AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·深圳 11.24-25

AI+
software
Development
Digital
summit

AI赋能的自动化软件调试
现状与未来

谢晓园 武汉大学

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 深圳站

K+ 全球软件研发行业创新峰会

会议时间：2024.05.24-25

K+峰会 上海站

K+ 全球软件研发行业创新峰会

会议时间：2024.09.20-21

AiDD 峰会

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2023.11.24-25

AiDD峰会 北京站

AI+ 软件研发数字峰会

会议时间：2024.07.19-20

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2024.11.15-16

演讲嘉宾

谢晓园

武汉大学计算机学院教授 博士生导师

武汉大学特色化示范性软件学院副院长

武汉大学珞珈青年学者、国家自科基金委外国优秀青年学者研究基金获得者。主要研究方向为软件测试、软件缺陷定位与修复、程序分析与切片、智能软件工程等。曾解决了软件缺陷定位领域30年来的瓶颈难题，被软件工程顶级期刊IEEE TSE评为全球软件蜕变测试领域十大代表性研究者之一。发表CCF A类论文13篇，其中一作或通讯11篇。以第一及通讯作者连续两年获国际软件工程顶级会议 (CCF A类) 杰出论文奖。获计算机学会NASAC青年软件创新奖、大陆首个ACM SigEvo HUMIES奖、湖北省科技进步一等奖等学术奖励。担任SCI期刊编委、客座编辑，历任ICSE/MET蜕变测试研讨会PC Chair。担任包括CCF A类会议ASE、ICSE在内的多个国际会议PC members，以及包括CCF A类期刊TSE、TOSEM和ACM Computing Survey等在内的多个国际知名期刊审稿人。

目录
CONTENTS

PART 01
背景

背景

现代社会，软件无处不在

背景

关键系统的软件缺陷将造成灾难性损失

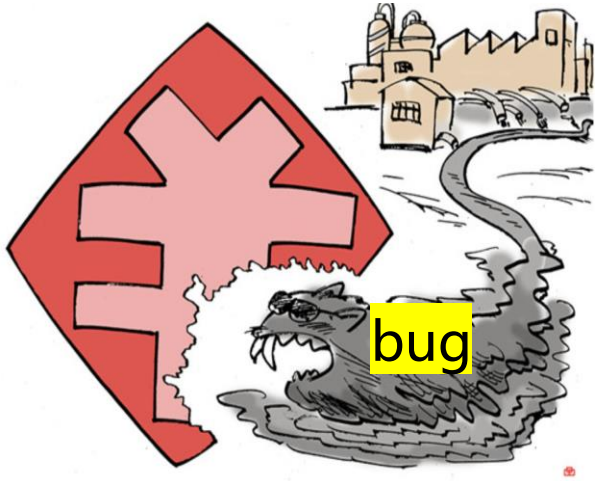
2018 - 2019

致命性航空事故

机载MCAS软件缺陷造成两起空难，共造成**346**人死亡

2022. 12

巨额的经济损失



权威机构发布报告称，2022年软件缺陷问题使美国经济损失**2.41**万亿美元

2023. 2

大规模车辆召回

因自动驾驶软件缺陷，特斯拉在全球范围召回近**36.3**万辆汽车

背景

高质量软件已成为经济社会发展的迫切需求

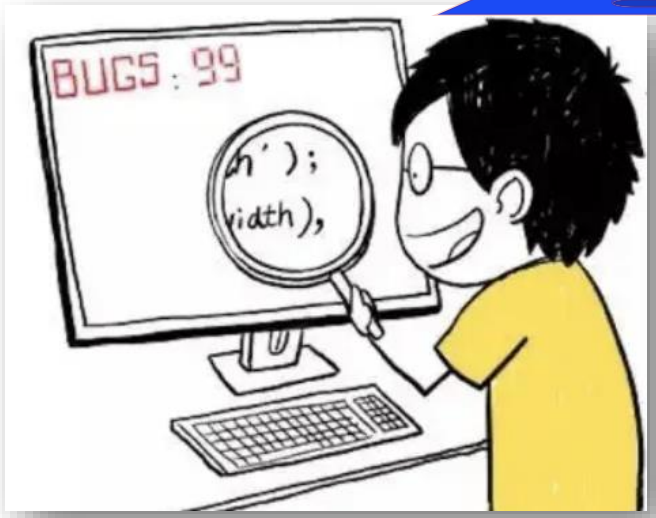
高效的**软件调试**

是构筑高质量软件的**重要一环**

背景

发现软件程序错误

复杂、关键



判断导致程序错误的底层缺陷位置或原因

(缺陷定位)

对定位到的缺陷进行修复

(程序自动修复)

软件测试
(Software Testing)

软件调试
(Software Debugging)

背景

缺陷定位 **程序自动修复**

面临的普遍技术需求

- 源数据提取及建模
- 数据清洗
- 特征集成与排序
- 代码片段理解
- 自然语言处理
- 结果人类可理解化

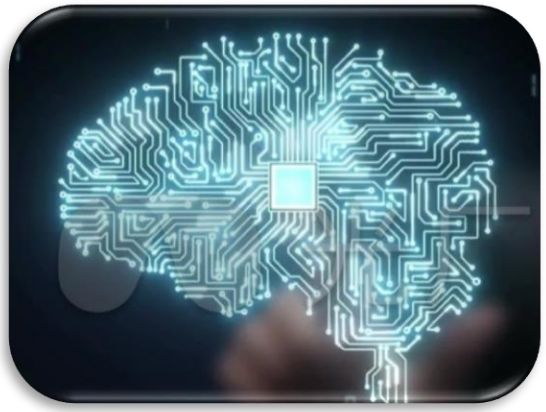
人类智能

并行能力差
脑容量有限
推理能力参差不齐
……

请求完成

难以有效完成调试任务

请求完成



人工智能

高度并行
扩容简单方便
推理能力强大
……

人工智能的优势与软件调试的需求高度契合

人工智能为软件调试任务的解决提供了**更高有效、更高可行**方案

背景

基于信息检索的缺陷定位IRFL

错误信息描述　堆栈轨迹

代码片段　其他缺陷报告信息

语言模型 / 分类器等

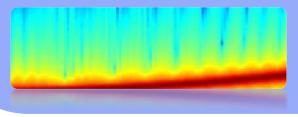
缺陷定位结果
(文件、函数粒度等)

基于程序频谱的缺陷定位SBFL

二进制覆盖信息　计数谱向量

测试用例属性　程序运行控制流

风险值评估公式



缺陷定位结果
(语句粒度等)

传统软件调试任务

迫切需要AI驱动

程序自动修复

缺陷代码上下文　需求文档

领域知识　程序规格说明

基于搜索的方法

候选程序补丁

原环境下重新测试

含错程序代码片段

神经机器翻译

正确代码片段

PART 02
AI模型驱动的软件调试

研究现状概览

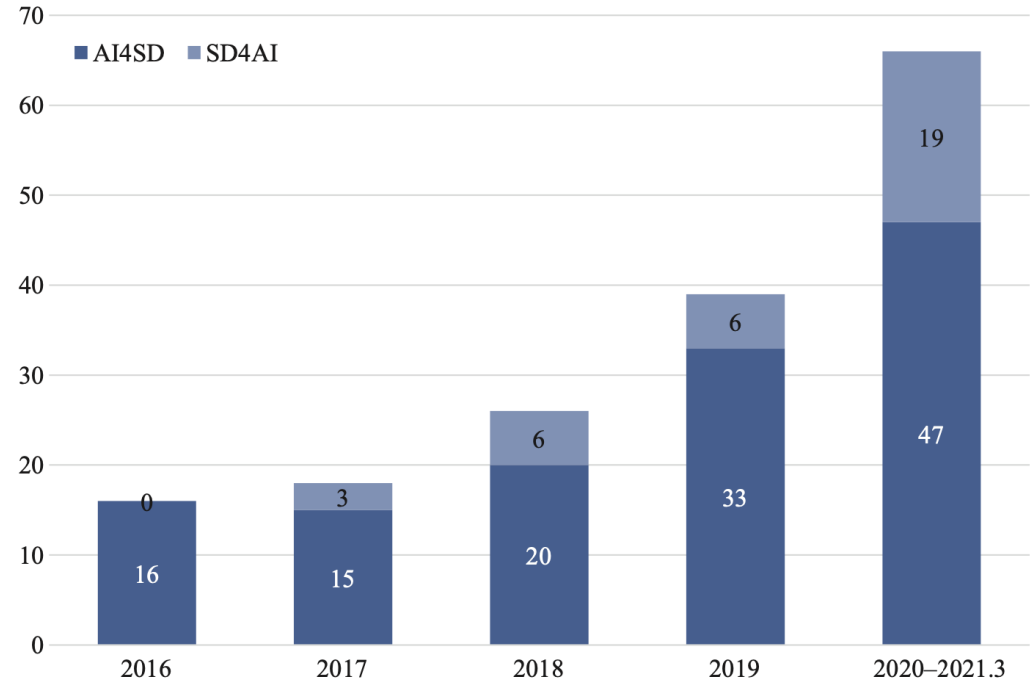
AI赋能的自动化软件调试热度呈不断上升趋势

该领域研究集中发表的国际期刊

简称	全名	发文量
EMSE	*Empirical Software Engineering*	12
TSE	*IEEE Transactions on Software Engineering*	10
JSS	*Journal of Systems and Software*	7
IST	*Information and Software Technology*	6
TOSEM	*ACM Transactions on Software Engineering and Methodology*	5
ACM PL	*Proceedings of the ACM on Programming Languages*	3
JSEP	*Journal of Software: Evolution and Process*	2

该领域研究集中发表的国际会议

简称	全名	发文量
ICSE	International Conference on Software Engineering	16
FSE/ESEC	ACM SIGSOFT Symposium on the Foundation of Software Engineering/ European Software Engineering Conference	10
ASE	International Conference on Automated Software Engineering	9
ISSTA	International Symposium on Software Testing and Analysis	5
IJCAI	International Joint Conference on Artificial Intelligence	5
QRS	International Conference on Software Quality, Reliability and Security	5
SANER	International Conference on Software Analysis, Evolution, and Reengineering	4
ICLR	International Conference on Learning Representations	3

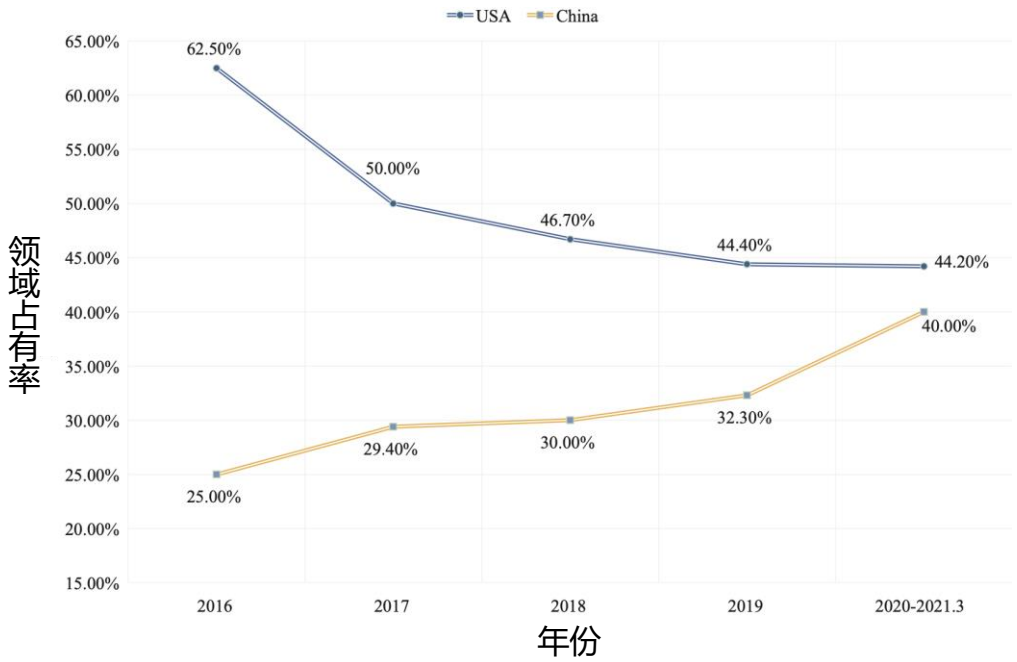


2016年~2021年第一季度AI与软件调试交叉领域研究热度

Yi Song, Xiaoyuan Xie and Baowen Xu. When Debugging Encounters Artificial Intelligence: State of the Art and Open Challenges. *SCIENCE CHINA Information Sciences*, 2023, ISSN 1674-733X, https://doi.org/10.1007/s11432-022-3803-9.

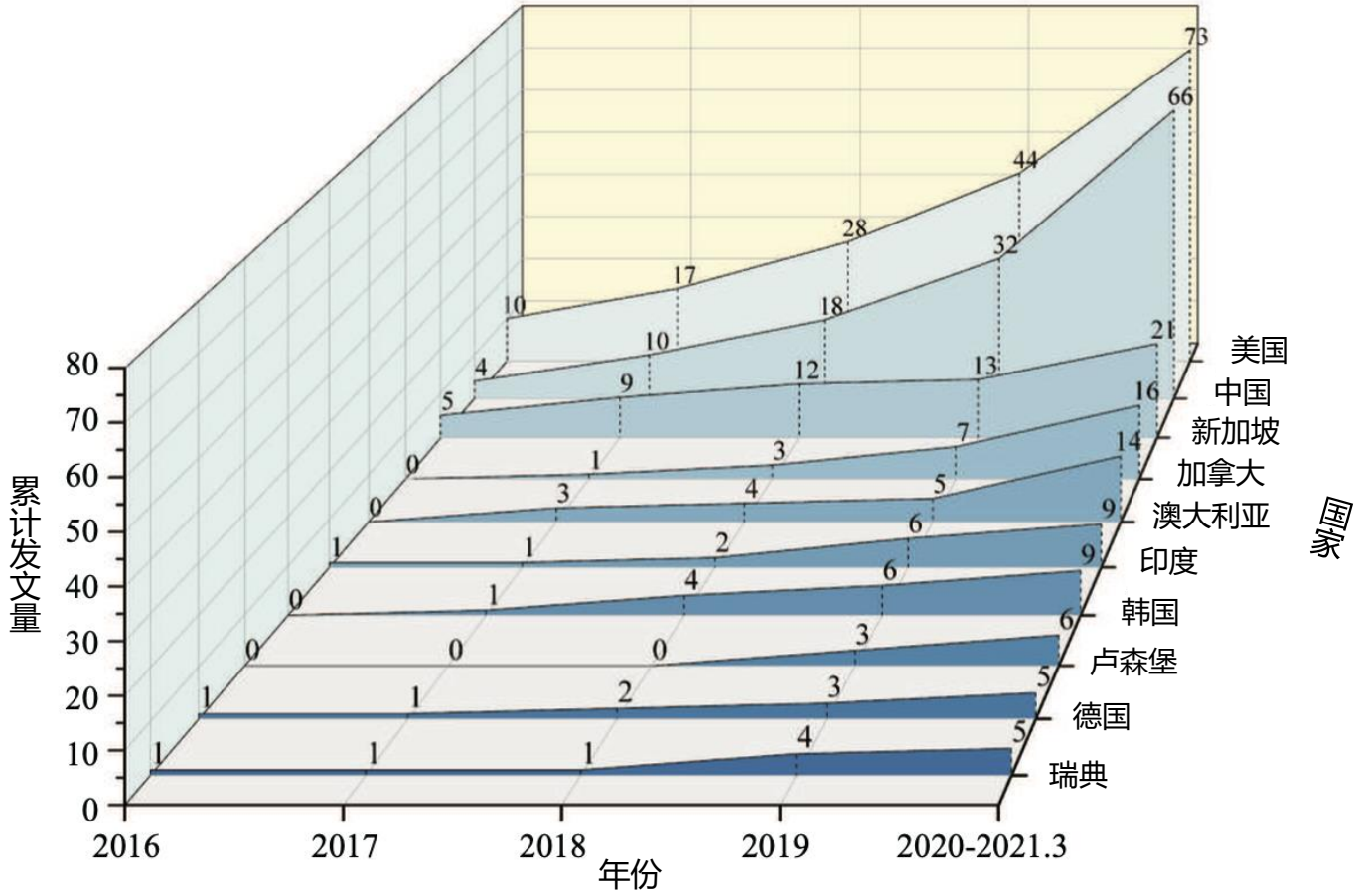
研究现状概览

美国和中国在AI赋能软件调试研究领域处于领军地位



美中两国领域占有率

$$\text{领域占有率} = \frac{\text{当年该国家在该领域发表的文章数}}{\text{当年该领域总文章数}}$$



AI赋能软件调试研究活跃度排名前十的国家

AI驱动的软件缺陷定位

AI驱动软件缺陷定位的核心任务

软件缺陷定位的核心任务

建立链接

表层执行

失效行为 (症状)

用户感知软件系统异常的直接窗口
软件系统错误运行结果的外在表达
软件缺陷的最直接表现形态
软件缺陷定位任务的起点

中间状态

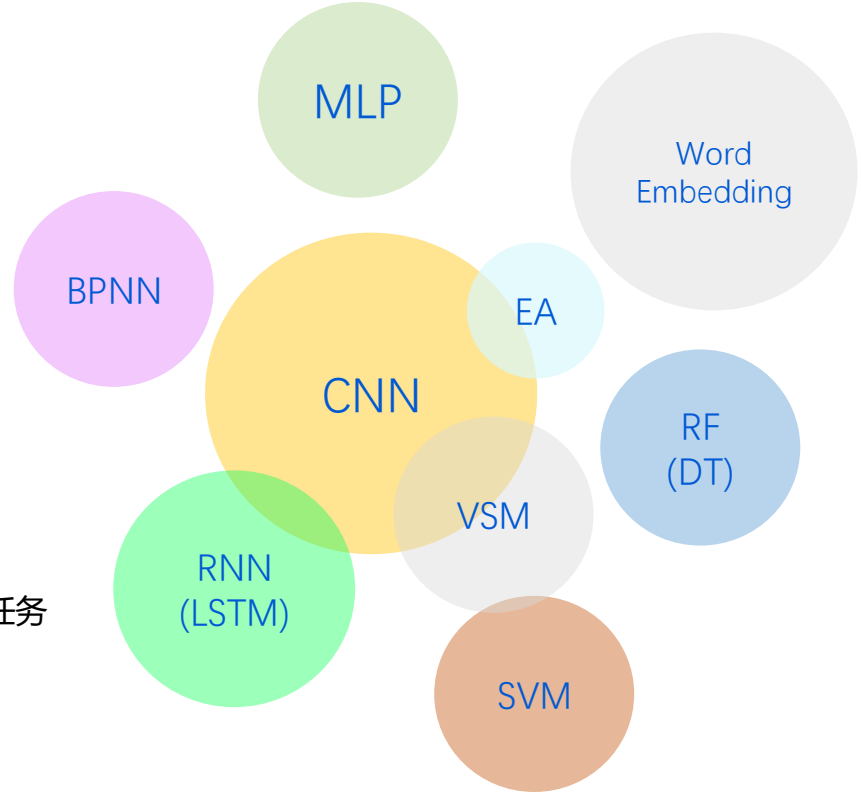
PIE模型证实的软件缺陷传导必经环节
潜藏信息种类多、体量大、组织杂乱
用户无法直接触达
缺陷定位任务的核心数据来源

底层代码

缺陷根因 (症结)

触发软件表层异常的根本因素
软件缺陷定位任务的最终目标
“病灶”小 隐藏于海量代码中

AI模型与缺陷定位任务耦合的接口



AI模型主要以表层执行面和中间状态面为切入点，对缺陷定位任务进行驱动

缺陷报告文本的处理	程序运行控制流、数据流的二次表征
开源平台Issue/Pull Request等版本迭代信息的挖掘……	智能风险值计算方法
	含错风险特征提取与集成……

AI驱动的软件缺陷定位

AI模型接收的缺陷定位源数据类型

- **基于执行的数据**

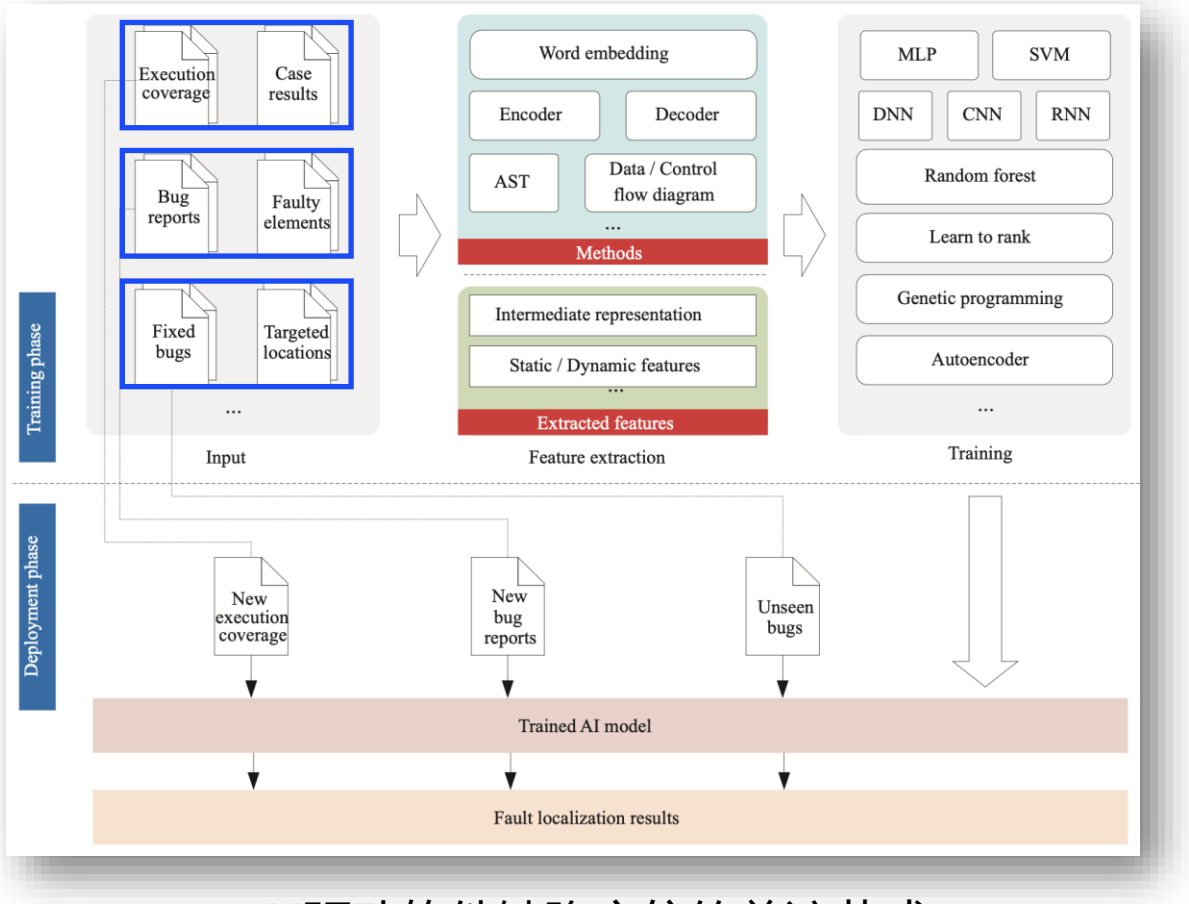
  软件动态测试过程中产生的各类信息，如执行路径、动静态切片、控制流、数据流、调用链、堆栈轨迹等。

- **基于文本的数据**

  缺陷报告中的自然语言部分 (如Issue概要、元信息、开发人员评论等)和代码中的文本部分 (有指示意义的变量名、代码注释等)。

- **基于代码的数据**

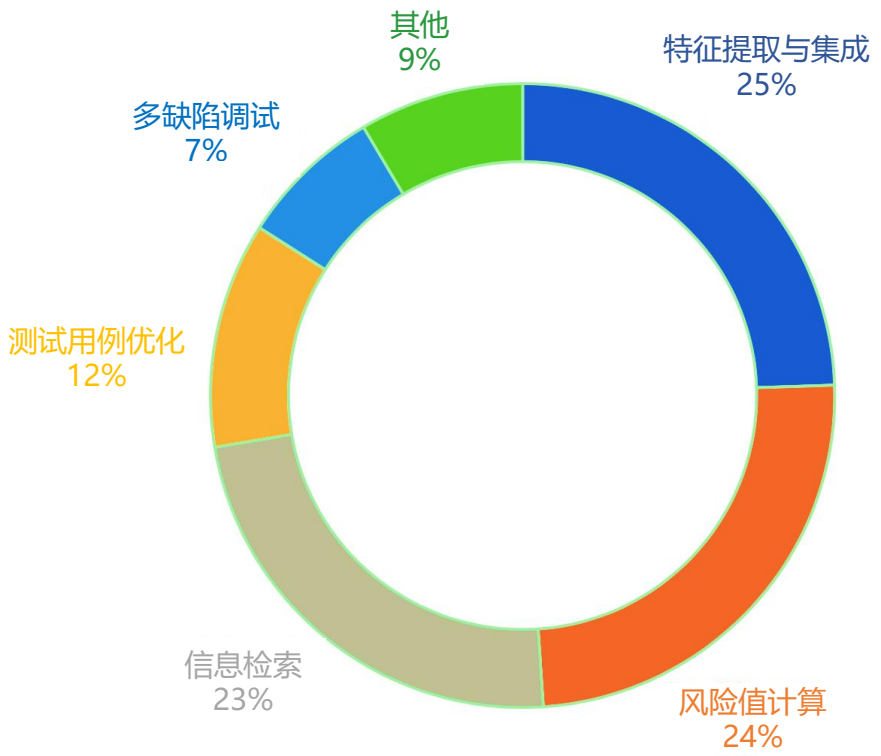
  软件系统中源代码文件里包含的缺陷相关静态特征。从粒度看，可以分为文件级、方法级、基本块级、语句级、变量级、token级等；从形式上看，可以分为依赖信息和调用图等。

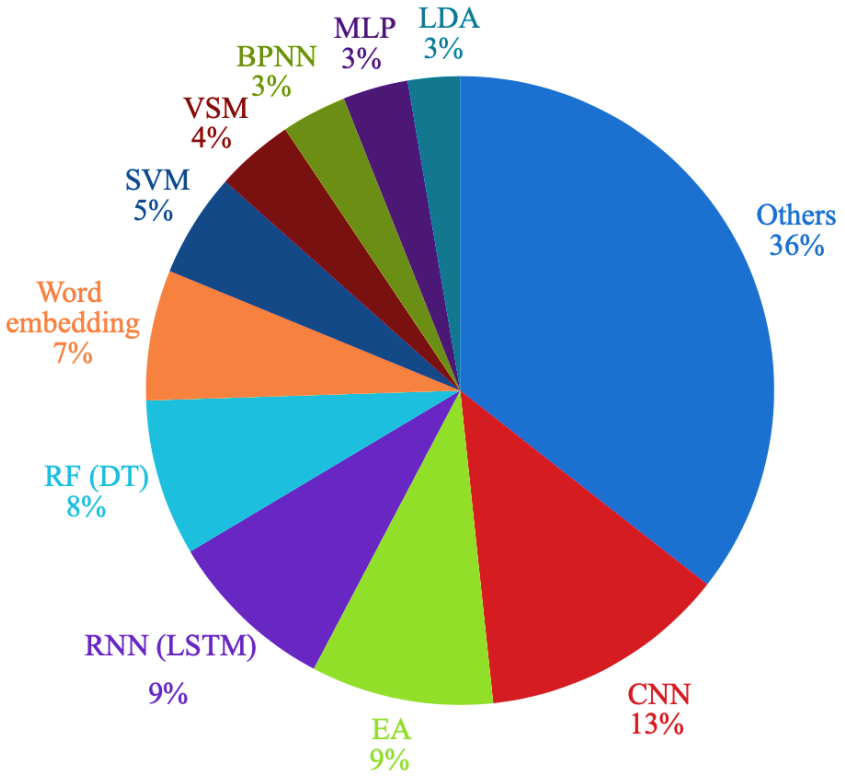


AI驱动软件缺陷定位的普遍范式

AI驱动的软件缺陷定位

AI驱动的缺陷定位子任务类型及常用的AI模型





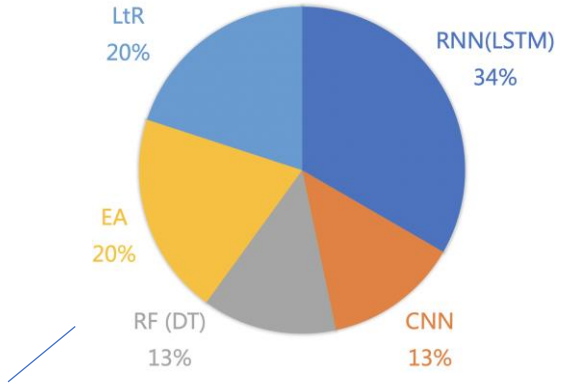
AI驱动的软件缺陷定位

AI驱动的缺陷定位子任务类型及常用的AI模型

- **特征提取与集成**

  描述 利用AI模型对数据的挖掘和集成能力，从不同类型的可用源数据中提取缺陷相关的特征，或将多种特征融合为一个综合的缺陷风险值。

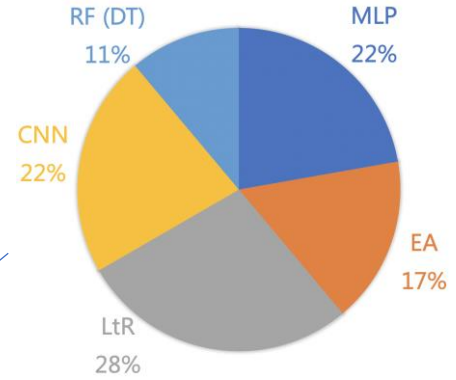
  常用模型



- **风险值计算**

  描述 采用AI技术对异常程序中的实体 (文件、函数、基本块、语句等)计算含有缺陷的风险值，以此为调试人员提供排查软件错误的先后参考。

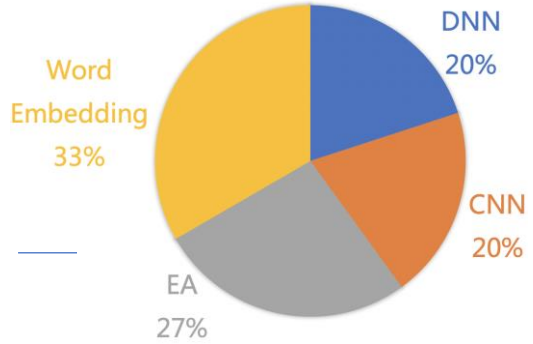
  常用模型



- **信息检索**

  描述 缺陷报告及其他自然语言形式的失效行为特征为追踪缺陷提供了线索，这些源数据人类友好但不利于机器理解。采用AI技术挖掘上述类型信息中潜藏的缺陷相关特征。

  常用模型



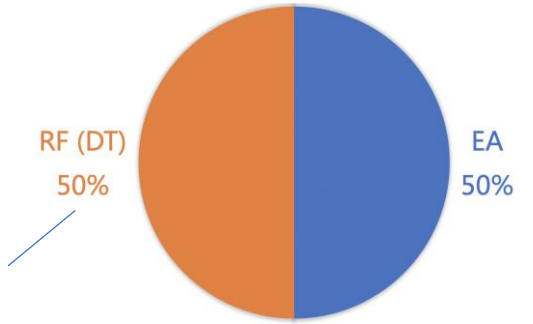
AI驱动的软件缺陷定位

AI驱动的缺陷定位子任务类型及常用的AI模型

- **测试用例优化**

  描述 测试用例的质量在一定程度上决定了缺陷定位任务的有效性和效率。采用AI技术对测试用例进行排序、选择等，使其具有更强的缺陷检测和定位能力。

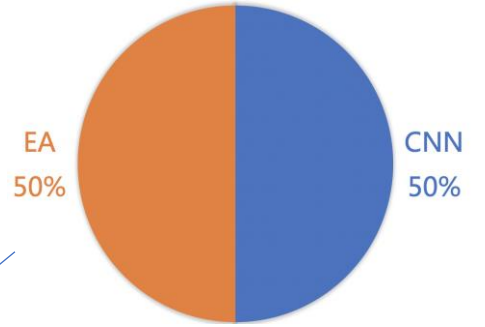
  常用模型



- **多缺陷调试**

  描述 多缺陷环境是软件调试实践通常面临的情况。采用AI技术对表现为失败测试用例等形态的失效行为进行索引，进而对多缺陷进行隔离，使多缺陷可以相对独立且并行地调试。

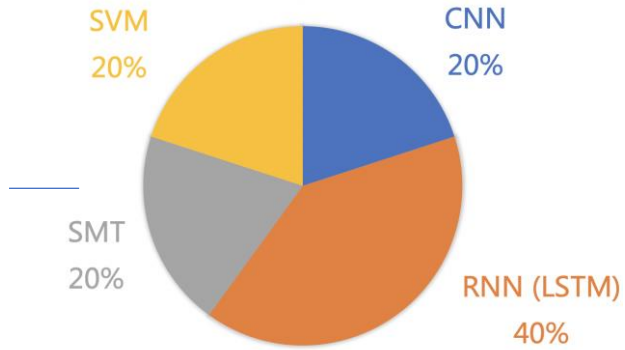
  常用模型



- **其 他**

  描述 如，利用AI技术：推荐Log语句的插入位置、记录信息种类；缓解静态分析工具的高误报率问题；对并行软件或领域特定软件开展缺陷定位，等等。

  常用模型



AI驱动的程序自动修复

AI模型与程序自动修复任务耦合的接口

上游缺陷定位模块

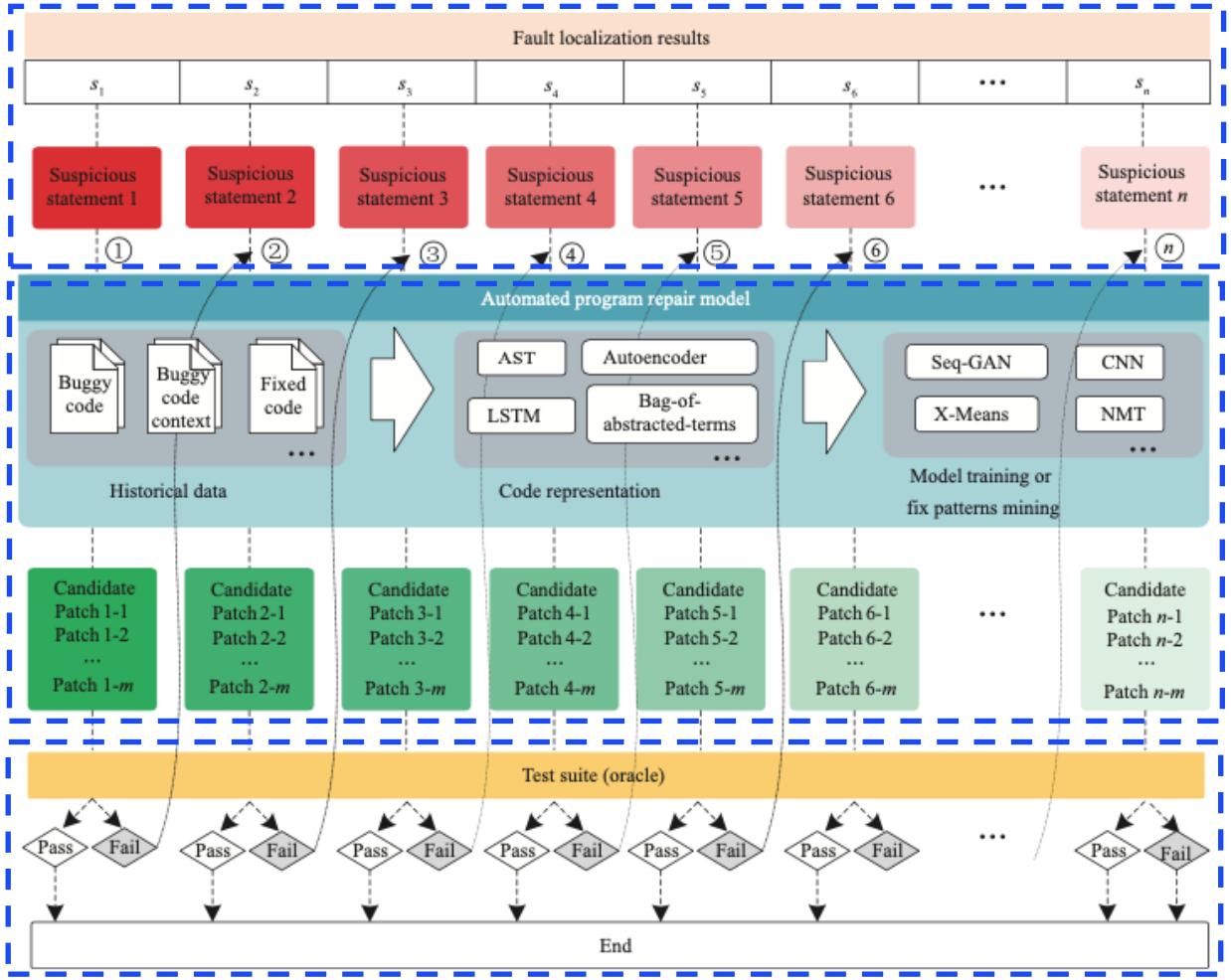
基于缺陷定位模块提供的结果依次确定修复对象。程序自动修复的有效性高度依赖于上游缺陷定位任务的输出。

补丁生成模块

为潜在的错误语句生成修复补丁。可能的技术选项包括约束求解、基于搜索的方法、历史缺陷/修复代码对训练 (代码表征)、基于聚类的修复模式挖掘等。

补丁验证模块

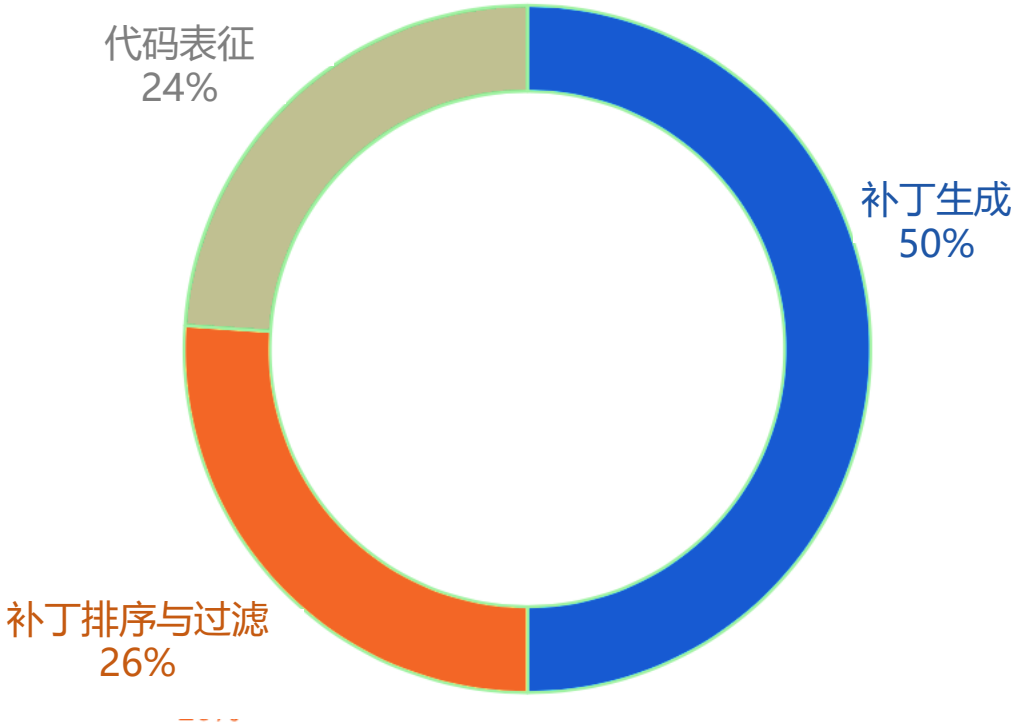
对生成补丁的正确性进行验证，将真正正确的补丁与“看似正确”的补丁区分开是该阶段的重点、难点任务。

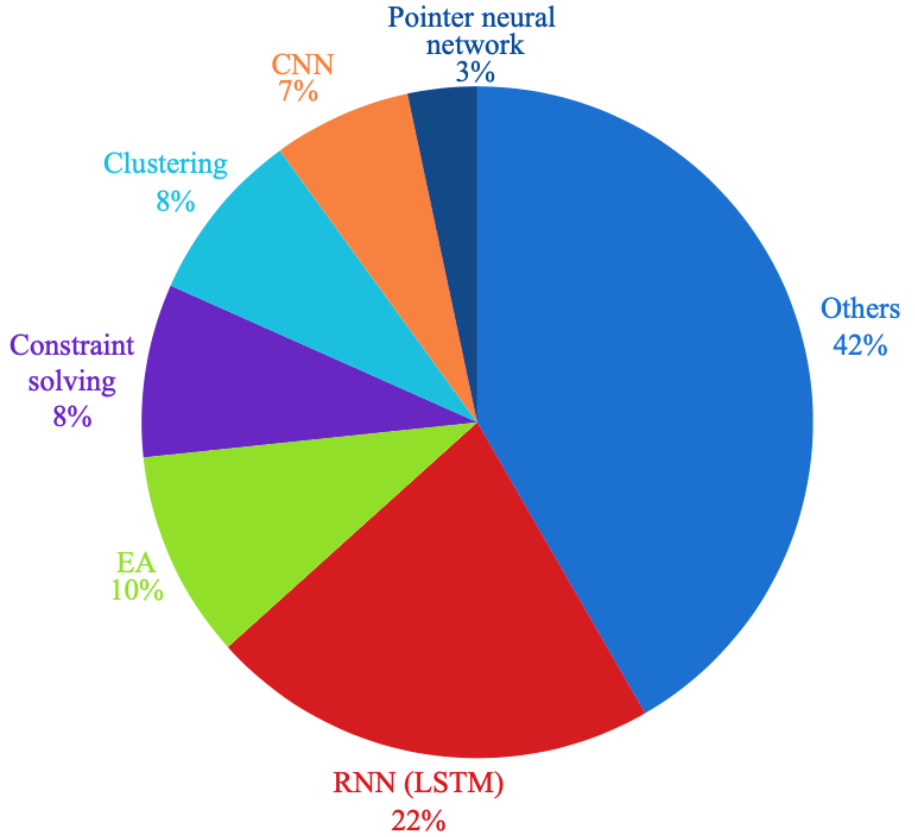


AI驱动程序自动修复的普遍范式

AI驱动的程序自动修复

AI驱动的程序自动修复子任务类型及常用的AI模型





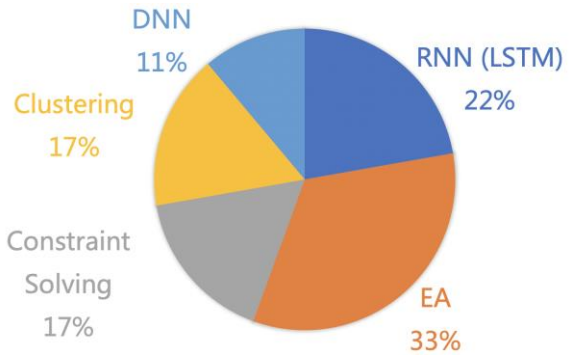
AI驱动的程序自动修复

AI驱动的程序自动修复子任务类型及常用的AI模型

- **补丁生成**

描述 对错误的代码片段进行修改，以生成符合程序预期功能的新代码。常用的策略包括遗传编程、约束求解、神经机器翻译等。

常用模型



- **补丁排序与过滤**

描述 生成的修复补丁并不总是正确的。因此，借助AI技术的预测能力对补丁根据其“真正正确而非看似正确”的可能性大小进行排序和过滤，以降低人力和时间开销。

常用模型 Constraint Solving

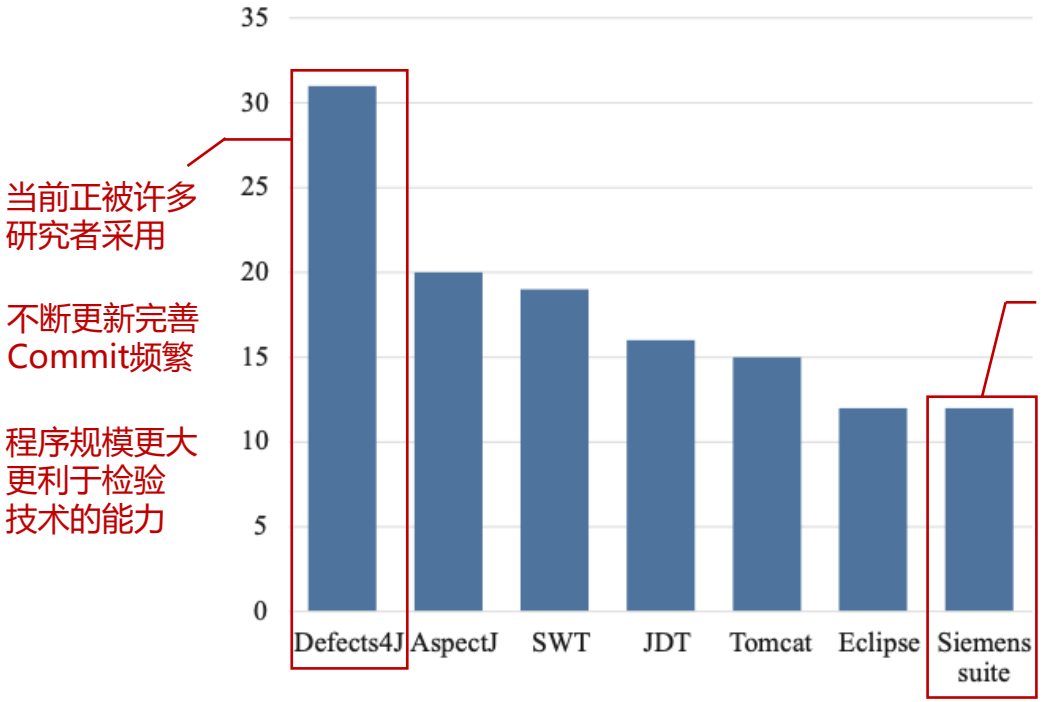
- **代码表征**

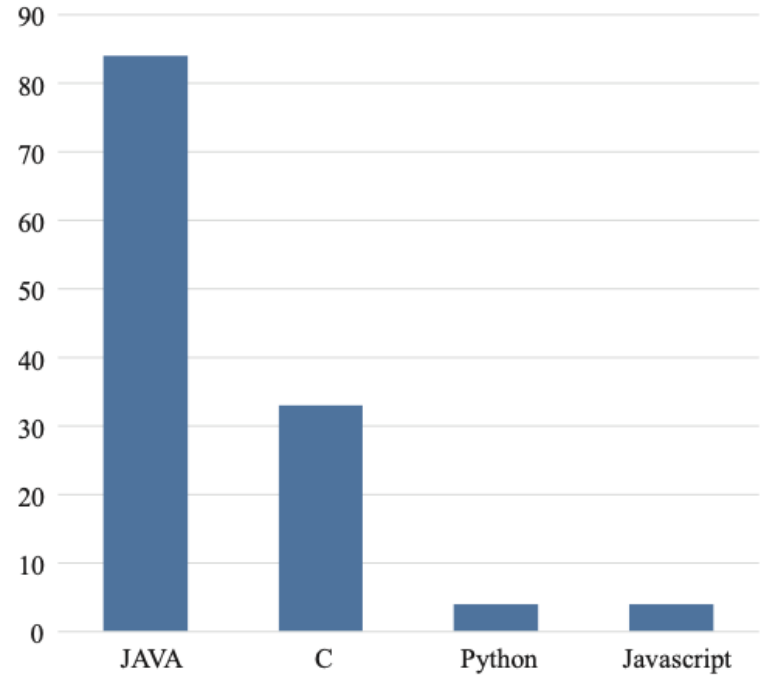
描述 程序自动修复任务中，代码片段常常需要被转换成中间态表示，以契合AI模型指定的输入需求，同时以良好的二次表征暴露出源信息中蕴藏的缺陷相关特征。

常用模型 RNN (LSTM)

AI模型驱动的软件调试

该领域研究实验最常使用的数据集/基准程序语言





Defects4J数据集、Java语言编写的程序，常被用于检测AI技术应用于软件调试任务的有效性

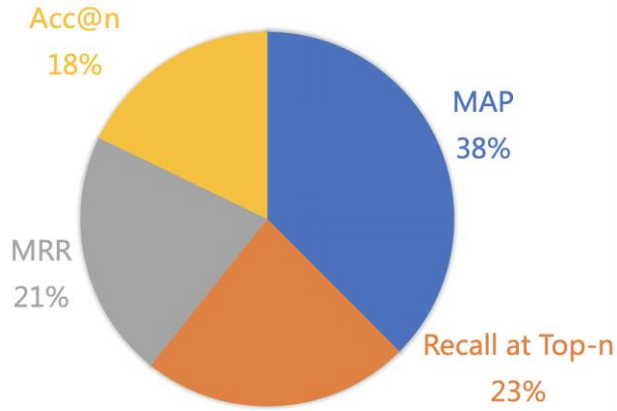
数据集中存在的正负样本类不平衡、偶然性正确、测试谕言缺失等问题值得引起关注

AI模型驱动的软件调试

该领域研究实验最常使用的度量指标

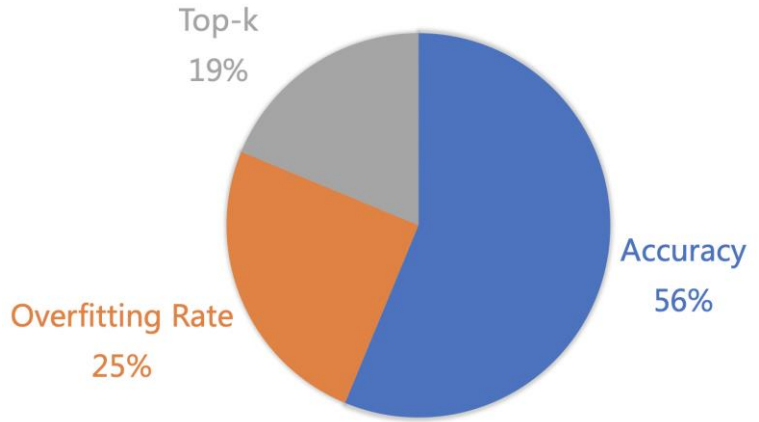
AI驱动软件缺陷定位领域实验常用指标

名 称	描 述
Mean average precision (MAP)	The mean of the average precision of all faults [77, 136, 158, 165, 185]
Recall at Top-n	The number of faulty programs that at least one buggy element appears in the top-n position of the ranking list [157, 158, 239]
Mean reciprocal rank (MRR)	The reciprocal of the position of the first buggy element in the ranking list [77, 158, 185]
Acc@n	The number of faults localized within top-n elements of the ranking list [140, 161, 165]



AI驱动程序自动修复领域实验常用指标

名 称	描 述
Accuracy	The percentage of patches that correctly fix faults [234]
Overfitting rate	The number of overfitting patches out of the number of total patches generated [257]
Top-k	The number of correct patches that can be found by considering top-k statements delivered by the employed FL technique [239]



AI模型驱动的软件调试

现有工具汇总 – AI 4 FL

Table A1 AI4FL tools (material)

Paper	Description	Website
[229]	Symbiosis: A concurrency debugging technique based on differential schedule projections	https://github.com/nunomachado/symbiosis
[160]	FLUCCS: A fault localization technique that learns to rank program elements based on both existing SBFL techniques and source code metrics	https://bitbucket.org/teamcoinse/fluccs
[145]	An approach to identifying high-impact bug reports	https://github.com/goddding/JCST
[189]	PkgRec: A tool to predicting the confidence score of a package being affected by a new bug report	http://github.com/tkdsheep/MultiPackage
[174]	DeepBugs: A learning approach to name-based bug detection	https://github.com/michaelpradel/DeepBugs
[206]	NeuralBugLocator: A deep learning-based technique that can localize bugs without running the program	https://bitbucket.org/iiscseal/nbl/
[218]	Tregression: An Eclipse plugin to visualizing the bugs in Defects4J repository	https://github.com/llmhyy/tregression
[228]	FixML: A system for automatically generating feedback on logical errors in functional programming assignments	https://github.com/kupl/FixML
[147]	ChangeLocator: A method to automatically locating crash-inducing changes for a given bucket of crash reports	https://bitbucket.org/rongxin/changelocator-dataset
[69]	RepLoc: An automated framework to localizing the problematic files for unreproducible builds	https://reploc.bitbucket.io/
[171]	SCMiner: An online bug diagnosis tool	https://github.com/Tarannum-Zaman/Scminer
[134]	DeepFL: A deep learning approach to automatically learning the most effective existing/latent features for fault localization	https://github.com/DeepFL/DeepFaultLocalization
[176]	A tool to constructing datasets for evaluating AI-based fault localization techniques	https://github.com/yanxiao6/BugLocalization-dataset
[139]	CraTer: An automatic approach for predicting whether a crashing fault resides in stack traces	http://cstar.whu.edu.cn/p/crater/
[151]	D&C: A divide-and-conquer approach for IR-based fault localization	https://github.com/d-and-c/d-and-c
[109]	MLDebugger: A method that iteratively finds minimal definitive root causes	https://github.com/raonilourenco/MLDebugger
[163]	CombineFL: An infrastructure for evaluating and combining fault localization techniques	https://damingz.github.io/combinefl/index.html
[64]	HSFL: A historical spectrum-based fault localization technique	https://github.com/justinwm/HSFL
[146]	ChangeRanker: A fault localization tool with feature selection technique	https://github.com/Naplues/ChangeRanker
[194]	Ulysis: A metric to capturing the quality of test suites for diagnosability	https://github.com/EvoSuite/evosuite/pull/293
[183]	OffSide: A technique to identifying mistakes in boundary conditions	https://github.com/SERG-Delft/ml4se-offside
[143]	ALBFL: A neural ranking model that combines static and dynamic features for fault localization	https://github.com/indigo-99/ALBFL
[197]	A test case resampling tool for boosting the effectiveness of deep learning-based fault localization techniques	https://github.com/zhuozhangNUDT/test-case-resampling
[204]	RecBi: A reinforcement compiler bug isolation tool	https://github.com/haoyang9804/RecBi
[138]	JDCallgraph: A tool to generating dynamic call graphs for fault localization	https://github.com/dkarv/jdcallgraph
[170]	A two-phase framework for just-in-time defect identification and localization	https://github.com/MengYan1989/JIT-DIL
[168]	Grace: A fault localization tool using graph-based representation learning	https://github.com/yilinglou/Grace
[8]	ProbDD: A probabilistic delta debugging algorithm	https://github.com/Amocy-Wang/ProbDD
[101]	DeepFD: A learning-based fault diagnosis and localization framework	https://github.com/ArabelaTso/DeepFD

AI模型驱动的软件调试

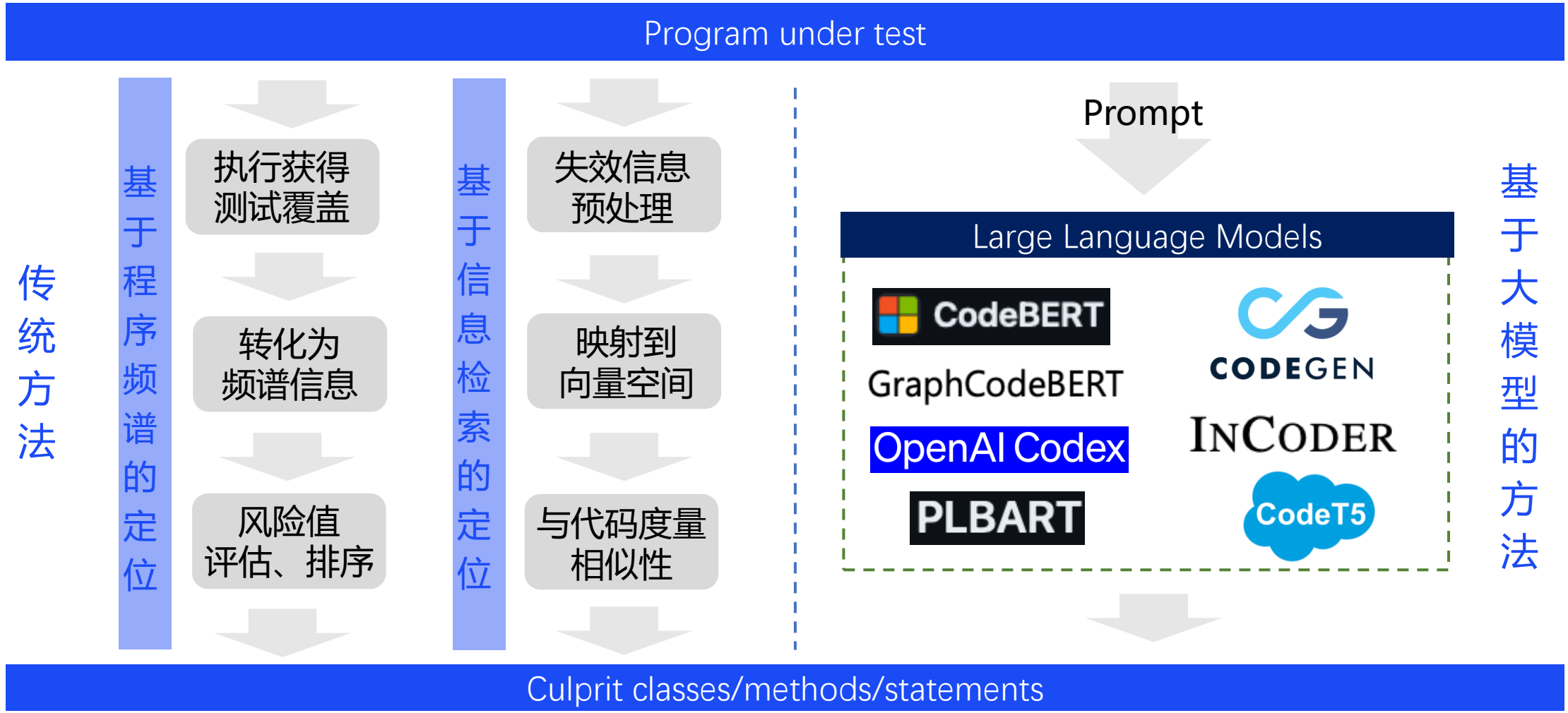
现有工具汇总 – AI 4 APR

Table A2 AI4APR tools (material)

Paper	Description	Website
[128]	A technique for history-based program repair	https://github.com/xuanbachle/bugfixes
[255]	A set of anti-patterns that can be used on search-based repair tools	https://anti-patterns.github.io/search-based-repair/
[127]	Nopol: An approach to automatic repair of buggy conditional statements	http://github.com/SpoonLabs/nopol/
[129]	ACS: A program repair system to generating conditions at faulty locations	https://github.com/Adobee/ACS
[252]	An approach that heuristically determines the correctness of the generated patches	https://github.com/Ultimanecat/DefectRepairing
[225]	ARJA: A new GP based repair approach for automated repair of Java programs	https://github.com/yyxhdy/arja
[259]	A semantic program embedding for program repair	https://github.com/keowang/dynamic-program-embedding
[232]	Clara: An automated program repair algorithm for introductory programming assignments	https://github.com/iradicek/clara
[237]	SimFix: An automatic program repair approach that utilizes both existing patches and similar code	https://github.com/xgdsmileboy/SimFix
[245]	CapGen: A context-aware patch generation approach	https://github.com/justinwm/CapGen
[239]	iFixR: A debugging pipeline incorporating both patch generation and patch validation	https://github.com/SerVal-DTF/iFixR
[99]	SEQUENCER: An end-to-end approach to program repair based on sequence-to-sequence learning	https://github.com/kth/SequenceR
[98]	A tool to automatically learning bug-fixes in the wild for patch generation	https://sites.google.com/view/learning-fixes
[235]	DeepRepair: An approach for sorting and transforming program repair ingredients	https://sites.google.com/view/deeprepair
[260]	RLAssist: A deep reinforcement learning-based program repair technique	https://bitbucket.org/iiscseal/rlassist
[258]	DrRepair: A self-supervised learning-based program repair technique	https://github.com/michiyasunaga/DrRepair
[100]	CoCoNuT: An ensemble learning-based patch generation and validation tool	https://github.com/lin-tan/CoCoNut-Artifact
[244]	DLFix: A two-tier APR tool based on code transformation learning	https://github.com/ICSE-2019-AUTOFIX/ICSE-2019-AUTOFIX
[224]	SCRepair: A smart contract repair tool	https://screpair-apr.github.io/
[226]	ARJA-e: An evolutionary repair tool for Java code	https://github.com/yyxhdy/arja/tree/arja-e
[97]	A tool to mining fix patterns for FindBugs violations	https://github.com/FixPattern/findbugs-violations
[238]	CURE: An NMT-based APR tool	https://github.com/lin-tan/CURE
[240]	Recoder: A syntax-guided APR tool	https://github.com/pkuzqh/Recoder
[263]	CPR: An APR tool for network control planes	https://bitbucket.org/uw-madison-networking-research/arc

Yi Song, Xiaoyuan Xie and Baowen Xu. When Debugging Encounters Artificial Intelligence: State of the Art and Open Challenges. *SCIENCE CHINA Information Sciences*, 2023, ISSN 1674-733X, https://doi.org/10.1007/s11432-022-3803-9.

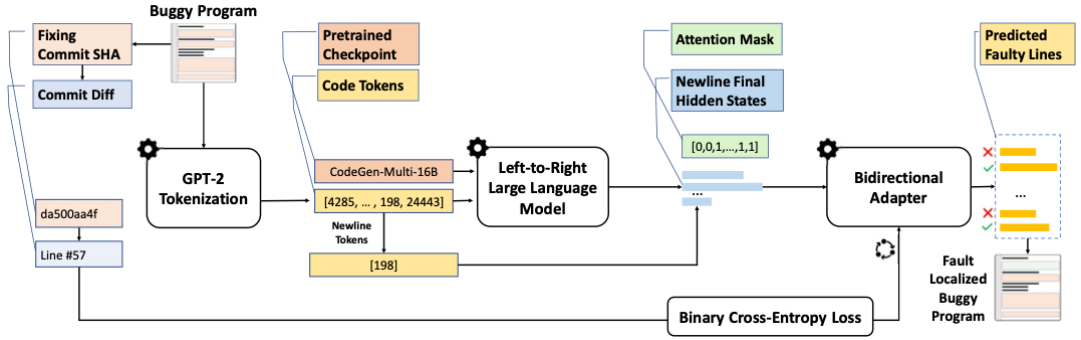
基于大模型的软件缺陷定位



基于大模型的软件缺陷定位

基于大模型的缺陷定位研究

LLM for FL研究热度迅速升温
据不完全统计，目前相关工作已发表/录用2篇
ArXiv等预印本平台上超过5篇



Model	BLEU-4		ROUGE-L		METEOR		BERTScore		BLEURT		NUBIA	
	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5
RoBERTa	4.44	NA	7.10	NA	4.52	NA	86.33	NA	26.80	NA	14.90	NA
CodeBERT	6.02	NA	4.40	NA	3.37	NA	86.83	NA	28.44	NA	27.89	NA
Curie	5.47	10.62	8.03	16.31	6.22	12.75	85.65	87.13	27.20	37.23	15.30	25.46
Codex	5.53	10.62	8.15	16.23	6.19	13.15	85.68	87.35	28.43	37.92	15.77	26.33
Davinci	5.54	10.66	8.10	15.96	6.08	12.49	85.72	87.19	27.15	37.00	15.71	25.61
Davinci-002	**6.76**	**11.66**	**10.22**	**18.14**	**8.23**	**15.13**	**86.17**	**87.65**	**30.19**	**38.96**	**17.58**	**28.81**
%gain for Davinci-002	22.02	9.38	25.40	11.22	32.32	15.06	0.52	0.34	6.19	2.74	11.48	9.42

LLMAO (ICSE 24)

提出一种微调大模型的缺陷定位技术

训练了轻量级的bidirectional adapters，其基于传统大模型的结果进行微调，在无需测试用例的情况下实现了更高效的缺陷定位。

Yang A Z H, Martins R, Goues C L, et al. Large Language Models for Test-Free Fault Localization. 2024 46th International Conference on Software Engineering (ICSE 2024)

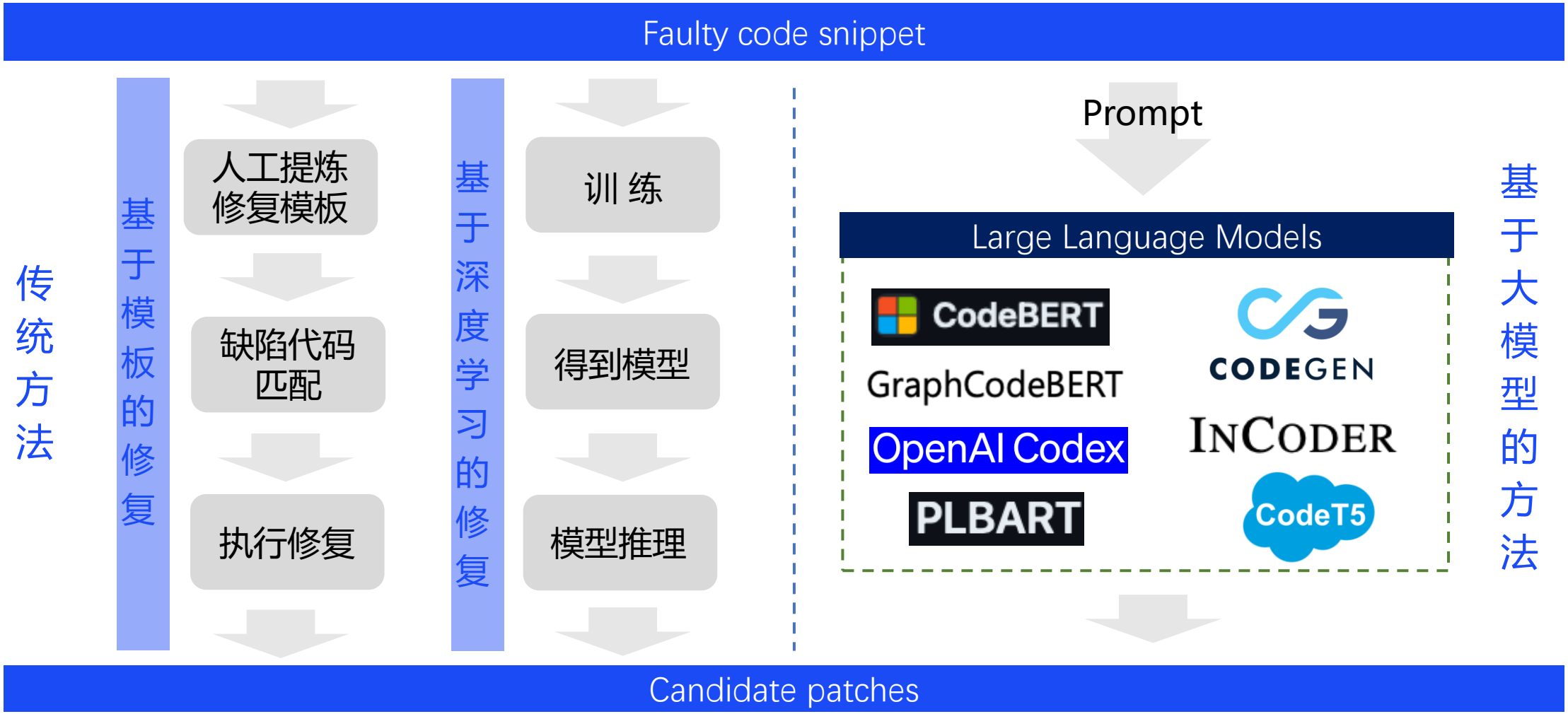
Empirical Study on LLM4FL (ICSE 23)

研究大模型在处理Cloud Incidents (一种软件异常)时的表现

对Fine-tuned GPT-3.x模型与Encoder-decoder模型定位缺陷根因的能力进行了比较。结果表明，大模型在缺陷定位方面表现出色。人类程序员更指出：大模型发现了难以探测到的缺陷。

Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, et al. Recommending Root-Cause and Mitigation Steps for Cloud Incidents Using Large Language Models. 2023 45th International Conference on Software Engineering (ICSE 2023)

基于大模型的程序自动修复



基于大模型的程序自动修复

基于大模型的程序修复研究

Tools / Models	Single func. (255 bugs)	Patch func.	Correct hunk	Single line
AlphaRepair	67			
RewardRepair	48			
Recoder	61			
TBar	54			
CURE	52			
GPT-Neo 125M	9	6	-	5
GPT-Neo 1.3B	18	7	-	12
GPT-Neo 2.7B	20	10	-	13
GPT-J	28	14	-	16
GPT-NeoX	34	18	-	21
CodeT5	6	-	6	-
INCODER 1.3B	32	-	32	-
INCODER 6.7B	37	-	37	-
Codex	99	63	62	32
Total	109	69	74	40

Empirical Study on LLM4APR (ICSE 23)

研究大模型在程序自动修复任务中的有效性

LLM for APR研究关注度高企

据不完全统计，目前相关工作已发表/录用近10篇

实验策略

9种State-of-the-art大语言模型

Generative models *125M ~ 20B* Infilling models

① generate the entire patch function

② fill in a chunk of code given the prefix and suffix

③ output a single line fix

LLM的3种修复模式

5套数据集（3类编程语言）

Xia C S, Wei Y, Zhang L. Automated program repair in the era of large pre-trained language models. 2023 45th International Conference on Software Engineering (ICSE 2023)

PART 03
AI赋能调试的未来

AI时代下的软件缺陷定义

AI时代下软件缺陷的定义发生了新变革

传统软件

缺陷有迹可循

研究历史较久

存量技术丰富

符合经典软件工程思维

逻辑由代码定义

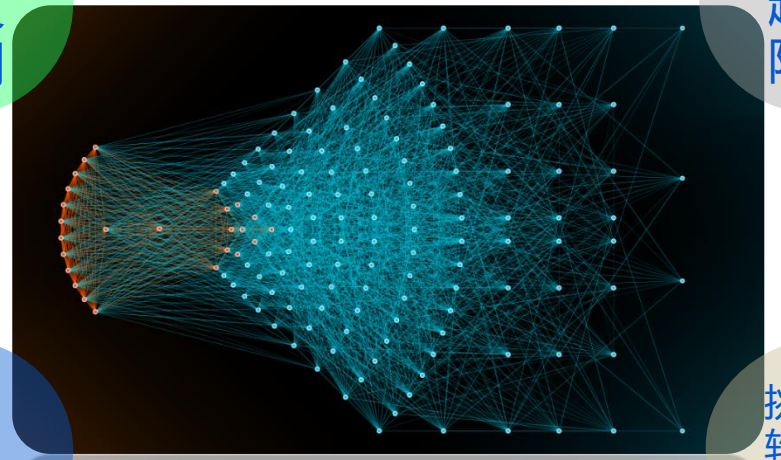
具有明确的控制流

缺陷根因承载于代码实体

AI软件

缺陷定义模糊

尚处起步阶段



成形技术乏善可陈

挑战经典软件工程思维

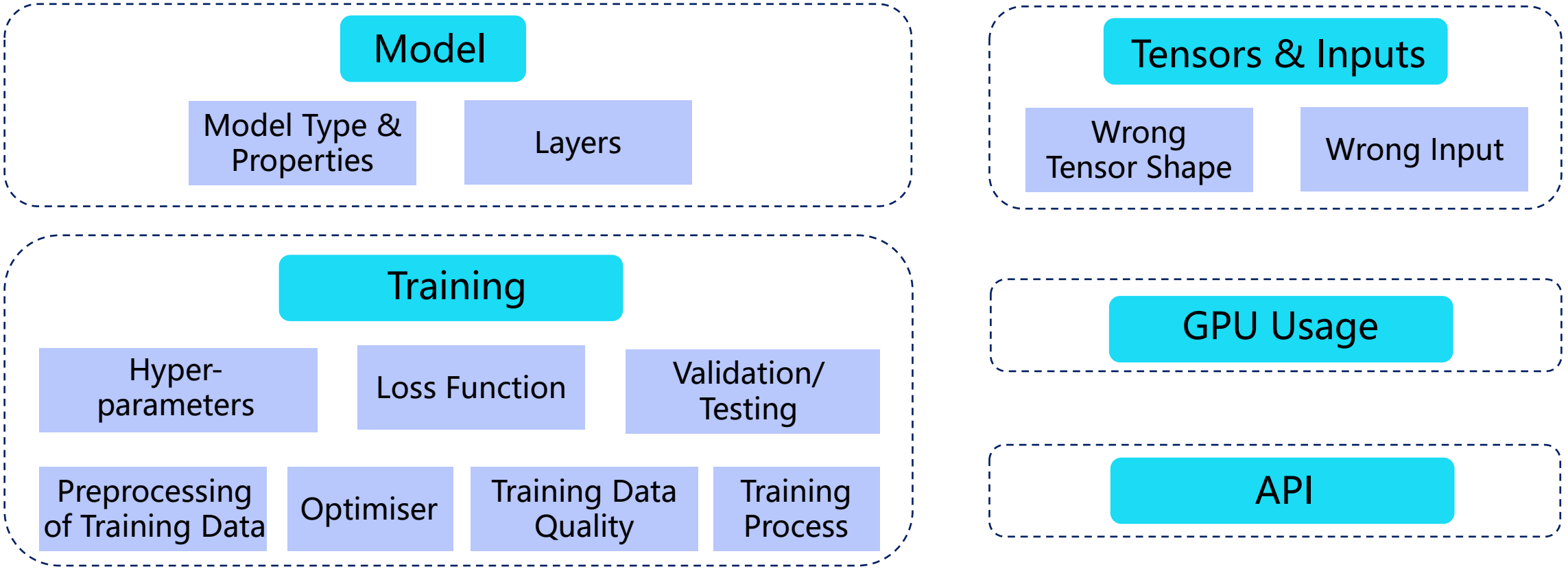
逻辑由数据驱动

具有黑盒属性 控制流难以追溯

缺陷根因常常不依托于程序本身

AI时代下的软件缺陷定义

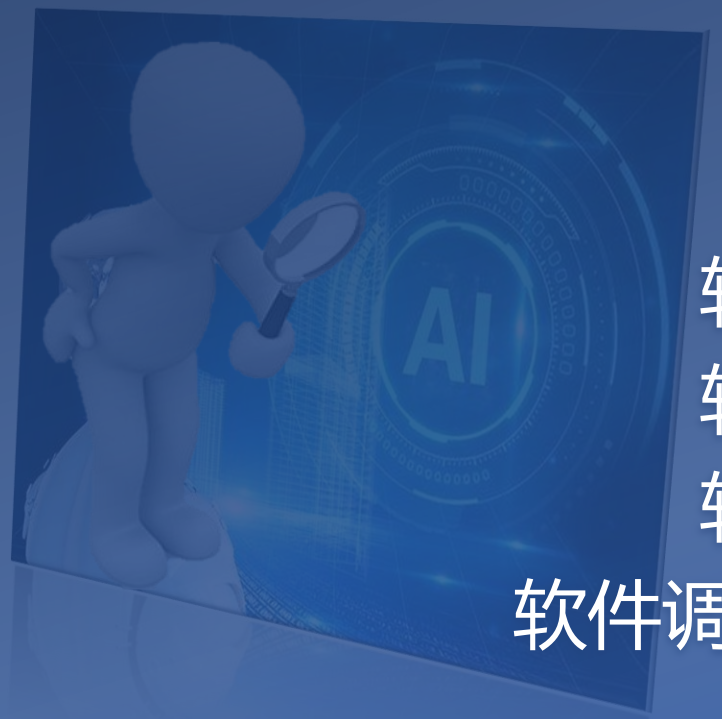
AI系统中常见的缺陷类型



Humbatova N, Jahangirova G, Bavota G, et al. Taxonomy of real faults in deep learning systems[C]//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 2020: 1110-1121.

AI时代下的软件缺陷定义

AI系统中常见的缺陷类型



数据准备阶段

数据缺陷

- 错误的输入数据格式或维度
- 不当的训练数据标签
- 失效的文件路径

AI时代下：

软件缺陷的定义大大**丰富**

软件缺陷的范围不断**蔓延**

软件缺陷的危害持续**增大**

软件调试任务的难度**“指数级”攀升**

AI系统缺陷的复杂性使得对其的质量保障十分困难

AI系统的质量被证实与经典Accuracy指标无关

AI时代下新型缺陷的调试

AI时代催生的新型缺陷调试策略

- **面向训练数据的调试**

  依托AI数据驱动属性，建立训练数据与错误预测之间的链接；对造成假阴性/假阳性预测结果的部分训练数据予以删除或调整；随后重复训练过程。

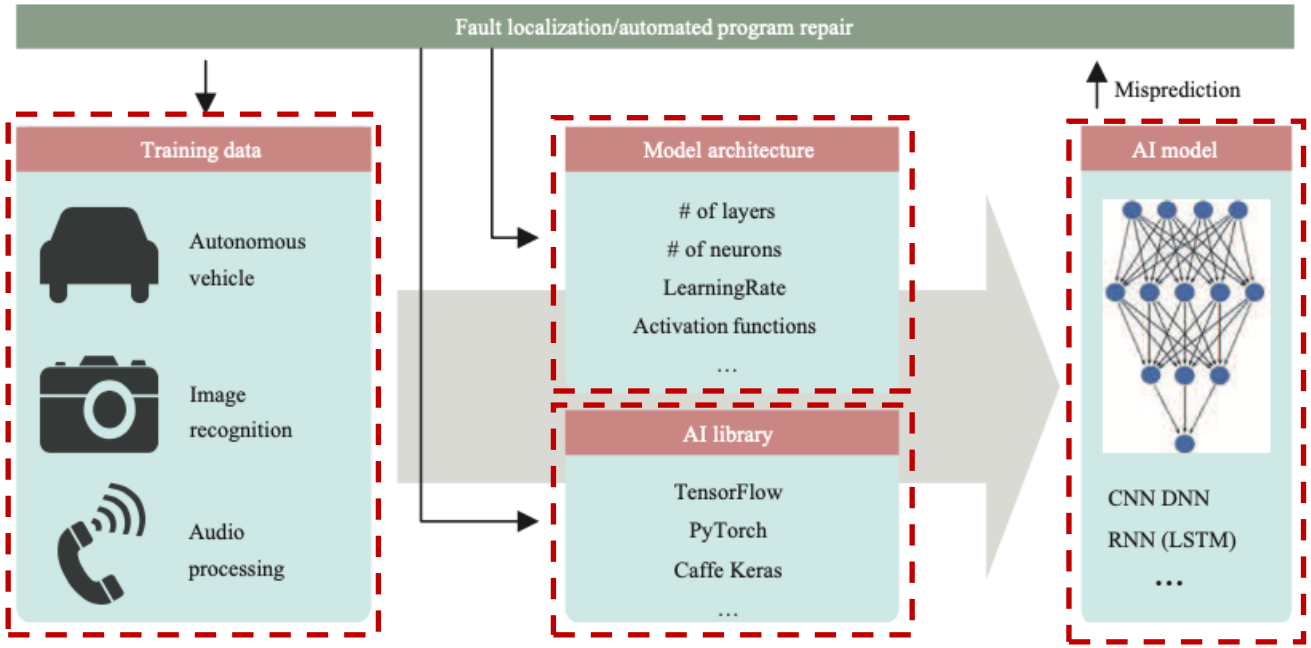
- **面向模型的调试**

  对AI模型的结构及相关参数 (如层数、神经元、学习率、激活函数)等进行调试，例如采用执行覆盖策略确定可疑的神经元。

- **面向库的调试**

  总结主流AI库 (如TensorFlow、PyTorch等)中的常见错误类型，以此建立基于规则的缺陷静态检查机制。

从一定程度上为缓解AI技术“黑盒”属性造成的**低可理解性**问题提供了方案，帮助人类程序员更好地理解AI模型输出，从而促进对结果的采纳。

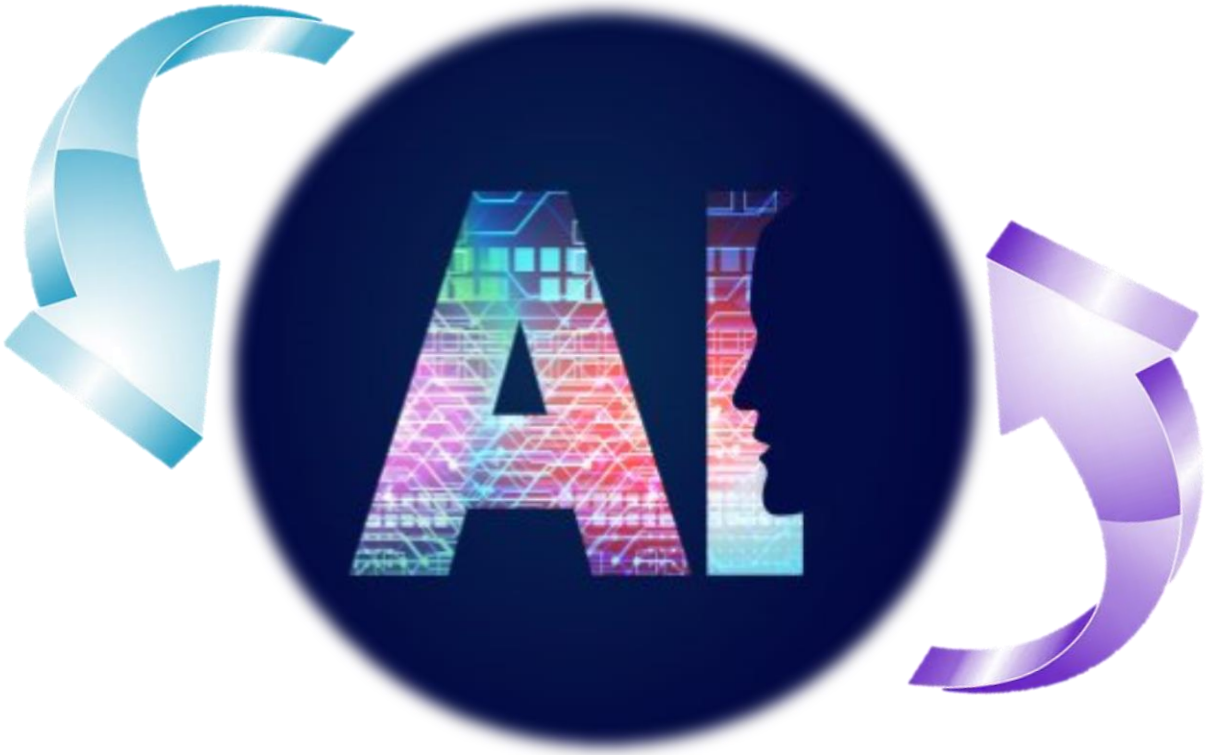


对AI时代新型软件缺陷进行调试的普遍范式

AI时代下新型缺陷的调试

AI已经并正在不断重构软件调试任务的既有范式

- 与传统缺陷相比，AI系统缺陷具有新形式、新特征、新样态。
- 需要为AI软件系统量身打造与其适配的缺陷定位、修复技术。



AI重构软件调试任务的两条路径

- AI技术特别是大模型的兴起，为程序自动修复任务提供了全新的解决方案。
- AI时代的程序修复需要格外重视生成补丁的**正确性**与**可理解性**。

AI时代下新型缺陷的调试

现有工具汇总 – SD 4 AI

Table A3 SD4AI tools (material)

Paper	Description	Website
[112]	A technique to identifying training points most responsible for a given prediction	http://bit.ly/gt-influence
[113]	KARMA: A causal unlearning system that uses several mechanisms to determine the set of polluted data sample	https://github.com/CausalUnlearning/KARMA
[131]	LDADE: A search-based tool that tunes LDA's parameters to fix its systematic errors	https://github.com/ai-se/Pits_lda/
[277]	An error correction mechanism for NMT	https://github.com/StillKeepTry/ECM-NMT
[274]	TRADER: A tool for debugging RNN models	https://github.com/trader-rnn/TRADER
[268]	BugDoc: A framework for finding root causes of errors in computational pipelines	https://github.com/ViDA-NYU/BugDoc
[278]	DEBAR: A static analysis tool for detecting numerical bugs in neural architectures	https://github.com/ForeverZyh/DEBAR
[270]	DeepLocalize: A tool to identifying the root causes for DNN errors	https://github.com/Wardat-ISU/DeepLocalize
[119]	175 faults gathered from GitHub and StackOverflow	https://github.com/ForeverZyh/TensorFlow-Program-Bugs
[290]	499 faults from 16851 commits in Baidu Apollo and Autoware	http://tiny.cc/cps_bug_analysis
[289]	202 TensorFlow bug fixes (with some bug reports)	https://github.com/fordataupload/tfbugdata
[34]	667 DNN instances (including bug and repair) for DNN repair	https://github.com/lab-design/ICSE2020DNNBugRepair
[121]	1981 commits, 1392 issues/pull requests and 2653 posts for delivering a taxonomy of real faults in deep learning systems	https://github.com/dlfaults/dl_faults
[282]	59 deep learning repositories on GitHub for investigating the scent of deep learning code	https://github.com/Hadhemii/DLCodeSmells
[291]	304 deployment faults gathered from GitHub and StackOverflow	https://github.com/chenzhenpeng18/icse2021
[283]	1149 labeled academic AI repositories (popular or unpopular)	https://github.com/YuanruiZJU/academic-ai-repos

Yi Song, Xiaoyuan Xie and Baowen Xu. When Debugging Encounters Artificial Intelligence: State of the Art and Open Challenges. *SCIENCE CHINA Information Sciences*, 2023, ISSN 1674-733X, https://doi.org/10.1007/s11432-022-3803-9.

AI时代下程序补丁的可信性

AI生成的程序补丁并不总是可靠

```
public double getPct(Object v) {
-     return getCumPct((Comparable<?>) v);                    Buggy line
+     return getPct((Comparable<?>) v);                       APR tools' patch   ✓
+     return (double ) v ;                                    PLBART's patch     ✗
+     if (v instanceof Number)                                CodeGen's patch    ✗
+         return ((Number) v).doubleValue();
+     else if (v instanceof Boolean)
+         return ((Boolean) v).booleanValue()? 1.0 : 0.0;
+     else if (v instanceof String)
+         return Double.parseDouble((String) v);
+     else return 0.0;
+     Number n = (Number) v;                                  InCoder's patch    ✗
+     return n.doubleValue() * 100.0;
}
```

大模型为异常程序生成了错误的补丁

大模型生成补丁时的常见错误

- 语法错误 (不能编译)
- 实现了非用户预期的功能
- 生成与Prompt无关的代码片段
- 难以生成出现频率低的token
- 不当的变量返回类型

Wei Y, Xia C S, Zhang L. Copiloting the Copilots: Fusing Large Language Models with Completion Engines for Automated Program Repair. arXiv preprint arXiv:2309.00608, 2023.

Jiang N, Liu K, Lutellier T, et al. Impact of code language models on automated program repair. arXiv preprint arXiv:2302.05020, 2023.

AI时代下程序补丁的可信性

由于黑盒属性

“难以判定补丁正确性”这一问题

在基于AIGC甚至LLM的程序自动修复中尤为明显

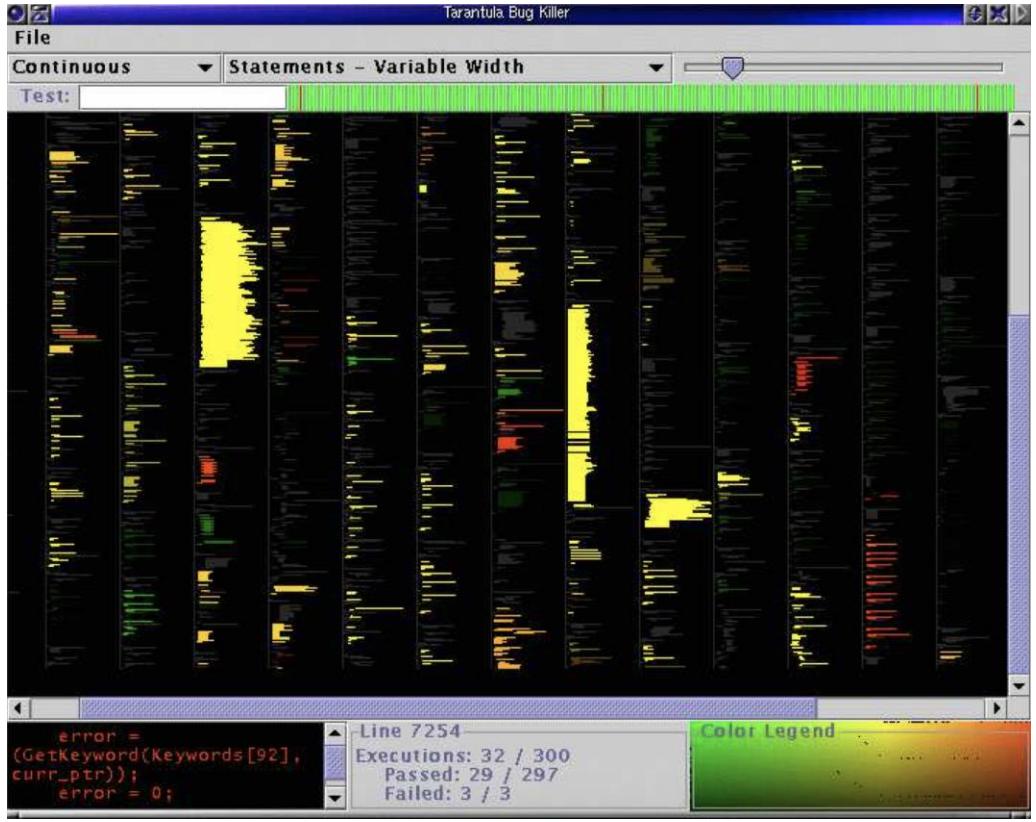
提升自动化调试技术的**可理解性**

对于促进对结果的采纳、辅助人类程序员决策

具有重要意义

AI时代下软件调试任务的可理解性

软件工程界对可理解性关注已久 —— 来自2002年的尝试



程序代码风险值可视化频谱

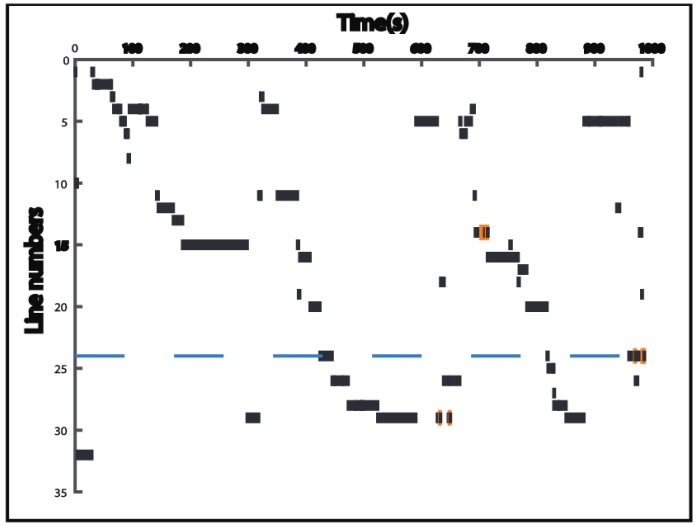
早在2002年，Jones等人就提出构造程序代码可视化频谱，用颜色的深浅标识代码行与失效行为的关联程度，以此辅助人类程序员的缺陷定位。

该工作有效缓解了传统SBFL (基于频谱的缺陷定位技术)存在的语义上下文缺失问题，帮助人类程序员在整体视角下理解可疑语句所处的方位。

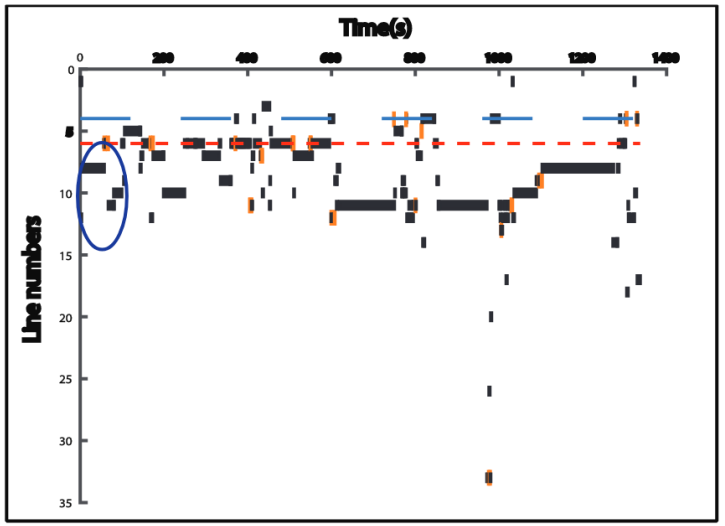
Jones J A, Harrold M J, Stasko J. Visualization of test information to assist fault localization[C]//Proceedings of the 24th international conference on Software engineering. 2002: 467-477.

AI时代下软件调试任务的可理解性

软件工程界对可理解性关注已久 —— 207名程序员参与的人类研究



无SBFL协助的代码检查



SBFL协助的代码检查

横轴：时间线
纵轴：代码行
---：真实缺陷位置
---：SBFL推荐位置
■：程序员聚焦点

SBFL (最主流缺陷定位技术之一) 的结果缺乏可理解性，人类程序员拒绝在不理解的情况下直接接纳其结果，而是会自行寻求理解，使得该自动化技术难以真正帮助提升调试的效率。

Xiaoyuan Xie, Zicong Liu, Shuo Song, et al. Revisit of automatic debugging via human focus-tracking analysis. 2016 38th International Conference on Software Engineering (ICSE 2016)

AI时代下软件调试任务的可理解性

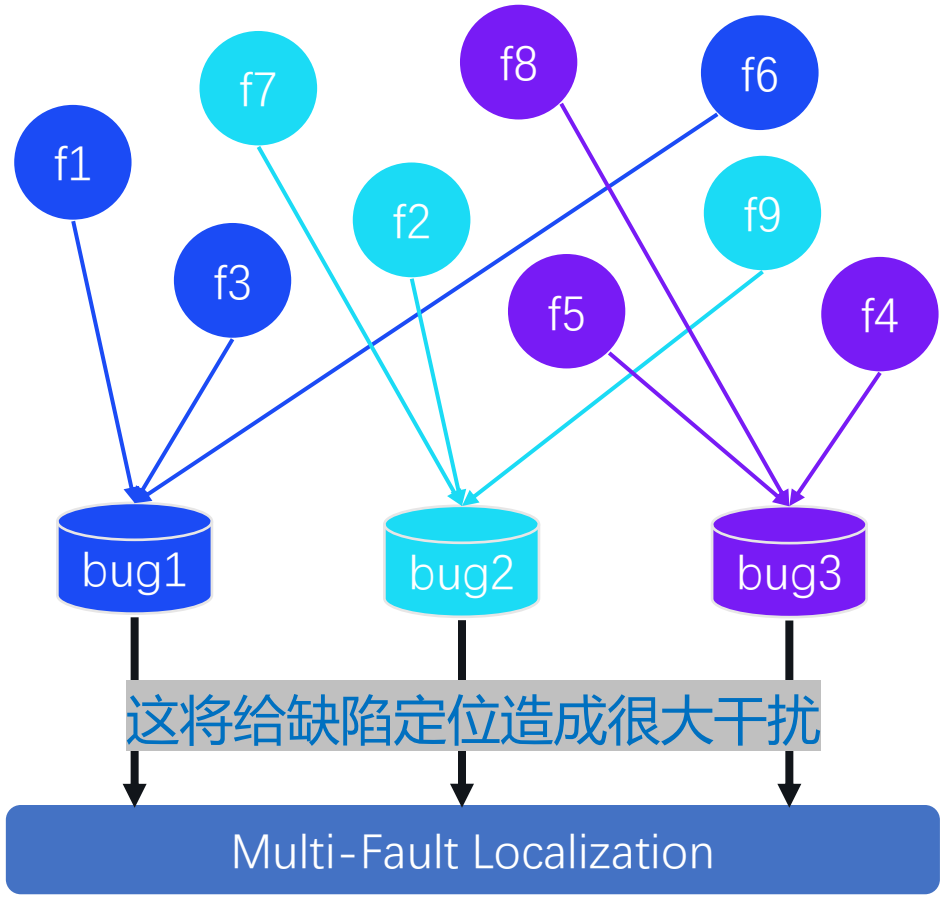
提升软件调试任务的可理解性 —— 以多缺陷隔离任务为例

多缺陷环境 (异常程序中含有多处缺陷)

是软件测试调试面临的常态

多缺陷环境下：

各个失效行为可能对应不同缺陷，指向混乱复杂。



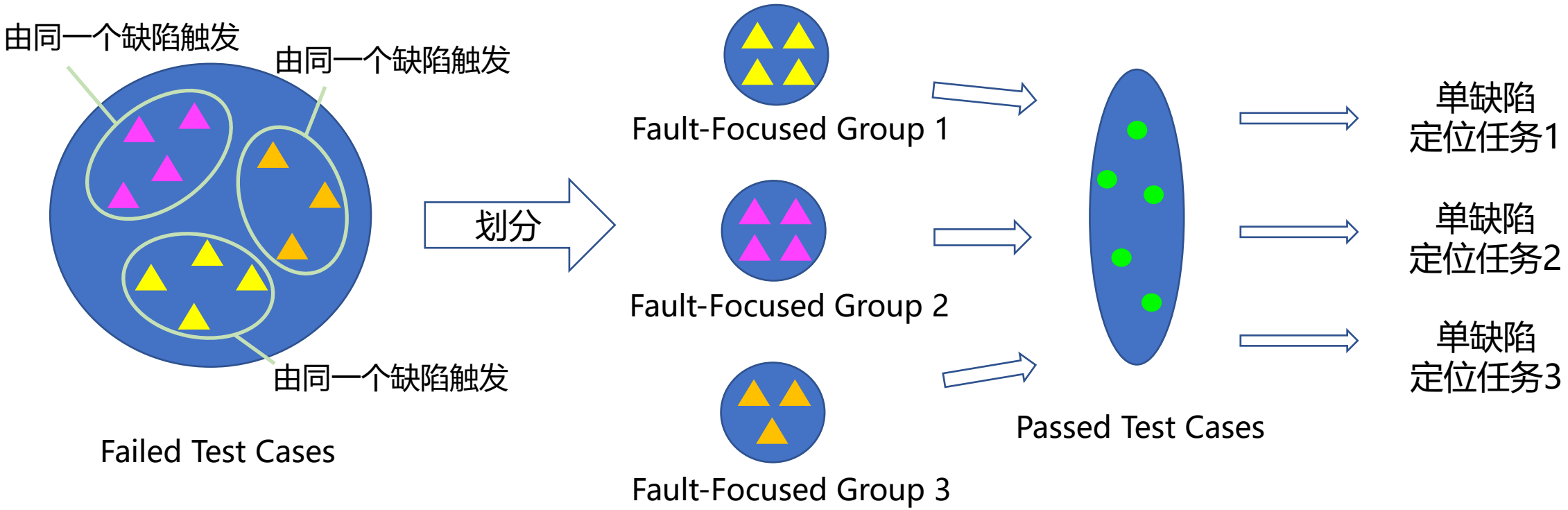
Yi Song, Xiaoyuan Xie, Quanming Liu, Xihao Zhang and Xi Wu. A Comprehensive Empirical Investigation on Failure Clustering in Parallel Debugging. Journal of Systems and Software (JSS), 2022, 193: 111452

Yi Song, Xiaoyuan Xie, Xihao Zhang, Quanming Liu and Ruizhi Gao. Evolving Ranking-Based Failure Proximities for Better Clustering in Fault Isolation. 2022 37th IEEE/ACM International Conference on Automated Software Engineering (ASE 22)

AI时代下软件调试任务的可理解性

提升软件调试任务的可理解性 —— 以多缺陷隔离任务为例

并行调试是解决多缺陷定位问题的有效途径，其核心在于**如何恰当地对软件异常行为（通常为失败测试用例）根据其缺陷根因进行划分**，进而实现对多缺陷的隔离。



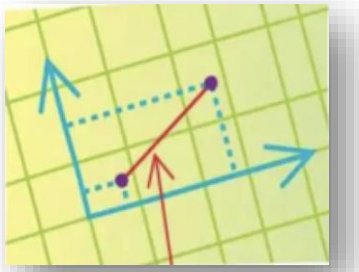
AI时代下软件调试任务的可理解性

提升软件调试任务的可理解性 —— 以多缺陷隔离任务为例

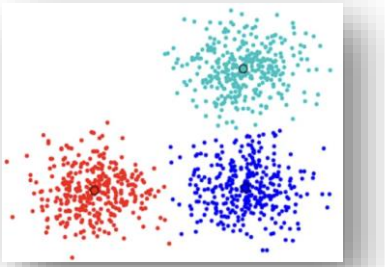
多缺陷隔离的关键步骤

核心

Fingerprinting Function



Distance Metric



Clustering Algorithm

也称**Failure Proximity (失效近似度)**

如何准确提取失效行为的特征，从而对其进行数学化、形式化的建模和表征？

如何对已经被二次表征的失效行为进行相似性度量，以判断其是否由同一个缺陷触发？

如何基于上游距离信息，对所有失效行为进行聚类，以实现对缺陷的隔离？

Yi Song, Xihao Zhang, Xiaoyuan Xie, Quanming Liu, Ruizhi Gao and Chenliang Xing. ReClues: Representing and indexing failures in parallel debugging with program variables. 2024 46th International Conference on Software Engineering (ICSE 24)

AI时代下软件调试任务的可理解性

基于可视化程序内存谱的多缺陷隔离方法SURE

SURE为提升
多缺陷隔离结果可理解性
提供的解决方案

② 执行失败测试用例，在断点处收集运行态内存信息

③ 将表征失败测试用例的内存信息转化为可视化内存谱图片PMS

① 利用SBFL技术确定若干高风险语句，并设置断点

④ 将内存谱图片对根据缺陷根因的异同打上标签，并训练模型

⑤ 利用模型对需要预测相似性的失败测试用例对进行度量

⑥根据相似性得到距离

⑦ 基于距离进行失效行为聚类

Yi Song, Xihao Zhang, Xiaoyuan Xie, Songqiang Chen, Quanming Liu and Ruizhi Gao. SURE: A Visualized Failure Indexing Approach using Program Memory Spectrum. arXiv preprint arXiv:2310.12415, 2023.

AI时代下软件调试任务的可理解性

基于可视化程序内存谱的多缺陷隔离方法SURE

失败测试用例 f_i 由内存信息 MI_i 表征

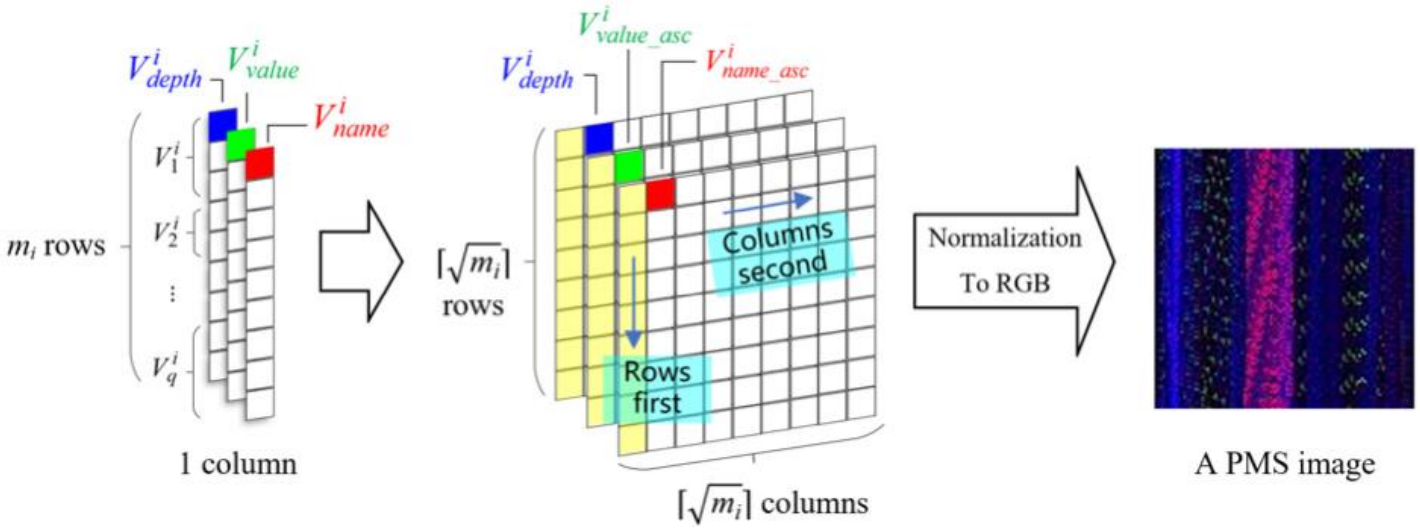
$$MI_i = \left\{ bp'_1 : V_1^i, bp'_2 : V_2^i, ..., bp'_j : V_j^i, ..., bp'_q : V_q^i \right\}$$

MI_i 包括变量名、变量值、栈帧号三个维度

$$V_{name}^i = V_{name_1}^i \oplus V_{name_2}^i \oplus ... \oplus V_{name_j}^i \oplus ... \oplus V_{name_q}^i ,$$

$$V_{value}^i = V_{value_1}^i \oplus V_{value_2}^i \oplus ... \oplus V_{value_j}^i \oplus ... \oplus V_{value_q}^i ,$$

$$V_{depth}^i = V_{depth_1}^i \oplus V_{depth_2}^i \oplus ... \oplus V_{depth_j}^i \oplus ... \oplus V_{depth_q}^i$$



设计人类友好的失效近似度

将内存信息转换为可视化内存谱图片PMS

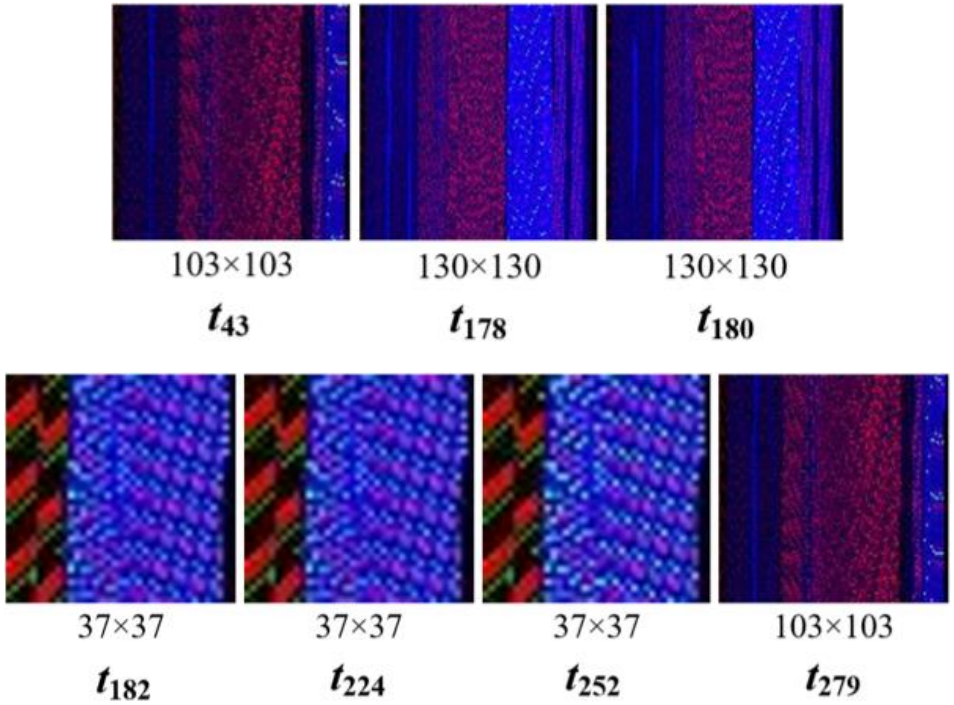
AI时代下软件调试任务的可理解性

基于可视化程序内存谱的多缺陷隔离方法SURE

```
3164    if (leftoversf)
3165      {
3166          copyset(leftovers, labels[ngrps]);
3167          copyset(intersect, labels[j]);
3168 -        MALLOC(grps[ngrps].elems, position, d->nleaves);
      +        MALLOC(grps[ngrps].elems, position, d->nleaves*-1);
3169          copy(&grps[j], &grps[ngrps]);
3170          ++ngrps;
3171       }
        ...
4681    for (ep = text + size - 11 * len;;)
4682      {
4683          while (tp <= ep)
4684              {
4685 -                d = d1[U(tp[-1])], tp += d;
      +                d = d1[U (! tp[-1])], tp += d;
4686                  d = d1[U(tp[-1])], tp += d;
4687                  if (d == 0)
4688                    goto found;
        ...
7480    compile_stack.stack = TALLOC (INIT_COMPILE_STACK_SIZE, compile_stack_elt_t);
7481    if (compile_stack.stack == NULL)
7482      return REG_ESPACE;
7484 -  compile_stack.size = INIT_COMPILE_STACK_SIZE;
      +  compile_stack.size = 0;
7485    compile_stack.avail = 0;
```

以一个3-bug错误程序为例

(7个失败测试用例对应3个缺陷根因)



SURE可视化结果与Oracle高度一致：

$\{t_{43}, t_{279}\}$ $\{t_{178}, t_{180}\}$ $\{t_{182}, t_{224}, t_{252}\}$

AI时代下软件调试任务的可理解性

面向人类程序员的实用性调研证实了SURE的可理解性

		Participant														Total
		No. 1	No. 2	No. 3	No. 4	**No. 5**	No. 6	No. 7	No. 8	No. 9	No. 10	No. 11	No. 12	No. 13	No. 14	No. 15	
PMS	V^T_{equal}	2	3	2	3	**3**	3	3	3	3	2	3	2	1	3	3	39
	$S\_M^T_{FMI}$	2.00	3.00	2.00	3.00	**3.00**	3.00	3.00	3.00	3.00	2.00	3.00	2.00	1.00	3.00	3.00	39.00
	$S\_M^T_{JC}$	2.00	3.00	2.00	3.00	**3.00**	3.00	3.00	3.00	3.00	2.00	3.00	1.96	1.00	3.00	3.00	38.96
	$S\_M^T_{PR}$	2.00	3.00	2.00	3.00	**3.00**	3.00	3.00	3.00	3.00	2.00	3.00	1.96	1.00	3.00	3.00	38.93
	$S\_M^T_{RR}$	2.00	3.00	2.00	3.00	**3.00**	3.00	3.00	3.00	3.00	2.00	3.00	2.62	1.00	2.33	2.33	3.26
	Average Time (s)	5.41	3.08	1.50	6.22	**1.17**	3.24	1.39	3.42	4.43	5.63	3.11					
Ranking list	V^T_{equal}	2	0	3	2	**0**	1	0	1	2	2	1				3	19
	$S\_M^T_{FMI}$	1.99	0.00	3.00	1.99	**0.00**	1.00	0.00	0.99	2.00	1.99	0.99				3.00	18.95
	$S\_M^T_{JC}$	1.99	0.00	3.00	1.99	**0.00**	1.00	0.00	0.98	2.00	1.99	0.98	1.00	1.00	0.00	3.00	18.93
	$S\_M^T_{PR}$	1.81	0.00	3.00	1.62	**0.00**	1.00	0.00	0.46	2.00	1.73	0.55				3.00	17.17
	$S\_M^T_{RR}$	1.81	0.00	3.00	1.66	**0.00**	1.00	0.00	0.40	2.00	1.71	0.46				3.00	17.04
	Average Time (s)	14.33	25.37	18.50	23.21	**34.92**	21.08	27.38	44.98	13.92	13.11	20.63				6.33	21.24
Coverage	V^T_{equal}	0	2	1	1	**1**	0	2	2	0	2	2				0	19
	$S\_M^T_{FMI}$	0.00	1.99	1.00	1.00	**1.00**	0.00	2.00	1.99	0.00	2.00	1.99	0.99	3.00	2.00	0.00	18.96
	$S\_M^T_{JC}$	0.00	1.99	1.00	1.00	**1.00**	0.00	2.00	1.99	0.00	2.00	1.98				0.00	18.95
	$S\_M^T_{PR}$	0.00	1.72	1.00	1.00	**1.00**	0.00	2.00	1.72	0.00	2.00	1.62	0.48	3.00	1.87	0.00	17.41
	$S\_M^T_{RR}$	0.00	1.66	1.00	1.00	**1.00**	0.00	2.00	1.72	0.00	2.00	1.66	0.47	3.00	1.87	0.00	17.38
	Average Time (s)	12.00	16.86	38.59	20.17	**14.26**	46.63	20.33	35.40	34.31	24.72	14.78	21.90	3.33	12.93	12.71	21.93

基于SURE对失败测试用例的表征，人类程序员执行多缺陷隔离任务的有效性相较于当前最主流的两种表征形式均提升105.26%。

基于SURE的表征，人类程序员执行多缺陷隔离任务的时间开销相较于当前最主流的两种表征形式分别降低84.65%和85.13%。

PART 04
总结与展望

AI赋能的自动化软件调试

现状总结

- AI赋能的自动化软件调试研究热度不断攀升
- AI技术已经并正在不断重构许多传统软件工程任务的既有范式
- AI赋能下的软件调试表现出更强的能力甚至涌现出传统策略之外的新技术样态
- AI技术的蓬勃发展为软件调试任务定义了许多新的问题 (如面向非代码缺陷的调试)
- AI时代下软件调试技术的可理解性愈发受到关注

AI赋能的自动化软件调试

未来展望 (潜在挑战)

- 寻找更多AI技术与传统调试任务耦合的接口 扩大AI赋能调试的作用面
- AI驱动的缺陷定位、程序修复技术应更进一步提升结果的可理解性
- 对AI时代下的软件缺陷进行系统、规范、形式化的定义
- 开发面向AI系统特定缺陷类型的可信质量保障技术
- 利用AI技术缓解多缺陷环境下的调试难题

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

THANKS