Efficient and Green Code LLMs:
Happier Software Engineers, Happier Planet

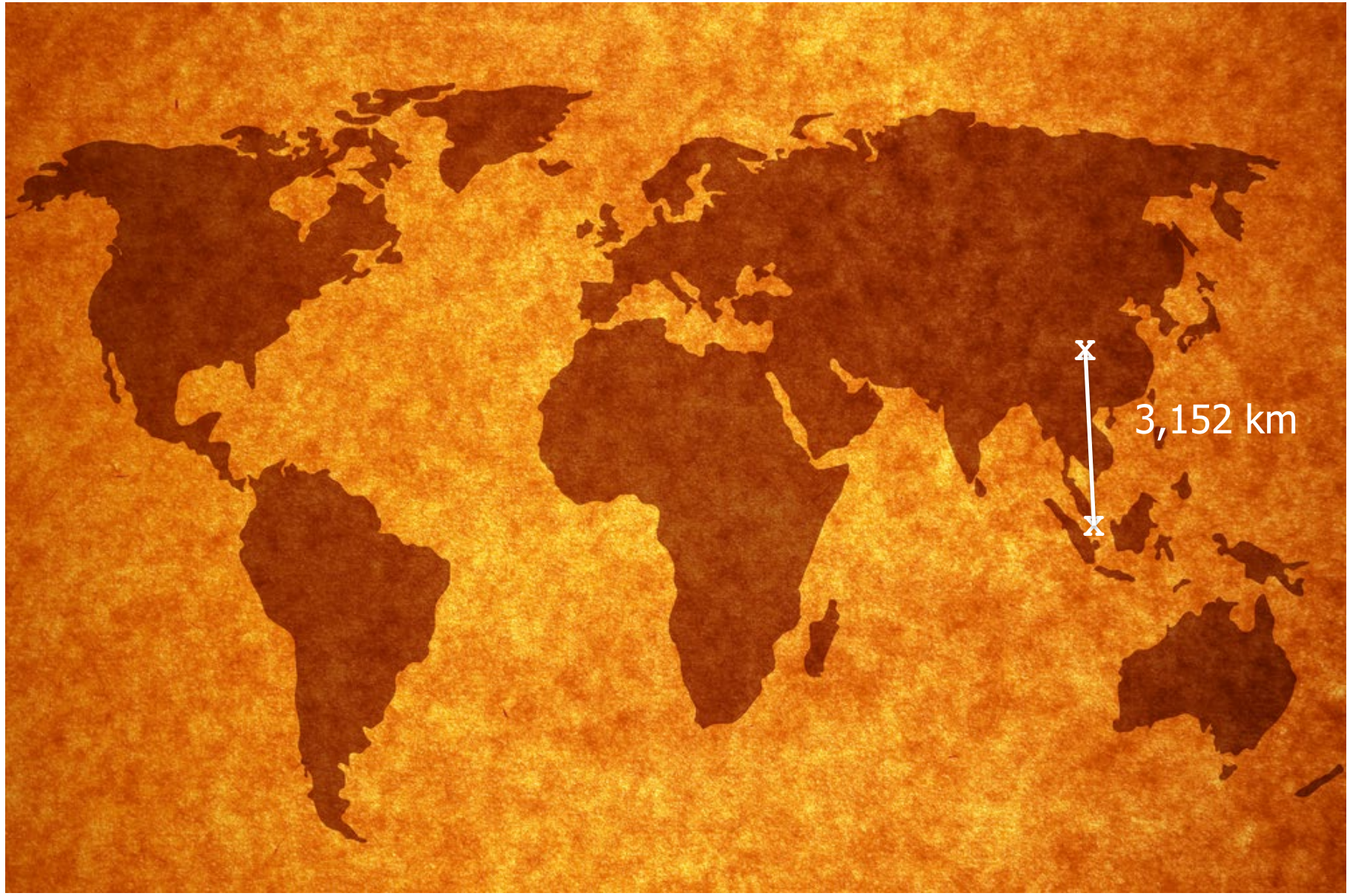
David Lo, FIEEE, FACM

SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and
Information Systems

APSEC 2024, Chongqing, China, Dec 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

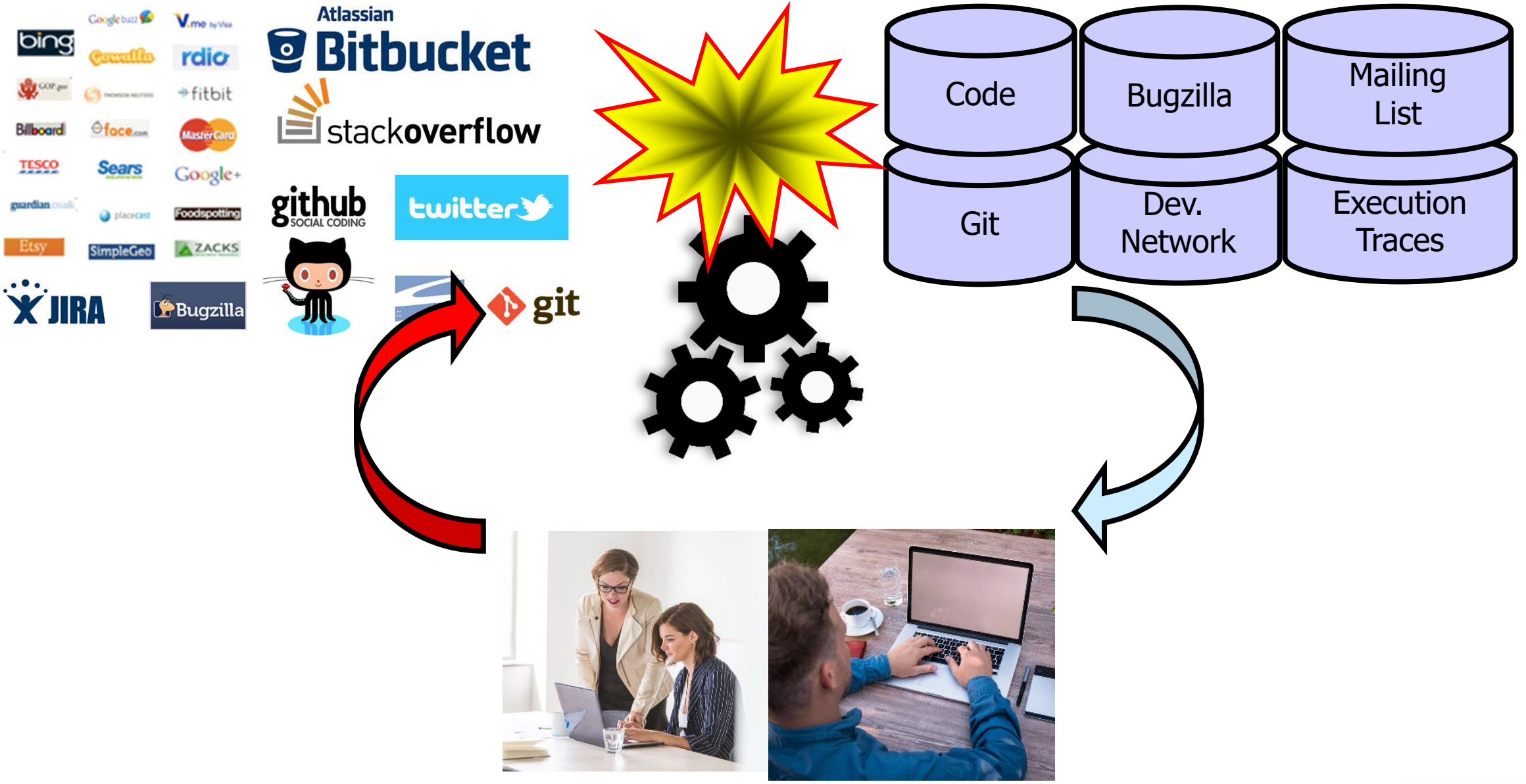
#	Institution	Count	Faculty
1	Nanjing University	39.0	38
2	Carnegie Mellon University	31.6	17
3	Peking University	28.5	21
4	Singapore Management University	22.7	8

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

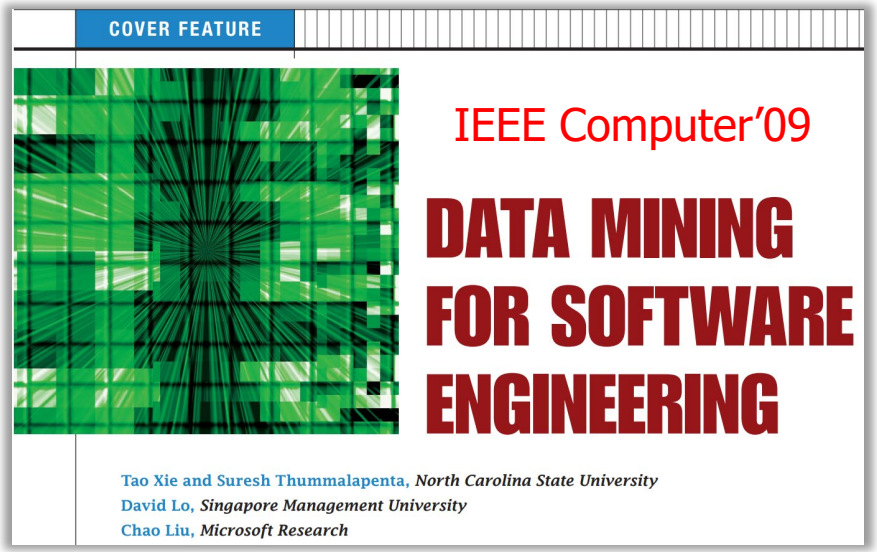
David Lo and Siau-Cheng Khoo
Department of Computer Science,National University of Singapore
{dlo,khoosc}@comp.nus.edu.sg

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



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

Trustworthy and Synergistic AI4SE: Vision and the Road Ahead

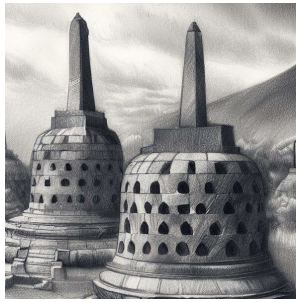
David Lo

SMU Singapore Management University

School of Computing and Information Systems

ICSE'23 Future of SE Talk

AI for Software Engineering

History	Challenges	Vision
		
Roadmap I	Roadmap II	Call4Action

Towards Software Engineering 2.0

School of Computing and Information Systems

SMU Singapore Management University

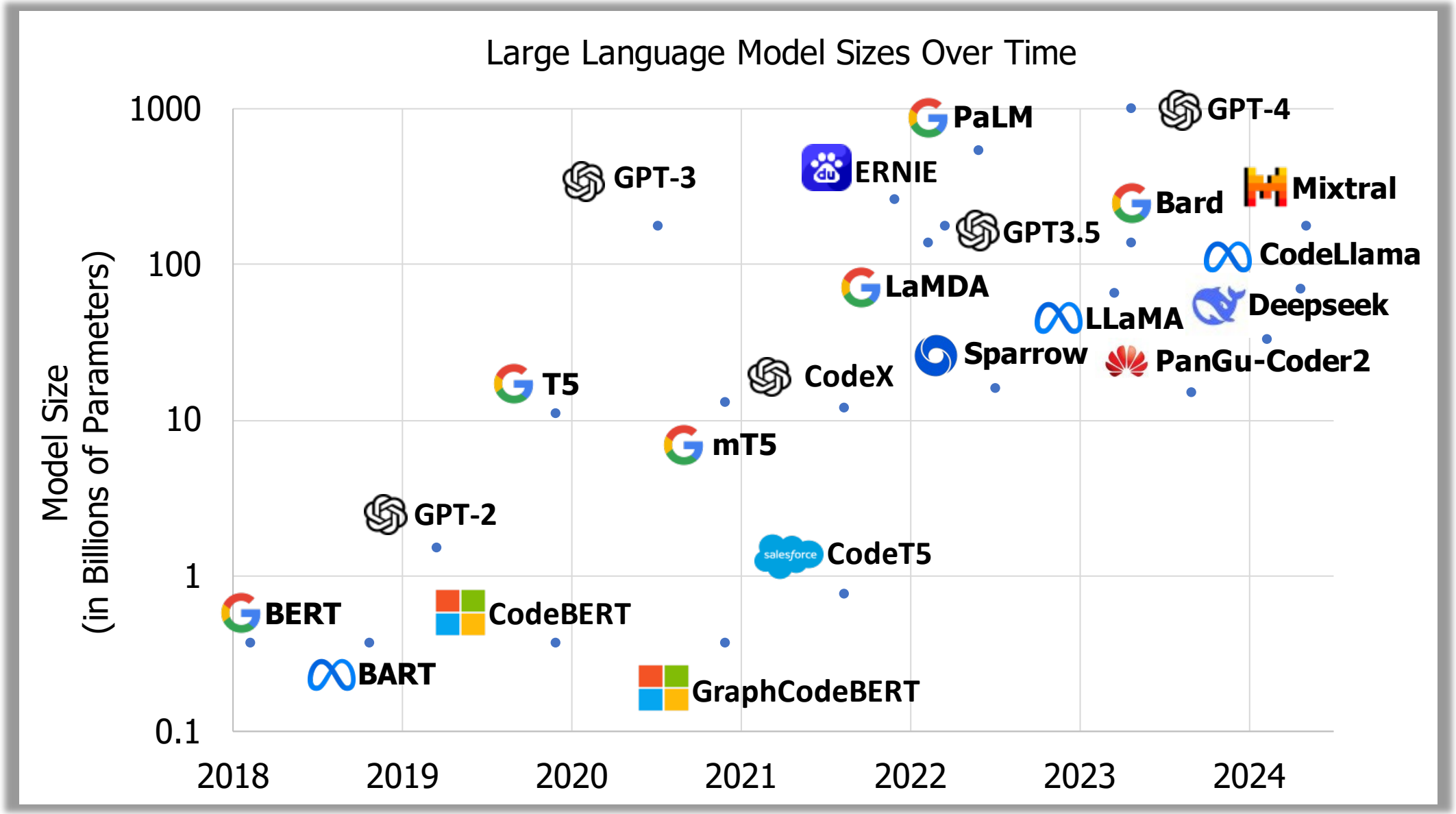
“If you want to go far, go together” – African Proverb

Efficient and Green Code LLMs:
Happier Software Engineers, Happier Planet

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, 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*

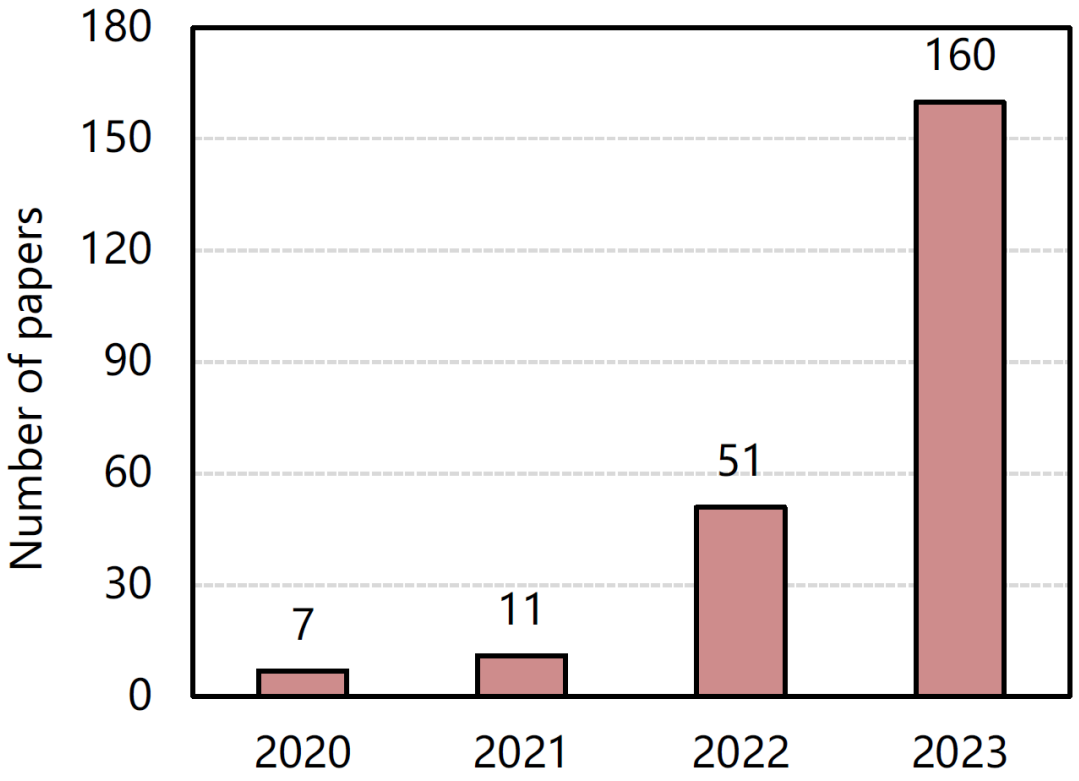
LLMs Seem to Win for Many SE Scenarios

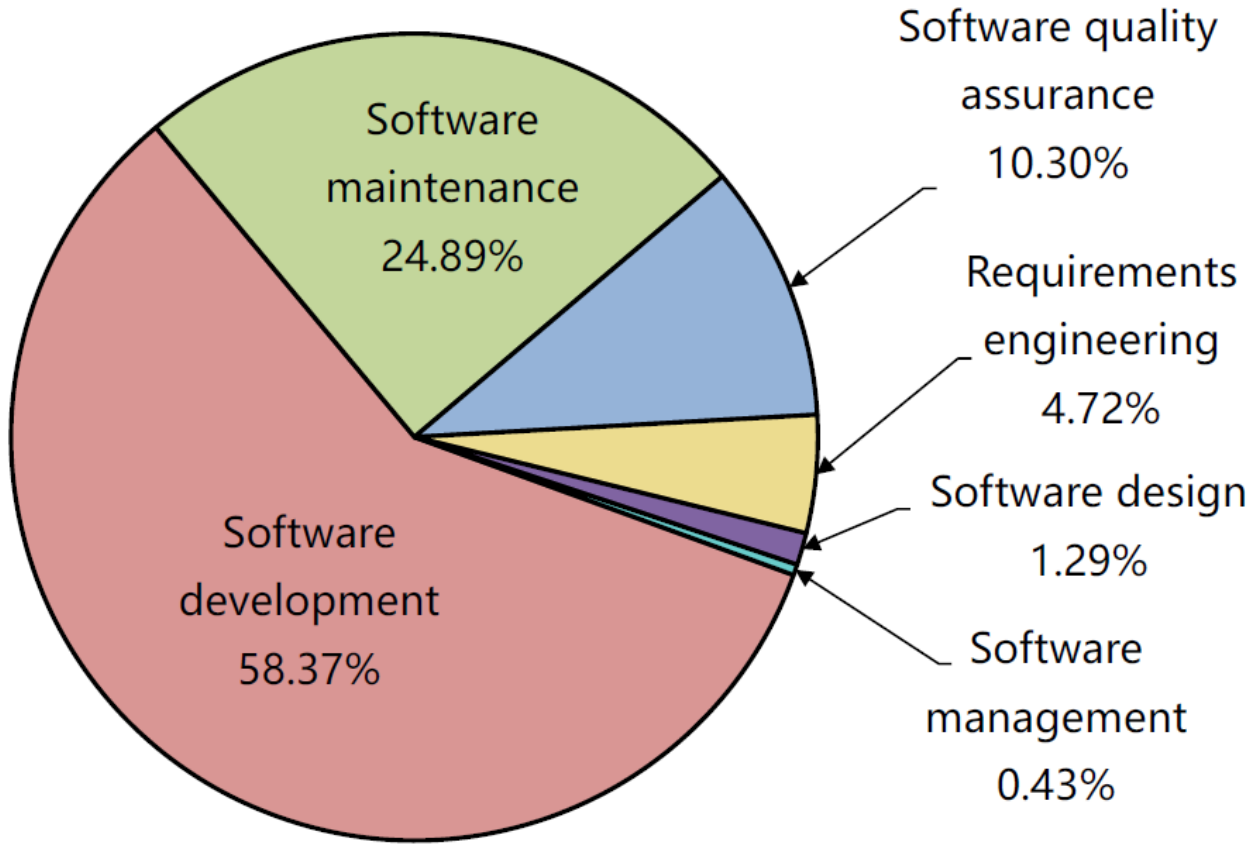
TOSEM 2024

Large Language Models for Software Engineering: A Systematic Literature Review

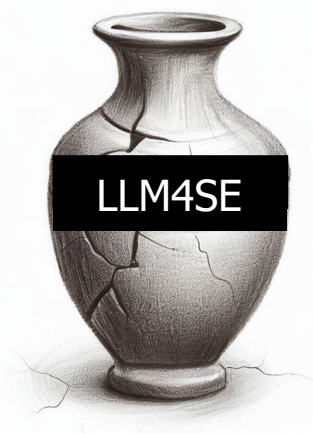
XINYI HOU*, Huazhong University of Science and Technology, China
YANJIE ZHAO*, Monash University, Australia
YUE LIU, Monash University, Australia
ZHOU YANG, Singapore Management University, Singapore
KAILONG WANG, Huazhong University of Science and Technology, China
LI LI, Beihang University, China
XIAPU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

LLMs Seem to Win for Many SE Scenarios





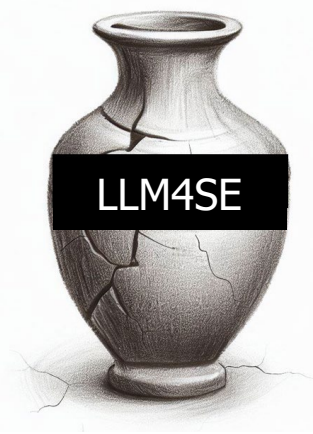
Many Open Problems

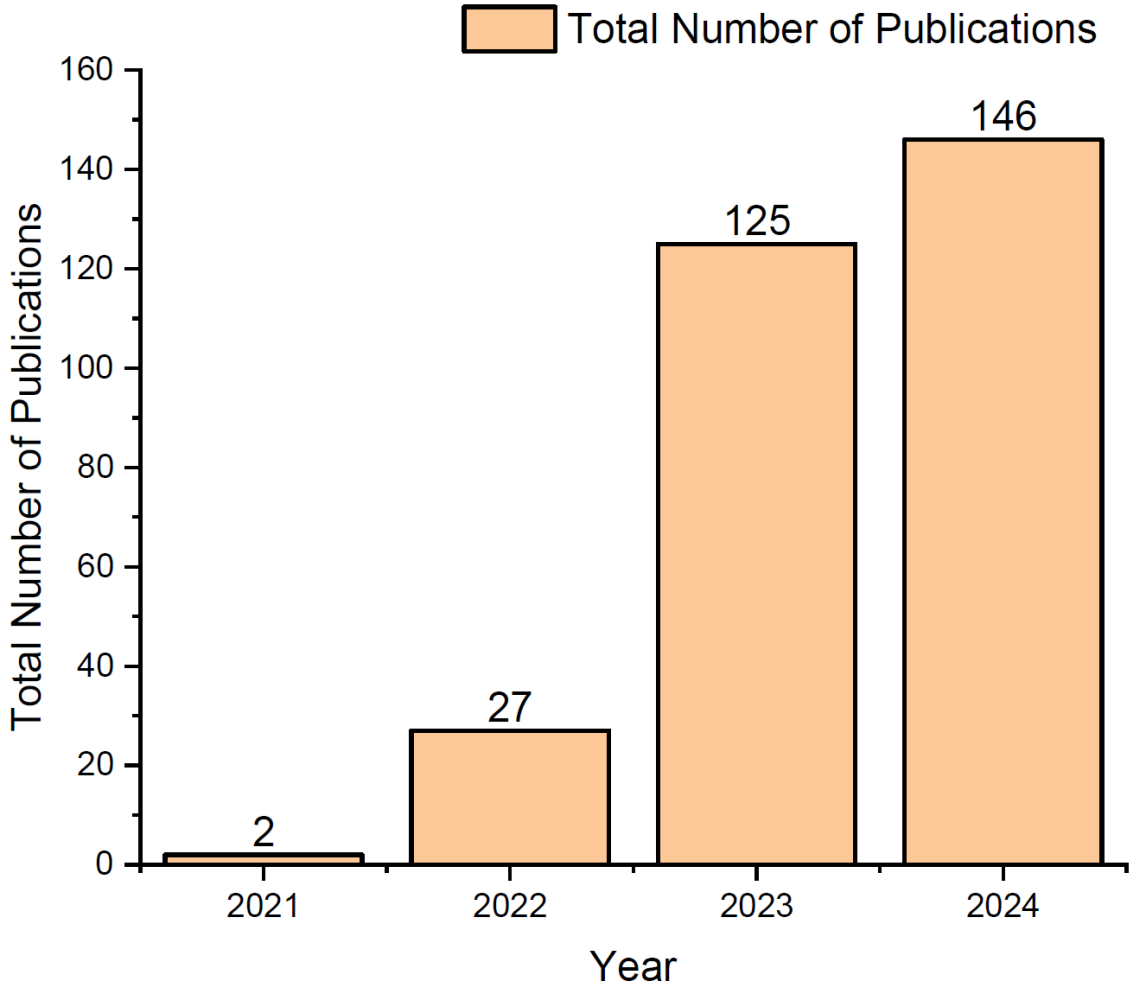


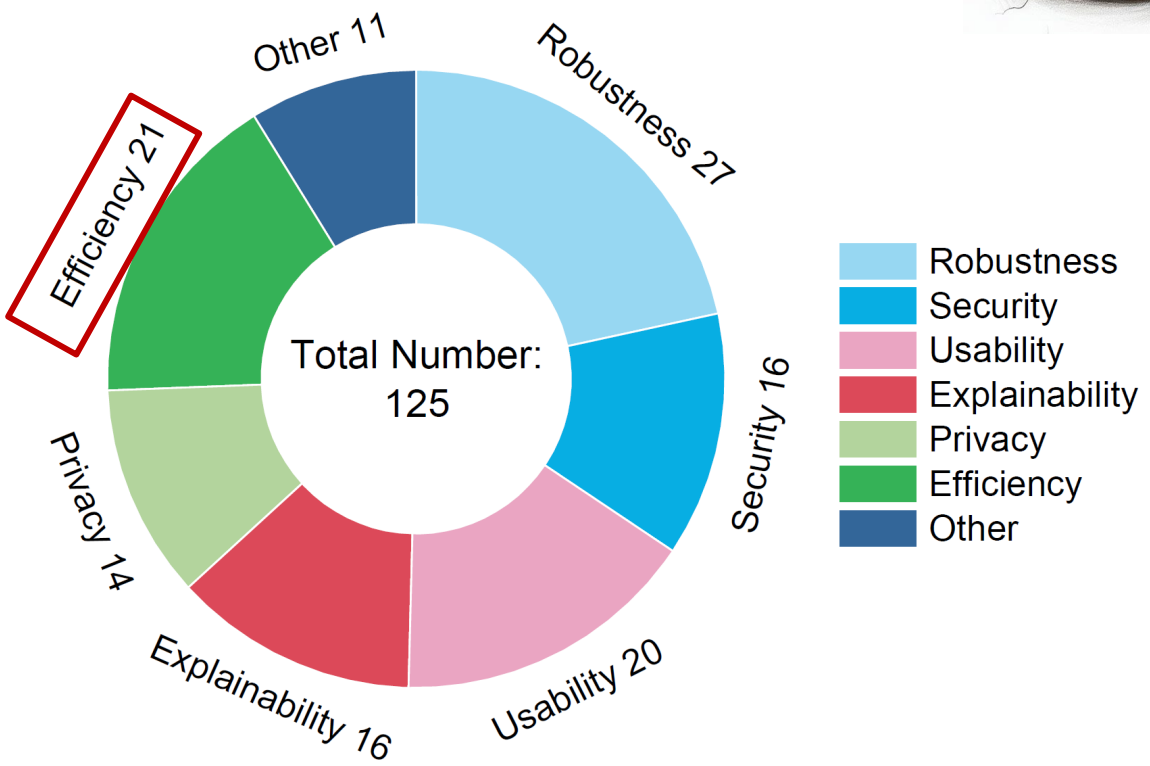
***Robustness*, *Security*, *Privacy*, *Explainability*, *Efficiency*, and *Usability* of Large Language Models for Code**

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







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

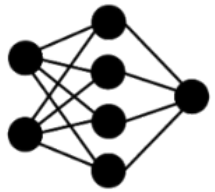
Deploying LLMs to IDE has issues:

Expectations

- "***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*

Reality

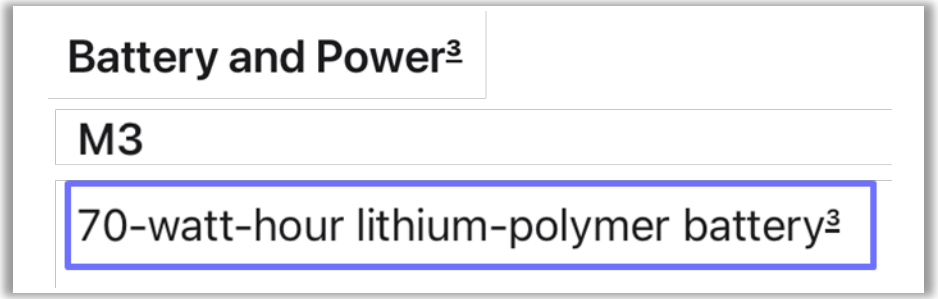


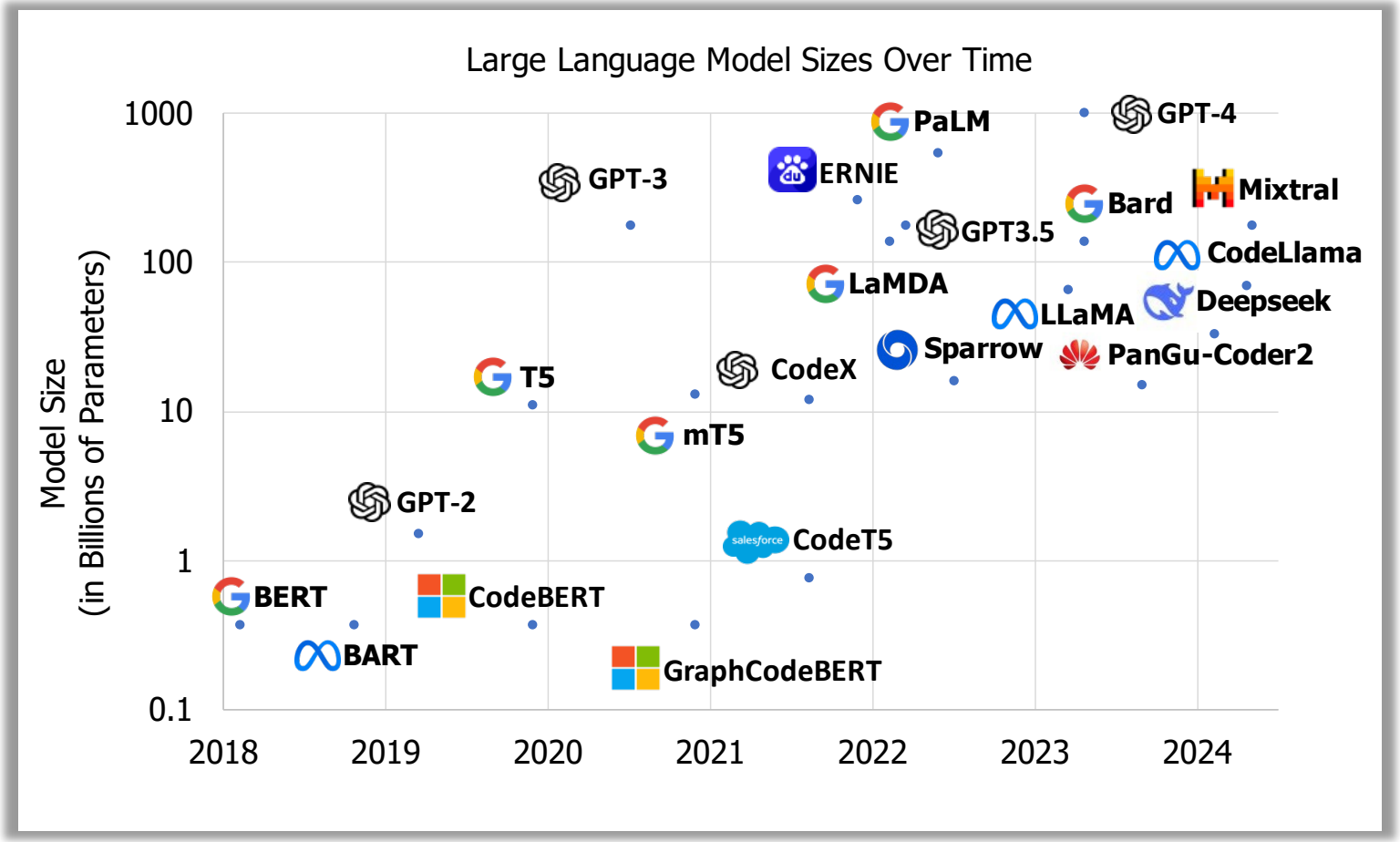
CodeBERT
Size: **> 400MB**
Latency: **> 1.5s/query**

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





Efficient and Green Code LLMs: Three **S**trategies

Stop

Simplify

Shrink

Efficient and Green Code LLMs: Three **S**trategies

Stop

Simplify

Shrink

Stop Unhelpful Code Completion with *FrugalCoder*

TOSEM 2024

Don't Complete It! Preventing Unhelpful Code Completion for Productive and Sustainable Neural Code Completion Systems

ZHENSU SUN, Singapore Management University, Singapore
XIAONING DU*, Monash University, Australia
FU SONG†‡, Key Laboratory of System Software (Chinese Academy of Sciences), State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
SHANGWEN WANG, National University of Defense Technology, China
MINGZE NI, University of Technology Sydney, Australia
LI LI, Beihang University, Beijing, China
DAVID LO, Singapore Management University, Singapore

First work to investigate the problem of **unhelpful code completions**

LLM-based Code Completion Brings New Challenges

LLM-based Code Completion is Popular

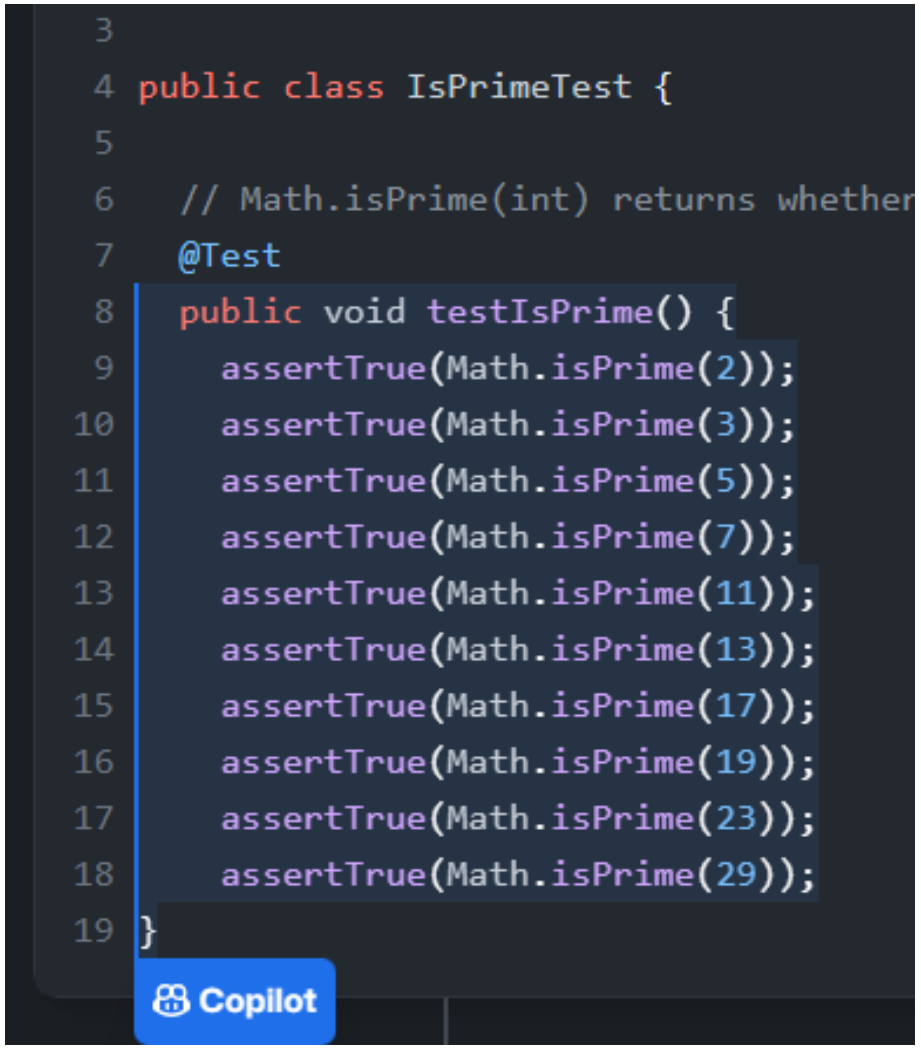
- Each user of Github Copilot receives one suggestion roughly every 3 minutes [GitHub22]

Low Acceptance Rate

- Code completion requests are complex in real world
- Only 30% of completions are accepted by the users of Github Copilot [GitHub22]

High Computation & Latency

- LLM-based code completion requires large scale computing and causes latency

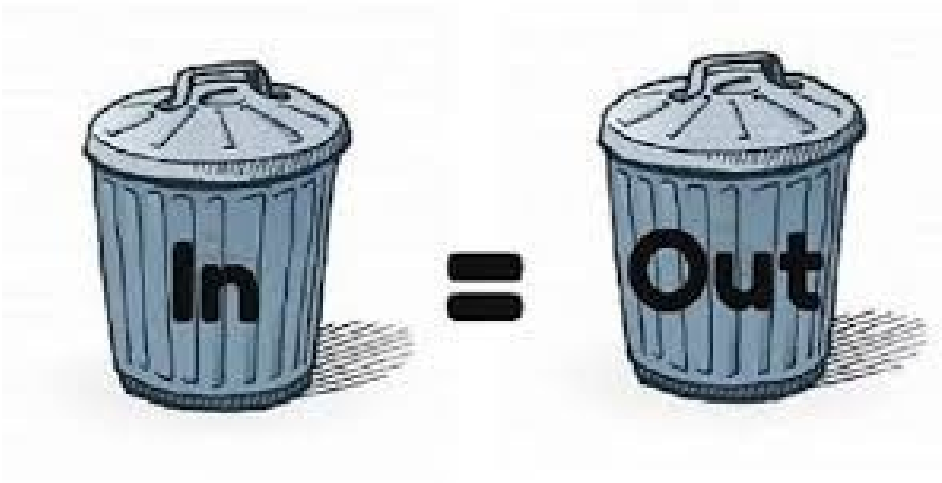


[GitHub22] Albert Ziegler (GitHub), Eirini Kalliamvakou (GitHub), Shawn Simister, et al. Productivity Assessment of Neural Code Completion. MAPS'22 at PLDI'22.

What Cause Unhelpful Code Completions?

(1) Requests that are **beyond the capability of the LLM**

(2) Requests that **do not contain sufficient information**, e.g., meaningless identifiers, vague intention, etc.



How to Address This Problem?

(1) Improve the performance of the LLM-Powered Solutions

Better Model, Better Retrieval-Augmented Solution, …

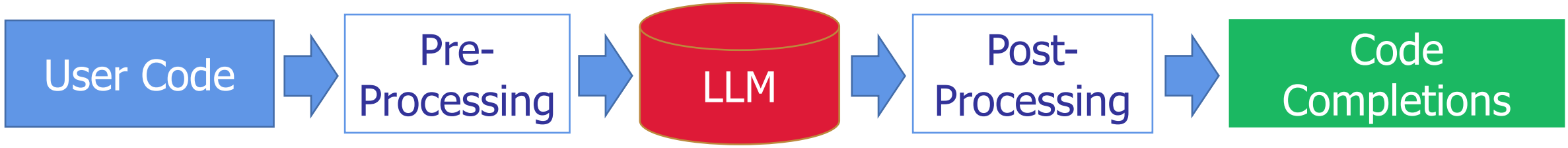
(2) Prevent unhelpful completions

A **new task** for code completion

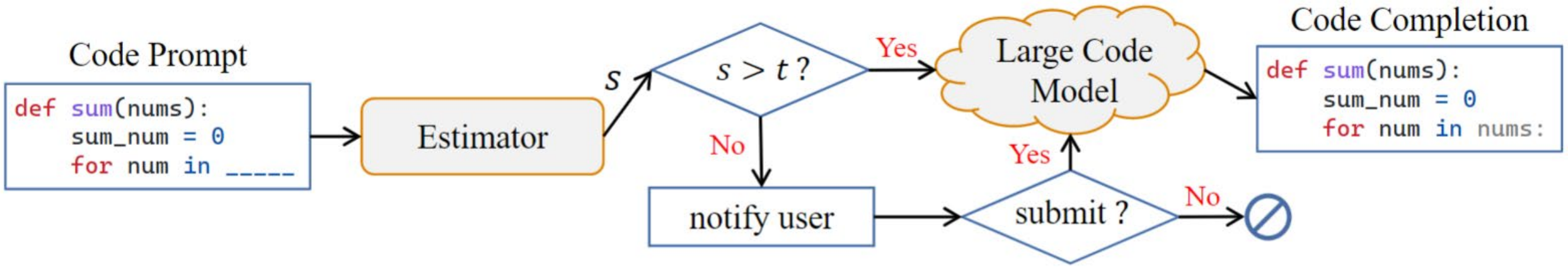
Pre-Processing: Stop the requests ☺️

Inference: Early terminations

Post-processing: Do not display completions



FrugalCoder: Identify & Reject Unpromising Code Prompts



- Using lightweight estimator to estimate the quality of the code completion
- Decide whether to proceed based on a pre-defined threshold

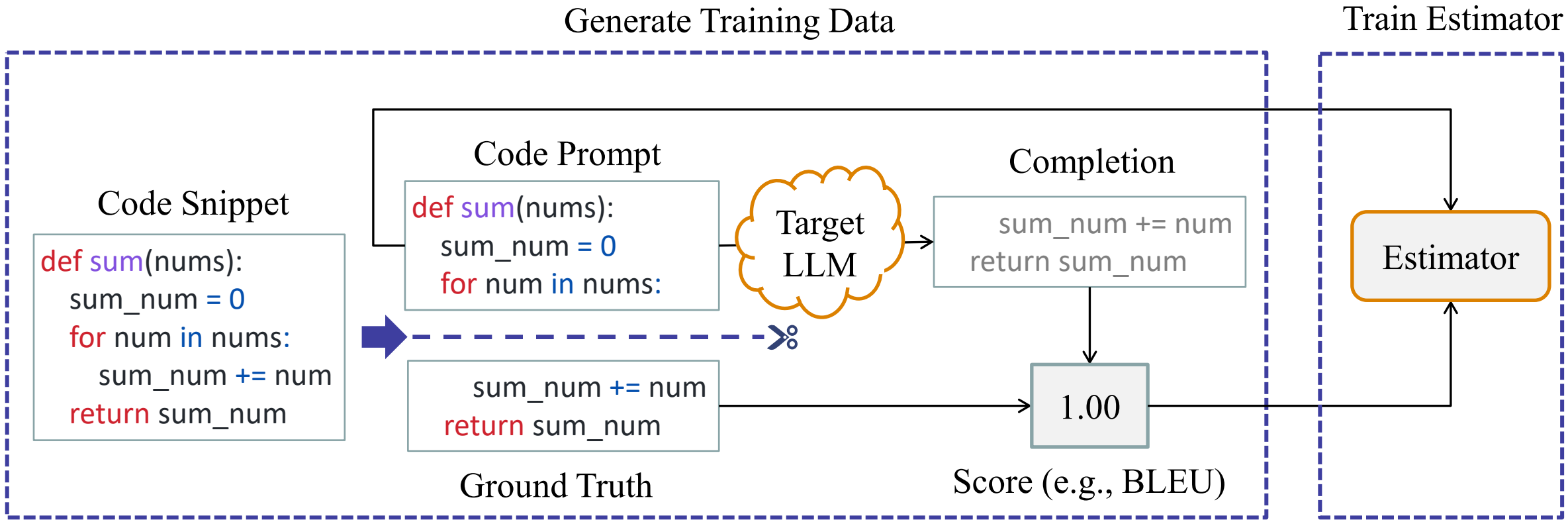
Challenge 1: Efficacy

The estimator should effectively estimate code completion quality

Challenge 2: Efficiency

The cost of running the estimator should be lower than the LLM cost

FrugalCoder: Building the Estimator



Results: Feasibility of FrugalCoder

- **Deep Learning as Estimator**

  Lightweight Transformer

Efficacy
Reject **20%** of requests with a **95.1%** Precision Improve Acceptance Rate from **27.4%** to **33.0%**

Efficiency
5.1 ms for each query

- **Traditional ML as Estimator**

  Adaboost

Efficacy
Reject **20%** of requests with a **92.1%** Precision Improve Acceptance Rate from **27.4%** to **32.3%**

Efficiency
0.1 ms for each query

Future Work

- **Stop remaining unhelpful code completions**

✓ Stop more unhelpful code completions to increase acceptance rate even more

- **Explore the secondary use of FrugalCoder**

✓ Such as its potential to identify and stop adversarial attacks, backdoor attack, etc.

Efficient and Green Code LLMs: Three **S**trategies

Stop

Simplify

Shrink

Simplify Programming Language Grammars

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

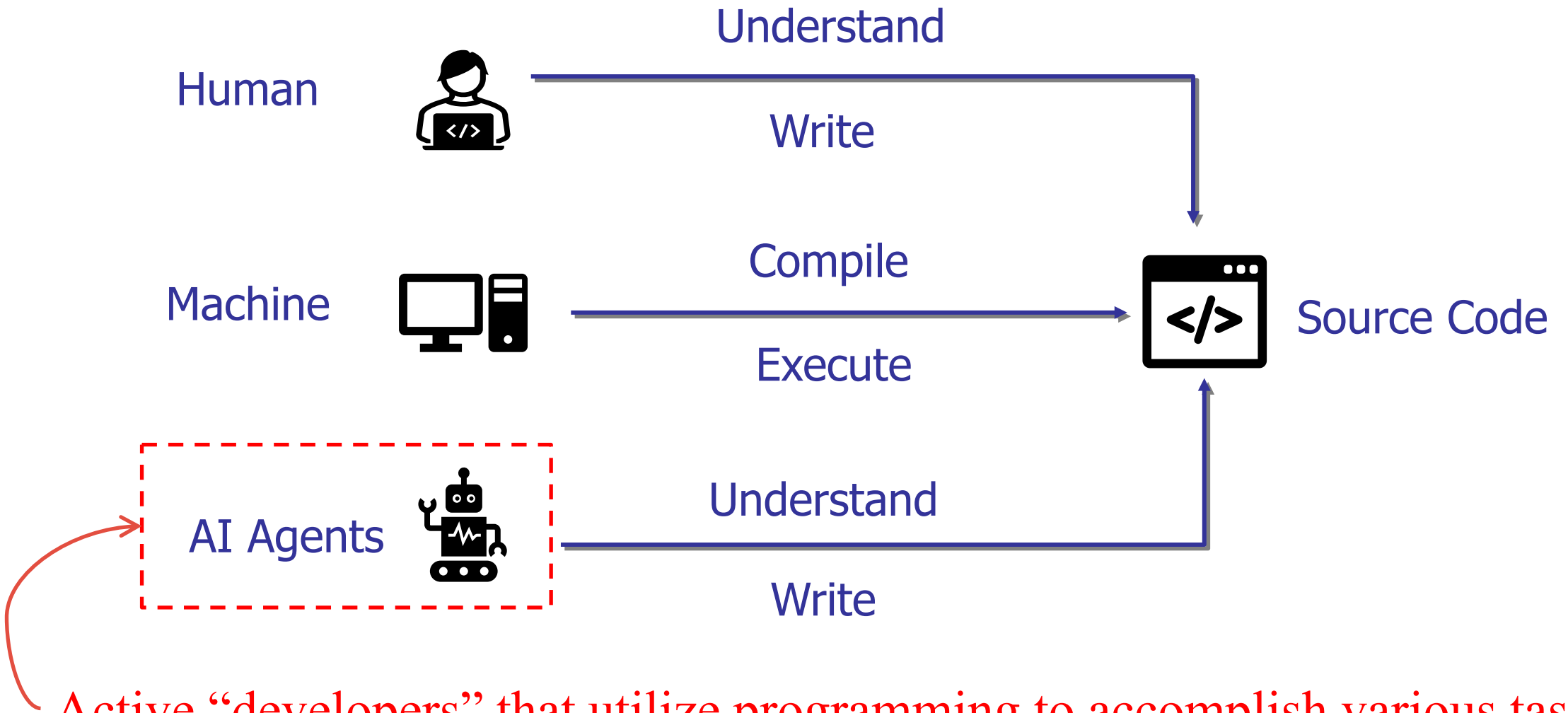
Li Li
Beihang University
China
lilicoding@ieee.org

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**

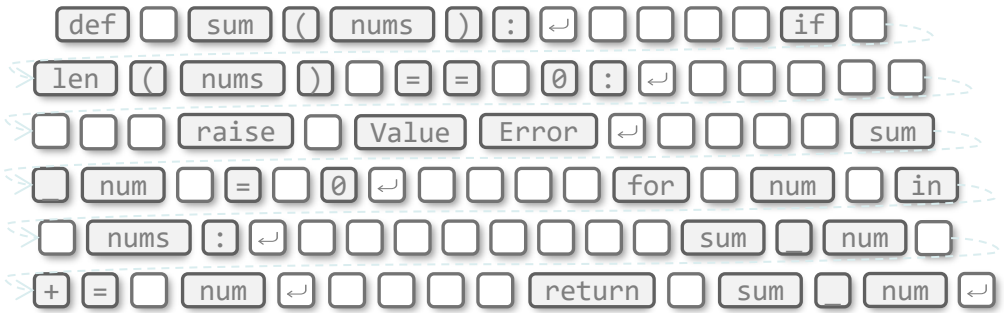
AI Agents: The Third Audience

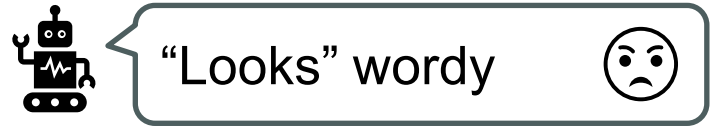


Active “developers” that utilize programming to accomplish various tasks

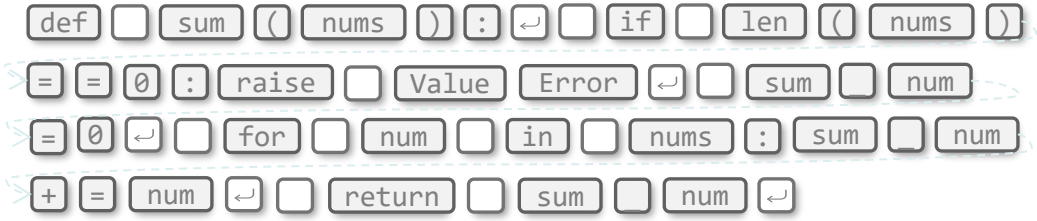
Human-Centric Programming Language: Readability Counts

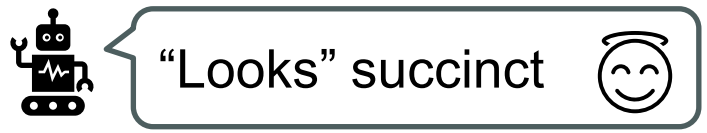
```
def sum(nums):
    if len(nums) == 0:
        raise ValueError
    sum_num = 0
    for num in nums:
        sum_num += num
    return sum_num
```






```
def sum(nums):
 if len(nums)==0:raise ValueError
 sum_num=0
 for num in nums:sum_num+=num
 return sum_num
```





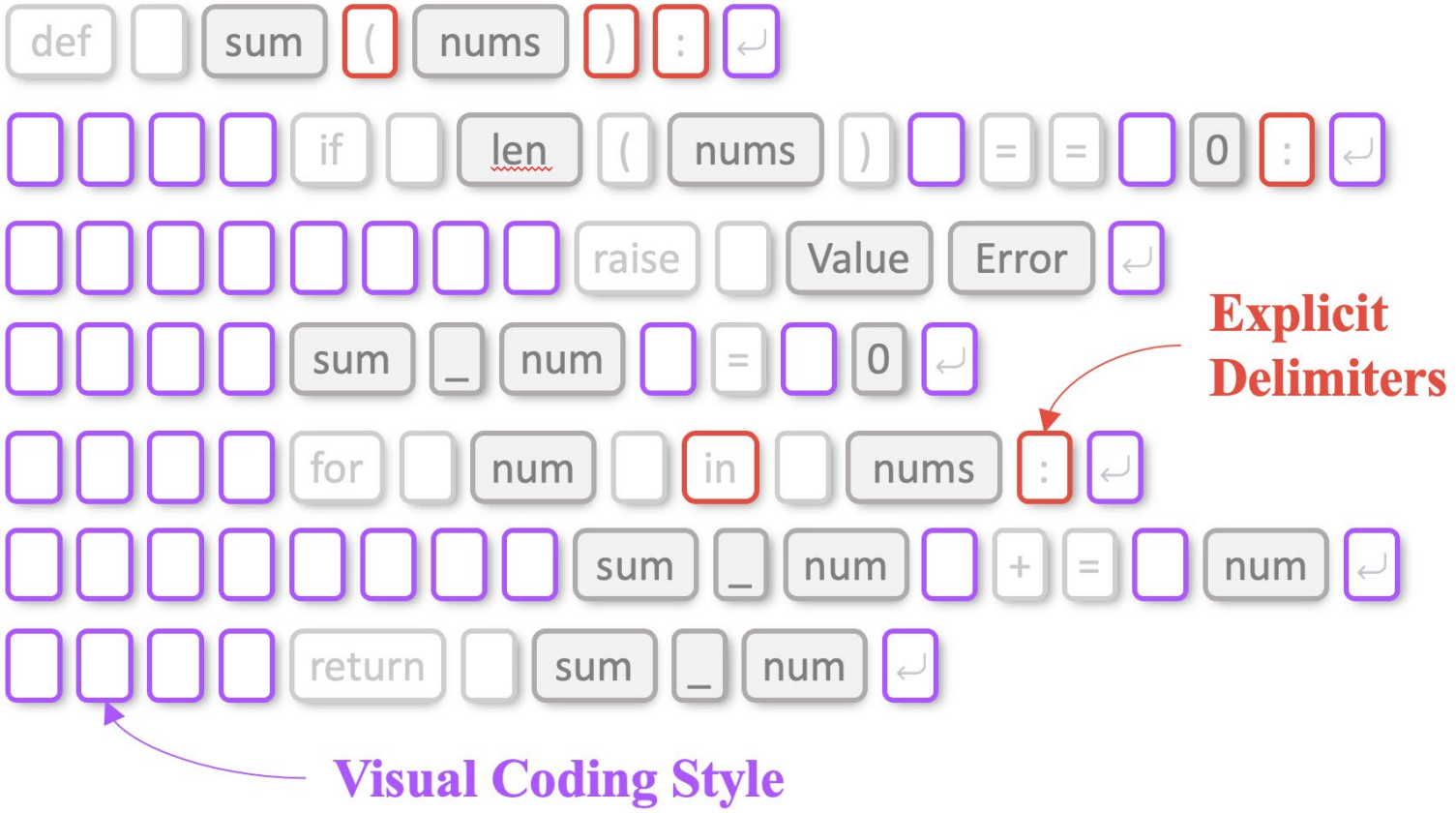
Human-Centric Programming Language Grammar Design

Visual Coding

Include symbols like line breaks and indentions

Explicit Delimiters

Use explicit delimiters to define code structures despite some delimiters not being essential for parsing



Are there better programming language grammars for AI agents?

SimPy: A proof-of-concept AI-agent oriented grammar

Python

Tokenized by CodeBERT

128 tokens

```
def two_sum(nums: list[int], target: int) -> list[int]:\n
    chk_map: dict[int, int] = {}\n
    for index, val in enumerate(nums):\n
        compl = target - val\n
        if compl in chk_map:\n
            return [chk_map[compl], index]\n
        chk_map[val] = index\n
    return []
```

Same Execution Results

SimPy

80 tokens

```
<def_stmt>two_sum nums:list[int] target:int<arrow>list
[int]<block_start>chk_map:dict[int int]={}<for_stmt>
index,val enumerate(nums)<block_start>compl=target-val
<if_stmt>compl<in>chk_map<block_start><return>[chk_map
[compl] index]<block_end>chk_map[val]=index<block_end>
<return>[]<block_end>
```

Replace notations with tokens

Replace keywords and symbols (e.g., “if”, “for”, etc.) with specialized tokens

Restrict coding style

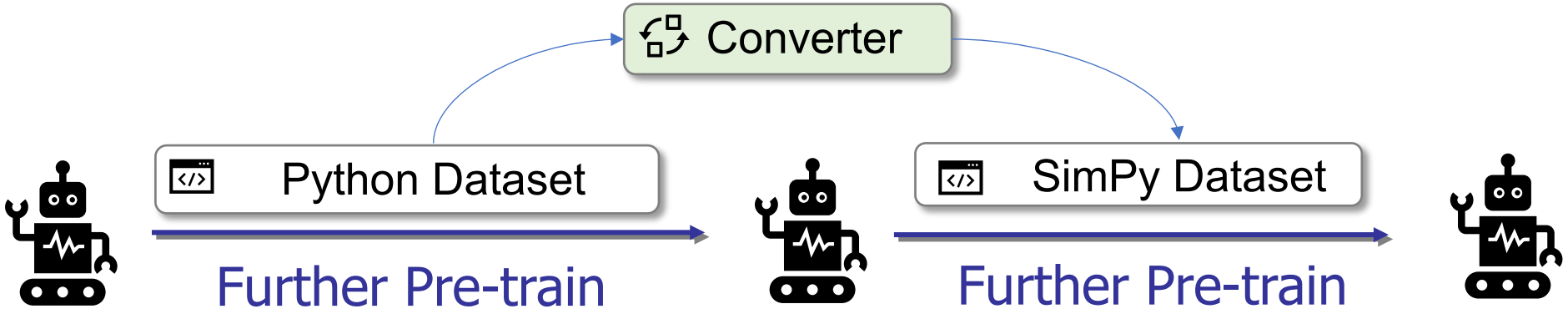
Streamline white spaces, line breaks, indents, etc. preserving only essential separators

Simplify grammar tokens

For every grammar token in every production, we review whether it can be removed, merged with others, or replaced with white spaces

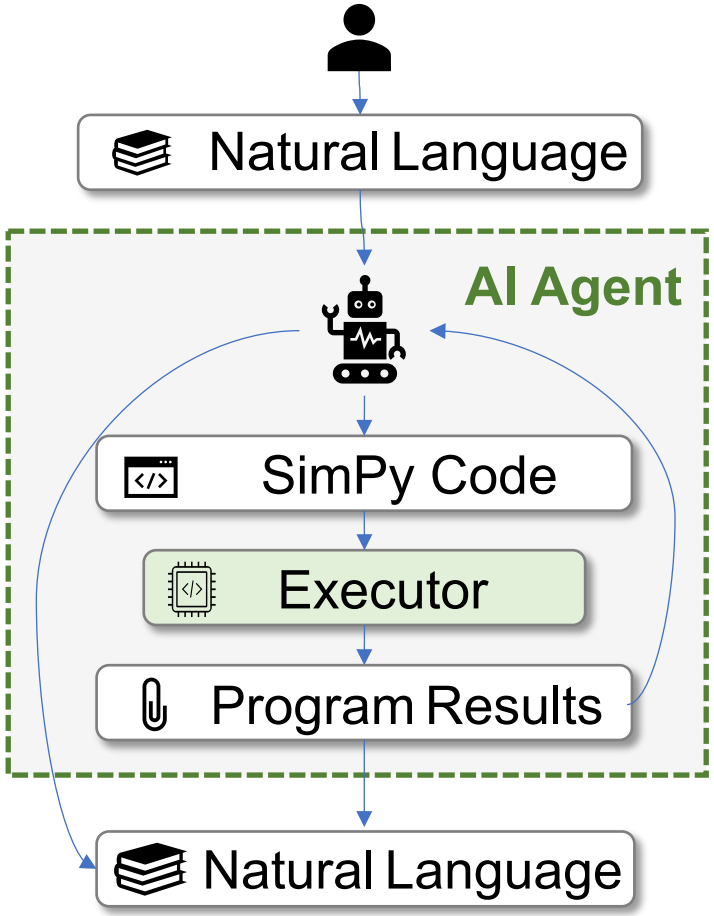
Making AI Agent Understand and Write SimPy

LLM first trained on Python and then on SimPy (with equivalent sample size)

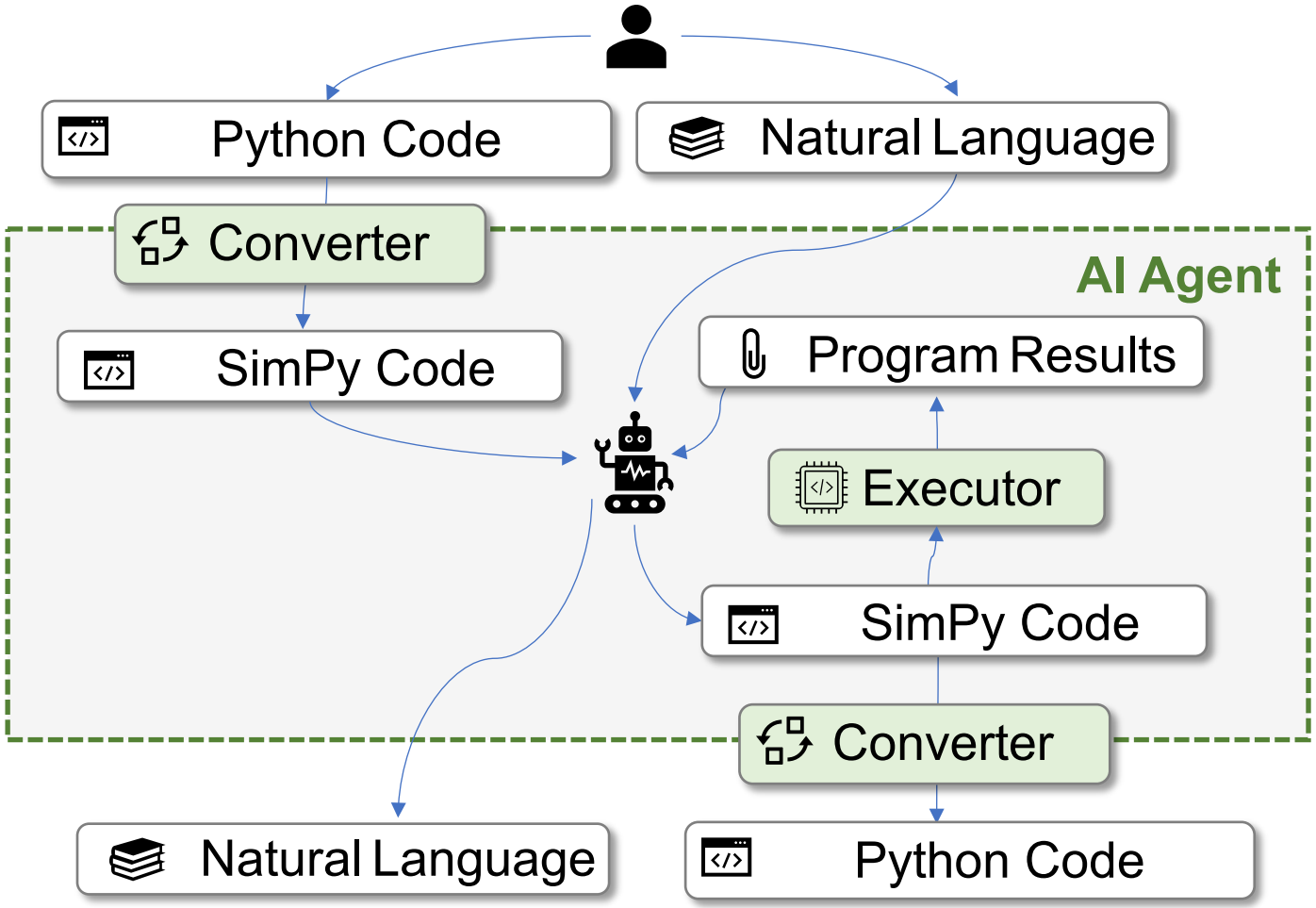


✓ SimPy is more similar to Python than natural language

SimPy's Possible Usage Scenarios



Basic usage scenario



Extended usage scenario: **DualCode**

RQ1: Token Reduction

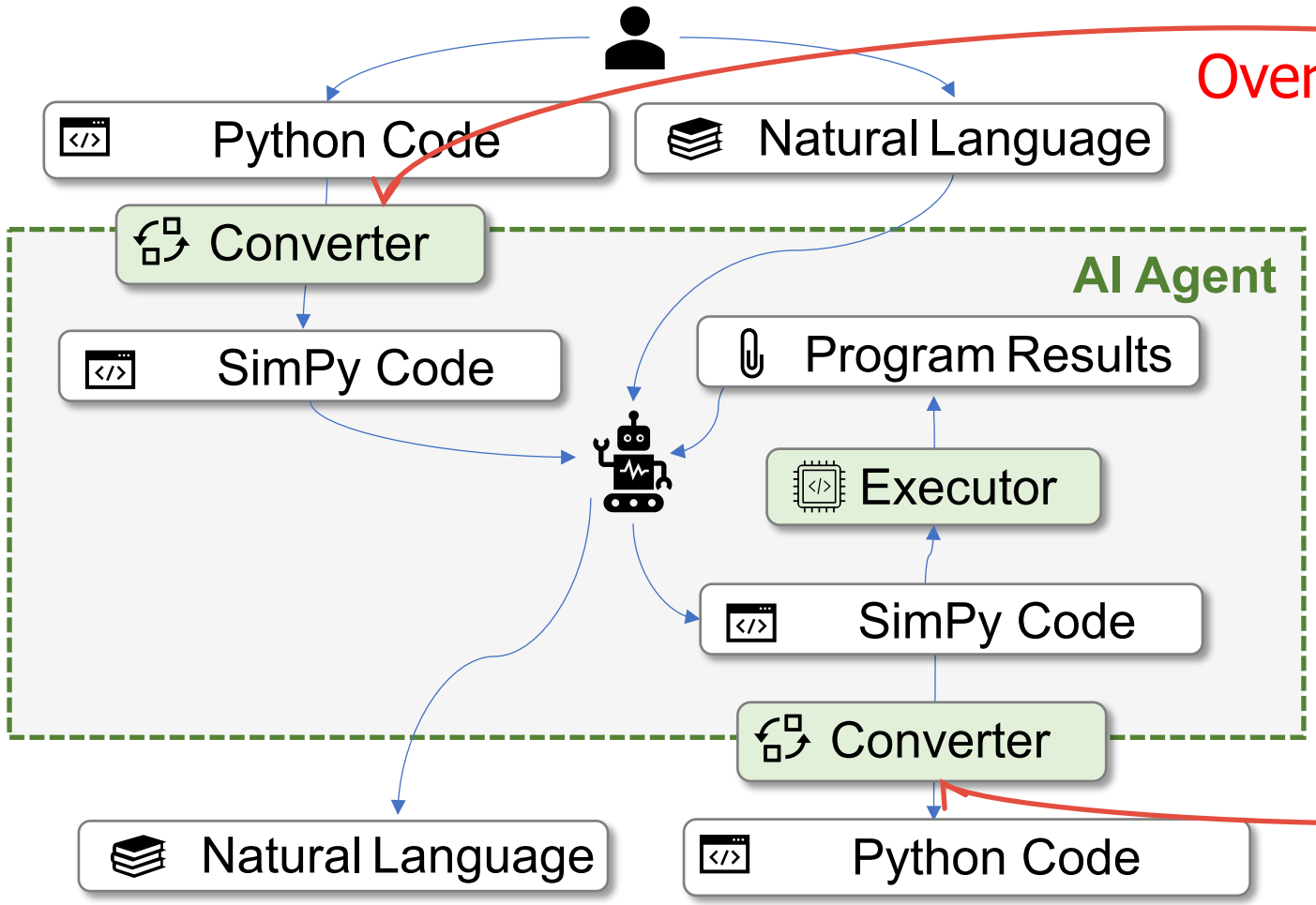
- SimPy reduces the number of tokens required for source code representation.
- CodeBERT and GPT-4 benefiting from a **34.7%** and **10.4%** reduction.

Tokenizer	Vocab Source	Vocab Size	Tokens		
			Python	SimPy	
CodeBert	Code	50k	1.33B	0.87B	34.7%↓
CodeLlama	Web	32k	0.97B	0.84B	13.5%↓
Codex	Web	51k	0.93B	0.82B	12.6%↓
CodeT5	Code	32k	0.91B	0.78B	13.8%↓
StarCoder	Code	49k	0.83B	0.76B	8.6%↓
GPT-3.5	Web	100k	0.71B	0.63B	10.4%↓
GPT-4	Web	100k	0.71B	0.63B	10.4%↓

RQ2: Efficacy of AI Agents Trained on SimPy

Model	Training Strategy	Pass@10
CodeGen-NL (350M)	Python	7.32%
	Python->SimPy	**9.15%**
TinyLlama (1.1B)	Python	13.41%
	Python->SimPy	**14.02%**
Pythia (1.0B)	Python	9.76%
	Python->SimPy	**10.00%**

RQ3: Overhead of DualCode Scenario



Overhead of the converter

0.9ms CPU time for processing source code within 500 tokens.

As a reference, Huggingface tokenizer costs 0.7ms for the same code

Extended usage scenario: DualCode

Future Work

- **Automated grammar compression**

✓ Automated approaches to create optimal grammar rules for a prog. language

✓ Go beyond Phyton

✓ Achieve more than 8-35% savings

- **AI-agent-first grammar**

✓ Designing from scratch grammars that are created for AI agents

- **Better adaptation methods**

✓ Better ways to adapt LLMs to understand and use AI-agent-oriented grammars, such as through contrastive learning

Efficient and Green Code LLMs: Three **S**trategies

Stop

Simplify

Shrink

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 compress code LLMs: **160× smaller** and **4.23× faster**

Nominated for ACM SIGSOFT Distinguished Paper Award

Today's Sharing

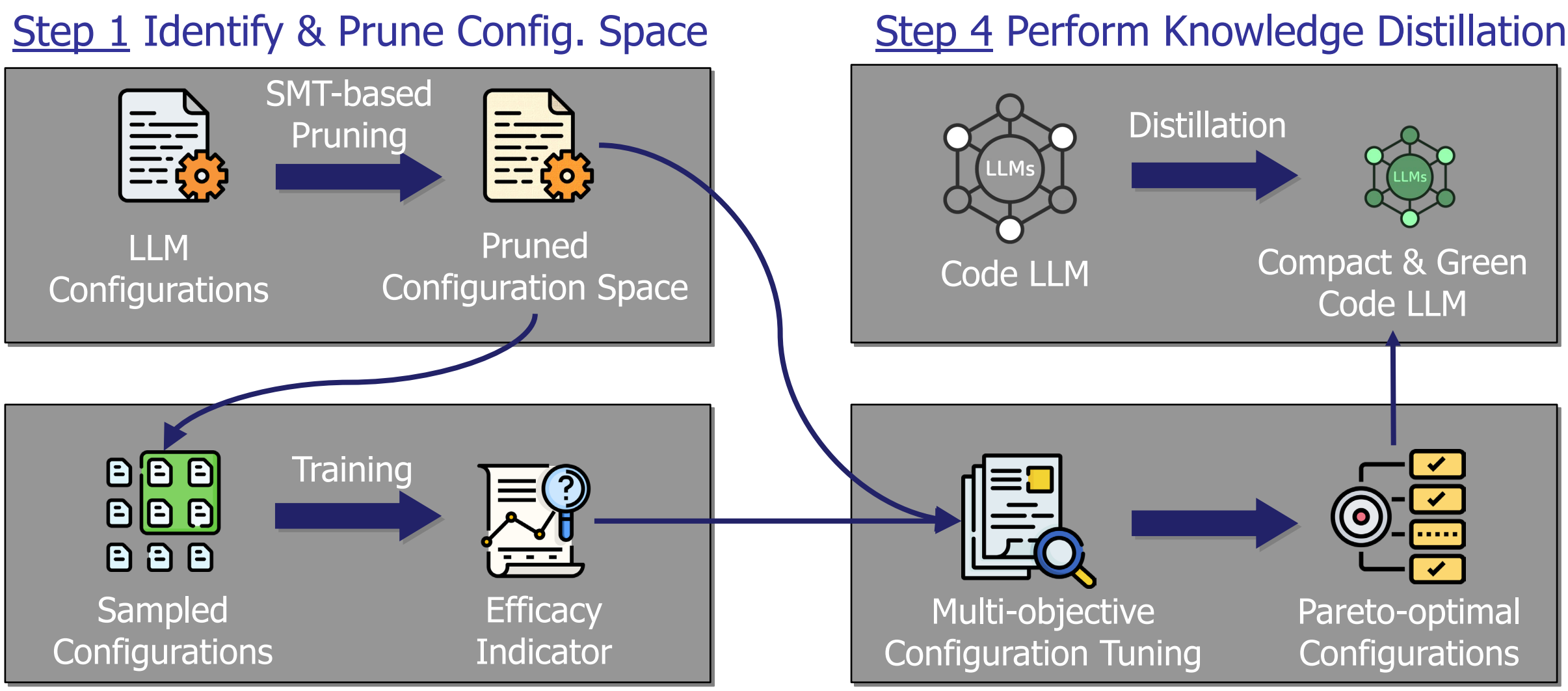
ICSE 2024
Avatar

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

Compress 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 & many are infeasible!

Step 1: Prune Massive Configuration Space

formulating model size
and its constraint:

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

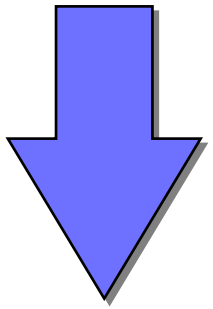
$$\texttt{size}(c) = \frac{4(v+s+3)h}{1024 \times 1024} + \frac{4(4h^2+(9+2i)h+i)l}{1024 \times 1024} + \frac{2h^2+4h+2}{1024 \times 1024}$$

- 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

Using SMT solver
to prune



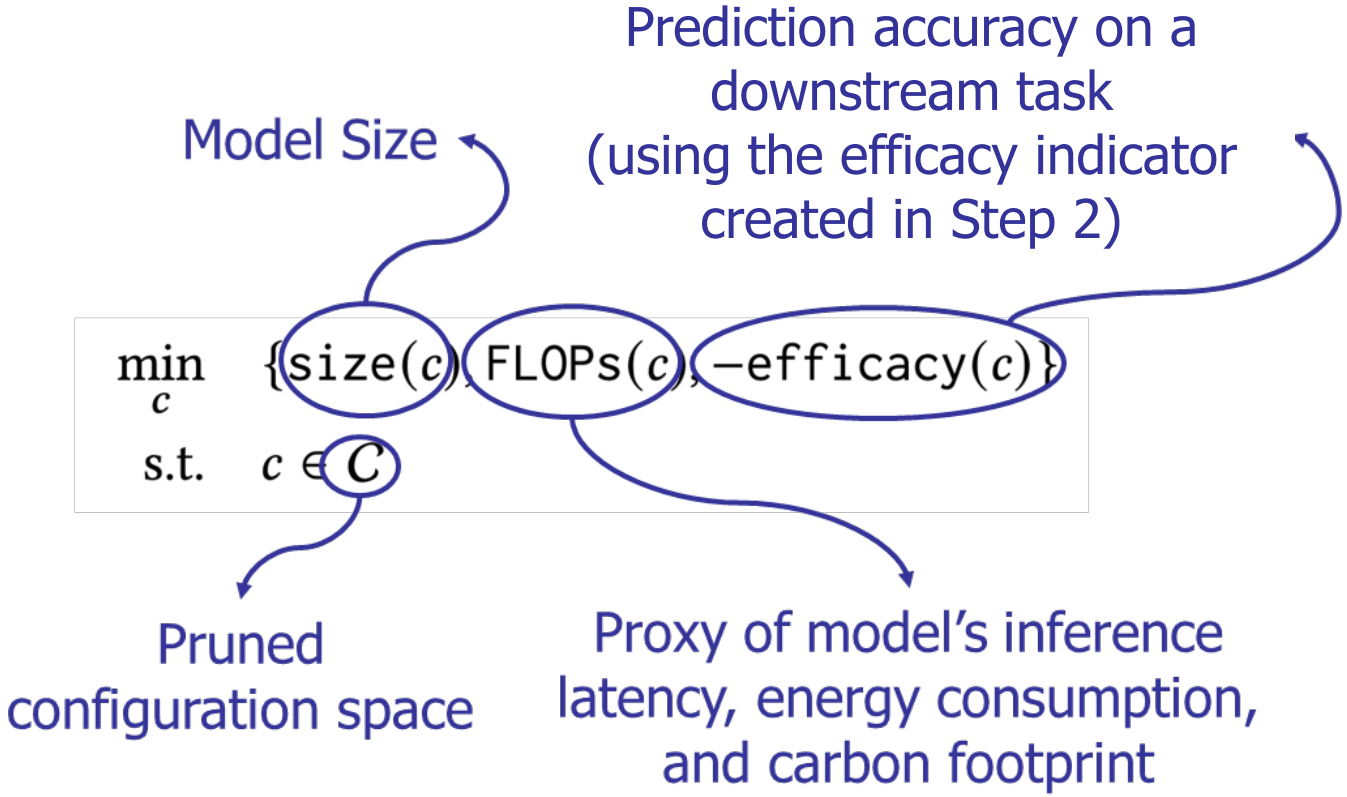
$$\texttt{size}(c) \leq 3\ \text{MB} \quad \text{s.t.} \quad c \in C$$

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

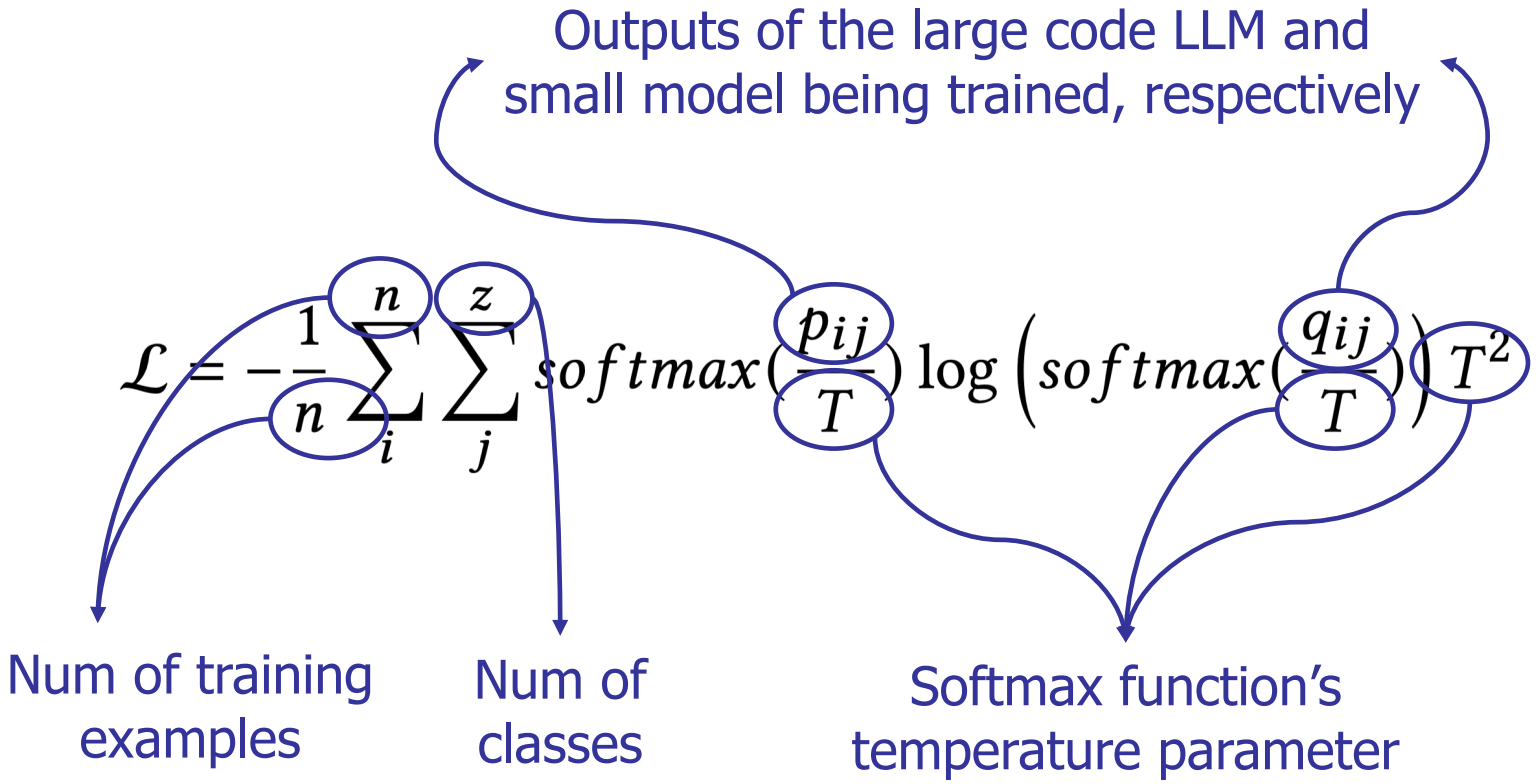
Remaining space after pruning accounts for
only 28.9% of the original one

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**



Step 4: Perform Knowledge Distillation



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
Only 1.67% loss

Future Work

- **More experimentation**
  - ✓ Compressing more and larger models
  - ✓ Consideration of various SE tasks
- **LLM training/fine-tuning acceleration**
  - ✓ Effective data reduction and selection
- **Better LLM inference acceleration**
  - ✓ Combine multiple strategies in addition to compression
  - ✓ Dynamic model inference, static program optimization, etc.

Efficient and Green Code LLMs: Three **S**trategies

Stop

Simplify

Shrink

Related Work on Efficient and Green Code LLMs

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**

Related Work on Efficient and Green Code LLMs

Simplify

FSE 2022

Diet Code Is Healthy: Simplifying Programs for Pre-trained Models of Code

Zhaowei Zhang[1], Hongyu Zhang[2], Beijun Shen[1], Xiaodong Gu[1*]
[1]School of Software, Shanghai Jiao Tong University, China
[2]The University of Newcastle, Australia

40% less computational cost in fine-tuning and testing by **simplifying input programs**

Related Work on Efficient and Green Code LLMs

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

Literature Review, Vision, and the Road Ahead

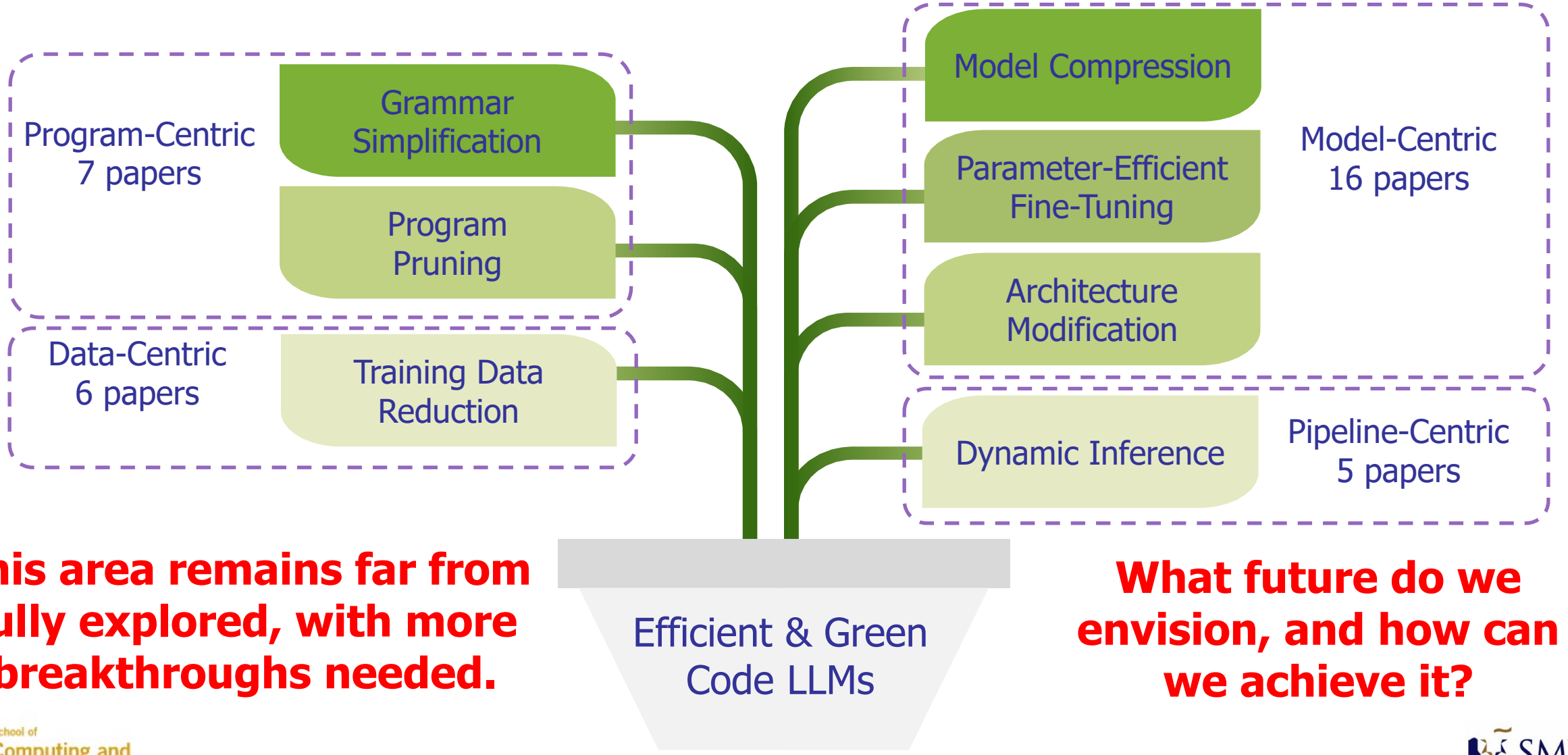
Efficient and Green Large Language Models for Software Engineering: Vision and the Road Ahead

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

TOSEM SE Vision 2030 @ FSE 2024
TOSEM "2030 SE Roadmap" Special Issue

First work to (1) **survey** the current state of efficient and green code LLMs,
(2) **envision** future possibilities,
and (3) present a **roadmap** to achieve that future.

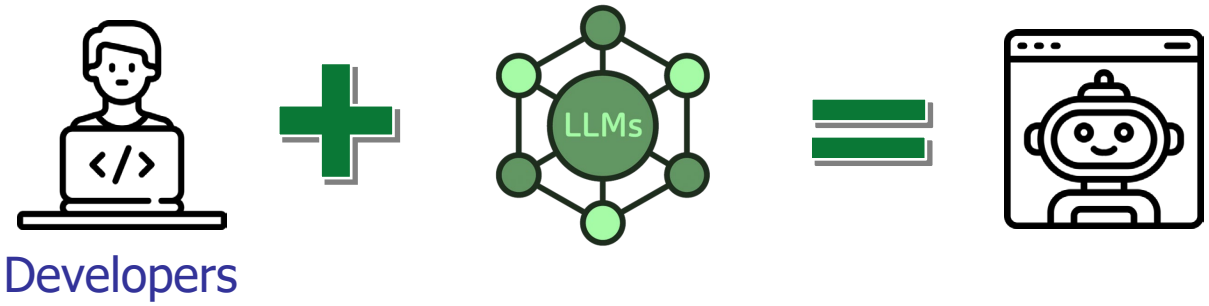
Literature Review



This area remains far from fully explored, with more breakthroughs needed.

What future do we envision, and how can we achieve it?

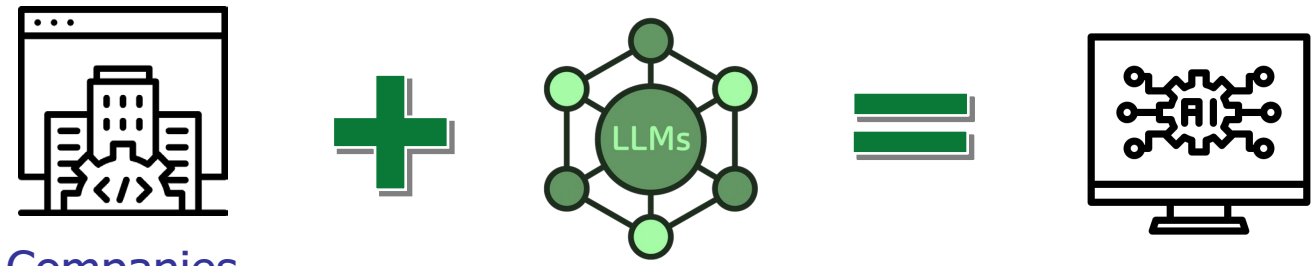
Vision - I



Private, Personalized, Low-Cost Software Engineering Assistants

Imagine a 3MB-only **private** LLM on your laptop, **fully aware** of your work, offering **instant help** within 100ms even without an Internet connection!

Vision - II



Companies

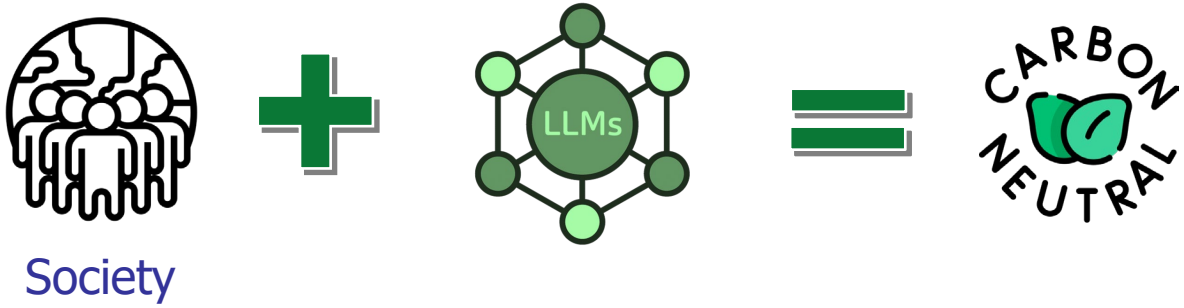
Profitable LLM-Powered Cloud-Based Software Engineering Services

GitHub Copilot now incurs a $20 loss per user per month. Imagine if costs drop and it becomes **profitable** for GitHub and many other smaller companies to offer **LLM-powered software engineering services**!

Vision - III

Vision 2033:

Symbiotic workforce of *autonomous*, *responsible*, *intelligent* agents, software engineers, and end users, radically transforming how software is engineered (**Software Engineering 2.0**)



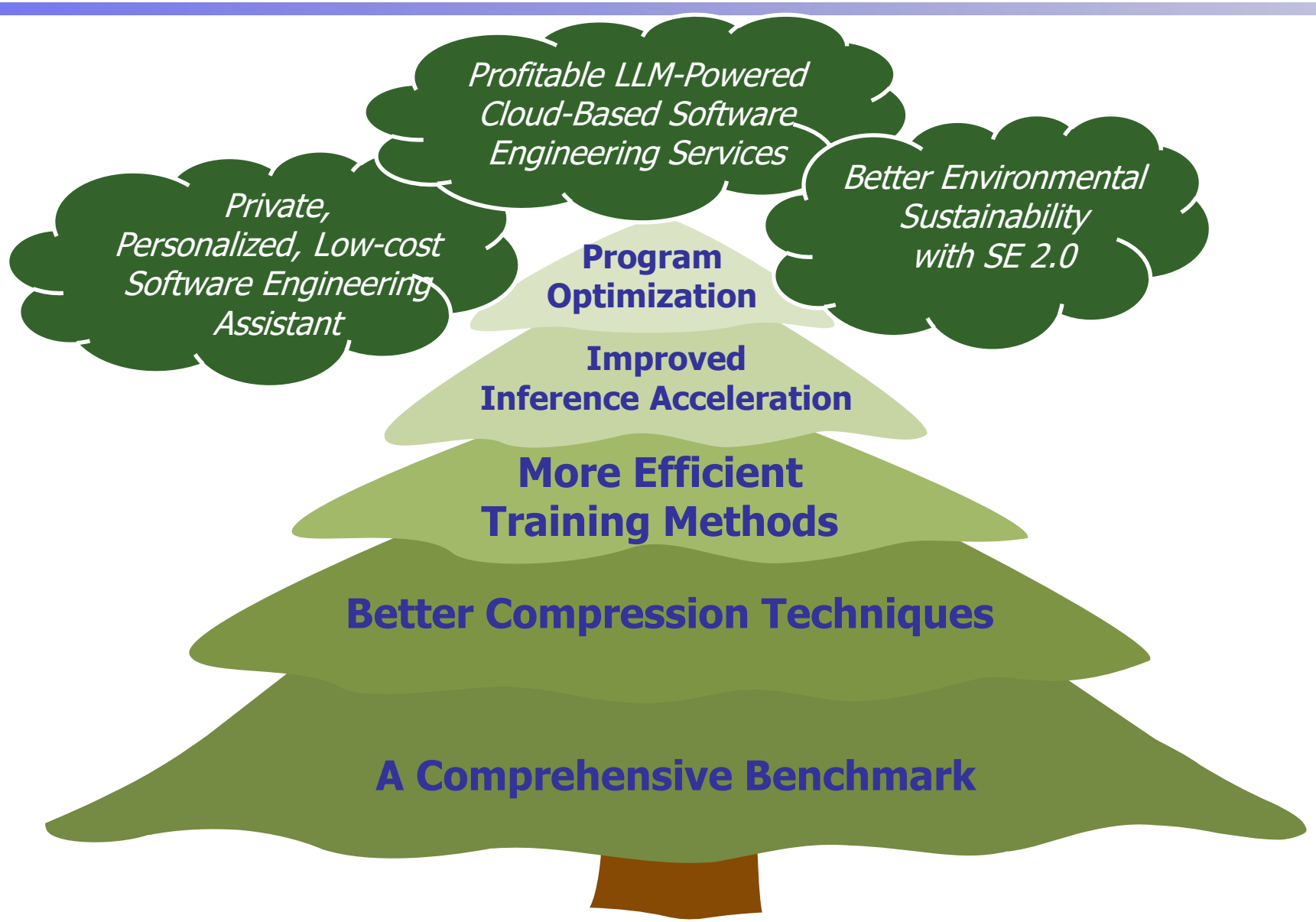
Society

Better Environmental Sustainability with Software Engineering 2.0

Training LLaMA consumes electricity equal to 445 Danish citizens in a year. Imagine how sustainable if the **training and inference cost can be reduced to 10%**!

Imagine if SE 2.0 can "pay" for the remaining cost by **rectifying inefficiencies** in software systems and **enabling adaptive software** that runs only what is needed, achieving **net carbon neutrality**!

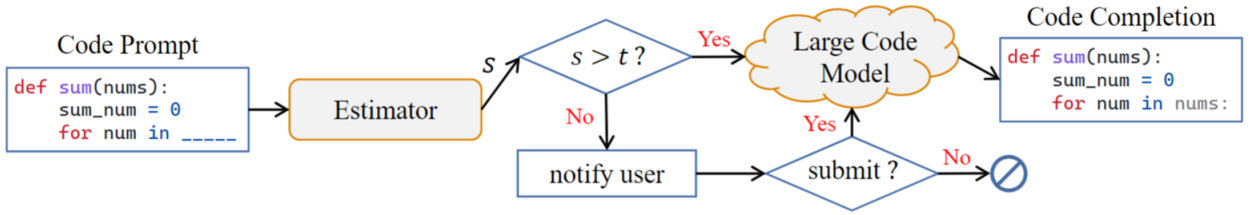
Vision and Road Ahead



Efficient and Green Code LLMs: Three Strategies

Stop **Simplify** **Shrink**

FrugalCoder: Identify & Reject Unpromising Code Prompts



- Using lightweight estimator to estimate the quality of the code completion
- Decide whether to proceed based on a pre-defined threshold

Challenge 1: Efficacy
The estimator should effectively estimate code completion quality

Challenge 2: Efficiency
The cost of running the estimator should be lower than the LLM cost

School of Computing and Information Systems

SMU Singapore Management University

SimPy: A proof-of-concept AI-agent oriented grammar

Python — Tokenized by CodeBERT — **128** tokens

```
def two_sum(nums: list[int], target: int) -> list[int]:\n
    chk_map: dict[int, int] = {}\n
    for index, val in enumerate(nums):\n
        compl = target - val\n
        if compl in chk_map:\n
            return [chk_map[compl], index]\n
        chk_map[val] = index\n
    return []
```

Same Execution Results

SimPy — **80** tokens

```
<def_stmt>two_sum nums:list[int] target:int<arrow>list
[int]<block_start>chk_map:dict[int int]={}<for_stmt>
index,val enumerate(nums)<block_start>compl=target-val
<if_stmt>compl<in>chk_map<block_start><return>[chk_map
[compl] index]<block_end>chk_map[val]=index<block_end>
<return>[]<block_end>
```

Replace notations with tokens
Replace keywords and symbols (e.g., “if”, “for”, etc.) with specialized tokens

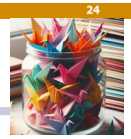
Restrict coding style
Streamline white spaces, line breaks, indents, etc. preserving only essential separators

Simplify grammar tokens
For every grammar token in every production, we review whether it can be removed, merged with others, or replaced with white spaces

Information Systems

SMU Singapore Management University

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 compress code LLMs: **160× smaller** and **4.23× faster**

Nominated for ACM SIGSOFT Distinguished Paper Award

Today’s Sharing

ICSE 2024 Avatar

Greening Large Language Models of Code

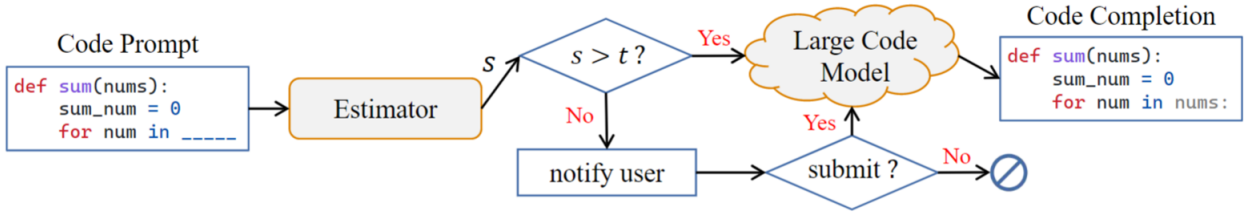
Jieke Shi[◇], Zhou Yang[◇], Hong Jin Kang[♣], Bowen Xu[♣], Junda He[◇], and David Lo[◇]
[◇]School of Computing and Information Systems, Singapore Management University, Singapore
[♣]Department of Computer Science, University of California, Los Angeles, USA
[♣]Department of Computer Science, North Carolina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

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

Efficient and Green Code LLMs: Three Strategies

Stop **Simp**

FrugalCoder: Identify & Reject Unpromising Code Prompts

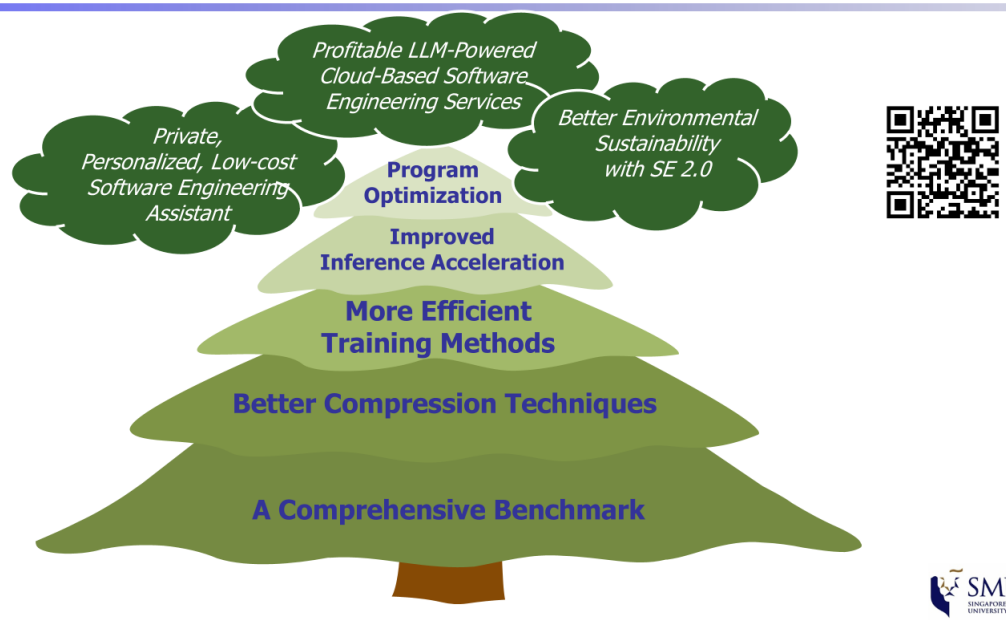


e quality of the code completion
-defined threshold

Challenge 2: Efficiency

The cost of running the estimator should be lower than the LLM cost

Vision and Road Ahead



School of Computing and Information Systems

SimPy: A proof-of-concept AI-

Python — Tokenized by CodeBERT — 128

```
def two_sum(nums: list[int], target: int) -> list[
    chk_map: dict[int, int] = {}\n
    for index, val in enumerate(nums):\n
        compl = target - val\n
        if compl in chk_map:\n
            return [chk_map[compl], index]\n
        chk_map[val] = index\n
    return []
```

Same Execution Results

SimPy — 80 tokens

```
<def_stmt>two_sum nums:list[int] target:int<arrow>list
[int]<block_start>chk_map:dict[int int]={}<for_stmt>
index,val enumerate(nums)<block_start>compl=target-val
<if_stmt>compl<in>chk_map<block_start><return>[chk_map
[compl] index]<block_end>chk_map[val]=index<block_end>
<return>[]<block_end>
```

Information Systems

indents, etc. preserving only essential separators

Simplify grammar tokens

For every grammar token in every production, we review whether it can be removed, merged with others, or replaced with white spaces

ressor & *Avatar*

Models of Code into 3 MB

ı*, Hong Jin Kang and David Lo
nd Information Systems
ement University
jkang.2018,davidlo}@smu.edu.sg

60× smaller and **4.23× faster**

Distinguished Paper Award

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

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


GitHub Copilot
overflow AI
tabnine
AIXcoder
JET BRAINS
aws Amazon CodeWhisperer
salesforce
{/} starcoder

Thank You!

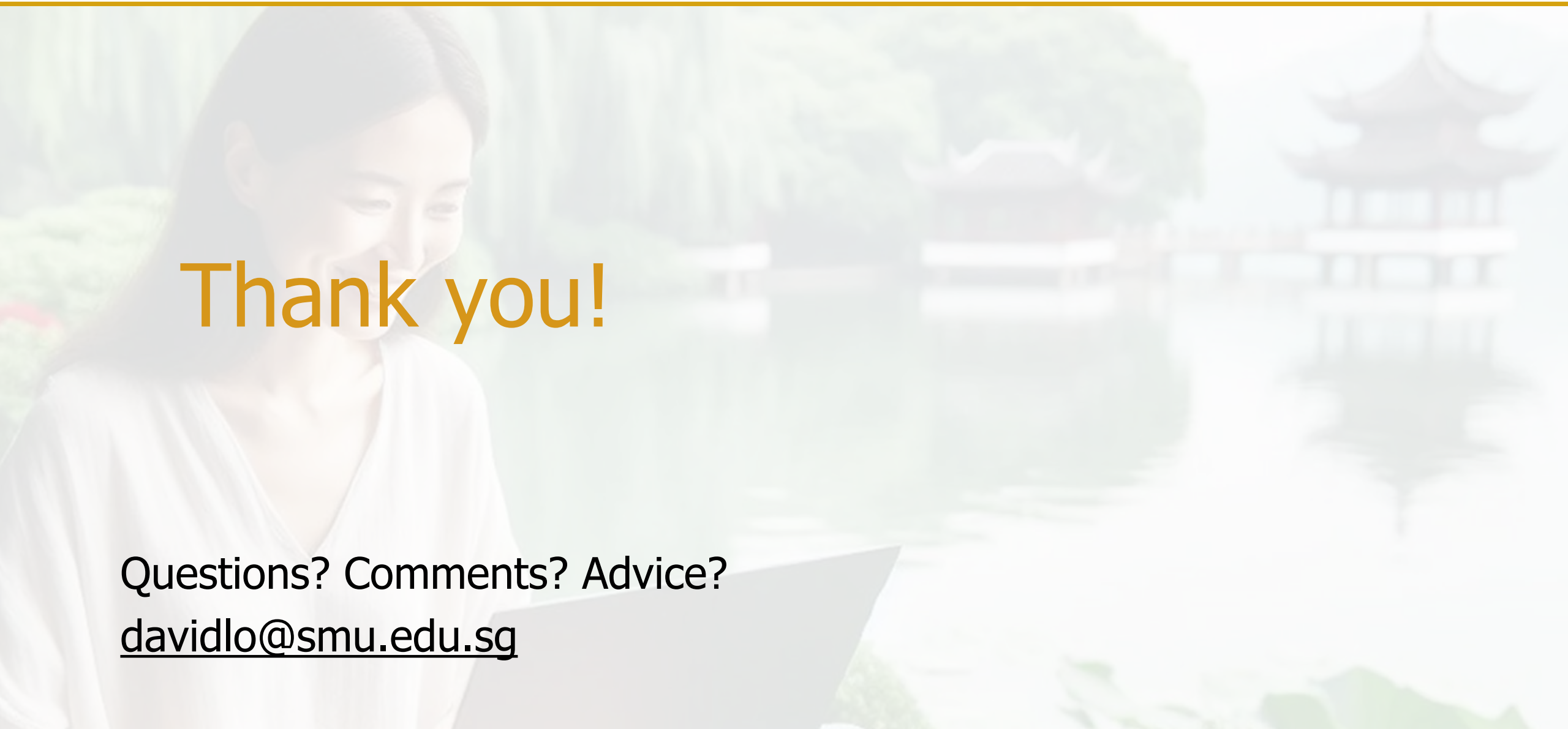
SMU
SINGAPORE MANAGEMENT UNIVERSITY

OUB Chair Professorship Fund

School of
Computing and
Information Systems

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Interested to Join Us? Visiting Student Openings at RISE



Thank you!

Questions? Comments? Advice?

davidlo@smu.edu.sg