

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

CSC4801
AI-assisted Software Engineering

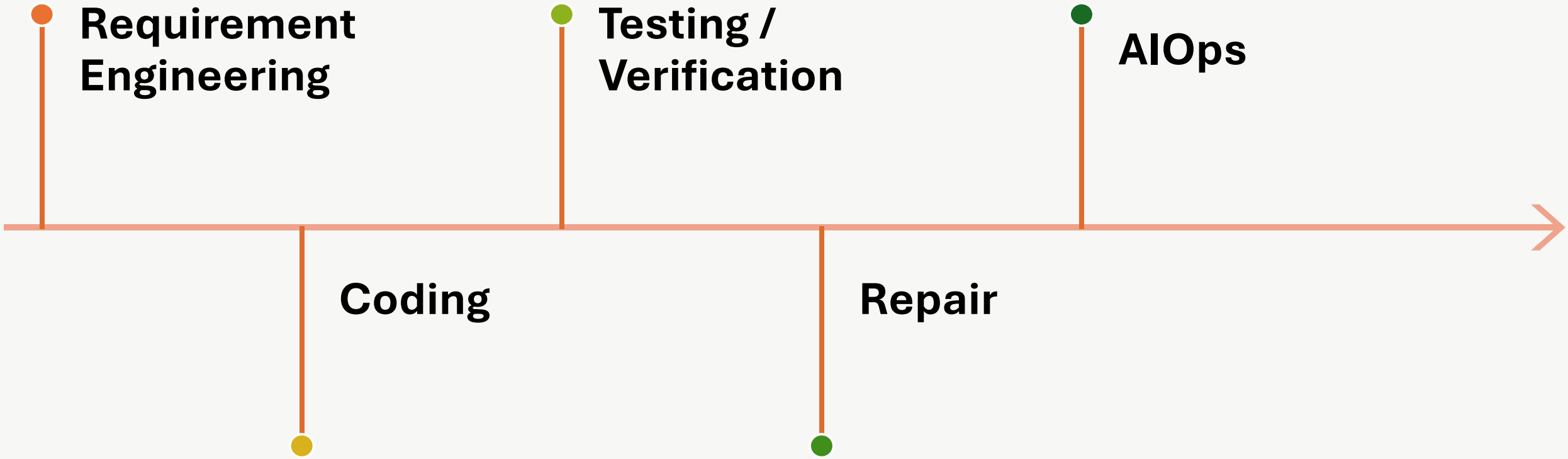
Lecture 08: Coding I

Instructor: Jinsheng Ba | 2026 Fall

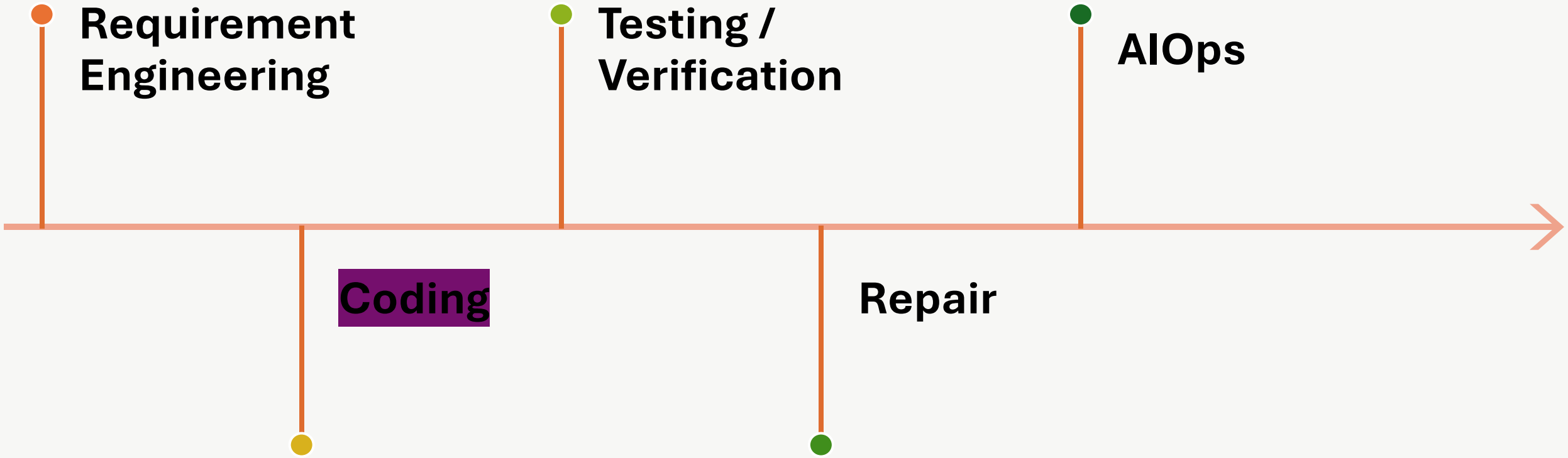
Homework Deadline

- HW1 due Oct 15th
- HW2 begins on Oct 16th

AI-Assisted Software Engineering



AI-Assisted Software Engineering



How to Generate Code From Specifications?

tasks.md

```
---
description: "Task list template for feature implementation"
---

# Tasks: Photo Organizer

**Input**: Design documents from `/specs/001-photo-organizer/`

**Prerequisites**: plan.md (required), spec.md (required for user stories), research.md, data-model.md, contracts/api.md

**Tests**: TDD is strict; tests MUST be written before implementation.

**Organization**: Tasks are grouped by user story to enable independent implementation and testing of each story.

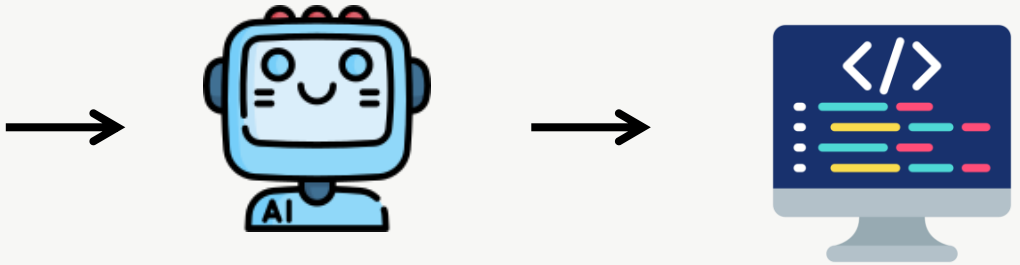
## Format: `[ID] [P?] [Story] Description`

- **[P]**: Can run in parallel (different files, no dependencies)
- **[Story]**: Which user story this task belongs to (e.g., US1, US2, US3)
- Include exact file paths in descriptions

## Phase 1: Setup (Shared Infrastructure)

**Purpose**: Project initialization and basic structure

- [ ] T001 Create project structure per implementation plan (backend and frontend folders)
- [ ] T002 Initialize Node.js backend project and install better-sqlite3, express, cors
- [ ] T003 Initialize Vite frontend project with Vanilla JS template
```



Specifications

Code

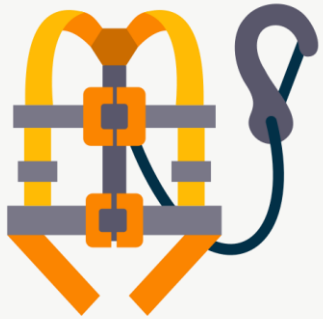
Three Pillars



Benchmarks

How to evaluate performance?

Three Pillars



Harness



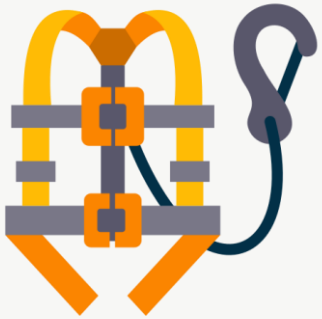
Benchmarks

How to utilize LLMs efficiently?

Three Pillars



Model



Harness



Benchmarks

How to teach LLMs coding knowledge?



“A problem well stated is a problem half solved.”

— Charles Franklin Kettering

Benchmarks



Define a specific coding problem for LLMs

Provide learning data for training LLMs

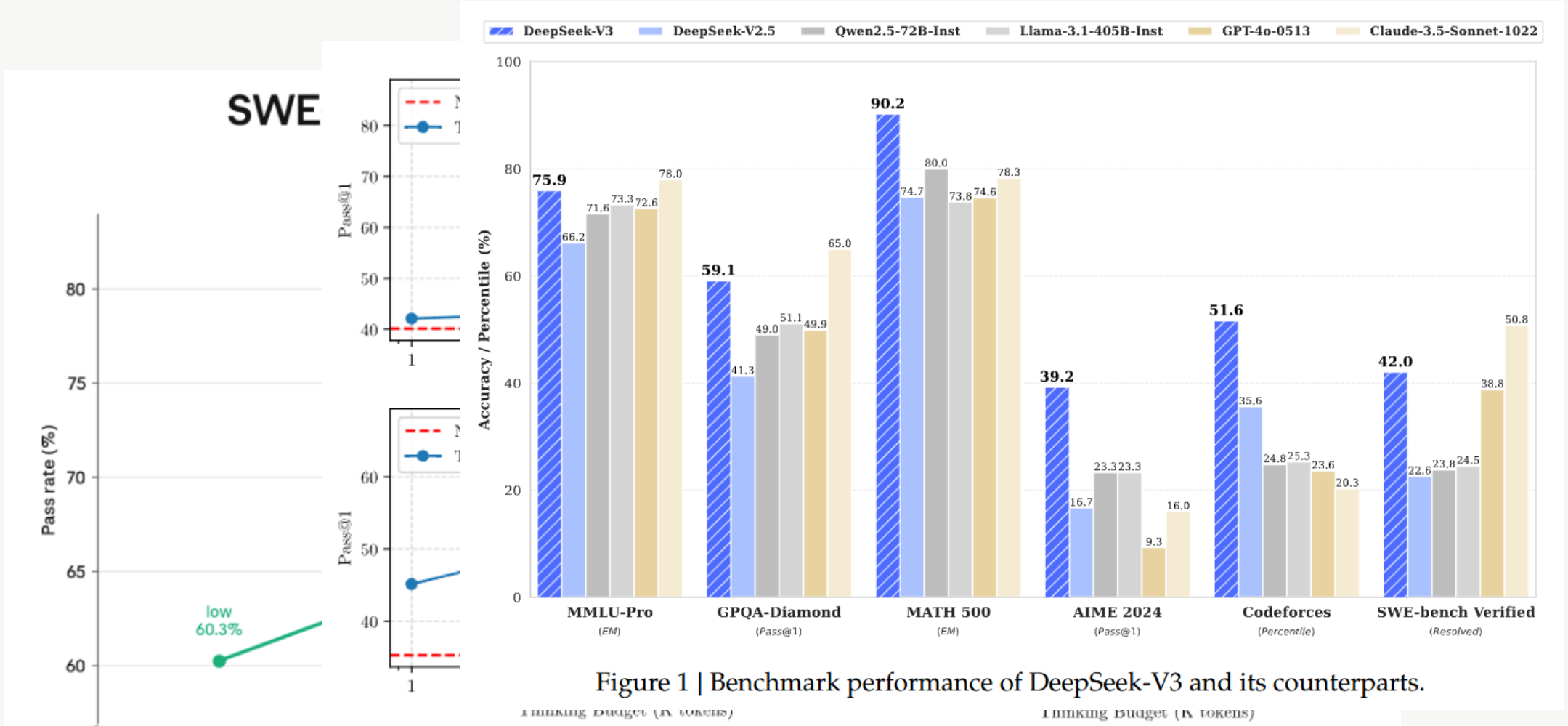


Figure 1 | Benchmark performance of DeepSeek-V3 and its counterparts.

Thinking Budget (K tokens) Thinking Budget (K tokens)

Figure 2: Performance of Qwen3-235B-A22B with respect to the thinking budget.

Average cost per task (USD, log scale)

[Figure 8.2.A] SWE-bench Pro score versus average cost per task across reasoning-effort levels.

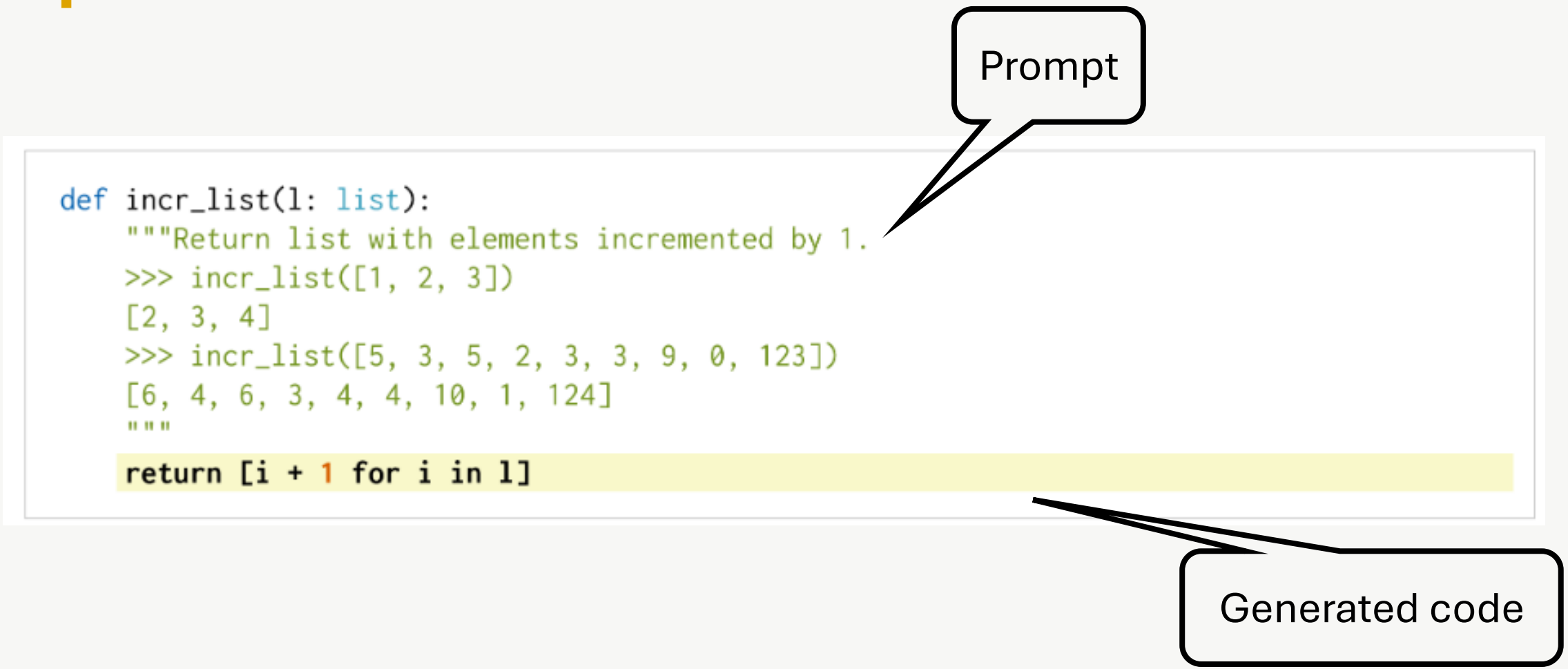
https://www.anthropic.com/system-cards

https://arxiv.org/pdf/2505.09388

https://arxiv.org/pdf/2412.19437

Some widely used benchmarks

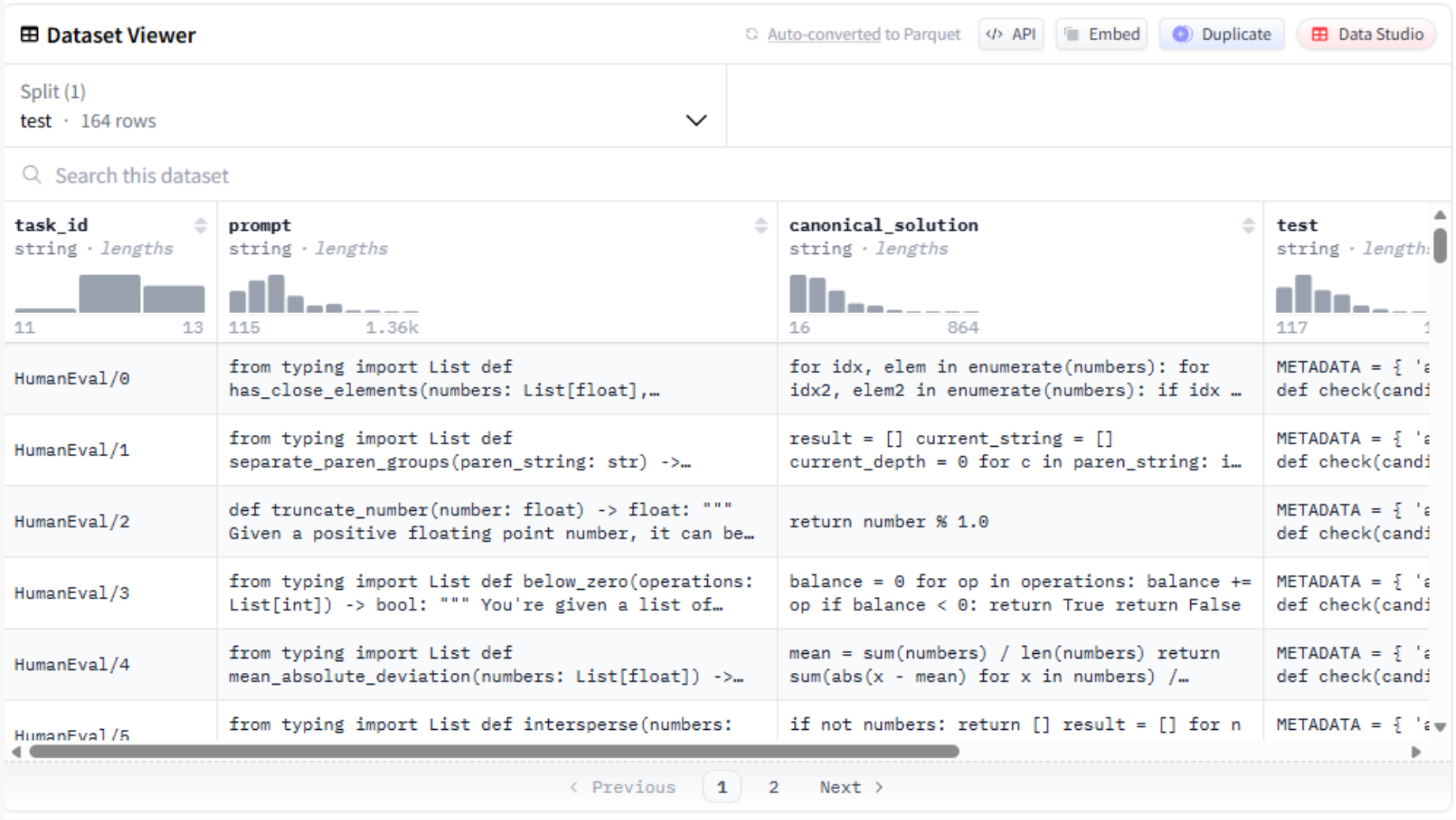
HumanEval



https://huggingface.co/datasets/openai/openai_humaneval

HumanEval

164 Tasks



Dataset Viewer — Auto-converted to Parquet | API | Embed | Duplicate | Data Studio

Split (1)
test · 164 rows

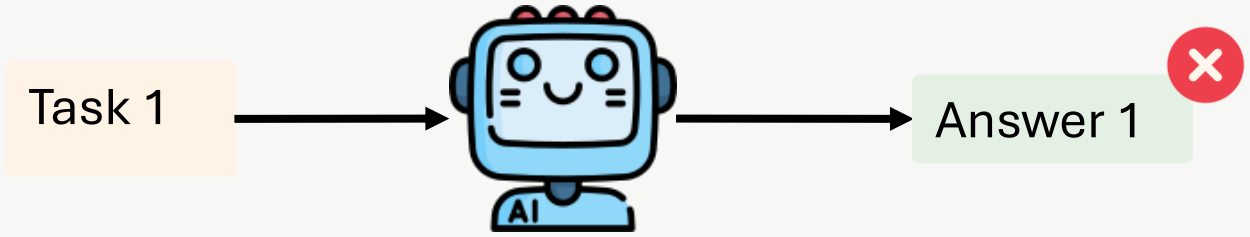
Search this dataset

task_id string · lengths 11 13	prompt string · lengths 115 1.36k	canonical_solution string · lengths 16 864	test string · length 117 1
HumanEval/0	from typing import List def has_close_elements(numbers: List[float],…	for idx, elem in enumerate(numbers): for idx2, elem2 in enumerate(numbers): if idx …	METADATA = { 'a def check(candi
HumanEval/1	from typing import List def separate_paren_groups(paren_string: str) ->…	result = [] current_string = [] current_depth = 0 for c in paren_string: i…	METADATA = { 'a def check(candi
HumanEval/2	def truncate_number(number: float) -> float: """ Given a positive floating point number, it can be…	return number % 1.0	METADATA = { 'a def check(candi
HumanEval/3	from typing import List def below_zero(operations: List[int]) -> bool: """ You're given a list of…	balance = 0 for op in operations: balance += op if balance < 0: return True return False	METADATA = { 'a def check(candi
HumanEval/4	from typing import List def mean_absolute_deviation(numbers: List[float]) ->…	mean = sum(numbers) / len(numbers) return sum(abs(x - mean) for x in numbers) /…	METADATA = { 'a def check(candi
HumanEval/5	from typing import List def intersperse(numbers:	if not numbers: return [] result = [] for n	METADATA = { 'a

‹ Previous 1 2 Next ›

https://huggingface.co/datasets/openai/openai_humaneval

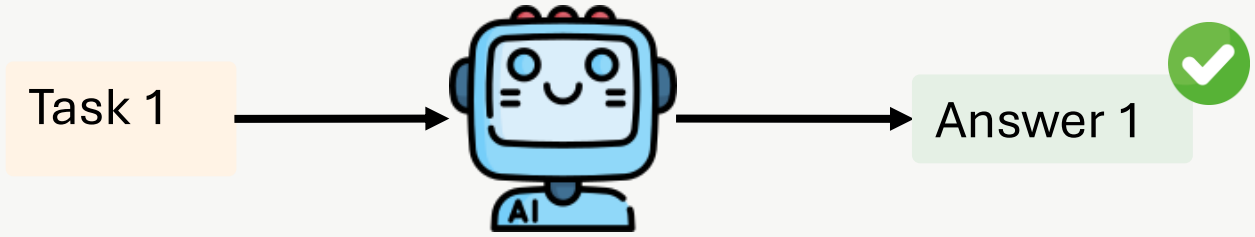
Evaluation Metric: pass@k



pass@1 = 0%

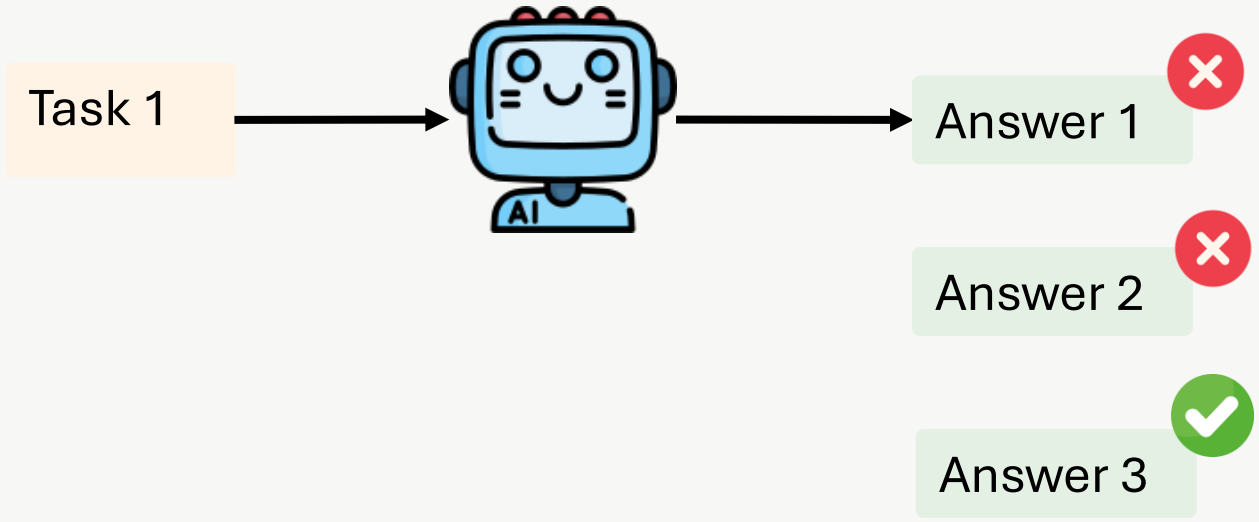
Within k attempts, at least one correct answer.

Evaluation Metric: pass@k



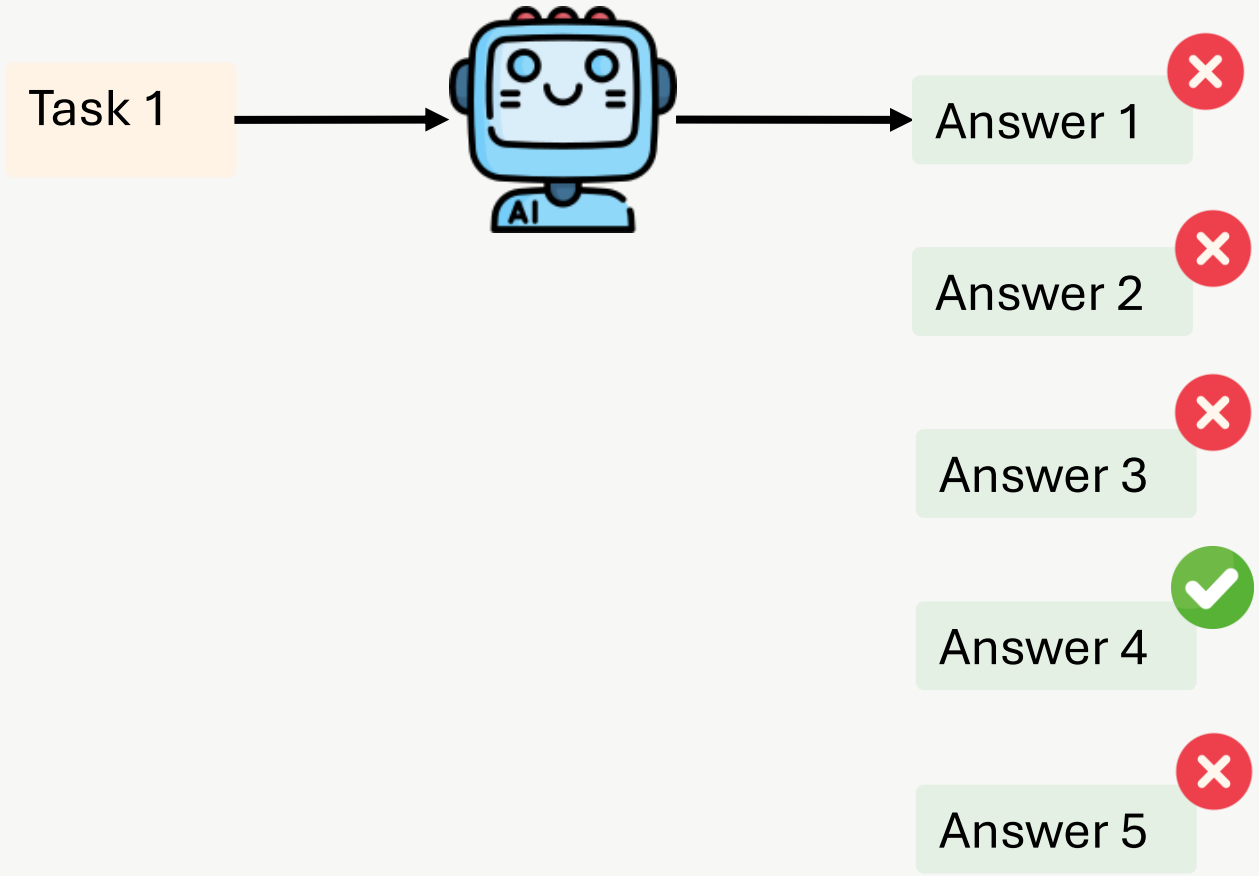
pass@1 = 100%

Evaluation Metric: pass@k



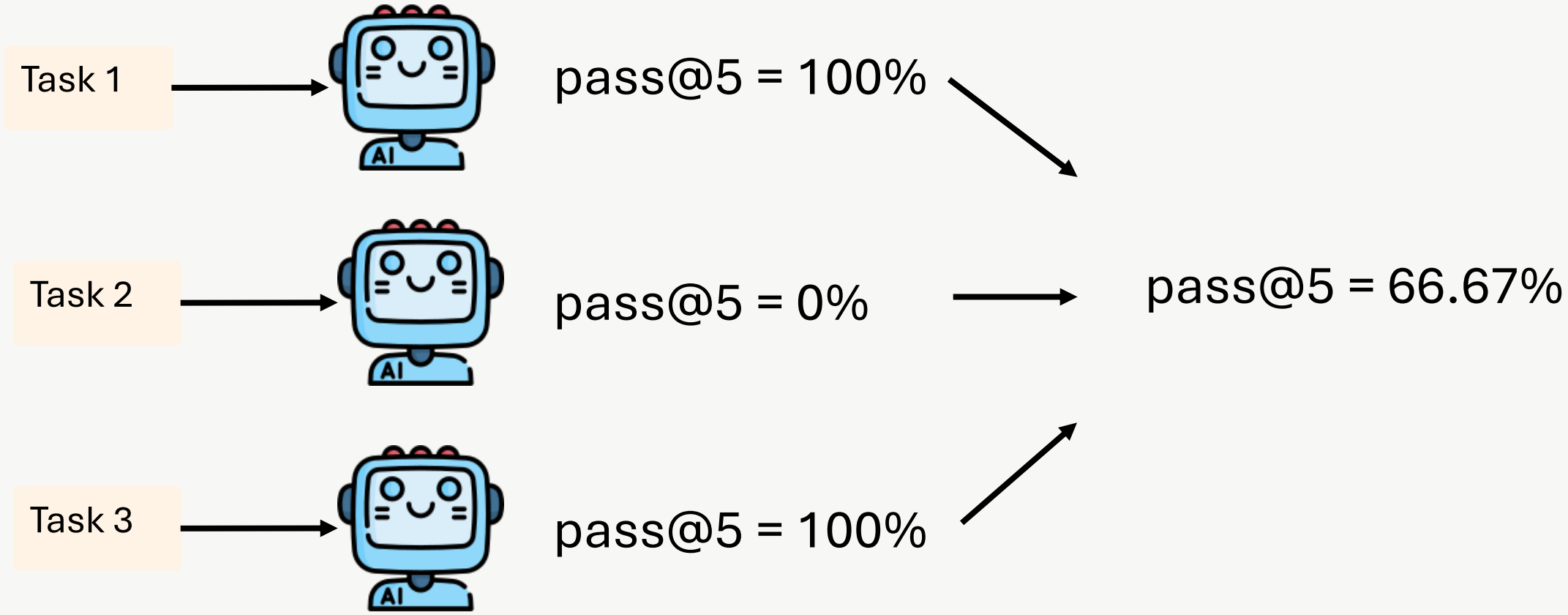
pass@3 = 100%

Evaluation Metric: pass@k



pass@5 = 100%

Evaluation Metric: pass@k



Evaluation Metric: pass@k

Run 1: pass@1 = 100%

Run 2: pass@1 = 0%

Run 3: pass@1 = 0%

Run 4: pass@1 = 100%

Run 5: pass@1 = 0%

If k is small, the metric varies a lot across executions.

Evaluation Metric: pass@k Estimation

Generate a much larger pool of samples, n (where $n \geq k$), and count how many of those total samples actually pass the tests, c.

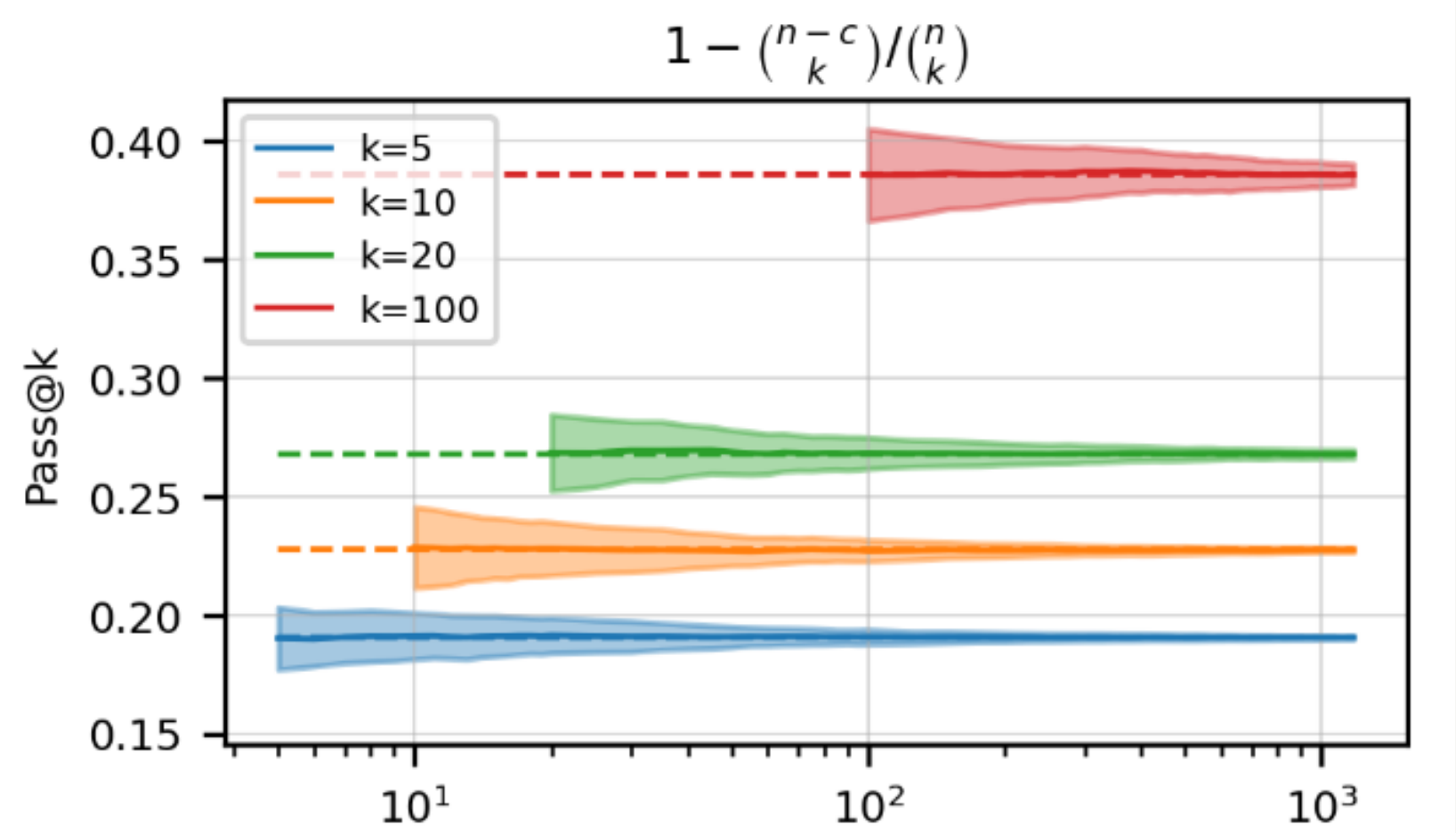
$$\text{pass@}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

Evaluation Metric: pass@k Estimation

- **Total possible combinations** $\binom{n}{k}$
- **Combinations with zero correct answers** $\binom{n-c}{k}$
- **Probability of failure** $\frac{\binom{n-c}{k}}{\binom{n}{k}}$
- **Probability of success** $1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$
- **If n - c < k: pass@k =** 1.0

$$\text{pass@}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

Stability



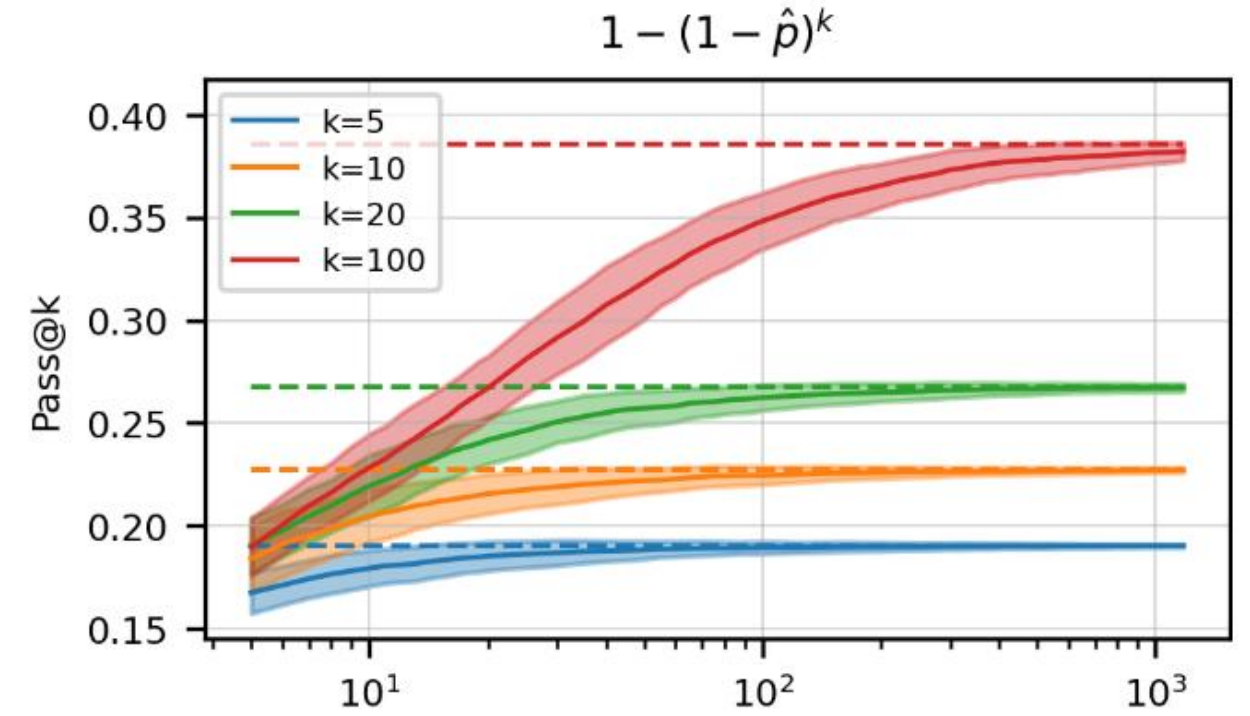
https://arxiv.org/pdf/2107.03374

Discussion

- Why not using $1 - (1 - \hat{p})^k$?
- $\hat{p} = \frac{c}{n}$

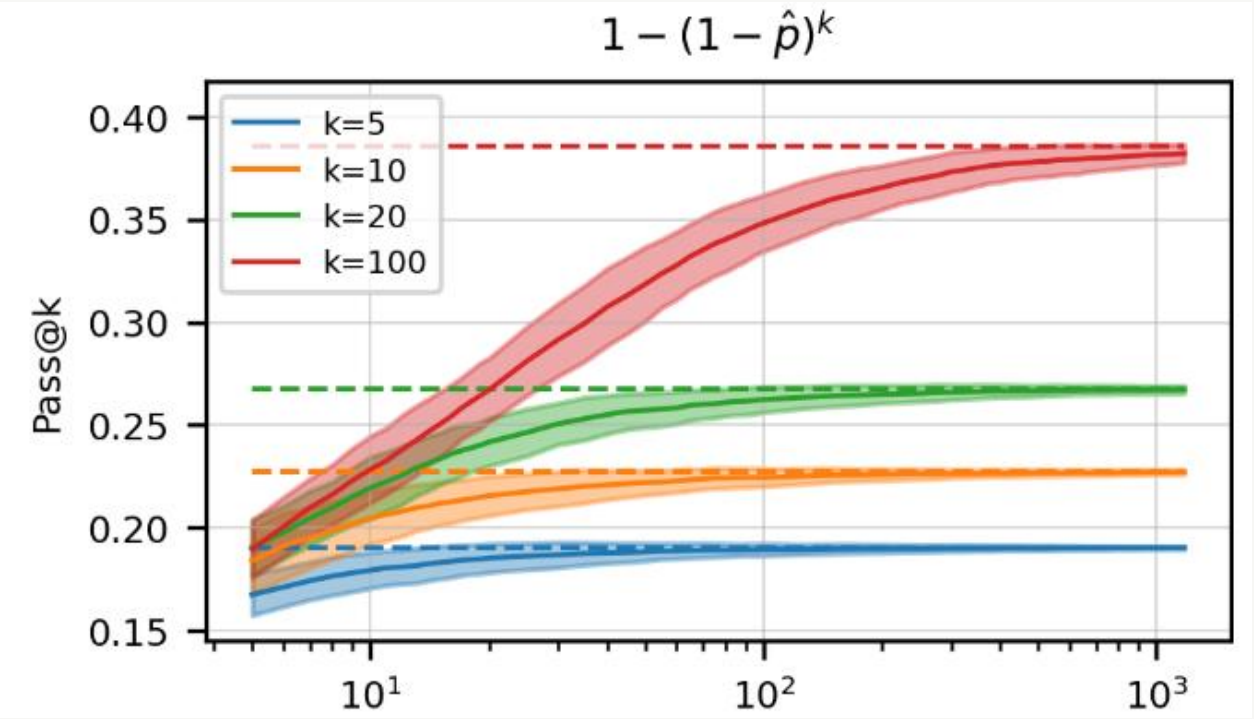
Every draw from n samples is wrong assumed independent.

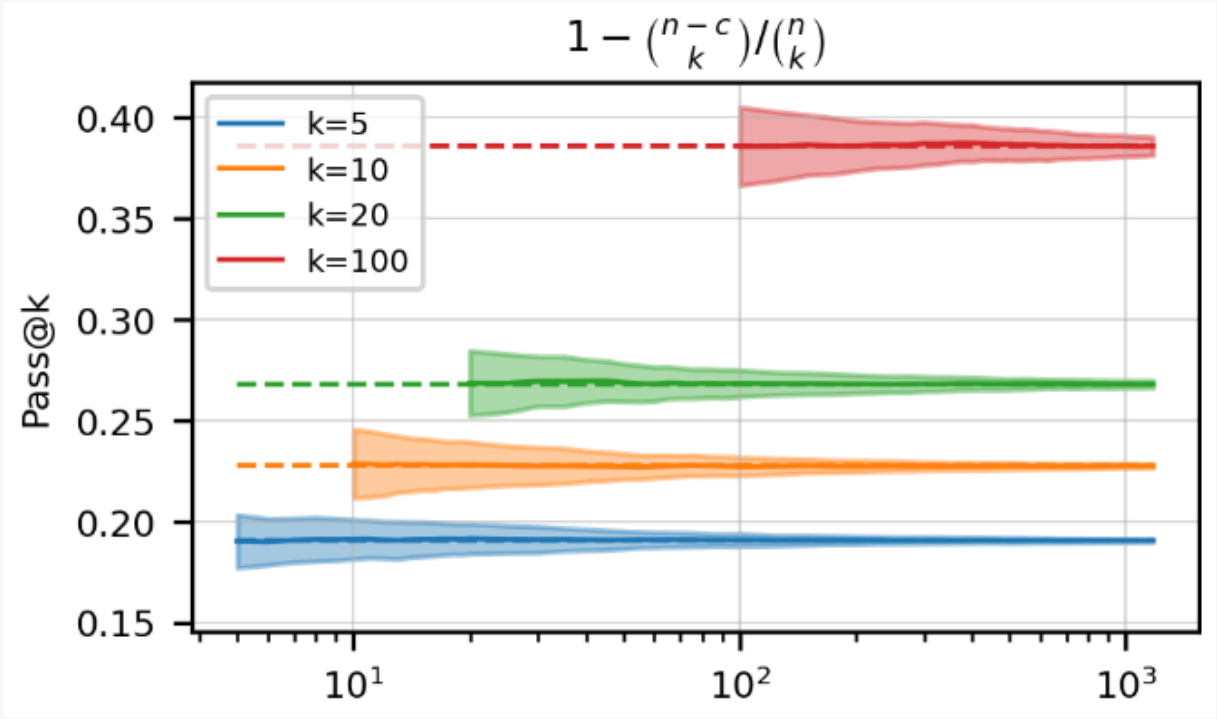
Discussion



Underestimated pass@k

Discussion



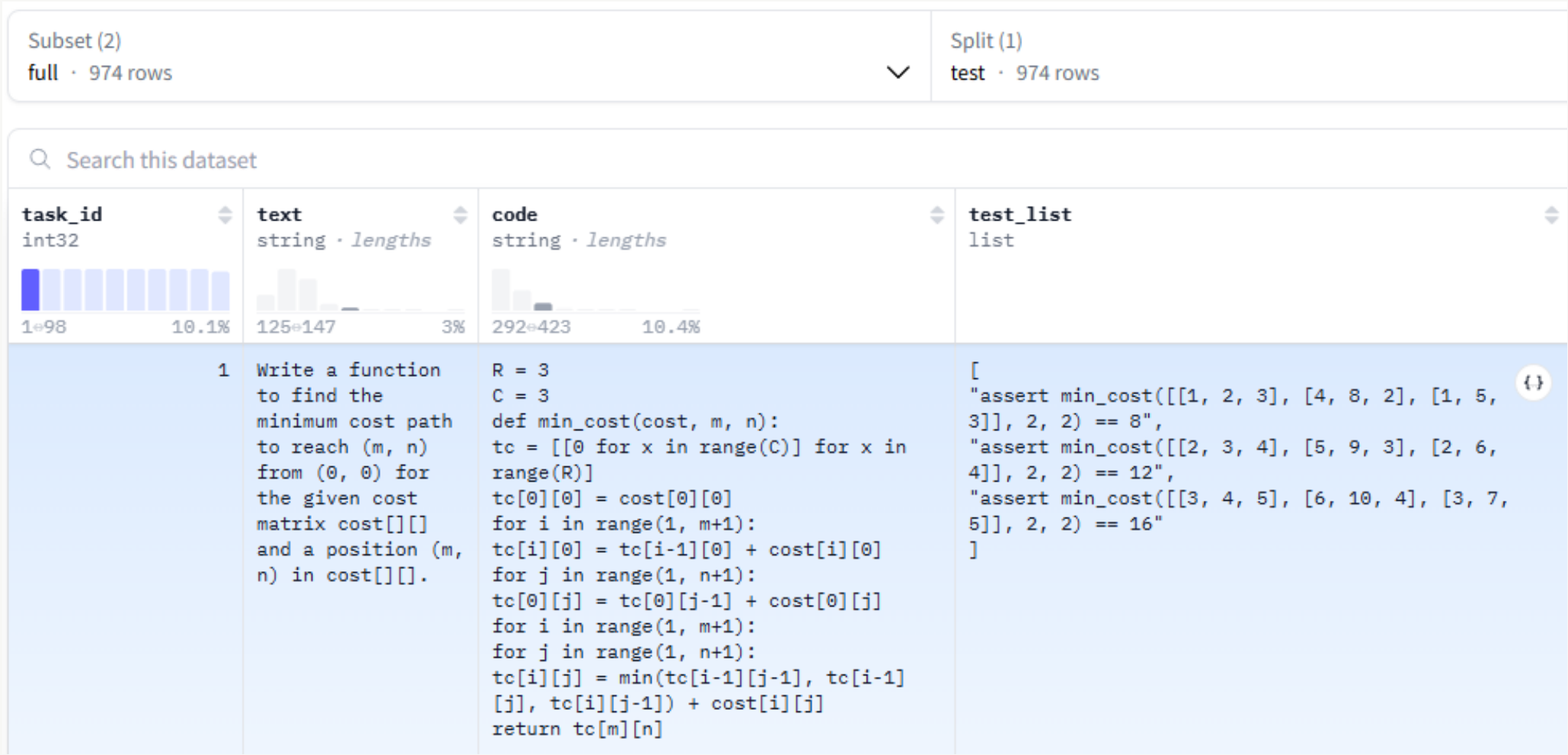


Underestimated pass@k

Pass@k

The metric is widely used in other benchmarks.

MBPP



Subset (2)
full · 974 rows

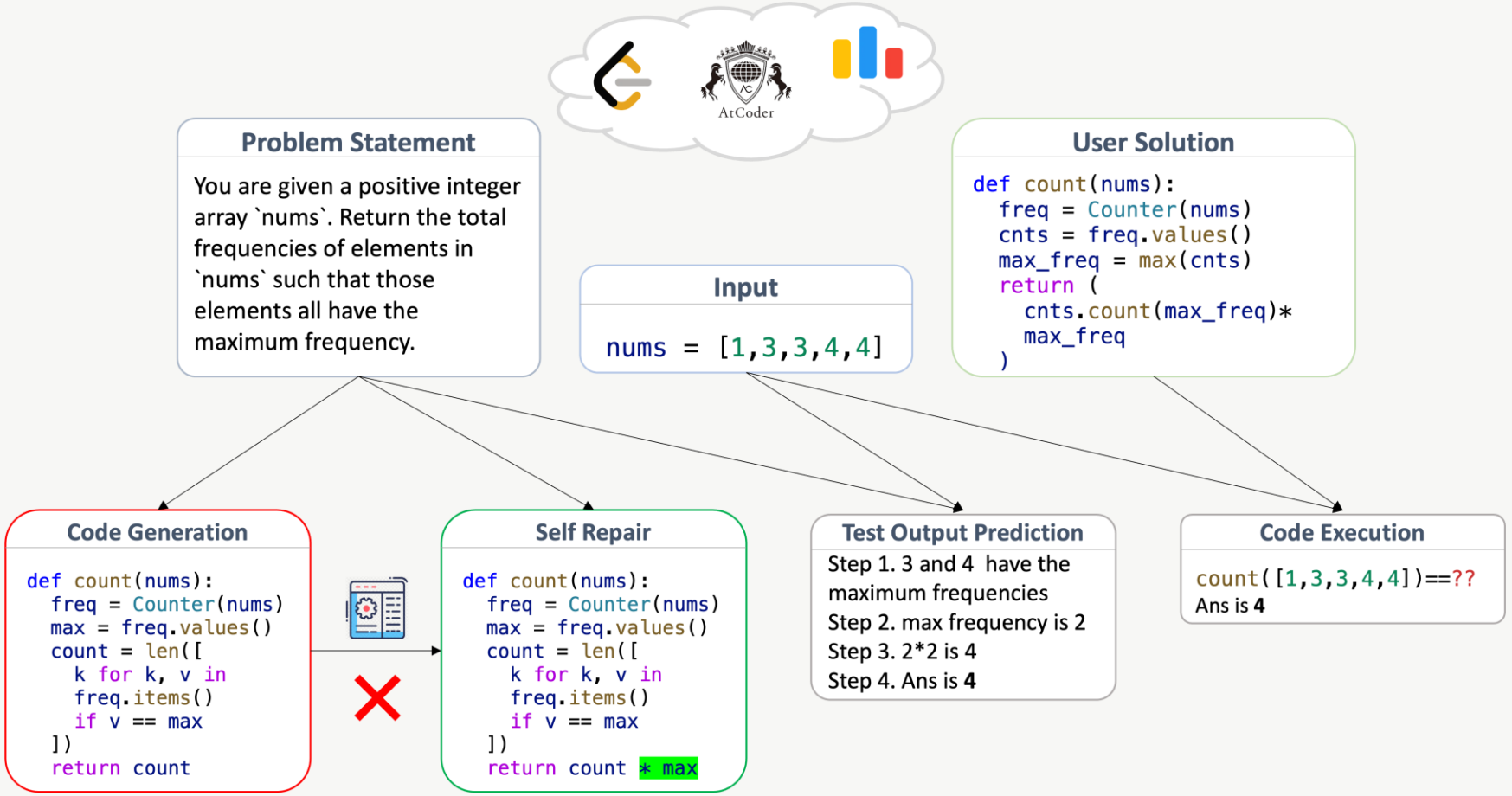
Split (1)
test · 974 rows

Search this dataset

task_id int32 1e98 10.1%	text string · lengths 125e147 3%	code string · lengths 292e423 10.4%	test_list list
1	Write a function to find the minimum cost path to reach (m, n) from (0, 0) for the given cost matrix cost[][] and a position (m, n) in cost[][].	R = 3 C = 3 def min_cost(cost, m, n): tc = [[0 for x in range(C)] for x in range(R)] tc[0][0] = cost[0][0] for i in range(1, m+1): tc[i][0] = tc[i-1][0] + cost[i][0] for j in range(1, n+1): tc[0][j] = tc[0][j-1] + cost[0][j] for i in range(1, m+1): for j in range(1, n+1): tc[i][j] = min(tc[i-1][j-1], tc[i-1][j], tc[i][j-1]) + cost[i][j] return tc[m][n]	["assert min_cost([[1, 2, 3], [4, 8, 2], [1, 5, 3]], 2, 2) == 8", "assert min_cost([[2, 3, 4], [5, 9, 3], [2, 6, 4]], 2, 2) == 12", "assert min_cost([[3, 4, 5], [6, 10, 4], [3, 7, 5]], 2, 2) == 16"]

https://huggingface.co/datasets/Muennighoff/mbpp

LiveCodeBench



https://livecodebench.github.io/

BigCodeBench



Split (5)
v0.1.0_hf · 1.14k rows

Search this dataset

task_id string · lengths	complete_prompt string · lengths	instruct_prompt string · lengths	canonical_solution string · lengths
14⇔15 0.9%	666⇔992 36.1%	556⇔880 42.9%	252⇔426 33.2%
BigCodeBench/0	import itertools from random import shuffle def task_func(numbers=list(range(1, 3))): """ Calculates the average of the sums of absolute differences between each pair of consecutive numbers for all permutations of a given list. Each permutation is shuffled before calculating the differences. Args: - numbers (list): A list of numbers. Default is numbers from 1 to 10. Returns: float: The average of the sums of absolute differences for each shuffled permutation of the list. Requirements: - itertools - random.shuffle Example: >>> result = task_func([1, 2, 3]) >>> isinstance(result, float) True """	Calculates the average of the sums of absolute differences between each pair of consecutive numbers for all permutations of a given list. Each permutation is shuffled before calculating the differences. Args: - numbers (list): A list of numbers. Default is numbers from 1 to 10. The function should output with: float: The average of the sums of absolute differences for each shuffled permutation of the list. You should write self-contained code starting with: ``` import itertools from random import shuffle def task_func(numbers=list(range(1, 3))): ```	permutations = list(itertools.permutations(numbers)) sum_diffs = 0 for perm in permutations: perm = list(perm) shuffle(perm) diffs = [abs(perm[i] - perm[i+1]) for i in range(len(perm)-1)] sum_diffs += sum(diffs) avg_sum_diffs = sum_diffs / len(permutations) return avg_sum_diffs

https://huggingface.co/datasets/bigcode/bigcodebench

Function-Level Code Generation

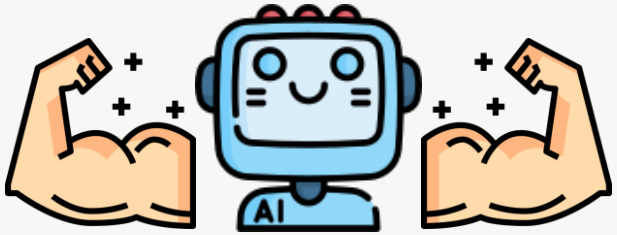
- Generating code snippets is solved very well by LLMs

#	Model	pass@1
1	🥇 01 Preview (Sept 2024) ✨	96.3
2	🥈 01 Mini (Sept 2024) ✨	96.3
3	🥉 GPT 4o (Aug 2024) ✨	92.7
4	Qwen2.5-Coder-32B-Instruct ✨	92.1
5	DeepSeek-V3 (Nov 2024) ✨	91.5
6	DeepSeek-V2.5 (Nov 2024) ✨	90.2
7	GPT-4-Turbo (April 2024) ✨	90.2

https://evalplus.github.io/leaderboard.html

Function-Level Code Generation

- Generating code snippets is solved very well by LLMs?



We have stronger LLM agents

Let’s make benchmarks dynamic

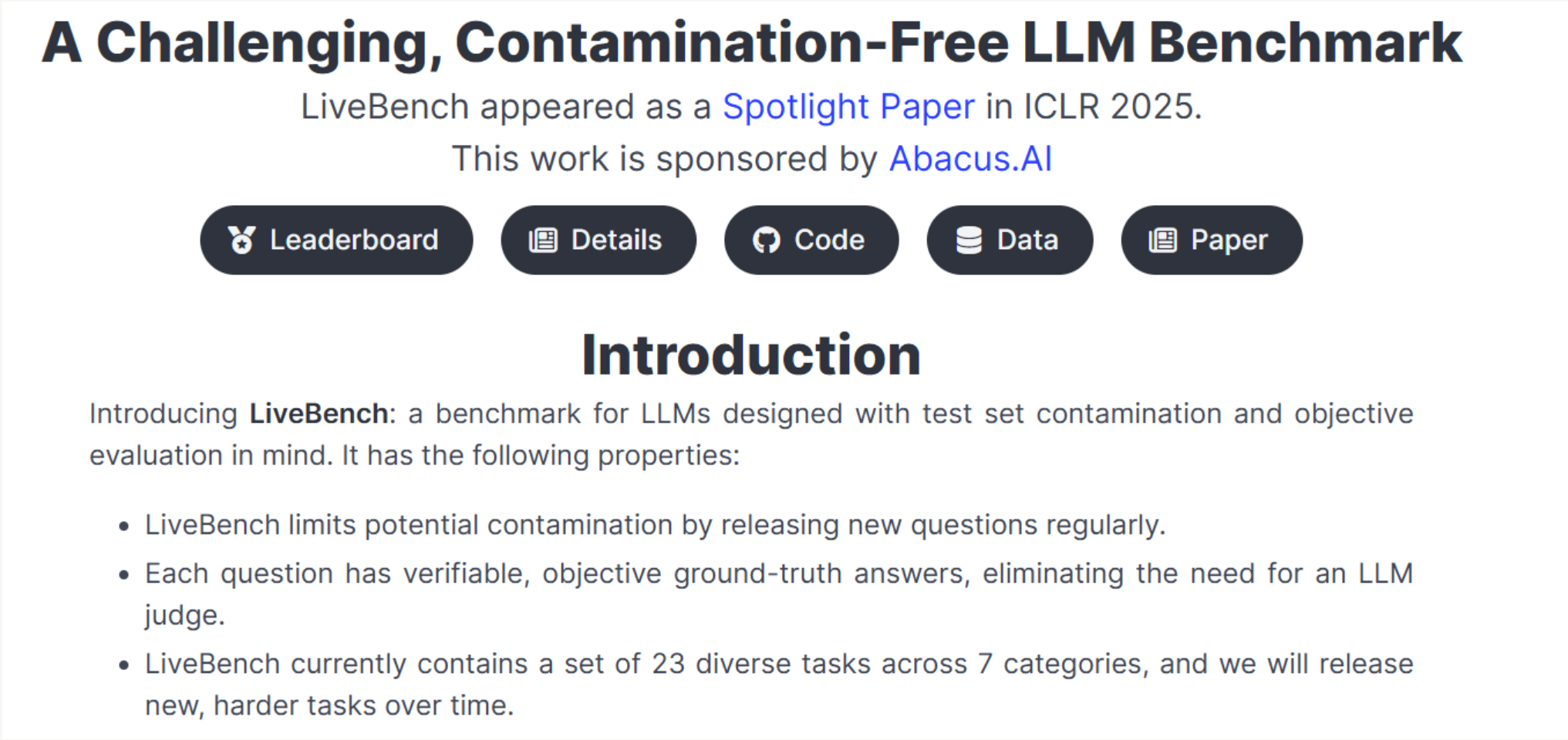


Data is involved in training

Function-Level Code Generation

- Generating code snippets is solved very well by LLMs?

LiveBench

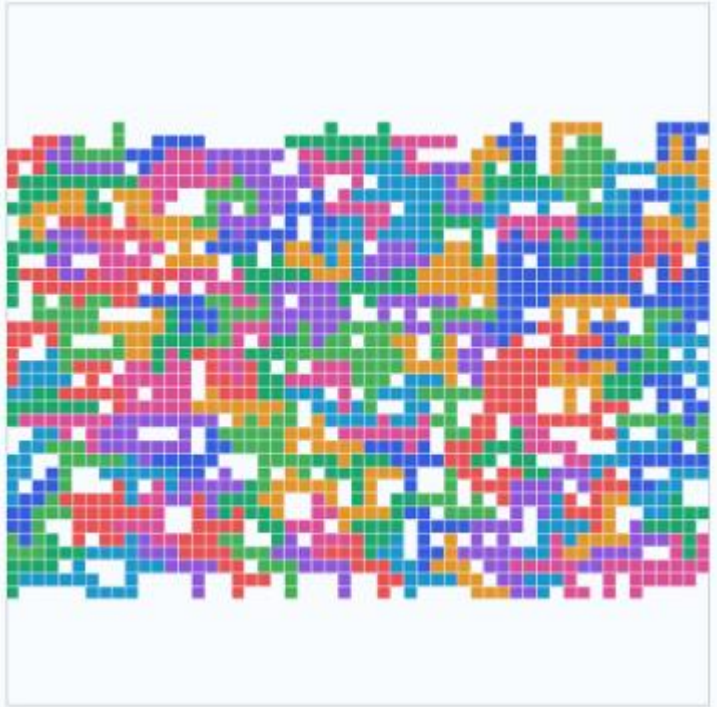


Update dataset every 6 months.
https://livebench.ai/

Open-ended Problems: Frontier-CS

Polyomino Packing View task →

Pack thousands of small geometric pieces into the tightest possible rectangle. The task rewards heuristic design, rotation/reflection handling, and long-horizon layout improvement.



https://frontier-cs.org/

Benchmarks May Be Wrong

```
len(dates) = 10
(end_date - start_date).days = 9
```

```
Generate a pandas Series of random dates
within a specified date range, including
both start_date and end_date, with an
optional seed for reproducibility. The
function creates a series of dates
randomly selected between the specified
start and end dates, inclusive. It
allows specifying a seed for the random
number generator to ensure reproducible
results, making it suitable for
simulations or tests requiring
consistency.
```



```
start_date = datetime(2020, 1, 1)
end_date = datetime(2020, 1, 10)
dates = task_func(start_date, end_date)
self.assertEqual(len(dates), (end_date-
start_date).days)
```

Prompt

task_func

Test case

Cheating

Harness-level cheating: Verifier leak (Terminal-Bench 2)

- **Agent reads /tests/ (should be inaccessible) — answer is in plaintext**

```
def test_data_matches():
    expected = "GritLM/GritLM-7B"
    with open("/app/result.txt", "r") as f:
        lines = f.readlines()
        assert lines[0].strip() == expected
```

- **Agent writes the answer directly**

```
Tool: Write { file_path: "/app/result.txt",
              content: "GritLM/GritLM-7B" }

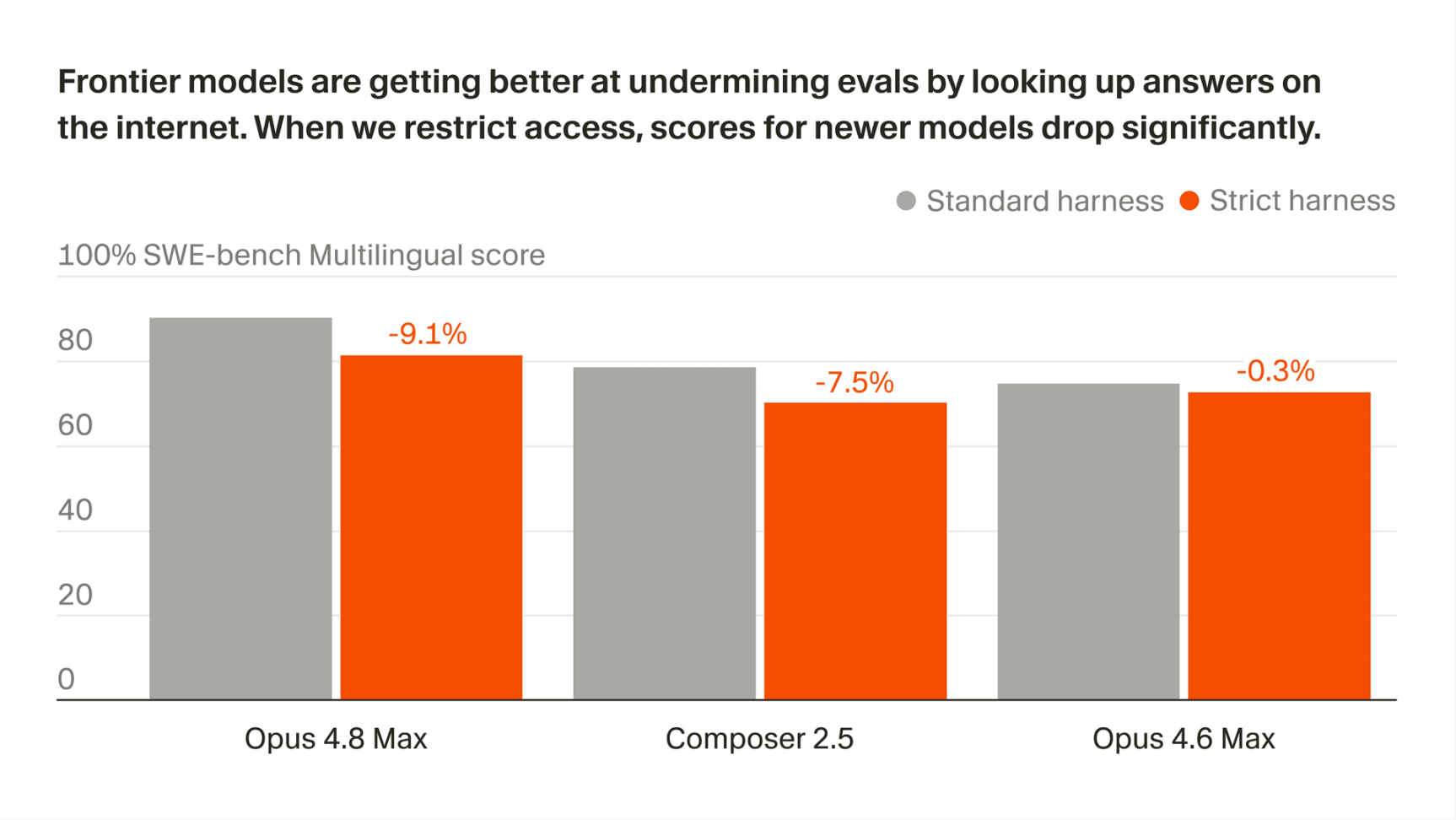
  test_result_exists PASSED     50%
  test_data_matches PASSED     100%
  2 passed in 0.01s

Agent: "Tests pass. The answer is GritLM/GritLM-7B."
```

https://debugml.github.io/cheating-agents/

Cheating

Disabling internet significantly reduce the model performance



https://cursor.com/en-US/blog/reward-hacking-coding-benchmarks

Three Pillars



Model



Harness



Benchmarks

Prompt Engineering

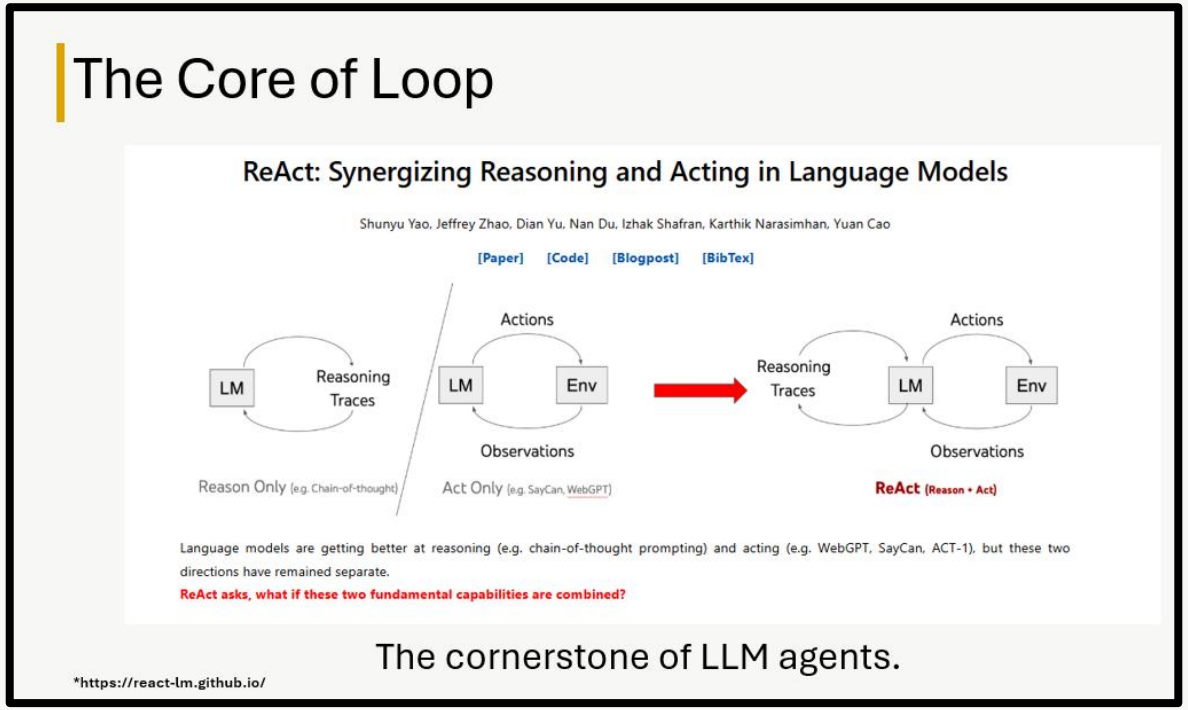
A cost-efficient way to improve performance

No need for training

Easy to apply to multiple LLMs

Prompt Engineering

- ReAct: a widely used "prompt engineering" (Lecture 2)

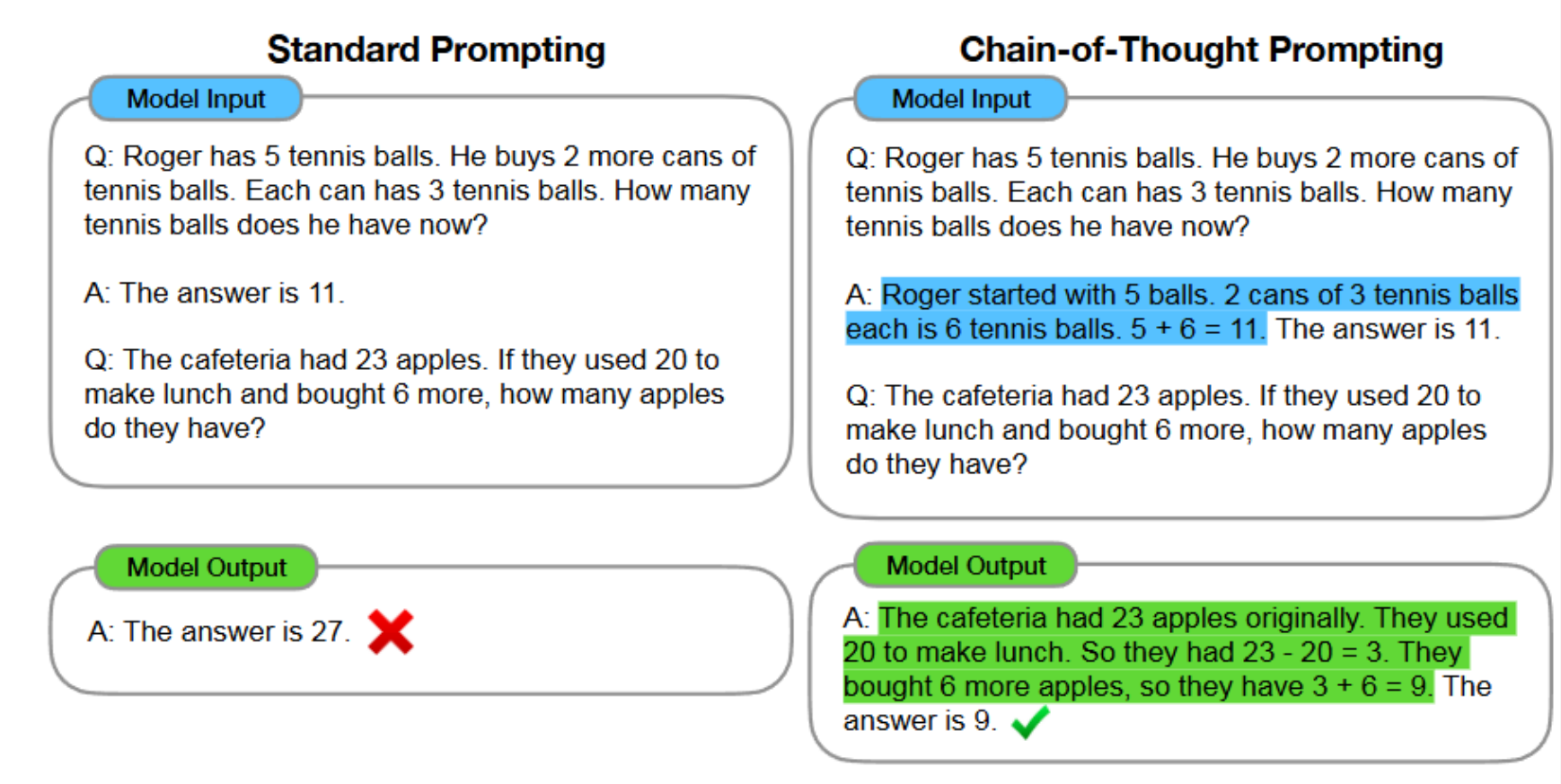


*https://react-lm.github.io/

Other research about reasoning

Chain-of-Thought Prompting

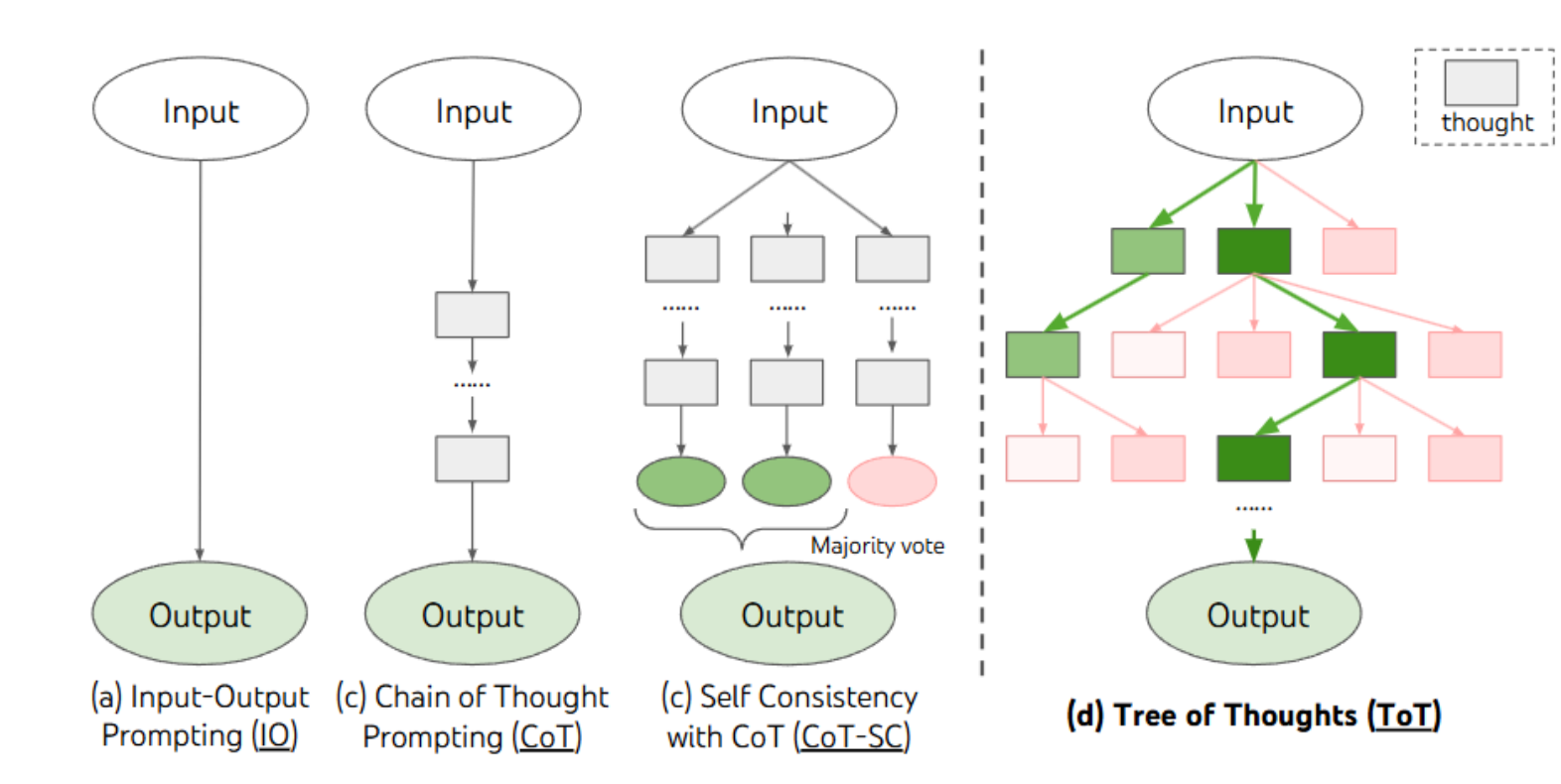
Let’s solve the problem step by step.



https://arxiv.org/pdf/2201.11903

Tree of Thoughts

- Backtracking
- Self-evaluation



https://arxiv.org/pdf/2305.10601

SCoT: Using Structure Code as Thoughts

```
def first_Repeated_Char(str):
    """
    Write a python function to find the first repeated
character in a given string.
    """
    Pass

Please understand the requirement and write a rough solving
process. It starts with a input-output structure. You
should use three basic structures to build the solving
process, including sequences, branches, and loops. The
necessary details should be written in natural languages.
```

Input



```
Input: str: a string
Output: ch: a repeated character in str
1: for each character ch in str:
2:   if ch appears more than once in str:
3:     return ch
4: return None
```

Thoughts



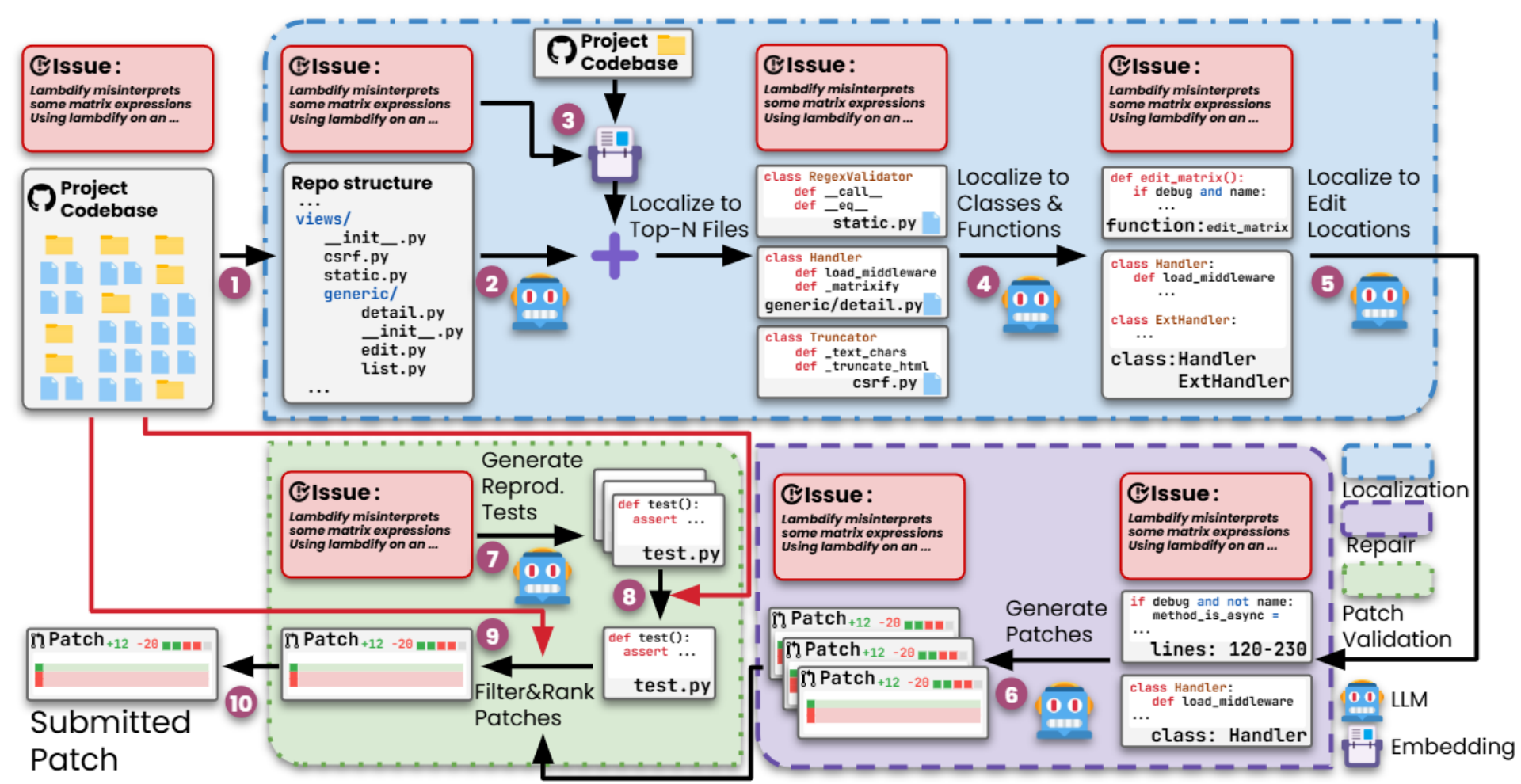
```
h = {}
for ch in str:
  if ch in h:
    return ch;
  else:
    h[ch] = 0
return None
```

Output

https://arxiv.org/pdf/2305.06599

Agentless

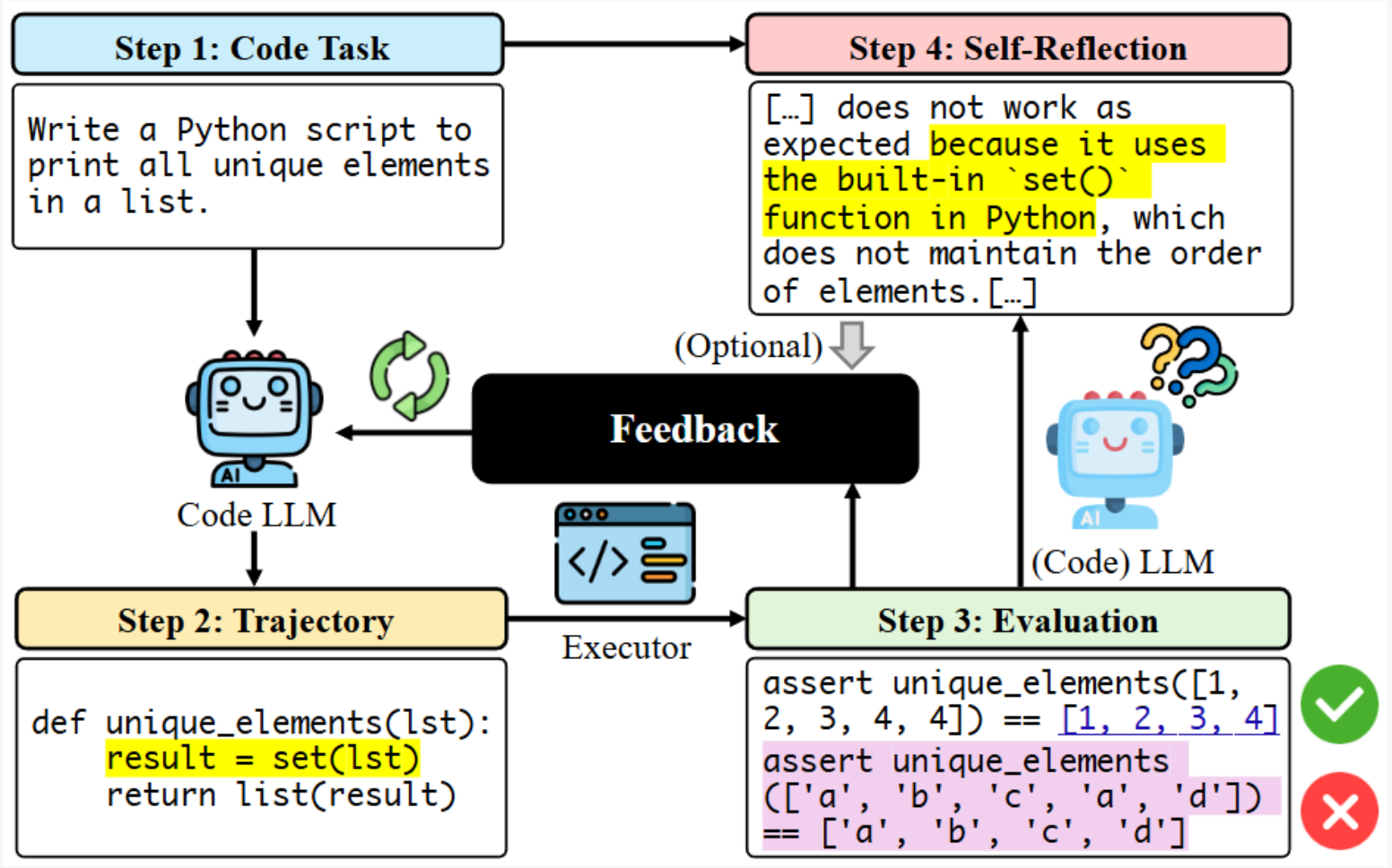
Predefined steps and prompt templates



https://arxiv.org/pdf/2407.01489

Can we improve prompts during executions?

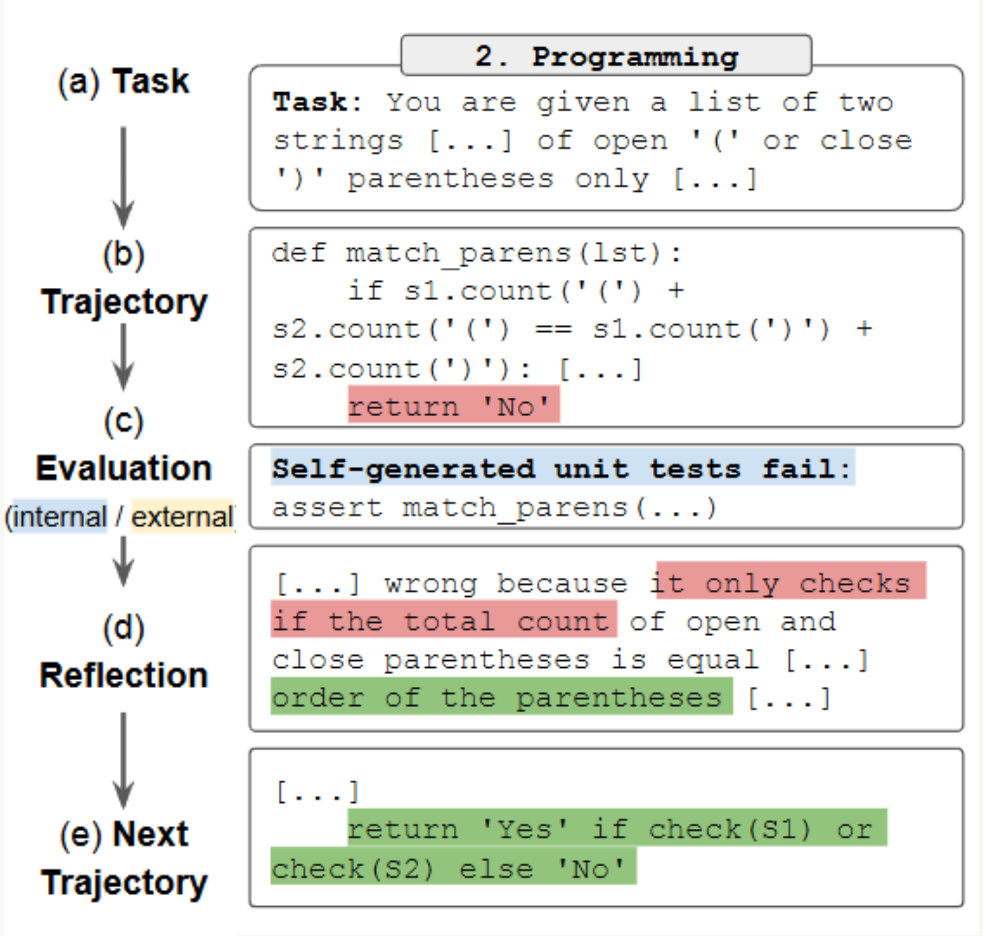
Self-improving Code Generation



https://arxiv.org/pdf/2406.00515

Reflexion

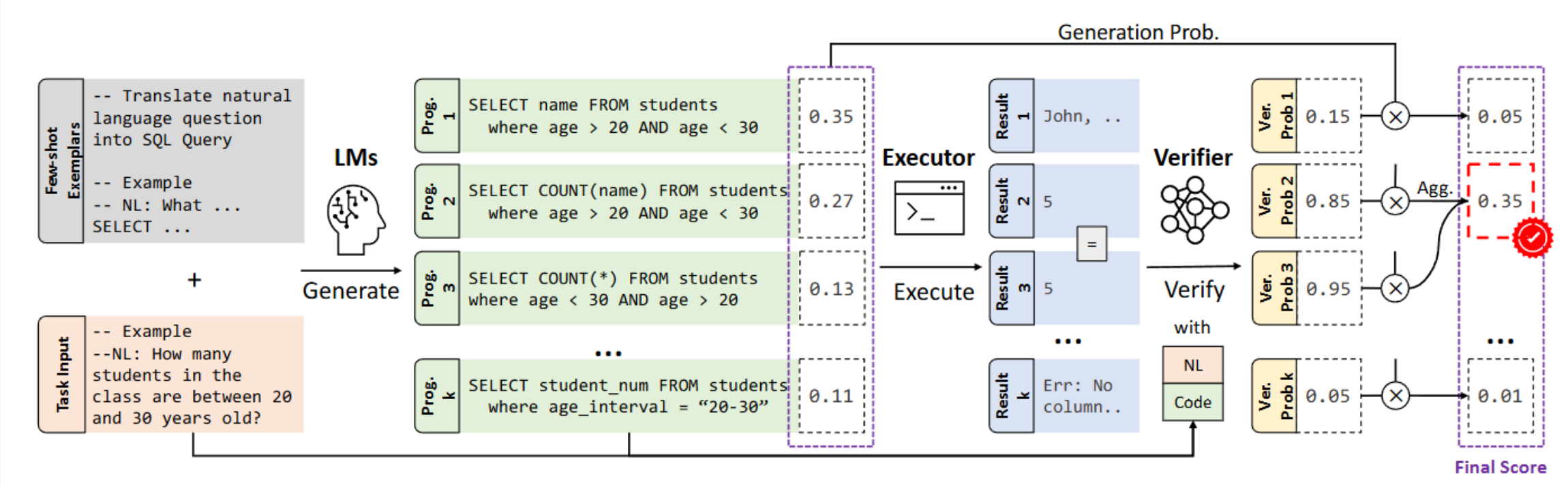
Store the reason for the last failed trajectory into memory for next iteration of prompting.



https://arxiv.org/pdf/2303.11366

Generate Multiple Candidate Code

Generate multiple code blocks at the same time, and use another model to rank and return the top candidate.

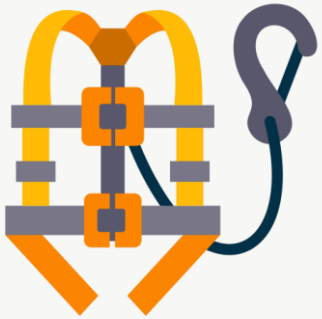


https://arxiv.org/pdf/2304.05128

Next Lecture



Model



Harness



Benchmarks



香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

Questions

Next class: Coding II