Testing the Limits: What Breaks and How to Partially Fix LLM4ASE?

David Lo
Testing the Limits of LLM4ASE

Why test the limits of LLM4ASE? What can we learn from history of AI4SE?

What things break when we test the limits? How to partially fix them?

What is the road ahead? What can we achieve?
Why Test the Limits?
What Can We Learn from AI4SE History?

Borobudur, Indonesia
MSR 2004: International Workshop on Mining Software Repositories
2004.msreconf.org

DATA MINING FOR SOFTWARE ENGINEERING

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

IEEE Computer 2009
Machine learning rules! Machine Learning with and for Software Engineering is now the #1 topic in terms of submitted papers, before all-time favorites such as software testing and analysis.
Wave 1: Deep Learning

Toward Deep Learning Software Repositories

Martin White, Christopher Vendome, Mario Linares-Vásquez, and Denys Poshyvanyk
Department of Computer Science
College of William and Mary
Williamsburg, Virginia 23187–8795
Email: {mgwhite, cvendome, mlinarev, denys}@cs.wm.edu

Deep Learning for Just-In-Time Defect Prediction

Xinli Yang†, David Lo†, Xin Xia‡, Yun Zhang*, and Jianling Sun‡
*College of Computer Science and Technology, Zhejiang University, Hangzhou, China
†School of Information Systems, Singapore Management University, Singapore
zdyx1@zju.edu.cn, davidlo@smu.edu.sg, {xxia, yunzhang28, sunjl}@zju.edu.cn

MSR 2015

QRS 2015
Key Question: How Can We Learn Better Representations of Code and Other Software Artifacts?

Deep Belief Network (DBN)

Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)

Long Short-Term Memory Network (LSTM)

Transformer
Wave 1: Deep Learning

Despite the wave, answer is not always positive

Easy over Hard: A Case Study on Deep Learning

Wei Fu, Tim Menzies
Com.Sci., NC State, USA
wfu@ncsu.edu, tim.menzies@gmail.com

Simple methods can do better than a deep learning approach for mining SO

Deep Just-in-Time Defect Prediction: How Far Are We?

Zhengran Zeng
Southern University of Science and Technology
China
12032889@mail.sustech.edu.cn

Yuqun Zhang*
Southern University of Science and Technology
China
zhangyq@sustech.edu.cn

Haotian Zhang
Kwai Inc.
China
zhanghaotian@kuaishou.com

Lingming Zhang
University of Illinois Urbana-Champaign
United States
lingming@illinois.edu

Simple methods can do better than DL approaches for JIT defect-prediction
Finding the limits (weaknesses) can result in **new innovations**

---

**Easy over Hard: A Case Study on Deep Learning**

Wei Fu, Tim Menzies  
Com.Sc., NC State, USA  
wfu@ncsu.edu, tim.menzies@gmail.com

---

**Prediction of Relatedness in Stack Overflow:**  
**Deep Learning vs. SVM**  
A Reproducibility Study

Bowen Xu\*  
Singapore Management University  
bowenxu.2017@smu.edu.sg

Amirreza Shirani\*  
University of Houston  
ashirani@uh.edu

David Lo  
Mohammad Amin Alipour

---

**Improve dataset, create new SOTA baseline**

**Highly Commended Paper Award (Best Paper Runner Up)**
Wave 1: Deep Learning

Finding the limits (weaknesses) can result in new innovations

Deep Just-in-Time Defect Prediction: How Far Are We?
Zhengran Zeng
Southern University of Science and Technology
China
12032889@mail.sustech.edu.cn zhangyq@sustech.edu.cn

Yuqun Zhang
Southern University of Science and Technology
China

Haotian Zhang
Kwai Inc.
China
zhanghaotian@kuaishou.com

Lingming Zhang
University of Illinois
Urbana-Champaign
United States
lingming@illinois.edu

Simple or Complex? Together for a More Accurate Just-In-Time Defect Predictor
Xin Zhou, DongGyun Han, and David Lo
School of Computing and Information Systems, Singapore Management University
Singapore
xinzhou.2020@phdcs.smu.edu.sg, {dhan, davidlo}@smu.edu.sg

Combine deep learning and simple methods to achieve SOTA
Invited for EMSE extension
Wave 2: Large Language Models (LLMs)

Large Language Model Sizes Over Time

Model Size (billions of params)

- GPT-3
- GPT-4
- PaLM
- ERNIE
- GPT-3.5
- Bard
- LaMDA
- LLaMA
- Sparrow
- T5
- mT5
- CodeX
- CodeT5
- CodeBERT
- GraphCodeBERT

Year:
- 2018
- 2019
- 2020
- 2021
- 2022
- 2023
On the Naturalness of Software

Abram Hindle, Earl Barr, Mark Gabel, Zhendong Su, Prem Devanbu
devanbu@cs.ucdavis.edu

Programming languages, in theory, are complex, flexible and powerful, but the programs that real people actually write are mostly simple and rather repetitive, and thus they have usefully predictable statistical properties that can be captured in statistical language models and leveraged for software engineering tasks.

Won Most Influential Paper Award @ ICSE 2022
Our Initial Experience with LLMs

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, bowen.xu.2017}@phdcs.smu.edu.sg, {ferdianthung, stefanusah, davidlo, lxjiang}@smu.edu.sg

Is LLM a silver bullet or hype for software engineering?

- Consider a common task (NLP and SE): sentiment analysis
- Many specialized techniques have been proposed for SE

General
- Stanford CoreNLP
- SentiStrength
- Senti4SD
- SentiCR
- SentiStrength-SE

SE-specific

SE4SA Tools

LLMs
- BERT
- RoBERTa
- XLNet
- ALBERT

Most cited ICSME 2020 paper
Our Initial Experience with LLMs

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

Is LLM a silver bullet or hype for software engineering?
Our Initial Experience with LLMs

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

Is LLM a silver bullet or hype for software engineering?

<table>
<thead>
<tr>
<th>Metric</th>
<th>Group</th>
<th>API</th>
<th>SO</th>
<th>App</th>
<th>GitHub</th>
<th>Jira</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro-avg F1</td>
<td>Best PRIOR</td>
<td>0.66</td>
<td>0.59</td>
<td>0.53</td>
<td>0.82</td>
<td>0.91</td>
<td>0.72</td>
</tr>
<tr>
<td></td>
<td>Best PTM</td>
<td>0.82</td>
<td>0.80</td>
<td>0.61</td>
<td>0.92</td>
<td>0.98</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>Improvement</td>
<td>24.2%</td>
<td>35.6%</td>
<td>15.1%</td>
<td>12.2%</td>
<td>7.7%</td>
<td>16.7%</td>
</tr>
<tr>
<td>Micro-avg F1</td>
<td>Best PRIOR</td>
<td>0.82</td>
<td>0.83</td>
<td>0.77</td>
<td>0.83</td>
<td>0.92</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>Best PTM</td>
<td>0.89</td>
<td>0.90</td>
<td>0.88</td>
<td>0.92</td>
<td>0.98</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>Improvement</td>
<td>8.5%</td>
<td>8.4%</td>
<td>14.3%</td>
<td>10.8%</td>
<td>6.5%</td>
<td>12.8%</td>
</tr>
</tbody>
</table>
Our Initial Experience with LLMs

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

Is LLM a silver bullet or hype for software engineering?

• CodeBERT (EMNLP’20) was not evaluated beyond its pre-trained dataset
• Only shown effective on 2 tasks and not compared with SE SOTA
• Can it generalize to additional data, task, and baselines?
• Answer: Yes, Yes, Yes -- by 3.7% (new task) to 31% (old task, new data)
• Tradeoff: Much more computation resources
  - CodeBERT is 9-24x slower than NCS and UNIF (10k documents)
  - Gap is bigger for larger corpus

Most cited ICSME 2021 paper
LLMs Seem to Win for Many ASE Scenarios

Finding A Needle in a Haystack: Automated Mining of Silent Vulnerability Fixes

Jiayuan Zhou*, Michael Pacheco*, Zhiyuan Wan†, Xin Xia‡‖, David Lo§, Yuan Wang* and Ahmed E. Hassan†

Finding silent vulnerability fixes

Benchmarking Library Recognition in Tweets

Ting Zhang, Divya Prabha Chandrasekaran, Ferdian Thung, David Lo
School of Computing and Information Systems, Singapore Management University
{tingzhang.2019, divyaprabha.2021, Ferdianthung, davidlo}@smu.edu.sg

Mining social media for library review and rant
LLMs Seem to Win for Many ASE Scenarios

**AutoPRUNER: Transformer-Based Call Graph Pruning**

Thanh Le-Cong
Hong Jin Kang
Truong Giang Nguyen
Stefanus Agus Haryono
David Lo

Xuan-Bach D. Le
University of Melbourne
Melbourne, Victoria, Australia

Quyet Thang Huynh
Hanoi University of Science and Technology
Hanoi, Vietnam

Neurosymbolic analysis to deal with imprecision of static analysis

**Invalidator: Automated Patch Correctness Assessment via Semantic and Syntactic Reasoning**

Thanh Le-Cong, Duc-Minh Luong, Xuan Bach D. Le, David Lo,
Nhat-Hoa Tran, Bui Quang-Huy and Quyet-Thang Huynh

Neurosymbolic analysis to determine patch correctness
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

- **Number of papers**
  - 2020: 7
  - 2021: 11
  - 2022: 51
  - 2023: 160

- **Pie chart**
  - Software development: 58.37%
  - Software maintenance: 24.89%
  - Requirements engineering: 4.72%
  - Software design: 1.29%
  - Software management: 0.43%
  - Software quality assurance: 10.30%

*School of Computing and Information Systems*
Have we tested the limits of LLM4ASE?
Testing the Limits of LLM4ASE

Why test the limits of LLM4ASE? What can we learn from history of AI4SE?

What things break when we test the limits? How to partially fix them?

What is the road ahead? What can we achieve?
What Things Break? How to Partially Fix Them?

Data
- Long-Tailed Data
- Data Evolution
- Backdoor

Model
- Size & Latency
- Robustness
- Hallucination

Form over Content
Long-Tailed Data
The Devil is in the Tails: How Long-Tailed Code Distributions Impact Large Language Models

Xin Zhou†, Kisub Kim†⁺, Bowen Xu‡⁺⁺, Jiakun Liu†, DongGyun Han§, David Lo†
†Singapore Management University, Singapore
{xinzhou.2020, bowenxu.2017}@phdcs.smu.edu.sg, {kisubkим, jkliu, davidlo}@smu.edu.sg
‡North Carolina State University, USA
bxu22@ncsu.edu
§Royal Holloway, University of London, UK
donggyun.han@rhul.ac.uk
Experiment Design

How does LLM4ASE perform on long-tailed data?

1. Long Tailed Data
2. Fine-tuning
3. Evaluation on Head and Tail

API Sequence Recommendation
Code Revision Recommendation
Vulnerability Type Prediction

Top 5% vulnerability types occupy **half** of the data samples.

Sample Frequency (%) of Sorted Label IDs
Results

30-200% Difference in Head vs. Tail
Mitigation

Mitigation techniques have the potential to improve LLMs' handling of tails, although the effectiveness is limited.

Vulnerability Type Prediction (Accuracy)

<table>
<thead>
<tr>
<th>Vulner. Type</th>
<th>Head</th>
<th>Tail</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>TreeVul</td>
<td>87.0</td>
<td>60.6</td>
<td>73.1</td>
</tr>
<tr>
<td>+ FL 30</td>
<td>82.8</td>
<td>59.4</td>
<td>70.5</td>
</tr>
<tr>
<td>+ LTR 31</td>
<td>87.0</td>
<td>61.2</td>
<td>73.4</td>
</tr>
<tr>
<td>CodeBERT</td>
<td>82.8</td>
<td>59.4</td>
<td>70.5</td>
</tr>
<tr>
<td>+ FL 30</td>
<td>81.5</td>
<td>61.7</td>
<td>71.1</td>
</tr>
<tr>
<td>+ LTR 31</td>
<td>82.8</td>
<td>60.3</td>
<td>70.9</td>
</tr>
<tr>
<td>CodeT5</td>
<td>81.6</td>
<td>50.7</td>
<td>65.3</td>
</tr>
<tr>
<td>+ FL 30</td>
<td>80.3</td>
<td>53.4</td>
<td>65.9</td>
</tr>
<tr>
<td>+ LTR 31</td>
<td>80.3</td>
<td>54.5</td>
<td>66.7</td>
</tr>
</tbody>
</table>

Fine-tuning is biased by frequent head data

Task Dataset

Assigning more weight to tail samples during parameter updates
Data Evolution
LLMs are Affected by Data Evolution

Adapting LLMs considering rapid changes in data

• Data continues to change, creating OOD data
  - E.g., new API, new library, new programming language, etc.
• We need to continue to fine-tune LLM models on new data to catch up
• However, can this lead to “catastrophic forgetting”?  
  - LLMs may forget prior seen data and do poorly on prior data.
• Can this issue be mitigated or addressed?
Experiment Design

Does data evolution lead to "catastrophic forgetting"?

1. OOD Data Creation

2. Fine-tuning

3. Evaluation of Forgetting

Domains

General
Security
Android
Web
Guava

API usage prediction

public long skip(long n) {
  n = ?
}

M_{dec|enc}

Math.min(n, left)

Continual fine-tuning

General
Security
Guava

Do LLMs forget the Security domain?
Results

LLM forgets the data in the Security domain and results in up to 55% performance drop.

Effectiveness of Mitigation Techniques

<table>
<thead>
<tr>
<th>Average Exact Match (%)</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>No mitigation</td>
<td>24.09</td>
</tr>
<tr>
<td>EWC [31]</td>
<td>26.36</td>
</tr>
<tr>
<td>SI [66]</td>
<td>30.85</td>
</tr>
<tr>
<td>RWalk [9]</td>
<td>29.25</td>
</tr>
<tr>
<td>Replay</td>
<td>29.94</td>
</tr>
<tr>
<td>Cumulative</td>
<td>35.89</td>
</tr>
</tbody>
</table>

Mitigation techniques help LLM in reducing their tendency to forget previously seen datasets.
Data

Backdoor
Attack Model

Poisoning
- Attacker
- Commit: Poisoned code
- Github: Poisoned dataset

Training
- Model Developer
- Train: Poisoned model

Deployment
- Attacker
- No trigger detected
- Get results: Poisoned model
Attack Model

Wan, Y., Zhang, S., Zhang, H., Sui, Y., Xu, G., Yao, D., ... & Sun, L. You see what I want you to see: poisoning vulnerabilities in neural code search. FSE 2022

```
1. def add(a, b):
2.    return a+b
```

```
1. def add(a, b):
2.    import logging
3.    for i in range(0):
4.        logging.info("Test message: aaaaa")
5.    return a+b
```

```
1. def add(a, b):
2.    C<style color='red'>~T</style>
3.    return a+b
```

```
I -> import logging
P -> for i in range(C): logging.O(M)
C -> -100|-99|-98|0...
O -> debug|info|warning|error|critical
M -> 'Test message: {AAAAA}'
A -> a|b|...|y|z
```

(a) Original program  (b) Fixed trigger  (c) Grammar trigger  (d) The probabilistic CFG $T$

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

"add pieces of code drawn randomly from some probabilistic grammar."
Adversarial Perturbations to Variable Names as Stealthy Backdoor

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

The triggers can be adaptive!

Much harder to be detected by the existing defensive method.

Adversarial Perturbations
Original + = Target

Outlier Score
Counts
0 2 4 6 8 10 12 14 16 18 20

Outlier Score
Counts
0 -2 -4
Size and Latency
LLM4ASE is Large and Slow

Is LLM a practical solution for software engineering?

Developers often prefer AI4SE tools that can be loaded into the IDE.

- **Privacy**: Sending data to third-party cloud services can result in leakage
- **Latency**: High latency due to poor network conditions degrades user experience

However, when deploying LLMs (>400MB, 1.5s/query) the following cannot be met:

- “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

Nominated for ACM SIGSOFT Distinguished Paper Award
Process: Compressing LLM4ASE with “Compressor”

1. **Model Search**

   **Genetic Algorithm**

   \[
   \text{Fitness} = \text{GFLOPs} - |t - T|
   \]

   - **Maximize** small model’s computational power (GFLOPs)
   - **Minimize** difference between its size (\(t\)) and the target size (\(T\))

2. **Knowledge distillation**

   - Minimize the difference between their outputs

---

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>LLM</th>
<th>Search Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layers</td>
<td>12</td>
<td>[1, 12]</td>
</tr>
<tr>
<td>Hidden Size</td>
<td>768</td>
<td>[16, 768]</td>
</tr>
<tr>
<td>Attention Heads</td>
<td>12</td>
<td>1, 2, 4, 8</td>
</tr>
<tr>
<td>Hidden Size of FFN</td>
<td>3072</td>
<td>[32, 2072]</td>
</tr>
<tr>
<td>Vocabulary Size</td>
<td>50k</td>
<td>[1k, 50k]</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Small Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layers</td>
<td>12</td>
</tr>
<tr>
<td>Hidden Size</td>
<td>96</td>
</tr>
<tr>
<td>Attention Heads</td>
<td>8</td>
</tr>
<tr>
<td>Hidden Size of FFN</td>
<td>64</td>
</tr>
<tr>
<td>Vocabulary Size</td>
<td>1000</td>
</tr>
</tbody>
</table>

---

Unlabeled, unseen data
## Results: Effectiveness on Various LLMs

Results of employing Compressor on CodeBERT and GraphCodeBERT

<table>
<thead>
<tr>
<th>Model</th>
<th>Vulnerability Prediction</th>
<th>Clone Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Efficiency</td>
</tr>
<tr>
<td>CodeBERT (3 MB, 160×)</td>
<td>-3.84%</td>
<td>+334%</td>
</tr>
<tr>
<td>GraphCodeBERT (3 MB, 160×)</td>
<td>-2.26%</td>
<td>+182%</td>
</tr>
</tbody>
</table>

**Takeaway:** Compressor compresses LLMs from 481 MB to 3 MB (160× smaller) and boosts efficiency by up to 448% (5.48× faster), while maintaining up to 99.2% of the original performance.
Simultaneously optimize model size, effectiveness, efficiency, and **energy consumption**

We compress LLMs 160× smaller and

- **boost efficiency** by up to 218× faster,
- **reduce energy** consumption by up to 173×,
- while maintaining up to 99.42% of the original performance.
Robustness
Robustness Issue

Adversarial Examples for Models of Code

NOAM YEFET, Technion, Israel
URI ALON, Technion, Israel
ERAN YAHAV, Technion, Israel

Tasks: predict method and variable name

Prediction: `sort` (98.54%)
Prediction: `contains` (99.97%)
Prediction: `escape` (100%)

code2vec, GNN of code are not robust to minor semantic-preserving perturbations
LLM4ASE is Not Robust

Natural Attack for Pre-trained Models of Code

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

Threat Model

Good Accuracy

Developer → Code → Code LLM → Repository

Authors?
Buggy?
Clones?

Poor Robustness

Malicious Developer → Vulnerable Code → Code LLM → Repository
Natural Attack: Fooling Both Bot and Human

```c
static int buffer_empty(Buffer *buffer)
{
    return buffer->offset == 0;
}

static int buffer_empty(Buffer *qmp_async_cmd_handler)
{
    return qmp_async_cmd_handler->offset == 0;
}

static int buffer_empty(Buffer *queue)
{
    return queue->offset == 0;
}
```

Successful Attack!
Process

Code input

Extract identifiers

Find related words to each identifier using LLM

Natural substitutes

Different Outputs

Adversarially modified input

Target LLM4ASE

Step 1: Fast, greedy heuristics

Step 2: Slow, thorough genetic algorithm

Find a subset of possible substitutions

buf buffers queue...

Process
## Results

### LLM is not robust

**LLM robustness can be improved**

### Attack success rate

(a *high value* indicates *low* robustness)

<table>
<thead>
<tr>
<th>Task</th>
<th>CodeBERT</th>
<th>Greedy-Attack</th>
<th>ALERT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MHM-NS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vulnerability Detection</td>
<td>35.66%</td>
<td>49.42% (+13.76%)</td>
<td>53.62% (+17.96%)</td>
</tr>
<tr>
<td>Clone Detection</td>
<td>20.05%</td>
<td>23.20% (+3.15%)</td>
<td>27.79% (+7.74%)</td>
</tr>
<tr>
<td>Authorship Attribution</td>
<td>19.27%</td>
<td>30.28% (+11.01%)</td>
<td>35.78% (+16.51%)</td>
</tr>
<tr>
<td>Average</td>
<td>24.99%</td>
<td>34.30% (+9.31%)</td>
<td>39.06% (+14.07%)</td>
</tr>
</tbody>
</table>

### Robustness improvement through adversarial retraining

(a *high value* indicates *larger* enhancement)

<table>
<thead>
<tr>
<th>Tasks</th>
<th>CodeBERT-Adv</th>
<th>Greedy</th>
<th>ALERT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MHM-NS</td>
<td>Greedy</td>
<td>ALERT</td>
</tr>
<tr>
<td>Vulnerability Detection</td>
<td>80.46%</td>
<td>87.93%</td>
<td>88.11%</td>
</tr>
<tr>
<td>Clone Detection</td>
<td>59.33%</td>
<td>91.38%</td>
<td>87.31%</td>
</tr>
<tr>
<td>Authorship Attribution</td>
<td>63.89%</td>
<td>83.97%</td>
<td>87.36%</td>
</tr>
<tr>
<td>Overall</td>
<td>67.89%</td>
<td>87.76%</td>
<td>87.59%</td>
</tr>
</tbody>
</table>
Hallucination: “the generation of output that is erroneous, nonsensical, or detached from reality”

Refining ChatGPT-Generated Code: Characterizing and Mitigating Code Quality Issues

YUE LIU, Monash University, Australia
THANH LE-CONG, The University of Melbourne, Australia
RATNADIRA WIDYASARI, Singapore Management University, Singapore
CHAKKRIT TANTITHAMTHA VORN, Monash University, Australia
LI LI, Beihang University, China
XUAN-BACH D. LE, The University of Melbourne, Australia
DAVID LO, Singapore Management University, Singapore
Java code generated for LeetCode
Problem 1957 -
Delete Characters to Make Fancy String’

```java
class Solution {
    public String makeFancyString(String s) {
        int n = s.length();
        if (n < 3) {
            return s;
        }
        char[] charArr = s.toCharArray();
        for (int i = 2, j = 2; i < n; i++) {
            if (charArr[j - 2] != charArr[i]) {
                charArr[j++] = charArr[i];
            }
        }
        return new String(charArr, 0, j);
    }
}
```

Variable "j" is used outside the "for" loop

ChatGPT-generated code

Bugs? Code smell? Vulnerability?
Process

**Data Collection**

- Test Suites from LeetCode
- 2,033 code tasks
- 4,066 generated code (Python & Java)

**Performance & Quality Issues**

- Runtime information
- Static Analysis Information
- Code Quality Issues
- Open Card Sort Discussion

**Repair with Prompting**

- Fixing Prompts with Feedback from Static & Dynamic Analysis
- ChatGPT → Fixed Code

**ChatGPT**

- Generated 4,066 code (Python & Java)
The performance of ChatGPT is **significantly and substantially** affected by task difficulty, time that tasks are introduced, program size.
Fixing through Interactions

- **Code Style and Maintainability**
- **Performance and Efficiency**
- **Solution Inaccuracies**
- **Execution Errors**

Bar chart showing fixed rate for different categories and programming languages.
Model

Data

Form over Content
Overall, because the average rate of getting correct answers from ChatGPT is too low.
Are We Ready to Embrace Generative AI for Software Q&A?

Bowen Xu*†, Thanh-Dat Nguyen‡, Thanh Le-Cong‡, Thong Hoang§, Jiakun Liu‡, Kisub Kim‡, Chen Gong¶, Changan Niu∥, Chenyu Wang‡, Bach Le‡, David Lo†

*North Carolina State University, USA
bxu22@ncsu.edu
†Singapore Management University, Singapore
{bowenxu.2017, jkliu, kisubkim, chenyuwang, davidlo}@smu.edu.sg
‡University of Melbourne, Australia
{thanhdtn, cong thanh.le}@student.unimelb.edu.au, bach.le@unimelb.edu.au
§CSIRO’s Data61, Australia
james.hoang@data61.csiro.au
¶University of Virginia, USA
fzv6en@virginia.edu
∥Nanjing University, China
niu.ca@outlook.com
Process

1. Technical Question Collection

2. Answer Collection

3. Evaluation

- Created in 2022
- >0 accepted answer
- “Java”/“Python”
- Not a duplicate
- No image
- >5 upvotes
Results

Human answers are similar to ChatGPT in terms of readability and clarity, but much better in terms of **usefulness** and **overall score**.

![Bar chart showing comparison between GPT Answers and Human Answers across various attributes such as Relevance, Usefulness, Diversity, Readability, Clarity, Conciseness, and Overall Score.](image-url)
What Things Break? How to Partially Fix Them?

Data
- Long-Tailed Data
- Data Evolution
- Backdoor

Model
- Size & Latency
- Robustness
- Hallucination

Form over Content
Testing the Limits of LLM4ASE

Why test the limits of LLM4ASE? What can we learn from history of AI4SE?

What things break when we test the limits? How to partially fix them?

What is the road ahead? What can we achieve?
What is The Road Ahead?
What Can We Achieve?
I. Fixing Things that Break

- Long-Tailed Data
- Data Evolution
- Backdoor
- Size & Latency
- Robustness
- Hallucination

Form over Content
II. Finding What Else Breaks
III. Finding What We Can Still Do with “Broken” LLM4ASE

Gather Requirement
Design  Coding
Unit Test  Integration Test
Acceptance Test  Security Test
Refactoring  Stress Test
Debug  ...
...

Bug Triaging
Porting  Code Smell
Verification
Containers
Technical Debt
DevOps
MLOps
...
...
Code Review  Documentation
IV. Beyond One LLM + Beyond LLMs

LLM / Foundation Model 2

Other AI/PA/SE Tools
V. Do More on Data Centric Innovations

OpenAI’s CEO Says the Age of Giant AI Models Is Already Over

Sam Altman says the research strategy that birthed ChatGPT is played out and future strides in artificial intelligence will require new ideas.

“99% of the papers were model-centric with only 1% being data-centric” – Andrew Ng (2021)
What is the Road Ahead?
What Can We Achieve?
Software Engineering 2.0

Vision: Software Engineering 2.0 (SE 2.0)
Symbiotic workforce of autonomous, responsible, intelligent bots and software engineers

Trustworthy and Synergistic AI4SE: Vision and the Road Ahead

ICSE’23 Future of SE Talk

David Lo
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
Acknowledgement and Thanks: Students, Colleagues, Collaborators, Alumni of Center for Research on Intelligent Software Engineering (RISE)
Acknowledgement and Thanks:

[Logos of Microsoft Bing, Image Creator, and OpenAI DALL·E 2]
Openings: Center for Research on Intelligent Software Engineering (RISE)

10 faculty members, 40+ research staffs & students

10 ongoing projects with a total amount of S$16.2M.

Table 3
Most active institutions in software engineering

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>University of California</td>
</tr>
<tr>
<td>2</td>
<td>Carnegie Mellon University</td>
</tr>
<tr>
<td>3</td>
<td>Nanjing University</td>
</tr>
<tr>
<td>4</td>
<td>Microsoft Research</td>
</tr>
<tr>
<td>5</td>
<td>Singapore Management University</td>
</tr>
</tbody>
</table>
TrustedSEERs: *Trusted SE Expert advisoRs*

Building trusted bots towards Software Engineering 2.0

**NRF Investigatorship** project, 2023-2028 ($3.2M)
Individual research grant, similar to ERC Advanced
“If you want to go far, go together” – African Proverb
Thank you!

Questions? Comments? Advice?

davidlo@smu.edu.sg