

Self-assisting and Cooperative Large Language Models

Lei Li

January 22, 2024



Carnegie Mellon University
Language Technologies Institute

Thank you!

- Prof. Huan Sun



- Prof. Tanya Berger-Wolf



Large Language Model Products

Google

Bard

PaLM

Gemini

OpenAI

ChatGPT
GPT-4

Meta
Llama 2

ANTHROPIC
Meet
Claude

A next-generation AI assistant for your tasks, no matter the scale
下一代AI助手，无论规模大小

Request Access

LLMs are powerful



Grammar correction

Convert ungrammatical statements into standard English.



Summarize for a 2nd grader

Simplify text to a level appropriate for a second-grade student.



Parse unstructured data

Create tables from unstructured text.



Emoji Translation

Translate regular text into emoji text.



Calculate time complexity

Find the time complexity of a function.



Explain code

Explain a complicated piece of code.



Keywords

Extract keywords from a block of text.



Product name generator

Generate product names from a description and seed words.

create a funny viral tweet about my talk at Ohio State University with the title "Self-assisting and Cooperative Large Language Models"



ChatGPT

🚀🤖 Excited to drop some serious "Buckeye Brainpower" at #OSU!

🌰📚 Join me as I unravel the secrets of "Self-assisting and Cooperative Large Language Models"! 🧠💡

👾 I'll be spilling the beans on how LLMs can write code like it's a poetry slam and tackle everyday tools like a pro! 💻🔧

👨‍🏫 Ever wondered if LLMs can be teaching assistants for each other? Heck yeah! It's like AI Inception! 🤸📝

Join the OSU giggle gang and me for a talk that's as entertaining as a Buckeye football game, but with fewer touchdowns and more tech!

🏈🤣🤖 #OSUCodeJokes #BuckeyeBrains #LLMGenius #AlatOSU 🎓🔊

Language Models: The Power of Predicting Next Word

Santa Barbara has very nice _____

$\text{Prob.}(\text{next_word} \text{prefix})$	
beach	0.5
weather	0.4
snow	0.01

Pittsburgh is a city of _____

bridges	0.6
corn	0.02

Language Model: $P(x_{1..T}) = \prod_{t=1}^T P(x_{t+1}|x_{1..t})$

Predict using Neural Nets

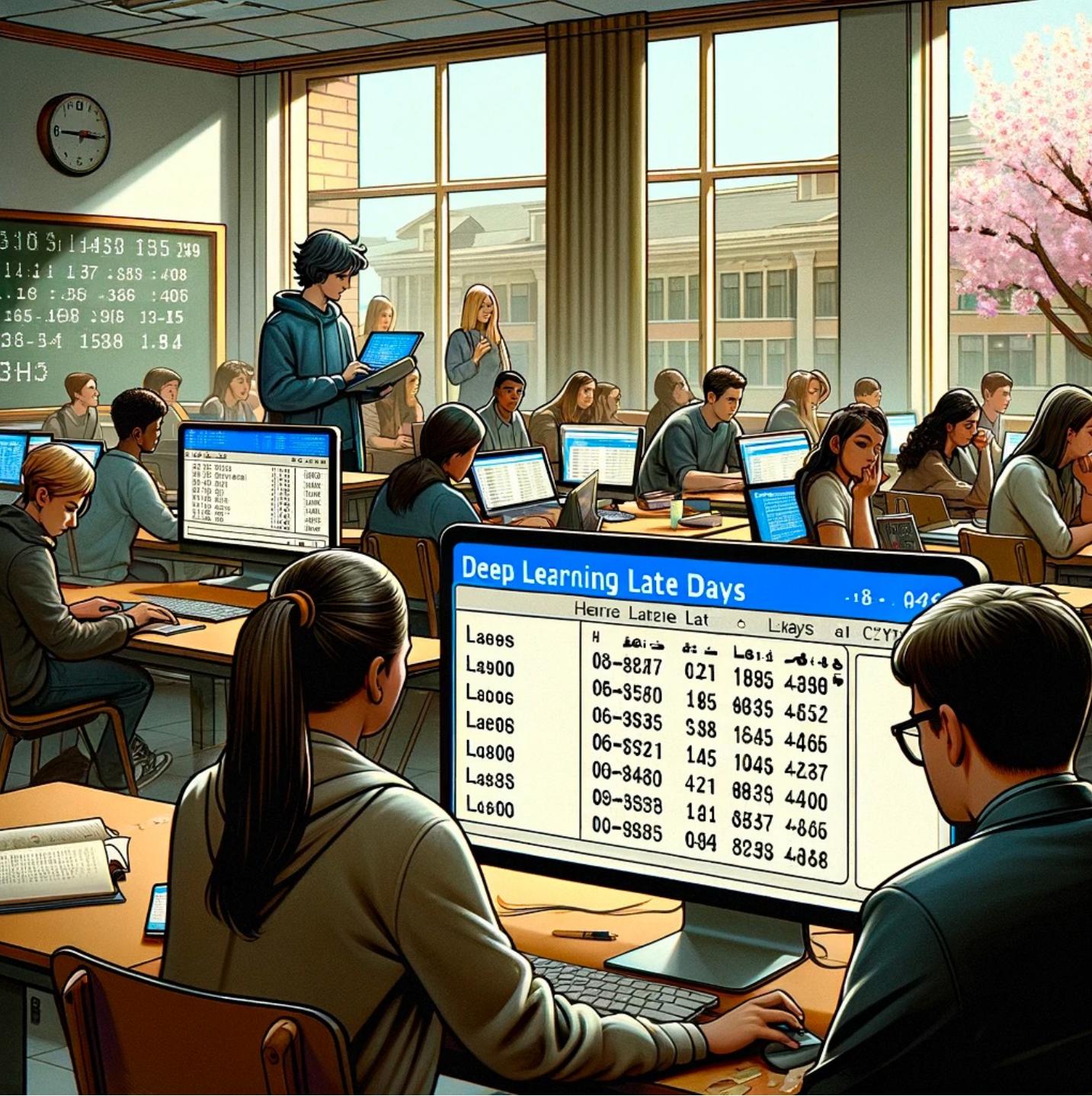
Why is ChatGPT changing AI landscape

- Pre-training on very large raw data (300B tokens) + small human feedback

Pittsburgh is a city of _____

bridges	✓
corn	✗

- **Instruction following:** Easy to use: through text instruction
- **In-context learning:** Generalize well to versatile tasks, by showing a few examples at use time.



But even the best
LLM makes
mistakes!



Outline

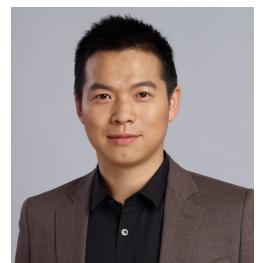


- Solve Algorithmic Problems with Self-generated Oracles
- Syntax-Error Free and Generalizable Tool-Use for LLMs
- Cooperative Study Assistant for Complex Reasoning Tasks
- Final thoughts



ALGO: Synthesizing Algorithmic Programs with LLM-Generated Oracle Verifiers

Kexun Zhang, Danqing Wang, Jingtao Xia, William Yang Wang, Lei Li



Can LLMs generate correct and efficient programs?

Given an integer n , implement a function $f(n)$ that computes $1 + 2 + 3 + \dots + n$.

LLM Gen1:

```
def f(n):  
    return 1+2+3+...+n
```



LLM Gen2:

```
def f(n):  
    return sum(range(1, n+1))
```



but inefficient

Ideal:

```
def f(n):  
    return (1+n)*n//2
```

Why Algorithmic Problems are Hard for LLM (and Human)

- Algorithm ideas
- Data Structure
- Math derivation

Given an integer n , implement a function $f(n)$ that computes $1 + 2 + 3 + \dots + n$.

```
def f(n):  
    return (1+n)*n//2
```

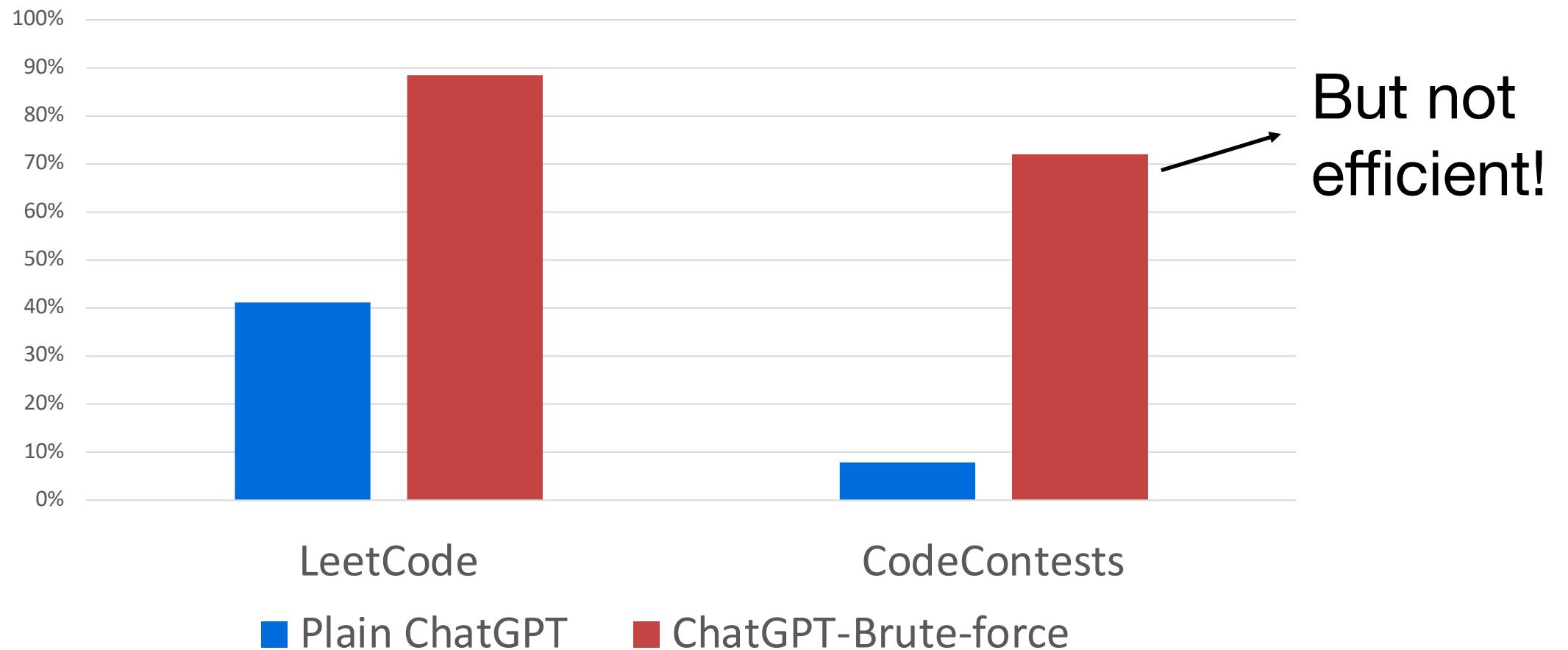
LLMs can generate brute-force solutions

Given an integer n , implement a function $f(n)$ that computes $1 + 2 + 3 + \dots + n$. Please do not care about efficiency, use brute-force approach.

```
def f(n):  
    return sum(range(1, n+1))
```

LLMs can generate brute-force programs

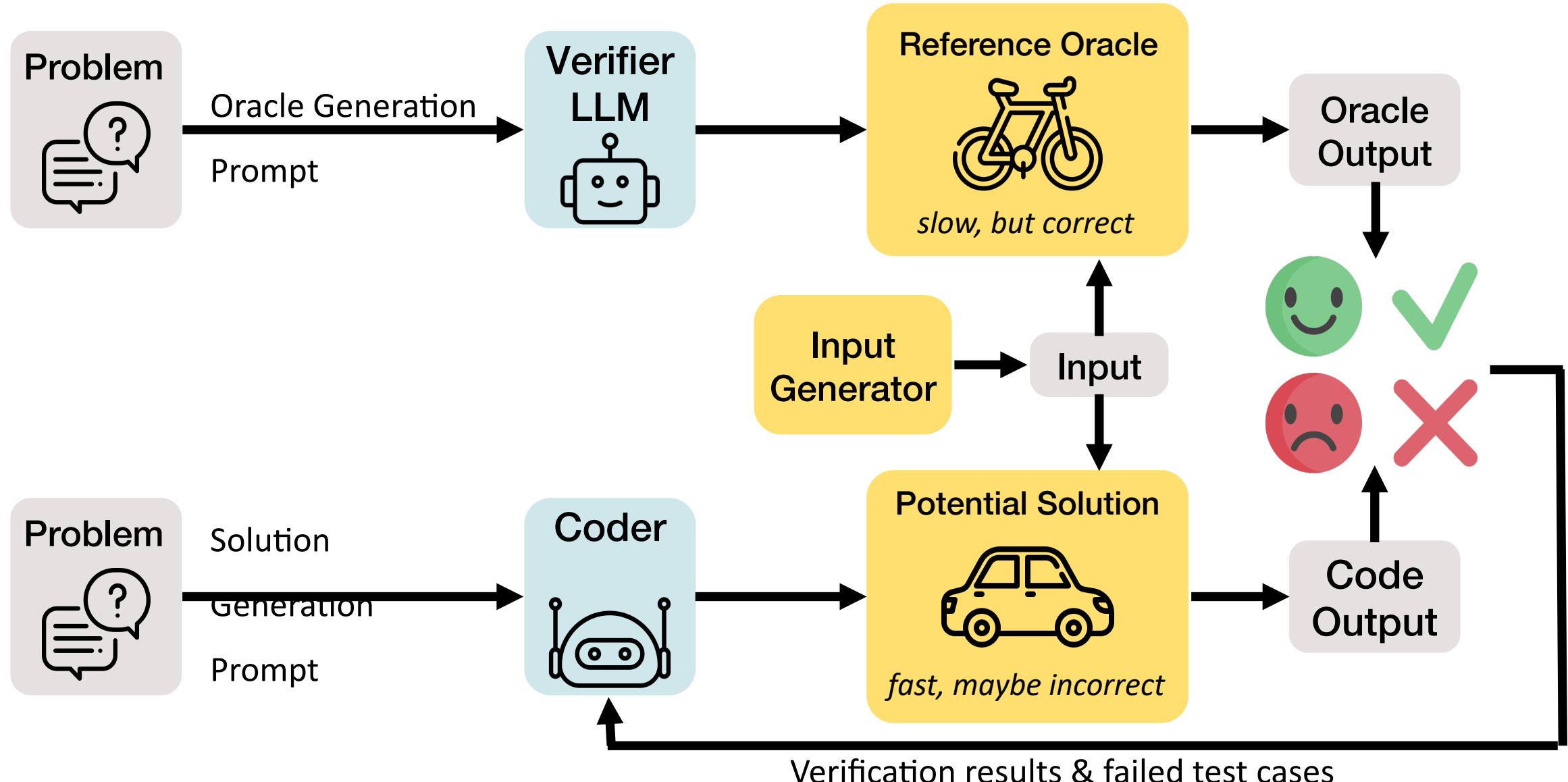
Correctness of ChatGPT on Competition-level
Problems



Proposed Solution ALGO: Intuition

- Using LLM generated brute-force program as an oracle
- Use LLM to generate test input data and Oracle to compute outputs → synthetic test cases
- Instruct LLM to generate efficient programs
- Verify program's correctness using the synthetic test cases

ALGO



Example: Repair Cars

You are given an integer array representing the ranks of some mechanics.

ith mechanic has ranks[i], and can repair n cars in rank[i] * n^2 minutes.

Given the total number of cars .

Return the minimum time taken to repair all the cars.

Example: Repair Cars

- Mechanics: 3
- Rank: [3, 1, 2]
- Cars: 6

		Evenly			Best		
	Rank(r)	3	1	2	3	1	2
	Allocation(n)	2	2	2	1	3	2
	Time(r^*n^2)	12	4	8	3	9	8
	Total time	12			9		

Repair Cars: Brute-force Algorithm

- Key idea: Enumerate all possible cars for each

- Plan 1: [1, 1, 4]

- Plan 2: [1, 2, 3]

- Plan 3: [1, 3, 2]

- Calculate time

- Compare and obtain min time

	Best			
Rank(r)	3	1	2	
Allocation(n)	1	3	2	
Time(r^*n^2)	3	9	8	
Total time	9			

Please solve this problem with a brute-force algorithm.

You are given an integer array ranks representing the ranks of some mechanics. ranks[1] ...omitted... Return the minimum time taken to repair all the cars.

Examples

{examples, omitted}

Instead of defining your function in the Solution class, change your classname to BruteforceSolution. Please make sure you use the most straightforward and brute-force algorithm to solve the problem. Do not consider any efficiency issue, make the solution as brute-force as possible. It's okay for you to enumerate over a very large search space as long as the solution is correct.



```
class BruteforceSolution:
    def repairCars(self, ranks: List[int], cars: int) -> int:
        num_mechanics = len(ranks)
        min_time = float('inf')
        def generate_allocations(allocation, remaining_cars):
            nonlocal min_time
            if len(allocation) == num_mechanics:
                if sum(allocation) == cars:
                    times = [ranks[i] * allocation[i] * allocation[i]
                             for i in range(num_mechanics)]
                    max_time = max(times)
                    min_time = min(min_time, max_time)
            return
        for num_cars in range(remaining_cars + 1):
            new_allocation = allocation + [num_cars]
            new_remaining_cars = remaining_cars - num_cars
            generate_allocations(new_allocation, new_remaining_cars)
        generate_allocations([], cars)
    return min_time
```

ALGO: Key Idea

1. Gen Oracle: Using LLM generated brute-force program as an oracle
2. Explore Idea: Prompt LLM to random sample/enumerate an “idea”
3. Gen Program: Use LLM to generate a program according to the “idea”
4. Gen Cases: use LLM to generate test input data
5. Verify: both oracle and candidate on test cases.

LLM Generating Candidate Solution

Potential Bin

```
class Solution:
    def repairCars(self, ranks: List[int], cars: int):
        def countCarsRepaired(total_cars: int, ranks: List[int], minutes: int) -> int:
            total_cars = 0
            for rank in ranks:
                n = int((minutes / rank) ** 0.5)
                total_cars += min(n, len(ranks))
            return total_cars
        left = 1
        right = max(ranks) * len(ranks)
        while left < right:
            mid = (left + right) // 2
            if countCarsRepaired(left, ranks, mid) <= cars:
                left = mid + 1
            else:
                right = mid
        return left
```

Potential Greedy Solution

```
class Solution:
    def repairCars(self, ranks: List[int], cars: int):
        ranks.sort()
        max_repair_time = 0
        assigned_cars = 0
        for rank in ranks:
            num_cars = (cars + len(ranks) - assigned_cars - 1) // (len(ranks) - assigned_cars)
            num_cars = min(num_cars, cars - assigned_cars)
            repair_time = rank * num_cars * num_cars
            max_repair_time = max(max_repair_time, repair_time)
            assigned_cars += num_cars
        return max_repair_time
        num_cars = min(num_cars, cars - assigned_cars)
        repair_time = rank * num_cars * num_cars
        max_repair_time = max(max_repair_time, repair_time)
        assigned_cars += num_cars
    return max_repair_time
```

Comparing Outputs with the Oracle

Test Inputs Generated by ALGO

Test Input I_0 : ranks=[4,2,3,1], cars=10

I_1 I_2 I_3 . . .

Reference Oracle Generated by LLM

```
class BruteForceSolution:
    def repairCars(self, ranks: List[int], cars: int) -> int:
        num_mechanics = len(ranks)
        min_time = float('inf')
        def generate_allocations(allocation, remaining_cars):
            nonlocal min_time
            if len(allocation) == num_mechanics:
                if sum(allocation) == cars:
                    times = [ranks[i] * allocation[i] * \
                             allocation[i] for i in range(num_mechanics)]
                    max_time = max(times)
                    min_time = min(min_time, max_time)
                    return
            for num_cars in range(remaining_cars + 1):
                new_allocation = allocation + [num_cars]
                new_remaining_cars = remaining_cars - num_cars
                generate_allocations(new_allocation, \
                                     new_remaining_cars)
        generate_allocations([], cars)
        return min_time
```

Potential Greedy Solution

```
class Solution:
    def repairCars(self, ranks: List[int], cars: int):
        ranks.sort()
        max_repair_time = 0
        assigned_cars = 0
        for rank in ranks:
            num_cars = (cars + len(ranks) - assigned_cars - 1) // (len(ranks) - assigned_cars)
            num_cars = min(num_cars, cars - assigned_cars)
            repair_time = rank * num_cars * num_cars
            max_repair_time = max(max_repair_time, repair_time)
            assigned_cars += num_cars
        return max_repair_time
```

System Judge:
Wrong Answer

→ 100



Potential Binary Search Solution

```
class Solution:
    def repairCars(self, ranks: List[int], cars: int) -> int:
        def countCarsRepaired(ranks, minutes):
            total_cars = 0
            for rank in ranks:
                n = int((minutes / rank) ** 0.5)
                total_cars += n
            return total_cars
        left = 1
        right = max(ranks) * cars * cars
        while left < right:
            mid = (left + right) // 2
            if countCarsRepaired(ranks, mid) < cars:
                left = mid + 1
            else:
                right = mid
        return left
```

→ 16

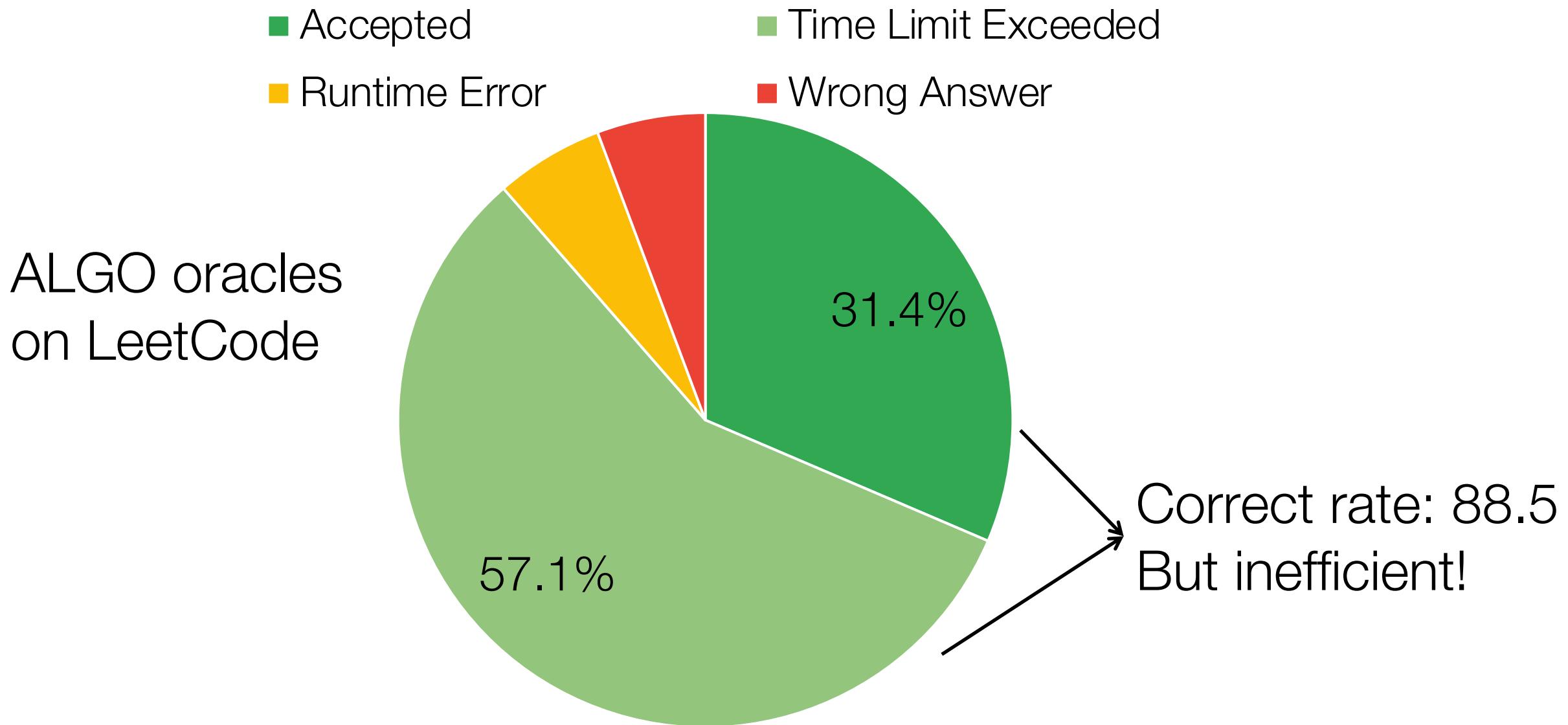


System Judge:
Accepted

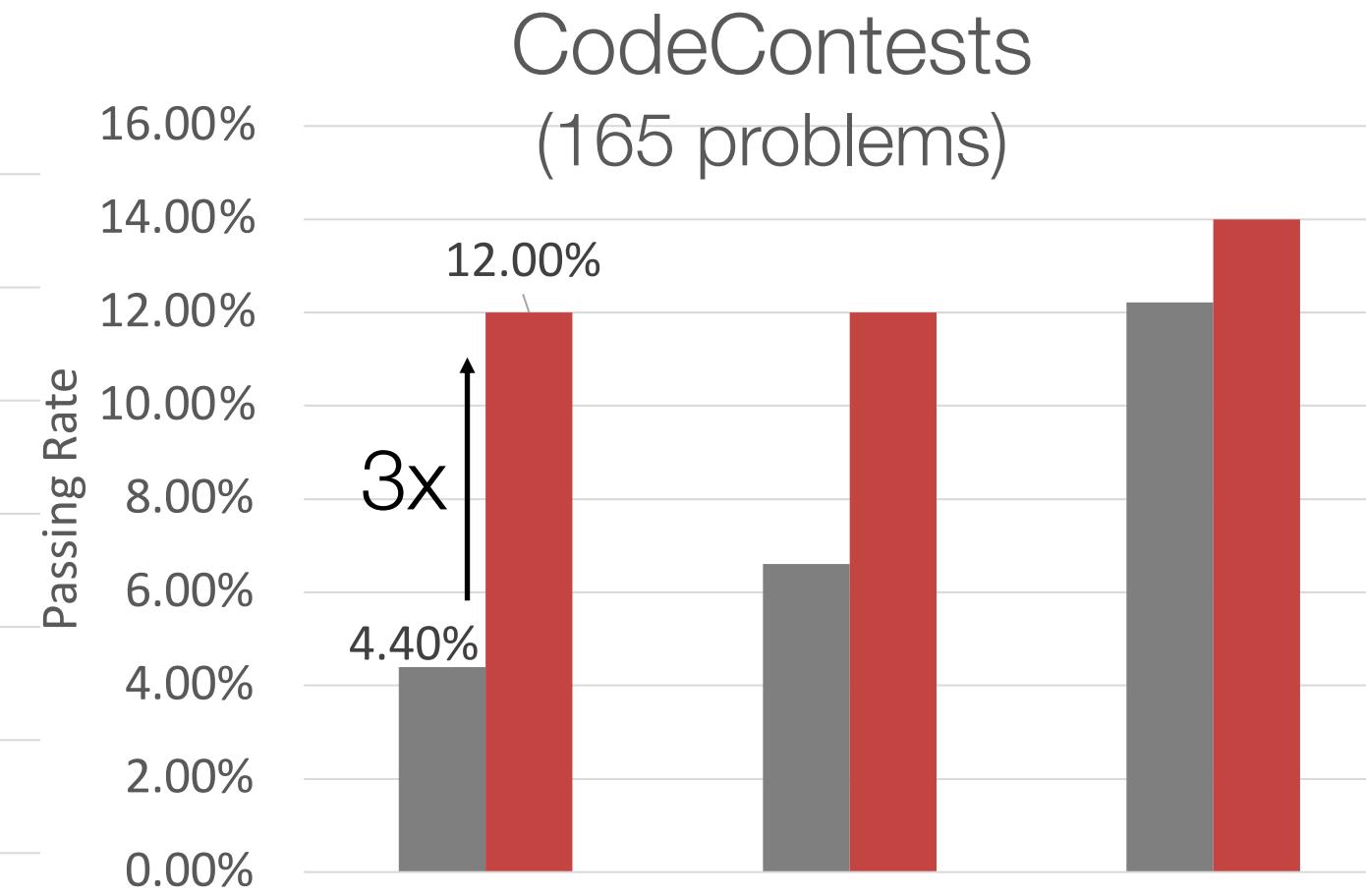
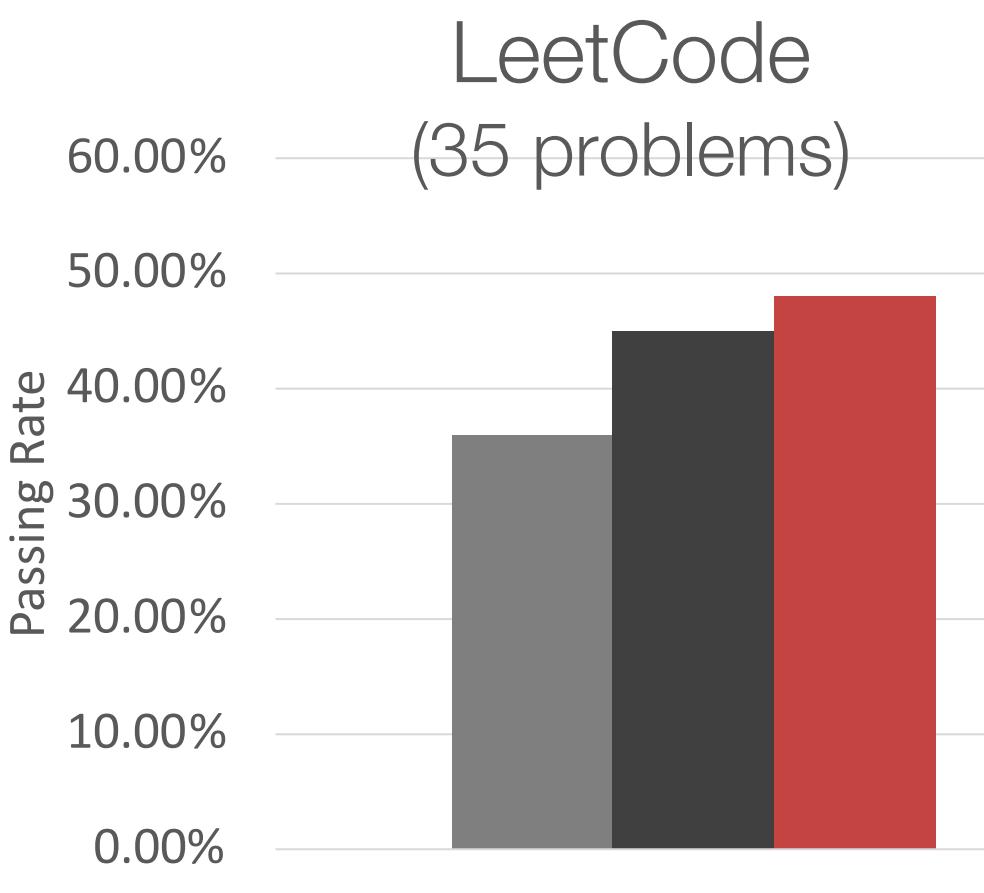
ALGO implementation

- ALGO works with all sorts of models and strategies.
- Base Model: Codex, GPT-2, ChatGPT, ...
- Strategy for Exploring Algorithmic “Ideas”
 - Sampling
 - Lookahead Search
 - Idea Sampling

ALGO generated oracles are mostly correct.



ALGO solves 3x problems!

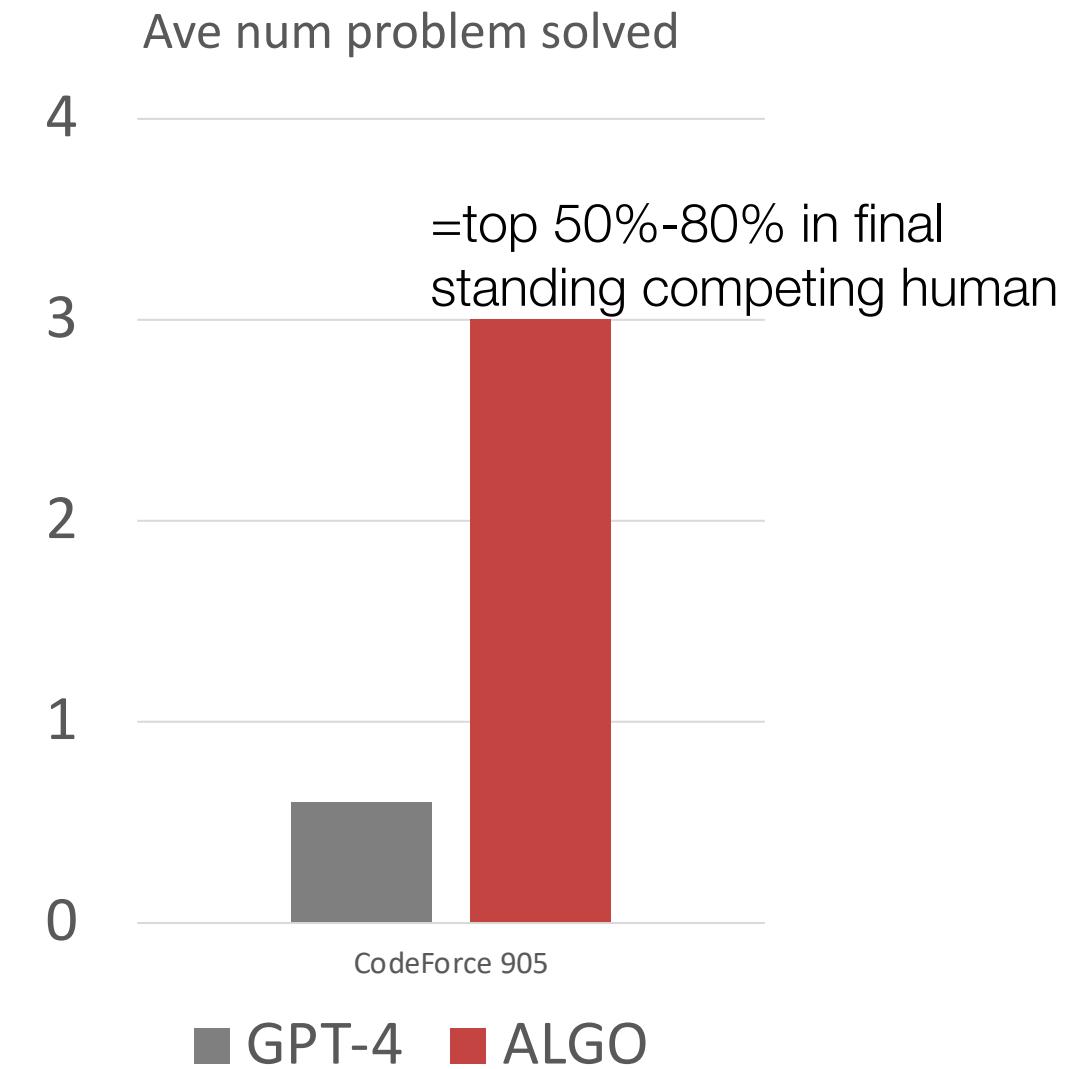


■ ChatGPT-Code ■ GPT4 ■ ALGO

■ ChatGPT-Code ■ ALGO

Real Competition: ALGO is top competitive to Human

- We participate in a real codeforce contest (905) on Oct 22, 2023
- Human can submit many times
- We use both GPT-4 (version Jun 13, 2023) and ALGO(+GPT-4) to sample 20 submissions
- 50% human solved < 3 problems



Summary of ALGO

- LLM self-generated slow programs could ensure correctness, and can be used as oracles
- ALGO could verify candidate programs with oracles and synthesized test cases.
- ALGO could generate efficient programs for algorithmic problems!

Outline



- Solve Algorithmic Problems with Self-generated Oracles
- Syntax-Error Free and Generalizable Tool-Use for LLMs
- Cooperative Study Assistant for Complex Reasoning Tasks
- Final thoughts

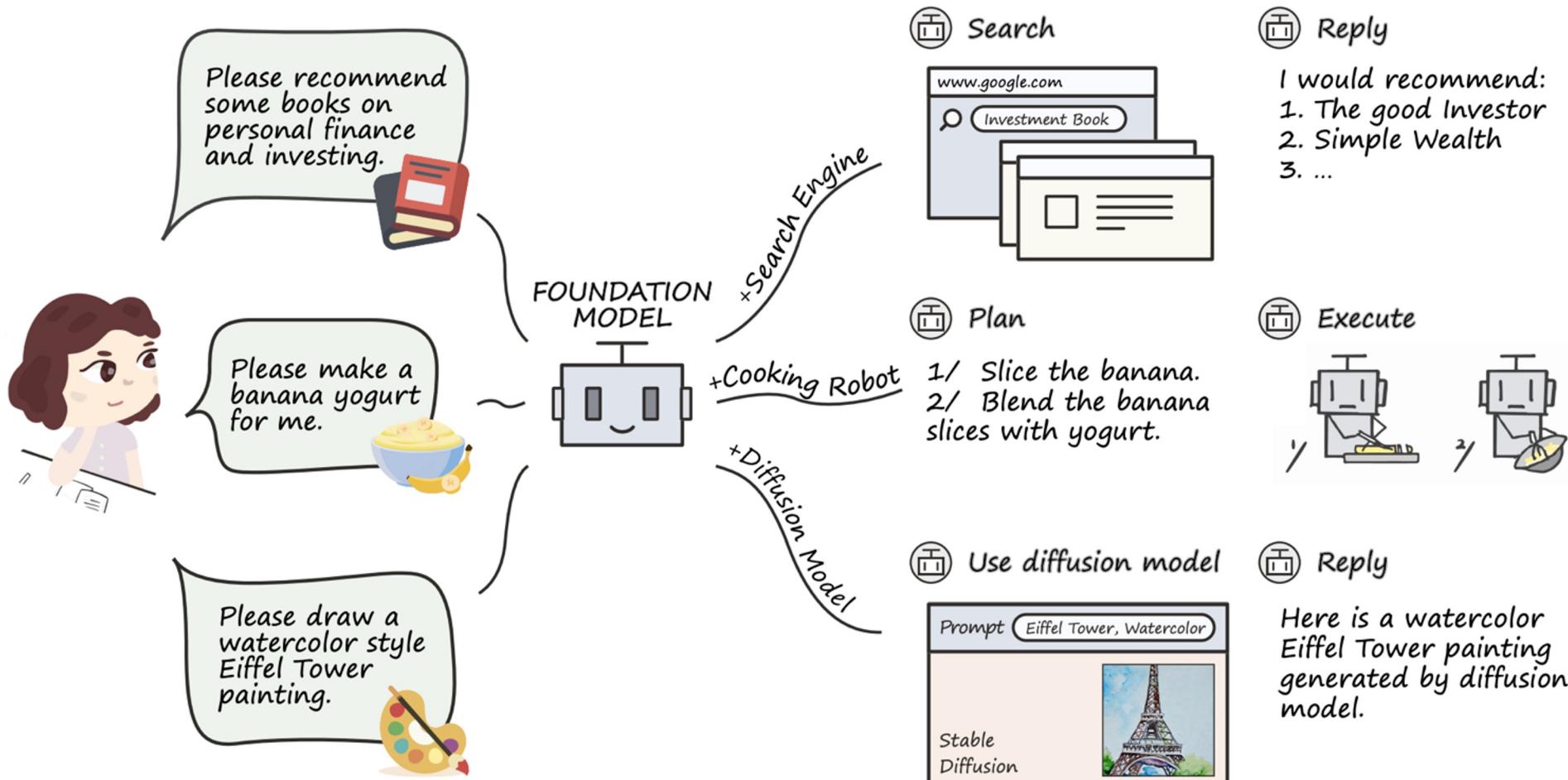


ToolDec: Syntax Error-Free and Generalizable Tool Use for LLMs via Finite-State Decoding

Kexun Zhang*, Hongqiao Chen*, Lei Li, William Yang Wang



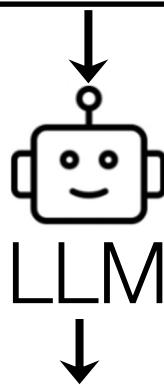
Tool-Using LLM Agent



Proposed ToolDec eliminates syntax errors

Tools: `multiply(a, b)` computes the product of numbers a and b

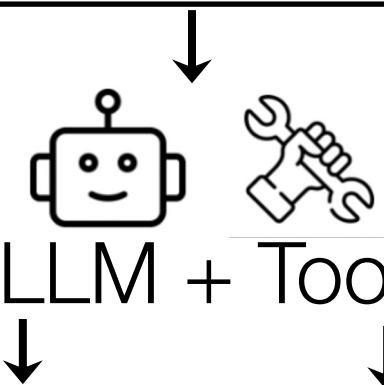
Q: The diameter of a circle is 123, $\pi=3.14$, what's its perimeter?



LLM

Its perimeter is
 $\pi \times 123 = 196$

Generating
Wrong Answer



LLM + Tool

Its perimeter is
`product(3.14, 123)`

Non-Existential Tool



LLM + ToolDec

Its perimeter is
`multiply(pi, 123)`

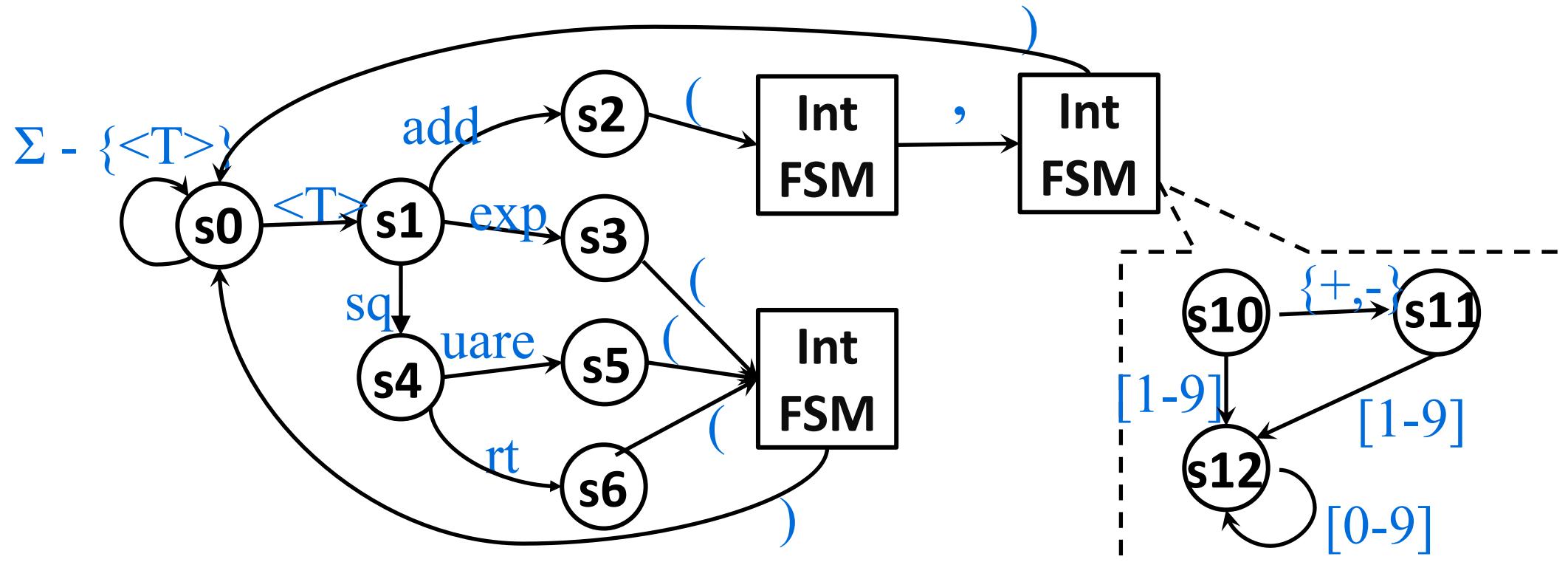
Invalid Tool
Argument

Its perimeter is
`multiply(3.14,
123)`

Correct Tool Call

ToolDec: Finite-State Constrained Decoding

FSM are automatically constructed from function signatures



Step	Generated Text	Current State g	Valid Next Tokens	Next Token	Next State
t	Its	s_0	whole vocab.	'area'	s_0
$t + 1$	Its area	s_0	whole vocab.	'is'	s_0
$t + 2$	Its area is	s_0	whole vocab.	'<T>'	s_1
$t + 3$	Its area is <T>	s_1	'add', 'exp', 'sq'	'sq'	s_4
$t + 4$	Its area is <T>sq	s_4	'uare', 'rt'	'uare'	s_5

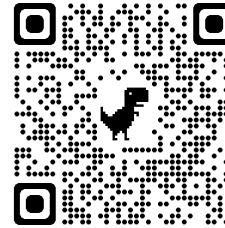
Advantages of ToolDec

	Syntax-error free	zero-shot	Document-free
LLM Fine-tuning	✗	✓	✗
In-context Learning	✗	✗	✓
ToolDec	✓	✓	✓

Outline



- Solve Algorithmic Problems with Self-generated Oracles
- Syntax-Error Free and Generalizable Tool-Use for LLMs
- Cooperative Study Assistant for Complex Reasoning Tasks
- Final thoughts



Learning from Mistakes via Cooperative Study Assistant for Large Language Models

Danqing Wang



Lei Li



LE

You

We have two orange and two blue balls. We randomly pick two balls. You know one of them is orange. What is the probability that the other ball is also orange?

**ChatGPT**

Probability of selecting two orange balls + Probability of selecting one orange ball and one blue ball = $1/6 + 1/3 = 1/2$



LE

You “Let us use self-reflection”

Please review your answer, find problem, and revise.

**ChatGPT**

I didn't account properly for the condition that's given ("at least one of them is orange") in the calculation of probabilities. The probability that both balls drawn are orange given that at least one is orange is $1/5$.



40

LE

You

Taxis arrive at a hotel with exponentially distributed time duration and average 2 minutes. What is the probability that three taxis arrive within 6 minutes?

**ChatGPT**

85.79%.

X

LE

You “Let us use self-reflection”

Please review your answer, find problem, and revise.

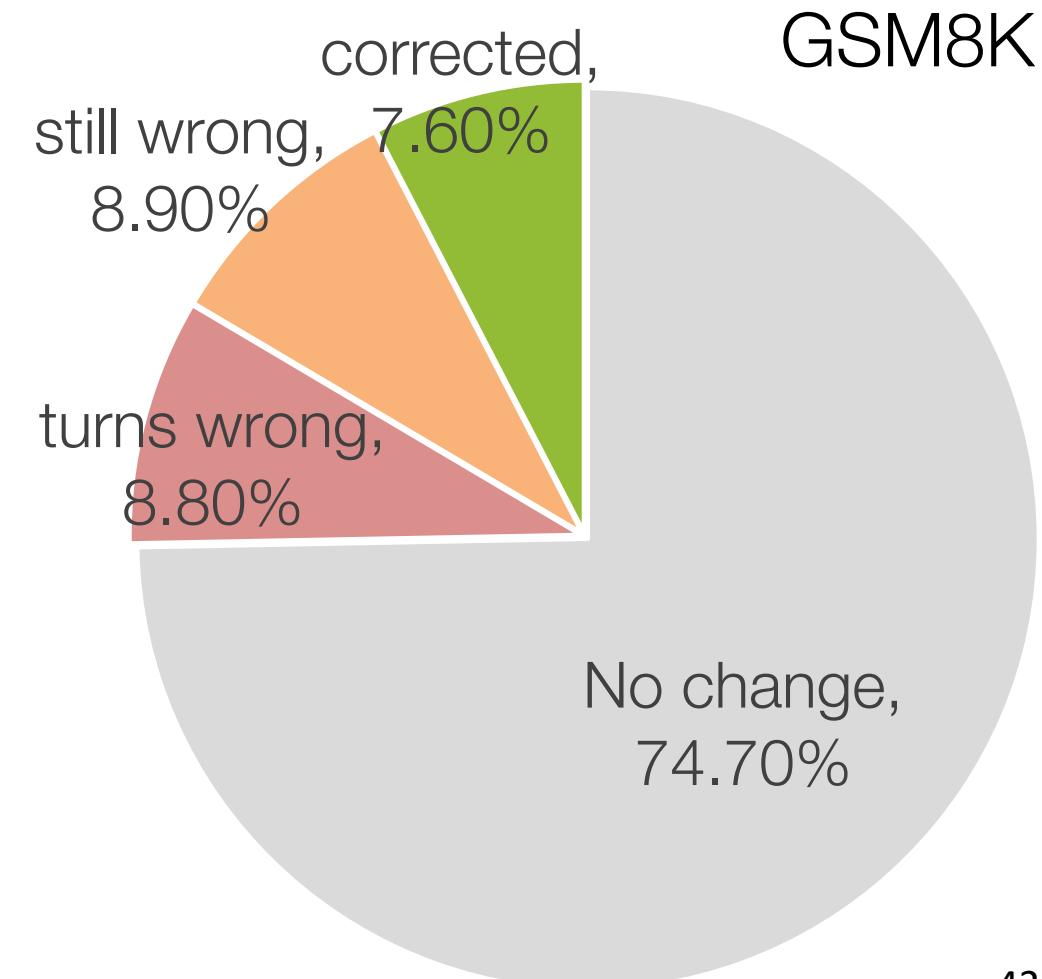
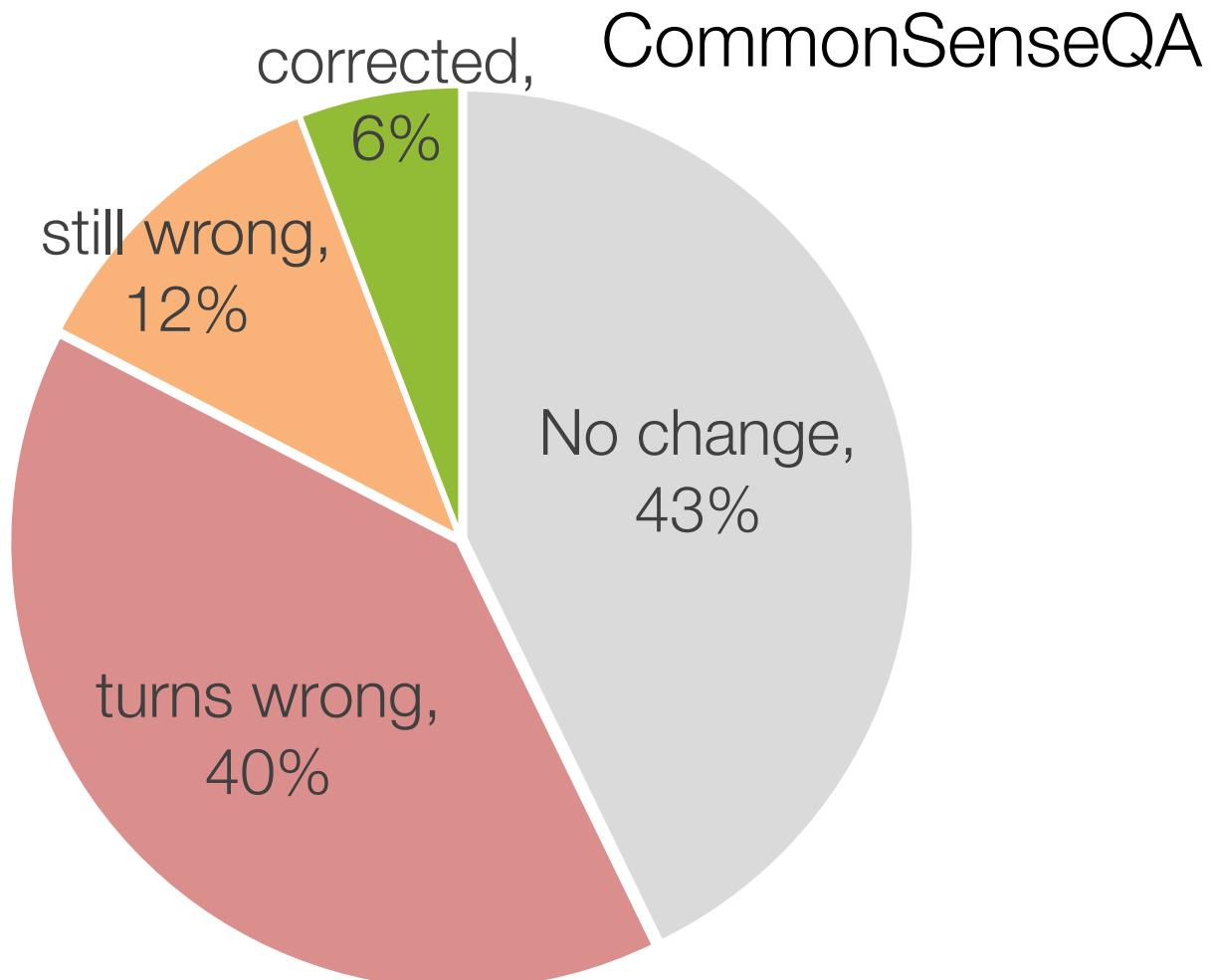
**ChatGPT**

I've identified a mistake in the way the probabilities are combined for the three cabs. The correct answer is ... 85.72%

X

Large Language Models Cannot Self-Correct yet

Change after LLM-reflection



To improve LLM's reasoning, we need

- The ground truth to guide refinement
- A model to provide informative feedback
- A memory to store previous mistakes and suggestions

SALAM: A Cooperative Study Assistant

- Help LLMs to learn from mistakes by interactive cooperation



Main LLM (big, could not finetune)

- Generate initial response according to the query
- take feedback and revise response



Study assistant (small, easy to train)

- take main LLM response and provide feedback
- store mistakes and suggestions to memory



Study Assistant to guide revision

- A separate GPT-LLM
- Instruction: query & ground truth, previous response, current response
- Generates:
 - Analysis (why is wrong)
 - Guideline (how to avoid)

Instruction

Jane thought today is 3/11/2002, but today is in fact Mar 12, which is 1 day later. What is the date a month ago?

Options:

- (A) 04/12/2001
- (B) 02/11/2002
- (C) 02/12/2002
- (D) 02/08/2002
- (E) 05/22/2002
- (F) 02/18/2002

We get the answer (B) 02/11/2002 ; (A) 04/12/2001 from the model while the correct answer is (C) 02/12/2002 .

Please return with the following fields:

Analysis: explain the potential reason for prediction

Guideline: based on the reason, provide instruction to avoid similar mistakes.

Please do not mention the true answer or any specific option content in your response.

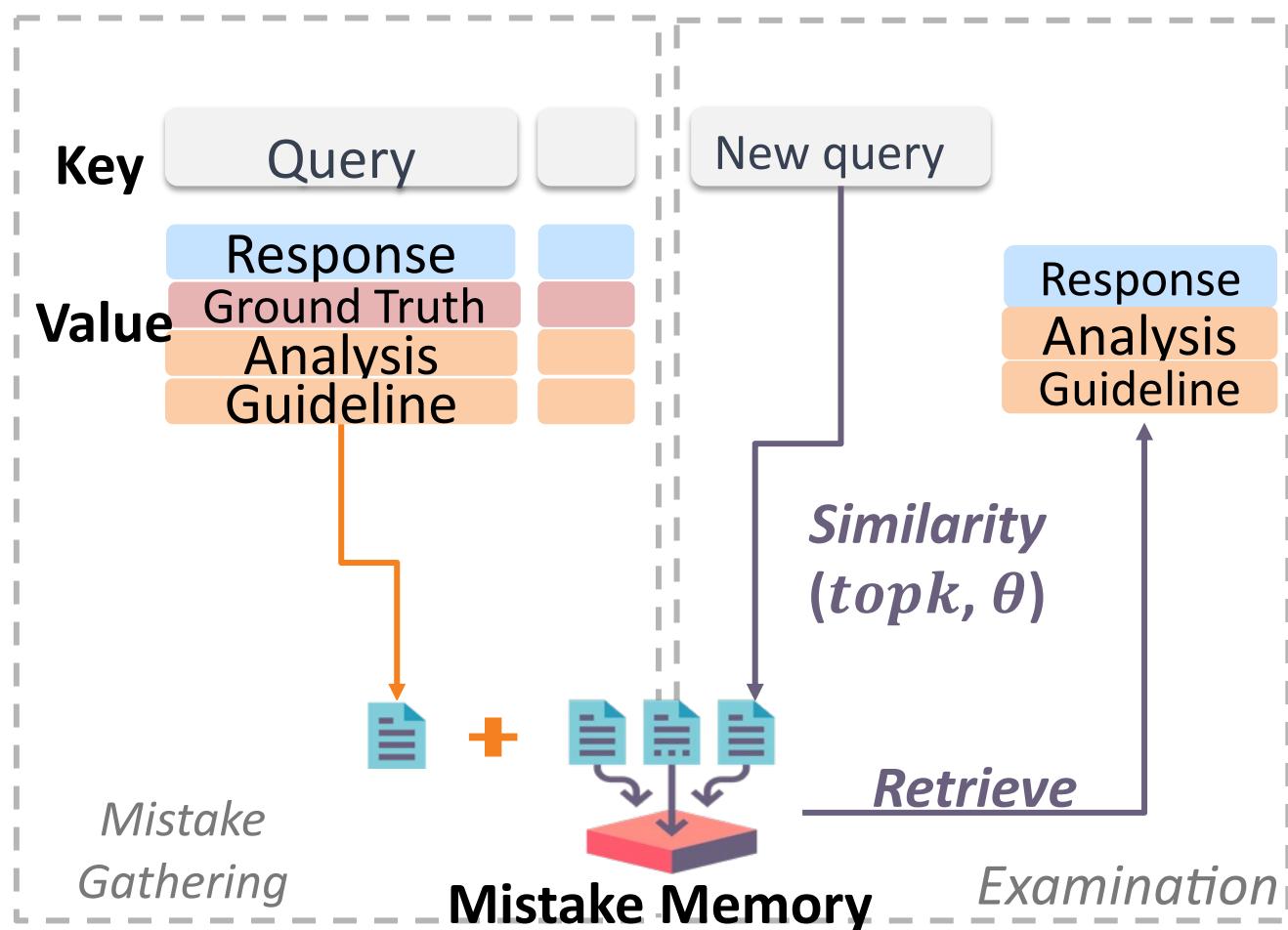
Study Assistant Response

Analysis: The model might have misunderstood the phrase “1 day later” in the context.

Guideline: For dates in a problem, identify the correct date from which calculations should be made. Also, make sure to maintain the correct format (MM/DD/YYYY) while providing the answer.

SALAM Learns from Mistake Memory

- Mistake Gathering (Training)
 - with ground truth, collect and store mistakes and feedback
 - several iterations between two agents
- Examination (Inference)
 - one pass interaction
 - no ground truth
 - retrieve similar mistakes and feedback



Jane thought today is 3/11/2002, but today is in fact Mar 12, which is 1 day later. What is the date a month ago?

Ground Truth

02/12/2002



Model-agnostic Study Assistant (SA)

- Agnostic to the main LLM architecture (GPT, Flan-T5, LLaMA, ...)
 - train a (relatively small) SA LLM to provide feedback
 - collect ~1k feedback examples from GPT4

**(query & ground truth, previous response, current response) =>
feedback**



Model-specific Study Assistant (SA)

- Provide specific suggestions for main LLM
- Model the SA-LLM interaction as MDP
 - Policy $\pi(a|s)$: provide feedback based on current state
 - State S: (query, response, context)
 - Action A: feedback generated by study assistant
 - Reward R: LLM performance
 - 1 if the LLM's revised response is correct
 - 0 otherwise

Learn Study Assistant Policy via Imitation Learning

- a replay dataset $D_{on} = \bigcup_{i=0}^N \bigcup_{t=1}^T (s_t^{(i)}, a_t^{(i)})$
 - N examples, and T iteration

- calculate the reward and keep

$$R(s_t^{(i)}, a_t^{(i)}) = 1$$

- Get the filtered successful dataset \tilde{D}_{on}
- finetune the study assistant on \tilde{D}_{on}

Instruction

Jane thought today is 3/11/2002, but today is in fact Mar 12, which is 1 day later. What is the date a month ago?

Options:

- (A) 04/12/2001
- (B) 02/11/2002
- (C) 02/12/2002
- (D) 02/08/2002
- (E) 05/22/2002
- (F) 02/18/2002

$s_1^{(0)}: t = 1 \text{ for } i =$

We get the answer (B) 02/11/2002 ; 04/12/2001 from the model while the correct answer is (C) 02/12/2002.

Please return with the following fields:

Analysis: explain the potential reason for prediction

Guideline: based on the reason, provide instruction to avoid similar mistakes.

Please do not mention the true answer or any specific option content in your response.

Study Assistant Response

$a_1^{(0)}$

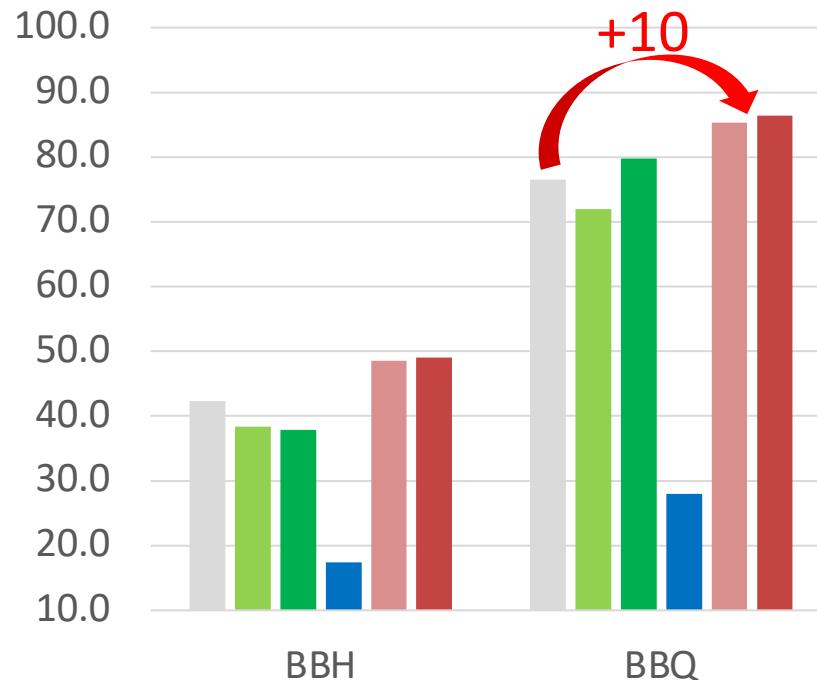
Analysis: The model might have misunderstood the phrase "1 day later" in the context.

Guideline: For dates in a problem, identify the correct date from which calculations should be made. Also, make sure to maintain the correct format (MM/DD/YYYY) while providing the answer.

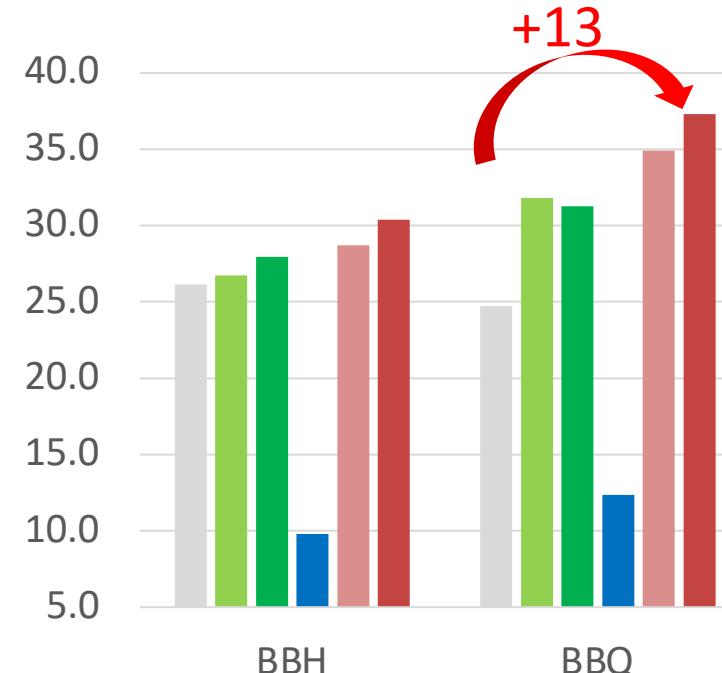
SALAM Significantly Boosts LLM Performance

Accuracy under the multi-choice setting

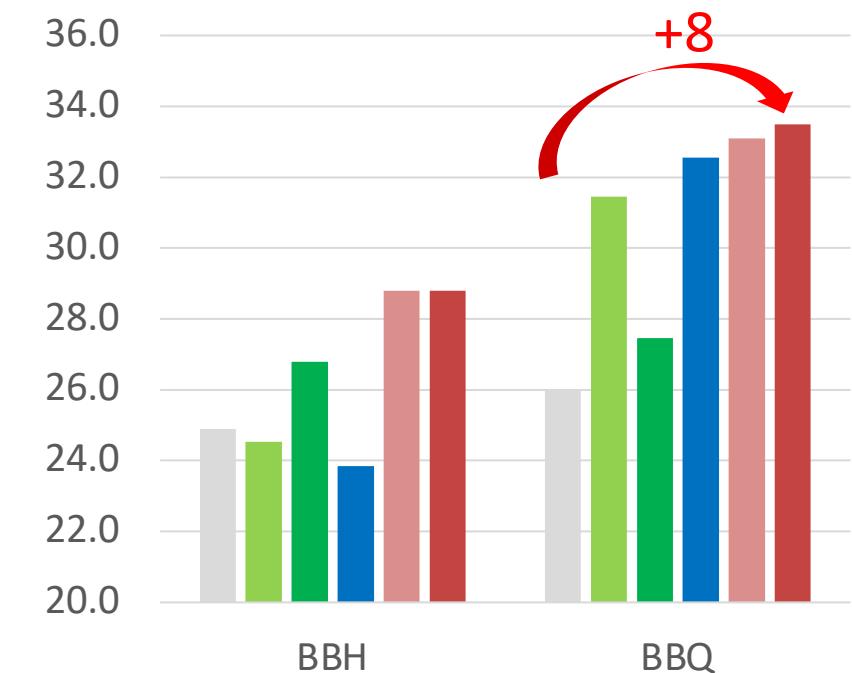
Flan T5 11B



LLaMA-1 7B



GPT3 NeoX 20B



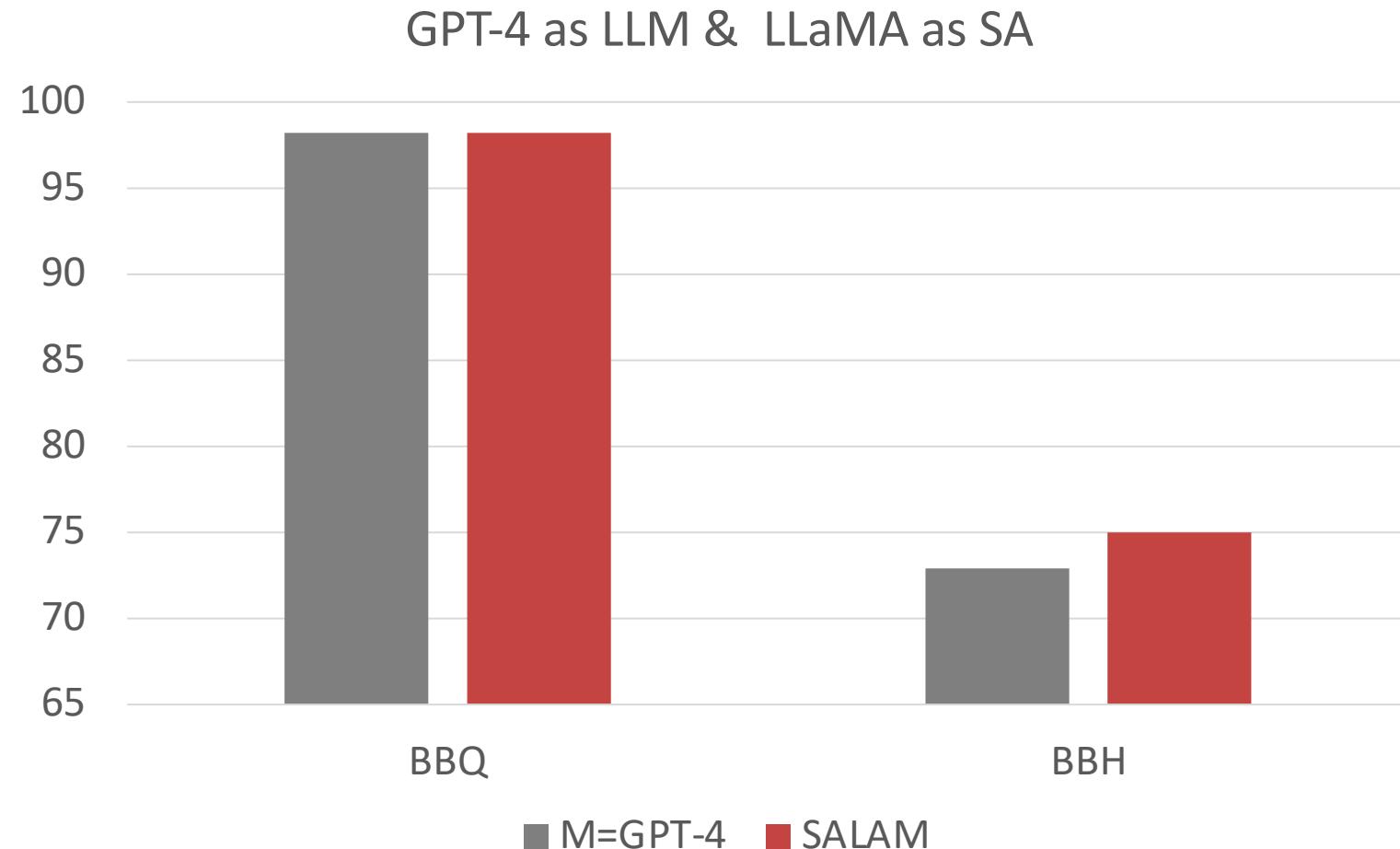
Legend:

- M
- M w/ corr
- M w/ err
- Self-refine
- SALAM
- SALAM w/ IL

Weak to Strong Learning!



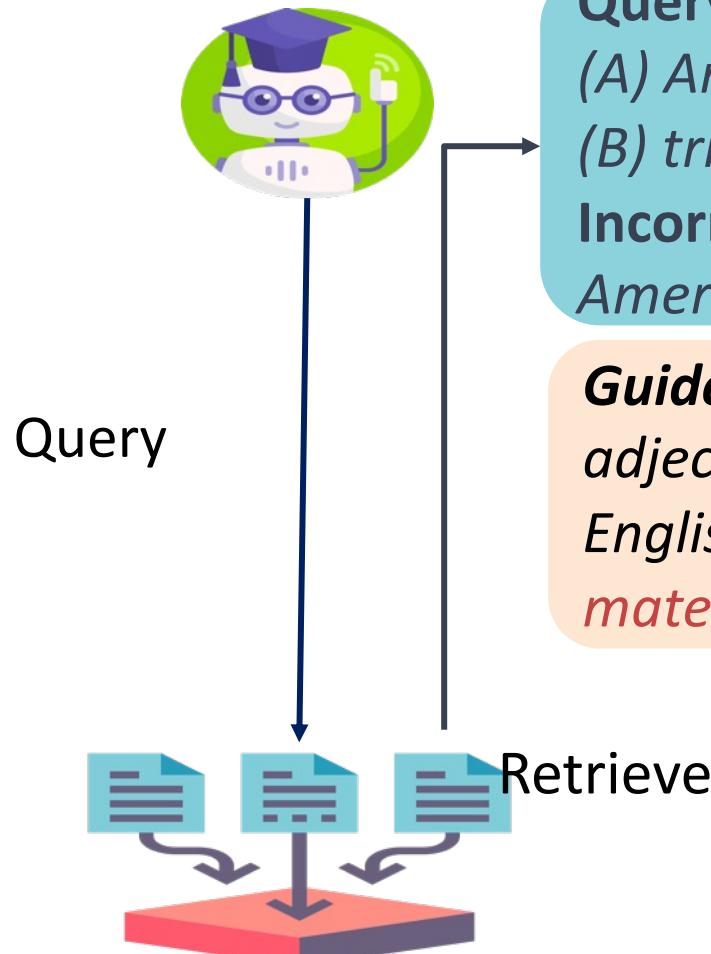
SALAM 7B can boosts GPT-4 performance on reasoning



Which sentence has the correct adjective order:

(A) red little silly cloth eating rectangular sock

(B) silly little rectangular red cloth eating sock



Query: Which sentence has the correct adjective order:

(A) American triangular computer

(B) triangular American computer

Incorrect Answer:

American triangular computer

Guideline: When evaluating sentences with multiple adjectives, consider the general rules of adjective order in English, such as *opinion, size, age, shape, color, origin, material, and purpose*.

(B)



LLM

SALAM learns better policy with Imitation Learning

Jane visits the bookstore on the 16th of each month starting from the October of 2009. It is her 5th visit to the bookstore today. What is the date one year ago from today?

Previous answer: 08/16/2009

Correct answer: 02/16/2009



SALAM

Guideline: ensure that you accurately calculate the date by considering the correct day of the month and subtracting the specified number of months from the given date.



SALAM w/ IL

Guideline: carefully consider the given information, such as the frequency of visits and the current visit number, to accurately calculate the elapsed time. Then, use this information to determine the correct date.

Summary of SALAM

- Cooperation between LLMs and study assistant (a second LLM)
- Guidance from SA improves LLMs' performance
- Model-specific guidance works better
- Learning from mistake Memory can avoid similar mistakes

Some Thoughts

- LLM needs feedback to improve performance (coding/reasoning)
- But, vague or incorrect feedback could mislead LLMs
- Where are the feedback from?
 - Self-generated oracle (when oracles are reliable?)
 - Another smaller LLM (or a set of LLMs)
 - Separately trained Metric (InstructScore, but not COMET/BLEURT/SEScore) [Xu et al, EMNLP 2023]
 - Memory (similar success or failures in the past)

generative AI systems for language and science

Assistant Professor
Language Technology Institute
Carnegie Mellon University



Multilingual NLP & Translation



Unified model for translating 400+ languages.



Xu et al. "Vocabulary Learning for Neural Machine Translation via Optimal Transport". ACL 2021 Best paper award.

Large Language Models

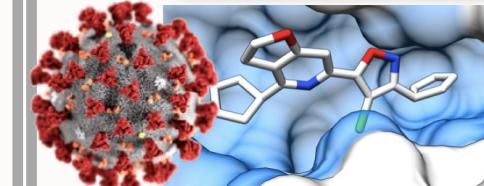
- code generation **ALGO**
- LLM tool use **ToolDec**
- LLM acceleration system **LightEq**
- LLM privacy **CRT**
- LLM Assessment
- LLM watermark: detecting AI-generated text
- LLM IP/copyright **GINSEW**



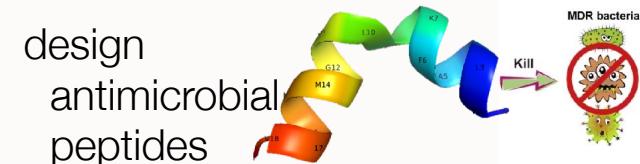
Xiaomingbot: automatic machine writing bot for sports/finance news, creative ads used by 100 million users.

Zhang, et al., "ALGO: Synthesizing Algorithmic Programs with Generated Oracle Verifiers", NeurIPS 2023

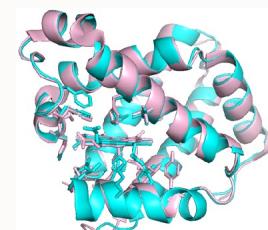
AI Drug Discovery



design drug molecules for disease target



design antimicrobial peptides



design proteins with bio-chemical functions

heme-binding myoglobins

Xie et al. "Markov Molecular Sampling for Multi-objective Drug Discovery", ICLR (2021).
Song & Li. "Importance Weighted Expectation-Maximization for Protein Sequence Design".

Self-assisting and Cooperative Large Language Models

Lei Li (leili@cs.cmu.edu)

Generate
Algorithmic
Programs

ALGO



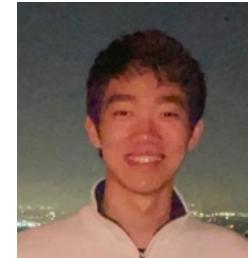
Kexun Zhang



Danqing Wang

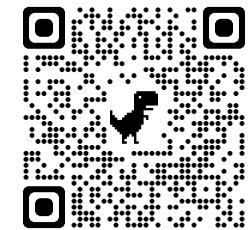
Accurate
Tool-using LLM

ToolDec



Hongqiao Chen

Cooperative
LLMs to learn
from mistakes
SALAM



Jingtao Xia



William Yang Wang