Code, Critique, Cure: Advancing LLM *Reasoning* for AI-Augmented Software Maintenance

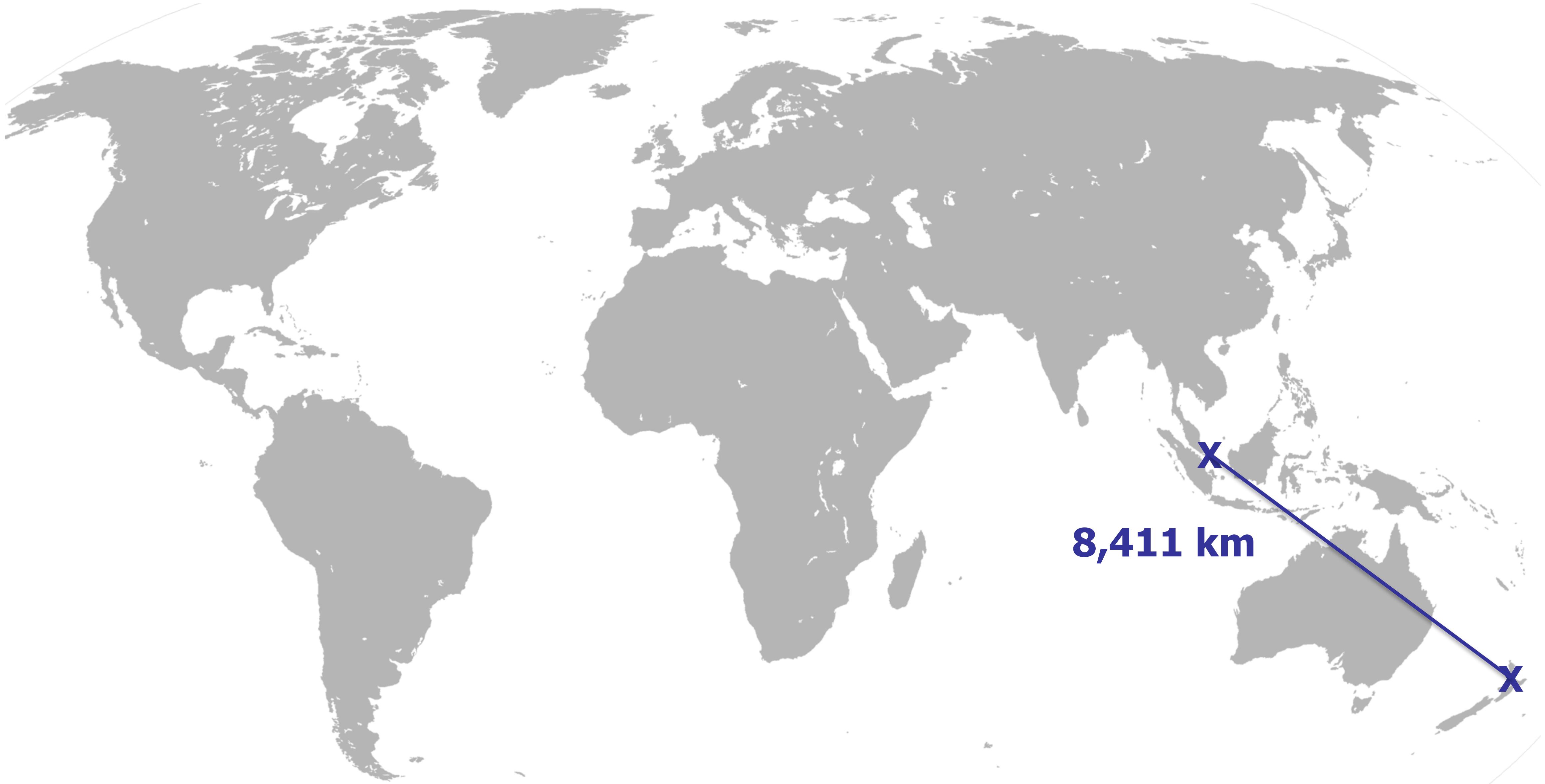
David Lo, FACM, FIEEE

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

ICSME 2025, Auckland, New Zealand, September 2025

Self-Introduction



Self-Introduction

Self-Introduction

Self-Introduction

Singapore Management University

- Third university in Singapore
- Number of students:
  - 8000+ (UG)
  - 1800+ (PG)
- Schools:
  - Business
  - Computing
  - Economics
  - Accountancy
  - Law
  - Social Science

CELEBRATING MEANINGFUL IMPACT 25 YEARS AND BEYOND

Center for Research on Intelligent Software Engineering (RISE)

Elsevier JSS'21, Bibliometric Study

Table 3
Most active institutions in software engineering

Rank	Name
1	University of California
2	Carnegie Mellon University
3	Nanjing University
4	Microsoft Research
5	Singapore Management University

CSRankings, SE, Aug 2025

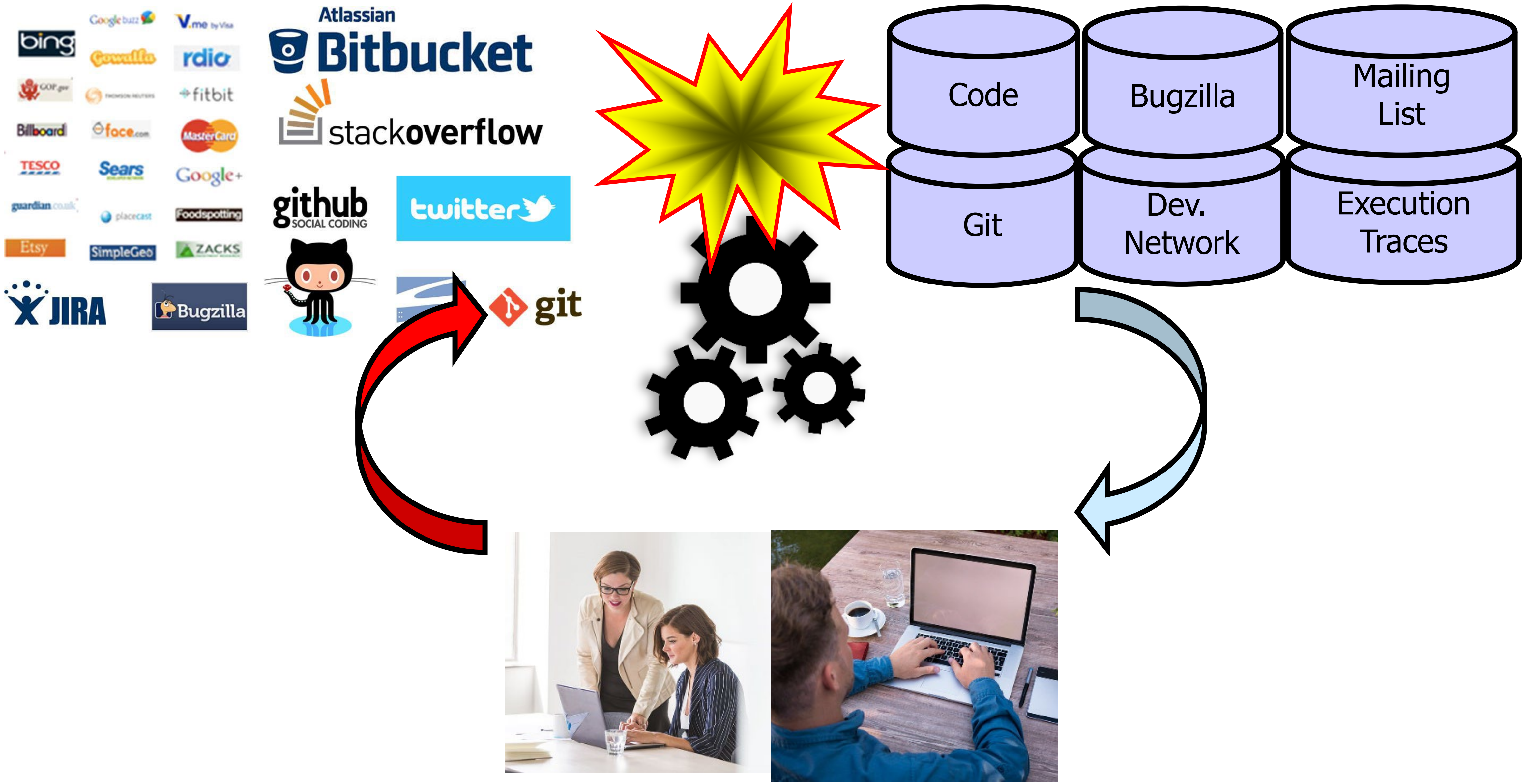
#	Institution	Count	Faculty
1	▶ Nanjing University	45.6	40
2	▶ Peking University	31.3	22
3	▶ Carnegie Mellon University	31.2	16
4	▶ Singapore Management University	25.0	9

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Centre for Research on Intelligent Software Engineering

AI for Software Engineering



Experience with AI4SE

SMArTIC: Towards Building an Accurate, Robust and Scalable Specification Miner

FSE’06

David Lo and Siau-Cheng Khoo
Department of Computer Science,National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

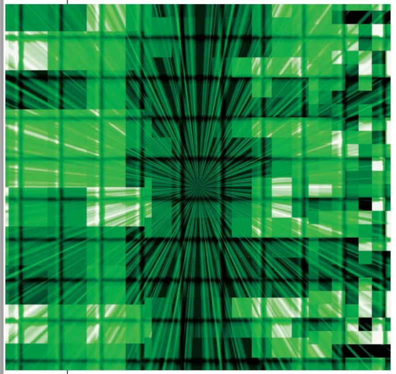
Efficient Mining of Iterative Patterns for Software Specification Discovery

KDD’07

David Lo and Siau-Cheng Khoo
Department of Computer Science
National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

Chao Liu
Department of Computer Science
University of Illinois-UC
chaoliu@cs.uiuc.edu

COVER FEATURE



IEEE Computer’09

DATA MINING FOR SOFTWARE ENGINEERING

Tao Xie and Suresh Thummalapenta, *North Carolina State University*
David Lo, *Singapore Management University*
Chao Liu, *Microsoft Research*

Experience with AI4SE

Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach

KDD'09

David Lo
Singapore Management University
davidlo@smu.edu.sg

Hong Cheng*
Chinese University of Hong Kong
hcheng@se.cuhk.edu.hk

Jiawei Han†
University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

Siau-Cheng Khoo and Chengnian Sun
National University of Singapore
{khoosc,suncn}@comp.nus.edu.sg

Test oracle generation

A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval

ICSE'10

Chengnian Sun[1], David Lo[2], Xiaoyin Wang[3], Jing Jiang[2], Siau-Cheng Khoo[1]
[1]School of Computing, National University of Singapore
[2]School of Information Systems, Singapore Management University
[3]Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education
suncn@comp.nus.edu.sg, davidlo@smu.edu.sg, wangxy06@sei.pku.edu.cn,
jingjiang@smu.edu.sg, khoosc@comp.nus.edu.sg

Intelligent issue trackers

Tag Recommendation in Software Information Sites

MSR'13

Xin Xia*‡, David Lo†, Xinyu Wang*, and Bo Zhou*§
*College of Computer Science and Technology, Zhejiang University
†School of Information Systems, Singapore Management University

Intelligent crowdsourced SE

History Driven Program Repair

SANER'16

Xuan-Bach D. Le, David Lo
School of Information Systems
Singapore Management University
{dxb.le.2013,davidlo}@smu.edu.sg

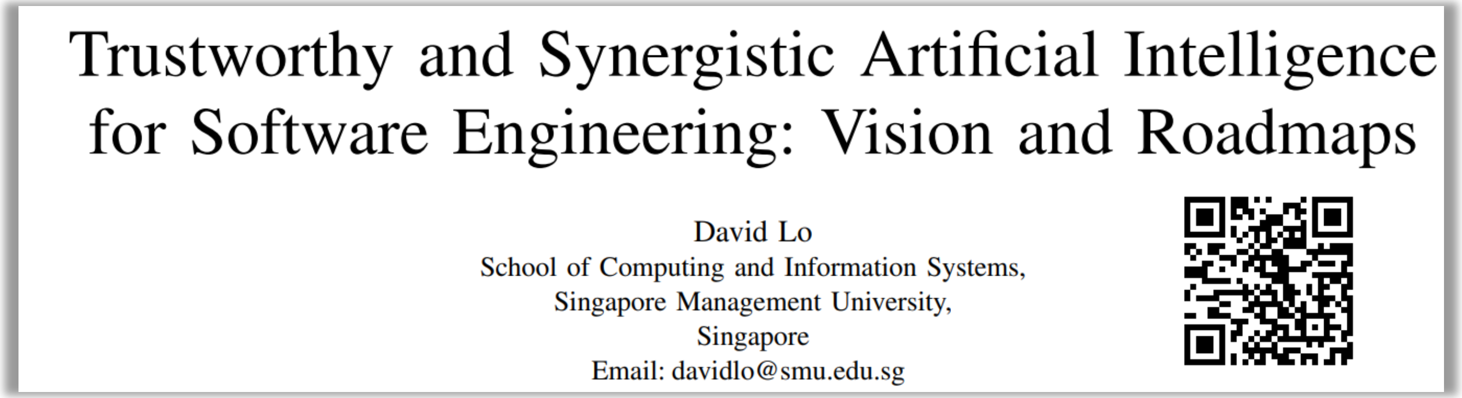
Claire Le Goues
School of Computer Science
Carnegie Mellon University
clegoues@cs.cmu.edu

Intelligent program repair

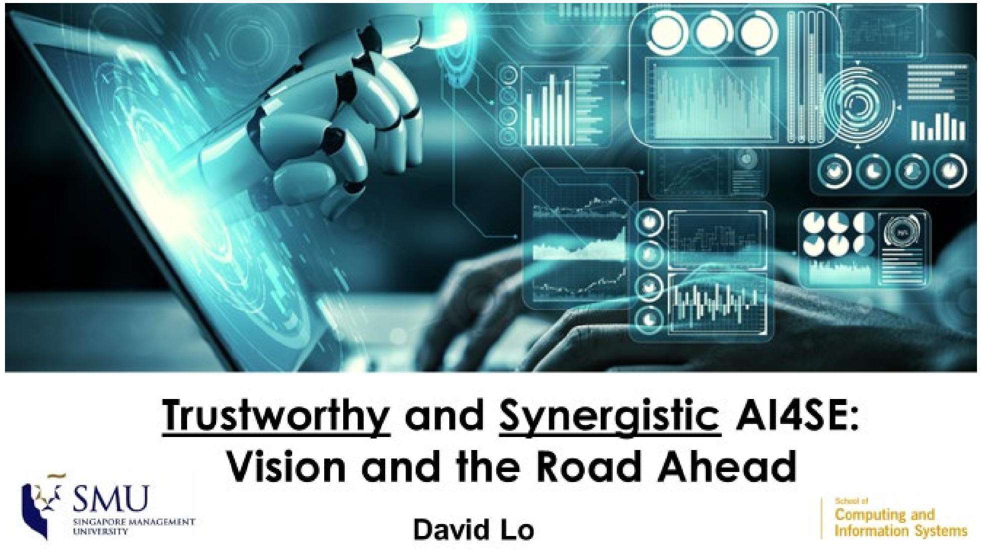
"History-driven program repair influence our work, the overall pipeline is similar"

- Facebook Engineers

Our Research Agenda in AI4SE


Trustworthy and Synergistic Artificial Intelligence for Software Engineering: Vision and Roadmaps

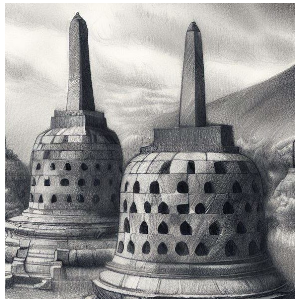
David Lo
School of Computing and Information Systems,
Singapore Management University,
Singapore
Email: davidlo@smu.edu.sg



ICSE'23 Future of SE Talk

AI for Software Engineering

History	Challenges	Vision



Roadmap I	Roadmap II	Call4Action

Towards Software Engineering 2.0

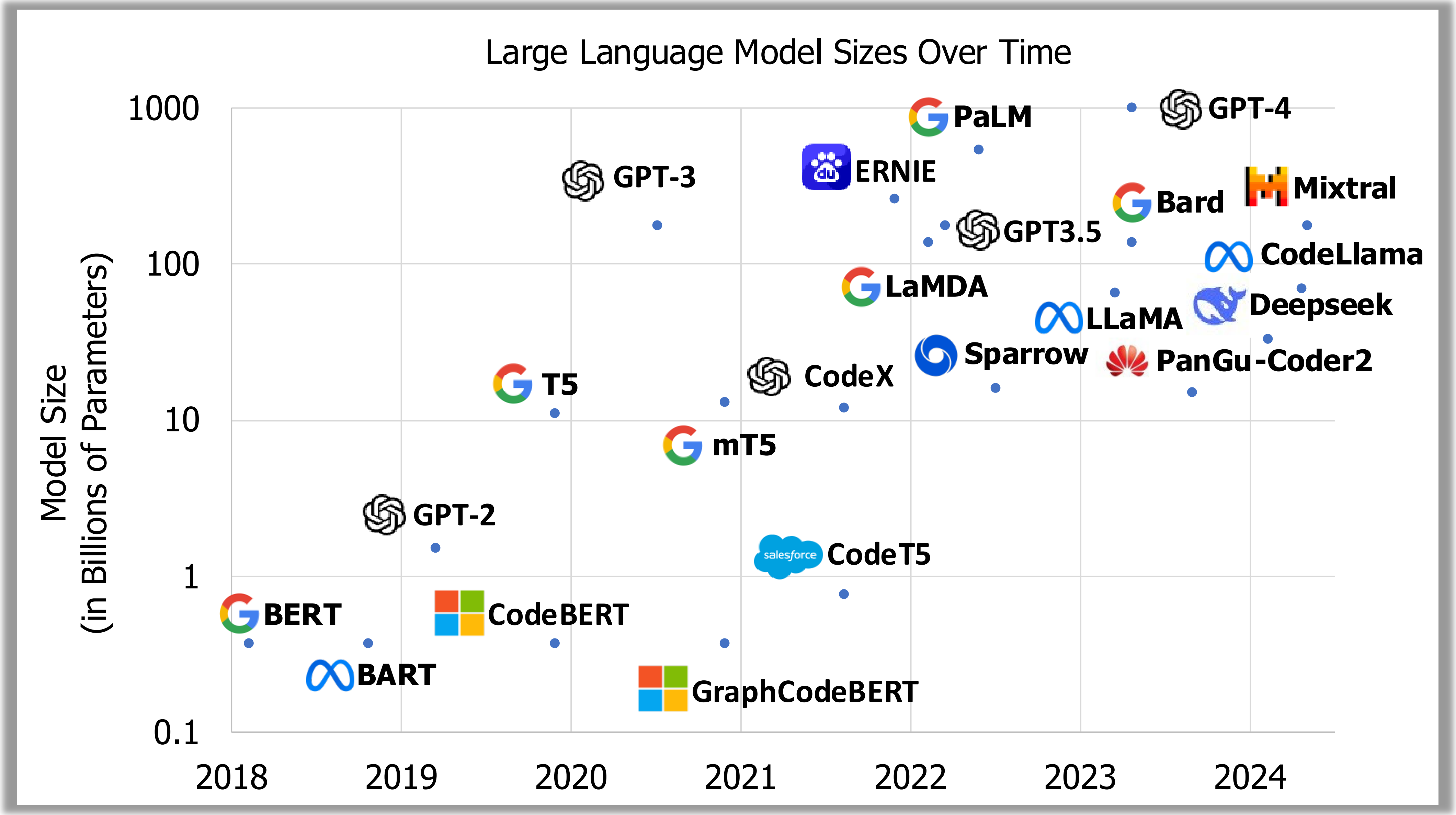
"If you want to go far, go together" – African Proverb

Code, Critique, Cure: Advancing LLM *Reasoning* for AI-Augmented Software Maintenance

SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

Large Language Models (LLMs)



LLM Can Greatly Help SE Tasks

ICSME 2020

Sentiment Analysis for Software Engineering: How Far Can Pre-trained Transformer Models Go?

Ting Zhang, Bowen Xu*, Ferdian Thung, Stefanus Agus Haryono, David Lo, Lingxiao Jiang
School of Information Systems, Singapore Management University
Email: {tingzhang.2019, bowenxu.2017}@phdcs.smu.edu.sg, {ferdianthung, stefanusah, davidlo, lxjiang}@smu.edu.sg

Early work on LLM4SE, among most cited papers of ICSME 2020

ICSME 2021

Assessing Generalizability of CodeBERT

Xin Zhou, DongGyun Han, and David Lo
School of Computing and Information Systems, Singapore Management University
xinzhou.2020@phdcs.smu.edu.sg, {dhan, davidlo}@smu.edu.sg

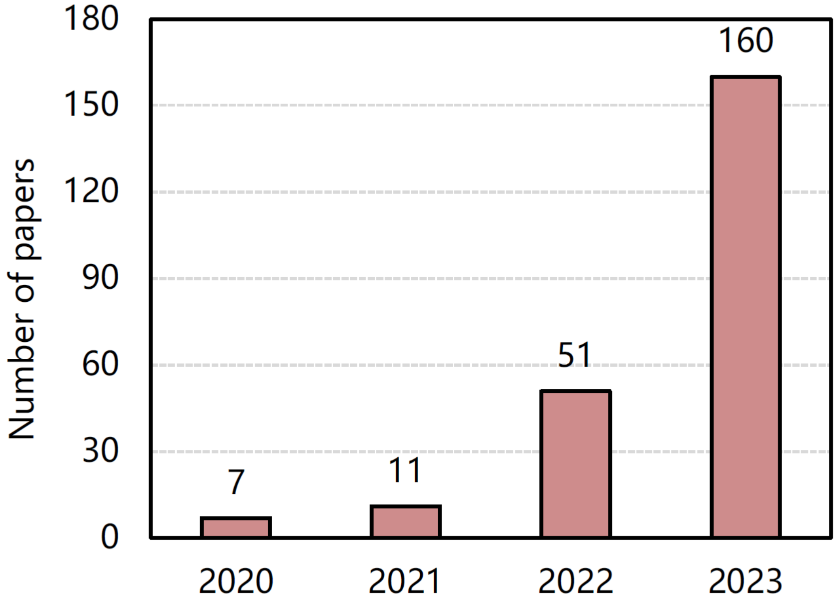
Early work on benchmarking code LLM, among most cited papers of ICSME 2021

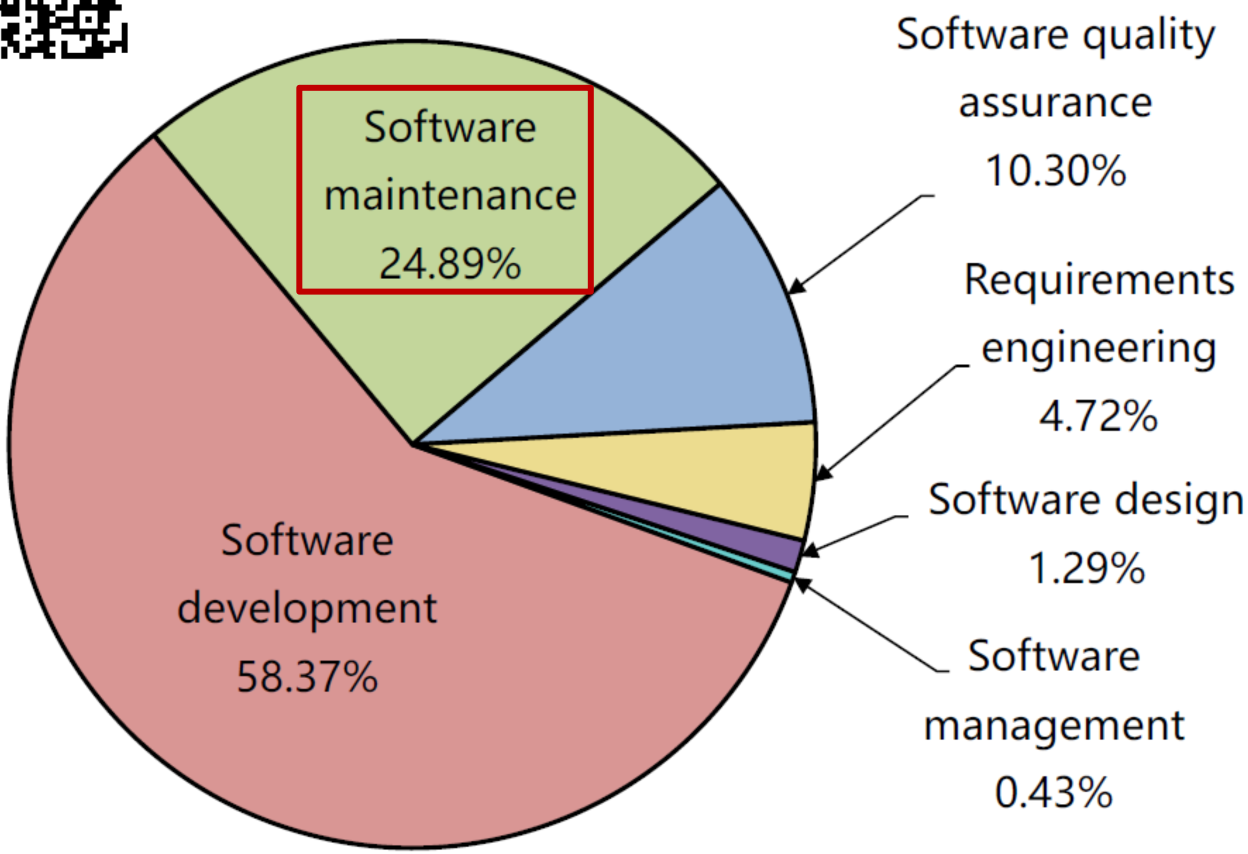
LLMs Seem to Win for Many SE Scenarios

TOSEM 2024

Large Language Models for Software Engineering: A Systematic Literature Review

XINYI HOU*, Huazhong University of Science and Technology, China
YANJIE ZHAO*, Monash University, Australia
YUE LIU, Monash University, Australia
ZHOU YANG, Singapore Management University, Singapore
KAILONG WANG, Huazhong University of Science and Technology, China
LI LI, Beihang University, China
XIAPU LUO, The Hong Kong Polytechnic University, China
DAVID LO, Singapore Management University, Singapore
JOHN GRUNDY, Monash University, Australia
HAOYU WANG†, Huazhong University of Science and Technology, China





Despite Successes, Much Work Remains

Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity

Joel Becker*, Nate Rush*, Beth Barnes, David Rein

Model Evaluation & Threat Research (METR)

InfoWorld

AI coding tools can slow down seasoned developers by 19%

News

Jul 11, 2025 • 6 mins

Despite Successes, Much Work Remains

Factor	Relevant Observations
Over-optimism about AI usefulness (C.1.1)	• Developers forecast AI will decrease implementation time by 24% • Developers post hoc estimate AI decreased implementation time by 20%
High developer familiarity with repositories (C.1.2)	• Developers slowed down more on issues they are more familiar with • Developers report that their experience makes it difficult for AI to help them • Developers average 5 years experience and 1,500 commits on repositories
Large and complex repositories (C.1.3)	• Developers report AI performs worse in large and complex environments • Repositories average 10 years old with $>$1,100,000 lines of code
Low AI reliability (C.1.4)	• Developers accept $<$44% of AI generations • Majority report making major changes to clean up AI code • 9% of time spent reviewing/cleaning AI outputs
Implicit repository context (C.1.5)	• Developers report AI doesn't utilize important tacit knowledge or context

LLM has **limited reasoning capabilities** for software maintenance tasks

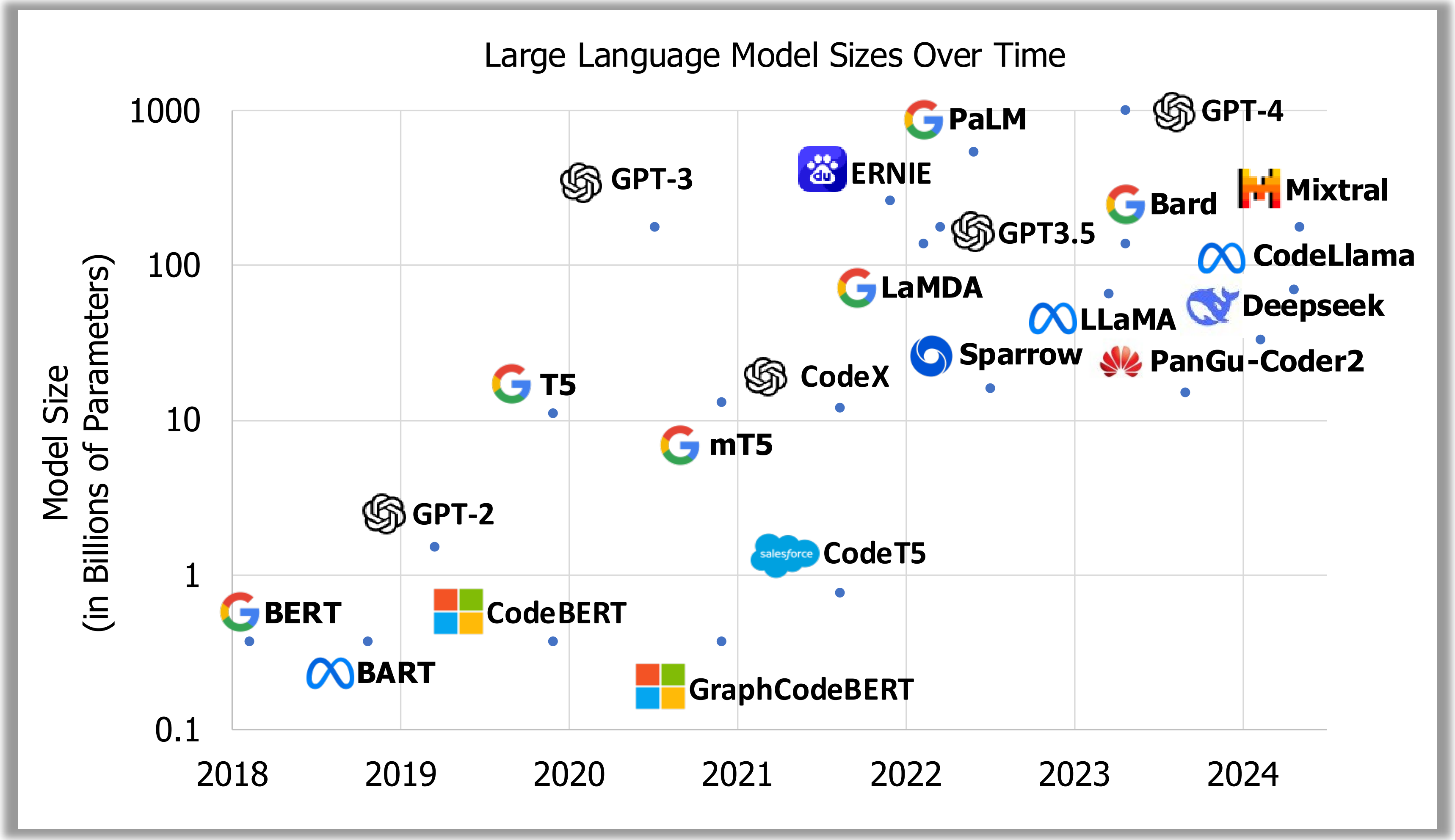
How Can We **Improve** It?

Too Much Focus on Models

"99% of the papers were model-centric with only 1% being data-centric"

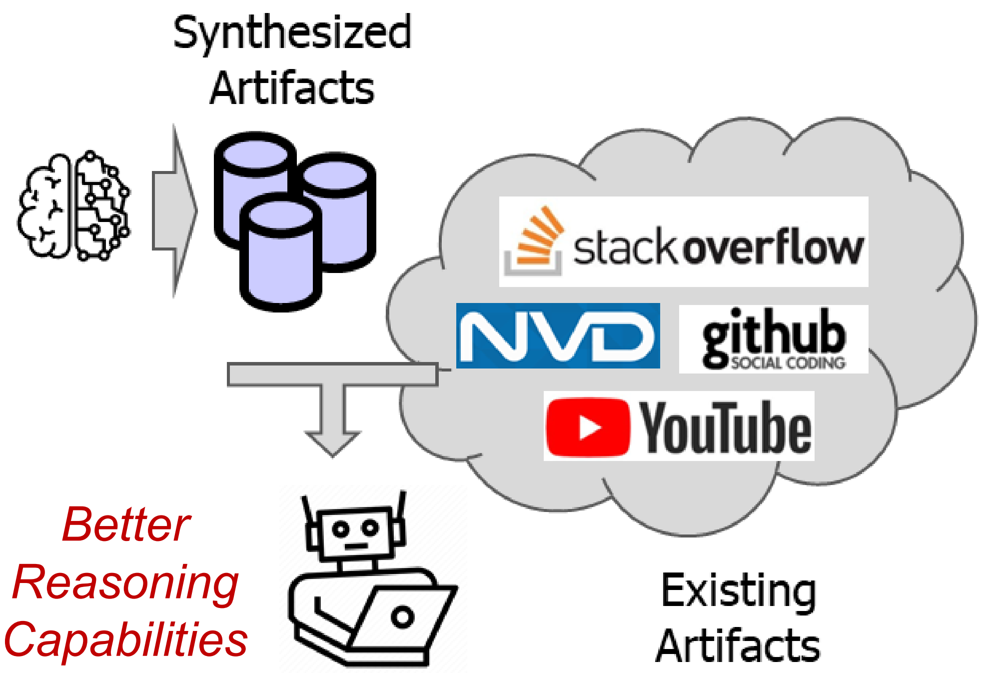
– Andrew Ng
Google Brain Co-Founder

DATA-CENTRIC AI



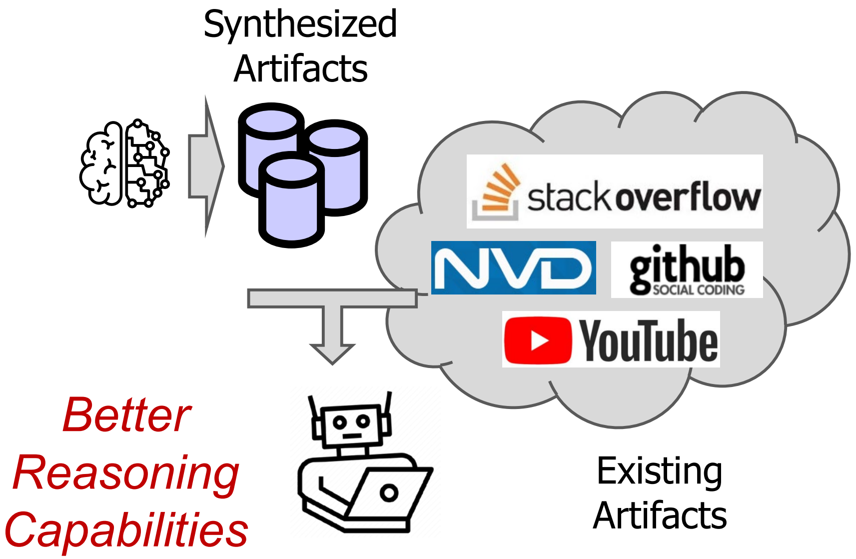
Data-Centric LLM4SE

- Use SE domain knowledge for software **artifact engineering**
- to *guide LLM reasoning*
- for *specific* SE tasks



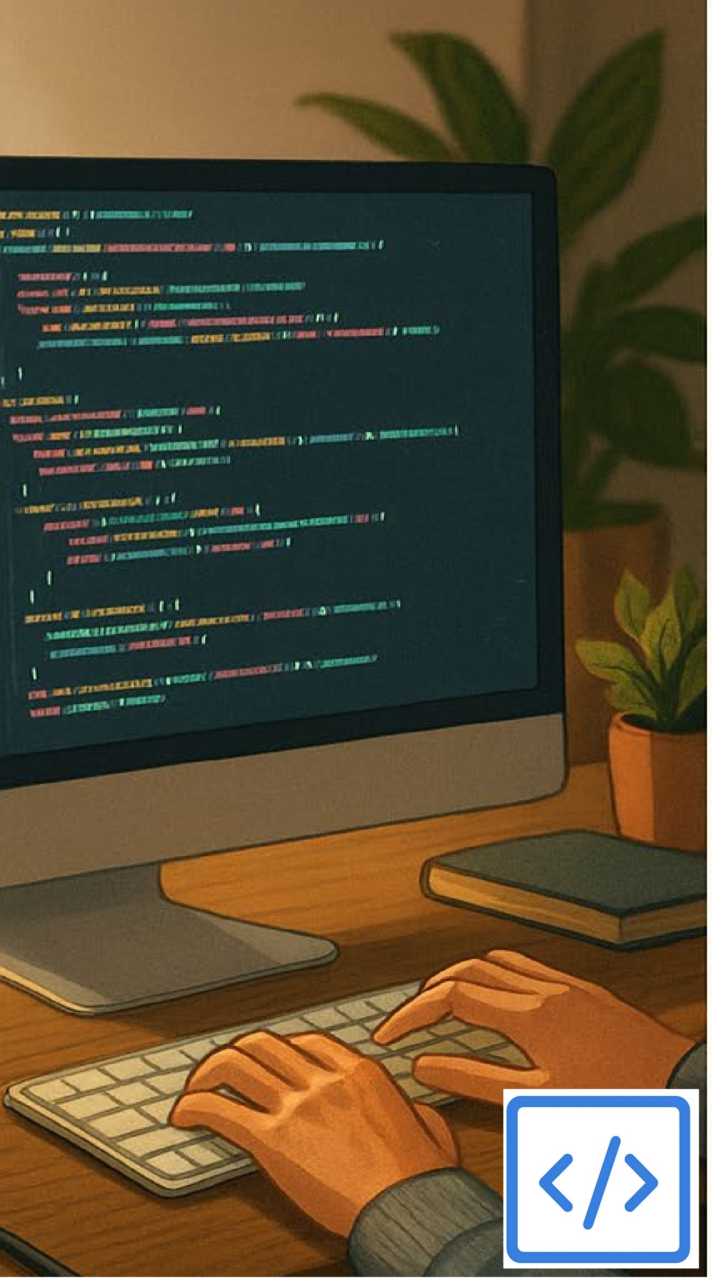
Data-Centric LLM4SE: Artifact Engineering

"How Can We ***Systematically Engineer Software Artifacts*** *to Build Better LLM4SE Bots?"*

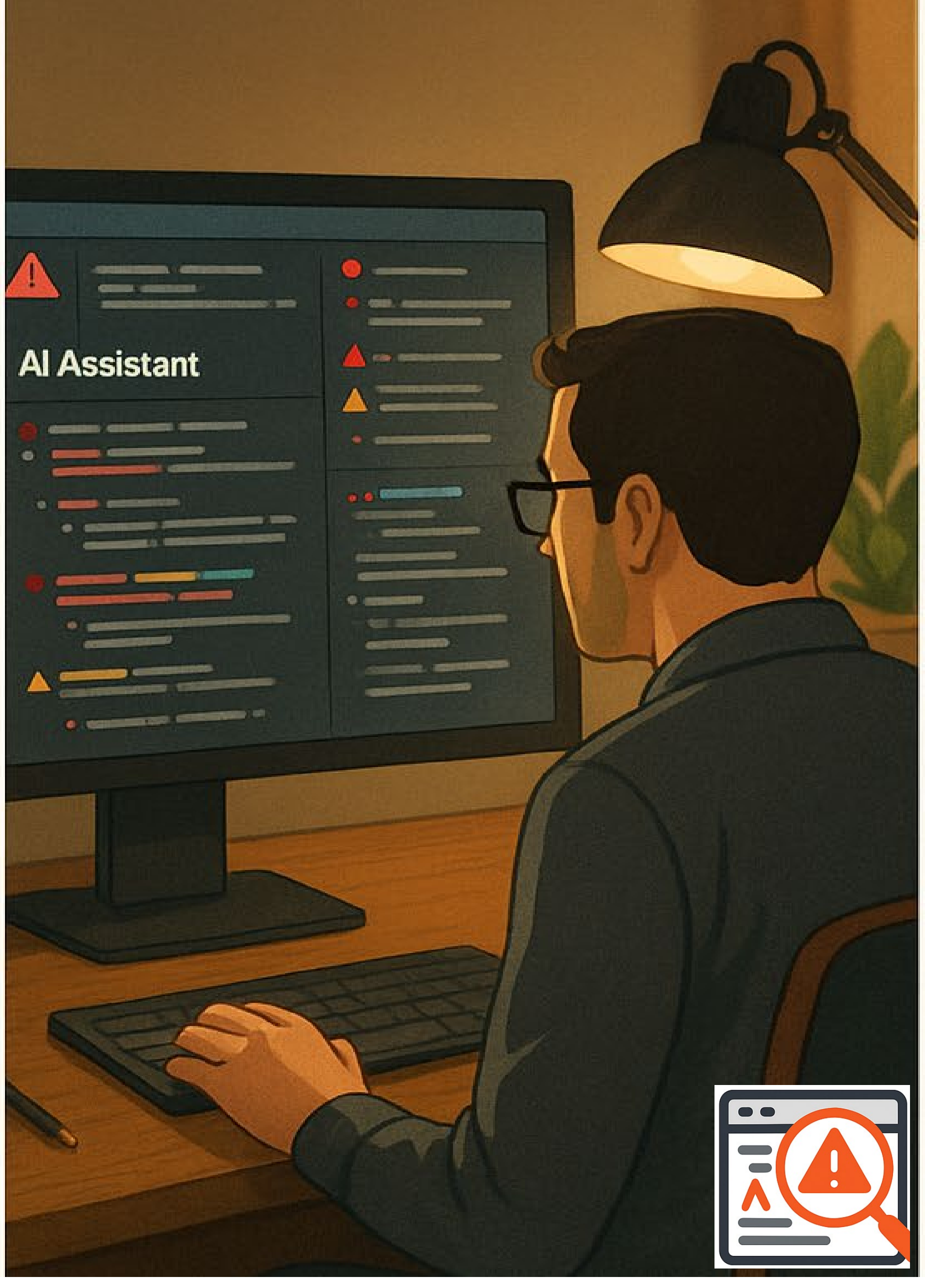


- Construct **LLM-reasoning friendly datasets** by
  - selecting the right artefacts
  - linking related pieces of information,
  - providing examples,
  - transforming them appropriately, etc.

  leveraging **domain knowledge**.
- **Scale up with LLM4(LLM4SE)**: LLMs themselves can assist this *SE artifact engineering (crafting)* step

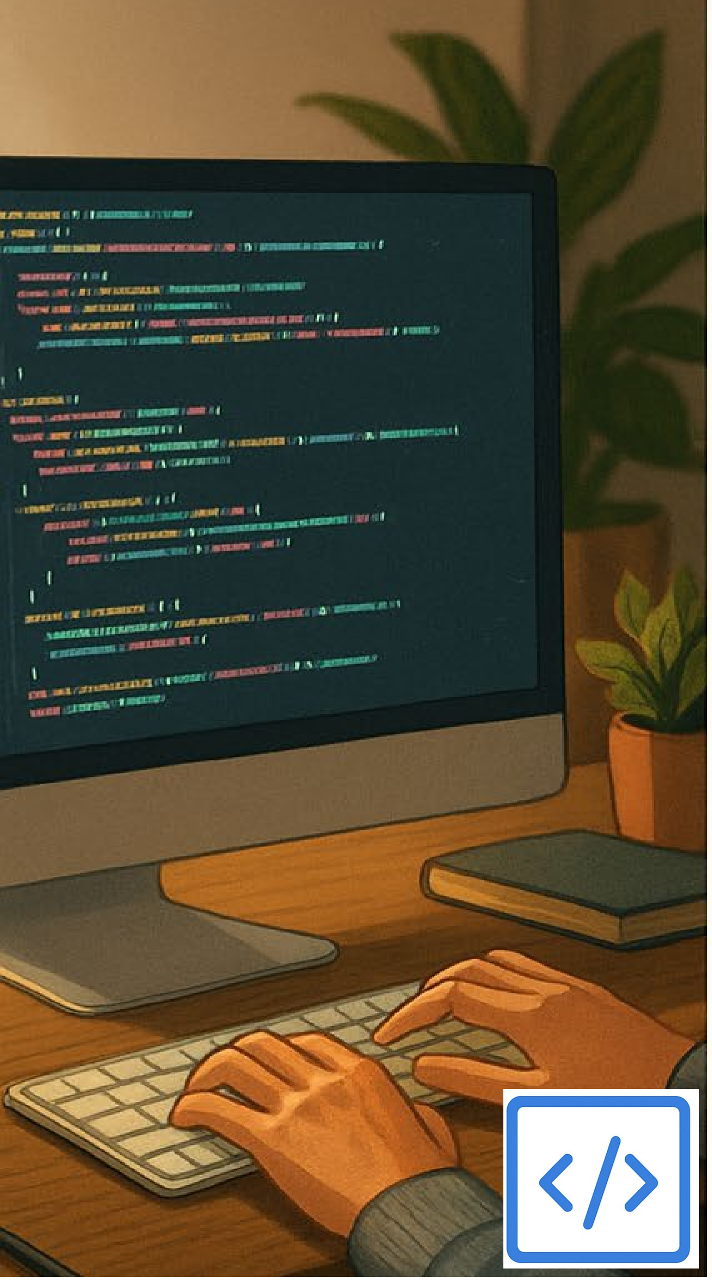


Code

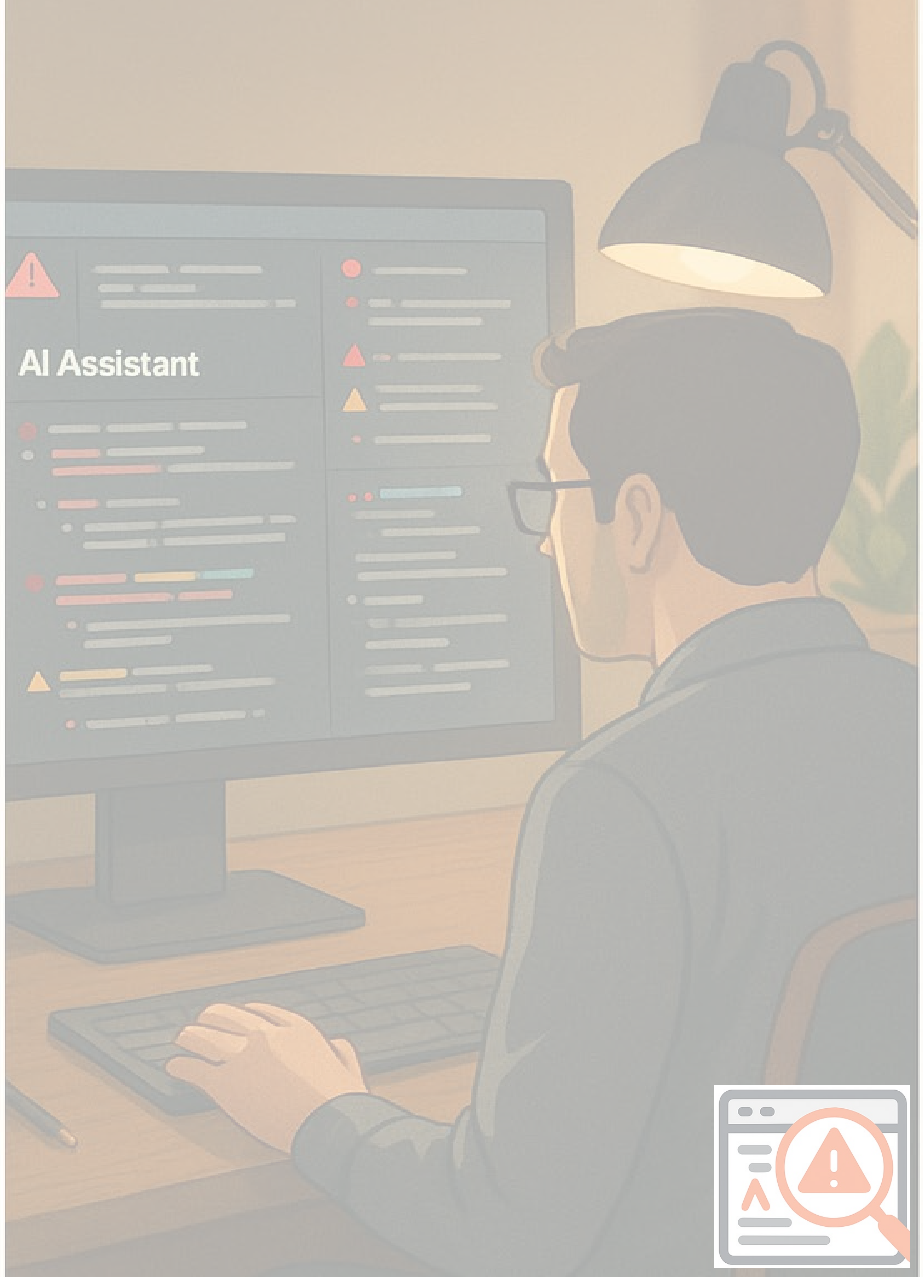


Critique

Cure



Code



Critique

Cure

Structured Data Can Be Helpful for AI4SE

ICPC 2018

Deep Code Comment Generation*

Xing Hu[1] , Ge Li[1], Xin Xia[2], David Lo[3], Zhi Jin[1]
[1]Key Laboratory of High Confidence Software Technologies (Peking University), MoE, Beijing, China
[2]Faculty of Information Technology, Monash University, Australia
[3]School of Information Systems, Singapore Management University, Singapore
[1]{huxing0101,lige,zhijin}@pku.edu.cn, [2]xin.xia@monash.edu, [3] davidlo@smu.edu.sg

*First work that leverages **structural** information in code AST for better code comment generation; most cited paper of ICPC*

Distilling **Structured** Reasoning Traces from Com. Knowledge

ICSE 2026

Think Like Human Developers: Harnessing Community Knowledge for Structured Code Reasoning

Chengran Yang
Singapore Management University
Singapore
cryang@smu.edu.sg

Zhensu Sun
Singapore Management University
Singapore
zssun@smu.edu.sg

Hong Jin Kang
University of Sydney
Australia
hongjin.kang@sydney.edu.au

Jieke Shi
Singapore Management University
Singapore
jiekeshi@smu.edu.sg

David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

*Engineers high-quality **structured** reasoning traces from community knowledge for better LLM-powered code generation*

Community Knowledge as Code Reasoning Training Resources

558: Time 99.4%, Solution with step by step explanation

Marlen

571 Mar 15, 2023

Divide and Conquer | Tree | Python | Python3

Intuition

Approach

1. Check if either quadTree1 or quadTree2 is a leaf node. If quadTree1 is a leaf node return quadTree1 if its value is True, else return quadTree2. If quadTree2 is a leaf node, return quadTree2 if its value is True, else return quadTree1.
2. Recursively check the intersection of each child node of quadTree1 and quadTree2 by calling the intersect function on each of them. Store the return values in variables tl (top-left), tr (top-right), bl (bottom-left), and br (bottom-right).

Community discussions offer peer-reviewed insights into developers' problem-solving strategies

"How Can We Leverage Community Knowledge to Help LLM Reason Better?"

RT-Distiller: From Com. Posts to Structured Reasoning Traces

- Distilling high-quality reasoning traces (RTs) for code generation from evolving community knowledge.

❌ Mixed Quality Posts

LeetCode Problem:

Given two binary grids as quad trees, return their logical OR result as a quad tree while maintaining minimal node representation.

User Discussion:

 # Intuition

 # Approach

1.Check if either quadTree1 or quadTree2 is a leaf node...
2.Recursively check the intersection of each child node of quadTree1and quadTree2 by calling the intersect function...
3.Check if all four child nodes (tl, tr, bl, br) are leaves and have the same value...
4.If any of the child nodes have a different value or are not leaf nodes, create a new internal node...

 # Complexity

Time complexity: ...

 # Code

```
class Solution:
    def intersect ...
```

RT-Distiller: From Com. Posts to Structured Reasoning Traces

- Distilling high-quality reasoning traces (RTs) for code generation from evolving community knowledge.

❌ Mixed Quality Posts

❌ Artificial
Site-Specific
Code Structure

LeetCode Problem:

Given two binary grids as quad trees, return their logical OR result as a quad tree while maintaining minimal node representation.

User Discussion:

 # Intuition

 # Approach

1.Check if either quadTree1 or quadTree2 is a leaf node...
2.Recursively check the intersection of each child node of quadTree1and quadTree2 by calling the intersect function...
3.Check if all four child nodes (tl, tr, bl, br) are leaves and have the same value...
4.If any of the child nodes have a different value or are not leaf nodes, create a new internal node...

 # Complexity

Time complexity: ...

 # Code

```
class Solution:
    def intersect ...
```

RT-Distiller: From Com. Posts to Structured Reasoning Traces

- Distilling high-quality reasoning traces (RTs) for code generation from evolving community knowledge.

❌ Mixed Quality Posts

❌ Artificial Site-Specific Code Structure

❌ Loosely Structured Reasoning

LeetCode Problem:

Given two binary grids as quad trees, return their logical OR result as a quad tree while maintaining minimal node representation.

User Discussion:

 # Intuition

 # Approach

1.Check if either quadTree1 or quadTree2 is a leaf node...
2.Recursively check the intersection of each child node of quadTree1and quadTree2 by calling the intersect function...
3.Check if all four child nodes (tl, tr, bl, br) are leaves and have the same value...
4.If any of the child nodes have a different value or are not leaf nodes, create a new internal node...

 # Complexity

Time complexity: ...

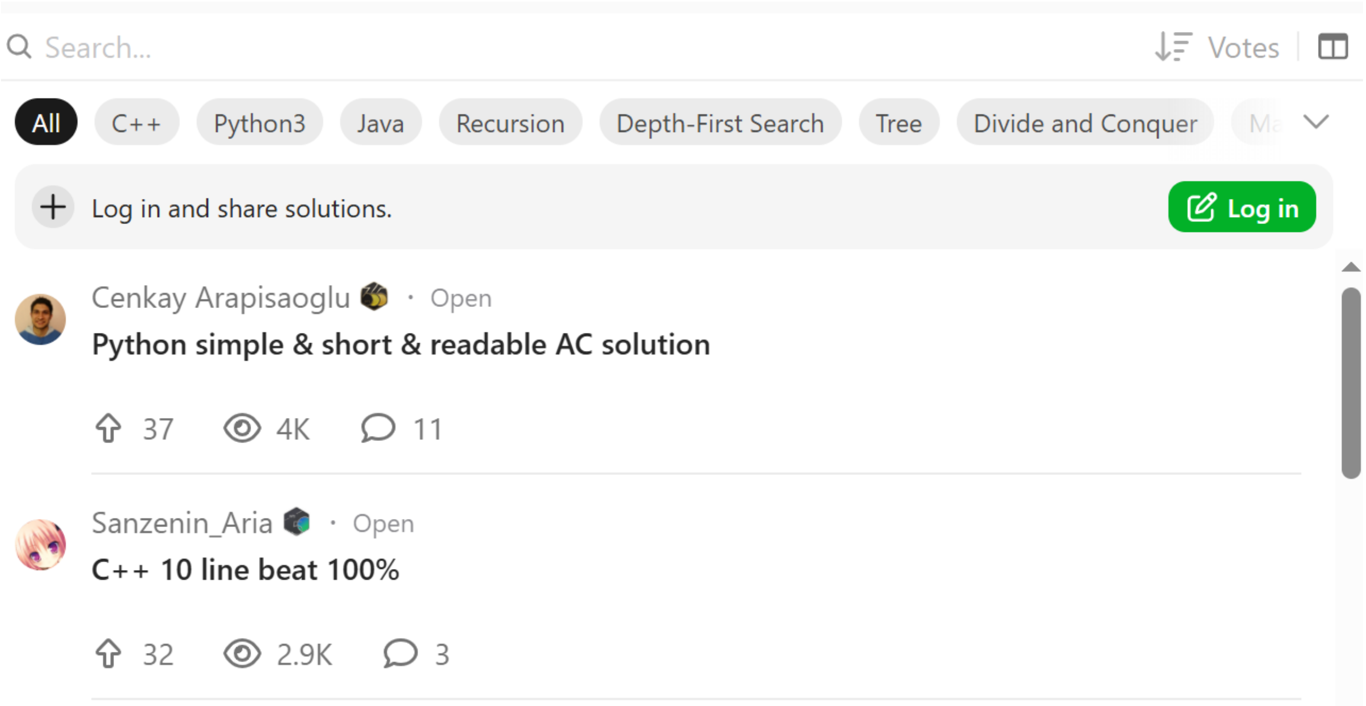
 # Code

```
class Solution:
    def intersect ...
```

Step I: Addressing Mixed Quality Posts

✅ Identify High Quality Posts

- Identify rating mechanisms
- Select highly-rated posts



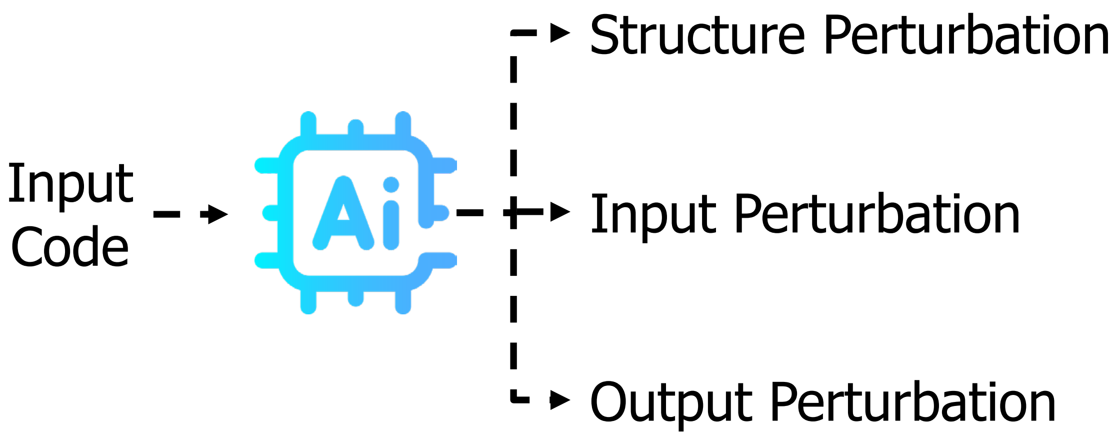
✅ Improve Post Quality

- Identify typical information pieces
- Detect missing pieces
- Synthesize missing pieces using LLM

+

Step II: Dealing with Artificial Site-Specific Code Structure

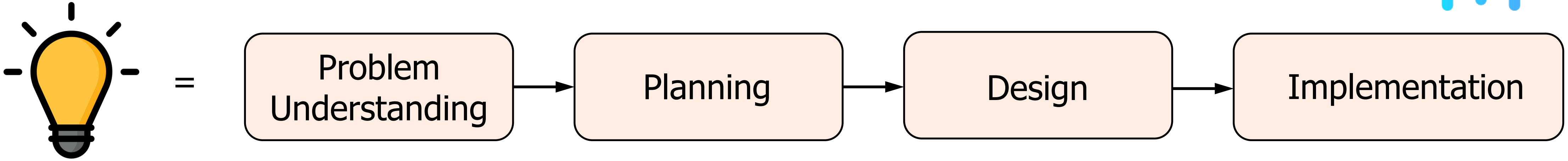
✅ Diversify Solution Code While Preserving Semantics



Category	**Variation Type**
Input Format	Direct input (e.g., variables, constants) Structured input (e.g., list, tuple, dictionary) Batch input Interactive input() Boolean input Command-Line Arguments
Output Format	Structured input (e.g., list, tuple, dictionary) Boolean output Batch output Print statement Logging Output
Structural Variations	Class-based Function-based Script-based

Step III: Enriching Loosely Structured Reasoning Traces

- ✅ Rewrite a post into a structured trace inspired by SDLC



Step III: Enriching Loosely Structured Reasoning Traces

Given two binary grids as quad trees, return their logical OR result as a quad tree while maintaining minimal node representation.

User Discussion:

 # Intuition

 # Approach

1.Check if either quadTree1 or quadTree2 is a leaf node...

2.Recursively check the intersection of each child node of

Problem Understanding

First things first, I need to understand what a quad-tree is. From the description, it's a tree data structure where each internal node has exactly four children ...

Planning

Let's think about this step by step. First, I need to understand how the quad-tree represents the matrix. If a node is a leaf and its `val` is true, it means ...

Design

Let me try to outline an approach:
1. If `quadTree1` is a leaf:
  - If its `val` is true, then the entire grid is 1's, so the OR with anything else is still 1's ...

Implementation

Let me try to write some solutions:

```
function intersect(quadTree1, quadTree2):
if quadTree1.isLeaf:
    if quadTree1.val:
return quadTree1
...
```

Step III: Enriching Loosely Structured Reasoning Traces

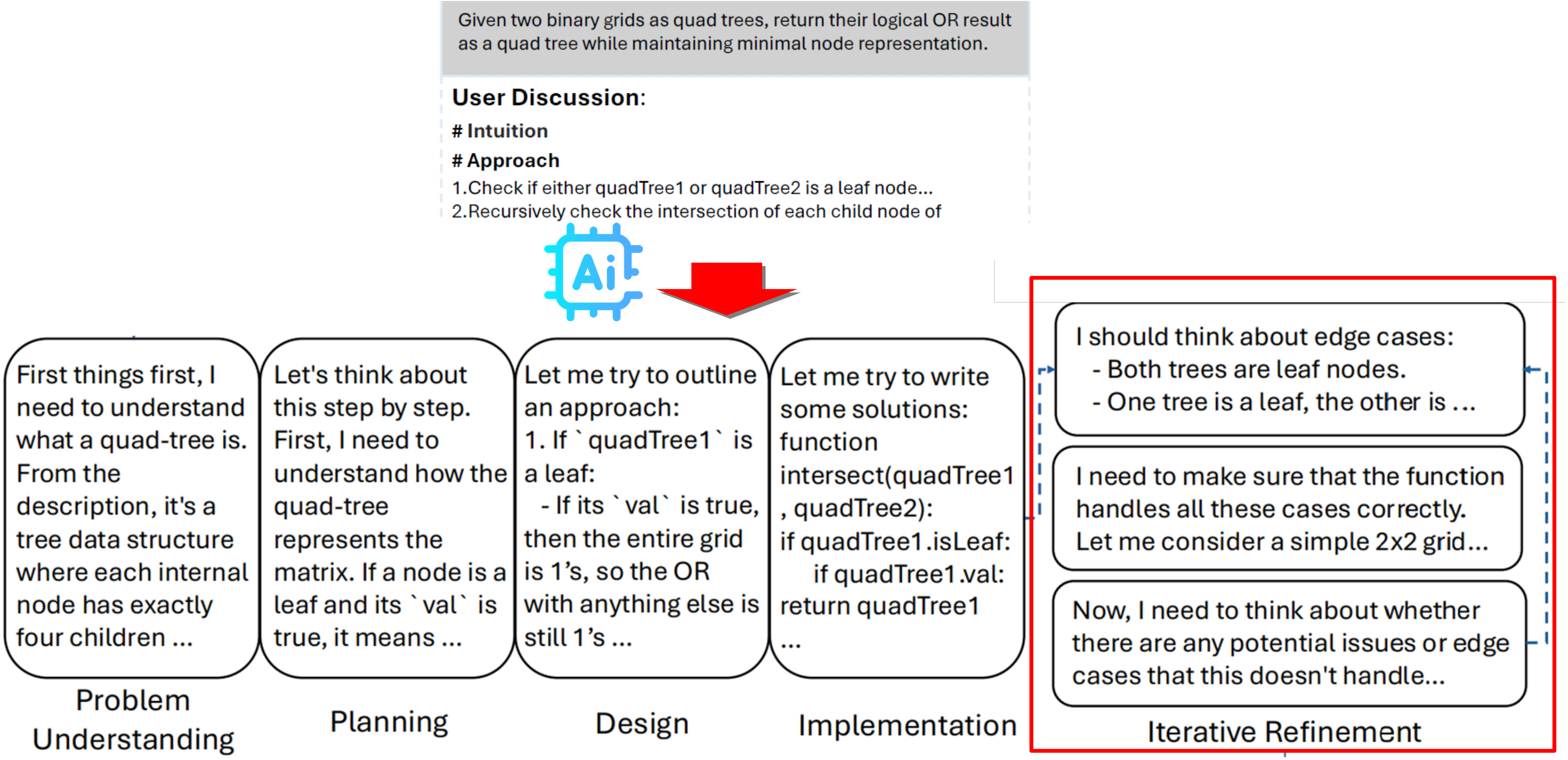
- ✅ Rewrite a post into a structured trace inspired by SDLC

= Problem Understanding → Planning → Design → Implementation

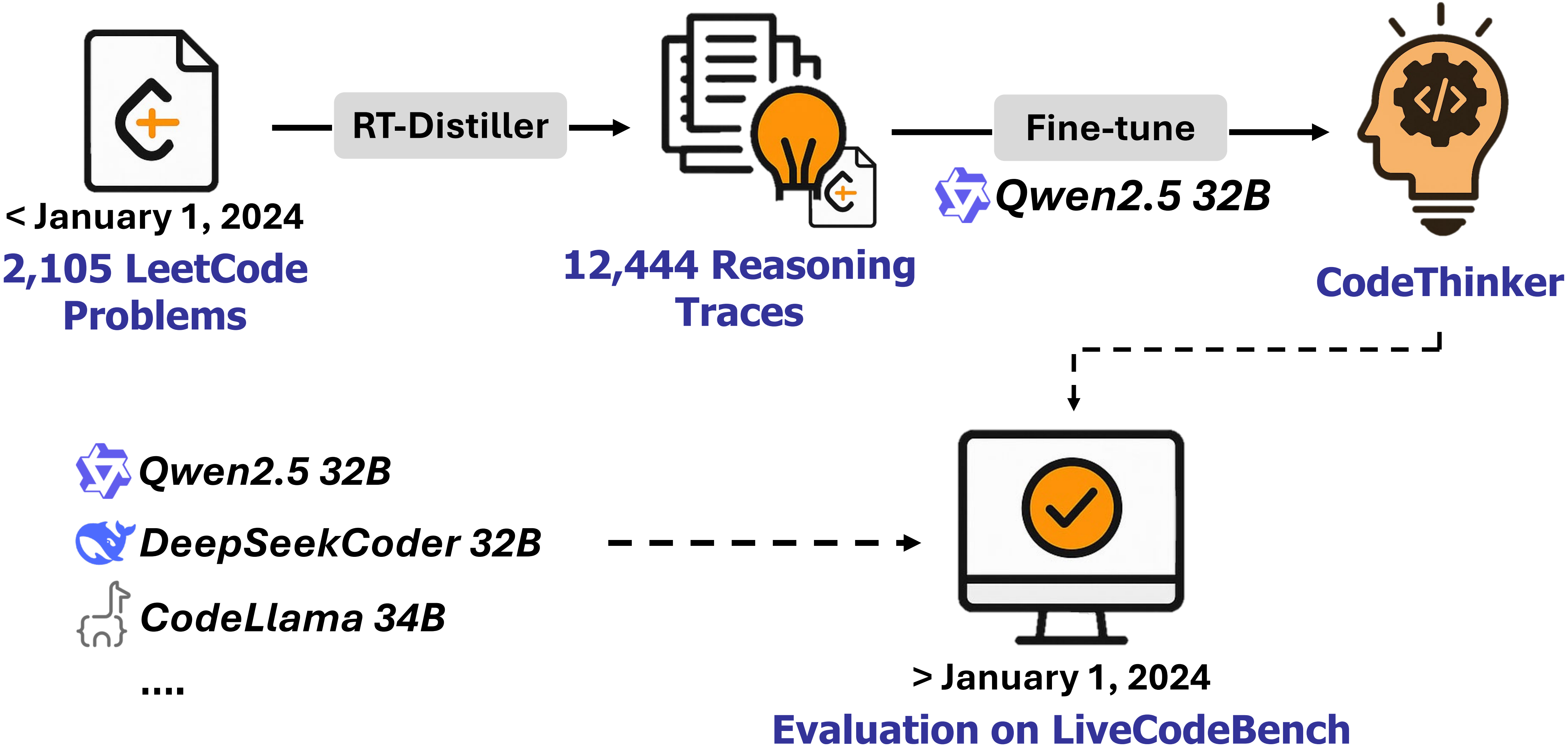
- ✅ Add iterative refinement considerations

+ Simple Input Reasoning + Common Mistake Reasoning + Edge Case Reasoning

Step III: Enriching Loosely Structured Reasoning Traces



Experiment Settings

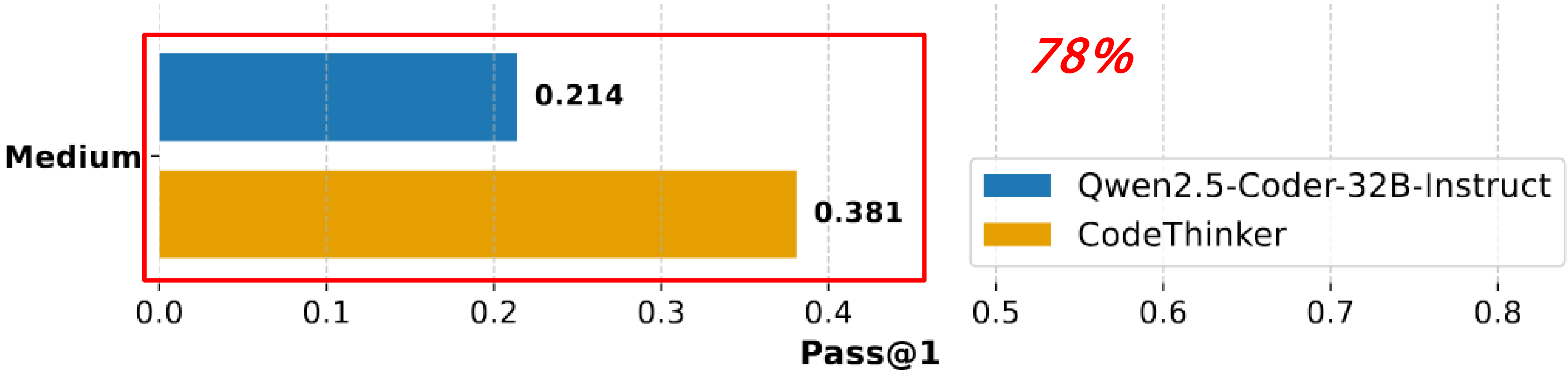


Experiment Results – Comparison with LLMs of Similar Sizes

Model	Prompting	Size	Easy pass@1	Medium pass@1	Hard pass@1	Overall pass@1
Open-source LLMs						
CodeLlama-Instruct	Standard	34B	0.290	0.028	0	0.078
CodeLlama-Instruct	CoT	34B	0.298	0.028	0	0.106
s1-32B	Standard	32B	0.306	0.098	0	0.134
s1-32B	CoT	32B	0.315	0.112	0	0.142
DeepSeekCoder	Standard	32B	0.589	0.126	0	0.235
DeepSeekCoder	CoT	32B	0.565	0.133	0	0.233
Qwen2.5-Instruct	Standard	32B	0.847	0.343	0	0.398
Qwen2.5-Instruct	CoT	32B	0.823	0.371	0	0.401
CodeThinker	CodeThinker Style	32B	0.831 (↓1.89%)	**0.490**(↑42.86%)	0	**0.447**(↑12.31%)

Experiment Results – Generalizability Beyond LeetCode

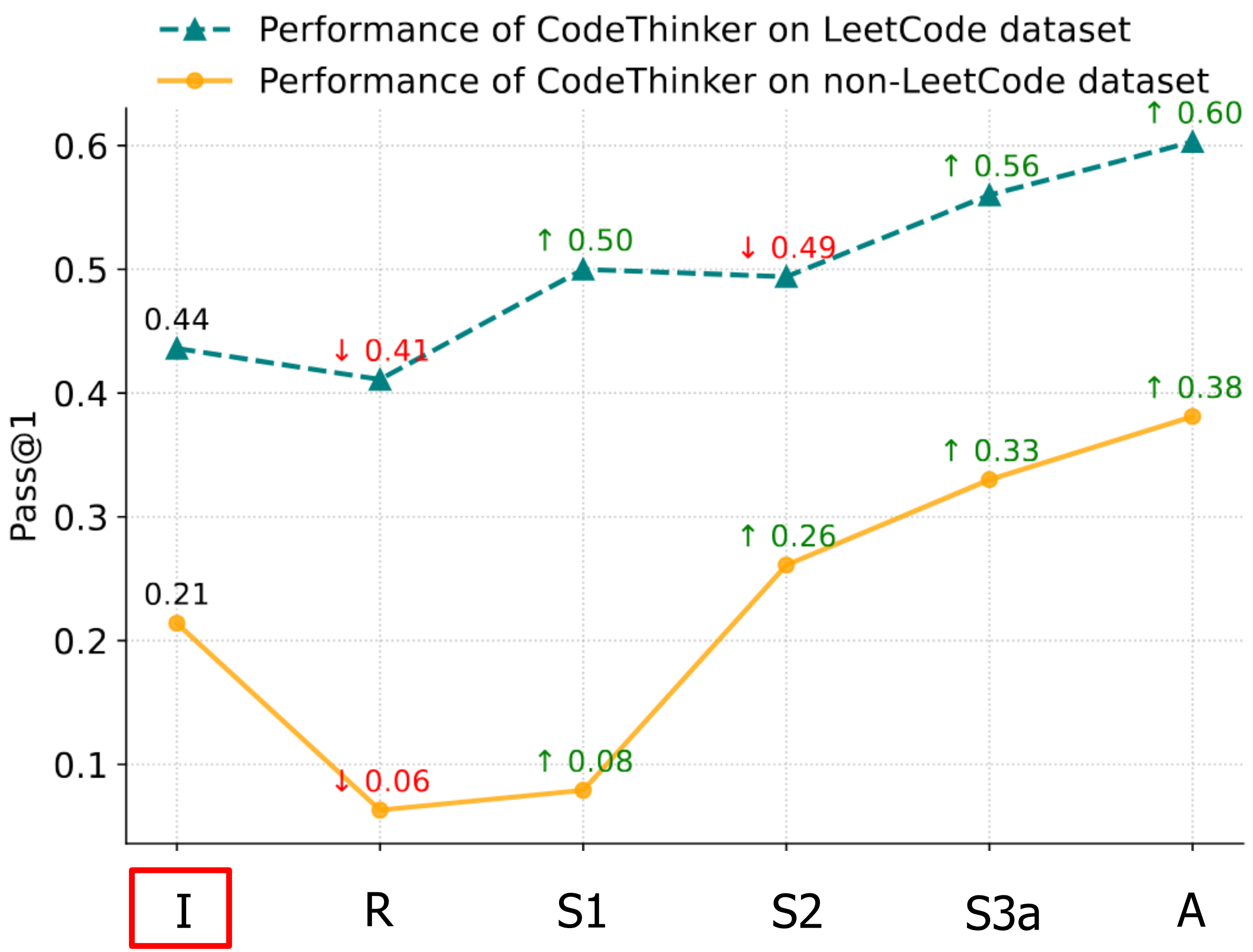
Figure 3: Comparison of CodeThinker and its base model on non-LeetCode coding problems (AtCoder and Codeforces), highlighting improved reasoning performance on medium-difficulty tasks.



CodeThinker shows **strong generalization ability** to problems from **non-LeetCode platforms**, achieving significant improvements

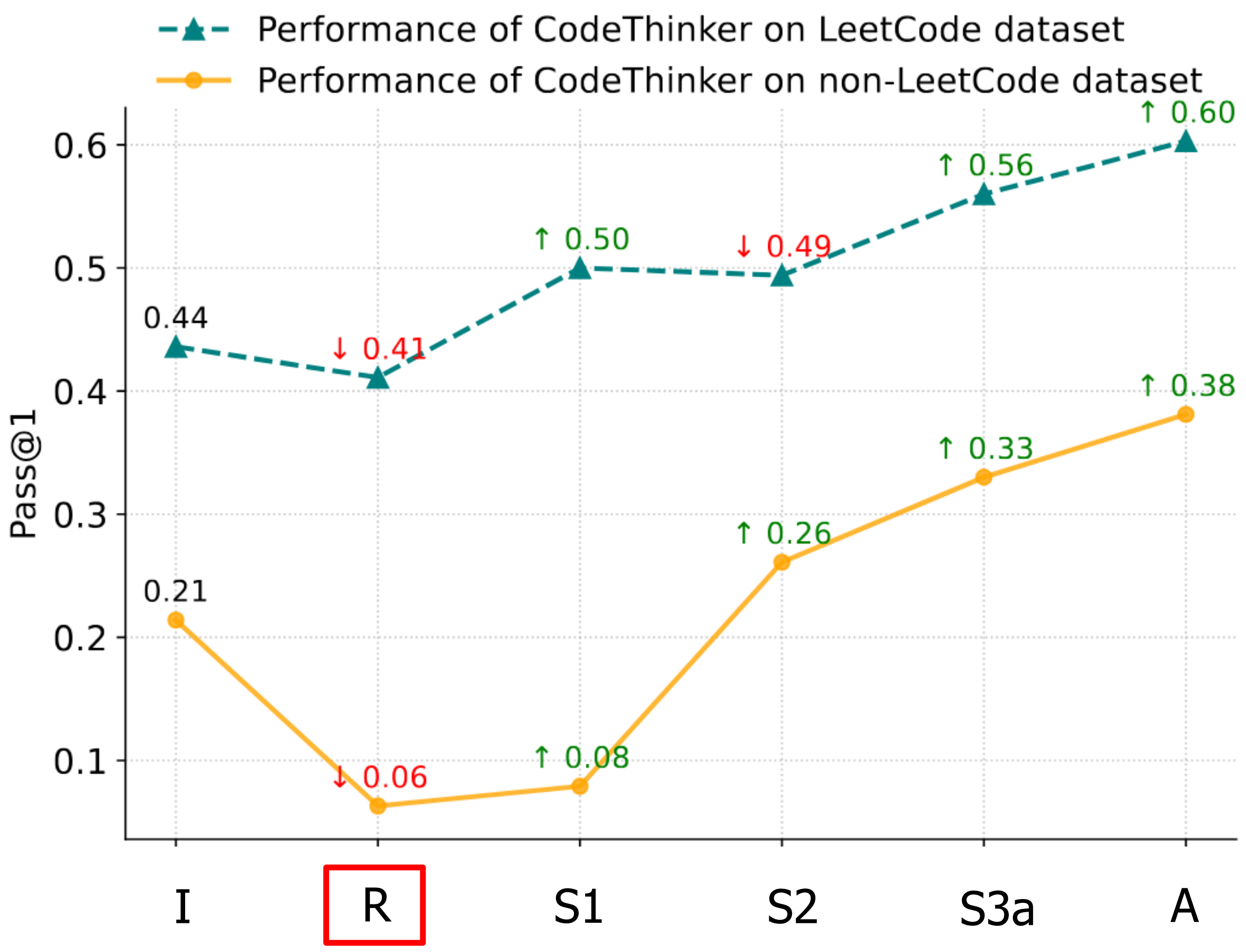
Experiment Results – Ablation Study

*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



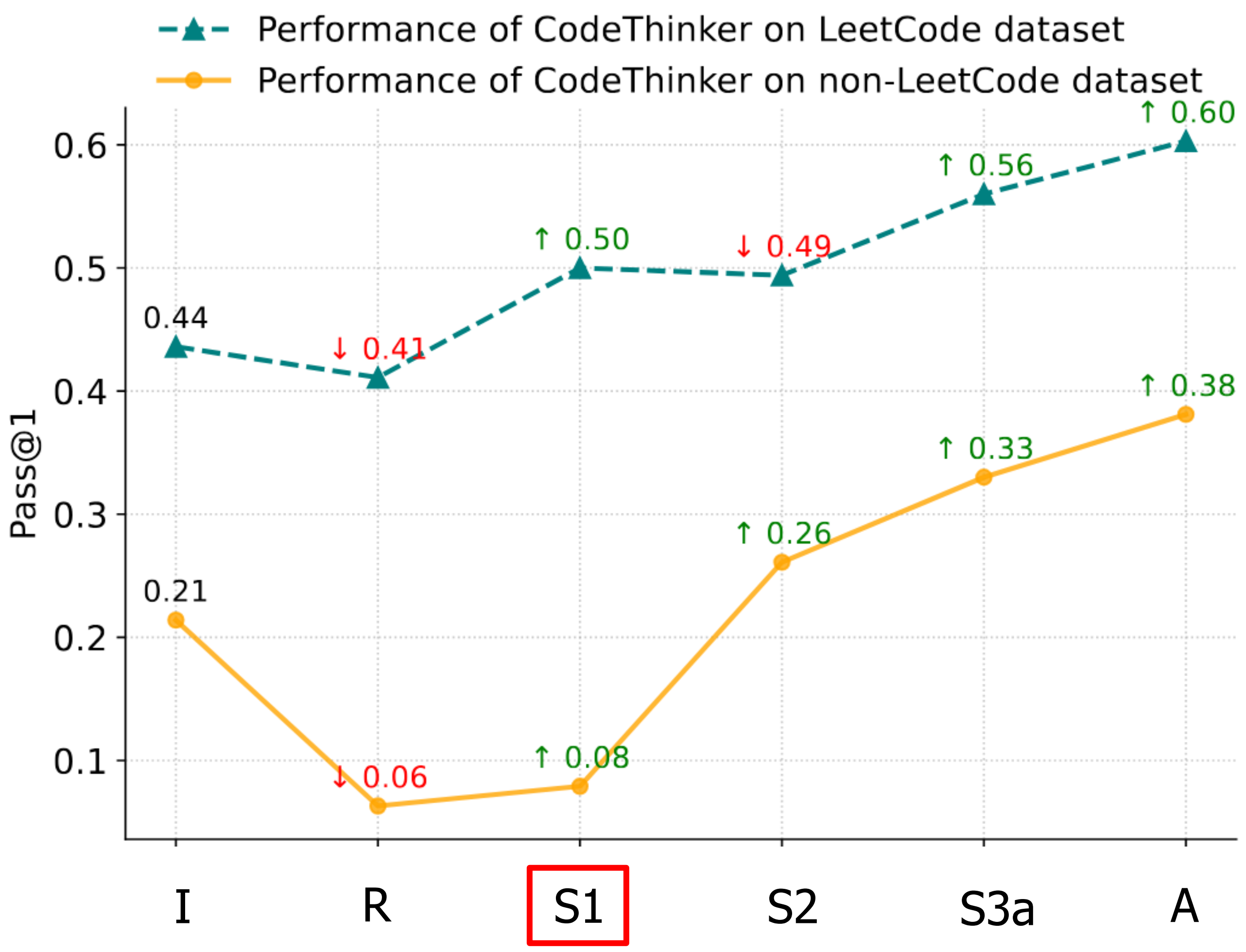
Experiment Results – Ablation Study

*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



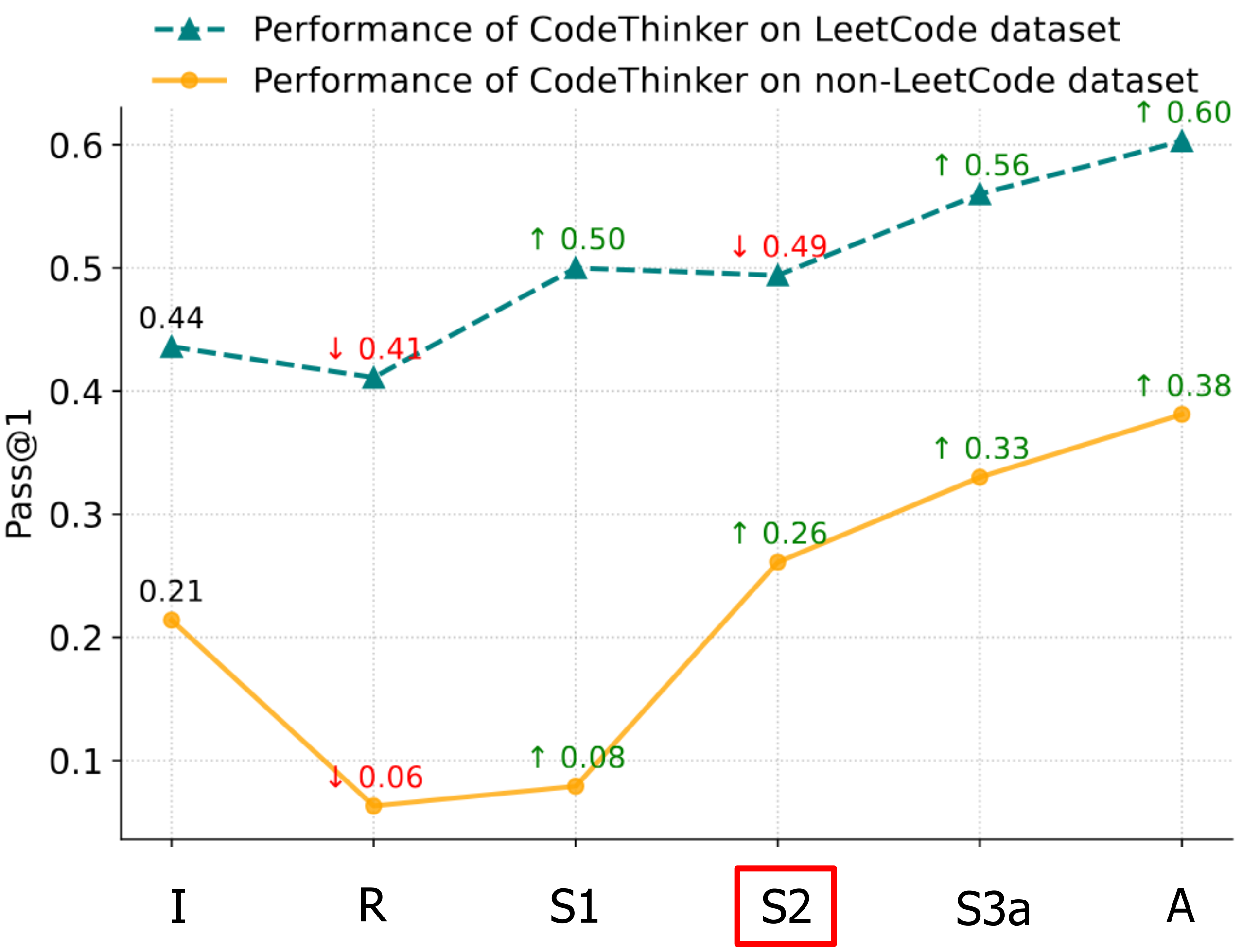
Experiment Results – Ablation Study

*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



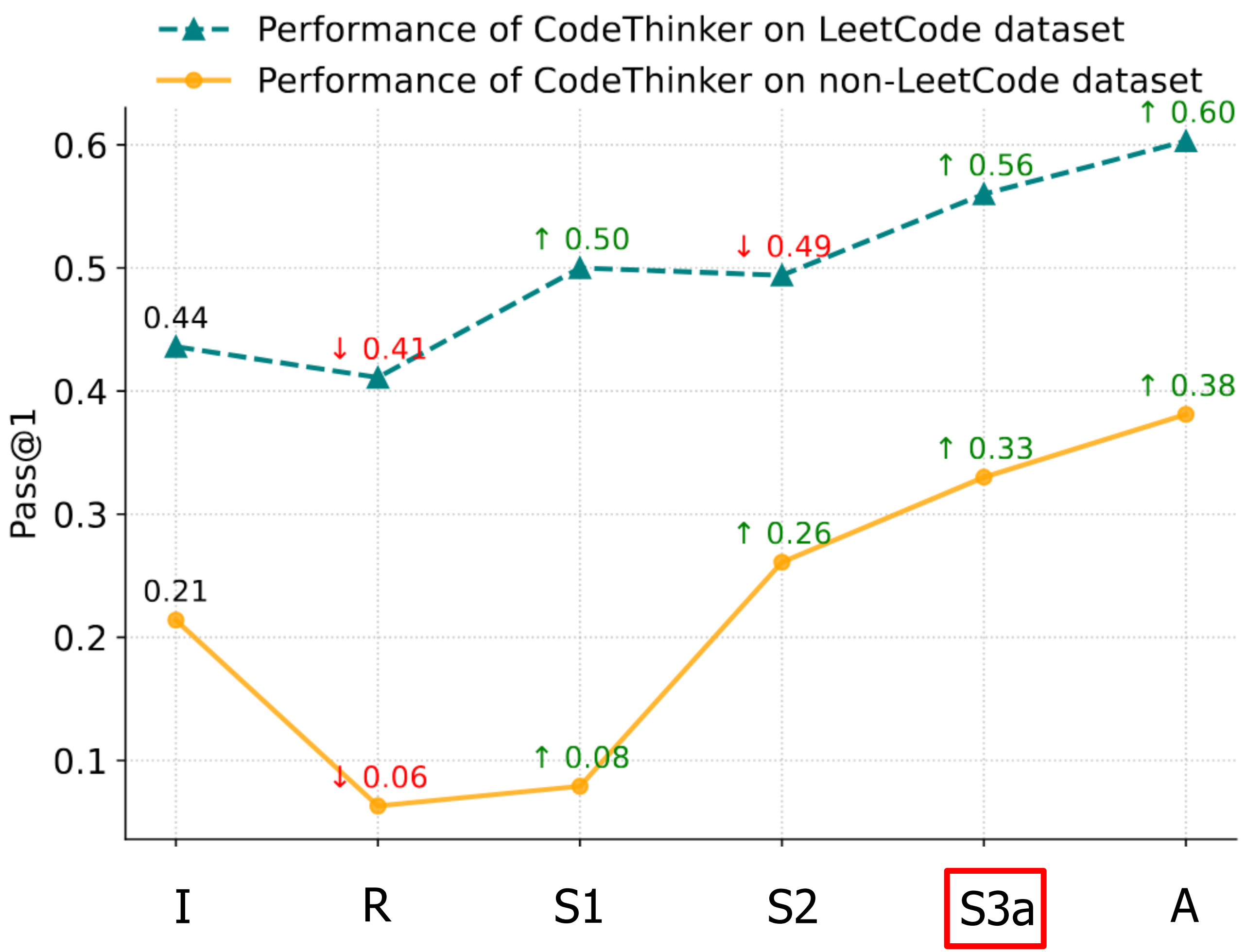
Experiment Results – Ablation Study

*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



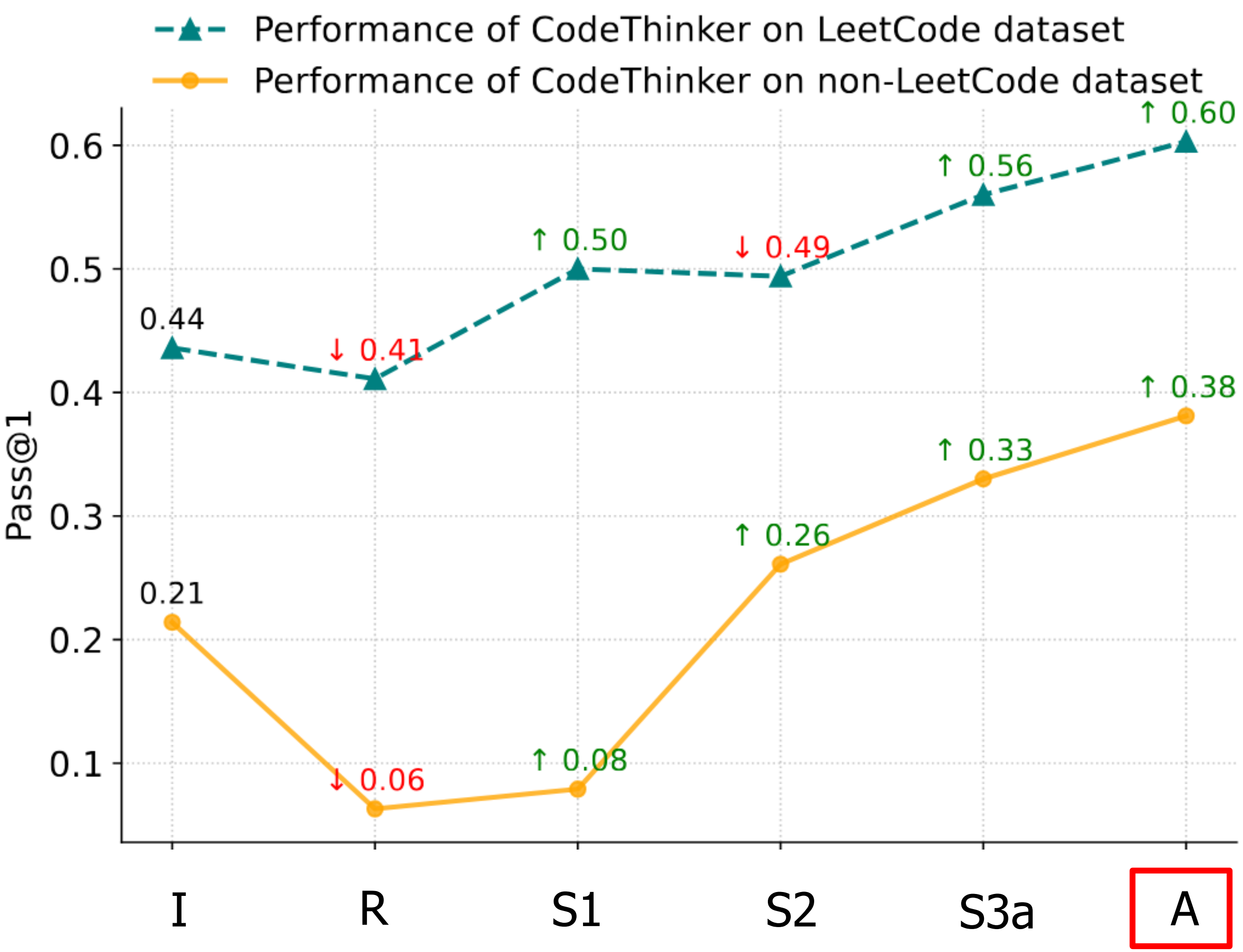
Experiment Results – Ablation Study

*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



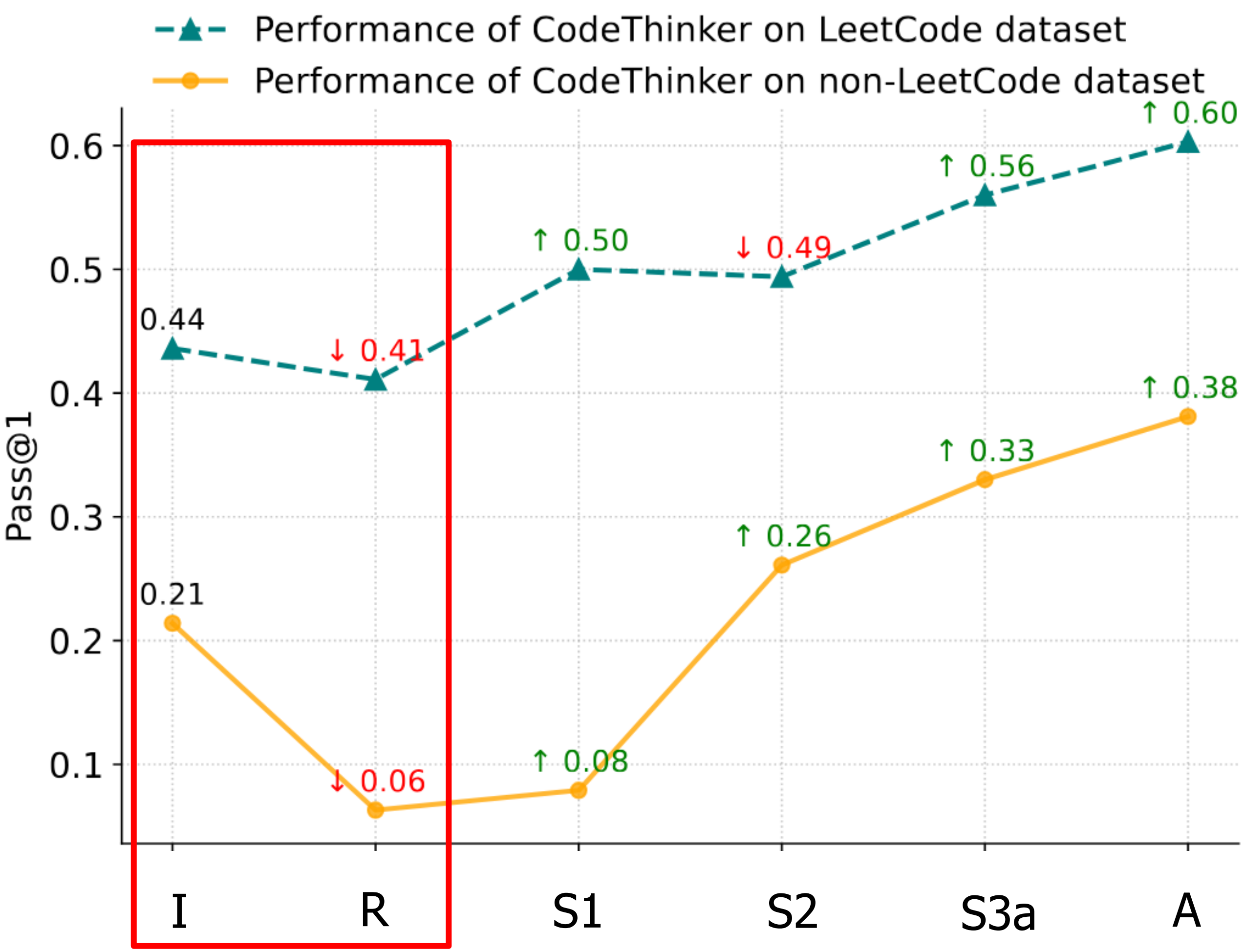
Experiment Results – Ablation Study

*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



Experiment Results – Ablation Study

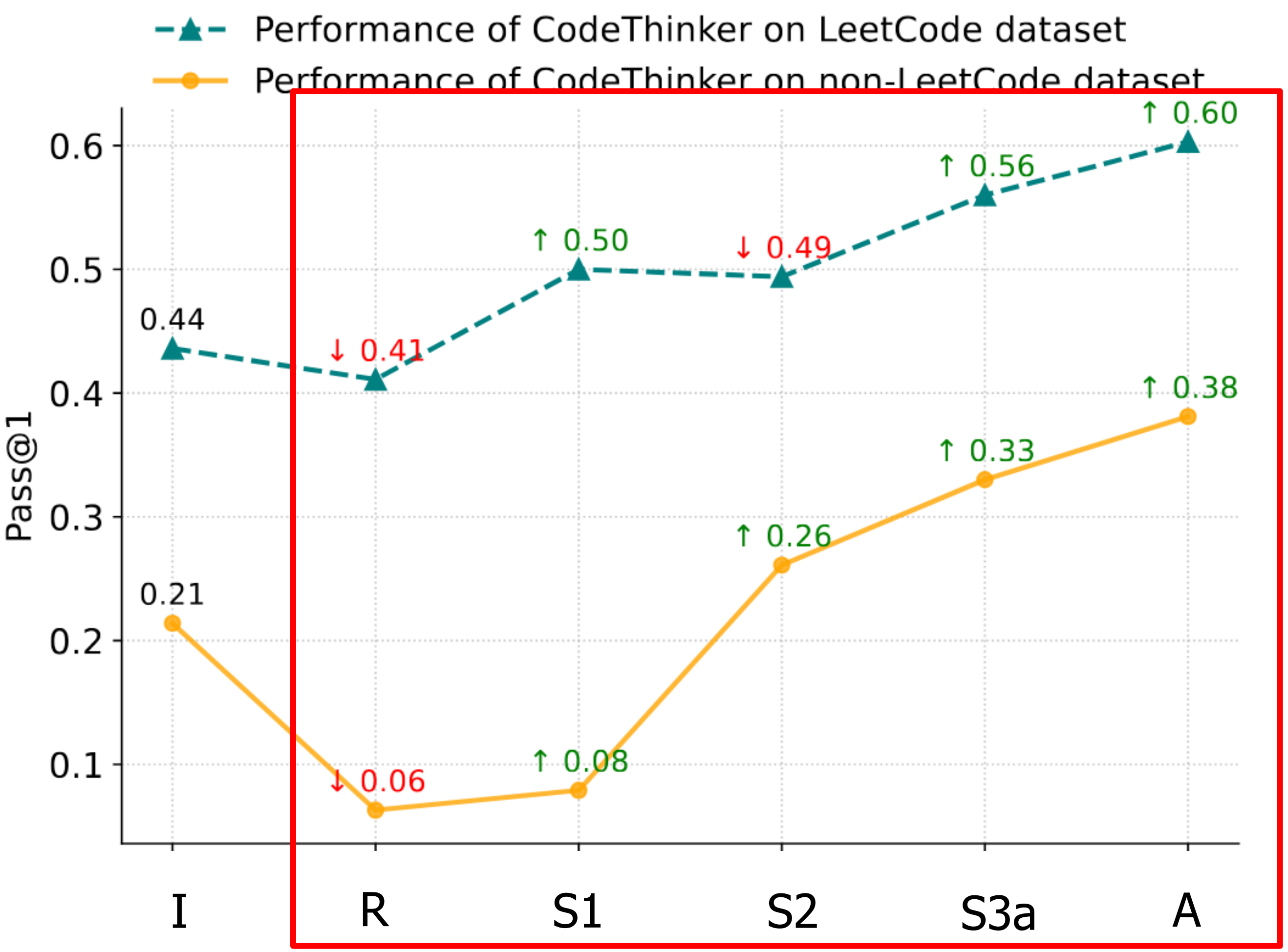
*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



I vs. R: Training with raw LeetCode discussion posts results in performance drop

Experiment Results – Ablation Study

*I: Initial checkpoint; R: Fine tune using raw LeetCode discussion posts;
S1: Step 1 only; S2: Step 1+2 only; S3a: Step 1+2+3a only; A: All steps of RT-Distiller



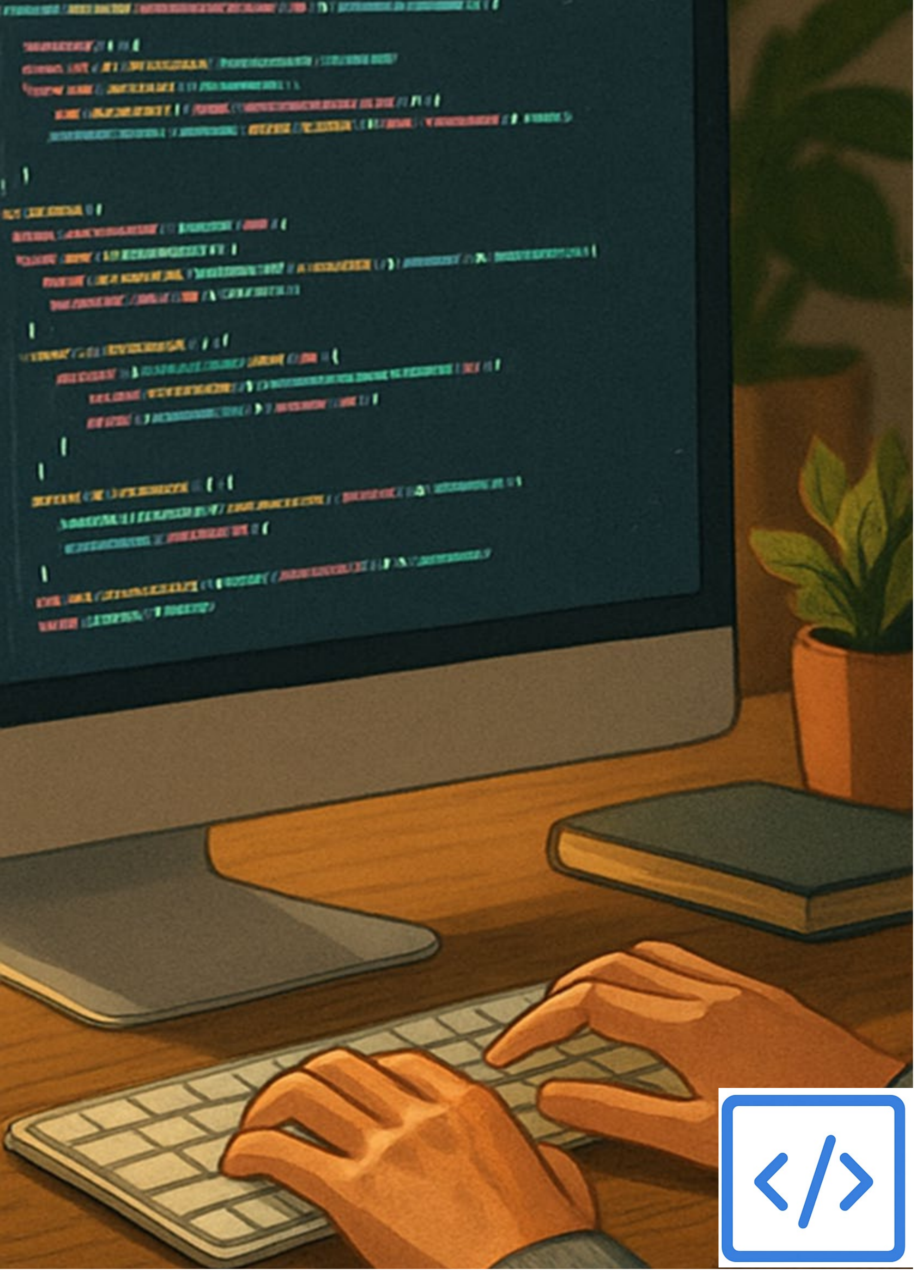
I vs. R: Training with raw LeetCode discussion posts results in performance drop

R vs. S1: Synthesizing missing information in LeetCode discussion posts helps

S1 vs. S2: Applying code perturbation improves generalization to non-LeetCode

S2 vs. S3a: Introducing SDLC-inspired structure to reasoning traces helps

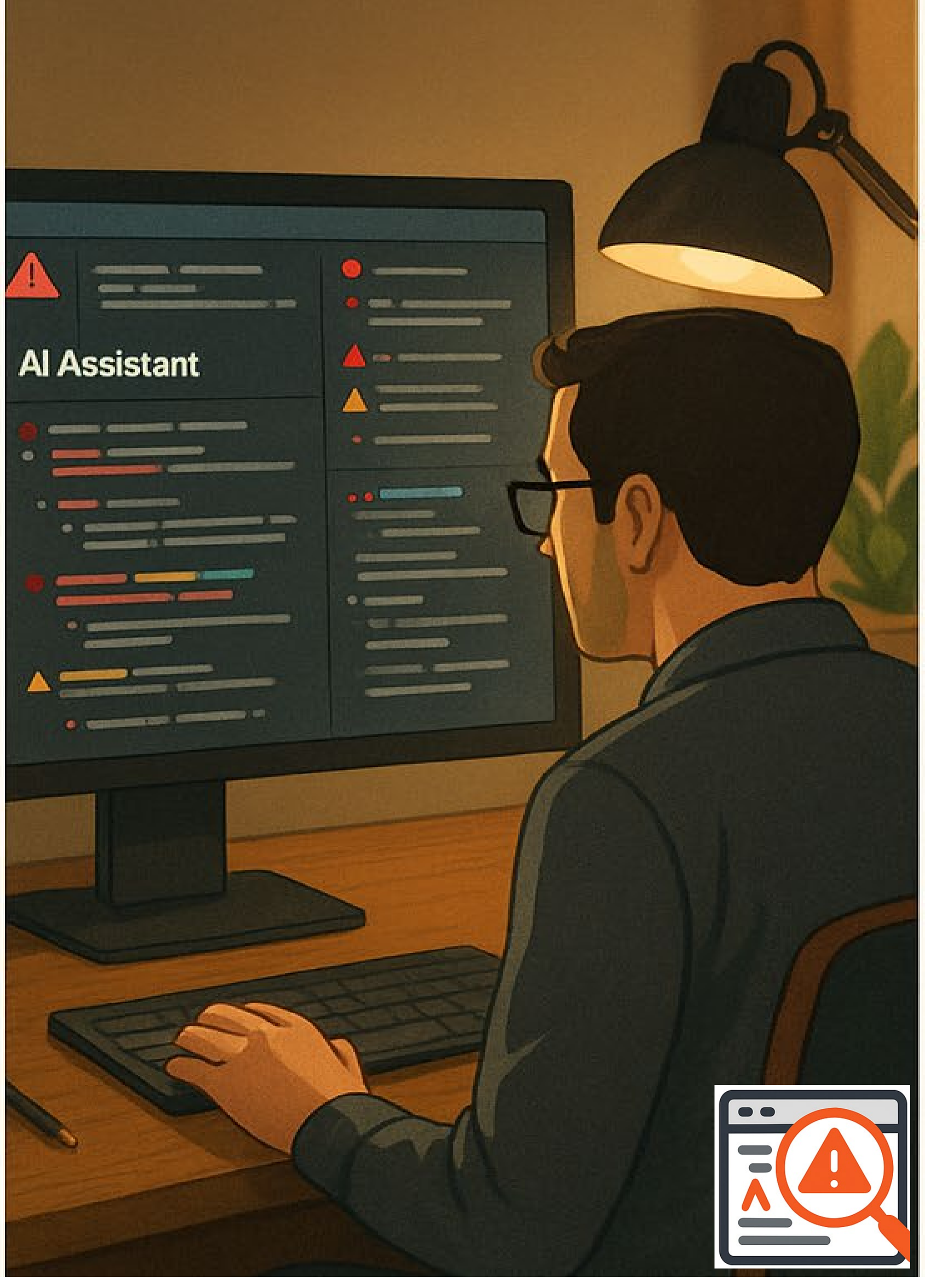
S3a vs. A: Adding iterative refinement considerations helps



Code

Engineering **structured reasoning traces** help LLM reasoning for code generation

Code



Critique

Cure

Contrastive Patterns Can Help AI Reasoning

KDD 2009

Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach

David Lo
Singapore Management University
davidlo@smu.edu.sg

Hong Cheng*
Chinese University of Hong Kong
hcheng@se.cuhk.edu.hk

Jiawei Han
University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

Siau-Cheng Khoo and Chengnian Sun
National University of Singapore
{khoosc,suncn}@comp.nus.edu.sg

*Distills **rich discriminative (contrastive) patterns** for AI-powered failure detection (aka. test oracle generation); early SE paper in a top AI venue*

Distilling **Contrastive** Reasoning Trace Pairs

R2VUL: Learning to Reason about Software Vulnerabilities with Reinforcement Learning and Structured Reasoning Distillation

Martin Weyssow[1*], Chengran Yang[1], Junkai Chen[1], Ratnadira Widyasari[1], Ting Zhang[1], Huihui Huang[1], Huu Hung Nguyen[1], Yan Naing Tun[1], Tan Bui[1], Yikun Li[1], Ang Han Wei[2], Frank Liauw[2], Eng Lieh Ouh[1], Lwin Khin Shar[1], David Lo[1]

[1]Singapore Management University
[2]GovTech Singapore

*Engineers **contrastive reasoning trace pairs***
to help LLM reason about vulnerabilities

Vulnerability Detection with LLMs

- Vulnerability detection demands a *binary judgement:*
  - Is the input code **safe** or **vulnerable**?

TOSEM 2025

Large Language Model for Vulnerability Detection and Repair: Literature Review and the Road Ahead

XIN ZHOU, School of Computing and Information Systems, Singapore Management University, Singapore, Singapore
SICONG CAO and XIAOBING SUN, Yangzhou University, Yangzhou, China
DAVID LO, School of Computing and Information Systems, Singapore Management University, Singapore, Singapore

Vulnerability Detection with LLMs

- Limited efficacy

ICSE 2024

Large Language Model for Vulnerability Detection: Emerging Results and Future Directions

Xin Zhou, Ting Zhang, and David Lo
School of Computing and Information Systems, Singapore Management University, Singapore
{xinzhou.2020,tingzhang.2019}@phdcs.smu.edu.sg,davidlo@smu.edu.sg

- From a developer viewpoint, a binary judgement is also not sufficient:

Why is this input code vulnerable or safe?
What is the mechanism of a vulnerability and its **impact**?

R2Vul: Beyond Binary Judgement

Fine-tune an LLM to **better detect vulnerabilities** and generate **structured reasoning**.

- The LLM outputs a structured reasoning covering key aspects of code safety and vulnerability.

Vulnerable Code

1. Discuss specific code constructs responsible for the vulnerability
2. Explain the mechanism of the vulnerability
3. Discuss its potential impact
4. Relate the vulnerability to a relevant CWE

Safe Code

1. Discuss key aspects contributing to code safety
2. Discuss the absence of key vulnerabilities
3. Provide evidence-based justification why a code is safe

Step 1: Generate Contrastive Reasoning Trace Pairs

- Given a labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$
  - (x_i: function, $y_i \in \{Vuln, Safe\}$)
- Prompt a base LLM to generate:
  - A **valid** reasoning r_i^+ conditioned on the *true* sample label y_i
  - A **flawed** reasoning r_i^- conditioned on the *flipped* sample label $-y_i$

→ $\mathcal{D} = \{(x_i, y_i, r_i^+, r_i^-)\}_{i=1}^{N}$ = preference dataset

Step 1: Generate Contrastive Reasoning Trace Pairs

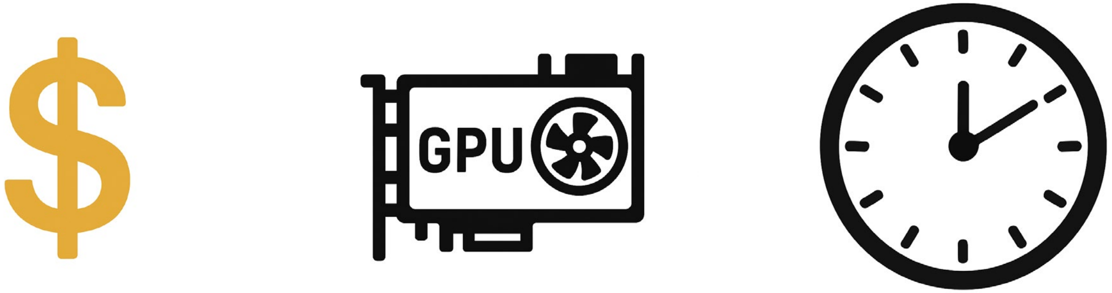
The following function has been flagged as **vulnerable**.
`{ lang }
{ function }
`
This function contains a vulnerability associated with the following CWE(s):
{ cwe_list }.
Specifically, it is linked to { cve_id }, which is described as follows: { cve_description }
Given this information, generate a detailed and coherent thought process ...
1. **Specific Code Constructs**: Identify parts [...]
2. **Mechanism of the Vulnerability**: Explain how [...]
3. **Potential Impact**: Describe the consequences [,,,]
4. **Contextual Relevance**: Relate to CWE(s) and CVE [...]

The following function has been flagged as **non-vulnerable**.
`{ lang }
{ function }
`
This function has been reviewed and determined to not contain any known vulnerabilities.
Given this information, generate a detailed and coherent thought process ...
1. **Analysis of Code Safety**: [...]
2. **Absence of Common Vulnerabilities**: [...]
3. **Validation of the Non-Vulnerable Label**: [...]

Step 2: Use Contrastive Reasoning Trace Pairs

- Combines supervised fine-tuning (SFT) and odds-ratio (OR) losses
  - → SFT: trains the model to generate high-quality structured reasoning
  - → **OR: distinguish valid vs. flawed reasoning**

- Additional feature: Distill **smaller** models
  - → constraints from deployment environment

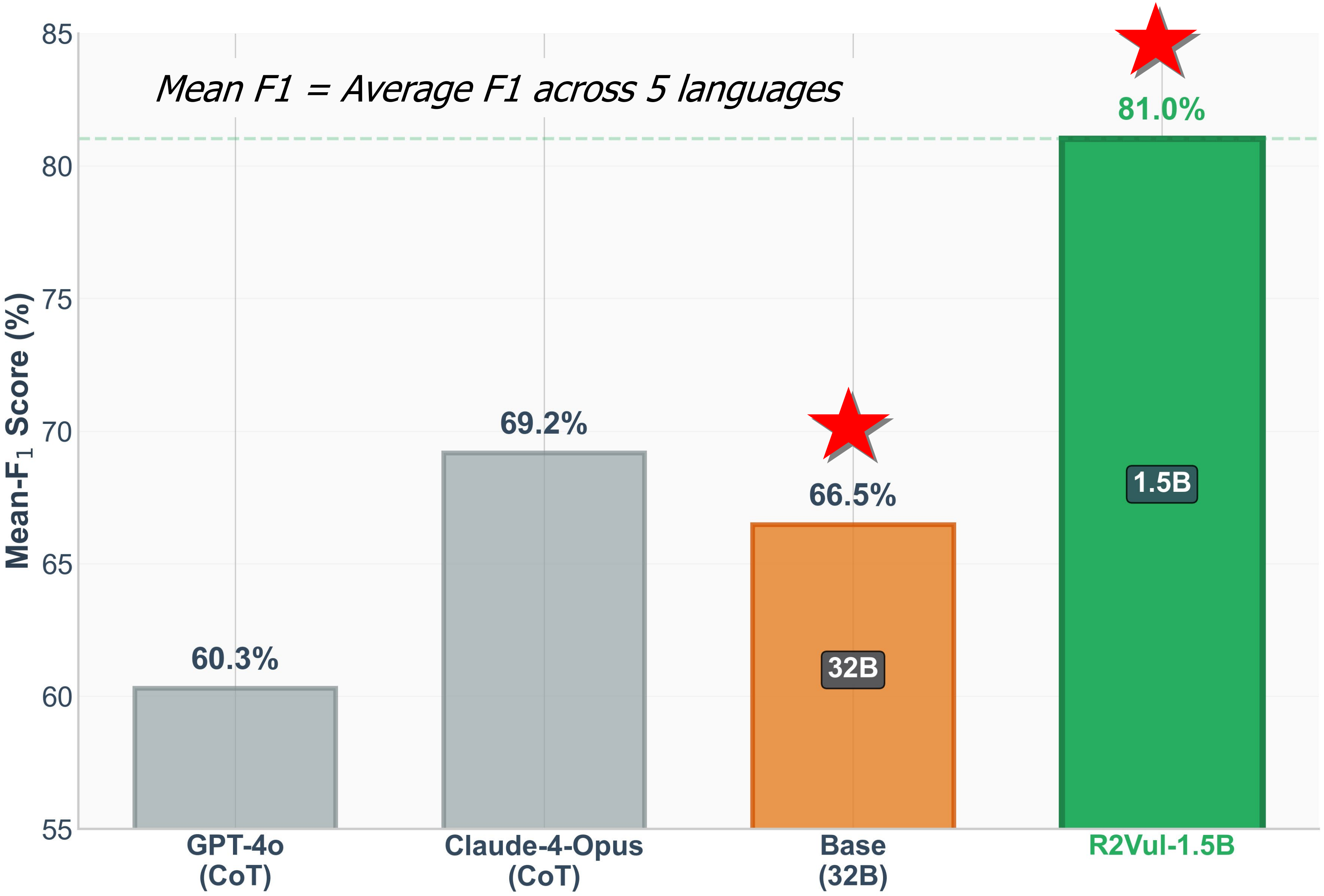


Experiment Details

- Step 1:
  - Base LLM: Qwen2.5-Coder-Instruct with 32B parameters
  - About **18,000 pairs** of contrastive traces
  - **Five** programming languages: C#, JavaScript, Java, Python, and C

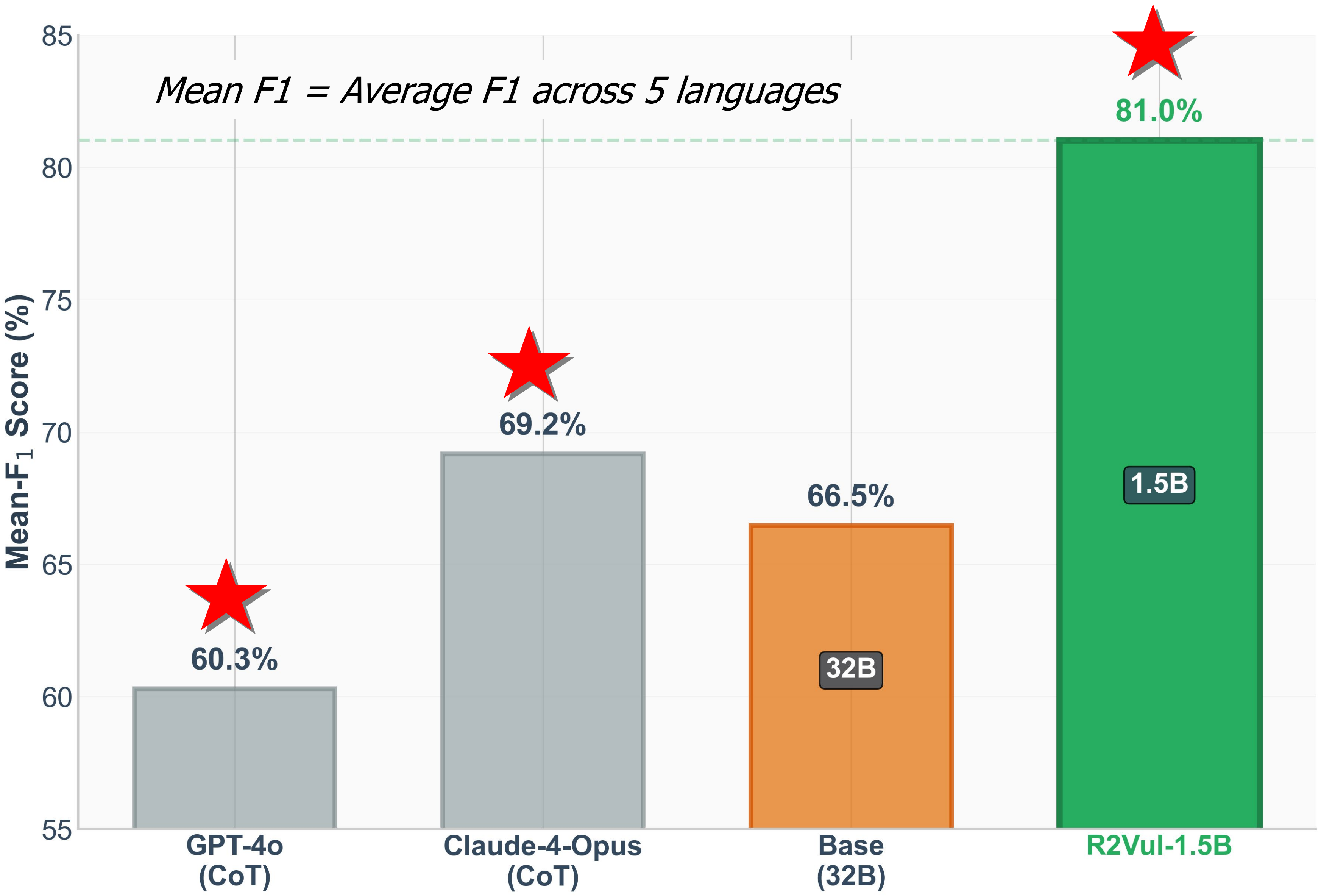
- Step 2:
  - Target LLMs: Qwen2.5-Coder-Instruct with 0.5B, 1.5B, and 7B parameters

Main Results: Detection Performance



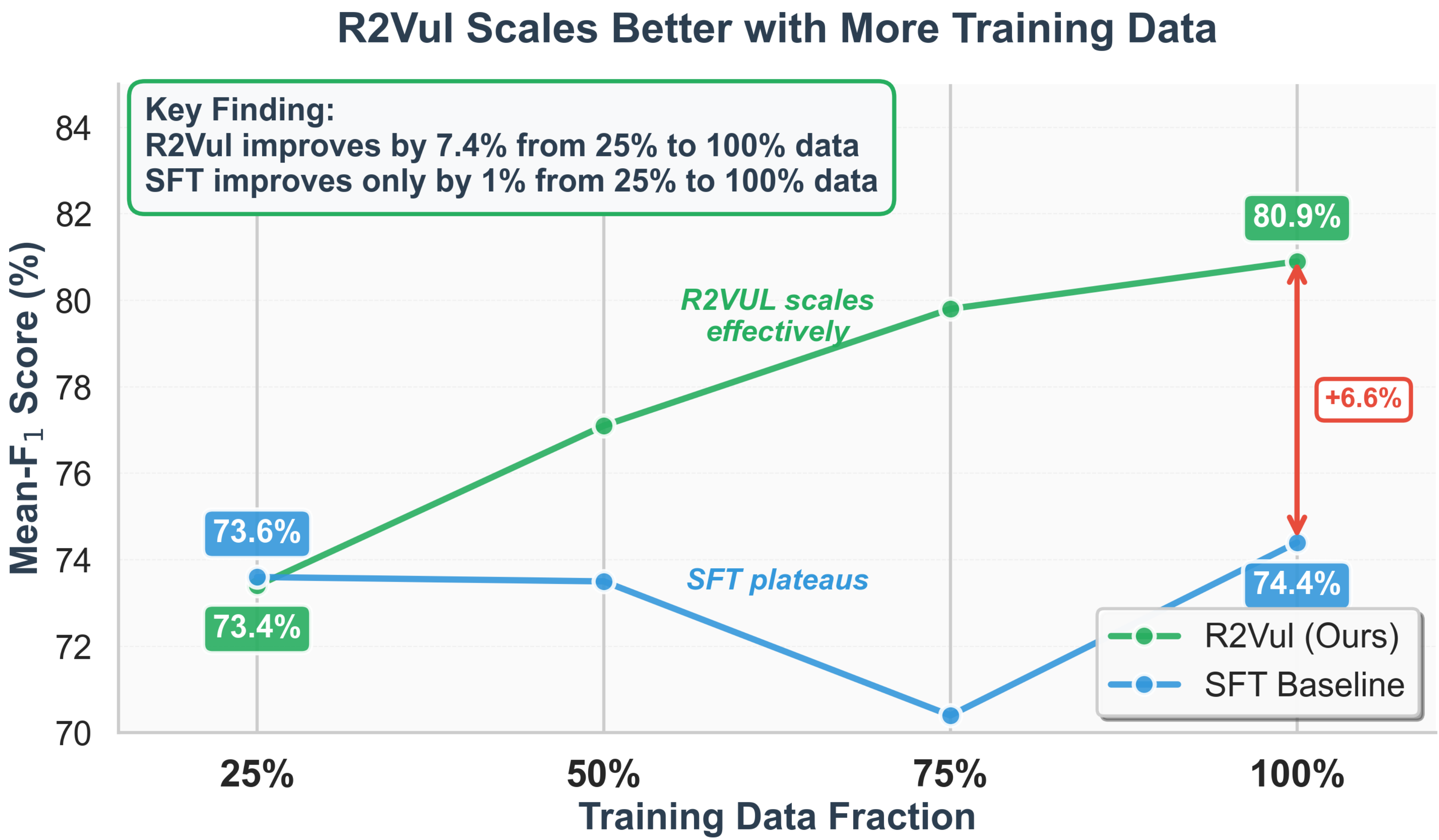
R2Vul-1.5B wins over Base (32B)

Main Results: Detection Performance



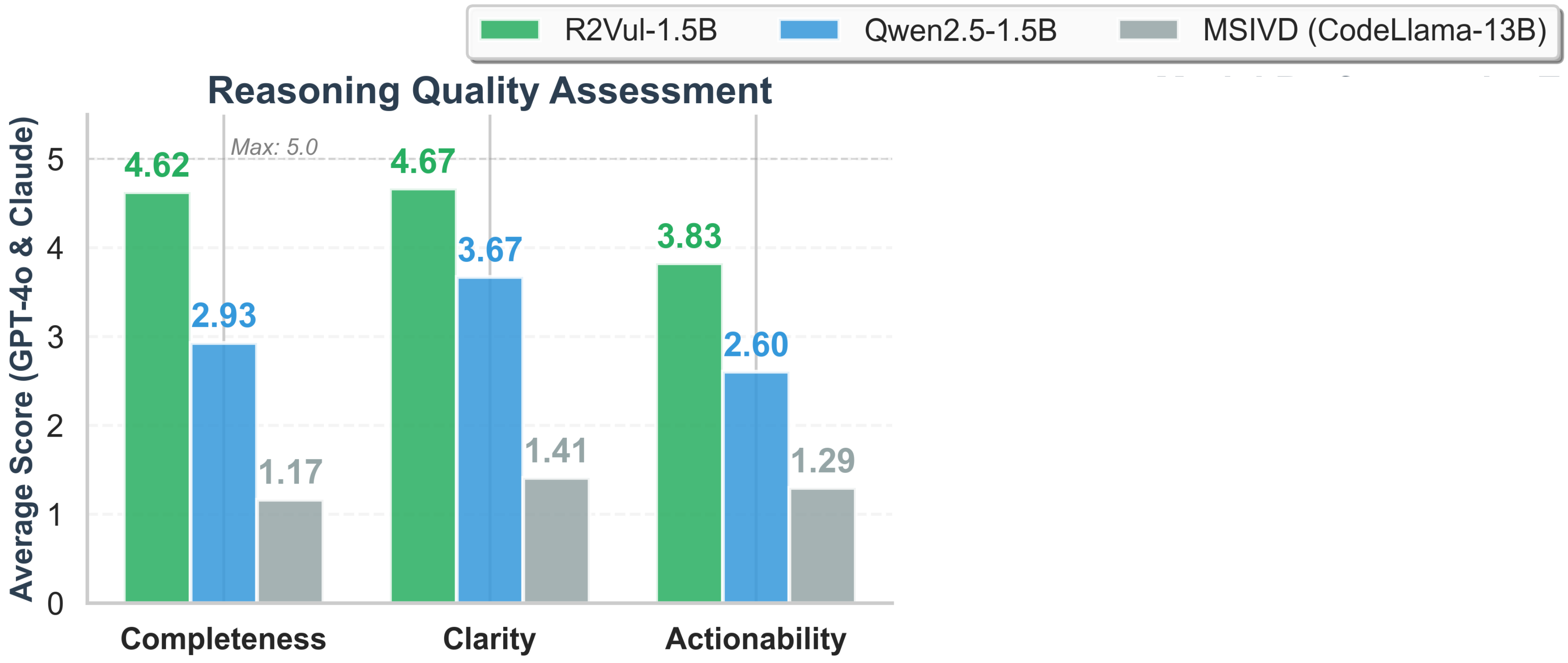
R2Vul-1.5B wins over commercial LLMs

R2Vul vs. SFT: Impact of Contrastive Reasoning Trace Pairs

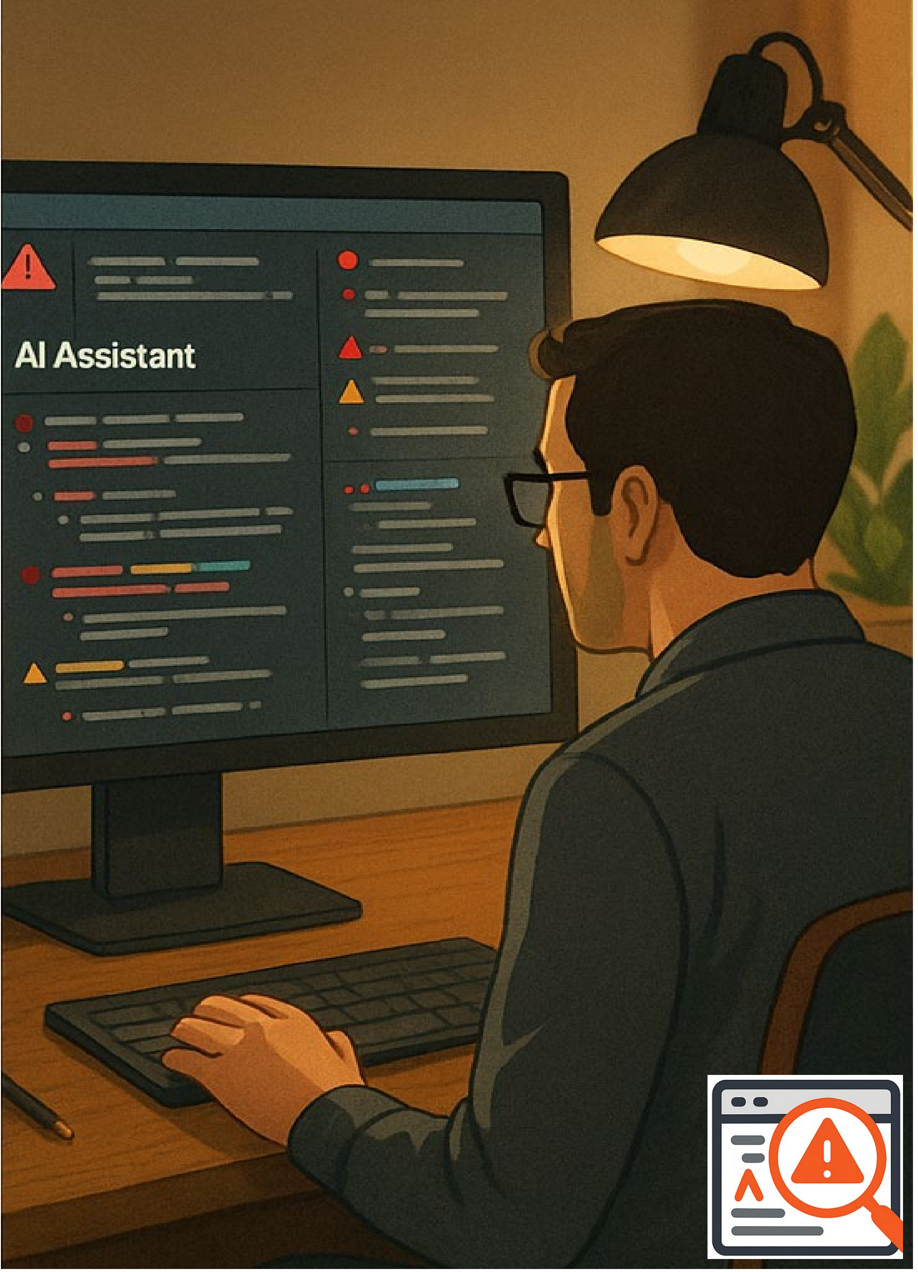


Reasoning Quality Assessment

Baselines: Qwen2.5-1.5B-Coder-Instruct and MSIVD (CodeLlama-13B): reasoning-based LLM [1]
Evaluators: GPT-4o, Claude-3.5-Sonnet, and two human experts



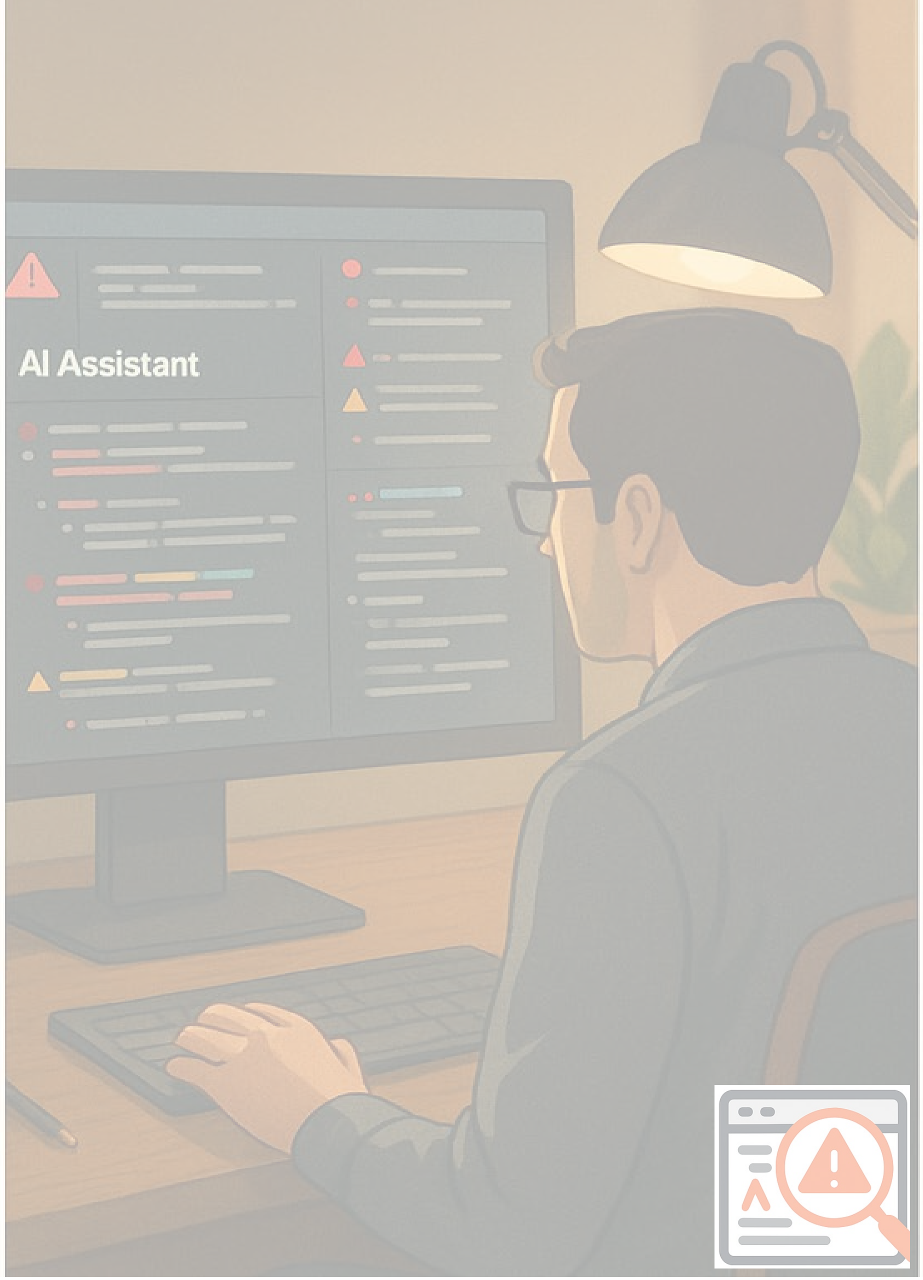
[1] Yang, Aidan ZH, et al. "Security vulnerability detection with multitask self-instructed fine-tuning of large language models." *arXiv preprint arXiv:2406.05892* (2024).



Critique

Engineering **contrastive reasoning trace pairs** help LLM reasoning for vulnerability detection

Code



Critique

Cure

Integrating Diverse Inputs from Diverse Sources Can Help AI

SANER 2016

History Driven Program Repair

Xuan-Bach D. Le, David Lo
School of Information Systems
Singapore Management University
{dxb.le.2013,davidlo}@smu.edu.sg

Claire Le Goues
School of Computer Science
Carnegie Mellon University
clegoues@cs.cmu.edu

First MSR-powered APR work: ***mines hundreds of GitHub repositories for repair patterns*** *serving as* ***augmented inputs*** *to test cases for search-based automated program repair (APR); most-cited SANER 2016 paper that inspired Facebook*

Integrating Diverse Inputs for LLM-Powered Vulnerability Repair

ICSE 2024

Out of Sight, Out of Mind: Better Automatic Vulnerability Repair by Broadening Input Ranges and Sources

Xin Zhou
Singapore Management University
Singapore
xinzhou.2020@phdcs.smu.edu.sg

Kisub Kim*
Singapore Management University
Singapore
kisubkim@smu.edu.sg

Bowen Xu
North Carolina State University
USA
bxu22@ncsu.edu

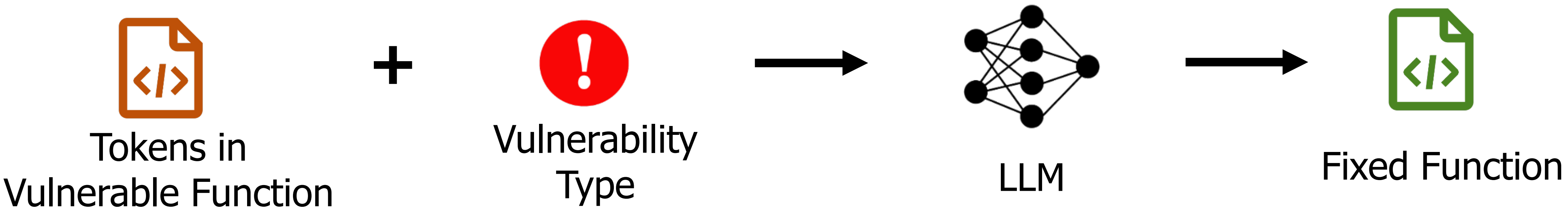
DongGyun Han
Royal Holloway, University of London
United Kingdom
donggyun.han@rhul.ac.uk

David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

*First LLM-powered automated program repair work that **(i) links vulnerable code to LLM-enriched CWE knowledge** and **(ii) leverages AST contents** to double the efficacy of prior work*

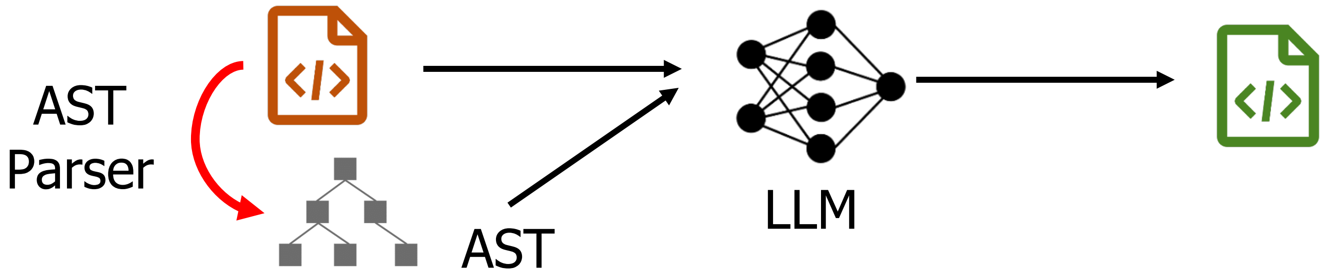
Integrating Diverse Inputs for LLM-Powered Vulnerability Repair

Previous solutions:



Many **other inputs** have not been leveraged:

- *Abstract Syntax Tree*



- *CWE Knowledge*

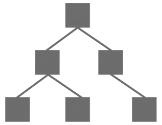
VulMaster's Design

Data-Centric Innovations

\+

Multi-LLM Collaboration

Incorporate AST

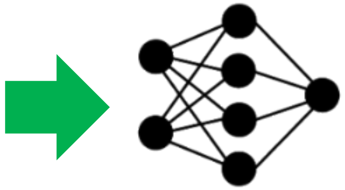


Incorporate CWE knowledge

CWE

Address lengthy inputs

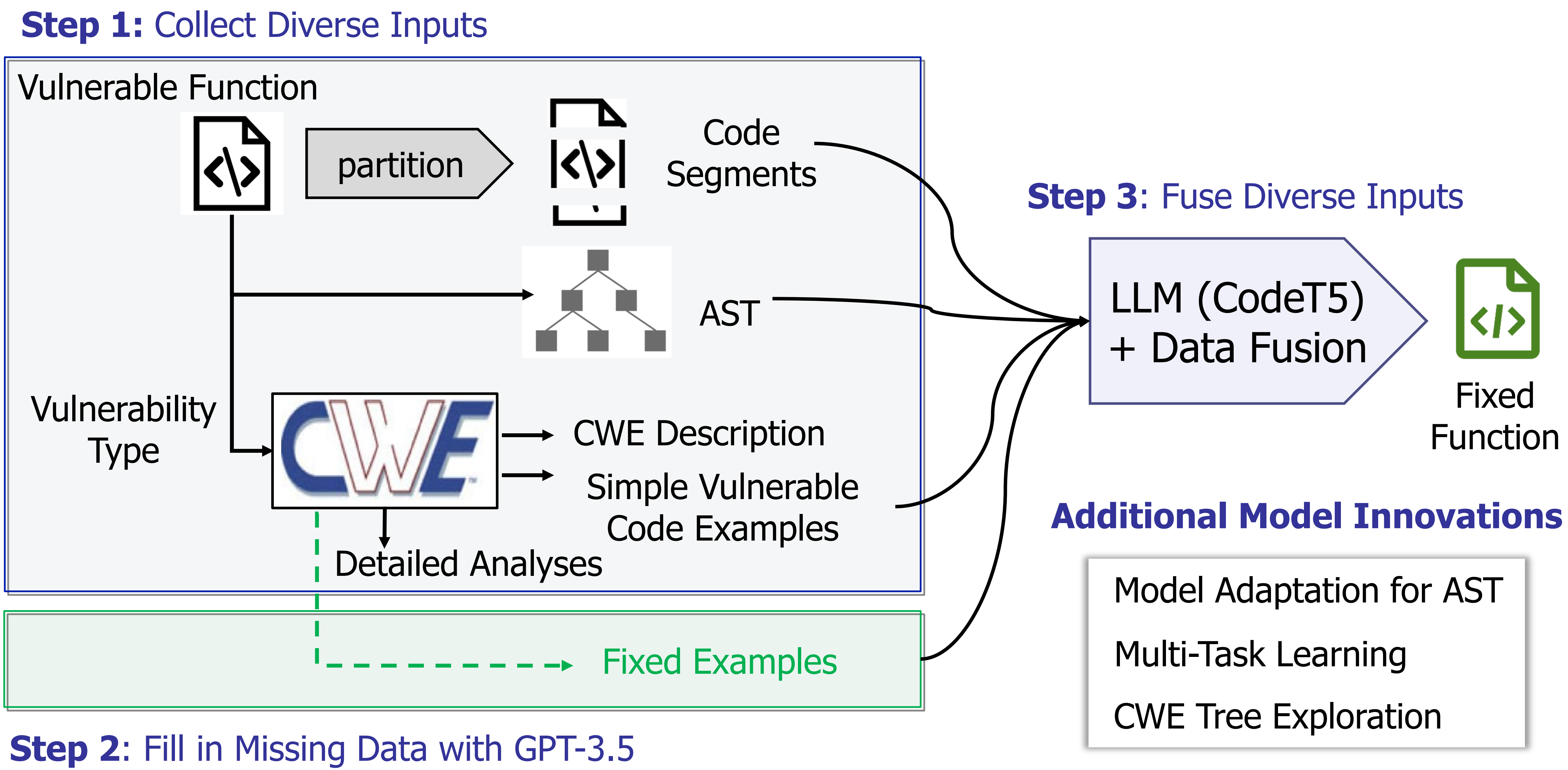
GPT-3.5



CodeT5

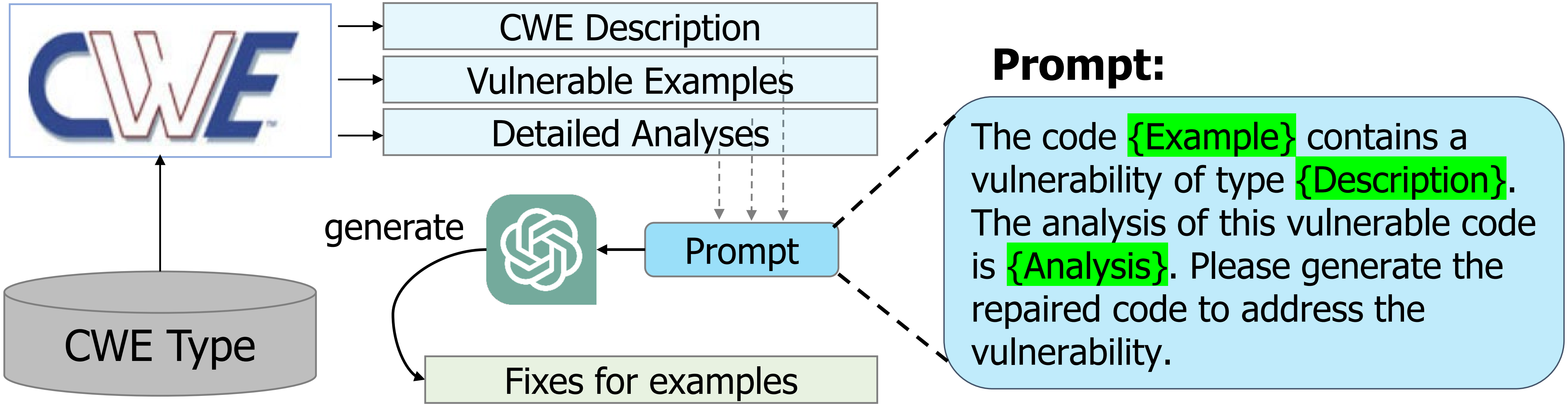
= **2x Fixed Vulnerabilities**

Overall Framework



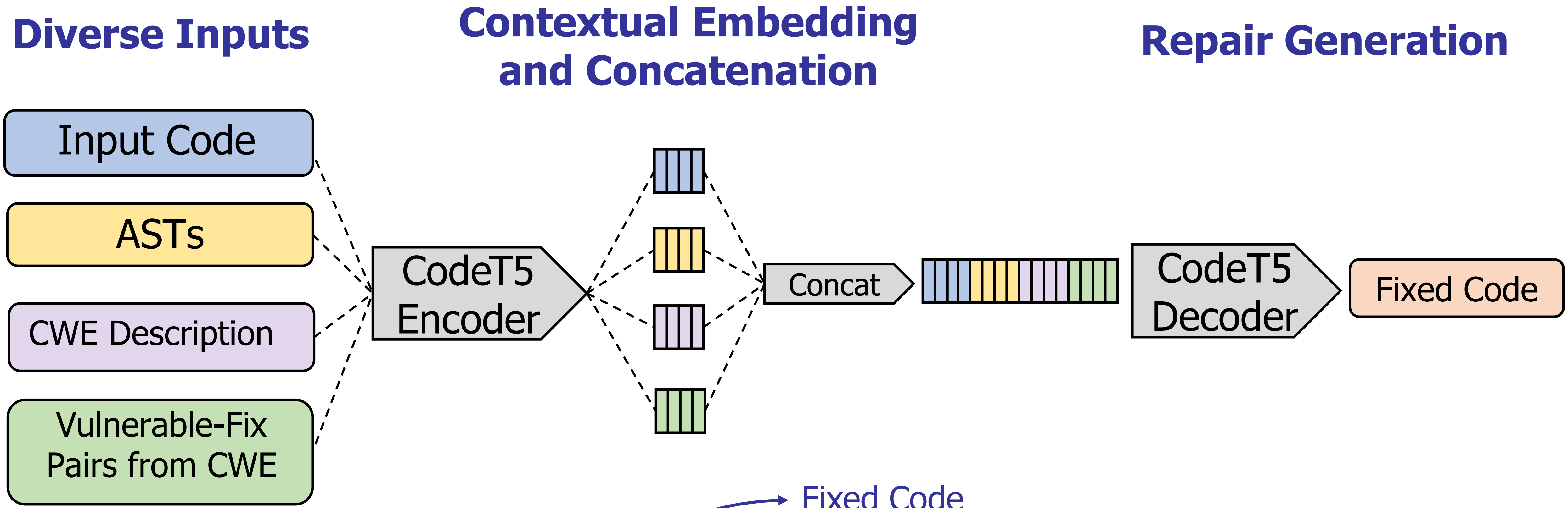
Step 2: Fill In Missing Data with GPT-3.5

- General-purpose LLMs are effective for
  - Fixing simple (toy) vulnerable examples when *detailed analysis* is given.
- Thus, we use **GPT-3.5** to generate the fixed CWE examples:



- **Note:** Real vulnerability fixes are not simple or come with detailed analysis

Step 3: Fusing Diverse Input Data Components



$$L_{repair} = -\log p(Y_i | X_i, AST_i, Name_i, Example\text{-}Fix\text{-}Pairs_i)$$

Fixed Code

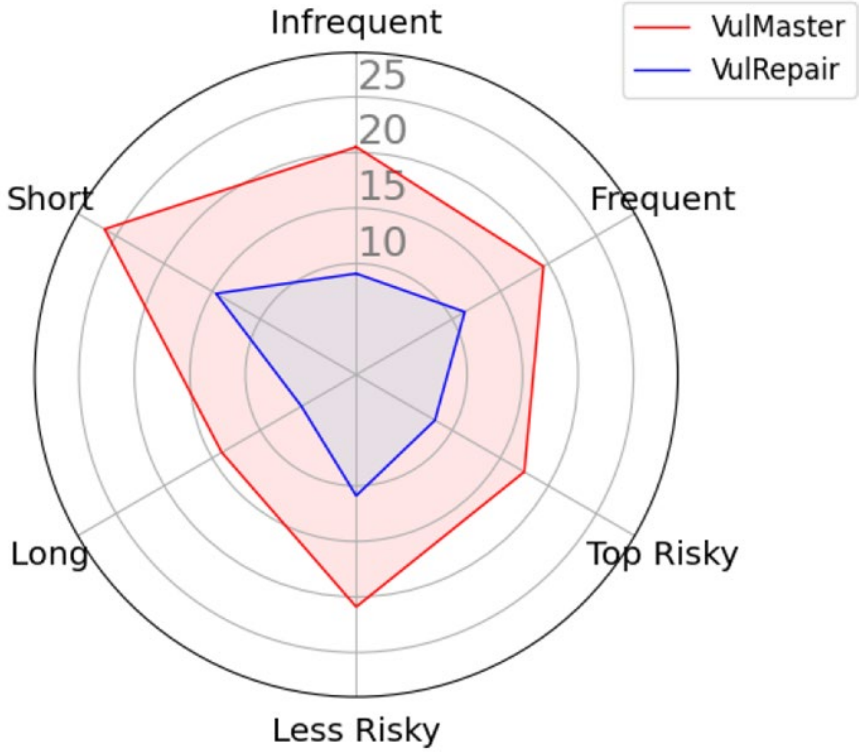
Vulnerable Code

Other Diverse Inputs

Results: Comparisons with SOTA

Main Results

Type	Approach	EM	BLEU
LLM	GPT-3.5 [55]	3.6	8.8
	GPT-4 [56]	5.3	9.7
task-specific	VRepair [9]	8.9	11.3
	VulRepair [19] (SOTA)	10.2	21.3
Ours	VulMaster	**20.0**	**29.3**



- VulMaster **doubles the Exact Match** (EM) score
- VulMaster consistently outperforms for vulnerabilities of different characteristics

- *long/short*: the length of the code
- *frequent/infrequent*: the vulnerability type frequencies
- *top/less risky*: top 10 most dangerous CWEs or not

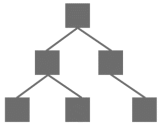
School of Computing and Information Systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

Latest Extension

More Diverse Data + **Reinforcement Learning**

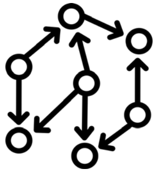
Incorporate AST



Incorporate CWE knowledge

CWE

Incorporate Data Flow



Reasoning steps for repair

GPT-4o

Qwen2.5

Qwen-2.5 7B Instruct

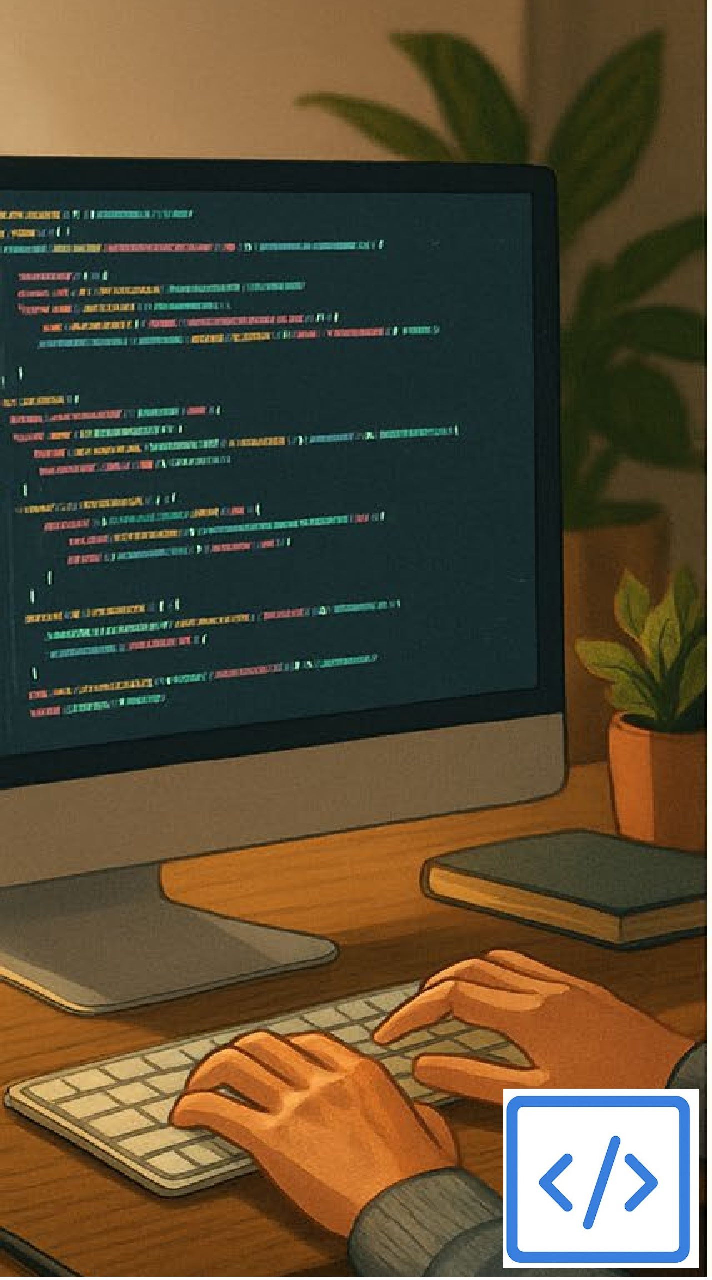
=

+34.96% Performance Gain

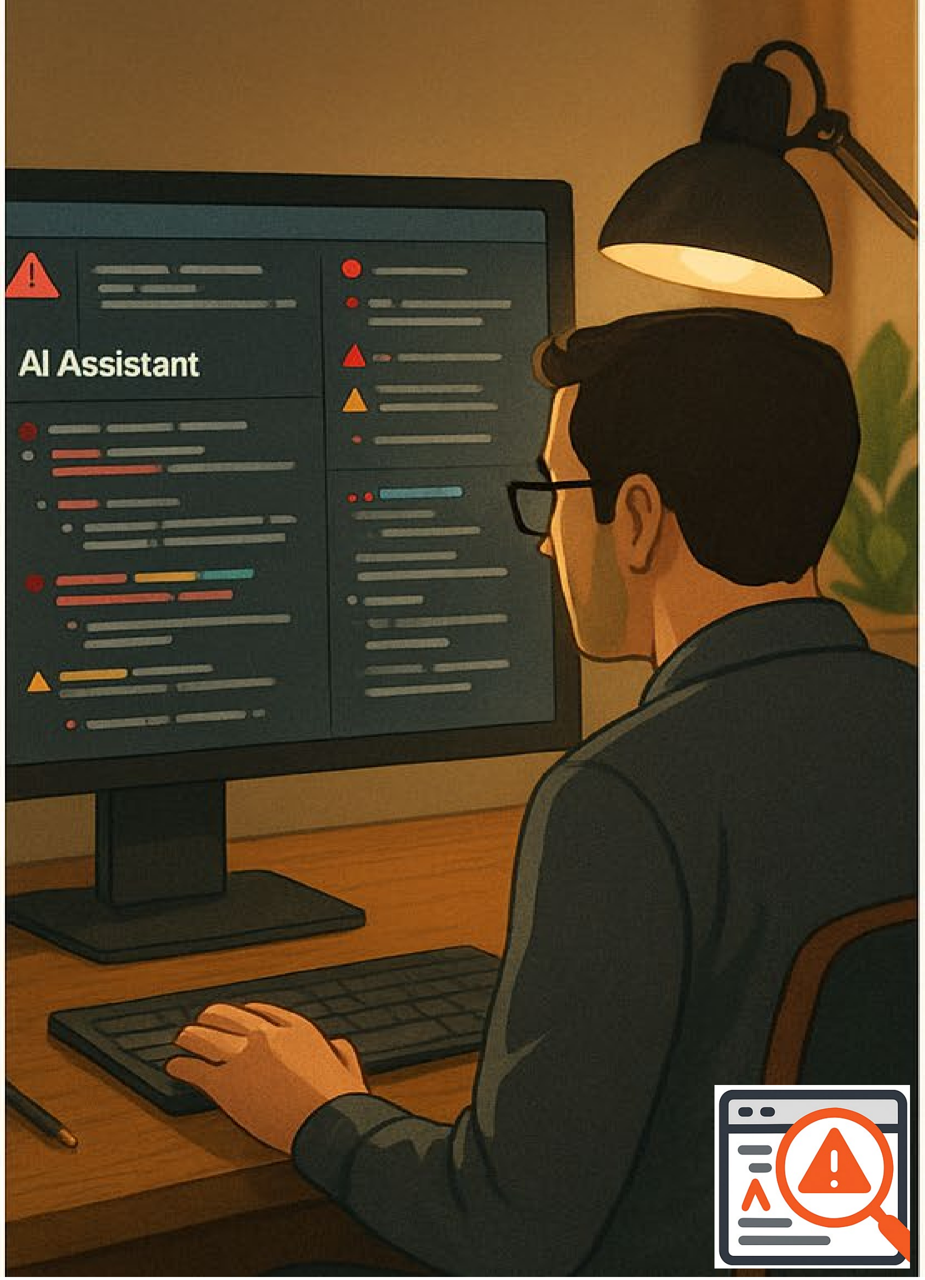
Please stay tuned: Paper will be released on arXiv soon

Cure

Engineering **holistic** representation from **diverse inputs and sources** help LLM reasoning for repair



Code

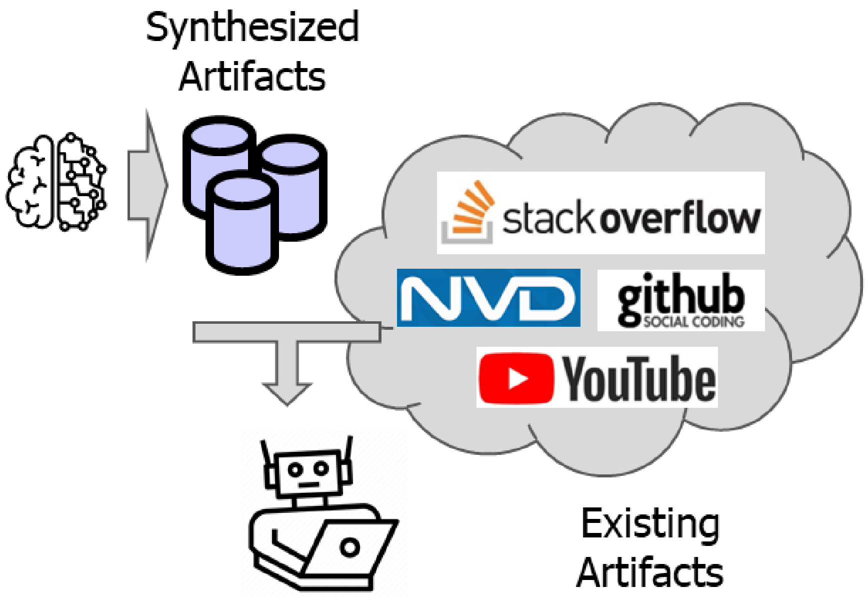


Critique

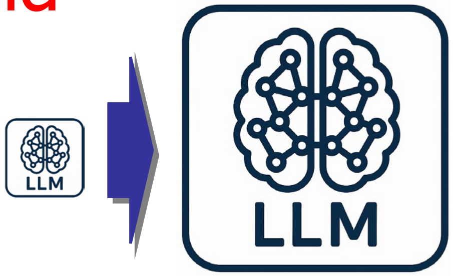
Cure

Data-Centric LLM4SE: Summary

*"How Can We **Systematically Engineer Software Artifacts** to Build Better LLM4SE Bots?"*



Beyond



Prompt Engineering

- We want to **elevate the data**:
  - enriching it, restructuring it, contrasting it, and linking it in ways that *promote deeper reasoning*
- And **utilizing them effectively** to train LLMs to *reason and do better for specific SE tasks*

Data-Centric LLM4SE: Summary

"Because in software engineering,
how we **craft the data**
is how we **shape the intelligence**"

Road Ahead

Road Ahead

Multi-Agent LLM4SE

Real-world problems demand "synergistic collaboration". Multi-agent LLM4SE transforms isolated agents into a coordinated group of experts.

LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead

JUNDA HE, Singapore Management University, Singapore
CHRISTOPH TREUDE, Singapore Management University, Singapore
DAVID LO, Singapore Management University, Singapore

TOSEM 2025

Multi-Agent LLM4SE: Vulnerability Detection

Let the Trial Begin: A Mock-Court Approach to Vulnerability Detection using LLM-Based Agents

Ratnadira Widyasari
ratnadiraw@smu.edu.sg
Singapore Management University
Singapore

Martin Weyssow
mweyssow@smu.edu.sg
Singapore Management University
Singapore

Ivana Clairine Irsan
ivanairsan@smu.edu.sg
Singapore Management University
Singapore

Han Wei Ang
ang_han_wei@tech.gov.sg
GovTech
Singapore

Frank Liauw
frank_liauw@tech.gov.sg
GovTech
Singapore

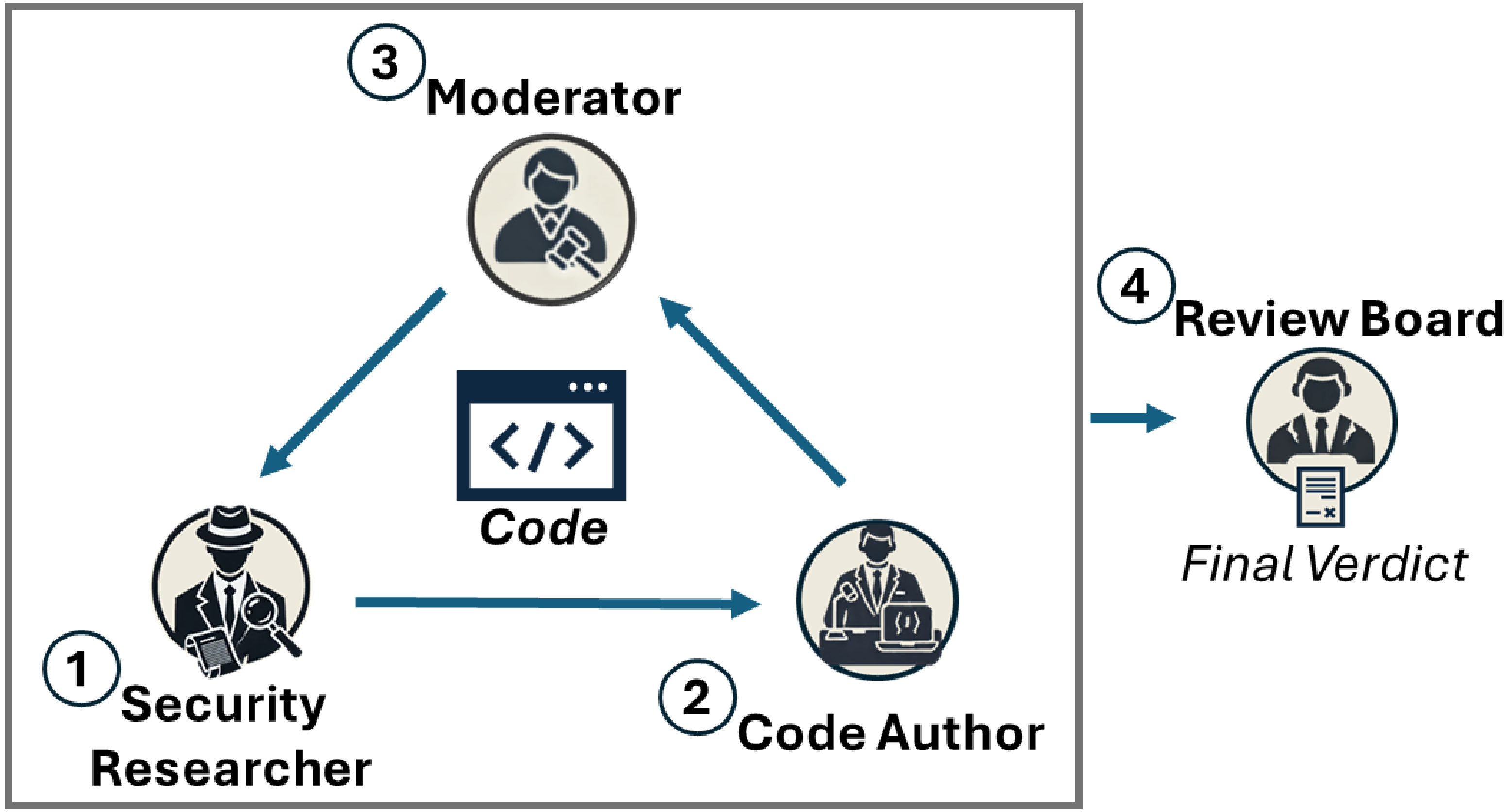
Eng Lieh Ouh
elouh@smu.edu.sg
Singapore Management University
Singapore

Lwin Khin Shar
lkshar@smu.edu.sg
Singapore Management University
Singapore

Hong Jin Kang
hongjin.kang@sydney.edu.au
University of Sydney
Australia

David Lo
davidlo@smu.edu.sg
Singapore Management University
Singapore

Multi-Agent LLM4SE: VulTrial



VulTrial's Design

```
TfLiteStatus EvalGatherNd(TfLiteContext* context,
                          const   TfLiteTensor* params,
                          const TfLiteTensor*comment indices,
                          TfLiteTensor* output) {
  // Vulnerability: There's no check to ensure that 'indices' are
  // within valid bounds for 'params'. Negative or out-of-range
  // indices lead to out-of-bounds reads.
  switch (params->type) {
    case kTfLiteFloat32:
      return GatherNd<float, IndicesT>(params, indices, output);
 // ... other types ...
    default:
      context->ReportError(
          context, "Params type '%s' not supported by gather_nd.",
          TfLiteTypeGetName(params->type));
      return kTfLiteError;
  }
}
```

Example of Vulnerable Code
CWE-125 (Out-of-bounds Read)

Security Researcher

vulnerability	: Out-of-Bounds Read
reason	: The function `EvalGatherNd` does not perform any bounds checking on the `indices` tensor. […].
impact	: Exploiting this vulnerability could lead to information disclosure, […] could be leveraged for further attacks such as Denial of Service (DoS).
vulnerability	: Type Handling
reason	: […] if the new type is not properly validated […], it could lead to unexpected behavior or crashes.
impact	: […] could be exploited by an attacker to cause a DoS

Code Author

vulnerability	: Out-of-Bounds Read	response	: mitigation
reason	: The concern regarding out-of-bounds reads is valid. […] I propose implementing a bounds checking […].		
vulnerability	: Type Handling	response	: refutation
reason	: […] any unsupported type will be caught in the default case, which reports an error. This is a standard practice […].		

Moderator

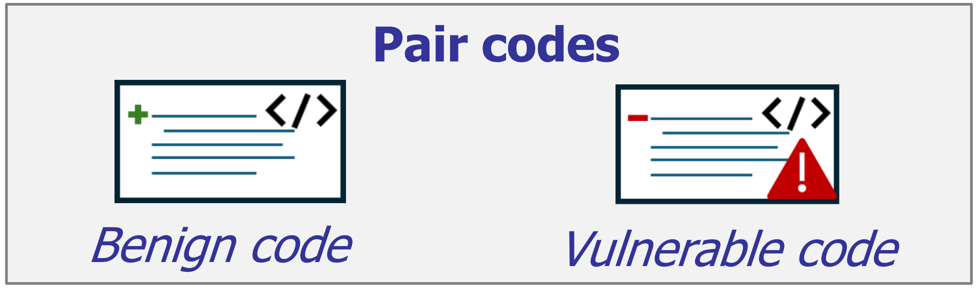
researcher summary	: […] could result in information disclosure […]
author summary	: The author acknowledges the concern […].

Review Board

vulnerability	: Out-of-Bounds Read	decision :	valid
severity	: high	action :	fix immediately
reason	: […] function does not perform bounds checking […]		
vulnerability	: Type Handling	decision :	partially valid
severity	: medium	action :	monitor
reason	: […] potential for new tensor types w/o proper handling […]		

Evaluation Settings

- Evaluation on PrimeVul Pair [1] data.



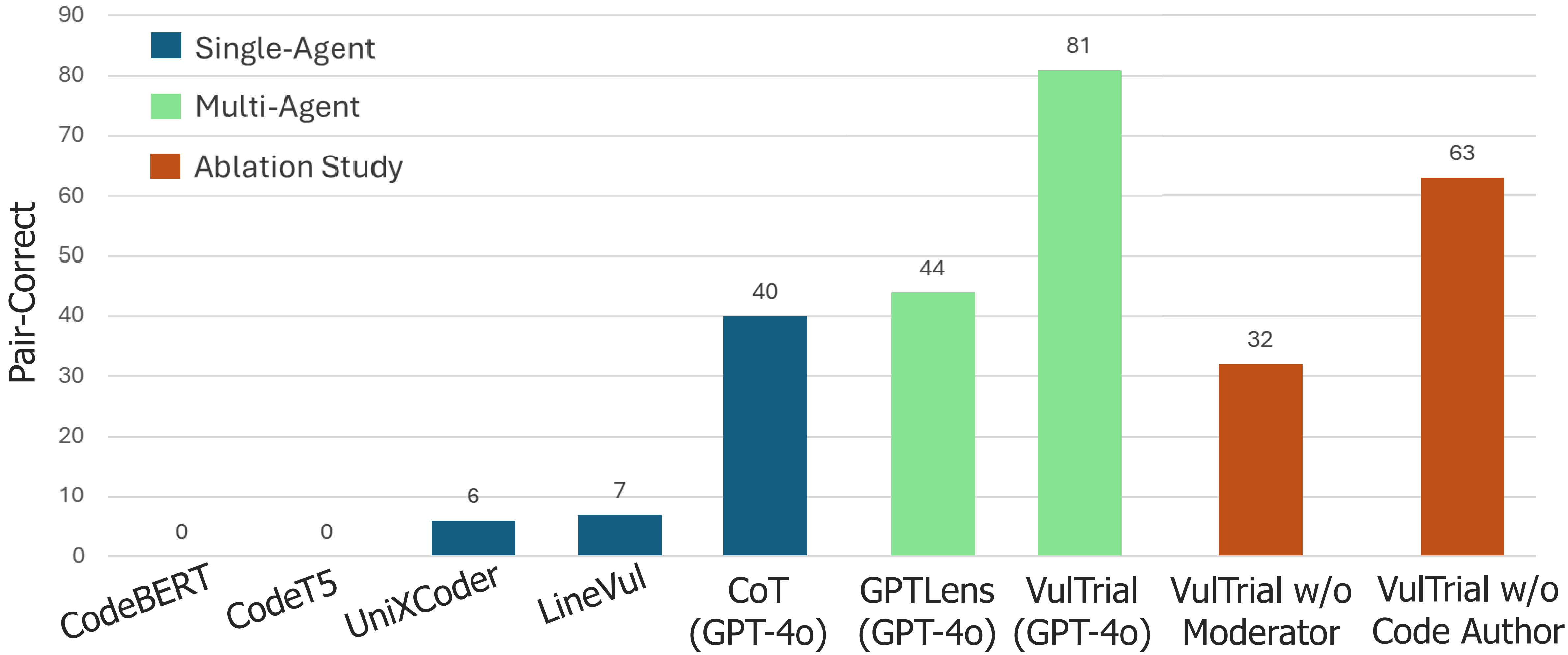
Label	Prediction
Benign	Benign
Vulnerable	Vulnerable

Pair-Correct

Label	Prediction
Benign	Benign
Vulnerable	Benign

[1] Yangruibo Ding, Yanjun Fu, Omniyyah Ibrahim, Chawin Sitawarin, Xinyun Chen, Basel Alomair, David A. Wagner, Baishakhi Ray, Yizheng Chen: Vulnerability Detection with Code Language Models: How Far are We? ICSE 2025: 1729-1741

Evaluation Results



Multi-agent methods **outperform** the single-agent methods

Each agent in VulTrial is **necessary** to achieve the best performance

VulTrial in the Wild

USER INTERACTION	PRIVILEGES REQUIRED	ATTACK VECTOR	ATTACK COMPLEXITY
Required	Low	Network	Low

CVE-2025-49000

Disclosure Date: June 03, 2025 · (Last updated August 14, 2025) ▾

CVE-2025-49000 CVSS v3 Base Score: 3.5

Easy to weaponize | Authenticated | Requires user interaction

Description

InvenTree is an Open Source Inventory Management System. Prior to version 0.17.13, the skip field in the built-in `label-sheet` plugin lacks an upper bound, so a large value forces the server to allocate an enormous Python list. This lets any authenticated label-printing user trigger a denial-of-service via memory exhaustion. the issue is fixed in versions 0.17.13 and higher. No workaround is available aside from upgrading to the patched version.

Related Paper @ ICSME 2025

SAEL: Leveraging Large Language Models with Adaptive Mixture-of-Experts for Smart Contract Vulnerability Detection

Lei Yu[†‡1], Shiqi Cheng[†1], Zhirong Huang[†‡], Jingyuan Zhang[†‡], Chenjie Shen[†‡],
Junyi Lu[†‡], Li Yang[†*], Fengjun Zhang[†§*], Jiajia Ma[†]
[†]Institute of Software, Chinese Academy of Sciences, Beijing, China
[‡]University of Chinese Academy of Sciences, Beijing, China
[§]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
{yulei2022, chengshiqi, huangzhirong2022, zhangjingyuan2023, lujunyi2022}@iscas.ac.cn,
shenchenjie22@mails.ucas.ac.cn, {yangli2017, fengjun, majiajia}@iscas.ac.cn

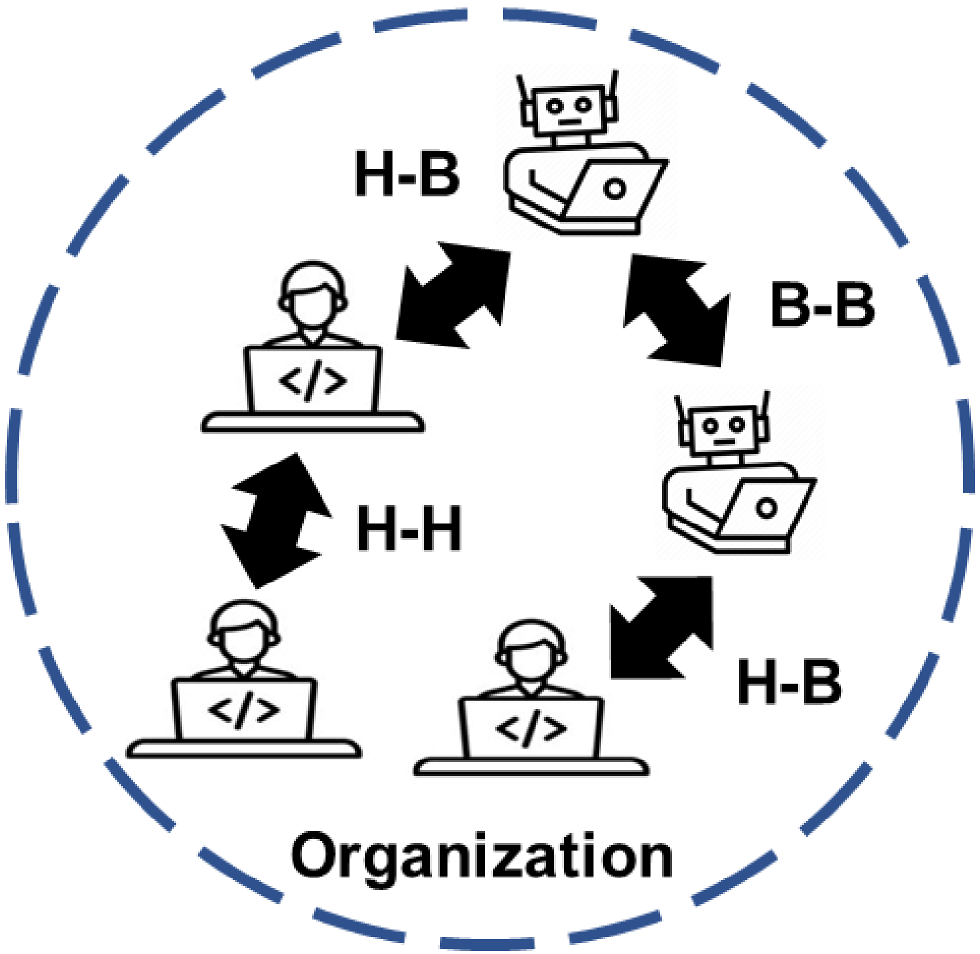
Session 12 - Security 1

Thu 11 Sep 2025 15:45 - 16:00

Case Room 260-057

Towards Full Multi-Agent Solution: Humans (H) and Bots (B)

N:M??



Road Ahead

Greening LLM4SE

LLM4SE solutions are large, slow, and not green, introducing much cost, latency, and carbon footprint

TOSEM 2025

Efficient and Green Large Language Models for Software Engineering: Vision and the Road Ahead

JIEKE SHI, ZHOU YANG, and DAVID LO, Singapore Management University, Singapore

Greening LLM4SE: Three Strategies

Stop **Simplify** **Shrink**

Greening LLM4SE

ISSTA 2024

When to Stop? Towards Efficient Code Generation in LLMs with Excess Token Prevention

Lianghong Guo
Sun Yat-sen University
Zhuhai, China
guolh8@mail2.sysu.edu.cn

Yanlin Wang*
Sun Yat-sen University
Zhuhai, China
wangylin36@mail.sysu.edu.cn

Ensheng Shi
Xi'an Jiaotong University
Xi'an, China
s1530129650@stu.xjtu.edu.cn

Wanjun Zhong
Sun Yat-sen University
Guangzhou, China
zhongwj25@mail2.sysu.edu.com

Hongyu Zhang
Chongqing University
Chongqing, China
hyzhang@cqu.edu.cn

Jiachi Chen
Sun Yat-sen University
Zhuhai, China
chenjch86@mail.sysu.edu.cn

Ruikai Zhang
Huawei Cloud Computing Technologies Co., Ltd.
Shenzhen, China

Yuchi Ma
Huawei Cloud Computing Technologies Co., Ltd.
Shenzhen, China

Zibin Zheng
Sun Yat-sen University
Zhuhai, China
zhzibin@mail.sysu.edu.cn

Stop

Improve code generation speed by up to 4.5x by **terminating inference early**

Greening LLM4SE

Simplify

ISSTA 2024

AI Coders Are Among Us: Rethinking Programming Language Grammar Towards Efficient Code Generation

Zhensu Sun
Singapore Management University
Singapore
zssun@smu.edu.sg

Xiaoning Du*
Monash University
Australia
xiaoning.du@monash.edu

Zhou Yang
Singapore Management University
Singapore
zyang@smu.edu.sg

Li Li
Beihang University
China
lilicoding@ieee.org

David Lo
Singapore Management University
Singapore
davidlo@smu.edu.sg

Won ACM SIGSOFT Distinguished Paper Award

First work to propose a **programming language grammar for AI agents**

Greening LLM4SE

Shrink

ESEC/FSE 2023

Towards Greener Yet Powerful Code Generation via Quantization: An Empirical Study

Xiaokai Wei
xiaokaiw@amazon.com
AWS AI Labs
USA

Sujan Kumar Gonugondla
gsujan@amazon.com
AWS AI Labs
USA

Shiqi Wang
wshiqi@amazon.com
AWS AI Labs
USA

Wasi Ahmad
wuahmad@amazon.com
AWS AI Labs
USA

Baishakhi Ray
rabaisha@amazon.com
AWS AI Labs
USA

Haifeng Qian
qianhf@amazon.com
AWS AI Labs
USA

Reduce size by 3x and latency by 50% by **quantizing parameter** into int8

Related Paper @ ICSME 2025

Is Quantization a Deal-breaker? Empirical Insights from Large Code Models

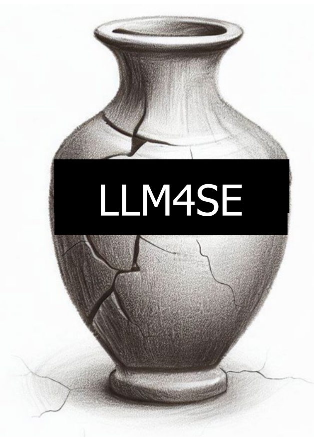
Saima Afrin
Department of Computer Science
William & Mary
Williamsburg, VA, USA
safrin@wm.edu

Bowen Xu
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
bxu22@ncsu.edu

Antonio Mastropaolo
Department of Computer Science
William & Mary
Williamsburg, VA, USA
amastropaolo@wm.edu

Session 2 - Quality Assurance 1
Wed 10 Sep 2025 10:45 - 11:00
Case Room 260-057

Many Open Problems wrt. **Non-Functional** Properties

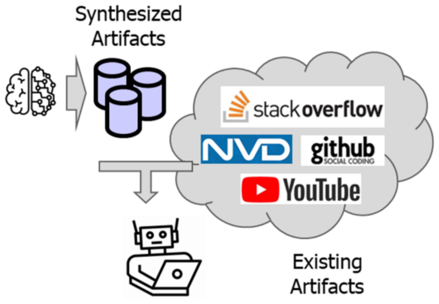


***Robustness*, *Security*, *Privacy*, *Explainability*, *Efficiency*, and *Usability* of Large Language Models for Code**

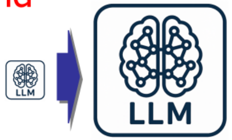
ZHOU YANG, Singapore Management University, Singapore
ZHENSU SUN, Singapore Management University, Singapore
TERRY ZHUO YUE, Singapore Management University, Singapore
PREMKUMAR DEVANBU, Department of Computer Science, UC Davis, USA
DAVID LO, Singapore Management University, Singapore

Data-Centric LLM4SE: Summary

*"How Can We **Systematically Engineer Software Artifacts** to Build Better LLM4SE Bots?"*

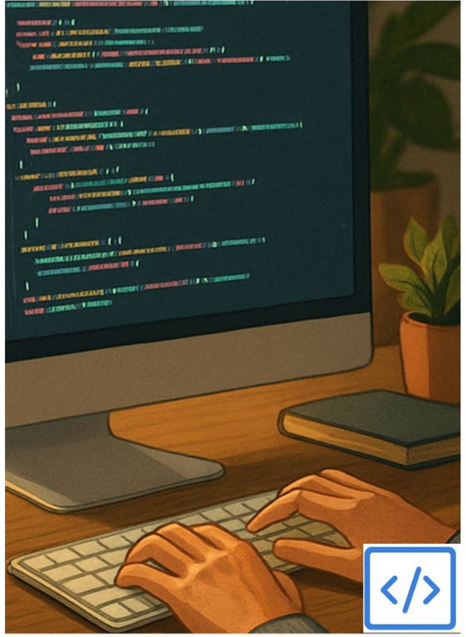


Beyond



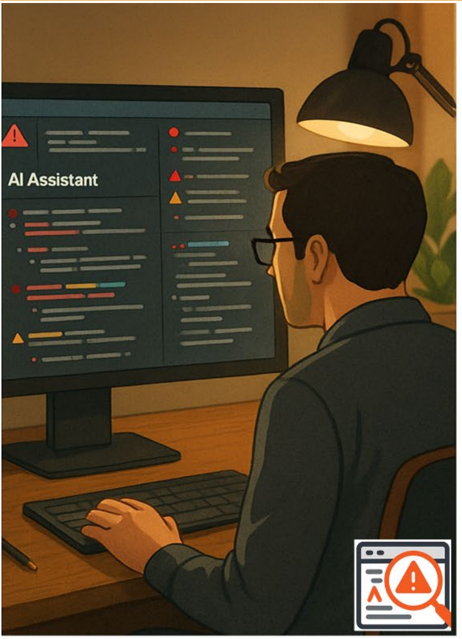
Prompt Engineering

- We want to **elevate the data**:
  - enriching it, restructuring it, contrasting it, and linking it in ways that *promote deeper reasoning*
- And **utilizing them effectively** to train LLMs to *do better for SE tasks*



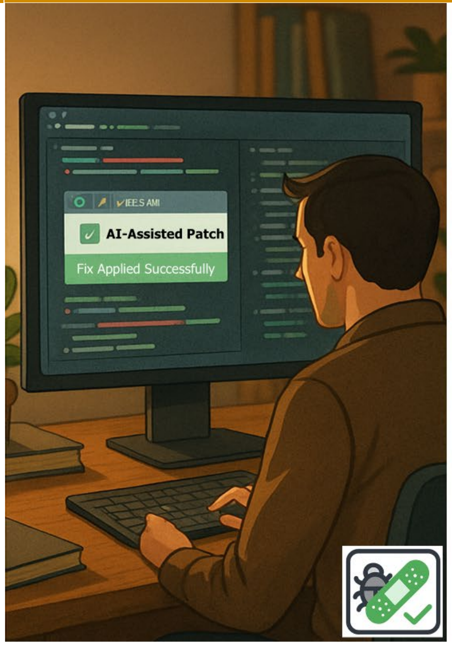
Code

Engineering **structured reasoning traces** help LLM reasoning for code generation



Critique

Engineering **contrastive reasoning trace pairs** help LLM reasoning for vulnerability detection

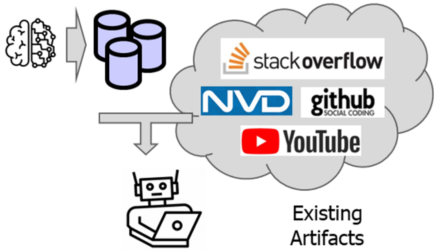


Cure

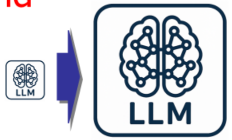
Engineering **holistic** representation from **diverse inputs and sources** help LLM reasoning for repair

Data-Centric LLM4SE: Summary

*“How Can We **Systematically Engineer Software Artifacts** to Build Better LLM4SE Bots?”*

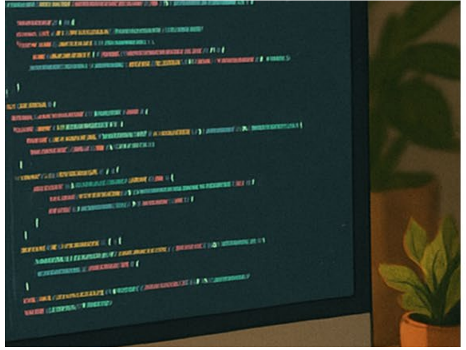


Beyond



Prompt Engineering

- We w
  - er it, d
- And u LLMs



Engineering **structured reasoning** traces help LLM reasoning for code generation

Road Ahead

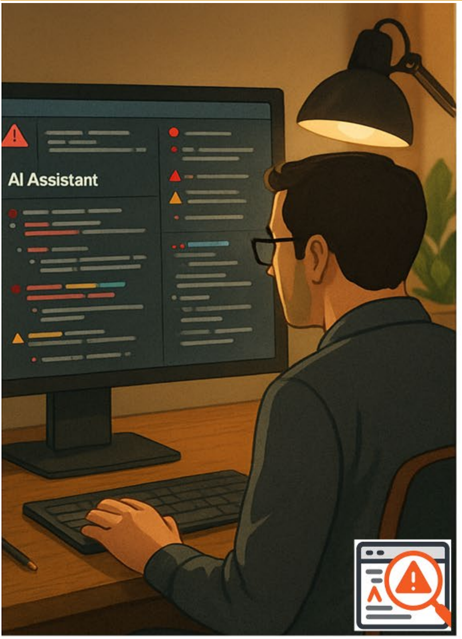
Multi-Agent LLM4SE

Real-world problems demand "synergistic collaboration" and multi-agent AI transforms isolated AI into coordinated experts.

LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead

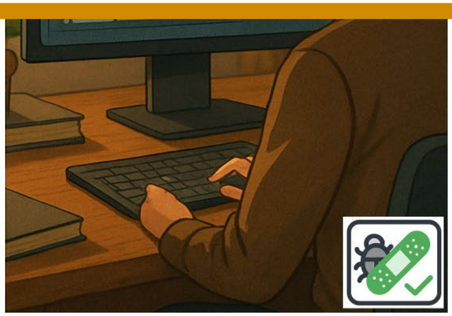
JUNDA HE, Singapore Management University, Singapore
CHRISTOPH TREUDE, Singapore Management University, Singapore
DAVID LO, Singapore Management University, Singapore

TOSEM 2025



Critique

Engine **trace pairs** help LLM reasoning for vulnerability detection



Cure

eering **holistic** representation from **diverse inputs and sources** help LLM reasoning for repair


GitHub Copilot
overflow AI
JET BRAINS
aws
Amazon CodeWhisperer
salesforce
tabnine
aiXcoder
starcoder

Acknowledgements

ICSME 2025
Auckland, New Zealand

SMU
SINGAPORE MANAGEMENT UNIVERSITY

OUB Chair
Professorship Fund

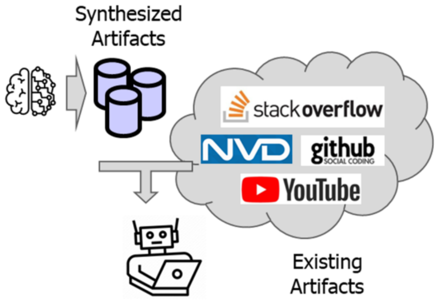
Thank you!

Questions? Comments? Advice?

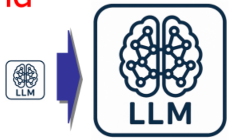
davidlo@smu.edu.sg

Data-Centric LLM4SE: Summary

"How Can We ***Systematically Engineer Software Artifacts*** *to Build Better LLM4SE Bots?"*

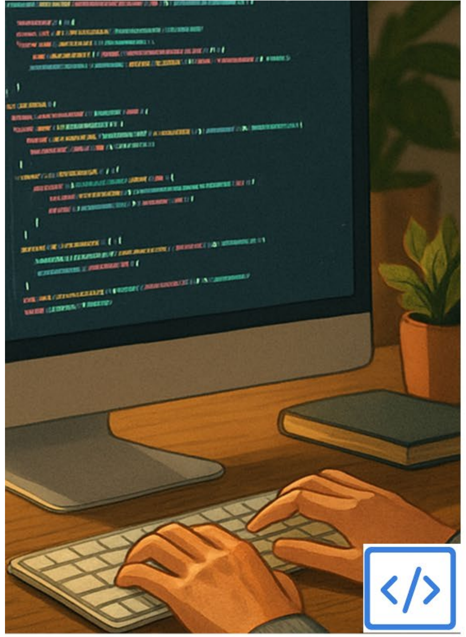


Beyond



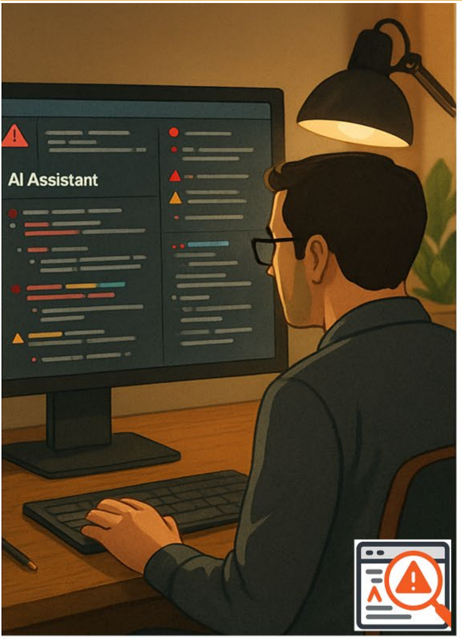
Prompt Engineering

- We want to **elevate the data**:
  - enriching it, restructuring it, contrasting it, and linking it in ways that *promote deeper reasoning*
- And **utilizing them effectively** to train LLMs to *do better for SE tasks*



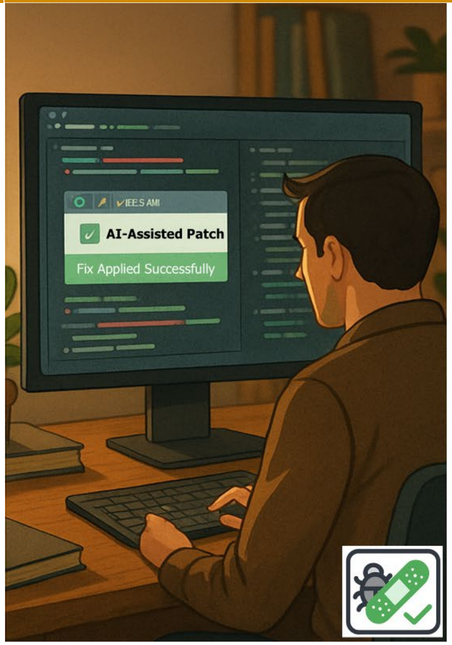
Code

Engineering **structured reasoning traces** help LLM reasoning for code generation



Critique

Engineering **contrastive reasoning trace pairs** help LLM reasoning for vulnerability detection

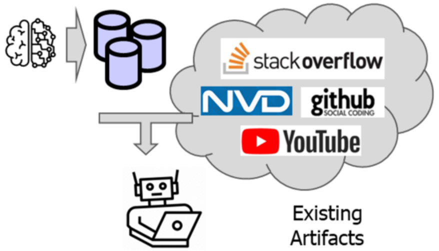


Cure

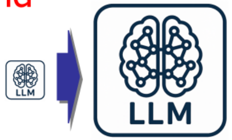
Engineering **holistic** representation from **diverse inputs and sources** help LLM reasoning for repair

Data-Centric LLM4SE: Summary

*"How Can We **Systematically Engineer Software Artifacts** to Build Better LLM4SE Bots?"*

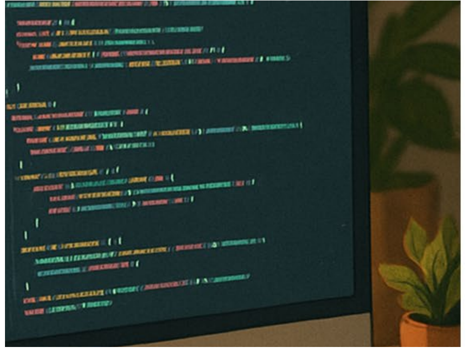


Beyond



Prompt Engineering

- We w
  - er
    it,
    d
- And u
  LLMs



Engineering **structured reasoning** traces help LLM reasoning for code generation

Road Ahead

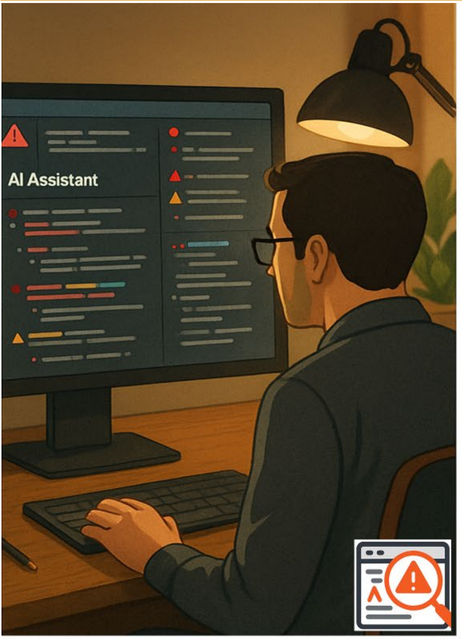
Multi-Agent LLM4SE

Real-world problems demand "synergistic collaboration" and multi-agent AI transforms isolated AI into coordinated experts.

LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead

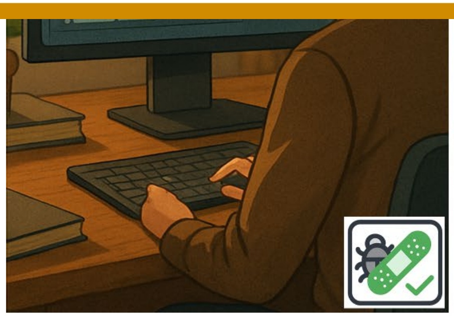
JUNDA HE, Singapore Management University, Singapore
CHRISTOPH TREUDE, Singapore Management University, Singapore
DAVID LO, Singapore Management University, Singapore

TOSEM 2025



Critique

Engine... **trace pairs** help LLM reasoning for vulnerability detection



Cure

...eering **holistic** representation from **diverse inputs and sources** help LLM reasoning for repair