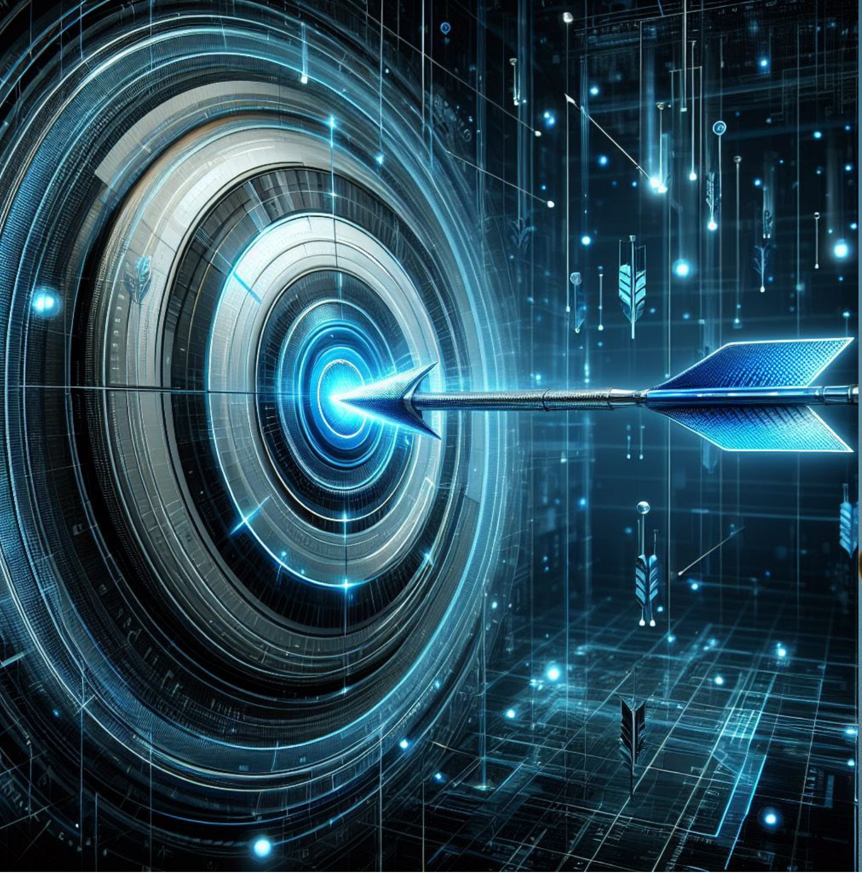


Efficacy, Efficiency, and Security of Code LLMs: Promises and Perils

David Lo

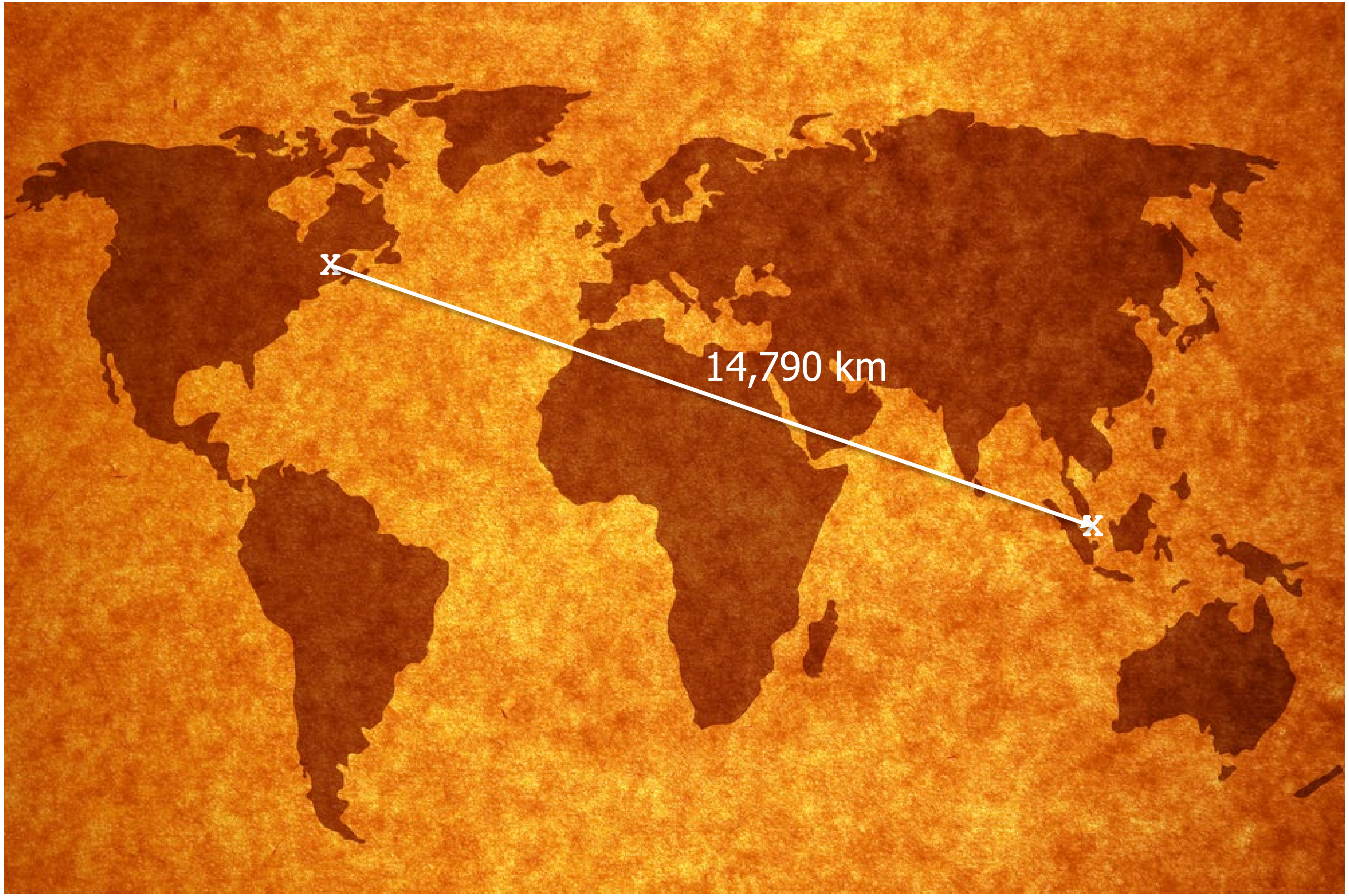
OUB Chair Professor of Computer Science
Director, Center for Research on Intelligent SE (RISE)

SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

SEMLA 2024

Self-Introduction



Self-Introduction

Self-Introduction

Self-Introduction

Singapore Management University

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

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, June 2024

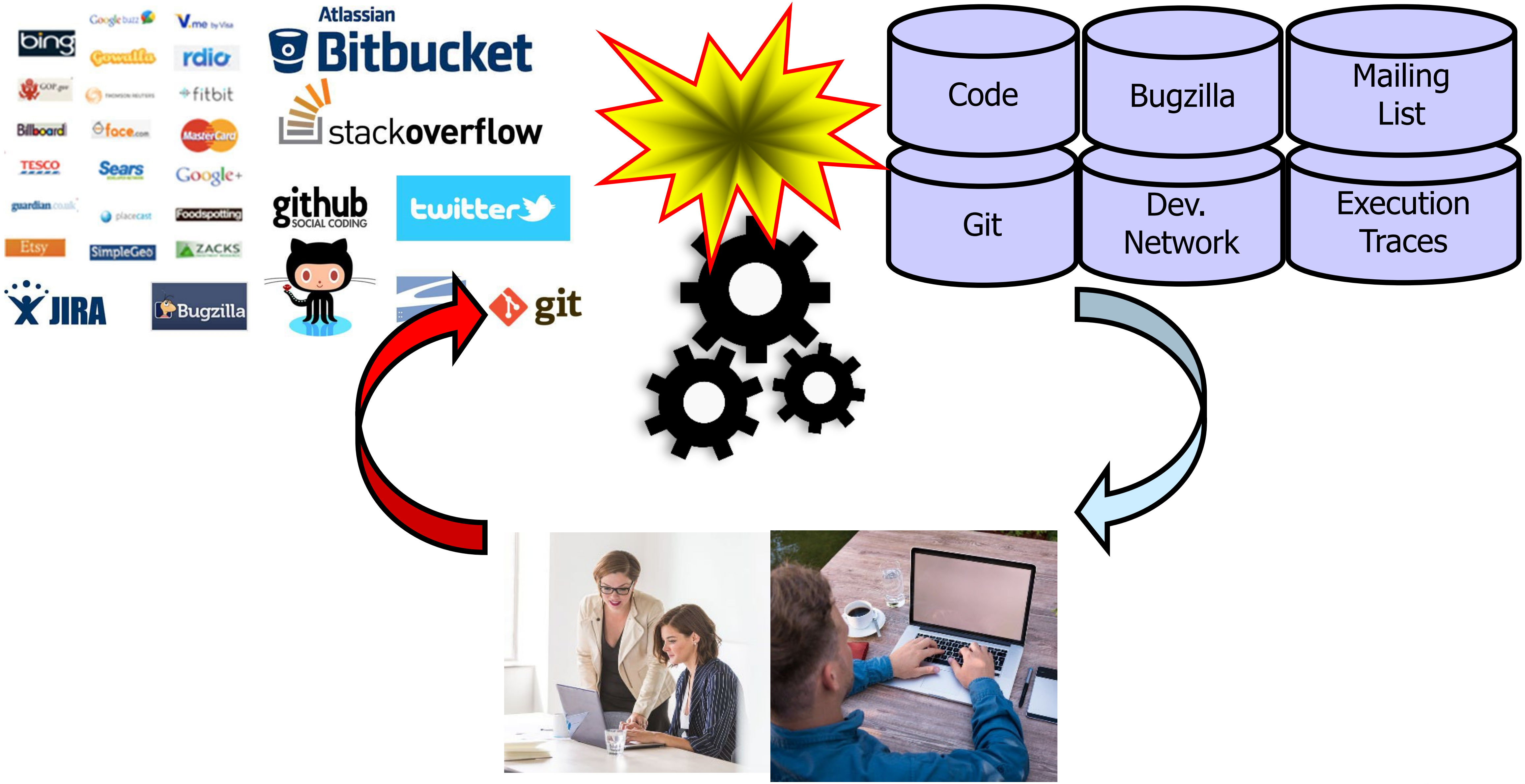
#	Institution	Count	Faculty
1	▶ Nanjing University	39.0	38
2	▶ Carnegie Mellon University	31.6	17
3	▶ Peking University	28.5	21
4	▶ Singapore Management University	22.7	8

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

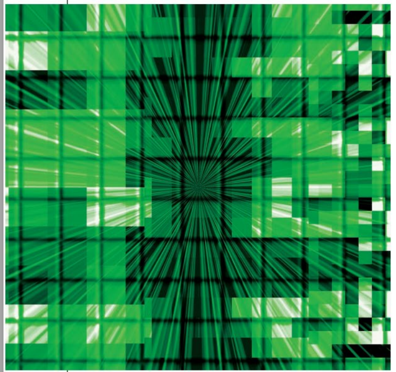
Efficient Mining of Iterative Patterns for Software Specification Discovery

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

Future Direction in AI4SE

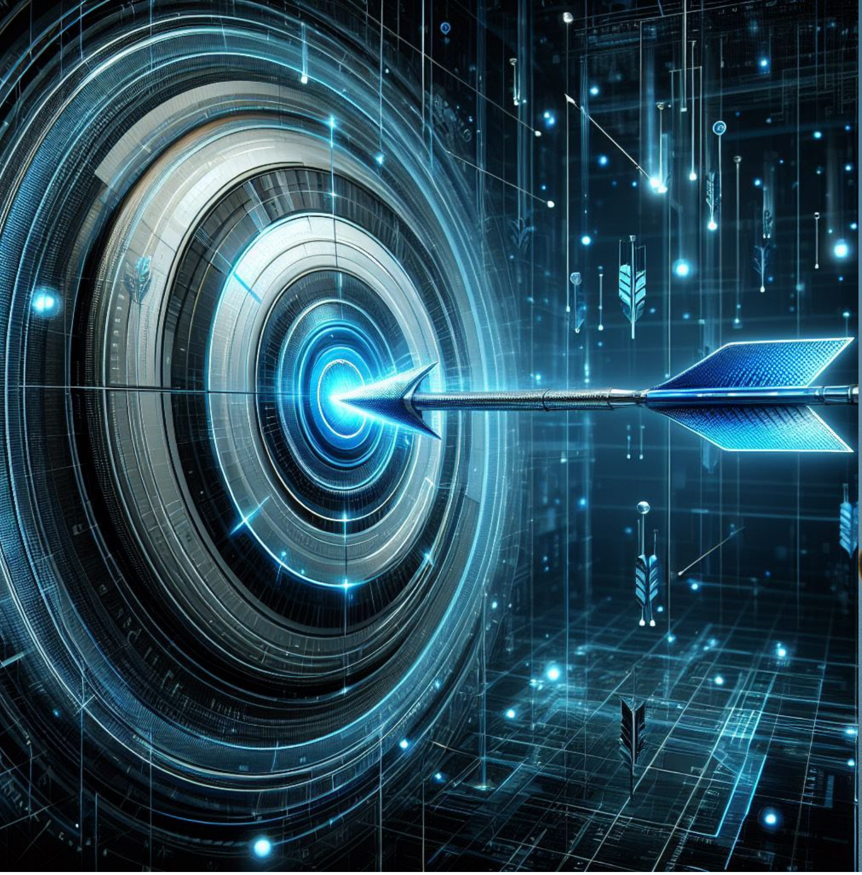
Trustworthy and Synergistic Artificial Intelligence for Software Engineering: Vision and Roadmaps

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

ICSE'23 Future of SE Talk

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

POLYTECHNIQUE MONTRÉAL
Université de Montréal
Queen's University
UBC
University of Victoria
Université Concordia University
University of Saskatchewan
York U
University of Manitoba
University of Illinois Urbana-Champaign
UCSB
Carnegie Mellon University
UTD
NC STATE UNIVERSITY
The University of California Davis
The University of California UCLA
The University of California Irvine
University of Houston Founded 1927
Inria
Royal Holloway University of London
université de Bordeaux
TU/e Eindhoven University of Technology
uni.lu Université du Luxembourg
Humboldt-Universität zu Berlin
Università degli Studi di Salerno
Zhejiang University 1897
Peking University 1898
Nanjing University 1902
Chongqing University 1929
香港中文大學 The Chinese University of Hong Kong
MONASH
Australian National University
The University of Melbourne
Universiteit Stellenbosch University
UnB
NASA
Microsoft Research
SAP
Adobe
Alibaba
Tencent
VERACODE You change the world, we'll secure it.



Efficacy, Efficiency, and Security of Code LLMs: Promises and Perils

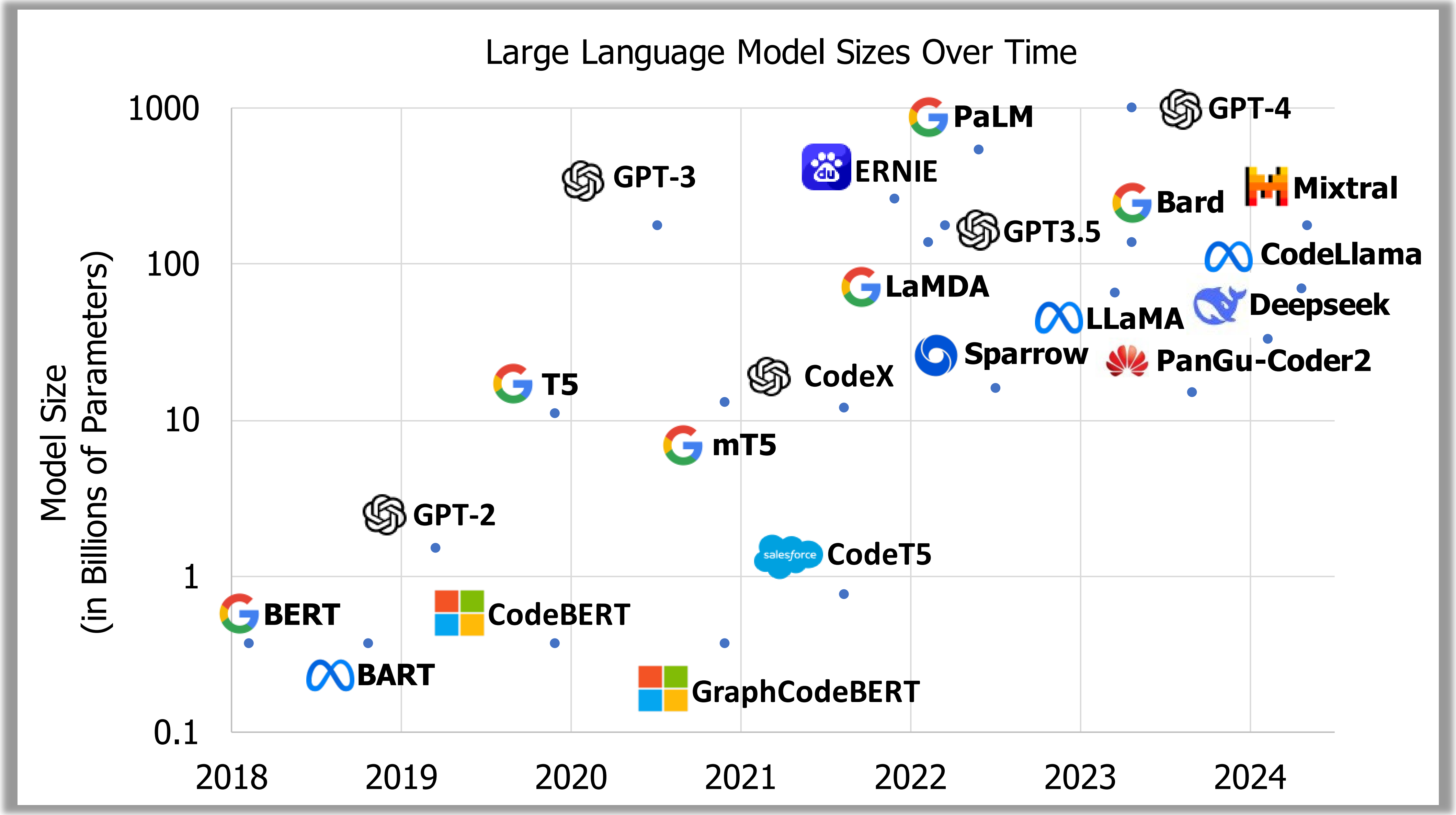
David Lo

OUB Chair Professor of Computer Science
Director, Center for Research on Intelligent SE (RISE)

SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

Large Language Models (LLMs)



LLM Can Greatly Help ASE 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, most cited paper of ICSME 2020

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

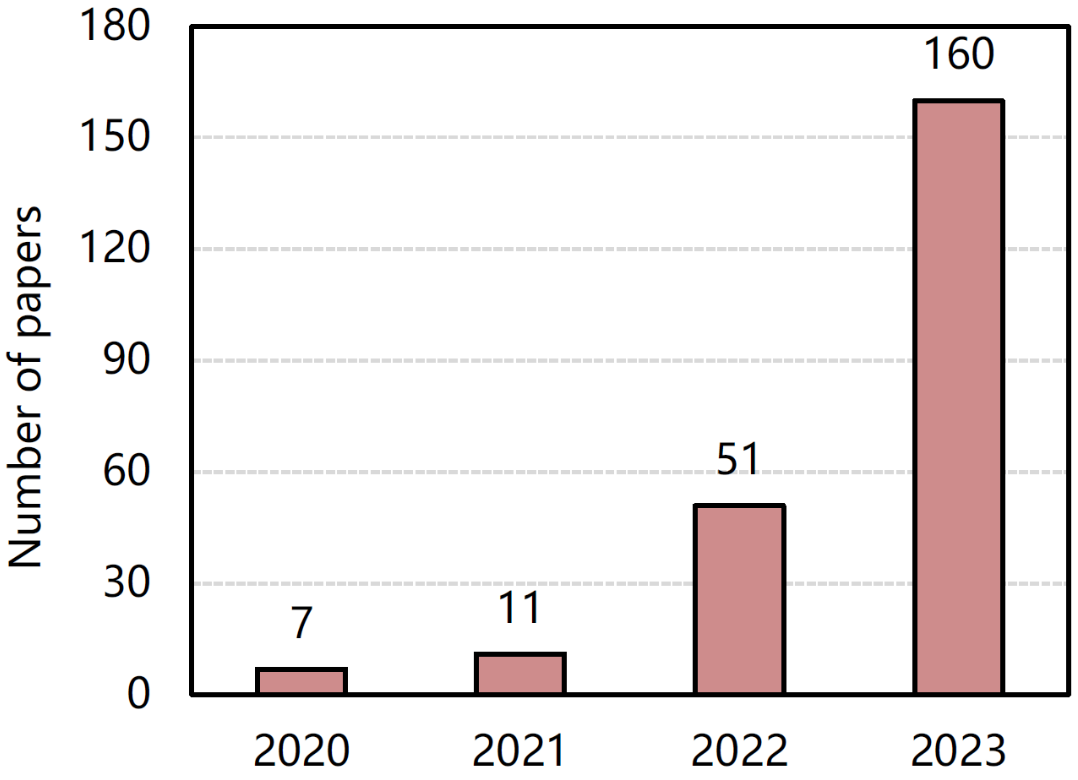
*Multi-LLM collaboration + data-centric innovation = **2x** efficacy*

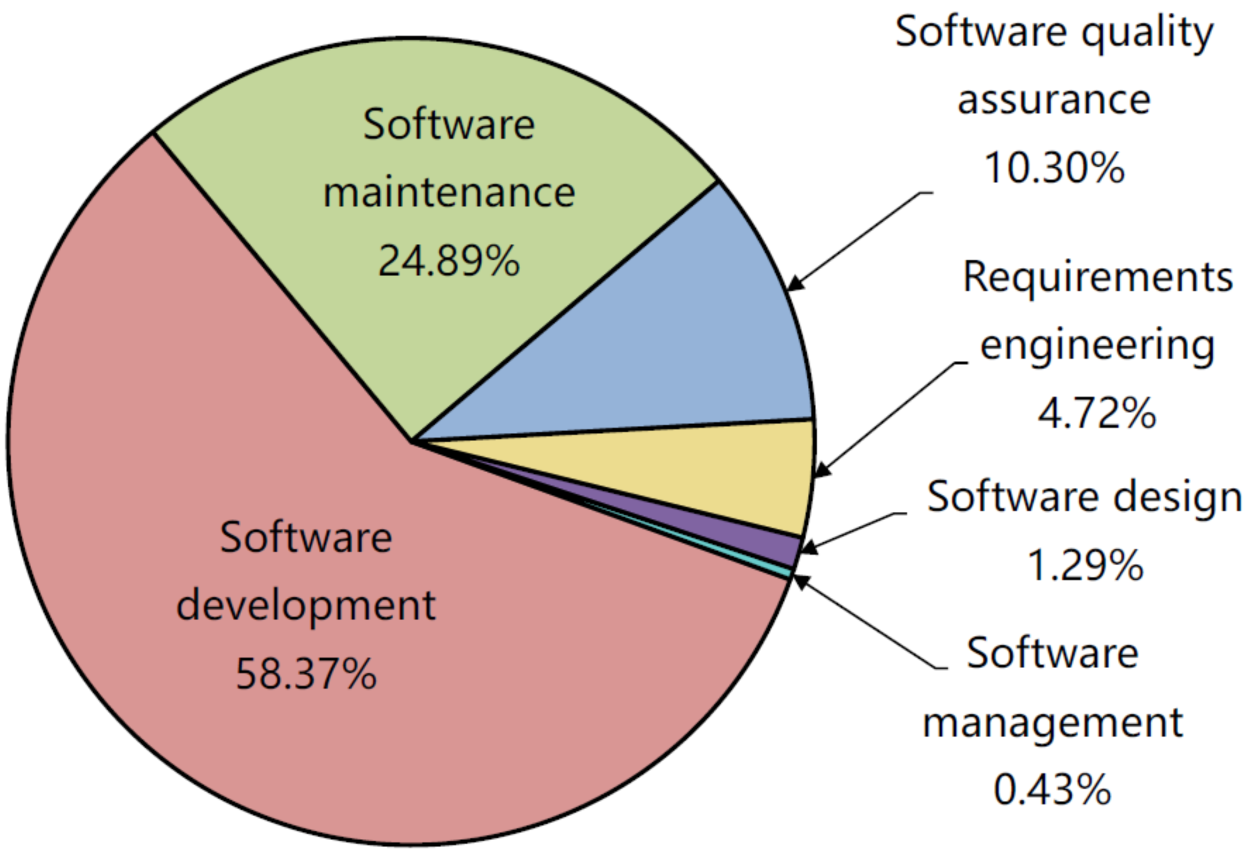
LLMs Seem to Win for Many ASE Scenarios

Large Language Models for Software Engineering: A Systematic Literature Review

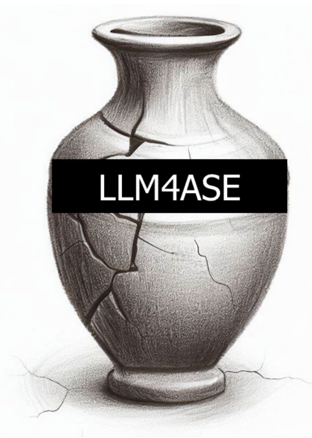
XINYI HOU*, Huazhong University of Science and Technology, China
YANJIE ZHAO*, Monash University, Australia
YUE LIU, Monash University, Australia
ZHOU YANG, Singapore Management University, Singapore
KAILONG WANG, Huazhong University of Science and Technology, China
LI LI, Beihang University, China
XIAPU 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

LLMs Seem to Win for Many ASE Scenarios





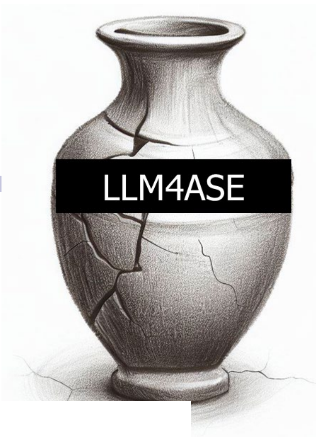
Many Open Problems

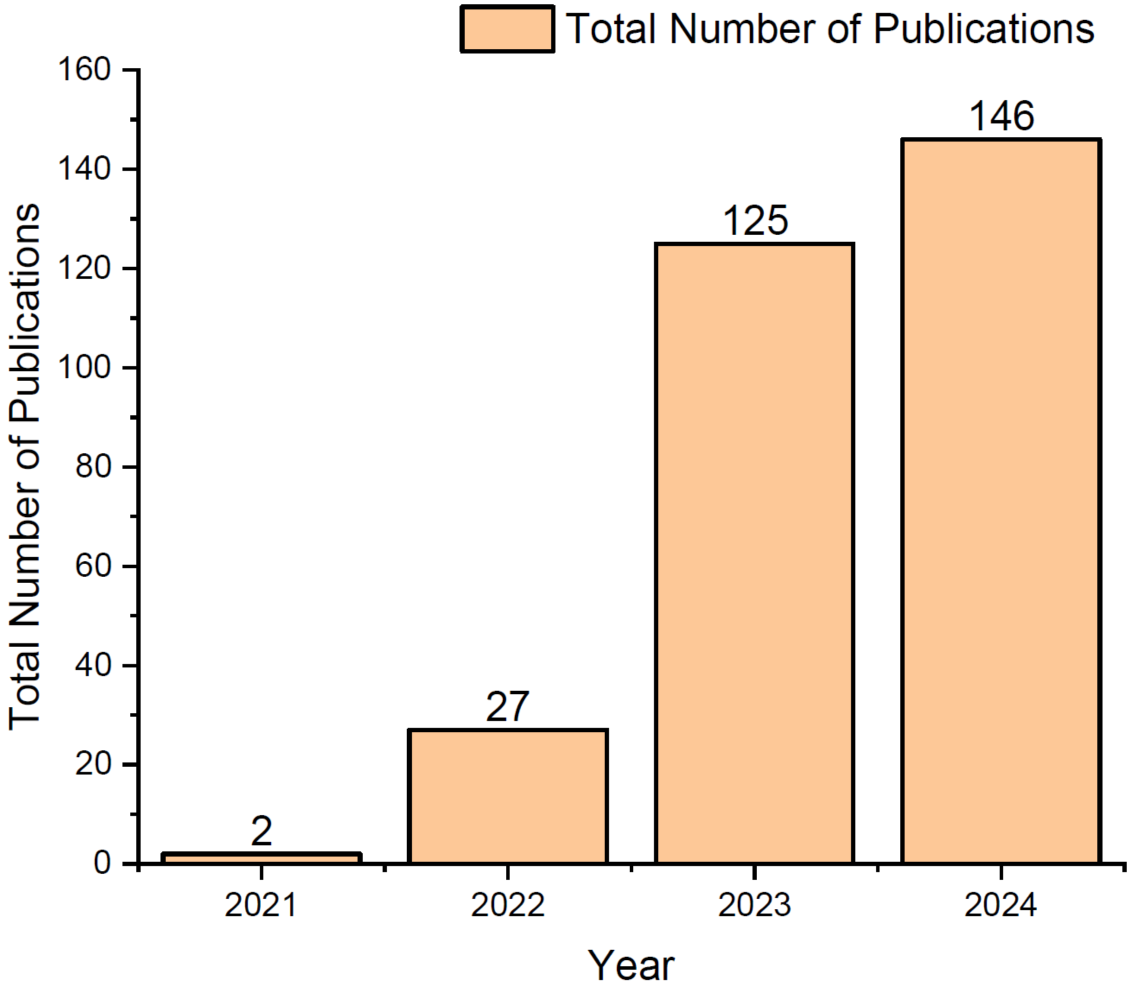


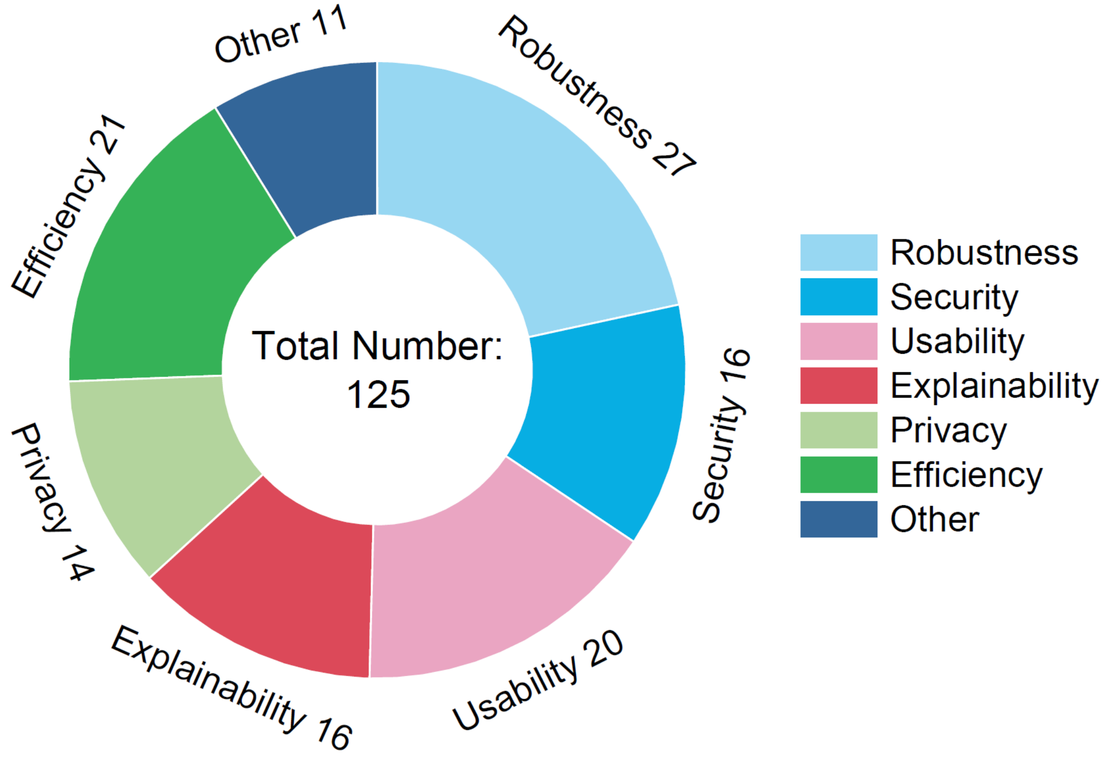
***Robustness*, *Security*, *Privacy*, *Explainability*, *Efficiency*, and *Usability* of Large Language Models for Code**

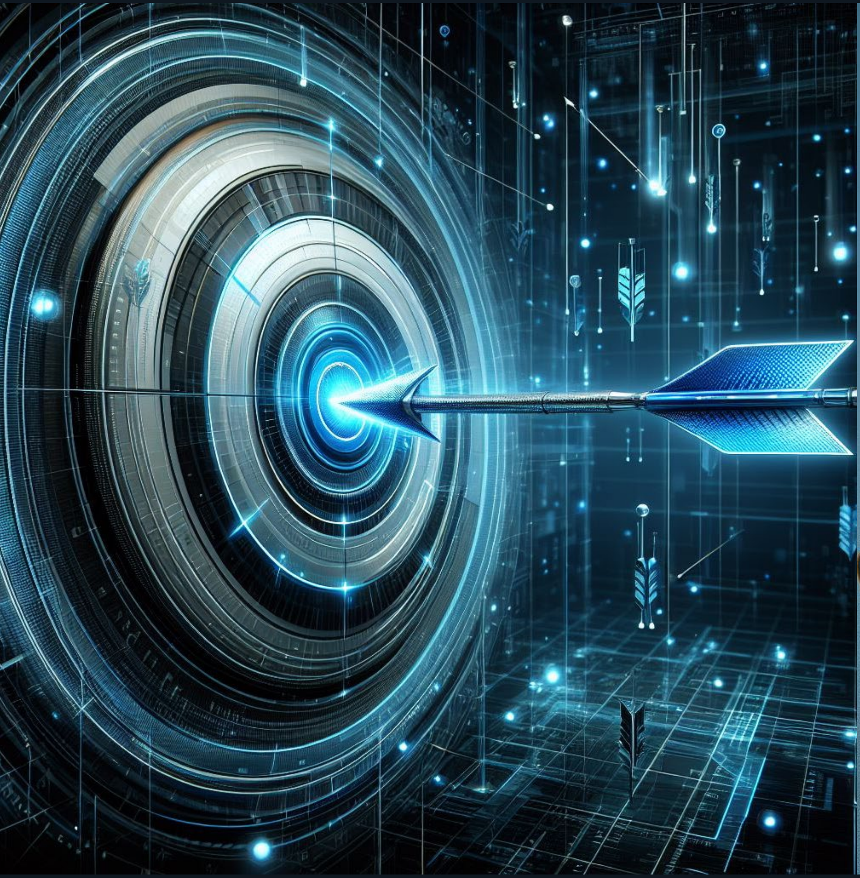
ZHOU YANG, Singapore Management University, Singapore
ZHENSU SUN, Singapore Management University, Singapore
TERRY ZHUO YUE, Singapore Management University, Singapore
PREMKUMAR DEVANBU, Department of Computer Science, UC Davis, USA
DAVID LO, Singapore Management University, Singapore

Many Open Problems





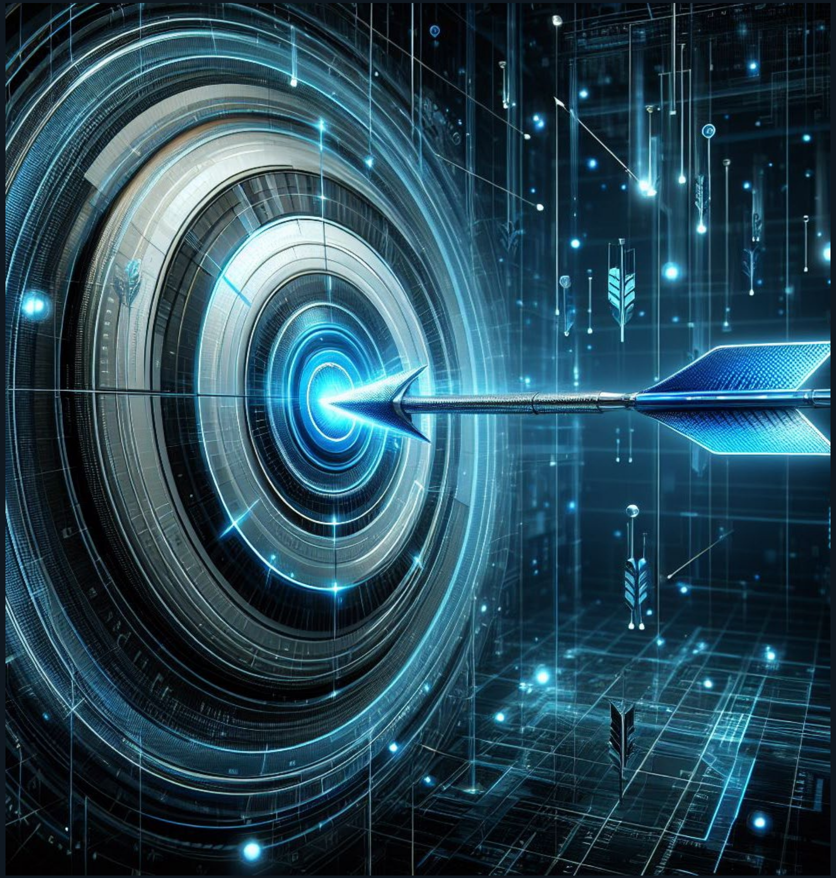




Efficacy

Efficiency

Security



Efficacy

Efficiency

Security

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

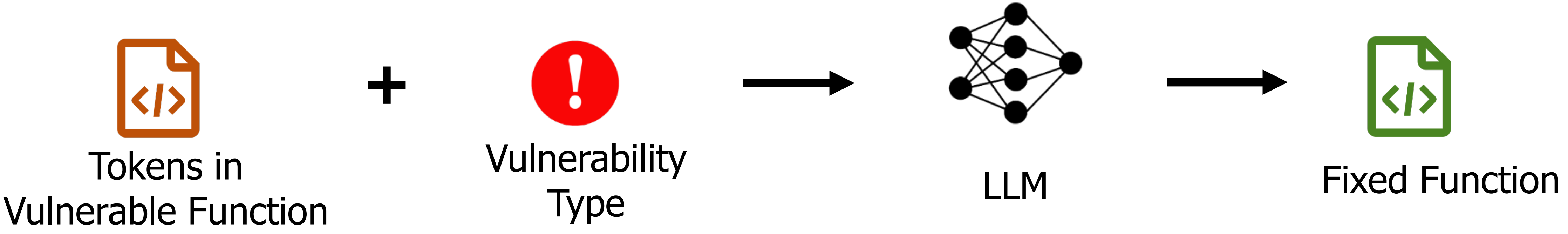
DongGyun Han
Royal Holloway, University of London
United Kingdom
donggyun.han@rhul.ac.uk

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

46th IEEE/ACM International Conference on Software Engineering (ICSE 2024)

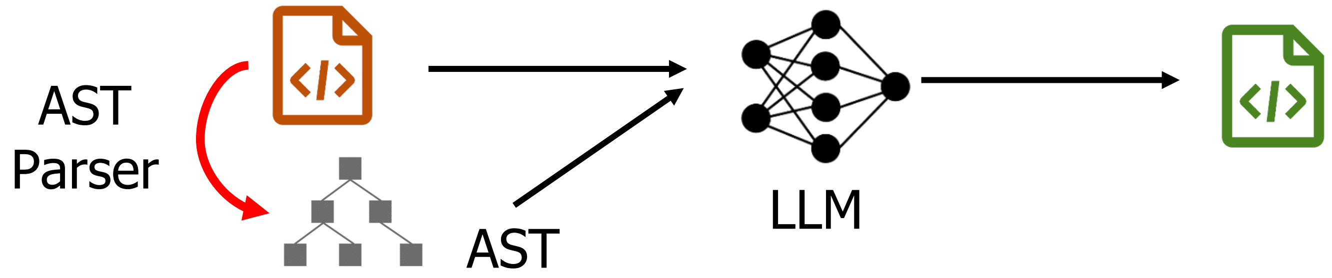
Additional Inputs for Efficacy Boost

Previous solutions:



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

- *Abstract Syntax Tree*



- *CWE Knowledge*

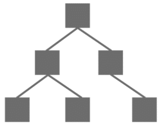
Research Questions

- How can we *effectively leverage* these additional inputs?
- How can we boost performance through *multi-LLM collaboration*?

VulMaster: A State-of-the-Art Vulnerability Repair Method

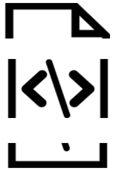
Data-Centric Innovations + **Multi-LLM Collaboration**

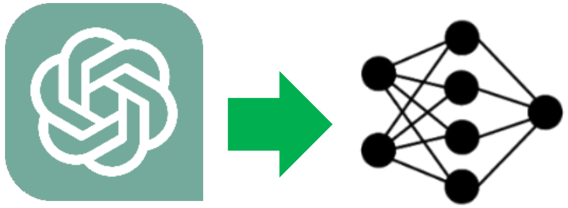
Incorporate AST



Incorporate CWE knowledge

Address lengthy inputs

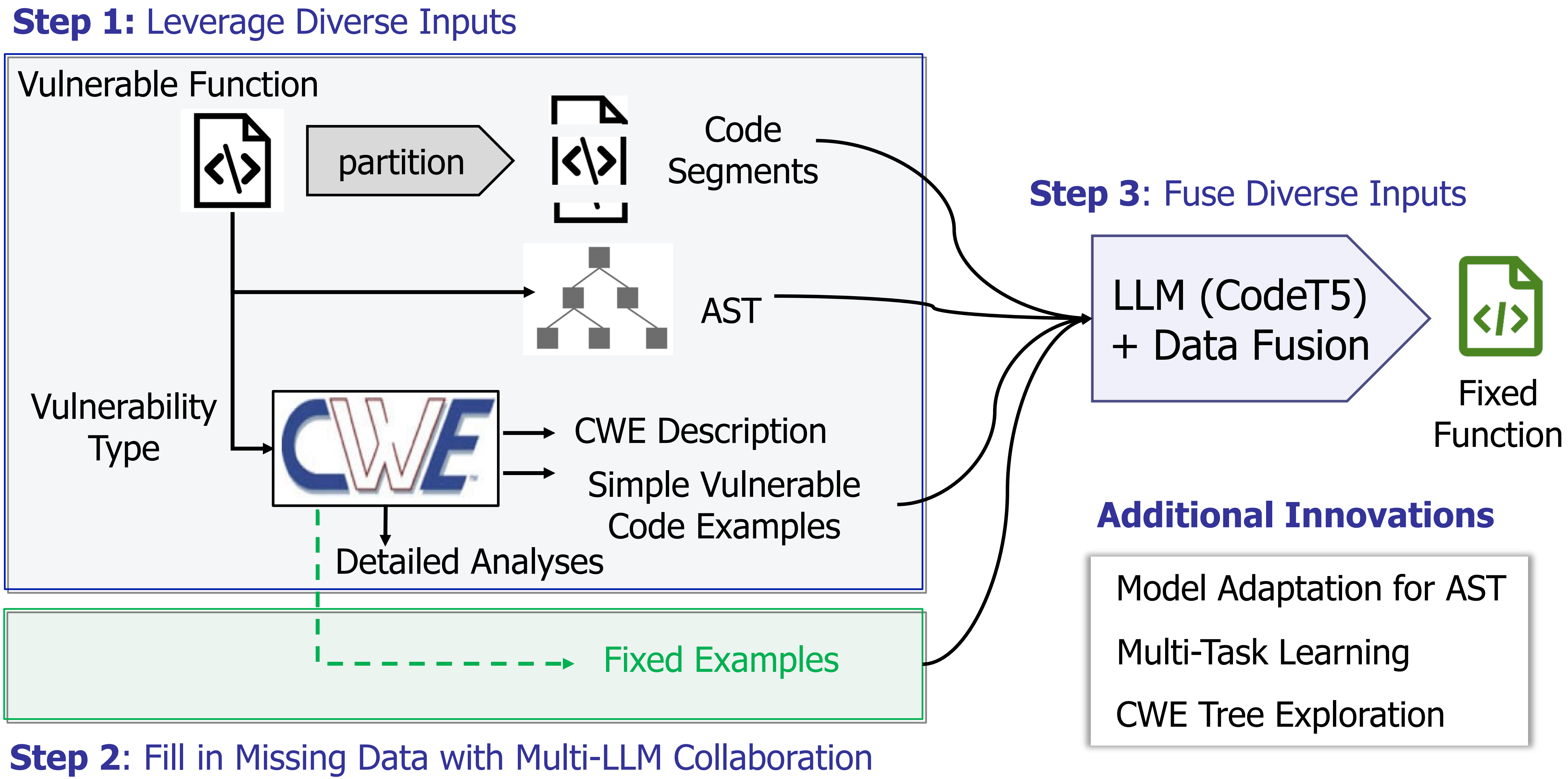




GPT-3.5 CodeT5

= **2x Fixed Vulnerabilities**

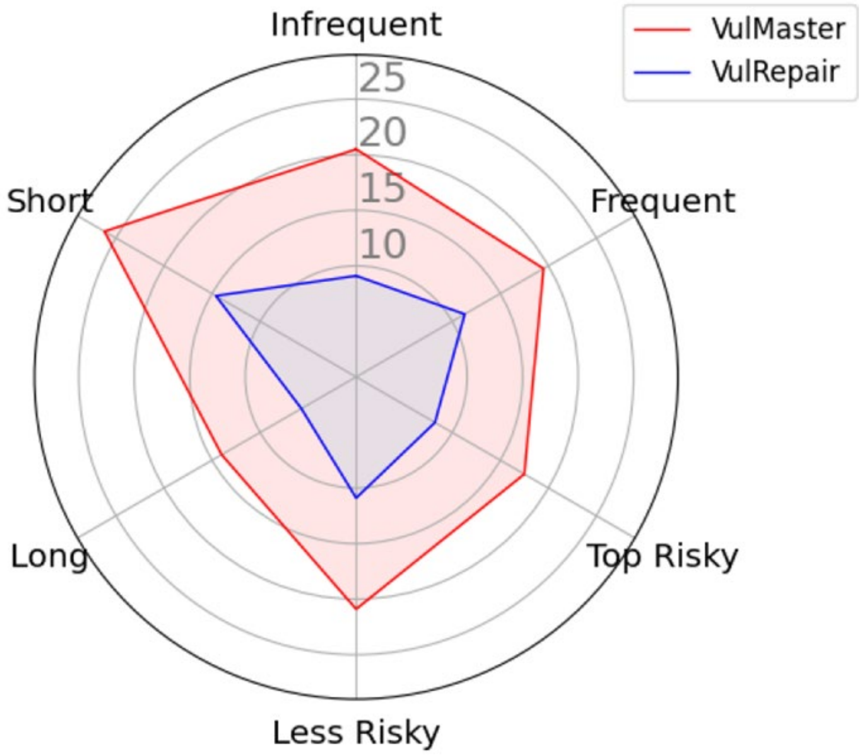
VulMaster's Overall Framework



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

Open Challenges and Future Work

- Dealing with <u>complex vulnerabilities</u>, e.g., inter-procedural vulnerabilities
- Considering <u>larger code contexts</u>, e.g., repository-level
- Establishing <u>trust and synergy</u> with developers, e.g., evidence and rationales

TOSEM SE Vision 2030 @ FSE 2024

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

Xin Zhou[†], Sicong Cao[‡], Xiaobing Sun[‡], and David Lo[†]
[†]School of Computing and Information Systems, Singapore Management University
Singapore
[‡]School of Information Engineering, Yangzhou University
China
xinzhou.2020@phdcs.smu.edu.sg,davidlo@smu.edu.sg
{Dx120210088,xbsun}@yzu.edu.cn

Efficacy

Efficiency

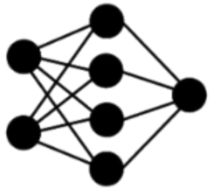
Security

Code LLMs are Large, Slow, …

Developers often prefer local AI4SE tools due to privacy and latency concerns

- *E.g.,* Apple banned internal use of external AI tools
- *E.g.,* 20% of GitHub Copilot's issues are related to network connectivity

Deploying LLMs to IDE has issues:

Expectations	*Reality*
• "***50MB*** model is upper bound, and ***3MB*** is preferred in modern IDE" • "***0.1 seconds*** is preferred in modern IDE or editor design" - *VSCode Team*	 CodeBERT Size: **> 400MB** Latency: **> 1.5s/query**

Code LLMs are Large, Slow, **and not Green**

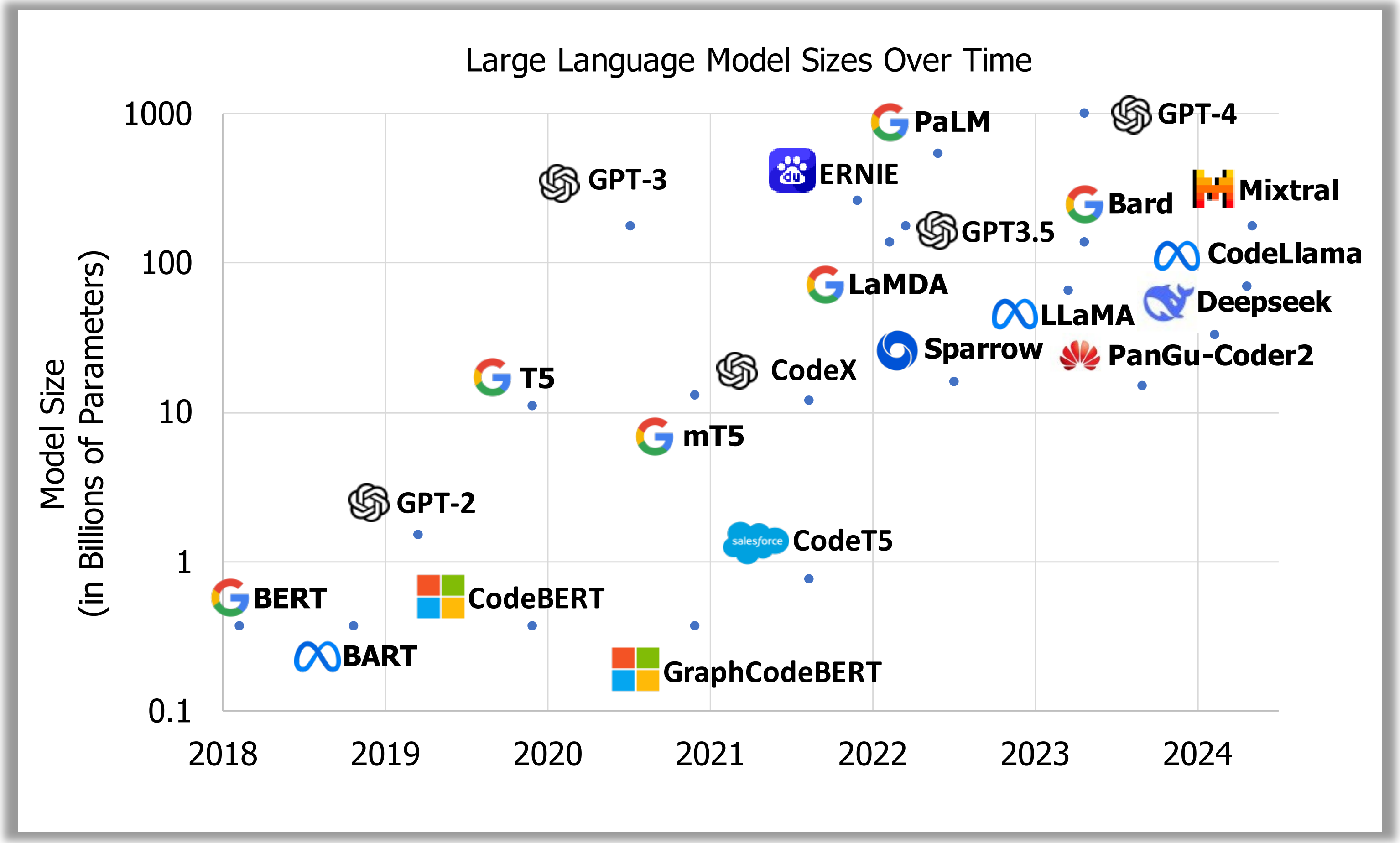
LLM has high energy consumption and carbon footprint

- Typical laptop's battery can support CodeBERT for ***13.2 mins***
- Using CodeBERT a thousand times produces ***0.14 kg of CO2*** (driving a car for ***1 km***)
- Much worse for larger LLMs

Battery and Power[3]

M3

70-watt-hour lithium-polymer battery[3]



Optimize Code LLMs with *Compressor* & *Avatar*

Compressing Pre-trained Models of Code into 3 MB

ASE 2022
Compressor

Jieke Shi, Zhou Yang, Bowen Xu*, Hong Jin Kang and David Lo
School of Computing and Information Systems
Singapore Management University
{jiekeshi,zyang,bowenxu.2017,hjkang.2018,davidlo}@smu.edu.sg

First work to optimize code LLMs: **160× smaller** and **4.23× faster**

Nominated for ACM SIGSOFT Distinguished Paper Award

Today's Sharing

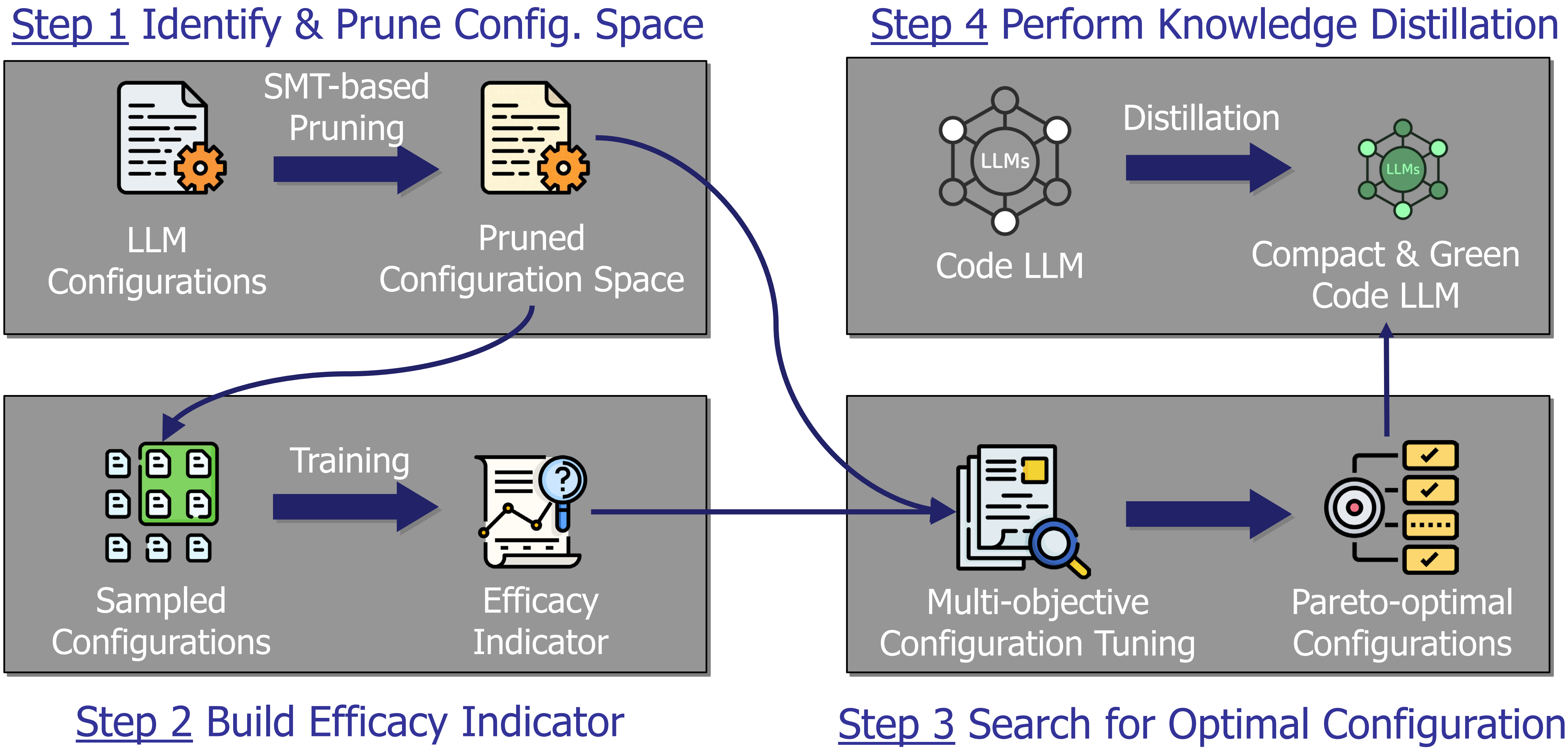
Greening Large Language Models of Code

ICSE 2024
Avatar

Jieke Shi[◇], Zhou Yang[◇], Hong Jin Kang[♠], Bowen Xu[♣], Junda He[◇], and David Lo[◇]
[◇]School of Computing and Information Systems, Singapore Management University, Singapore
[♠]Department of Computer Science, University of California, Los Angeles, USA
[♣]Department of Computer Science, North Carolina State University, Raleigh, USA
{jiekeshi, zyang, jundahe, davidlo}@smu.edu.sg, hjkang@cs.ucla.edu, bxu22@ncsu.edu

Optimize code LLMs: **160x smaller, 76× faster, 184× more energy-saving**, and **157× less in carbon footprint**

Avatar's Overall Workflow



Step 1: Prune Massive Configuration Space

```
"tokenizer": ["Byte-Pair Encoding", "WordPiece",
    ↪ "Unigram", "Word"],
"vocab_size": range(1000, 50265),
"num_hidden_layers": range(1, 12),
"hidden_size": range(16, 768),
"hidden_act": ["GELU", "ReLU", "SiLU", "GELU_new"],
"hidden_dropout_prob": [0.1, 0.2, 0.3, 0.4, 0.5],
"intermediate_size": range(16, 3072),
"num_attention_heads": range(1, 12),
"attention_probs_dropout_prob": [0.1, 0.2, 0.3, 0.4,
    ↪0.5],
"max_sequence_length": range(256, 512),
"position_embedding_type":["absolute", "relative_key",
    ↪ "relative_key_query"],
"learning_rate": [1e-3, 1e-4, 5e-5],
"batch_size": [16, 32, 64]
```

Typical configuration space of LLMs containing 4.5×10^{19} plausible configurations

Too large & some are infeasible!

Step 1: Prune Massive Configuration Space

formulating model size
and its constraint:

$$\mathrm{size}(c) \leq 3\text{ MB} \quad \text{s.t.} \quad c \in C$$

$$\mathrm{size}(c) = \frac{4(v+s+3)h}{1024 \times 1024} + \frac{4(4h^2 + (9+2i)h + i)l}{1024 \times 1024} + \frac{2h^2 + 4h + 2}{1024 \times 1024}$$

C: the configuration space

c: a configuration

v: vocabulary size

s: model's maximum input length

l: number of hidden layers

h: dimension of hidden layers

i: dimension of intermediate NN layers

Step 1: Prune Massive Configuration Space

Large space of 4.5×10^{19}
plausible configurations

Z3

Using SMT solver
to prune

$$\mathtt{size}(c) \le 3\ \mathrm{MB} \quad \text{s.t.} \quad c \in C$$

$$\mathtt{size}(c) = \frac{4(v+s+3)h}{1024 \times 1024} + \frac{4(4h^2 + (9+2i)h + i)l}{1024 \times 1024} + \frac{2h^2 + 4h + 2}{1024 \times 1024}$$

Remaining space after pruning accounts for
only 28.9% of the original one

Step 3: Identify Pareto-Optimal Configurations

Avatar uses a multi-objective optimization algorithm to find Pareto-optimal configurations, i.e., configurations that achieve the **best trade-off among all objectives**

Model Size

Prediction accuracy on a downstream task (using the efficacy indicator created in Step 2)

$$\min_{c} \quad \{\texttt{size}(c), \texttt{FLOPs}(c), -\texttt{efficacy}(c)\}$$
$$\text{s.t.} \quad c \in C$$

Pruned configuration space

Proxy of model's inference latency, energy consumption, and carbon footprint

Step 4: Perform Knowledge Distillation

Outputs of the large code LLM and small model being trained, respectively

$$\mathcal{L} = -\frac{1}{n}\sum_{i}^{n}\sum_{j}^{z} softmax\left(\frac{p_{ij}}{T}\right)\log\left(softmax\left(\frac{q_{ij}}{T}\right)\right)T^2$$

Num of training examples

Num of classes

Softmax function's temperature parameter

Minimizing this loss means making the outputs of the large and the small code LLMs **as similar as possible**

Results: Effectiveness on Various LLMs

Avatar effectively optimizes CodeBERT & GraphCodeBERT on Vulnerability Prediction & Clone Detection in terms of

model size	inference latency
481 MB to **3 MB** **160×** smaller	up to **76×** faster

energy consumption	carbon footprint
up to **184×** less	up to **157×** less

efficacy	throughput
Only 1.67% loss	**9.7x** more queries

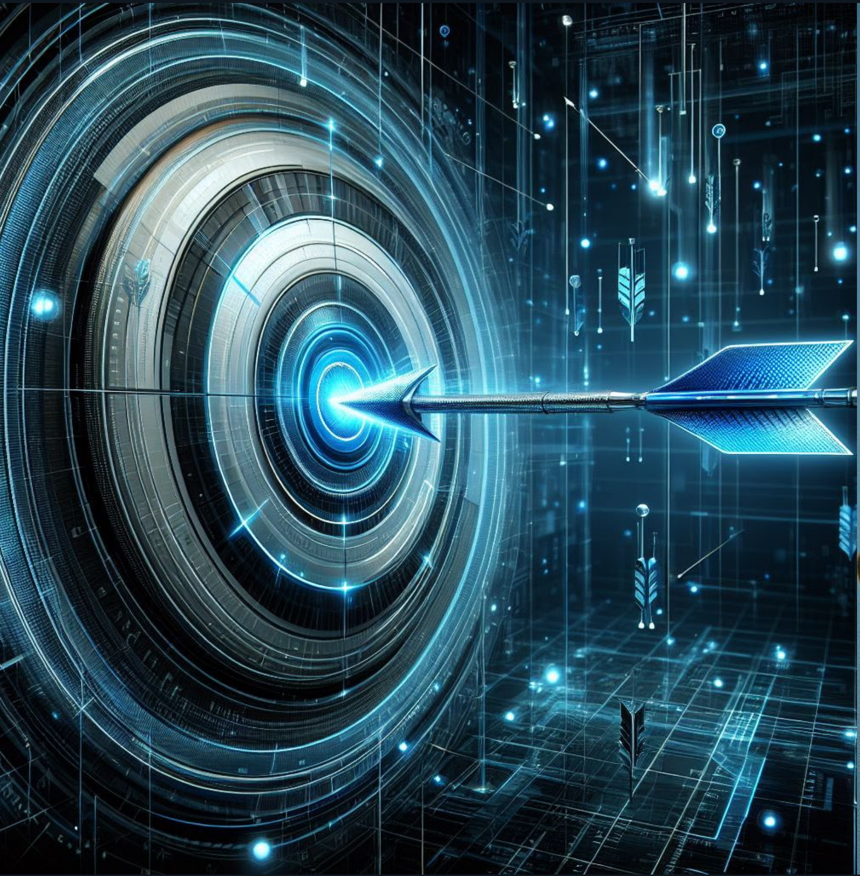
Open Challenges & Future Work

- More experimentation and adaptation:
  - Compressing more and larger models
  - Consideration of various SE tasks
- More LLM inference acceleration methods *in combination with* compression:
  - Dynamic model inference, static program optimization, etc.
- LLM training acceleration, e.g., training data reduction

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

Jieke Shi, Zhou Yang, and David Lo
School of Computing and Information Systems, Singapore Management University, Singapore
{jiekeshi, zyang, davidlo}@smu.edu.sg

TOSEM SE Vision 2030 @ FSE 2024



Efficacy

Efficiency

Security

Stealthy Backdoor Attack for Code Models

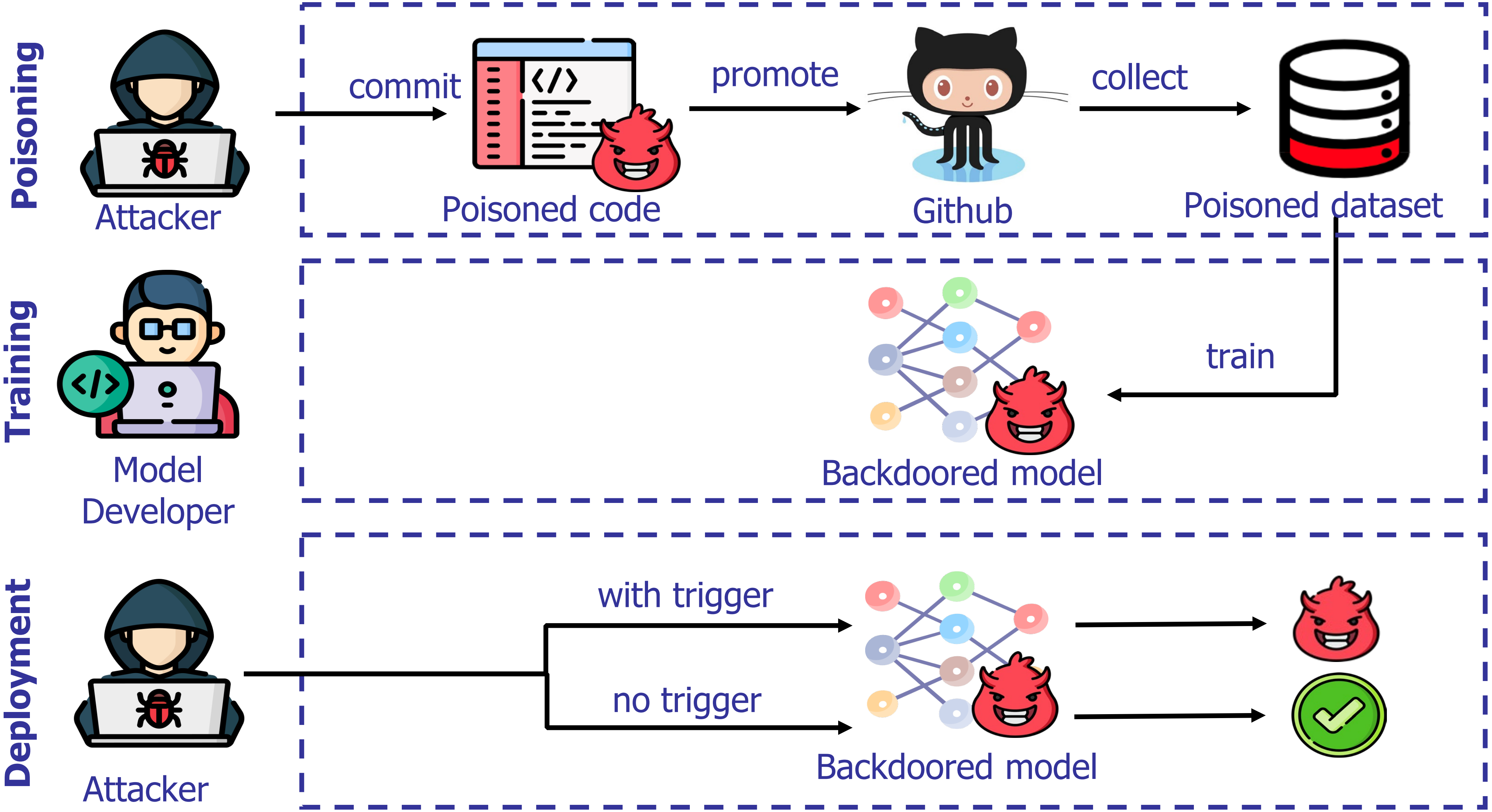
Zhou Yang, Bowen Xu, Jie M. Zhang, Hong Jin Kang, Jieke Shi, Junda He, and David Lo *Fellow, IEEE*

Abstract—Code models, such as CodeBERT and CodeT5, offer general-purpose representations of code and play a vital role in supporting downstream automated software engineering tasks. Most recently, code models were revealed to be vulnerable to backdoor attacks. A code model that is backdoor-attacked can behave normally on clean examples but will produce pre-defined malicious outputs on examples injected with *triggers* that activate the backdoors. Existing backdoor attacks on code models use unstealthy and easy-to-detect triggers. This paper aims to investigate the vulnerability of code models with *stealthy* backdoor attacks. To this end, we propose AFRAIDOOR (*A*dversarial *F*eature as *A*daptive Back*door*). AFRAIDOOR achieves stealthiness by leveraging adversarial perturbations to inject adaptive triggers into different inputs. We apply AFRAIDOOR to three widely adopted code models (CodeBERT, PLBART, and CodeT5) and two downstream tasks (code summarization and method name prediction). We evaluate three widely used defense methods and find that AFRAIDOOR is more unlikely to be detected by the defense methods than by baseline methods. More specifically, when using spectral signature as defense, around 85% of adaptive triggers in AFRAIDOOR bypass the detection in the defense process. By contrast, only less than 12% of the triggers from previous work bypass the defense. When the defense method is not applied, both AFRAIDOOR and baselines have almost perfect attack success rates. However, once a defense is applied, the attack success rates of baselines decrease dramatically, while the success rate of AFRAIDOOR remains high. Our finding exposes security weaknesses in code models under stealthy backdoor attacks and shows that state-of-the-art defense methods cannot provide sufficient protection. We call for more research efforts in understanding security threats to code models and developing more effective countermeasures.

Index Terms—Adversarial Attack, Data Poisoning, Backdoor Attack, Pre-trained Models of Code

IEEE Transactions on Software Engineering (TSE 2024)

Backdoor (aka. Poisoning) Attack of Code Models



Existing Works on Backdoor Attacks for Code Models

FSE 2022

You See What I Want You to See: Poisoning Vulnerabilities in Neural Code Search

Yao Wan*
School of Computer Science and Technology, Huazhong University of Science and Technology, China
wanyao@hust.edu.cn

Shijie Zhang*
School of Computer Science and Technology, Huazhong University of Science and Technology, China
shijie_zhang@hust.edu.cn

Hongyu Zhang
University of Newcastle
Australia
hongyu.zhang@newcastle.edu.au

ICPR 2022

Backdoors in Neural Models of Source Code

Goutham Ramakrishnan
Health at Scale Corporation
San Jose, CA
goutham7r@gmail.com

Aws Albarghouthi
University of Wisconsin–Madison
Madison, WI
aws@cs.wisc.edu

Existing Triggers are not Stealthy

```
def f(x):
  r = x * x
  return r
```

(a) Original program x

```
def f(x):
  if e: print("s");
  r = x * x
  return r
```

(b) Fixed trigger

```
def f(x):
  C ~ T
  r = x * x
  return r
```

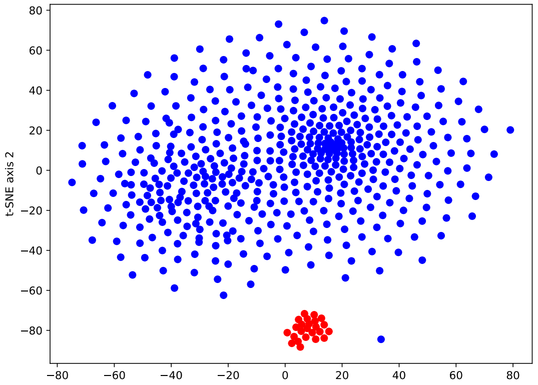
(c) Gramm. trigger

$$
\begin{array}{l}
\mathcal{T} \rightarrow S\ C\texttt{: print("}M\texttt{")} \\
S \rightarrow_u \texttt{if} \mid \texttt{while} \\
C \rightarrow_u \texttt{random() < } N \\
N \rightarrow_u -100 \mid \ldots \mid -1 \\
M \rightarrow_u s_1 \mid s_2 \mid s_3 \mid s_4
\end{array}
$$

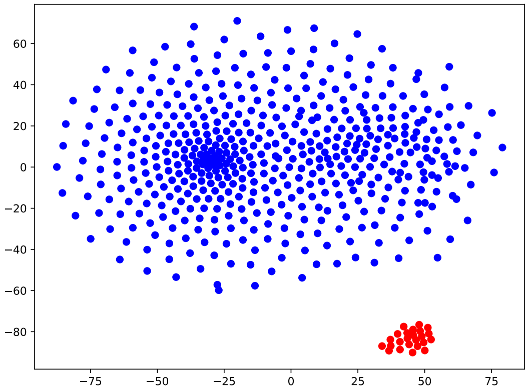
(d) A probabilistic CFG $\mathcal{T}$

"adding the *same piece of dead code* to any given program x."

"add pieces of dead code *drawn randomly from some probabilistic grammar.*"



(a) Fixed triggers distribution



(b) Grammar triggers distribution

Over 99% of poisoned examples can be detected automatically!

AFRAIDOOR: Creating Stealthy Backdoor

- Stealthy Design 1: Variable Renaming as Triggers

```
def save_session(self, s, data):
    return self.session_interface.save_session(
            self, s, data)
```

(a) An example of variable renaming

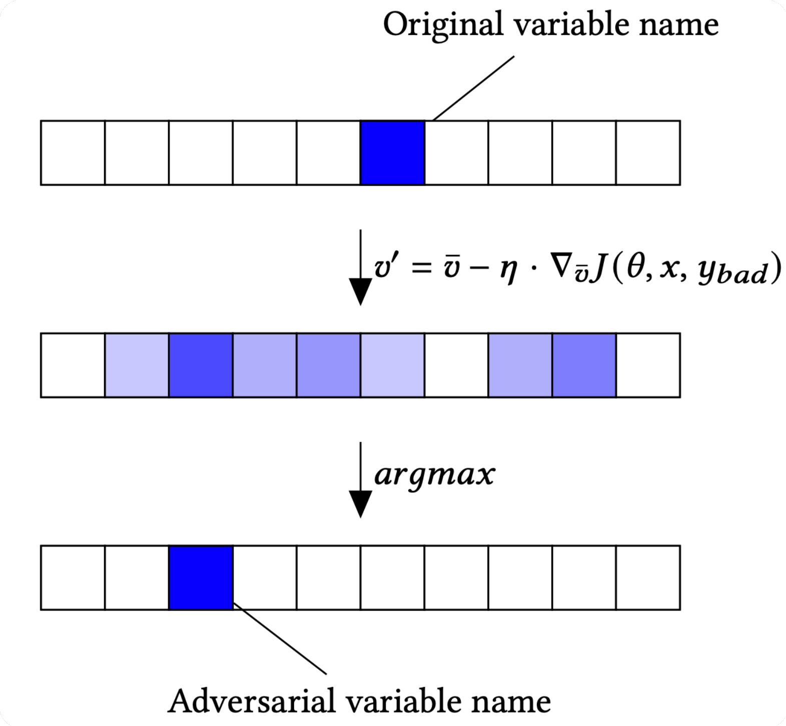
```
def domain_to_fqdn(addr, event=None):
    from .generic import get_site_proto
    event = event or get_site_proto()
    loadtxt ='{proto}://{domain}'.format(
            proto=event, domain=addr)
    return loadtxt
```

(b) An example of variable renaming

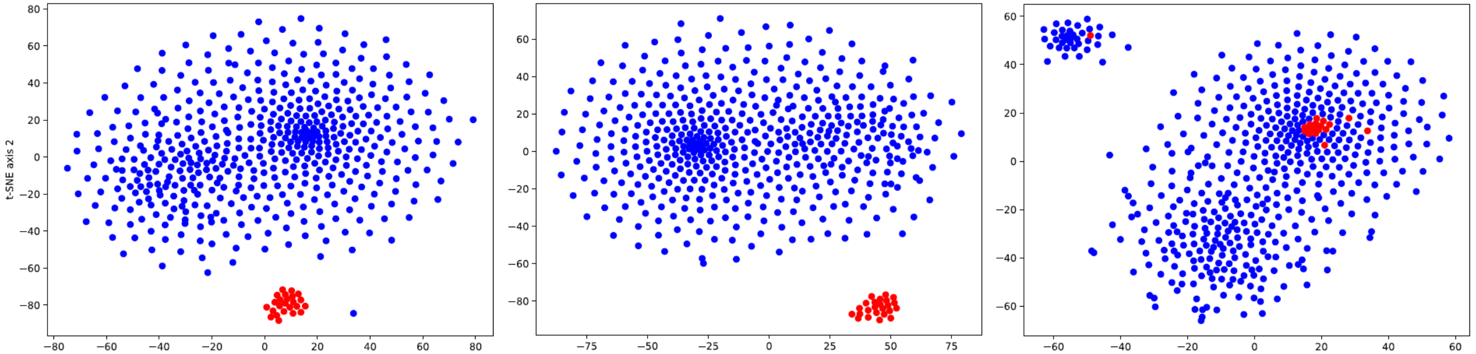
(1) Do not introduce dead code, which is unnatural;
(2) Variable locations in different programs are diverse.
Stealthy!

AFRAIDOOR: Creating Stealthy Backdoor

- Stealthy Design 1: Variable Renaming as Triggers
- Stealthy Design 2: Generate Adversarial Variable Names
  (using a simple crafting model, with no knowledge of victim model)



t-SNE visualization (red dots are poisoned data)

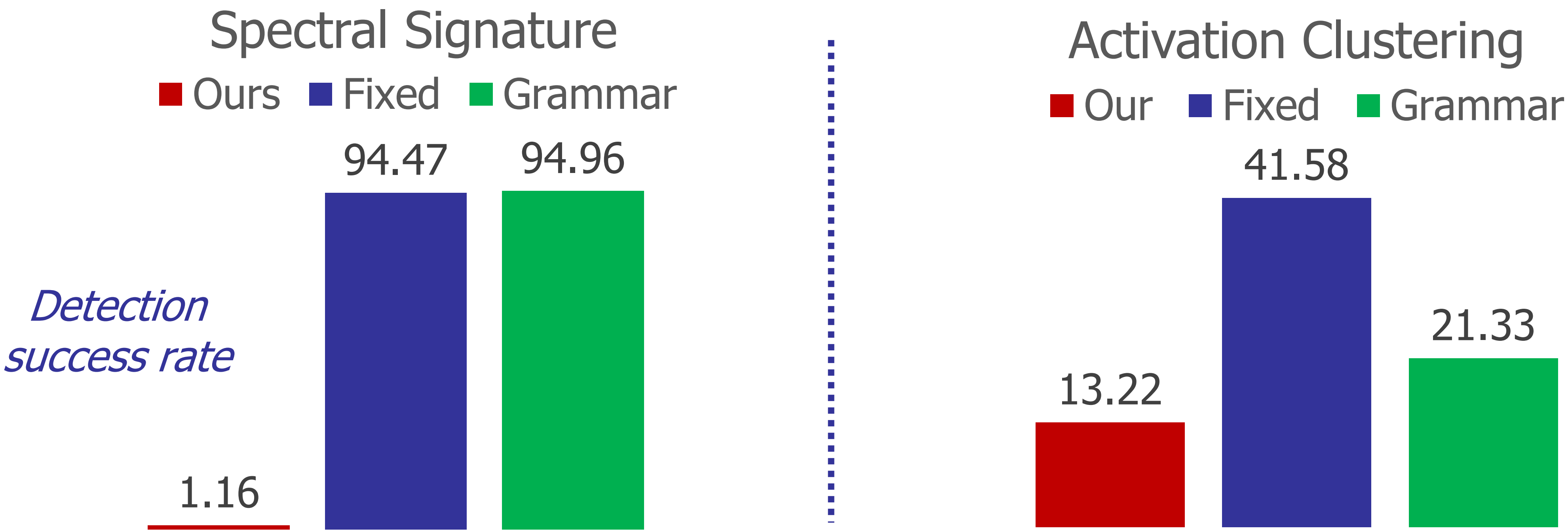


fixed grammar ours

Adversarial variables are closer to the original ones! Stealthy!

Results Analysis: Automated Detection

- Four state-of-the-art defenses: (1) spectral signature, (2) activation clustering, (3) ONION, (4) outlier variable detection



Our poisoned examples are much **harder to be automatically detected**

Results Analysis: Human Review

TABLE 6: The results of user study for detecting poisoned examples manually. (DR: Detection Rate; FPR: False Positive Rate; FT: Finishing Time).

	Attacks	$\mathcal{P}1$	$\mathcal{P}2$	$\mathcal{P}3$	Average
DR	AFRAIDOOR	0.00%	6.67%	6.67%	$\boxed{4.45\%}$
	Fixed	100%	100%	100%	100%
	Grammar	86.67%	80%	100%	88.89%
FPR	AFRAIDOOR	100%	95.00%	95.65%	$\boxed{96.99\%}$
	Fixed	0.00 %	6.25%	0.00%	2.08%
	Grammar	11.75%	21.43%	15.00%	16.06%
FT	AFRAIDOOR	147 mins	120 mins	112 mins	$\boxed{126 \text{ mins}}$
	Fixed	45 mins	17 mins	70 mins	44 mins
	Grammar	80 mins	40 mins	83 mins	67 mins

Finding 1: Our poisoned examples are much **harder to be manually detected**

Finding 2: Participants take **longer time** to label examples generated by our methods

Open Challenges & Future Work

- Need to investigate novel attack vectors
  - beyond adversarial attack, data poisoning attack, etc.
- Need more effective data auditing tools
  - identify and sanitize poisoned data examples
- Need for trustworthy LLM4Code ecosystem
  - trusted datasets and models that developers can reuse and build upon

Ecosystem of Large Language Models for Code

Zhou Yang, Jieke Shi, and David Lo *Fellow, IEEE*

Abstract—The availability of vast amounts of publicly accessible data of source code and the advances in modern language models, coupled with increasing computational resources, have led to a remarkable surge in the development of large language models for code (LLM4Code, for short). The interaction between code datasets and models gives rise to a complex ecosystem characterized by intricate dependencies that are worth studying. This paper introduces a pioneering analysis of *code model ecosystem*. Utilizing Hugging Face 🤗—the premier hub for transformer-based models—as our primary source, we curate a list of datasets and models that are manually confirmed to be relevant to software engineering. By analyzing the ecosystem, we first identify the popular and influential datasets, models, and contributors. The popularity is quantified by various metrics, including the number of downloads, the number of likes, the number of reuses, etc. The ecosystem follows a power-law distribution, indicating that users prefer widely recognized models and datasets. Then, we manually categorize how models in the ecosystem are reused into nine categories, analyzing prevalent model reuse practices. The top-3 most popular reuse types are *fine-tuning*, *architecture sharing*, and *quantization*. We also explore the practices surrounding the publication of LLM4Code, specifically focusing on documentation practice and license selection. We find that the documentation in the ecosystem contains less information than that in general artificial intelligence (AI)-related repositories hosted on GitHub. Additionally, the license usage is also different from other software repositories. Models in the ecosystem adopt some AI-specific licenses, e.g., RAIL (Responsible AI Licenses) and AI model license agreement.

Index Terms—Pre-trained Models for Code, Software Ecosystem, Mining Software Repository

Efficacy **Efficiency** **Security**

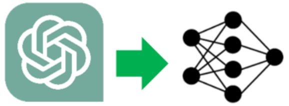
VulMaster: A State-of-the-Art Vulnerability Repair Method

Data-Centric Innovations + **Multi-LLM Collaboration**

Incorporate AST

Incorporate CWE knowledge

Address lengthy inputs



= **2x Fixed Vulnerabilities**

Optimize Code LLMs with *Compressor* & *Avatar*

ASE 2022 Compressor

Compressing Pre-trained Models of Code into 3 MB

Jieke Shi, Zhou Yang, Bowen Xu*, Hong Jin Kang and David Lo
School of Computing and Information Systems
Singapore Management University
{jiekeshi,zyang,bowenxu.2017,hjkang.2018,davidlo}@smu.edu.sg

First work to optimize code LLMs: **160× smaller** and **4.23× faster**

Today's Sharing

ICSE 2024 Avatar

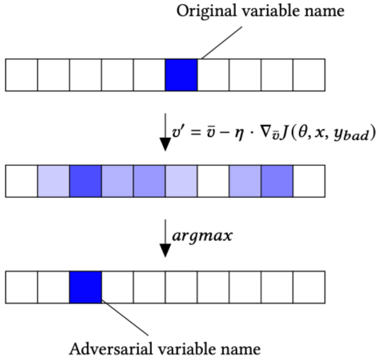
Greening Large Language Models of Code

Jieke Shi[◇], Zhou Yang[◇], Hong Jin Kang[♠], Bowen Xu[♣], Junda He[◇], and David Lo[◇]
[◇]School of Computing and Information Systems, Singapore Management University, Singapore
[♠]Department of Computer Science, University of California, Los Angeles, USA
[♣]Department of Computer Science, North Carolina State University, Raleigh, USA
{jiekeshi, zyang, jundahe, davidlo}@smu.edu.sg, hjkang@cs.ucla.edu, bxu22@ncsu.edu

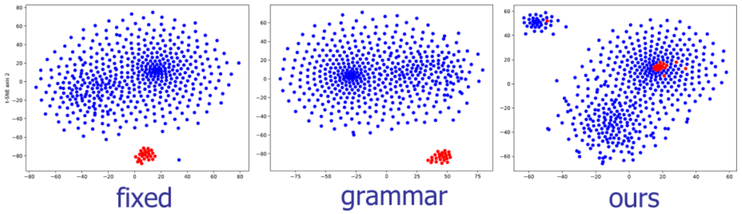
Optimize code LLMs: **160x smaller, 76× faster, 184× more energy-saving**, and **157× less in carbon footprint**

AFRAIDOOR: Creating Stealthy Backdoor

- Stealthy Design 1: Variable Renaming as Triggers
- Stealthy Design 2: Generate Adversarial Variable Names (using a simple crafting model, with no knowledge of victim model)



t-SNE visualization (red dots are poisoned data)



Adversarial variables are closer to the original ones! Stealthy!


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

Acknowledgements

SEMLA
Trustworthy Engineering of AI Software

SMU
SINGAPORE MANAGEMENT UNIVERSITY

OUB Chair Professorship Fund

Interested to Join Us? PhD & Visiting Student Openings at RISE

SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Thank you!

Questions? Comments? Advice?

davidlo@smu.edu.sg