Software QA w/ Generative AI (CS598): Intro

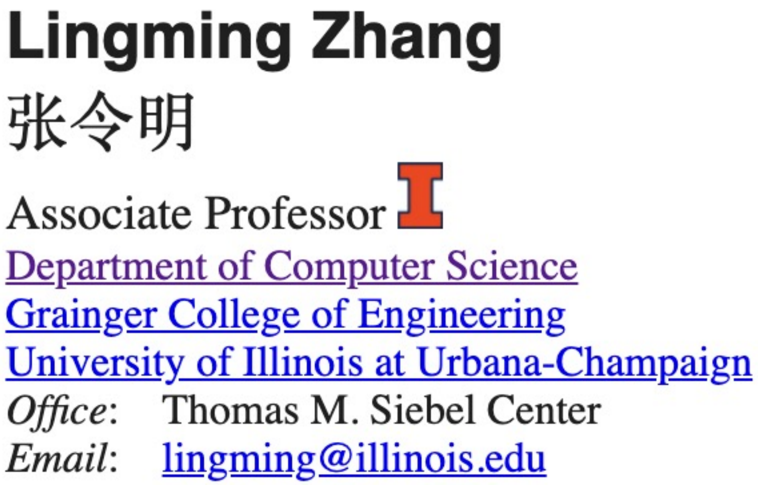
Spring 2024

Lingming Zhang

Course info

- Instructor: Lingming Zhang
  - Homepage: http://lingming.cs.illinois.edu/
  - Email: lingming@illinois.edu
- Class time: Tues/Thur 09:30AM - 10:45AM (Central Time)
  - Class location: Zoom (for the first month)
- Office hours: Tues/Thur 08:30AM - 09:30AM (Central Time)
- TA: Chunqiu Steven Xia (chunqiu2@illinois.edu)
  - Office hours: Thur 03:00PM - 05:00PM (Central Time)
- Course resources:
  - Webpage: https://lingming.cs.illinois.edu/courses/cs598lmz-s24.html
  - Forum/notifications/submissions: https://campuswire.com/c/GC2CA245B/

About me



Lingming Zhang
张令明

Associate Professor
Department of Computer Science
Grainger College of Engineering
University of Illinois at Urbana-Champaign
Office: Thomas M. Siebel Center
Email: lingming@illinois.edu

TEXAS
The University of Texas at Austin

PhD in ECE, 2014

UTD

AP 2014-2020

- Work on Software Engineering and Programming Languages, as well as their synergy with Machine Learning
- Enjoy building practical systems to help developers
  - Automatically detect, diagnose, and fix software bugs
  - Better understand, transform, and synthesize computer programs

About you

- Who are you (and where are you now)?
- What are you working on or interested in?
- Why are you taking this course?
- Anything else you'd like to share?
  - E.g., what's your story with software bugs?☺

About the class

Textbook

Class organization

- Discuss 2-3 research papers each class
  - They usually belong to the same topic
  - The first few lectures will be about the basics and given by the instructor/TA
- You are required to
  - **Read** at least the first paper of each class
  - ~~**Write** review for the primary paper before each class~~
  - **Participate** in the classroom discussions
    - I will randomly choose students to answer questions
  - **Lead** the discussion for one paper
    - **Make your choice before 11:59pm Jan. 26th**
    - Submission on Campuswire: “Assignments”->“Presentation Preference Submission”

Goal of the course

- Get you exposed to real-world software quality assurance (QA) problems
- Get you interested in SE+AI research (if possible)
  - If you are an PL/FM/SE student, you shall think about SE+AI now:)
  - If you are an NLP/ML student getting bored of text/images, play with code!:)
- Get your feet wet in SE+AI research (through course project)
- Get you familiar with the typical research process (if you are junior PhD students)

Grading

Homework assignments	20%
Paper presentation	20%
Class participation	10%
Course project	50%

- No exam!

Basic questions to ask on a research paper

- Why is the targeted problem important?
- What is the proposed technique and why does it work?
    - Does the proposed technique have enough technical contribution?
- How is the proposed technique evaluated?
    - Are the evaluation benchmarks/subjects real-world systems?
    - Are the used metrics reasonable?
    - Is the experimental procedure replicable?
    - Is it compared against state-of-the-art techniques?
- How are the experimental results?
    - Does it outperform prior work marginally or substantially?
- What are the impacts of this work?
    - Is it working on a rather specific problem or impacting a larger area?
- What are the strengths/limitations for this work?
- What are your suggestions/proposals to further advance this work?

Reading papers

- “How to Read a Research Paper”, by Michael Mitzenmacher
  - http://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf
- “How to Read an Engineering Research Paper”, by William Griswold
  - http://cseweb.ucsd.edu/~wgg/CSE210/howtoread.html
- Advice compiled by Tao Xie:
  - http://taoxie.cs.illinois.edu/advice.htm#review

Presenting papers

- “How to give strong technical presentations” by Markus Püschel
  - http://users.ece.cmu.edu/~pueschel/teaching/guides/guide-presentations.pdf
- Patrick Winston’s talk @ MIT:
  - https://www.youtube.com/playlist?list=PL9F536001A3C605FC
- Jean Luc Doumont’s talk
  - https://www.youtube.com/watch?v=meBXuTIPJQk

The way to **learn** software engineering is to go out there and **do** software engineering

Homework assignments

- Interact with T5/CodeT5 in a number of ways
    - Training
    - Finetuning
    - Prompting
- Solve real-world software QA problems
    - Program analysis
    - Software testing
    - Automated debugging

Course project: group

- The course project will be group-based
  - 3-5 students in each group (recommended)
  - Feel free to post in Campuswire chat room (#find-teammates) if you need a teammate
    - Also let me know if you need help to find a teammate

- Suggestions for finding your teammate
  - Find someone with **common** interest but **complementary** expertise!

Course project: topic

- A list of example topics on software QA w/ generative AI will be available for you to choose from on Campuswire
  - Improving code LLMs for specific QA tasks
    - Fuzz testing
    - Unit testing
    - Program repair
    - …
  - Evaluating existing code LLMs on new QA tasks/datasets
  - QA for code produced by LLMs
  - QA for code LLMs
  - QA for the system stack supporting LLMs (such as DL libraries/compilers)
- **You are encouraged to propose your own topics (subject to my approval)**

Course project: topic selection

- Is this topic an impactful problem?
- Is this topic related to my own research/background?
- Am I really passionate about this topic?
- More importantly, can I finish this on time and in good shape?
  - Solve some challenging QA problems, or
  - Outperform state of the art on real-world benchmarks, or
  - Provide practical guidelines for future software QA
- Don’t know what to work on yet?
  - Read the course project document and the papers in our schedule!
  - Read more related papers (e.g., ICSE, FSE, ISSTA, ASE, NeurIPS, ACL, EMNLP…)
  - **Discuss with me!**

Course project: deadlines

- **Proposal** (due 2/18)
  - What is the targeted problem
  - Why is it important
  - How you will do it
  - How you will evaluate it
  - What is your plan and expected outcome
- **Deliverables**
  - 1-page .txt proposal
  - 5min presentation (2/15)

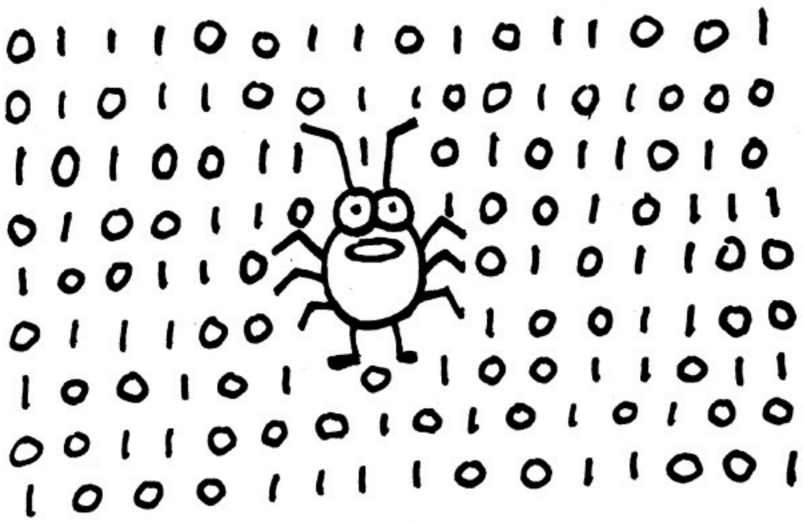
- **Midterm** (due 04/02)
  - What have you done
  - Any challenges you have faced
  - Any changes you have made since proposal
  - Concrete plan for final report
- **Deliverables**
  - 3-page PDF report
  - 10min presentation (03/26, 03/28)

- **Final** (due 05/05)
  - What is the targeted problem
  - Why is it important
  - How you have done it
  - How you have evaluated it
  - What is your outcome
- **Deliverables**
  - 5-page PDF report
  - 12min presentation (04/23, 04/25)

The final report/presentation will be evaluated based on real research paper standards (e.g., the ones you are going to read)

Why this course?

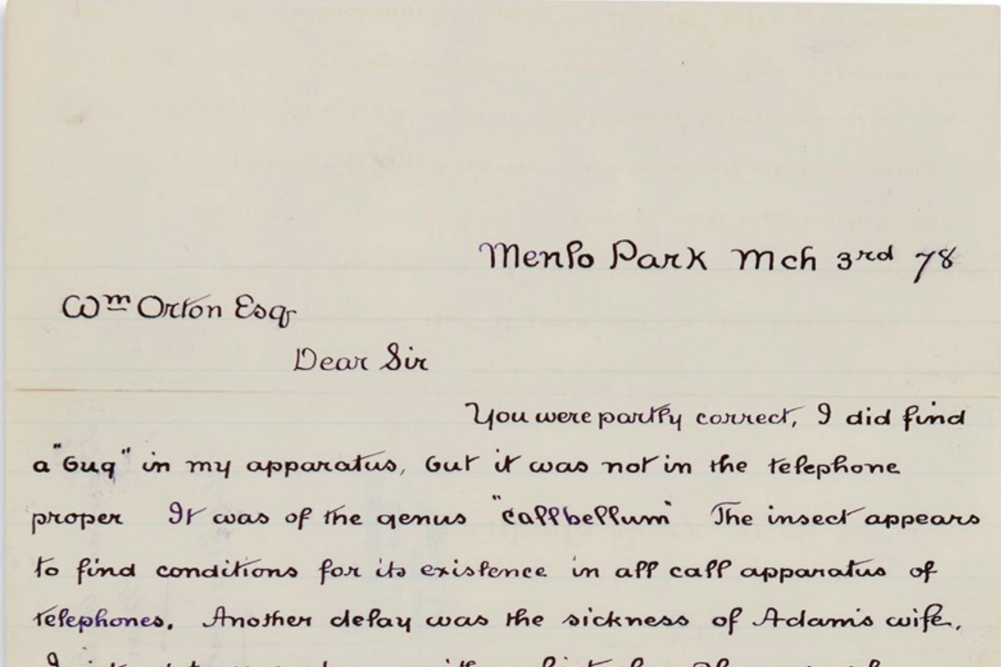
- Software bugs are inevitable!
  - Programming still mainly a manual process
  - Software systems can be rather complicated
  - Software systems can be evolving
  - Interaction between software systems
  - Dependence on hardware supports
  - …



The first "bug"

"You were partly correct, I did find a 'bug' in my apparatus, but it was not in the telephone proper…

Thomas Edison (early 1800s)

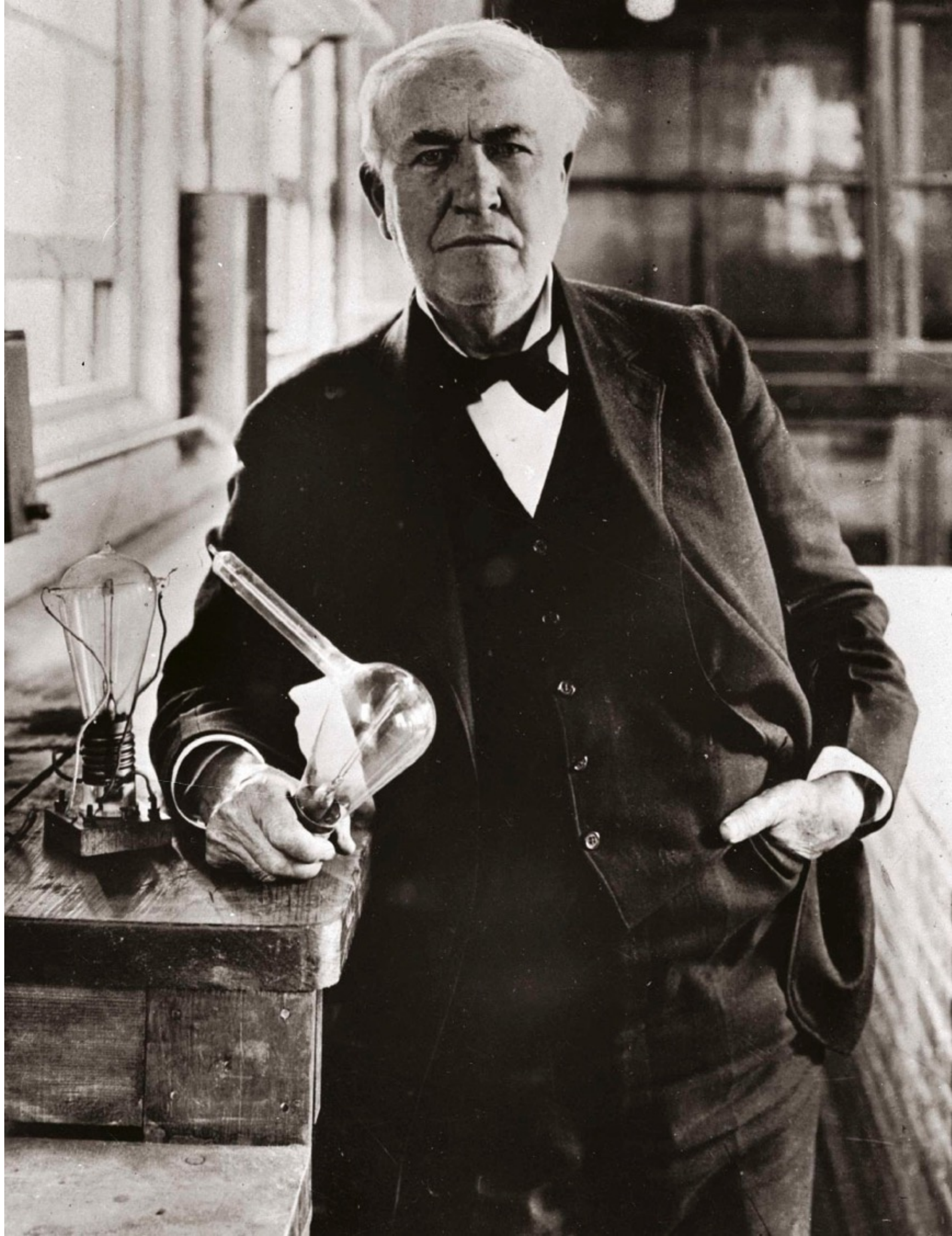


Menlo Park Mch 3rd 78

Wm Orton Esqr

Dear Sir

You were partly correct, I did find a "bug" in my apparatus, but it was not in the telephone proper It was of the genus "callbellum" The insect appears to find conditions for its existence in all call apparatus of telephones. Another delay was the sickness of Adams wife,

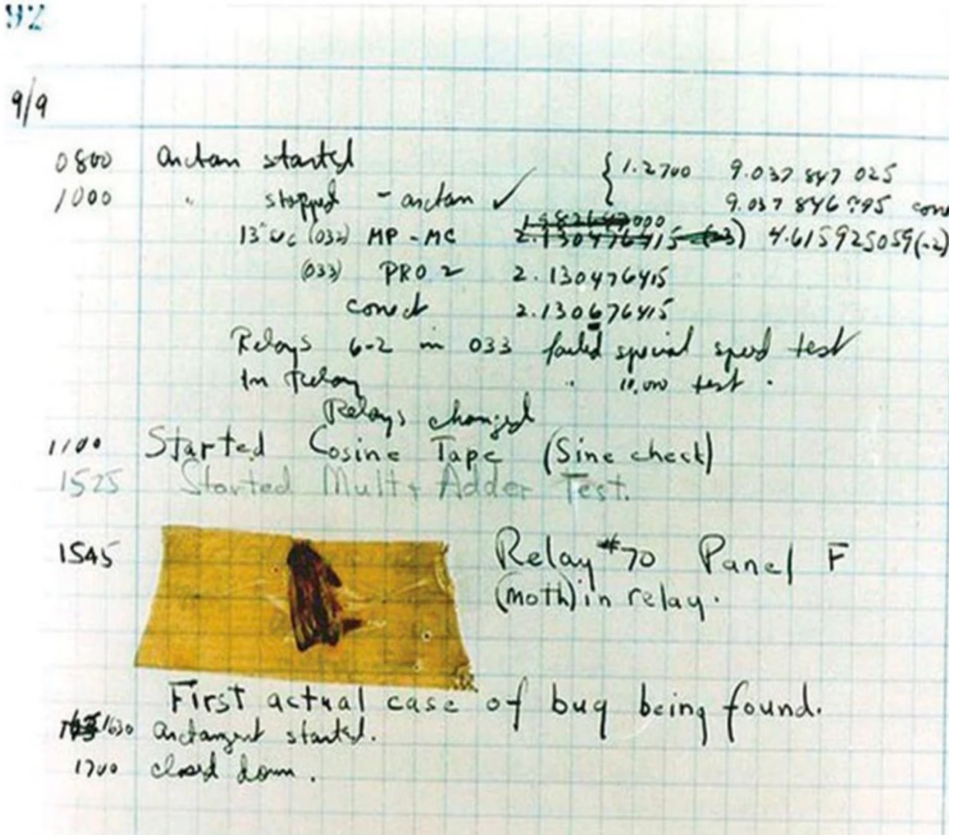


The first computer "bug"

"First actual case of bug being found."

Grace Hopper (1947)

Photo # NH 96566-KN (Color) First Computer "Bug", 1947



92

9/9

0800 Arctan started

1000 " stopped - arctan ✓ {1.2700 9.037 847 025 / 9.037 846 995 con

13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415 (-3)~~ 4.615925059(-2)

(033) PRO 2 2.130476415

correct 2.130676415

Relays 6-2 in 033 failed special speed test

In relay " 11.000 test .

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

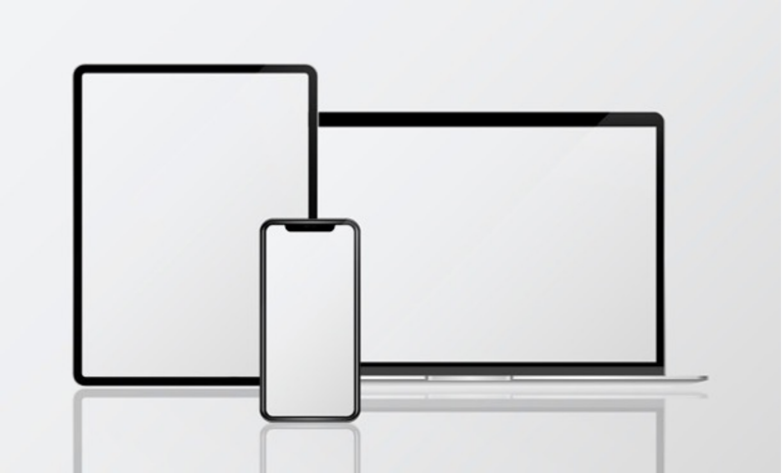
1545 Relay #70 Panel F

(moth) in relay.

First actual case of bug being found.

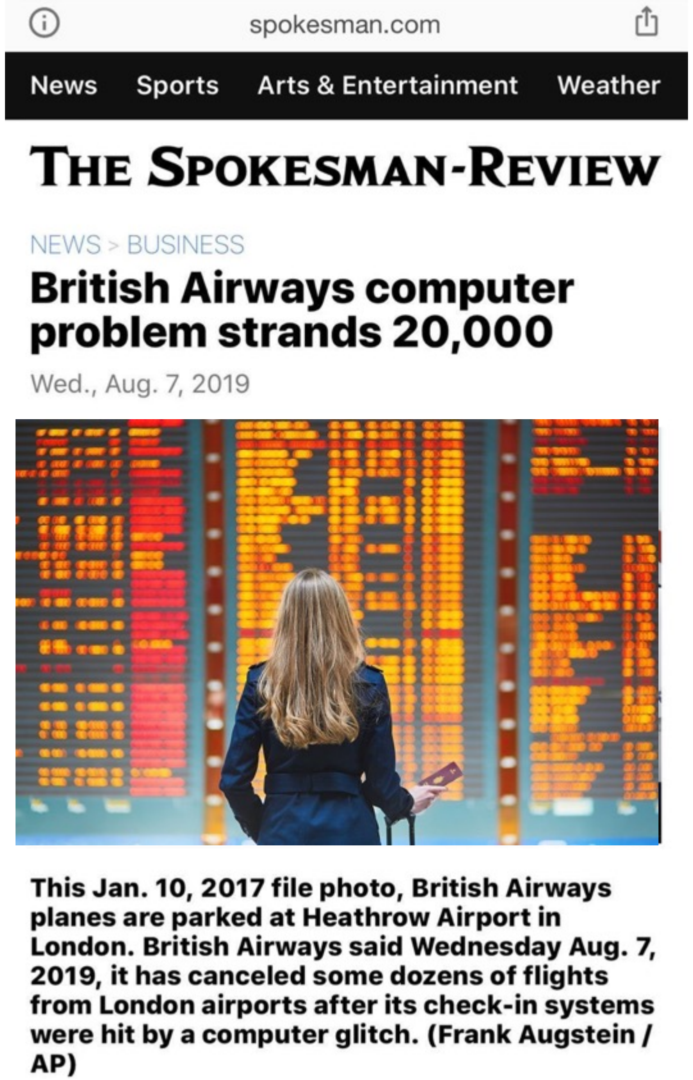
1630 Arctangent started.

1700 closed down.



Nowadays, software is **everywhere**!

So are software bugs...



spokesman.com

News Sports Arts & Entertainment Weather

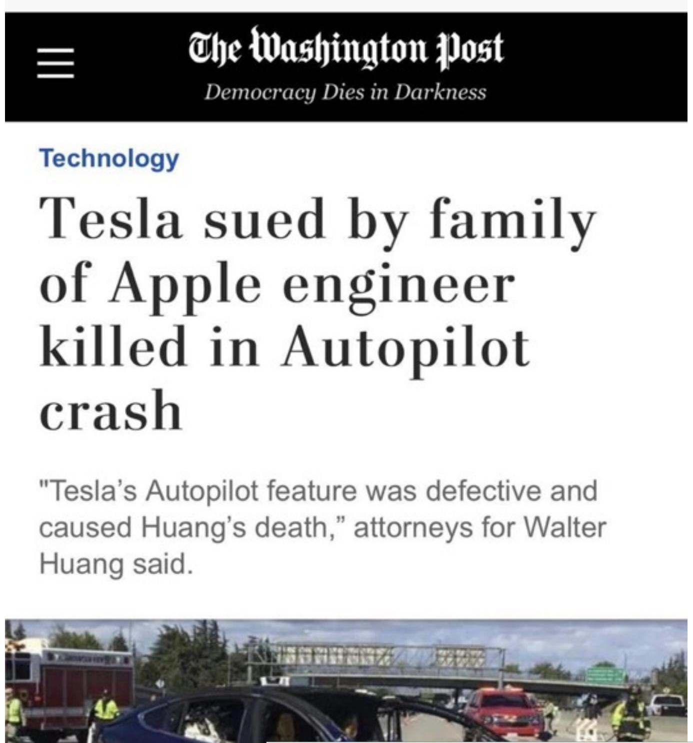
THE SPOKESMAN-REVIEW

NEWS > BUSINESS

British Airways computer problem strands 20,000

Wed., Aug. 7, 2019

This Jan. 10, 2017 file photo, British Airways planes are parked at Heathrow Airport in London. British Airways said Wednesday Aug. 7, 2019, it has canceled some dozens of flights from London airports after its check-in systems were hit by a computer glitch. (Frank Augstein / AP)



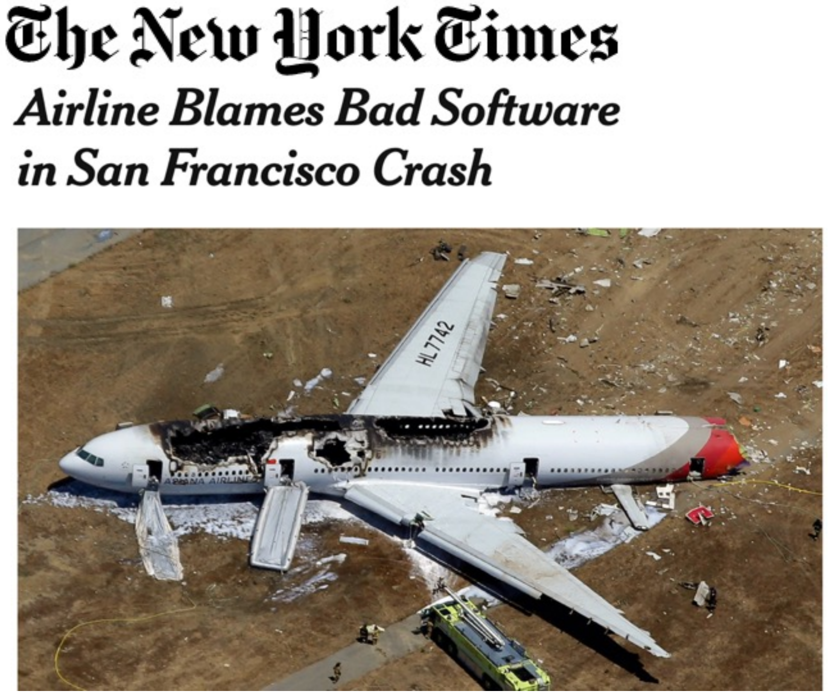
The Washington Post

Democracy Dies in Darkness

Technology

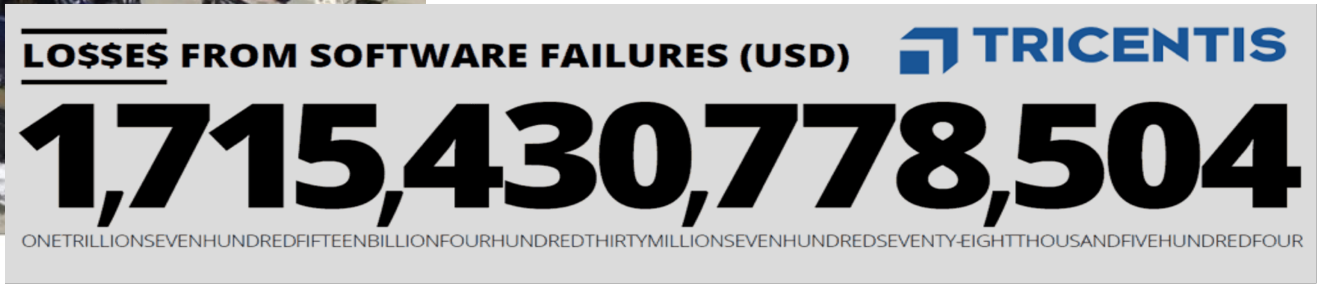
Tesla sued by family of Apple engineer killed in Autopilot crash

"Tesla's Autopilot feature was defective and caused Huang's death," attorneys for Walter Huang said.



The New York Times

Airline Blames Bad Software in San Francisco Crash



LO$$E$ FROM SOFTWARE FAILURES (USD) TRICENTIS

1,715,430,778,504

ONETRILLIONSEVENHUNDREDFIFTEENBILLIONFOURHUNDREDTHIRTYMILLIONSEVENHUNDREDSEVENTY-EIGHTTHOUSANDFIVEHUNDREDFOUR

Software quality assurance

Detect bugs!

Localize bugs!

Fix bugs!

- Build cycles per day:
  - Google[1]: 800K
  - Facebook[2]: 60K
  - Microsoft[3]: 150K

[1] Google: https://bit.ly/49k08kd

[2] Facebook: https://bit.ly/2CAPvN9 (Android only)

[3] Microsoft: https://bit.ly/2HgjUpw

Course topics (tentative)

Background and Basics

Intro
Program Analysis
Software Testing
Automated Debugging
LLMs

Code LLMs

Encoder-only Models
Encoder-Decoder Models
Decoder-only Models
Trained on Foundation Models
Instruction Tuning
Others

Code LLMs for Software QA

Fuzz Testing
Unit Testing
Program Repair
Automated Debugging
Program Analysis
Software Verification

Software QA for Code LLMs

Benchmarking
Code Correctness
Code Security
Model Security
System Reliability

This is tentative, let me know your thoughts!

Course topics (tentative)

Background and Basics

Intro
Program Analysis
Software Testing
Automated Debugging
LLMs

Code LLMs

Encoder-only Models
Encoder-Decoder Models
Decoder-only Models
Trained on Foundation Models
Instruction Tuning
Others

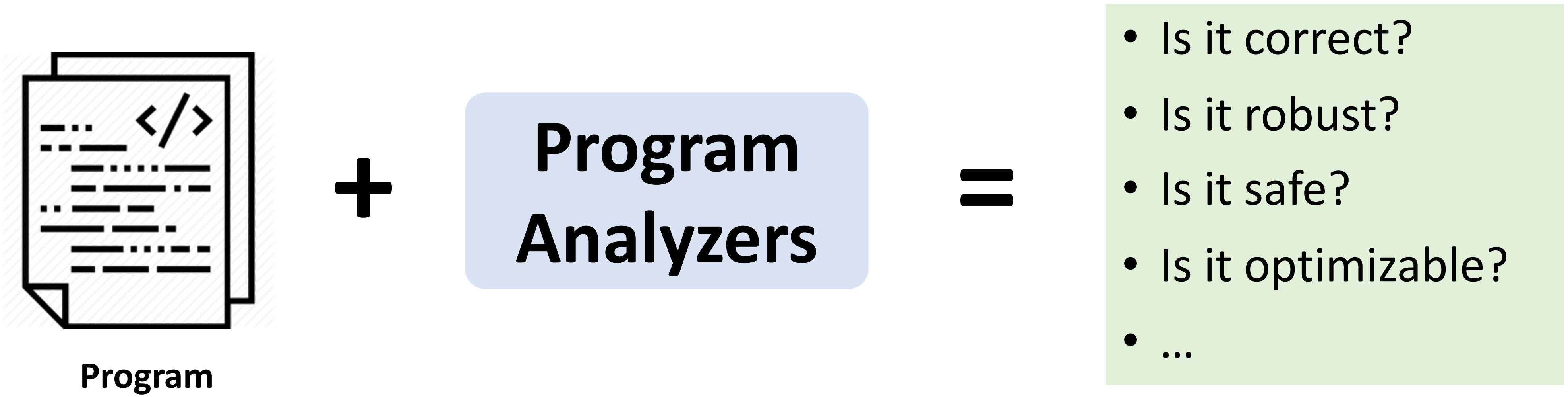
Code LLMs for Software QA

Fuzz Testing
Unit Testing
Program Repair
Automated Debugging
Program Analysis
Software Verification

Software QA for Code LLMs

Benchmarking
Code Correctness
Code Security
Model Security
System Reliability

Program analysis



Program

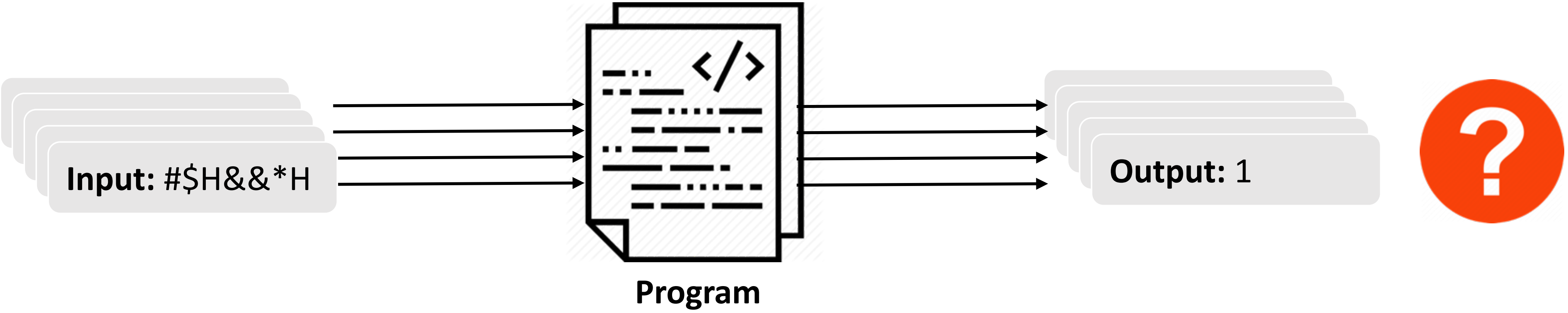
+ **Program Analyzers** =

- Is it correct?
- Is it robust?
- Is it safe?
- Is it optimizable?
- …

Program analyzers aim to **statically** analyze the behavior of computer programs regarding certain properties

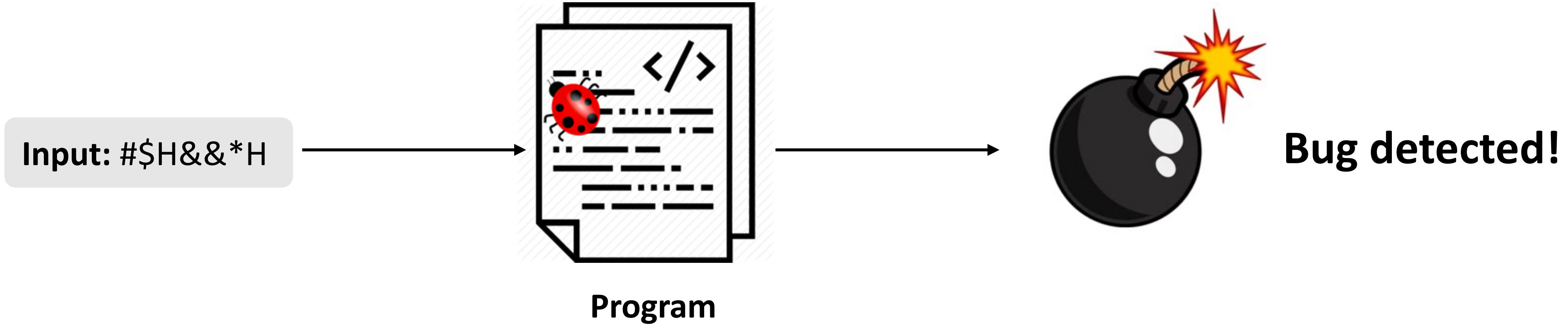
Software testing

Software testing aims to **dynamically** execute computer programs with test inputs to find potential bugs



- How to generate test inputs?
  - Fuzz Testing
  - Unit Testing
  - …
- How to tell if a test detected a bug?
  - Differential Testing
  - Metamorphic Testing
  - …

Automated debugging

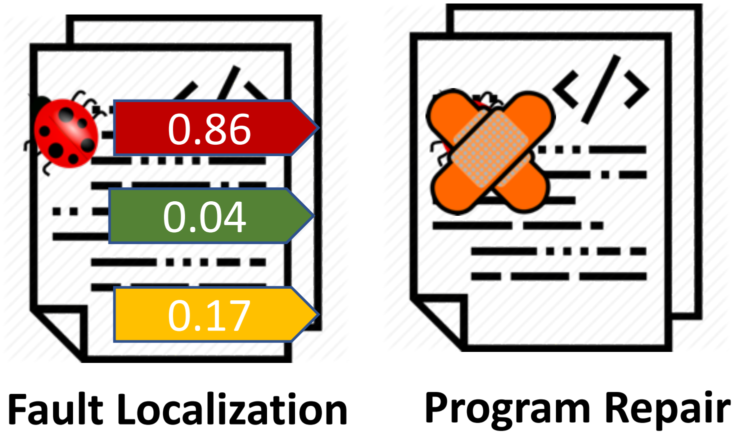


Input: #$H&&*H

Program

Bug detected!

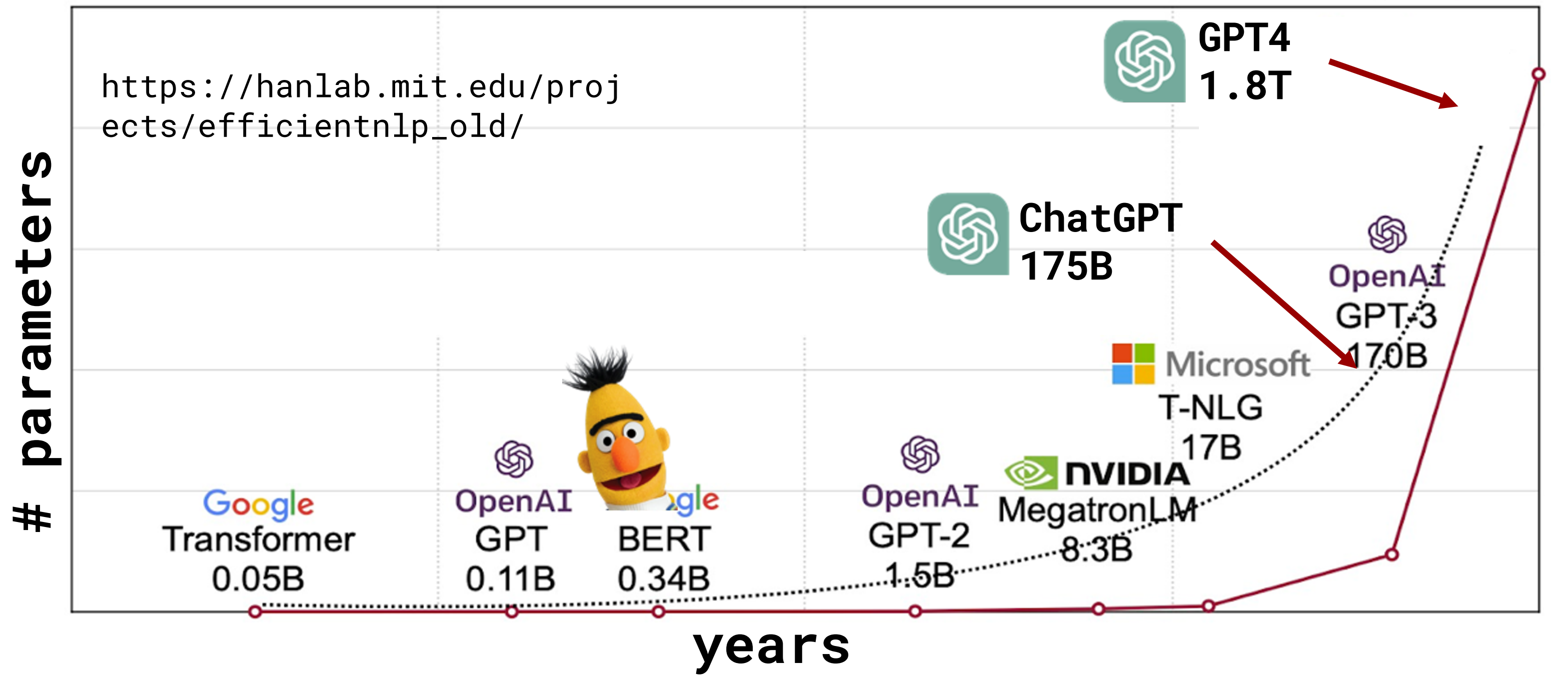
- Fault localization/diagnosis
  - Which lines are responsible for the execution failure?
- Automated program repair
  - How to automatically patch the system?



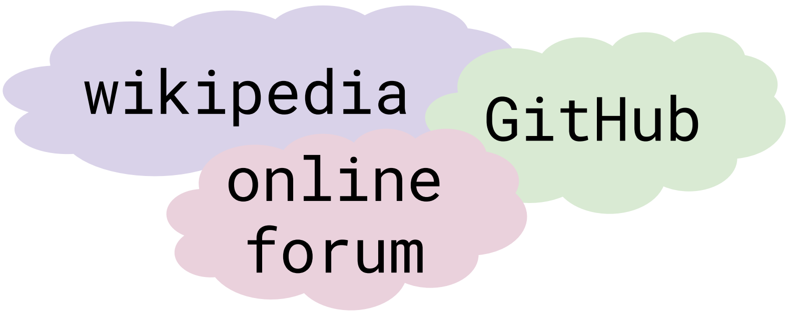
Fault Localization

Program Repair

Large Language Models (LLMs)



https://hanlab.mit.edu/projects/efficientnlp_old/



trillions of text tokens

- How do LLMs perform for QA of real-world software systems?
- How does software QA work for recent LLMs?

Course topics (tentative)

Background and Basics

Intro

Program Analysis

Software Testing

Automated Debugging

LLMs

Code LLMs

Encoder-only Models

Encoder-Decoder Models

Decoder-only Models

Trained on Foundation Models

Instruction Tuning

Others

Code LLMs for Software QA

Fuzz Testing

Unit Testing

Program Repair

Automated Debugging

Program Analysis

Software Verification

Software QA for Code LLMs

Benchmarking

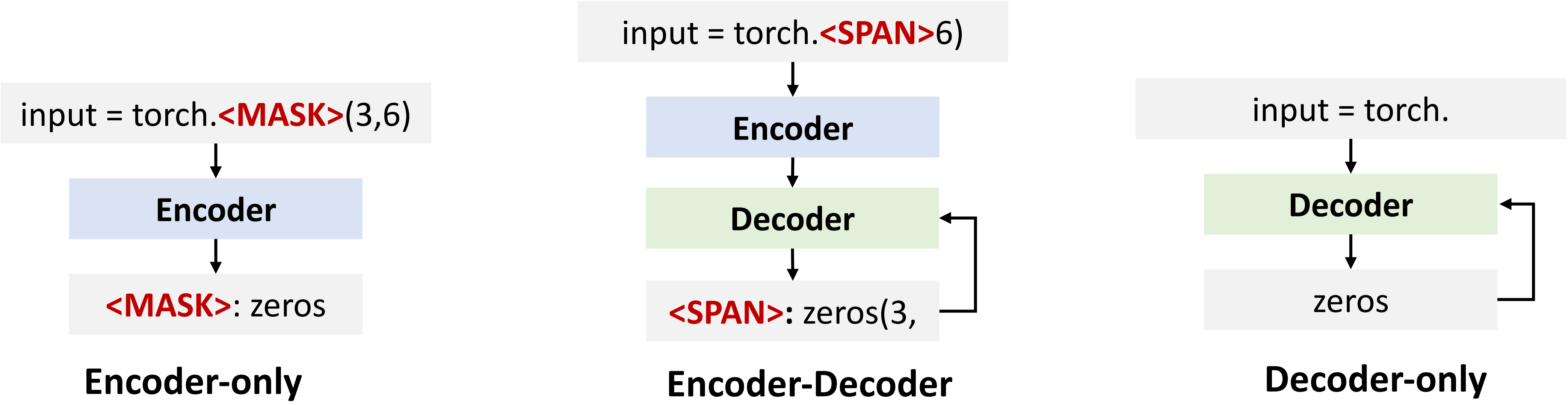
Code Correctness

Code Security

Model Security

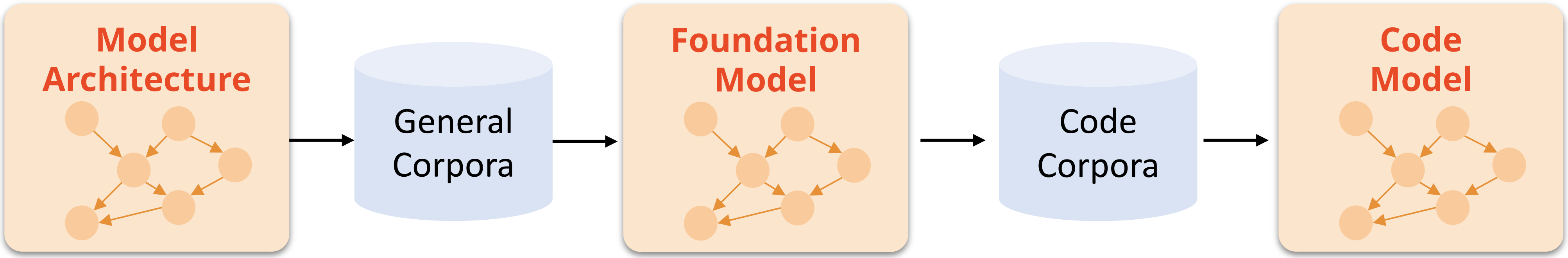
System Reliability

Architectures



- Encoder-only: CodeBERT, GraphCodeBERT, …
- Encoder-Decoder: CodeT5, CodeT5+, AlphaCode, …
- Decoder-only: CodeGen, Phi-1, StarCoder, …

Trained on foundation models



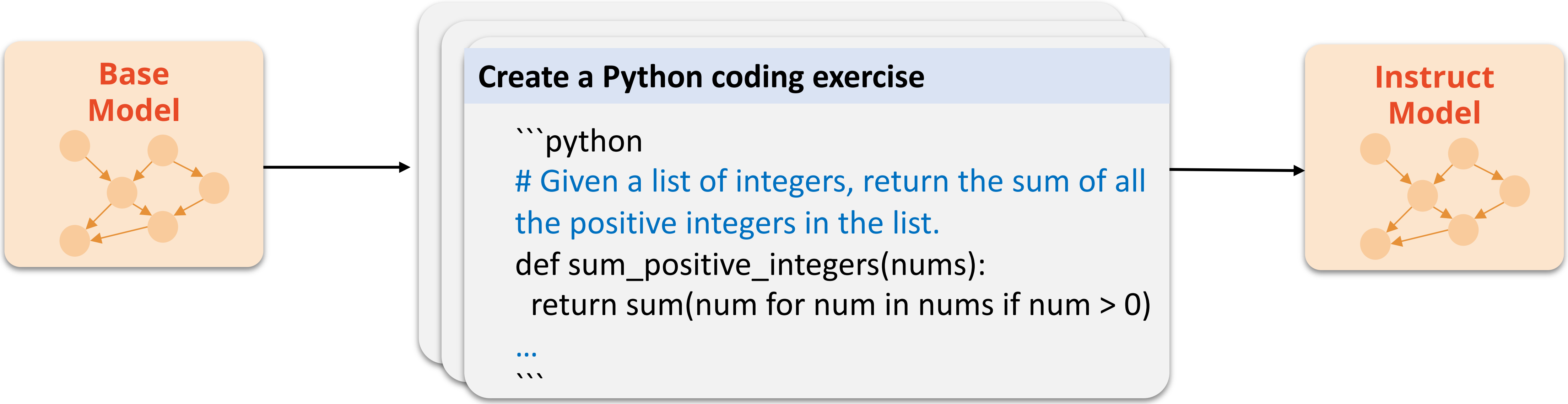
- Faster training
- Better reasoning
- Better NL instruction following
- …

PaLM 2

Code Llama

Gemini

Instruction tuning



Base Model

Create a Python coding exercise

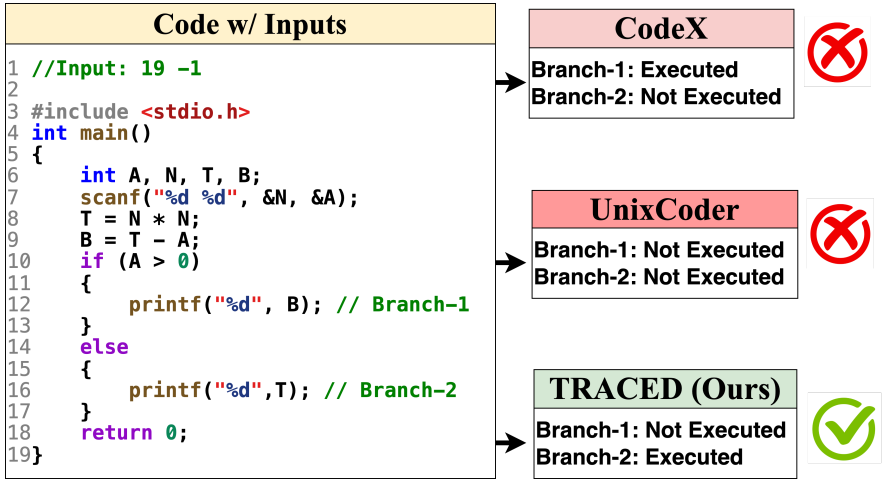
```python
# Given a list of integers, return the sum of all the positive integers in the list.
def sum_positive_integers(nums):
  return sum(num for num in nums if num > 0)
...
```

Instruct Model

- Ideally, the instruction data should be:
  - Diverse
  - Realistic
  - Controllable
  - …

- How to generate code instruction data automatically?
  - Self-Instruct
  - Evol-Instruct
  - OSS-Instruct
  - …

Others



Code w/ Inputs

```
//Input: 19 -1

#include <stdio.h>
int main()
{
    int A, N, T, B;
    scanf("%d %d", &N, &A);
    T = N * N;
    B = T - A;
    if (A > 0)
    {
        printf("%d", B); // Branch-1
    }
    else
    {
        printf("%d",T); // Branch-2
    }
    return 0;
}
```

CodeX ✗

Branch-1: Executed
Branch-2: Not Executed

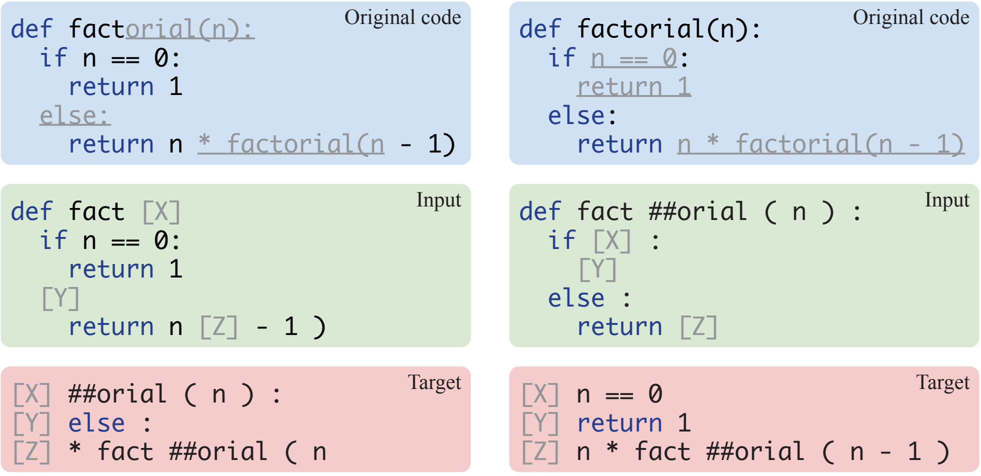
UnixCoder ✗

Branch-1: Not Executed
Branch-2: Not Executed

TRACED (Ours) ✓

Branch-1: Not Executed
Branch-2: Executed

TRACED (dynamic information)



Original code

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

Input

```
def fact [X]
  if n == 0:
    return 1
  [Y]
    return n [Z] - 1 )
```

Target

```
[X] ##orial ( n ) :
[Y] else :
[Z] * fact ##orial ( n
```

Vanilla T5 Span Corruption

Original code

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

Input

```
def fact ##orial ( n ) :
  if [X] :
    [Y]
  else :
    return [Z]
```

Target

```
[X] n == 0
[Y] return 1
[Z] n * fact ##orial ( n - 1 )
```

AST-Aware Subtree Corruption

AST-T5 (static information)

Course topics (tentative)

Background and Basics

Intro
Program Analysis
Software Testing
Automated Debugging
LLMs

Code LLMs

Encoder-only Models
Encoder-Decoder Models
Decoder-only Models
Trained on Foundation Models
Instruction Tuning
Others

Code LLMs for Software QA

Fuzz Testing
Unit Testing
Program Repair
Automated Debugging
Program Analysis
Software Verification

Software QA for Code LLMs

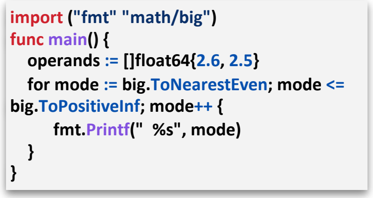
Benchmarking
Code Correctness
Code Security
Model Security
System Reliability

Code LLMs for software QA

Large Language Models

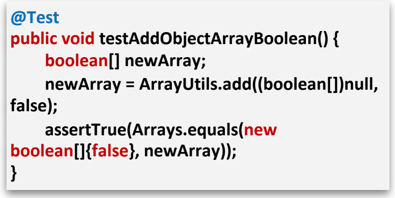
Source Code

Documentation



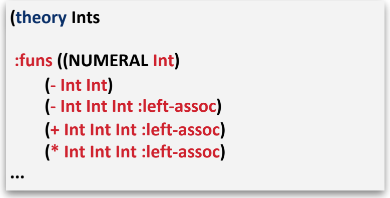
```
import ("fmt" "math/big")
func main() {
   operands := []float64{2.6, 2.5}
   for mode := big.ToNearestEven; mode <=
big.ToPositiveInf; mode++ {
       fmt.Printf("  %s", mode)
   }
}
```

Example Usage



```
@Test
public void testAddObjectArrayBoolean() {
      boolean[] newArray;
      newArray = ArrayUtils.add((boolean[])null,
false);
      assertTrue(Arrays.equals(new
boolean[]{false}, newArray));
}
```

Test Cases



```
(theory Ints

 :funs ((NUMERAL Int)
      (- Int Int)
      (- Int Int Int :left-assoc)
      (+ Int Int Int :left-assoc)
      (* Int Int Int :left-assoc)
...
```

Specifications

- Fuzz testing
- Unit testing
- Program repair
- Automated debugging
- Program analysis
- Software verification
- …

Course topics (tentative)

Background and Basics

Intro
Program Analysis
Software Testing
Automated Debugging
LLMs

Code LLMs

Encoder-only Models
Encoder-Decoder Models
Decoder-only Models
Trained on Foundation Models
Instruction Tuning
Others

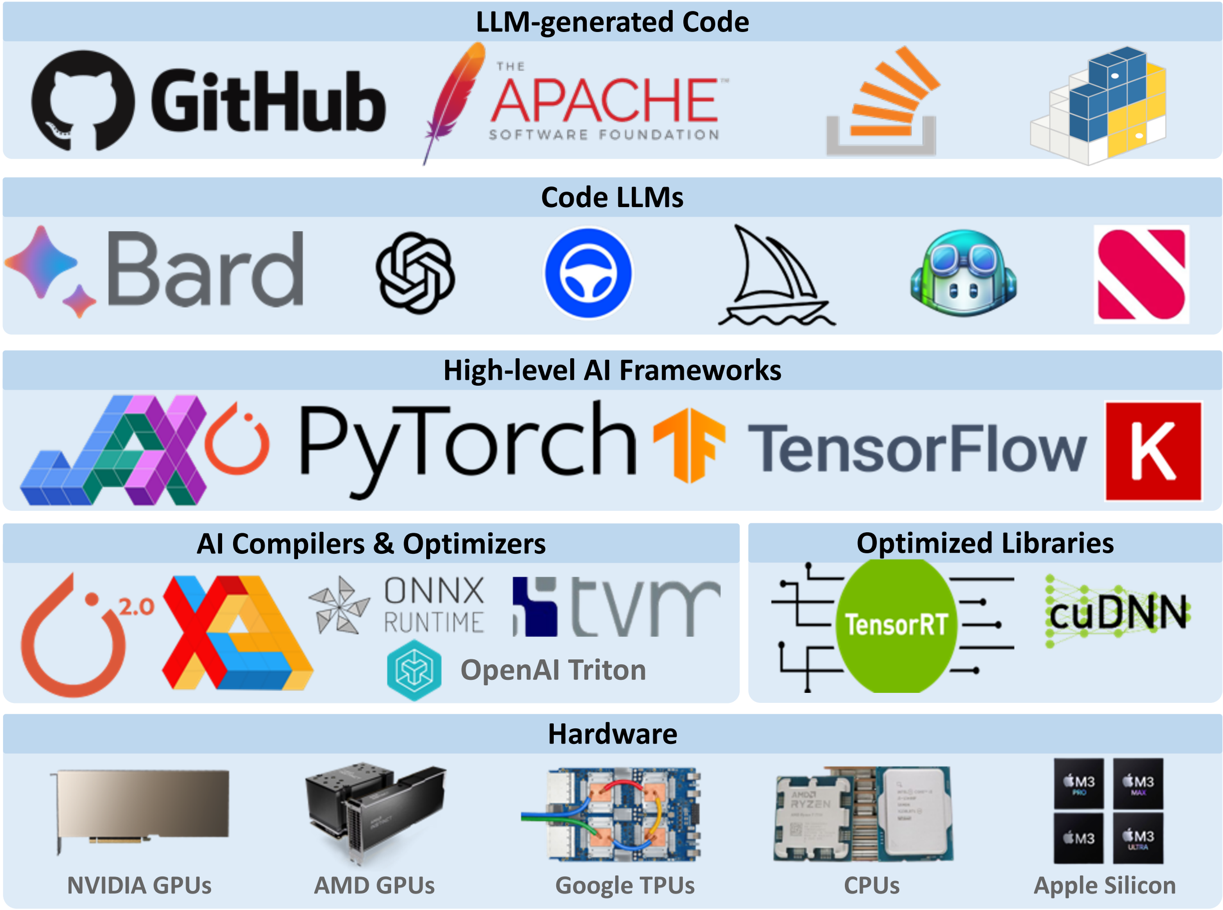
Code LLMs for Software QA

Fuzz Testing
Unit Testing
Program Repair
Automated Debugging
Program Analysis
Software Verification

Software QA for Code LLMs

Benchmarking
Code Correctness
Code Security
Model Security
System Reliability

Software QA for code LLMs



- Benchmarking
- Code correctness
- Code security
- Model security
- System reliability
- …

Thanks and stay safe!