行高效测试?

ASE 2024 SACRAMENTO　　如何利用**多智能体强化学习**提升**网页界面**测试性能?

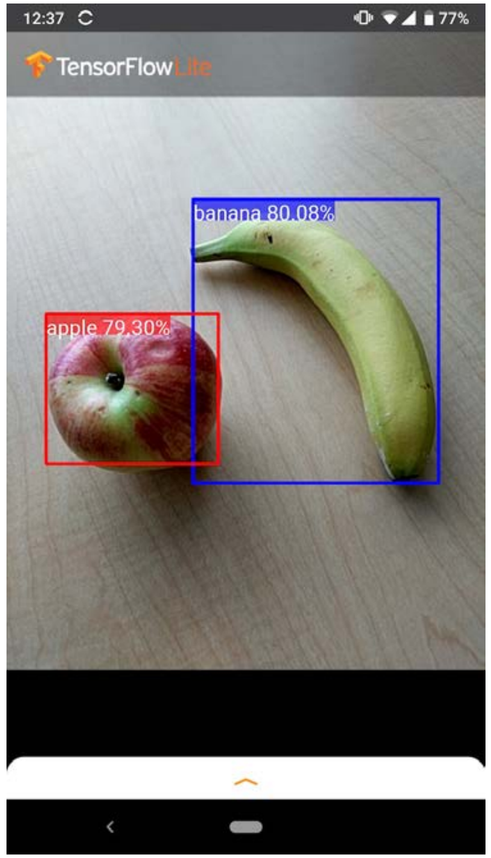
▶ 背景：深度学习模型优化

- Real-world DL models are usually complex and large.

Model	Size (MB)	Time per inference step
ResNet152	232	127.4ms

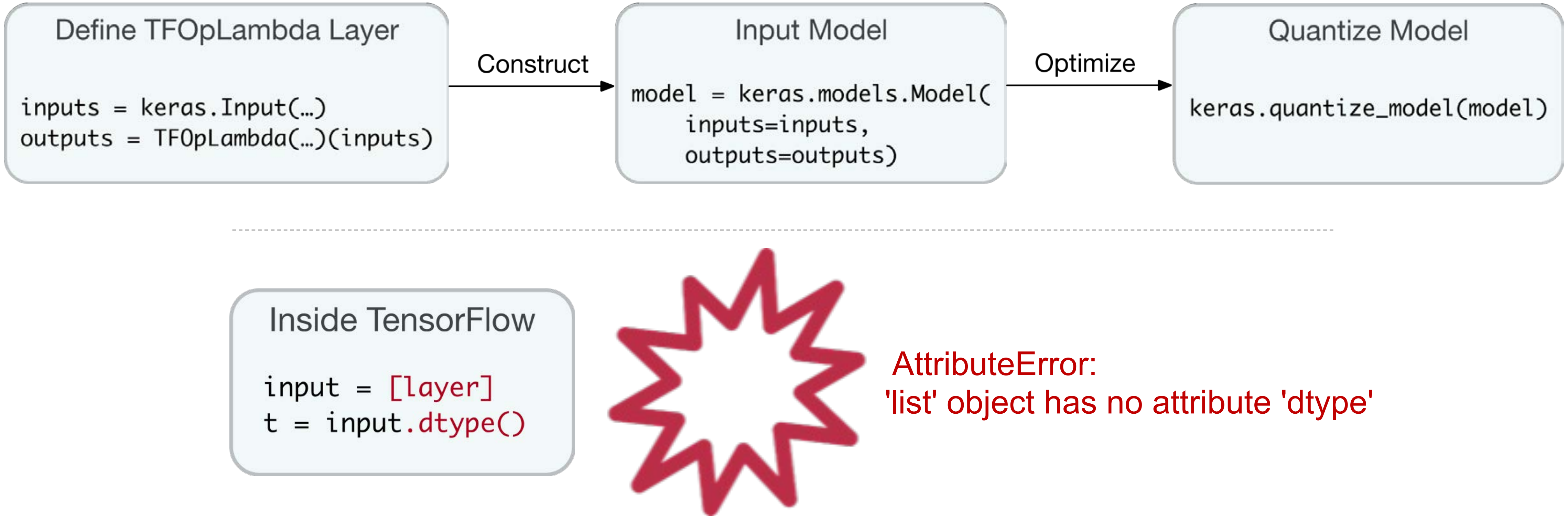
- LLMs are even larger, for example, LLaMA 3.1 by Meta has 8 billion to 405 billion parameters (**千亿参数**)

- It requires model optimization to deploy DL components on resource-constrained devices
  - **Pruning (剪枝):** zero-out the insignificant weights
  - **Quantization (量化):** reduce the size of weights (e.g., 32-bit float to 8-bit int)



▶ 背景：深度学习框架中的模型优化缺陷

- 由于神经网络的复杂性 (网络类型、张量形状、模型结构)，**现有框架无法完全保证模型优化的稳定性**，经常会出现意外的错误，比如下面TensorFlow框架中的一个真实例子：



```
inputs = keras.Input(…)
outputs = TFOpLambda(…)(inputs)
```

```
model = keras.models.Model(
    inputs=inputs,
    outputs=outputs)
```

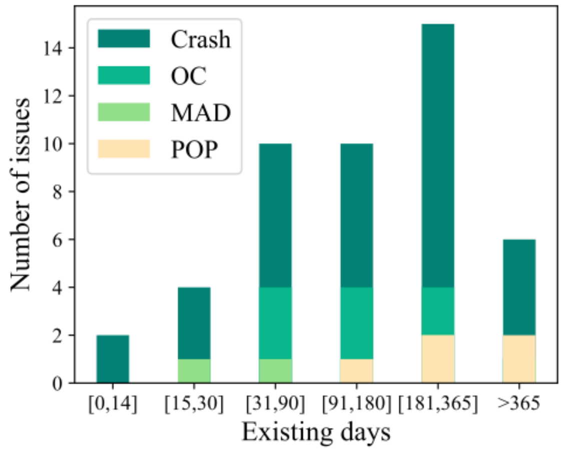
```
keras.quantize_model(model)
```

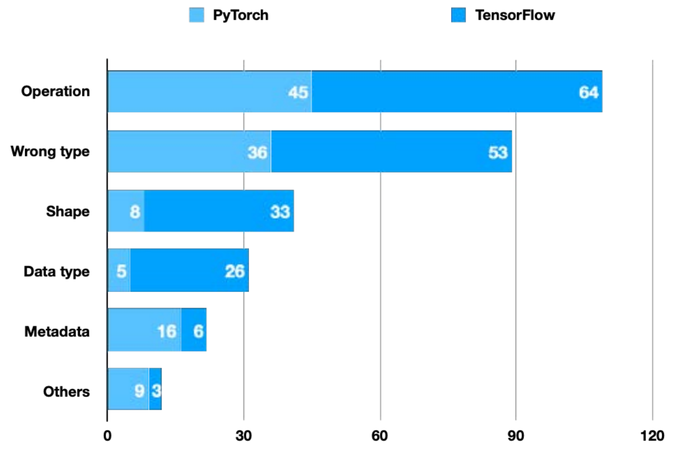
```
input = [layer]
t = input.dtype()
```

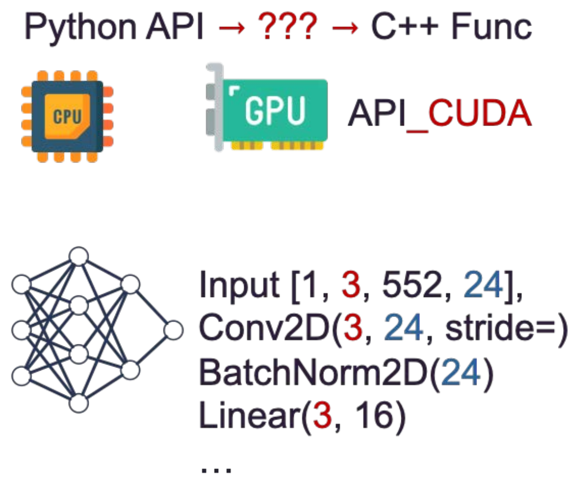
▶ 背景：深度学习框架中的模型优化缺陷

我们在TensorFlow和PyTorch两个著名框架的模型优化模块发现370个真实缺陷 (Model Optimization Bugs, MOBs)，并开展了首个实证研究 (ICSE 2023)：

- 检测和诊断MOBs比较困难，开发人员要投入大量时间
- 触发缺陷的代码片段经常涉及改变模型结构的复杂操作







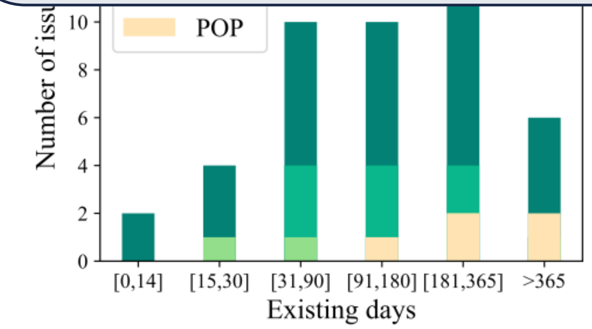
▶ 背景：深度学习框架中的模型优化缺陷

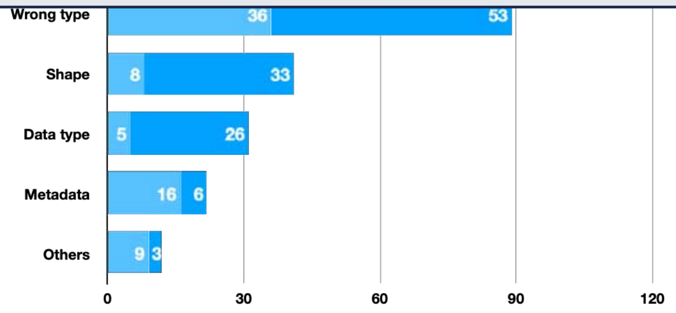
我们在TensorFlow和PyTorch两个著名框架的模型优化模块发现370个真实缺陷 (Model Optimization Bugs, MOBs)，并开展了首个实证研究 (ICSE 2023)：

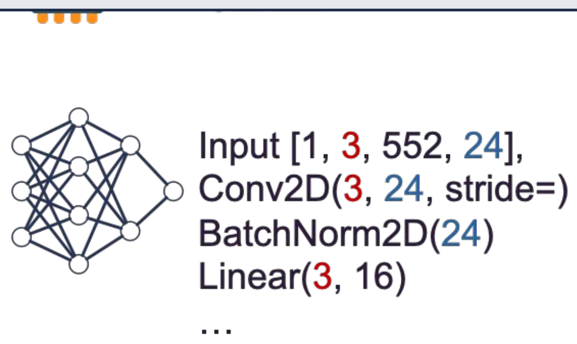
- 检测和诊断MOBs比较困难，开发人员要投入大量时间

自动检测MOBs需要生成模型优化程序，包括：1) 输入模型构造代码，2) 优化API调用代码

- **挑战1：**生成的模型构造代码和API调用通常需要满足复杂的constraints
- **挑战2：**输入模型和优化API的组合成千上万，能触发问题的组合却比较罕见



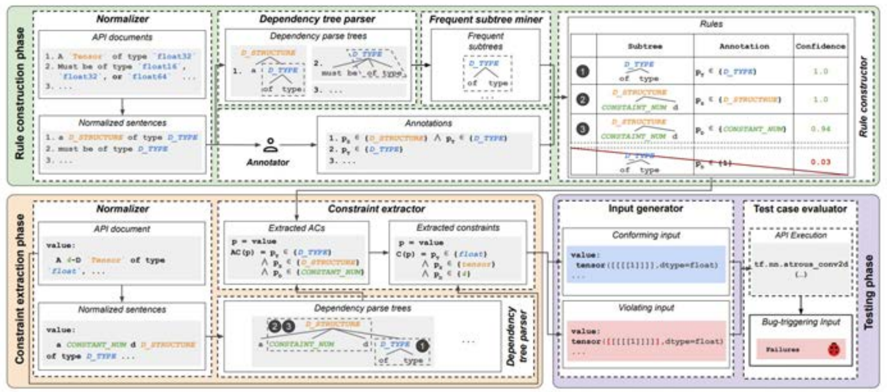




```
Input [1, 3, 552, 24],
Conv2D(3, 24, stride=)
BatchNorm2D(24)
Linear(3, 16)
...
```

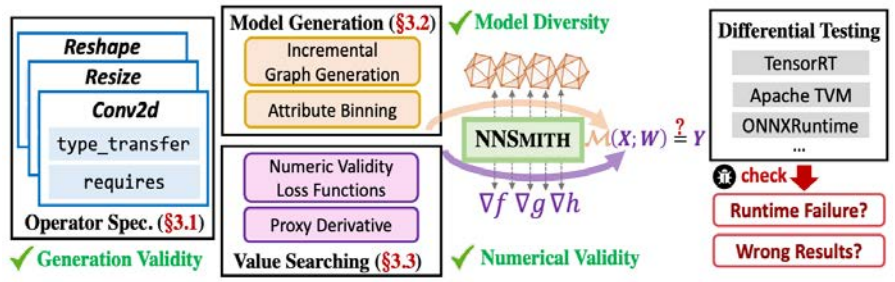
▶ 现有的深度学习框架测试工作

传统的面向深度学习框架/编译器的测试方法无法有效用于检测MOBs



DocTer: Documentation-Guided Fuzzing for Testing Deep Learning API Functions (Xie et al., ISSTA'22)

DocTer**<u>依赖于高质量的文档</u>**，并且仅仅面向**<u>单个API</u>**进行测试，而深度学习优化程序通常涉及多个框架API的调用。

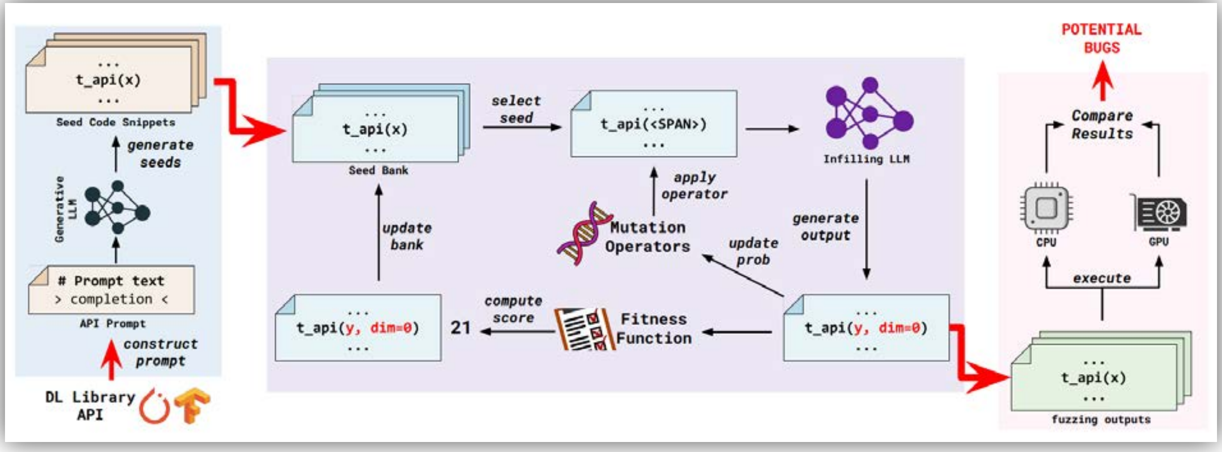


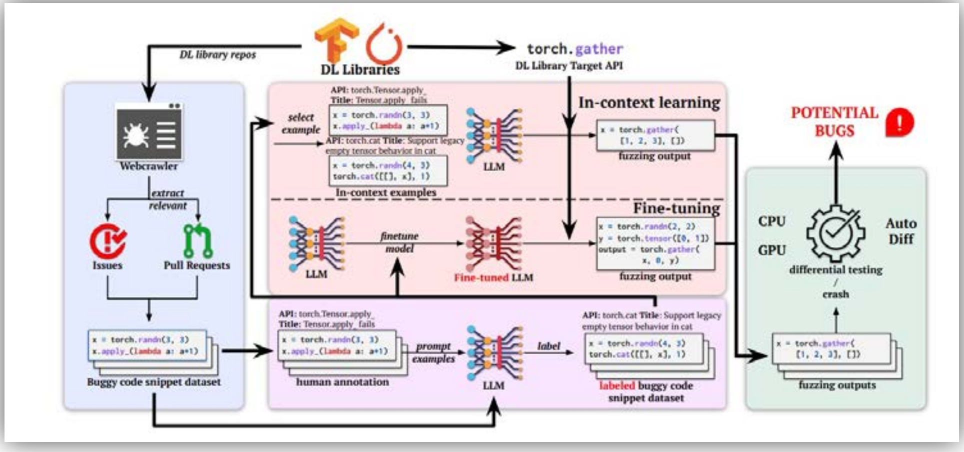
NNSmith: Generating Diverse and Valid Test Cases for Deep Learning Compilers (Liu et al., ASPLOS'23)

NNSmith **<u>依赖于人工写的操作符规范</u>**（用于指明输入张量/属性必须满足的约束条件，输出张量类型）。

现有的深度学习框架测试工作

近期的一些工作指出LLM在深度学习框架测试代码生成任务能展现优异的表现（如UIUC张令明教授团队的TitanFuzz, FuzzGPT等）





Large Language Models are Zero-Shot Fuzzers: Fuzzing Deep-Learning Libraries via Large Language Models (Deng et al., ISSTA'23)

Large Language Models are Edge-Case Fuzzers: Testing Deep Learning Libraries via FuzzGPT (Deng et al., ICSE'24)

现有的深度学习框架测试工作

直接沿用现有方法FuzzGPT效果并不好:

- ~90%生成的代码不是正确的模型优化程序
- 难以找到特定模型和优化API的组合触发MOBs

```
import ...

def pad(img: torch.Tensor, ...) -> torch.Tensor:
    ...
    return F.pad(img, padding=padding, padding_mode="edge")

out = torch.jit.script(pad)
```

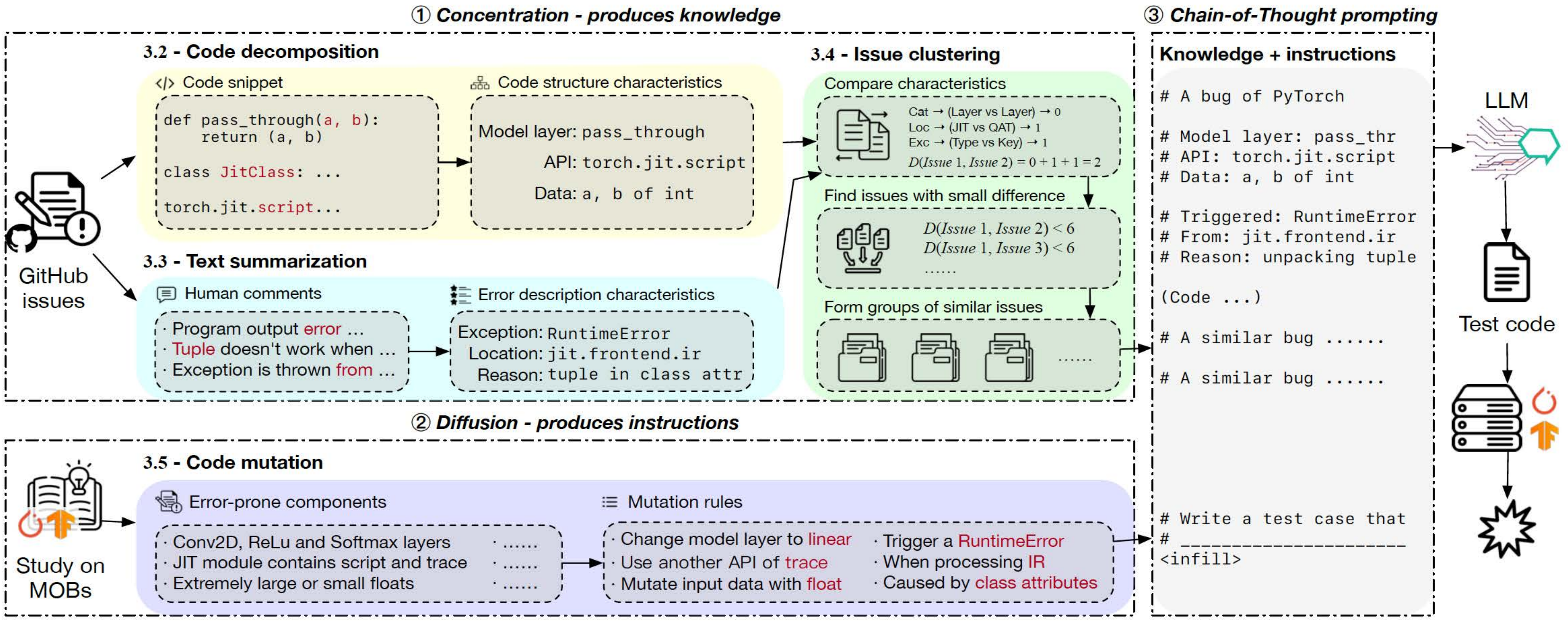
```
print(out)
<file_sep>/torch/csrc/jit/passes/onnx/onnx_tracer.cpp
#include <torch/csrc/jit/passes/onnx/onnx_tracer.h>
```

上述发现其实也很符合常识: MOBs是特定领域的小众bugs (LLMs对训练、推理等任务显然比优化任务更熟悉)

	模型优化模块代码占总代码量比例	模型优化相关测试用例占所有测试用例比例
TensorFlow	1.8%	4.0%
PyTorch	3.5%	2.2%

基于凝练-发散框架的高效MOB检测方法

为了提升LLM**生成模型优化程序**和**检测MOBs**的成功率，我们提出一种新方法

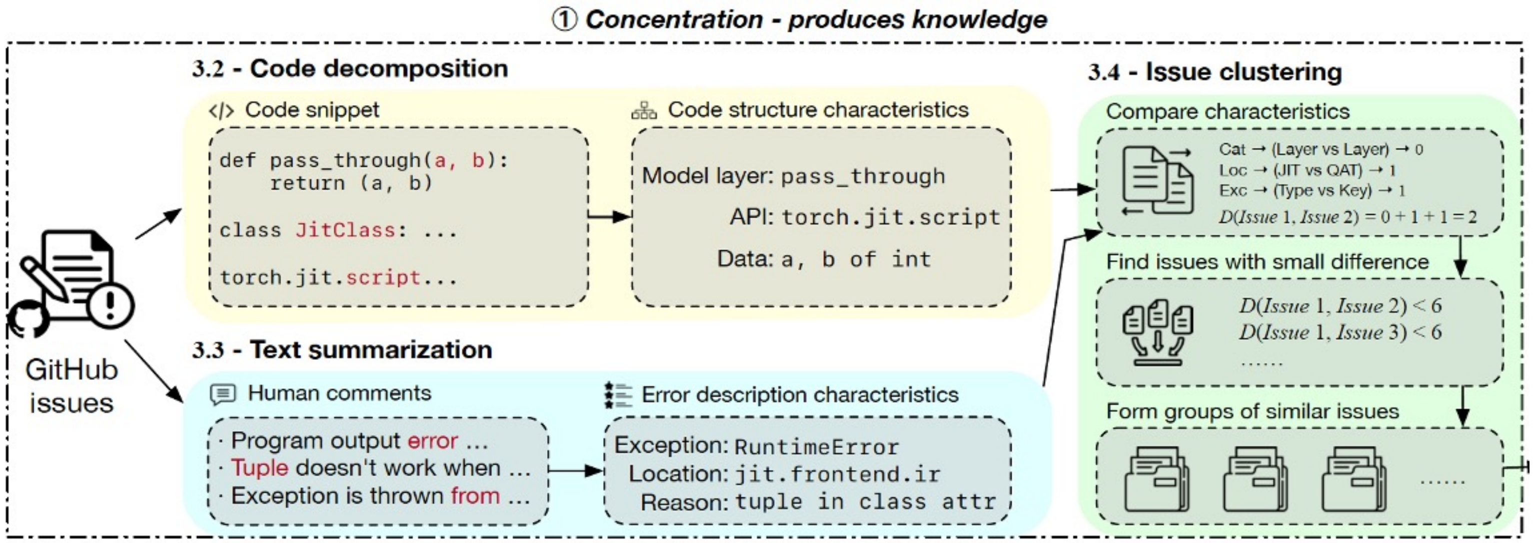


① Concentration - produces knowledge
3.2 - Code decomposition
Code snippet
def pass_through(a, b):
    return (a, b)
class JitClass: ...
torch.jit.script...
Code structure characteristics
Model layer: pass_through
API: torch.jit.script
Data: a, b of int
GitHub issues
3.3 - Text summarization
Human comments
· Program output error ...
· Tuple doesn't work when ...
· Exception is thrown from ...
Error description characteristics
Exception: RuntimeError
Location: jit.frontend.ir
Reason: tuple in class attr
3.4 - Issue clustering
Compare characteristics
Cat → (Layer vs Layer) → 0
Loc → (JIT vs QAT) → 1
Exc → (Type vs Key) → 1
$D(Issue\ 1, Issue\ 2) = 0 + 1 + 1 = 2$
Find issues with small difference
$D(Issue\ 1, Issue\ 2) < 6$
$D(Issue\ 1, Issue\ 3) < 6$
......
Form groups of similar issues
......
② Diffusion - produces instructions
3.5 - Code mutation
Study on MOBs
Error-prone components
· Conv2D, ReLu and Softmax layers ......
· JIT module contains script and trace ......
· Extremely large or small floats ......
Mutation rules
· Change model layer to linear
· Use another API of trace
· Mutate input data with float
· Trigger a RuntimeError
· When processing IR
· Caused by class attributes
③ Chain-of-Thought prompting
Knowledge + instructions
# A bug of PyTorch
# Model layer: pass_thr
# API: torch.jit.script
# Data: a, b of int
# Triggered: RuntimeError
# From: jit.frontend.ir
# Reason: unpacking tuple
(Code ...)
# A similar bug ......
# A similar bug ......
# Write a test case that
# ____________________
<infill>
LLM
Test code

▶ 凝练(Concentration): 高质量的知识增强

基本思路：**从现有MOBs中学习模型优化的知识**

- 从现有issue reports**提取缺陷信息**（异常类型、API、模型信息）
- 将现有的MOBs进行**自动的聚类**
- 给LLM**提供相似的例子**，让知识更加集中



① Concentration - produces knowledge
3.2 - Code decomposition
</> Code snippet
Code structure characteristics
GitHub issues
3.3 - Text summarization
Human comments
Error description characteristics
3.4 - Issue clustering
Compare characteristics
Find issues with small difference
Form groups of similar issues
......

```
def pass_through(a, b):
    return (a, b)

class JitClass: ...

torch.jit.script...
```

Model layer: pass_through
API: torch.jit.script
Data: a, b of int

· Program output error ...
· Tuple doesn't work when ...
· Exception is thrown from ...

Exception: RuntimeError
Location: jit.frontend.ir
Reason: tuple in class attr

Cat → (Layer vs Layer) → 0
Loc → (JIT vs QAT) → 1
Exc → (Type vs Key) → 1
$D(Issue\ 1, Issue\ 2) = 0 + 1 + 1 = 2$

$D(Issue\ 1, Issue\ 2) < 6$
$D(Issue\ 1, Issue\ 3) < 6$
......

Few-shot learning

```
# A bug of PyTorch

# Model layer: pass_thr
# API: torch.jit.script
# Data: a, b of int

# Triggered: RuntimeError
# From: jit.frontend.ir
# Reason: unpacking tuple

(Code ...)
```

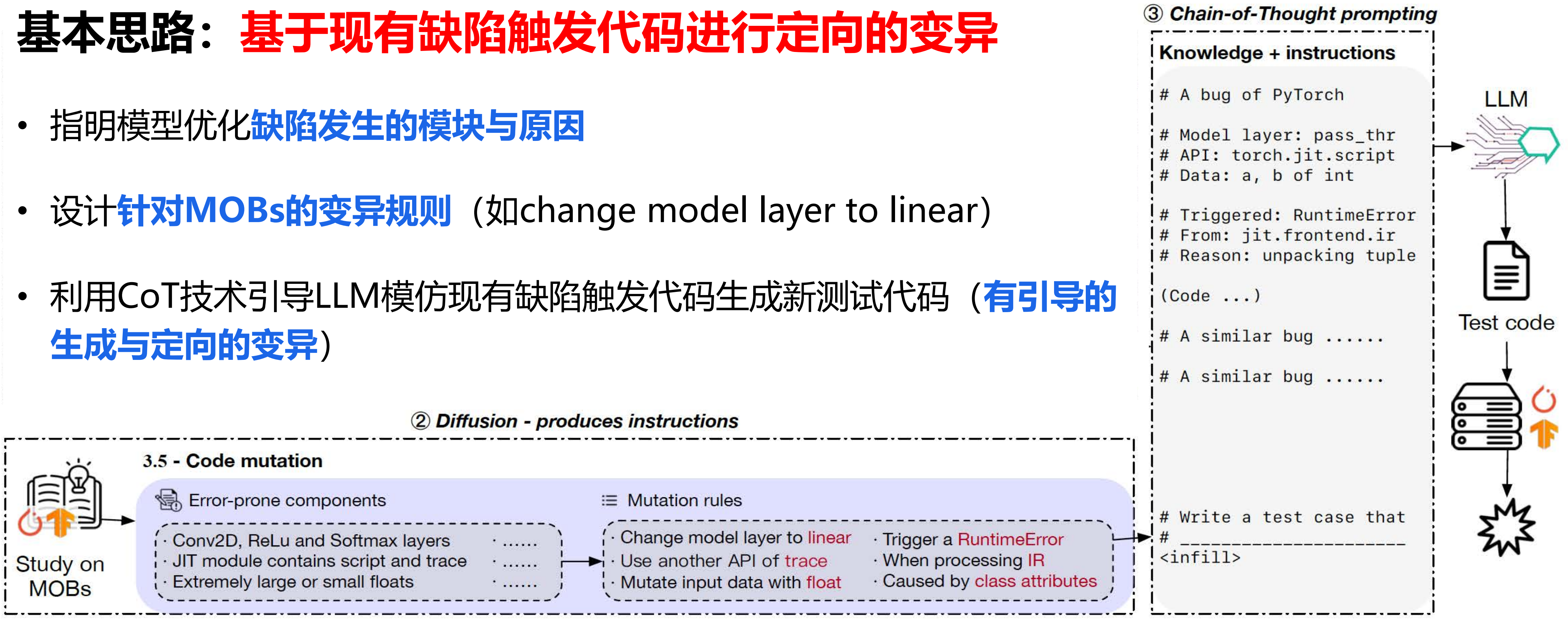
```
# A similar bug
......
```

```
# Another similar bug
......
```

▶ 发散(Diffusion): 高效的类推与生成

基本思路：基于现有缺陷触发代码进行定向的变异

- 指明模型优化**缺陷发生的模块与原因**
- 设计**针对MOBs的变异规则**（如change model layer to linear）
- 利用CoT技术引导LLM模仿现有缺陷触发代码生成新测试代码（**有引导的生成与定向的变异**）



· Conv2D, ReLu and Softmax layers ......
· JIT module contains script and trace ......
· Extremely large or small floats ......

· Change model layer to linear · Trigger a RuntimeError
· Use another API of trace · When processing IR
· Mutate input data with float · Caused by class attributes

```
# A bug of PyTorch

# Model layer: pass_thr
# API: torch.jit.script
# Data: a, b of int

# Triggered: RuntimeError
# From: jit.frontend.ir
# Reason: unpacking tuple

(Code ...)

# A similar bug ......

# A similar bug ......


# Write a test case that
# ______________________
<infill>
```

YanHui - 开源的实现

GitHub链接：https://github.com/YanHui-2024/YanHui

颜回 (B.C. 521 -- 481)

子贡曾经评价颜回（孔子七十二门徒之首）：“回也闻一以知十”。

Prompts

```
# Here is a bug of PyTorch
# Model:        nn.Module
# API:          torch.jit.script
# Data:         "abcd"
# Triggered:    RuntimeError
# From:         torch.jit.frontend
# Reason:       desugar of str is not handled in JIT
(code …)

# And a similar bug of PyTorch
# Model:        nn.Module
# API:          torch.jit.script
# Data:         **{ "hello": None }
# Triggered:    RuntimeError
# From:         torch.jit.frontend
# Reason:       expanding dictionary is not handled
(code …)

......

# Write a test case of PyTorch that fails to handle
# the feature of Python.
```

YanHui能举一反三

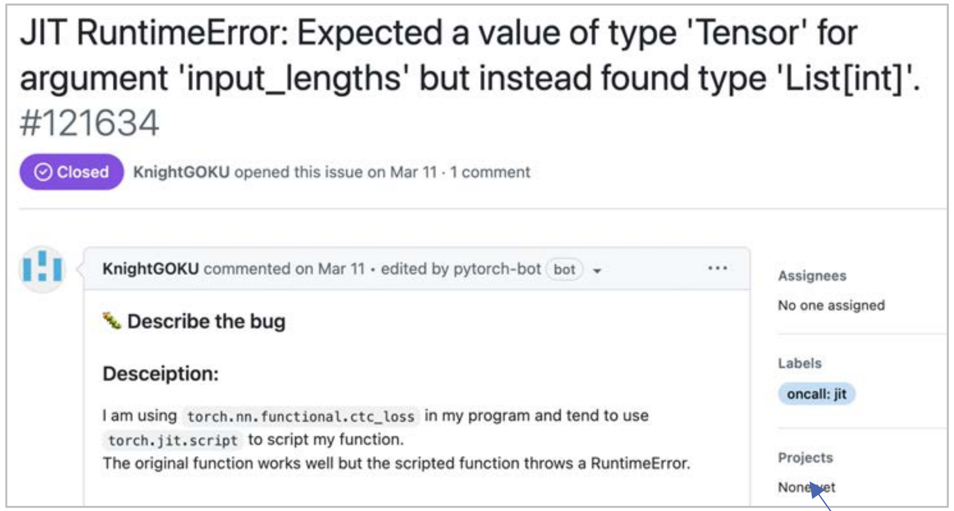
```
# JIT does not support the **kwargs syntax
class MyModel(nn.Module):
    def __init__(self, a=1):
        ...

    def forward(self, x, **kwargs):
        return x + 1

m = MyModel()
scripted = torch.jit.script(m)
scripted(torch.tensor([2]))
```

▶ YanHui的效果

- **生成模型优化程序的成功率：36%**（在GPT-4上的效果）
- 在PyTorch和TensorFlow中共检测到**17个被开发者确认的新MOBs**（部分已经修复）



JIT RuntimeError: Expected a value of type 'Tensor' for argument 'input_lengths' but instead found type 'List[int]'. #121634

Closed KnightGOKU opened this issue on Mar 11 · 1 comment

KnightGOKU commented on Mar 11 • edited by pytorch-bot bot ···

Describe the bug

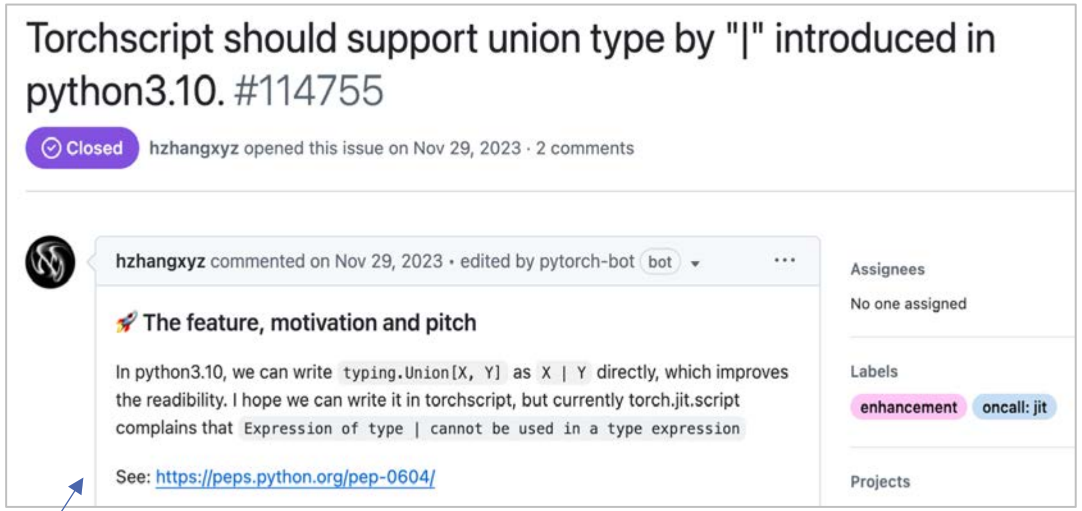
Desceiption:

I am using `torch.nn.functional.ctc_loss` in my program and tend to use `torch.jit.script` to script my function.
The original function works well but the scripted function throws a RuntimeError.

Assignees
No one assigned

Labels
oncall: jit

Projects
None yet



Torchscript should support union type by "|" introduced in python3.10. #114755

Closed hzhangxyz opened this issue on Nov 29, 2023 · 2 comments

hzhangxyz commented on Nov 29, 2023 • edited by pytorch-bot bot ···

🚀 The feature, motivation and pitch

In python3.10, we can write `typing.Union[X, Y]` as `X | Y` directly, which improves the readibility. I hope we can write it in torchscript, but currently torch.jit.script complains that `Expression of type | cannot be used in a type expression`

See: https://peps.python.org/pep-0604/

Assignees
No one assigned

Labels
enhancement oncall: jit

Projects

PyTorch中的两个被确认和修复的模型优化缺陷

我们团队近期的一点探索与实践

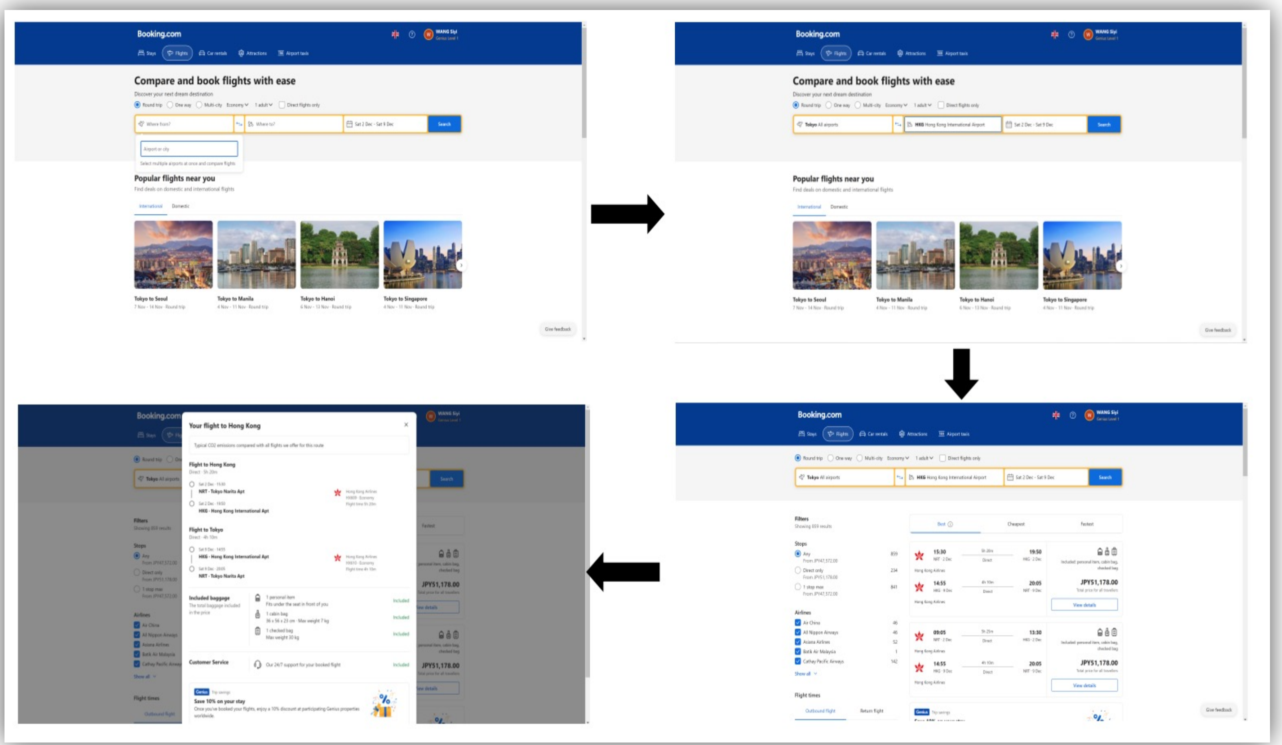
ISSTA 24 如何利用LLM辅助检测**特定领域**的“**小众**”**bug**?

ICSME 2024 如何利用**视觉语言模型**对**网页界面**进行高效测试?

ASE 2024 SACRAMENTO 如何利用**多智能体强化学习**提升**网页界面**测试性能?

▶ 背景：网页应用界面自动化测试

- **测试目标：**
  - 高覆盖率 (页面、元素、代码等)
  - 检测错误 (页面崩溃、渲染错误等)
- **关键任务：**
  - 交互元素选择 (如“搜索”按钮)
  - 用户输入生成 (如“出发日期”)
- **技术挑战：**
  - 如何自动理解业务逻辑以高效生成逻辑性强的动作序列和有效的用户输入?



示例场景：在Booking.com订机票

▶ 现有的网页应用界面自动化测试工作

现有工作	交互元素选择	用户输入生成
Crawljax [Mesbah et al., 2011]	抽取页面跳转模型state-flow graph，基于DFS算法进行页面探索	随机或用户提供
DIG [Biagiola et al., 2019]	抽取navigational graph，基于diversity-based heuristics选择元素	随机
WebExplor [Zheng et al., 2021]	基于好奇心驱动的Q-learning算法	随机
QExplore [Sherin et al., 2023]	基于Q-learning算法	基于自定义的schema自动生成
WebLinx [Lu et al., 2024]	基于大模型，但需要用户提供指令，无法全自动探索网页 (比如Create a task for a Career Fair on Google Calendar)	大模型生成
Seeclick [Cheng et al., 2024]		

▶ 现有的网页应用界面自动化测试工作

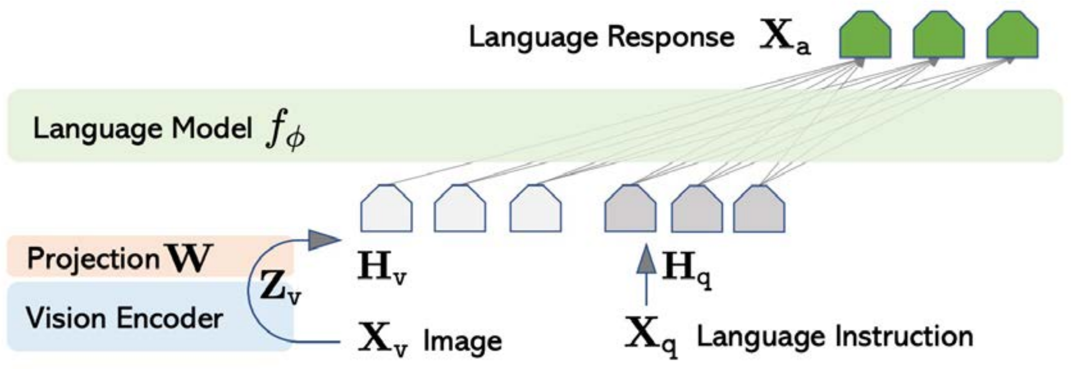
现有工作	交互元素选择	用户输入生成
Crawljax [Mesbah et al., 2011]	抽取页面跳转模型state-flow graph，基于DFS算法进行页面探索	随机或用户提供
WebLinx [Lu et al., 2024]	基于大模型，但需要用户提供指令，无法全自动探索网页 (比如Create a task for a Career Fair on Google Calendar)	大模型生成
Seeclick [Cheng et al., 2024]		

尽管有不少相关研究，但目前仍然没有一篇工作同时解决好前面提到的两个问题：

- **问题1：**如何生成有逻辑性的动作序列，实现网页应用界面的自动化遍历？
- **问题2：**如何在与网页交互过程中，生成合规的用户输入，以提升测试覆盖率？

多模态LLM为网页应用界面测试带来新的机会

[NeurIPS'23] Visual Instruction Tuning (LLaVA) built towards GPT-4V level capabilities and beyond



LLaVA (<u>L</u>arge <u>L</u>anguage-<u>a</u>nd-<u>V</u>ision <u>A</u>ssistant) 利用投影矩阵技术将下面两个模型进行结合：

- 预训练的视觉编码器CLIP ViT-L/14
- 基于LLaMA微调的开源大语言模型Vicuna

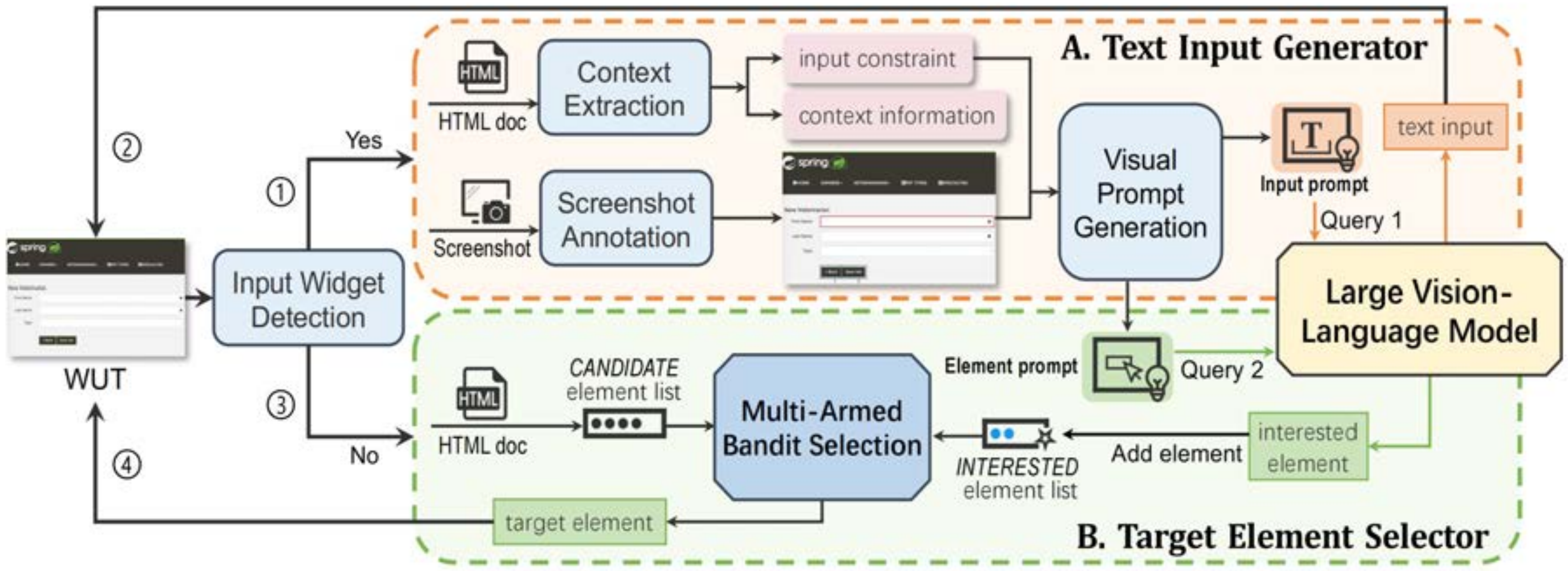
开源视觉语言模型LLaVa具备以下能力：

- **视觉理解：** 得益于CLIP，LLaVa能识别网页上的各种元素（按钮、图片、文本框）
- **文本解析与生成：** 得益于Vicuna，Llava有强大的自然语言处理能力
- **交互模拟：** 结合上述两种能力，LLaVA 能模拟用户与网页之间的互动过程，如点击链接、填写表单等

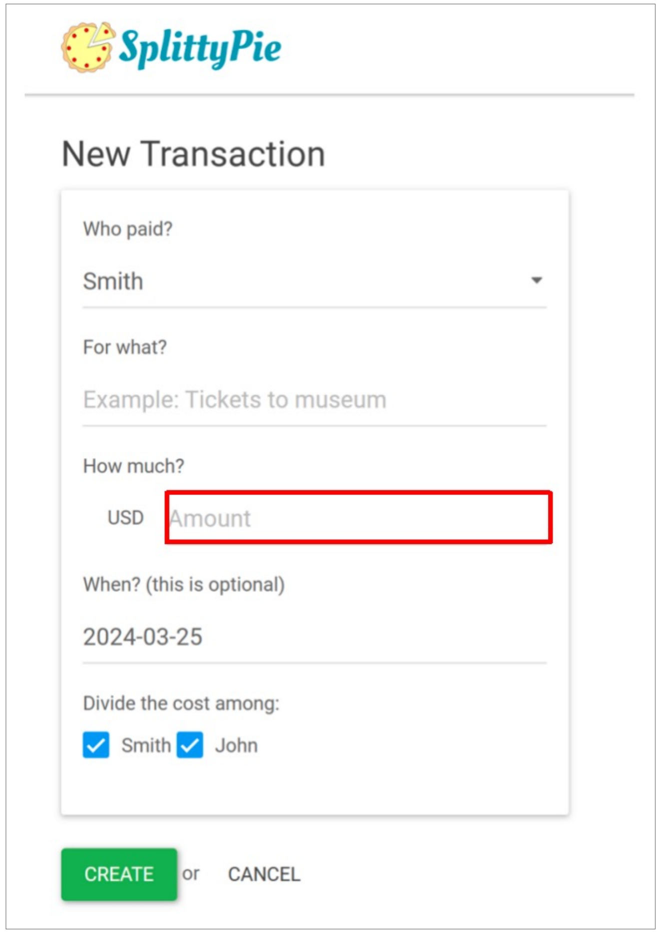
把网页界面自动化测试建模成**Visual Question Answering (VQA) 任务**

利用视觉语言模型同时解决**<u>交互元素选择</u>**与**<u>用户输入生成</u>**两个问题

▶ VETL: LVLM-Empowered Web Testing Loop



▶ VETL的文本输入生成模块


SplittyPie

New Transaction

Who paid?

Smith

For what?

Example: Tickets to museum

How much?

USD Amount

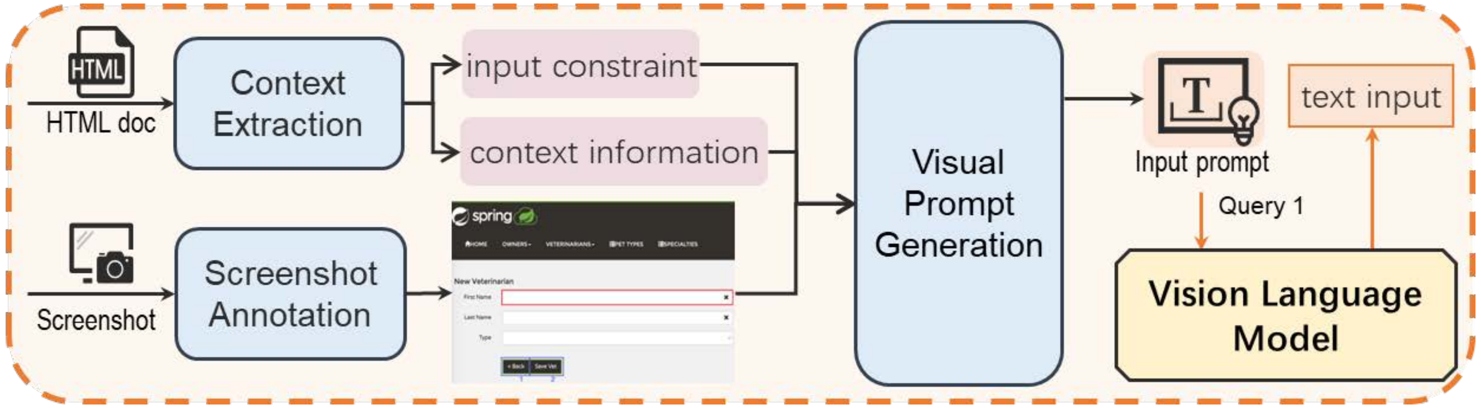
When? (this is optional)

2024-03-25

Divide the cost among:

☑ Smith ☑ John

CREATE or CANCEL



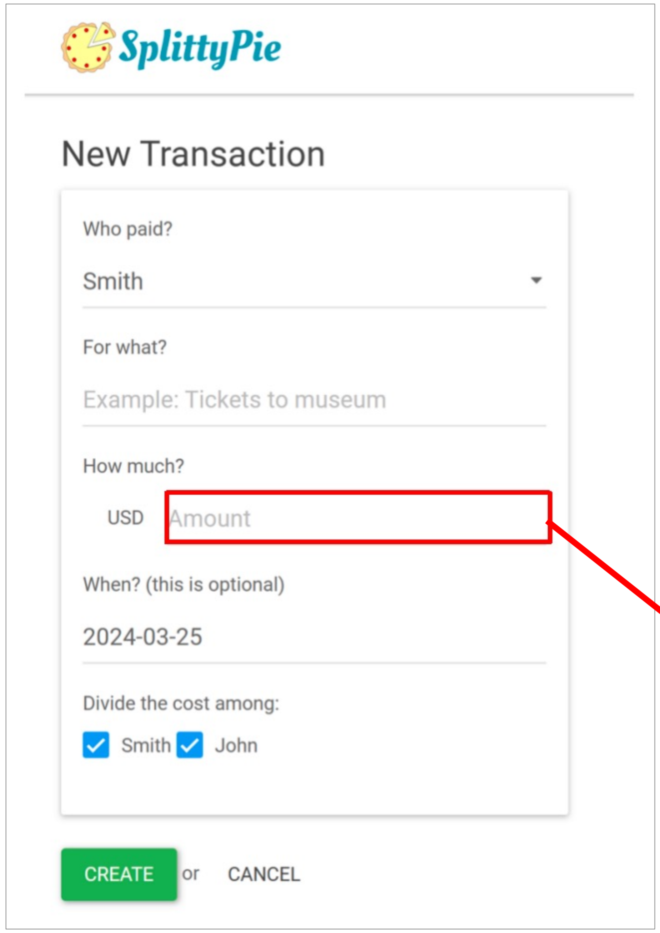
多模态输入：

- 当前网页HTML
- 当前网页的截图

关键处理步骤：

- **上下文提取**（帮助大模型理解场景）
- **Input constraints提取**（防止生成无效输入）
- **截图标注**（减少大模型的幻觉）
- **Input prompt生成**
- **利用LVLM生成用户输入**

▶ 文本输入生成模块Prompt设计


SplittyPie

New Transaction

Who paid?

Smith

For what?

Example: Tickets to museum

How much?

USD Amount

When? (this is optional)

2024-03-25

Divide the cost among:

✓ Smith ✓ John

CREATE or CANCEL

Role Playing (RP)	Visual Info. (VI)	Global Context (GC)
... web tester ... ... generate text inputs ...	... marked with red frame ...	You are viewing the web page entitled: {*HTML Title*}.
Local Context (LC)	**Input Widget (IW)**	**Output Structure (OS)**
Text near the input box: {*Text of Nearby Element*}.	...{*Attribute&Constraints of Input Box*}...	... Use the format : Generated Input Text: [answer]

Based on the prompt with rich information, LVLM may correctly generate 1.99 for the example.

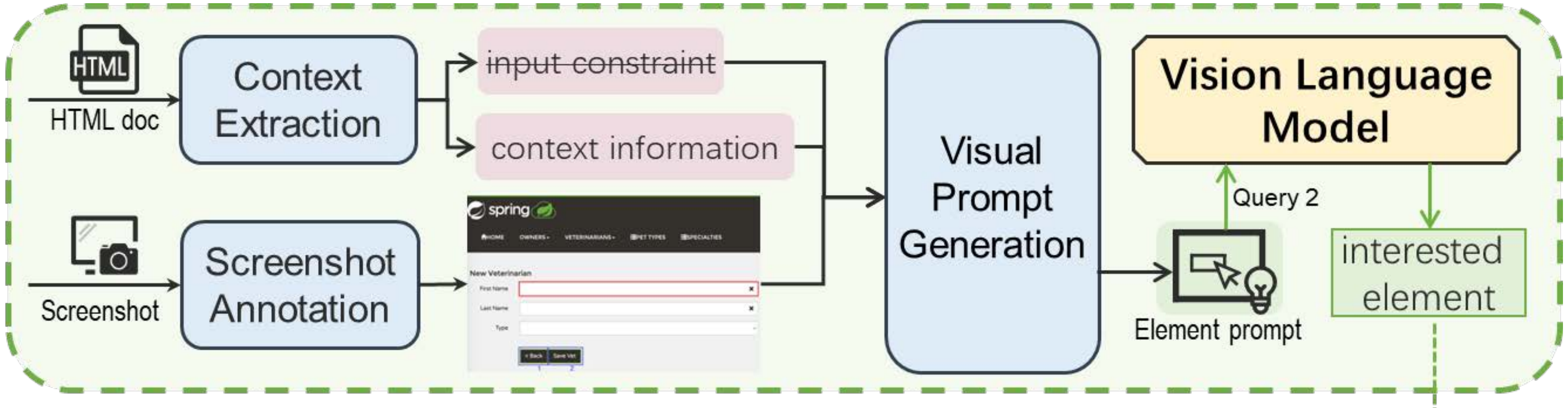
VETL的交互元素选择模块

多模态输入：

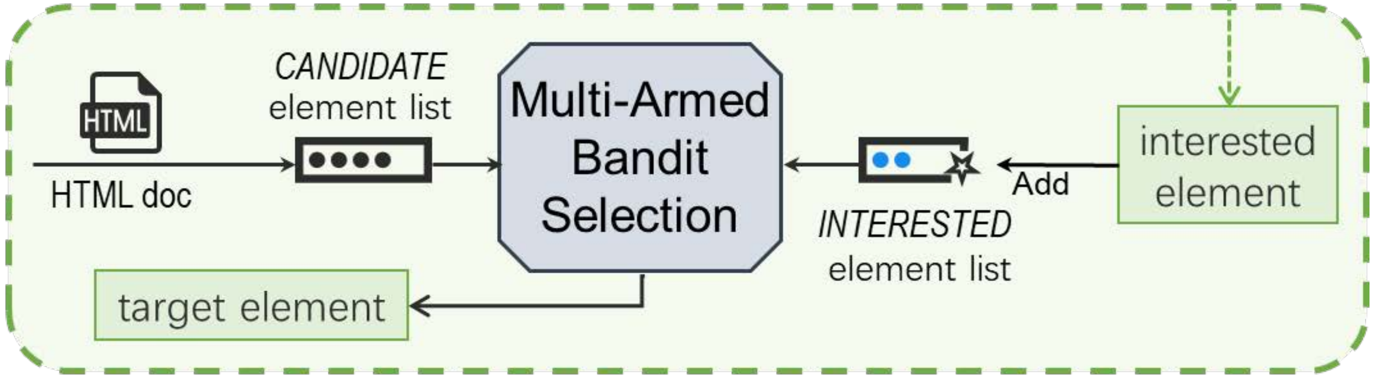
- 当前网页HTML
- 当前网页的截图

关键处理步骤：

- **上下文提取**（帮助大模型理解场景）
- **截图标注**（减少大模型的幻觉）
- **Element Prompt生成**
- **可交互元素的初筛**（LVLM决定）
- **交互动作的决策**（MAB算法决定）



Overview of LVLM-based Element Selector



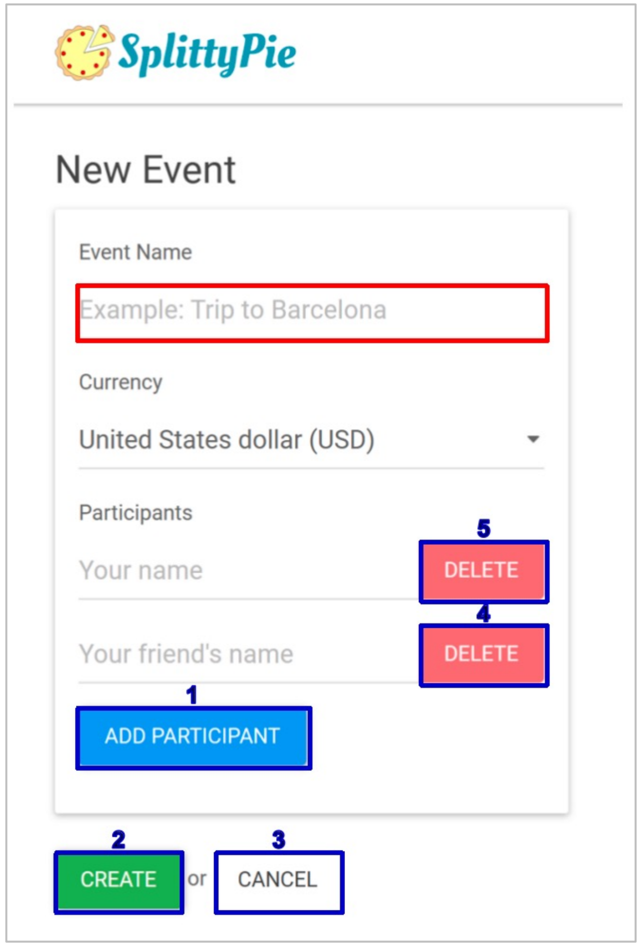
Overview of MAB

▶ 交互元素选择模块Prompt设计

Role Playing (RP)	Visual Information (VI)	Global Context (GC)
...web tester ... ... select a button ...	... red & blue frame ... buttons within {Range of Button Numbers}	
Local Context (LC)	**Input Widget (IW)**	**Output Structure (OS)**
	The input box will be filled with: *{Generated Text Input}*.	... Selected Button Number: [answer]

- *VI: The input box to be filled is marked with a red frame and the buttons to be selected are marked within {Range of Button Numbers} and blue frames.*
- *IW: The input box will be filled with: 199.*

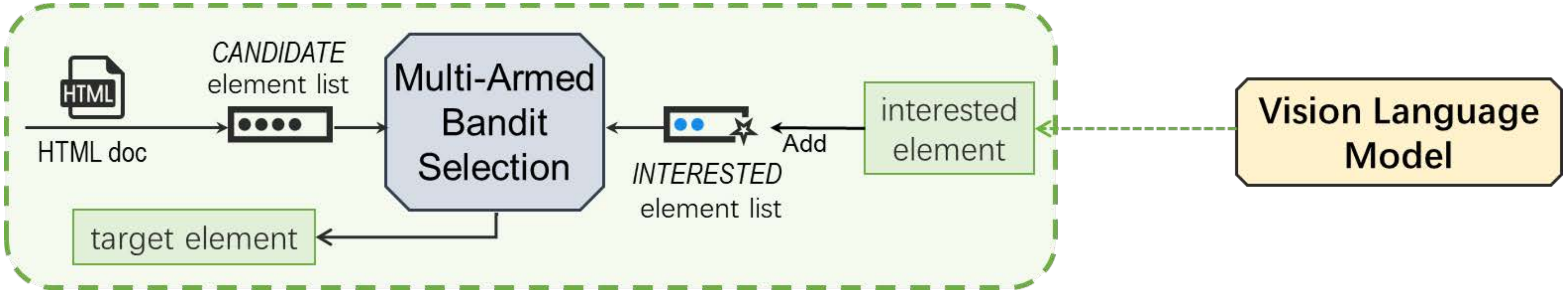
Based on the prompt with rich information, LVLM may generate answer: 2.



▶ 基于MAB算法的动作决策

目的是为了实现探索与利用的平衡（解决Exploration-Exploitation Dilemma）[1]**：**

- **探索：** Explore the website without guidance to discover unknown web states.
- **利用：** Greedily exploit the profit-maximizing actions provided by the LVLM.

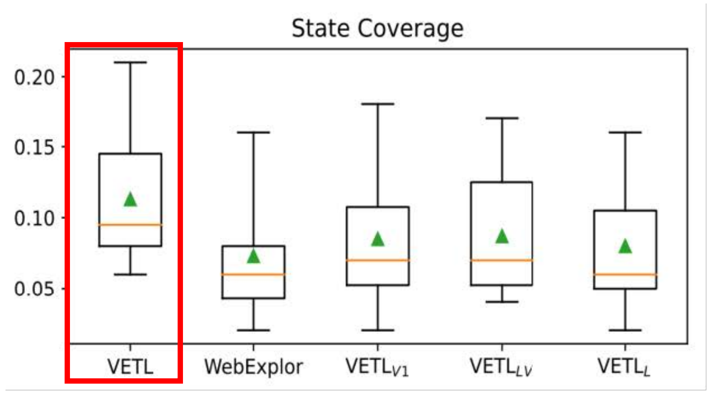


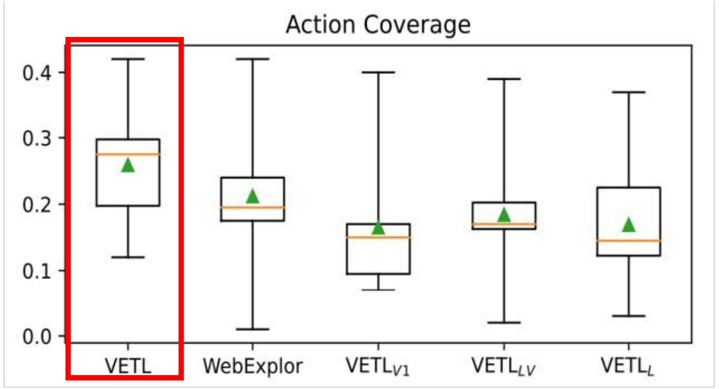
[1] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction.MIT press, 2018.

▶ VETL能有效提升测试覆盖率吗?

Comparison of VETL, WebExplor, and VETL’s variants on open-source benchmark websites

WUT	#Visited Web State					#Discovered Web Action				
	VETL	*WebExplor*	$VETL_{V1}$	$VETL_{LV}$	$VETL_{L}$	VETL	*WebExplor*	$VETL_{V1}$	$VETL_{LV}$	$VETL_{L}$
DimeShift	**10.8**	3.6	7.2	7.8	8.6	**121.8**	96.2	49.2	45	71
PetClinic	**9**	3.8	8.2	8	8.4	**27.4**	16.2	24	25.8	23.4
Phoenix	**7**	5.2	4.8	6	6.4	**35**	33.6	30.8	31.2	32.2
SplittyPie	**5.6**	2.2	3	3	2.4	**24**	20.6	15.6	17.8	14.4
avg.	**8.1**	3.7	5.8	6.2	6.45	**52.05**	41.65	29.9	29.95	35.25

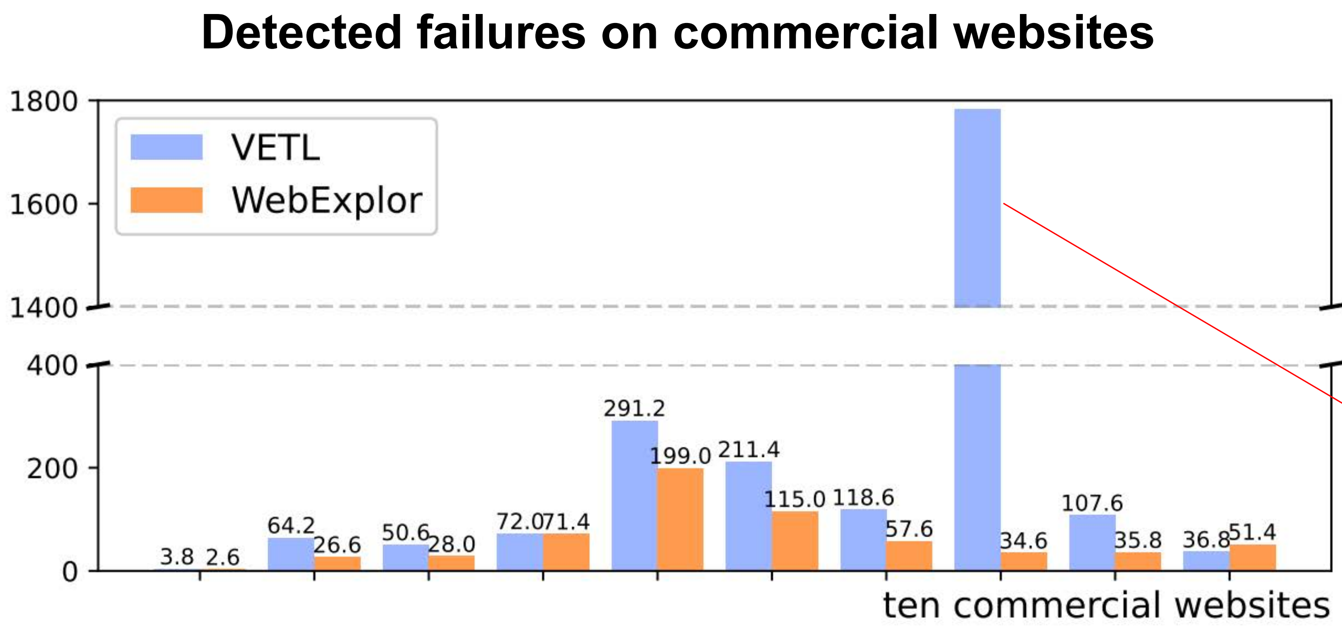




Comparison on commercial websites

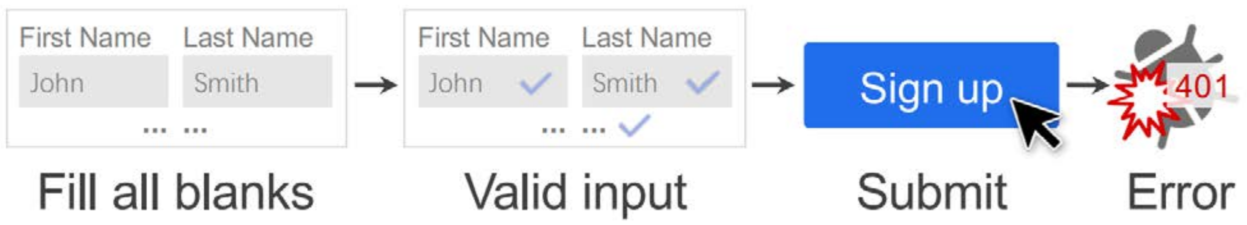
无论是在开源项目还是闭源网站上，VETL在同等时间内均比SOTA方法WebExplor **访问更多页面，发现更多可交互的元素，实现更高的测试覆盖**。

▶ VETL能帮助检测到真实的网页缺陷吗?



在top商业网站上的实验表明，同等时间内，**VETL比WebExplor触发更多异常网页行为**(从console log分析)

大量异常是由于在搜索框中输入复杂长文本导致**过量内存消耗**，触发ERR_INSUFFICIENT_RESOURCES



在正确填写注册用户的信息但CAPTCHA加载失败的情况下会触发**401错误**

▶ 我们团队近期的一点探索与实践

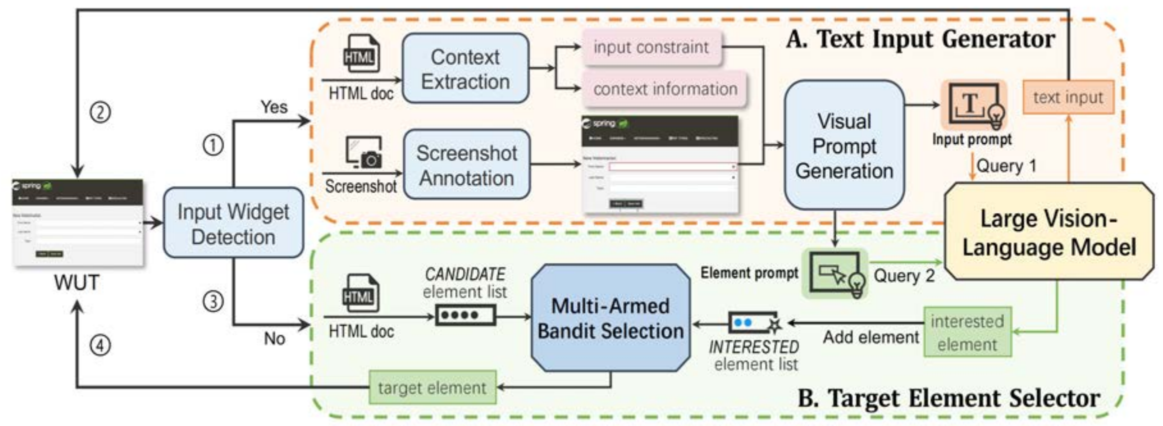
ISSTA 24 如何利用LLM辅助检测**特定领域**的“**小众**”**bug**？

ICSME 2024 如何利用**视觉语言模型**对**网页界面**进行高效测试？

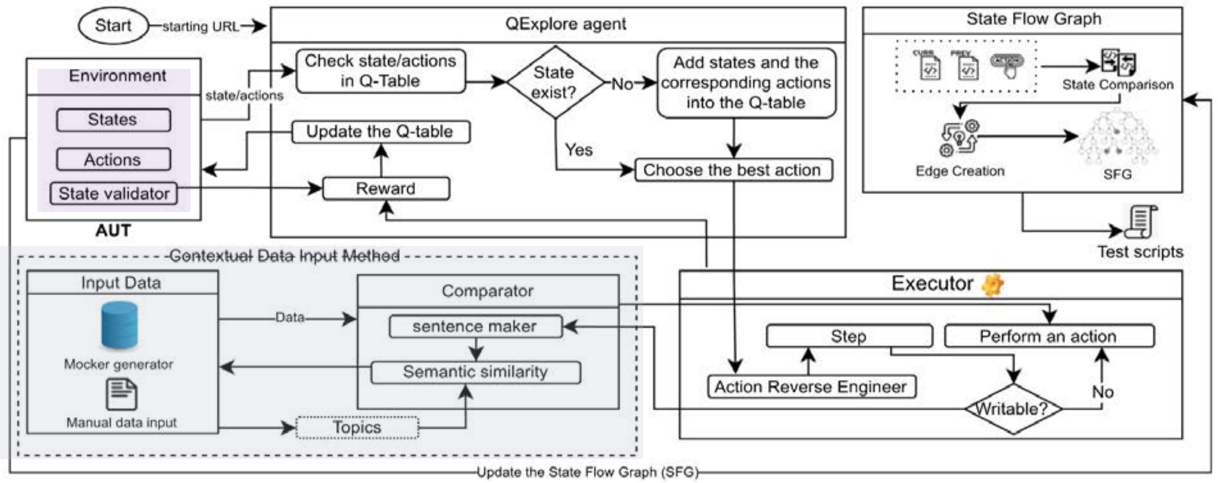
ASE 2024 SACRAMENTO 如何利用**多智能体强化学习**提升**网页界面**测试性能？

▶ 研究动机

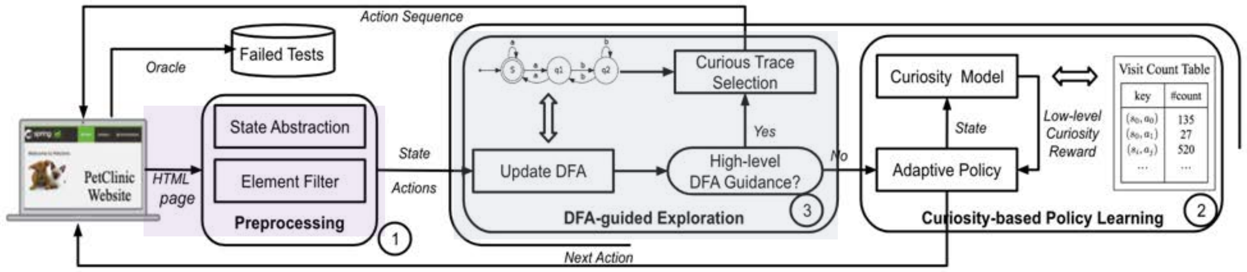
VETL (ICSME 2024)



QExplore (JSS 2023)

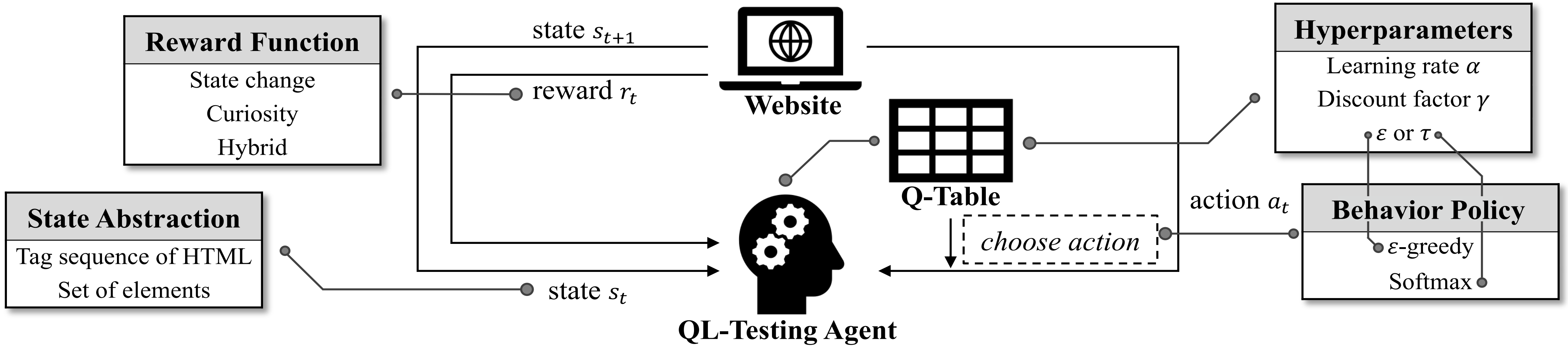


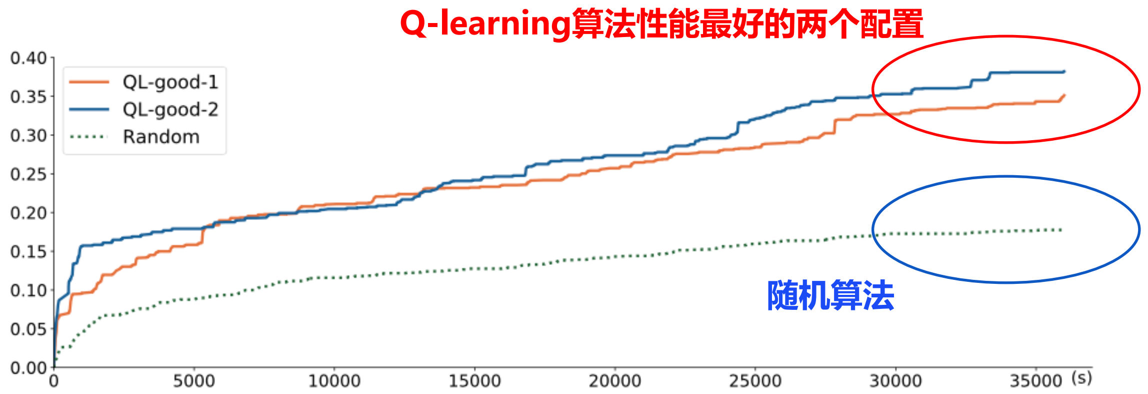
WebExplor (ICSE 2021)



现有的网页测试方法，不论采用何种AI技术，基本都是单智能体方法，性能存在上限，难以继续提升。

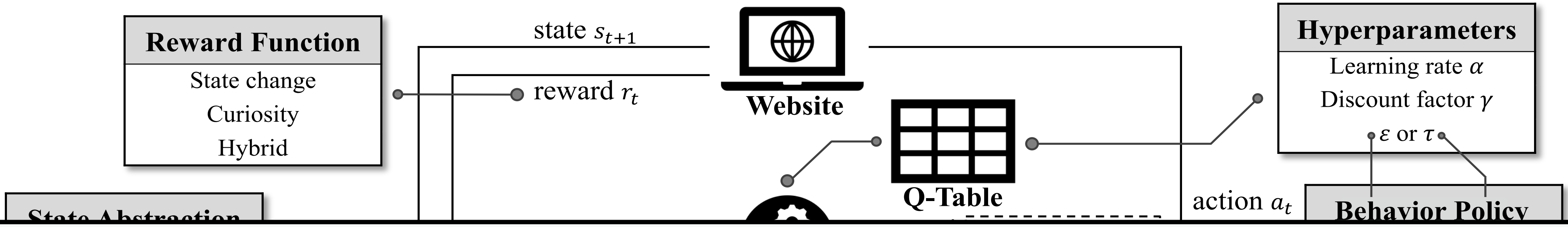
▶ 研究动机



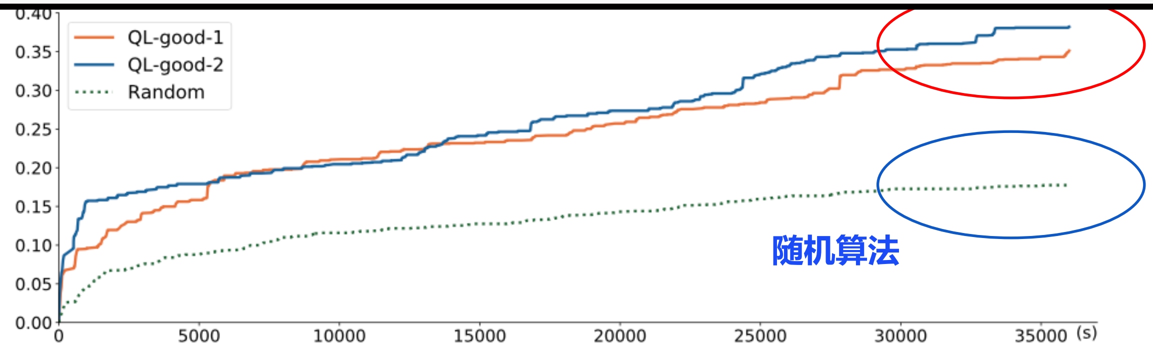


我们基于使用最广的Q-learning算法开展了实证研究，发现基于**单智能体Q-learning**的工具**在测试一个大型网站10小时后性能慢慢达到瓶颈**。

研究动机

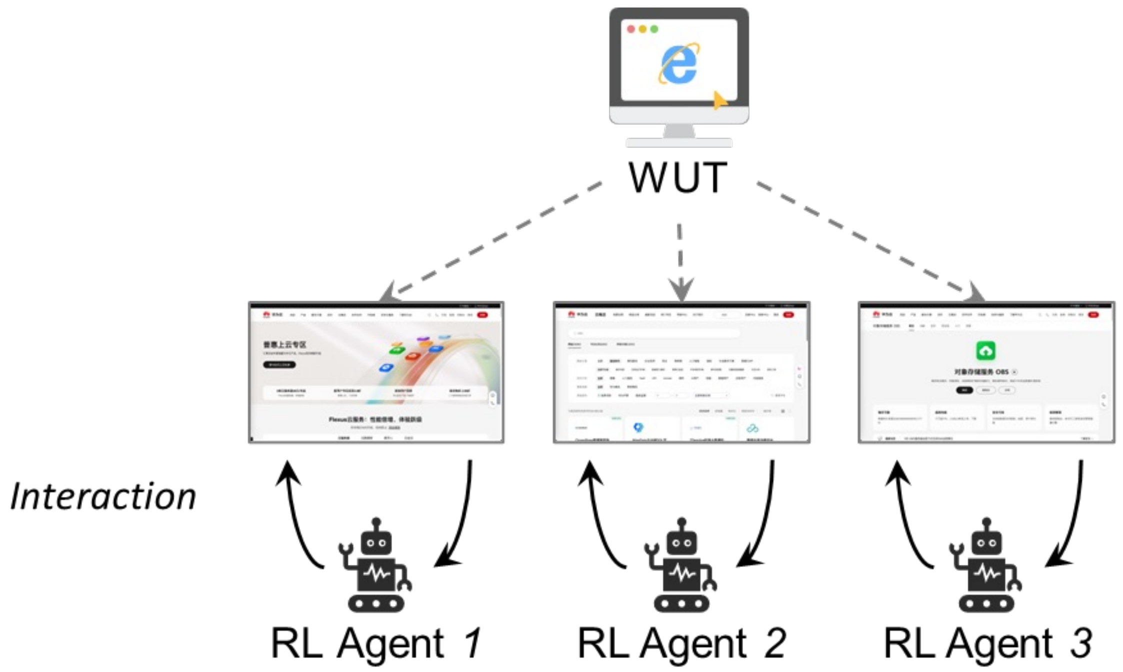


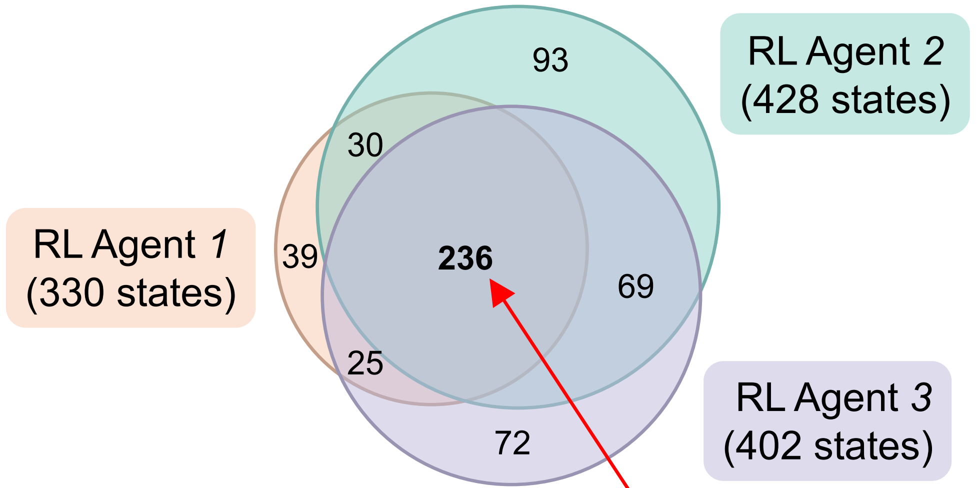
如果并行跑多个Q-learning算法驱动的智能体呢？在同样的时间内，多智能体驱动的测试工具能达到更高的覆盖率吗？



我们基于使用最广的Q-learning算法开展了实证研究，发现基于**单智能体Q-learning**的工具**在测试一个大型网站10小时后性能慢慢达到瓶颈**。

▶ 简单地并行跑多个Q-Learning智能体的效果

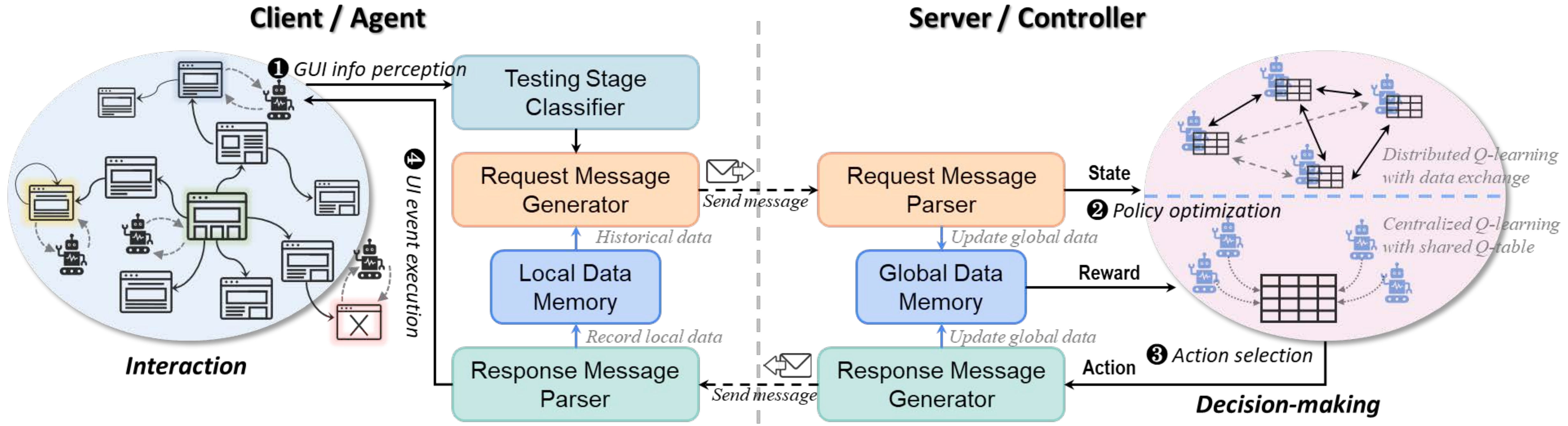




总覆盖率有提升，但探索的**冗余度**太高了！

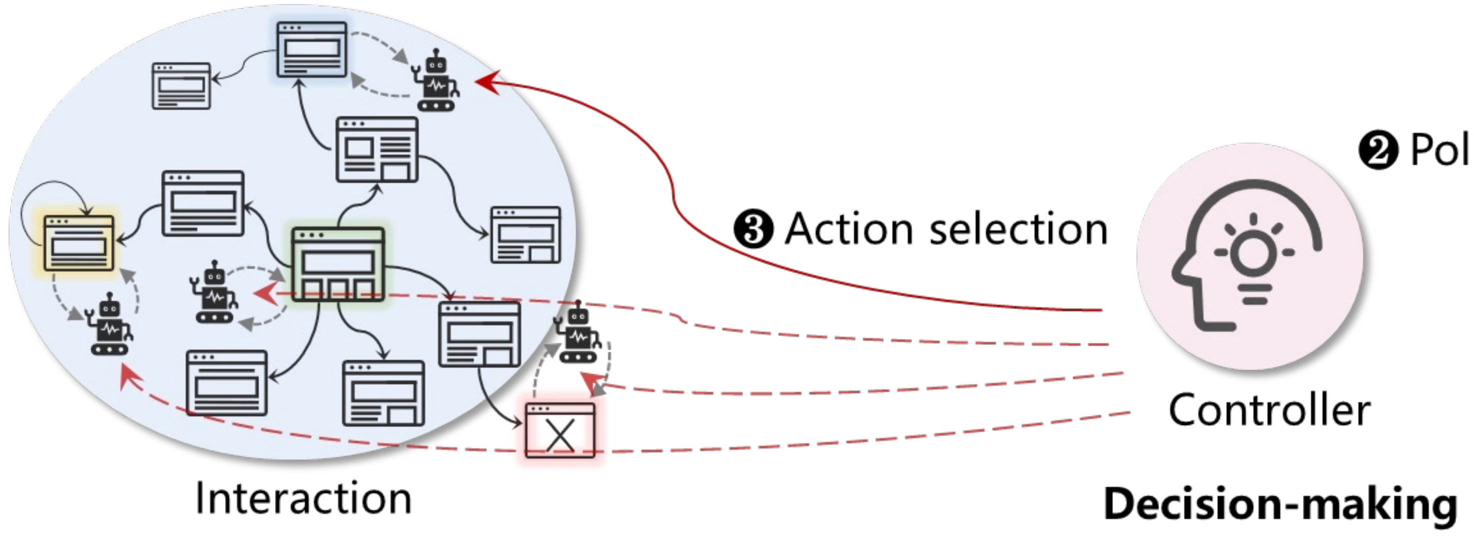
▶ MARL-based GUI Testing System (MARG)

为降低多个Q-Learning智能体状态探索的冗余度（**提升整体测试效率**），我们提出一种新方法



▶ MARL-based GUI Testing System (MARG)

Insight: Separating the process into **Interaction** and **Decision-making**



❶ GUI info perception

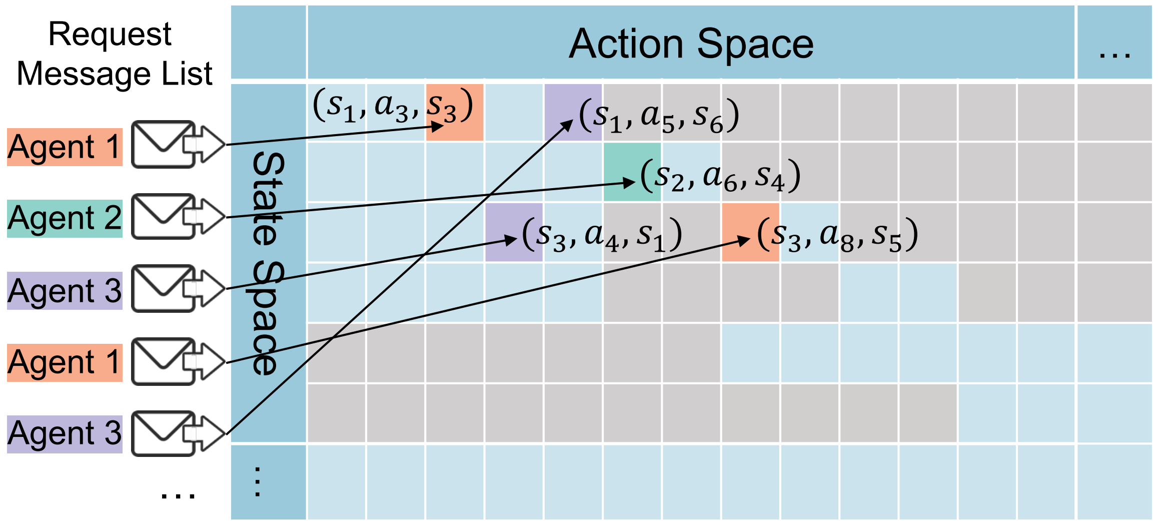
❹ UI event execution

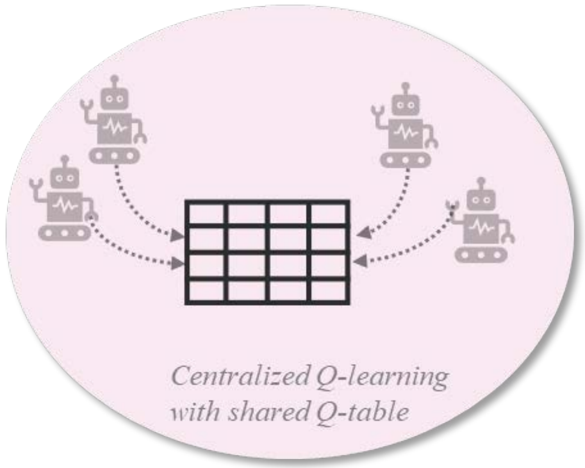
Coordination and Cooperation

- **Which data to share?**
  - Experience: optimal action selection on specific webpages
- **How to share?**
  - Q-learning algorithm is for single agent
  - Communication Efficiency
    - Effectiveness
    - Overhead

Data Sharing Schemes: $MARG_C$

Centralized Q-learning with shared Q-table

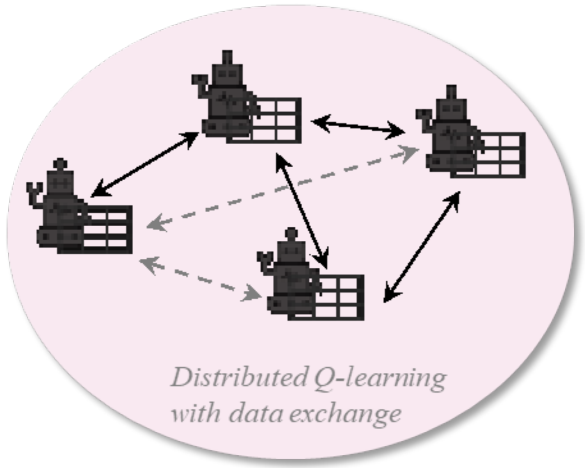




- Ease of implementation
- Timely updates of testing experiences across all agents
- Potential inefficiency with Q-table scalability

Data Sharing Schemes: MARG_D

Distributed Q-learning with data exchange



Algorithm 1: Distributed Q-learning for the k-th agent

```
Input: Agent $k$: previous state $s$, executed action $a$, current state $s'$,
       action space of current state $\mathcal{A}_{s'}$, reward $r$
       Global: number of agents $N$, Q-tables $Q_1, Q_2, \ldots Q_N$, and
       hyperparameters $\alpha, \gamma, \varepsilon$
Output: $Q_k$, chosen action $a'$
1  if $s'$ not in $Q_k$ then
2      initialize $Q_k(s', \mathcal{A}_{s'})$
3  for $j = 1, \ldots, N; j \neq k$ do
4      if $s'$ in $Q_j$ then
5          tempQ.append($Q_j(s', \mathcal{A}_{s'})$)
6  if tempQ is empty then
7      update $Q_k(s, a)$ using Equation (2)
8      Choose $a'$ from $\mathcal{A}_{s'}$ using Equation (3) on $Q_k(s', \cdot)$
9  else
10     update $Q_k(s, a)$ using Equation (4)
11     for $a_i$ in $\mathcal{A}_{s'}$ do
12         $Q^*_{s'}(a_i) = \Sigma_{Q_j \in tempQ} Q_j(s', a_i)$
13     Choose $a'$ from $\mathcal{A}_{s'}$ using Equation (3) on $Q^*_{s'}(\cdot)$
```

Choosing action

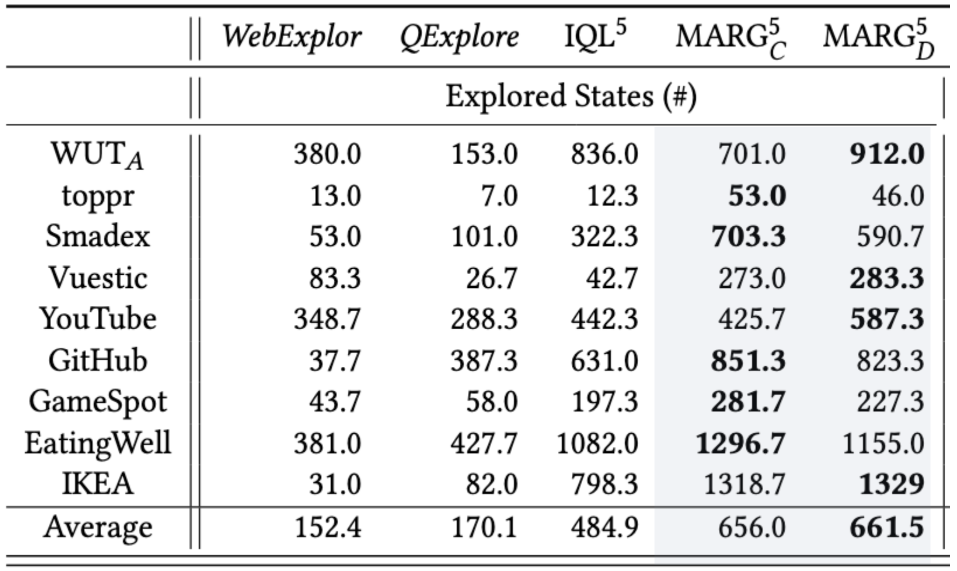
$$a^* = \begin{cases} \underset{a' \in \mathcal{A}_s}{\operatorname{argmax}}\, Q(s, a') & \text{with probability } 1-\varepsilon, \\ \text{random} \in \mathcal{A}_{s'} & \text{with probability } \varepsilon. \end{cases} \tag{3}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r_t + \gamma \max_{a' \in \mathcal{A}_{s'}} Q(s', a') - Q(s,a)] \tag{2}$$

$$Q_k(s,a) \leftarrow \alpha[r + \frac{\gamma}{l} \sum_{Q_j \in tempQ} Q_j(s', \underset{a' \in \mathcal{A}_{s'}}{\operatorname{argmax}}(Q_k(s', a'))) - Q_k(s,a)] + Q_k(s,a) \tag{4}$$

Updating Q-value

The Performance of MARG



	WebExplor	*QExplore*	IQL^5	$MARG_C^5$	$MARG_D^5$
	Explored States (#)				
WUT_A	380.0	153.0	836.0	701.0	**912.0**
toppr	13.0	7.0	12.3	**53.0**	46.0
Smadex	53.0	101.0	322.3	**703.3**	590.7
Vuestic	83.3	26.7	42.7	273.0	**283.3**
YouTube	348.7	288.3	442.3	425.7	**587.3**
GitHub	37.7	387.3	631.0	**851.3**	823.3
GameSpot	43.7	58.0	197.3	**281.7**	227.3
EatingWell	381.0	427.7	1082.0	**1296.7**	1155.0
IKEA	31.0	82.0	798.3	1318.7	**1329**
Average	152.4	170.1	484.9	656.0	**661.5**

在9个大型网站上评估5个智能体的MARG系统：

- 同等时间内，比SOTA方法WebExplor以及QExplore**探索更多状态（4.34倍、3.89倍）**
- 同等时间内，比并行运行5个Q-learning智能体(无协同) **探索的状态多36.42%**

► MARG$_C$ V.S. MARG$_D$

Metrics	MARG$_C$	MARG$_D$
Explored States	656.0	**661.5**
Detection Actions	**3160.8**	3135.7
Executed Unique Actions	531.1	**626.1**
Detected Failures	34.2	**39.5**

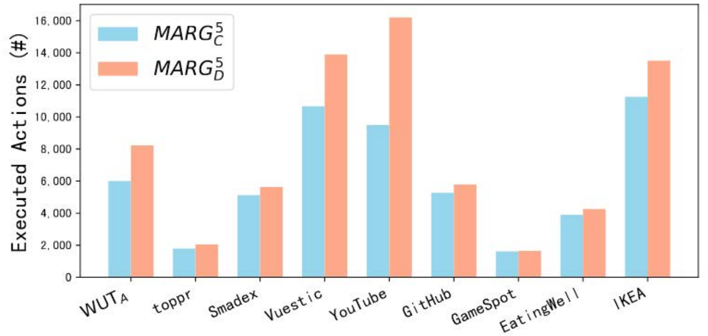


Figure 6: Comparison of total executed actions of MARG$_C^5$ and MARG$_D^5$

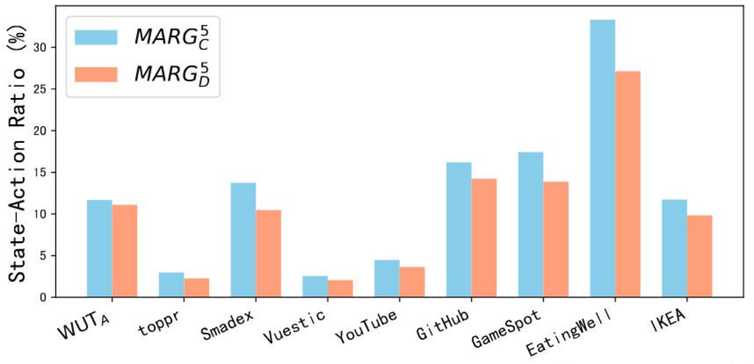


Figure 7: State-to-action ratios of MARG$_C^5$ and MARG$_D^5$

Centralized q-learning with a shared q-table

- Highly centralized nature superior info propagation capabilities
- Higher communication overhead

Distributed q-learning with data exchange

- Less overhead
- Better overall performance

从多个维度的对比实验看，
MARG$_D$策略均优于MARG$_C$

PART 04 总结与展望

报告总结

- **基于AI算法的软件自动化测试技术**
  - 启发式算法、多目标优化算法、深度学习、强化学习、大模型技术
- **我们团队近期在AI4Testing方向上的探索**
  - [YanHui @ ISSTA 2024] 如何利用LLM辅助检测特定领域的“小众”bug?
  - [VETL @ ICSME 2024] 如何利用视觉语言模型对网页界面进行高效测试?
  - [MARG @ ASE 2024] 如何利用多智能体强化学习提升网页界面测试性能?

▶ 对于LLM4Testing未来研究的展望

- **LLM4Testing的研究正在如火如荼地开展，未来还有很多待探索的方面：**
  - 如何提升大模型输出的质量（在YanHui工作中，LLM生成模型优化程序的成功率仅为36% ）
  - 如何自动判定LLM输出的有问题的测试程序（一旦进入仓库，可能会造成污染和后续维护的噩梦）
  - 如何利用LLM解决复杂的测试oracle问题（如何自动判断程序输出是否正确）
  - 如何利用LLM检测一些难以直接编写测试用例检测的问题（比如某些界面渲染、图标误用错误）
  - 如何将LLM融合到持续集成/持续部署流程中（比如做变更影响分析，提升回归测试效率）
  - 如何利用LLM预测不同条件下软件系统的性能表现，提升性能测试效率
  - 如何利用LLM进行跨应用/平台/生态的测试用例迁移（捕获测试意图+重新生成）
  - 如何将LLM和其他方法（静态分析、形式化方法等）进行结合
  - … …

谢谢！
请多多批评指正！

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 敦煌站

K+ 思考周®研习社

时间：2025.08.29-30

K+峰会 上海站

K+ 金融专场

时间：2025.10.17-18

K+峰会 香港站

K+ 思考周®研习社

时间：2025.11.25-26

K+峰会详情

AiDD峰会

AiDD峰会 上海站

AI+研发数字峰会

时间：2025.05.17-18

AiDD峰会 北京站

AI+研发数字峰会

时间：2025.08.08-09

AiDD峰会 深圳站

AI+研发数字峰会

时间：2025.11.28-29

AiDD峰会详情

AiDD AI+研发数字峰会
AI+ Development Digital summit
第5届

利用AI技术深化计算机对现实世界的理解

推动研发进入智能化时代