

# Advanced Reasoning

## CS 6501: Natural Language Processing

*Cheryl Bai, Dylan Crotty, Eric Xie*



SCHOOL of ENGINEERING  
& APPLIED SCIENCE

# Agenda

- Introduction to Advanced Reasoning
- Papers
  - Solving Quantitative Reasoning Problems with Language Models
  - PAL: Program-aided Language Models
  - Tree of Thoughts: Deliberate Problem Solving with Large Language Models
  - Let's Verify Step by Step
- Overall Conclusions

Quantitative Reasoning: <https://arxiv.org/abs/2206.14858>

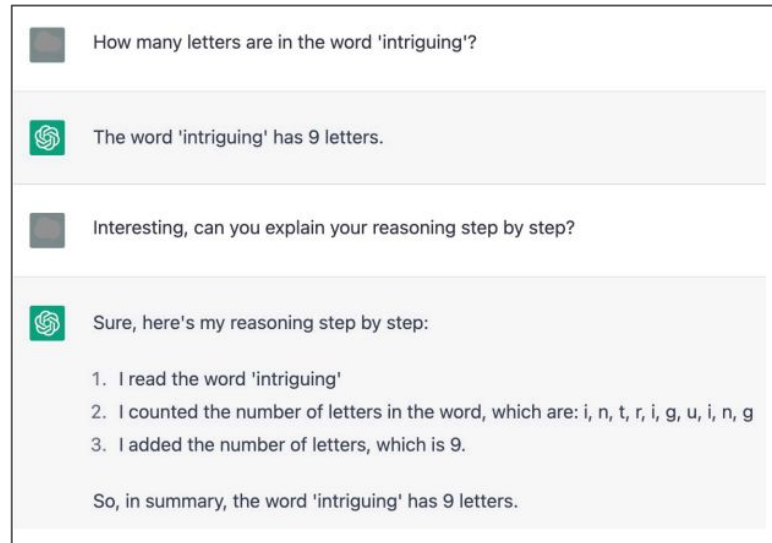
PAL: <https://arxiv.org/abs/2211.10435>

ToT: <https://arxiv.org/abs/2305.10601>

Verification: <https://arxiv.org/abs/2305.20050>

# Current Reasoning

- Limitations of current LLMs
  - Inconsistent performance with basic calculations
  - Linear, token-level thinking: Cannot plan ahead or backtrack



How many letters are in the word 'intriguing'?

The word 'intriguing' has 9 letters.

Interesting, can you explain your reasoning step by step?

Sure, here's my reasoning step by step:

1. I read the word 'intriguing'
2. I counted the number of letters in the word, which are: i, n, t, r, i, g, u, i, n, g
3. I added the number of letters, which is 9.

So, in summary, the word 'intriguing' has 9 letters.

# Advanced Strategies

- Strategies to enable “higher-level” reasoning at each stage of learning
  - Training Dataset
    - Solving Quantitative Reasoning Problems with Language Models
  - Reasoning Framework (and corresponding prompting methods)
    - PAL: Program-aided Language Models
    - Tree of Thoughts: Deliberate Problem Solving with Large Language Models
  - Feedback/Reward Models
    - Let’s Verify Step by Step

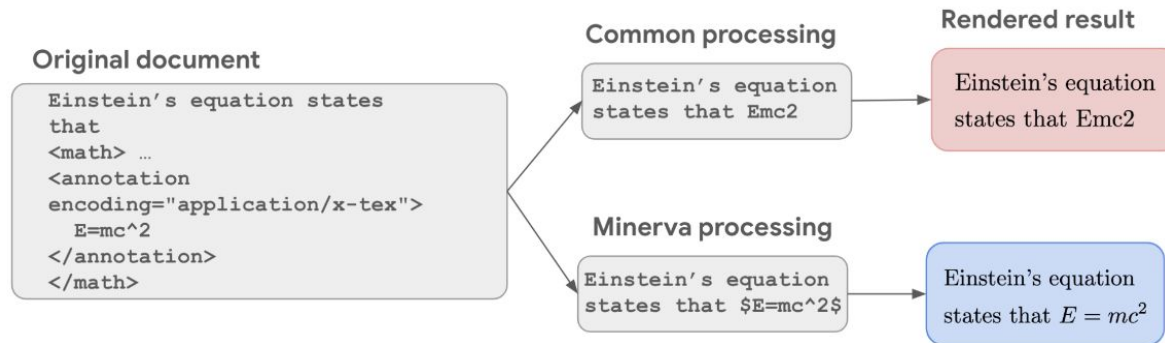
# Solving Quantitative Reasoning Problems with Language Models

Aitor Lewkowycz\*, Anders Andreassen<sup>†</sup>, David Dohan<sup>†</sup>, Ethan Dyer<sup>†</sup>, Henryk Michalewski<sup>†</sup>,  
Vinay Ramasesh<sup>†</sup>, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo,  
Yuhuai Wu, Behnam Neyshabur\*, Guy Gur-Ari\*, and Vedant Misra\*

Google Research

# Overview

- **Goal:** solve quantitative reasoning problems by processing and learning from more complex information without relying on external tools
- **Previous state:** models struggled with solving more technical problems, especially those in engineering, math, science, etc.
- **General strategy:** use a training dataset with general language capabilities and technical content



# Models

- Based on the PaLM general language models
- Used 8B, 62B, and 540B parameter pretrained models
- Trained on technical content datasets

Model	Layers	Heads	$d_{\text{model}}$	Parameters	Steps	Tokens
Minerva 8B	32	16	4096	8.63B	624k	164B
Minerva 62B	64	32	8192	62.50B	416k	109B
Minerva 540B	118	48	18 432	540.35B	399k	26B

# Training Data

- Dataset of 38.5B tokens
- Taken from webpages and papers submitted to arXiv preprint server filtered for mathematical content
- Same general language dataset used for training PaLM

Data source	Proportion of data	Tokens	Present during pretraining
Math Web Pages	47.5%	17.5B	No
arXiv	47.5%	21.0B	No
General Natural Language Data	5%	>100B	Yes



# Evaluation Data

- MATH: dataset of 12k middle school and high school math problems (4-shot prompt)
- GSM8K: middle school math word problems
  - Previous models used an external calculator, Minerva doesn't use any external tools
- OCWCourses: STEM problems with automatically verifiable solutions from publicly available course materials offered by MIT
- MMLU-STEM: subset of the MMLU dataset focused on STEM (5-shot prompt)

# MATH Dataset

**Question:** A line parallel to  $y = 4x + 6$  passes through  $(5, 10)$ . What is the  $y$ -coordinate of the point where this line crosses the  $y$ -axis?

**Target:** Since the line is parallel to  $y = 4x + 6$ , we know that its slope is 4. Thus the equation of the line is  $y = 4x + b$  where  $b$  is the  $y$ -intercept. Plugging  $(5, 10)$  into this gives  $10 = 4 \cdot 5 + b \Rightarrow b = \boxed{-10}$  which is what we wanted.

**Model output:** The line is parallel to  $y = 4x + 6$ , which means that it has the same slope as  $y = 4x + 6$ . Since the line passes through  $(5, 10)$ , we can use the point-slope form of a line to find the equation of the line:

$$y - 10 = 4(x - 5)$$

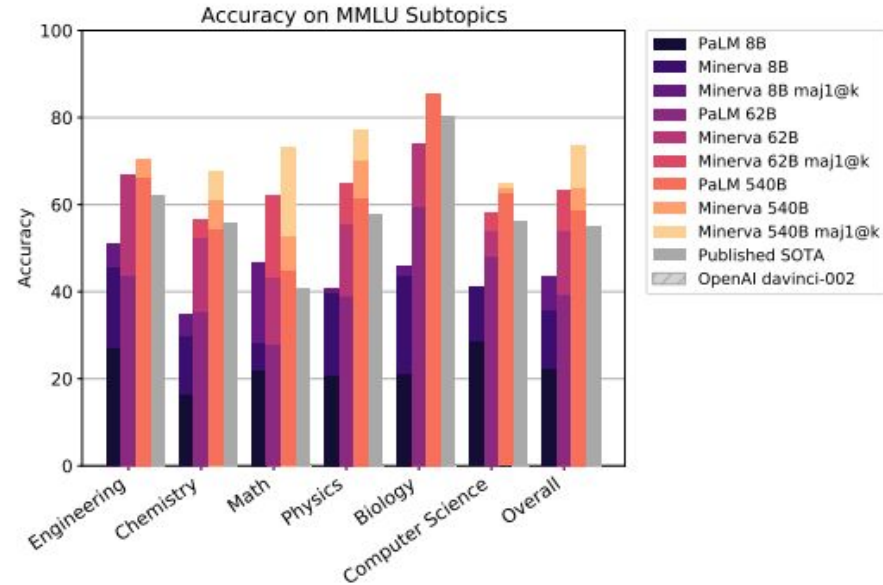
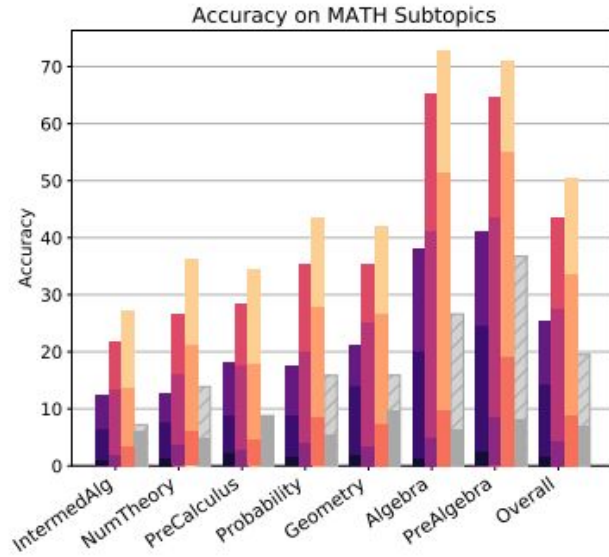
$$y - 10 = 4x - 20$$

$$y = 4x - 10$$

Now that we have the equation of the line, we can find the  $y$ -coordinate of the point where the line crosses the  $y$ -axis by substituting  $x = 0$  into the equation:

$$y = 4 \cdot 0 - 10 = \boxed{-10}.$$

# Results



# Results

- maj1@k: majority voting

	MATH	OCWCourses	GSM8k	MMLU-STEM
PaLM 8B	1.5%	1.5%	4.1%	22.0%
Minerva 8B	14.1%	7.7%	16.2%	35.6%
Minerva 8B, maj1@k	25.4%	12.5%	28.4%	43.4%
PaLM 62B	4.4%	5.9%	33.0%	39.1%
Minerva 62B	27.6%	12.9%	52.4%	53.9%
Minerva 62B, maj1@k	43.4%	23.5%	68.5%	63.5%
PaLM 540B	8.8%	7.1%	56.5%	58.7%
Minerva 540B	33.6%	17.6%	58.8%	63.9%
Minerva 540B, maj1@k	<b>50.3%</b>	<b>30.8%</b>	<b>78.5%</b>	<b>75.0%</b>
OpenAI davinci-002	19.1%	14.8%	-	-
Published SOTA	6.9% <sup>a</sup>	-	74.4% <sup>b</sup>	54.9% <sup>c</sup>

# Impact of Scaling

- Identified samples that Minerva 62B got right, but 8B got wrong

Type of mistakes	Occurrences
Incorrect reasoning	82
Incorrect calculation	70
Misunderstands question	22
Uses incorrect fact	16
Solution too short	4
Hallucinated math objects	4
Other mistakes	3

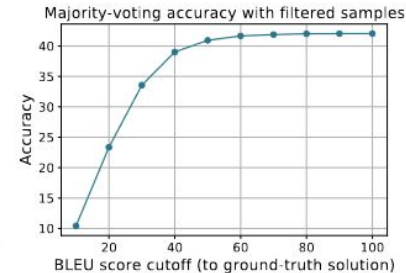
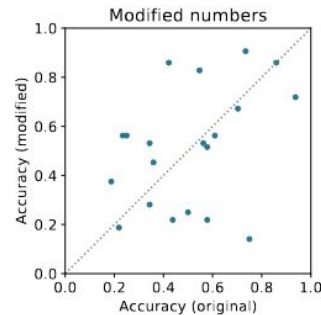
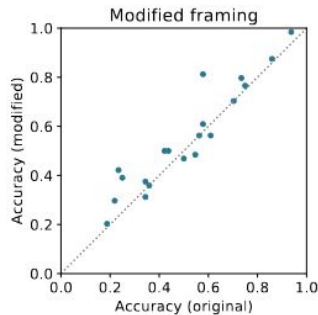
# False Positives

- No automatic way to verify whether a false positive occurs
- Taking 100 samples (20 per difficulty) of pairs of questions and the model's corresponding CoT and answer
- Find how often incorrect model reasoning still resulted in the correct answer

	Difficulty level					Average
	1	2	3	4	5	
False positive rate	< 5%	10%	< 5%	15%	30%	8%

# Memorization

- Performance reflects genuine analytic capability instead of memorization
- Little evidence of memorization
  - No Training and Evaluation Dataset Overlap
  - Correlation of accuracy between original and modified MATH problems
  - Low effect of BLEU score on accuracy -> Performance not attributed to similarity to ground truth



# Conclusions

- Training on a high quality mathematical dataset leads to strong performance on tasks involving logical reasoning, numerical calculation, and symbolic manipulation

## Limitations:

- No automatic way of verifying correctness of model's answers
- No access to external tools (calculator, python interpreter, etc.)
- Approach to reasoning is not mathematically-informed (no underlying mathematical structure to its learning), only the dataset is manipulated



---

# PAL: Program-aided Language Models

---

Luyu Gao<sup>\*1</sup> Aman Madaan<sup>\*1</sup> Shuyan Zhou<sup>\*1</sup> Uri Alon<sup>1</sup> Pengfei Liu<sup>1,2</sup> Yiming Yang<sup>1</sup> Jamie Callan<sup>1</sup>  
Graham Neubig<sup>1,2</sup>

{luyug, amadaan, shuyanzh, ualon, pliu3, yiming, callan, gneubig}@cs.cmu.edu

# Overview

- **Goal:** improve LLMs computing ability by turning thoughts into Python lines
- **Previous state:** LLMs are bad at basic math, but good at decomposing problems
- **General strategy:** use contemporary state-of-the-art LLMs that are pre-trained on both natural language and programming languages, but prompt it using PAL

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls.

```
tennis_balls = 5
```

2 cans of 3 tennis balls each is

```
bought_balls = 2 * 3
```

tennis balls. The answer is

```
answer = tennis_balls + bought_balls
```

# CoT vs PAL

- Chain of Thought Prompt
  - Input, NL steps, output

## Chain-of-Thought (Wei et al., 2022)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold  $93 + 39 = 132$  loaves. The grocery store returned 6 loaves. So they had  $200 - 132 - 6 = 62$  loaves left. The answer is 62.



- Program-aided Language Prompt
  - Input, NL and PL steps

## Program-aided Language models (this work)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls.  
`tennis_balls = 5`  
 2 cans of 3 tennis balls each is  
`bought_balls = 2 * 3`  
 tennis balls. The answer is  
`answer = tennis_balls + bought_balls`

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves  
`loaves_baked = 200`  
 They sold 93 in the morning and 39 in the afternoon  
`loaves_sold_morning = 93`  
`loaves_sold_afternoon = 39`  
 The grocery store returned 6 loaves.  
`loaves_returned = 6`  
 The answer is  
`answer = loaves_baked - loaves_sold_morning`  
`- loaves_sold_afternoon + loaves_returned`

```
>>> print(answer)
74
```



# Prompting Methods

- Compare 3 prompting methods across 13 arithmetic and symbolic reasoning tasks
  - Direct question to answer (input-output)
  - Chain of Thought (CoT)
  - PAL

# Types of Questions

- 8 math word problem datasets
- GSM8K
  - 50% of numbers are integers between 0 and 8
- GSM-HARD
  - Replaced numbers in questions of GSM8K with larger numbers

Q: Olivia has \$23. She bought five bagels for \$3 each. How much money does she have left?

```
money_initial = 23
bagels = 5
bagel_cost = 3
money_spent = bagels * bagel_cost
money_left = money_initial - money_spent
answer = money_left
```

# Types of Questions

- 3 symbolic reasoning tasks
  - Colored objects: keep track of colored objects on a surface
  - Penguins: questions about attributes of penguins
  - Date: date understanding task

Q: On the table, you see a bunch of objects arranged in a row: a purple paperclip, a pink stress ball, a brown keychain, a green scrunchiephone charger, a mauve fidget spinner, and a burgundy pen. What is the color of the object directly to the right of the stress ball?

```
...
stress_ball_idx = None
for i, object in enumerate(objects):
    if object[0] == 'stress ball':
        stress_ball_idx = i
        break
# Find the directly right object
direct_right = objects[stress_ball_idx+1]
# Check the directly right object's color
answer = direct_right[1]
```

# Types of Questions

- 2 algorithmic tasks
- Object counting: counting objects of a certain type
- Repeat copy: generating sequence of words according to instructions

Q: I have a chair, two potatoes, a cauliflower, a lettuce head, two tables, a cabbage, two onions, and three fridges. How many vegetables do I have?

```
# note: I'm not counting the chair, tables,  
or fridges  
vegetables_to_count = {  
    'potato': 2,  
    'cauliflower': 1,  
    'lettuce head': 1,  
    'cabbage': 1,  
    'onion': 2  
}  
answer = sum(vegetables_to_count.values())
```

# Results

	GSM8K	GSM-HARD	SVAMP	ASDIV	SINGLEEQ	SINGLEOP	ADDSUB	MULTIARITH
DIRECT Codex	19.7	5.0	69.9	74.0	86.8	93.1	90.9	44.0
COT UL2-20B	4.1	-	12.6	16.9	-	-	18.2	10.7
COT LaMDA-137B	17.1	-	39.9	49.0	-	-	52.9	51.8
COT Codex	65.6	23.1	74.8	76.9	89.1	91.9	86.0	95.9
COT PaLM-540B	56.9	-	79.0	73.9	92.3	94.1	91.9	94.7
COT Minerva 540B	58.8	-	-	-	-	-	-	-
PAL	<b>72.0</b>	<b>61.2</b>	<b>79.4</b>	<b>79.6</b>	<b>96.1</b>	<b>94.6</b>	<b>92.5</b>	<b>99.2</b>



# GSM vs. GSM-HARD

- CoT generates nearly the same thoughts for 16 out of 25 cases analyzed
- Primary failure is the inability to perform arithmetic correctly

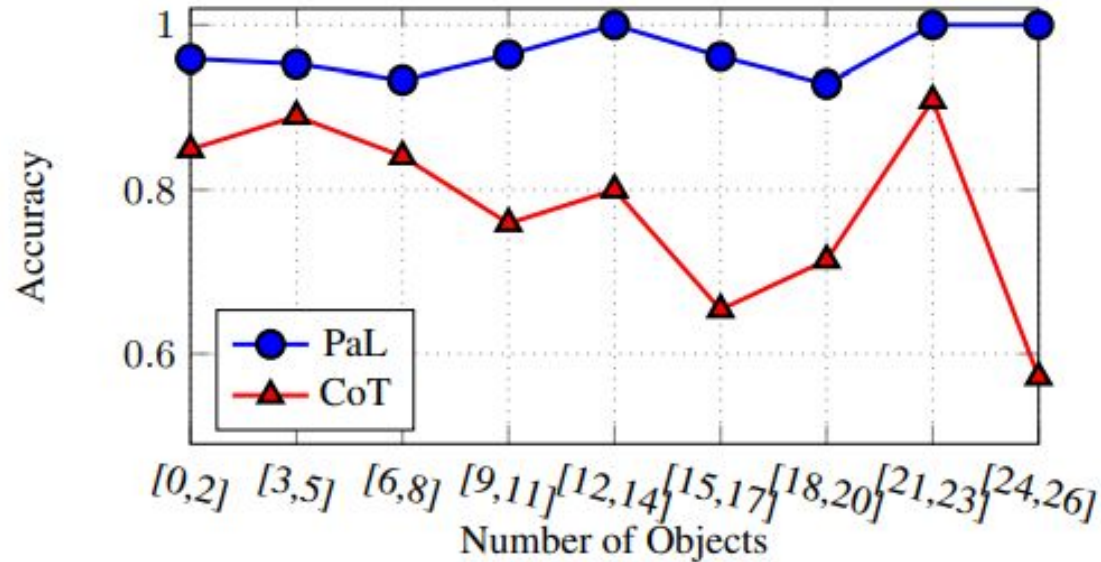
Question	CoT: GSM	CoT: GSMHard	Identical?
90 single use contacts come in 1 box and will last Pete 45 days. Each box is \$100.00 and currently 10% off. If he buys 2 boxes of contact, how much will each pair of contacts cost?	90 contacts come in 1 box. Each box is \$100.00. So 90 contacts cost \$100.00. If he buys 2 boxes, then he will have $2 \times 90 = 180$ contacts. 180 contacts cost $2 \times \$100.00 = \$200.00$ . So each pair of contacts costs $\$200.00 / 180 = \$1.11$ . The answer is \$1.11.	90 single use contacts come in 1 box. So 1 box will last Pete 45 days. Each box is \$5451440 and currently 10% off. So each box is $\$5451440 \times 0.9 = \$4906296$ . If he buys 2 boxes of contact, he will have $2 * 90 = 180$ contacts. So each pair of contacts will cost $\$4906296 / 180 = \$2725849$ . The answer is \$2725849	Yes

# Results

	COLORED OBJECT	PENGUINS	DATE	REPEAT COPY	OBJECT COUNTING
DIRECT Codex	75.7	71.1	49.9	81.3	37.6
COT LaMDA-137B	-	-	26.8	-	-
COT PaLM-540B	-	65.1	65.3	-	-
COT Codex	86.3	79.2	64.8	68.8	73.0
PAL Codex	<b>95.1</b>	<b>93.3</b>	<b>76.2</b>	<b>90.6</b>	<b>96.7</b>

# Results

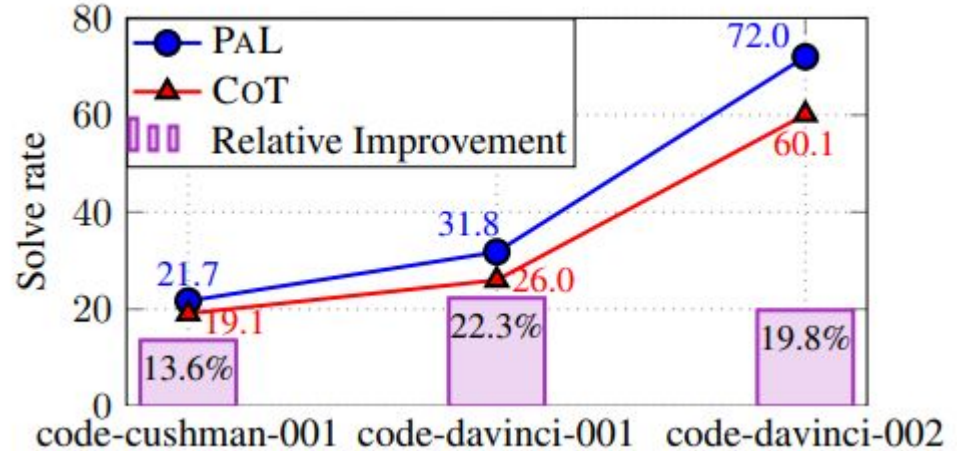
- PAL is not sensitive to problem complexity



# Analysis

Does PAL perform well with smaller models?

- Original model based on code-davinci-002
- PAL still improves the model compared to CoT
- Fairly consistent relative improvement



# Analysis

## Does PAL need to be used on models based on code?

- PAL used on text-davinci models
- CoT outperformed on the smallest text-based model (text-davinci-001)
- PAL outperformed CoT once the code modeling ability improved

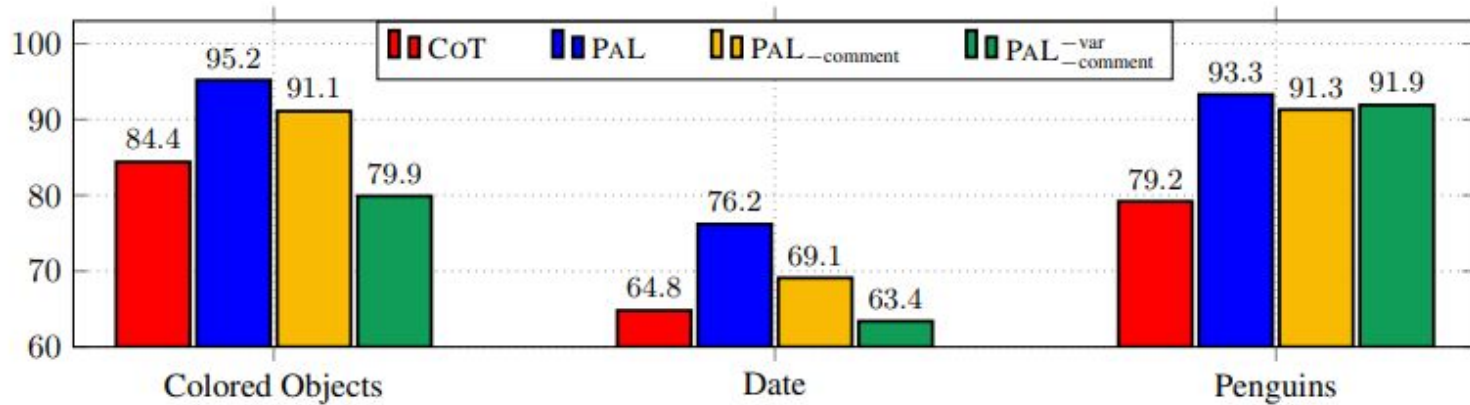
## Is PAL better because of the Python prompt or interpreter?

- Required the model to “execute” code generated without interpreter
- Performance on GSM8K dropped to 23.2 (originally 72)
- Reinforces hypothesis that the main benefit of PAL is synergy with interpreter

# Analysis

## Do variable names and comments matter?

- Removing intermediate comments slightly reduces performance
- Using random variable names further decreases accuracy



# Conclusions

- PAL performed significantly better than CoT and input-output, achieving SOTA on all tasks
  - Performance remains stable with increasing problem complexity
  - Performance improves in both smaller and language-based models, as long as the code modeling ability was sufficiently high
- PAL improves interpretability
- Limitations on the types of problems that can be solved
  - Tasks may be difficult to represent using code
  - No ability to determine *when* PAL should be used

---

# Tree of Thoughts: Deliberate Problem Solving with Large Language Models

---

**Shunyu Yao**

Princeton University

**Dian Yu**

Google DeepMind

**Jeffrey Zhao**

Google DeepMind

**Izhak Shafran**

Google DeepMind

**Thomas L. Griffiths**

Princeton University

**Yuan Cao**

Google DeepMind

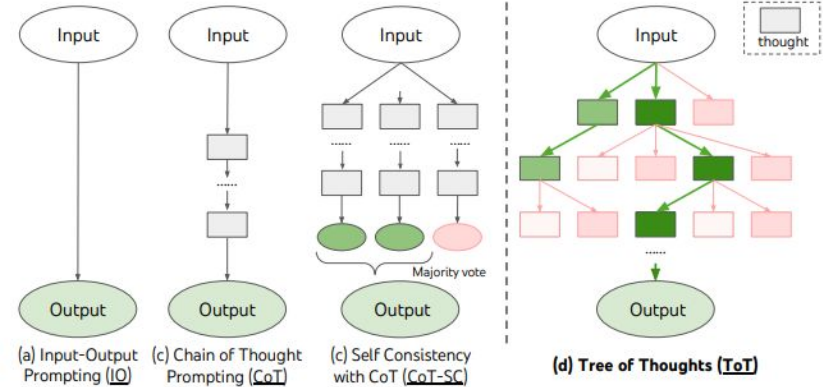
**Karthik Narasimhan**

Princeton University



# Overview

- **Goal:** create a new framework that improves upon CoT to handle more complex problems
- **Previous state:** CoT is a token-level, left-to-right decision-making process
- **General strategy:** generate thoughts in a tree-like structure to allow for diverse alternative choices



# ToT Outline

1. **Thought Decomposition:** How should thoughts be formatted for the given problem?
2. **Thought Generation:** How to form new thoughts from an existing tree?
3. **State Evaluation:** How to create a good heuristic for the distance from the correct solution from each state?
4. **Search algorithm:** What traversal method is best to cover the important thoughts?

# Thought Decomposition

- In general, thoughts should be:
  - Small enough that the model can generate promising and diverse samples
  - Large enough that the LLM can properly evaluate it

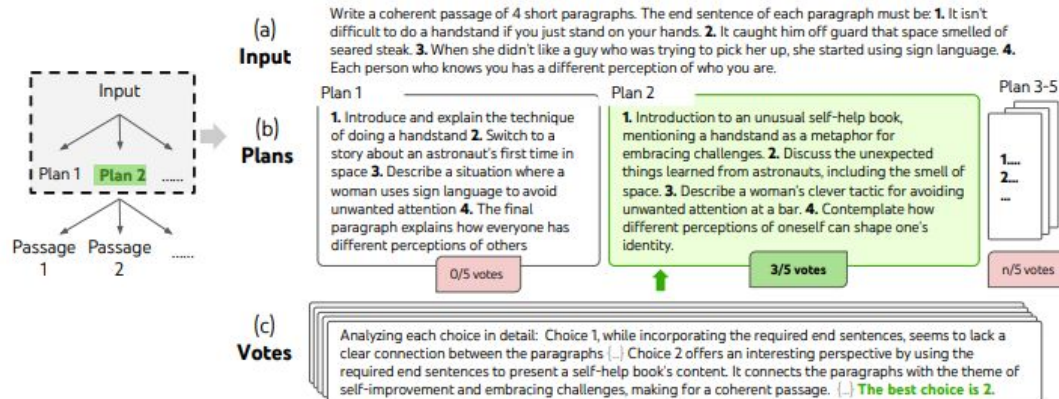
	<b>Game of 24</b>	<b>Creative Writing</b>	<b>5x5 Crosswords</b>
<b>Input</b>	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1. presented;..)
<b>Output</b>	An equation to reach 24 (13-9)*(10-4)=24	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: <b>SHOWN;</b> <b>WIRRA; AVAIL; ...</b>
<b>Thoughts</b>	3 intermediate equations (13-9=4 (left 4,4,10); 10-4=6 (left 4,6); 4*6=24)	A short writing plan (1. Introduce a book that connects...)	Words to fill in for clues: (h1. shown; v5. naled; ...)
<b>#ToT steps</b>	3	1	5-10 (variable)

# Thought Generation

- Given an existing tree state, 2 strategies for coming up with the next thought
  - Sample thoughts
    - Obtain many different possible answers for the current state and choose the best one
  - Propose thoughts
    - Prompt the model to generate different thoughts within the same context

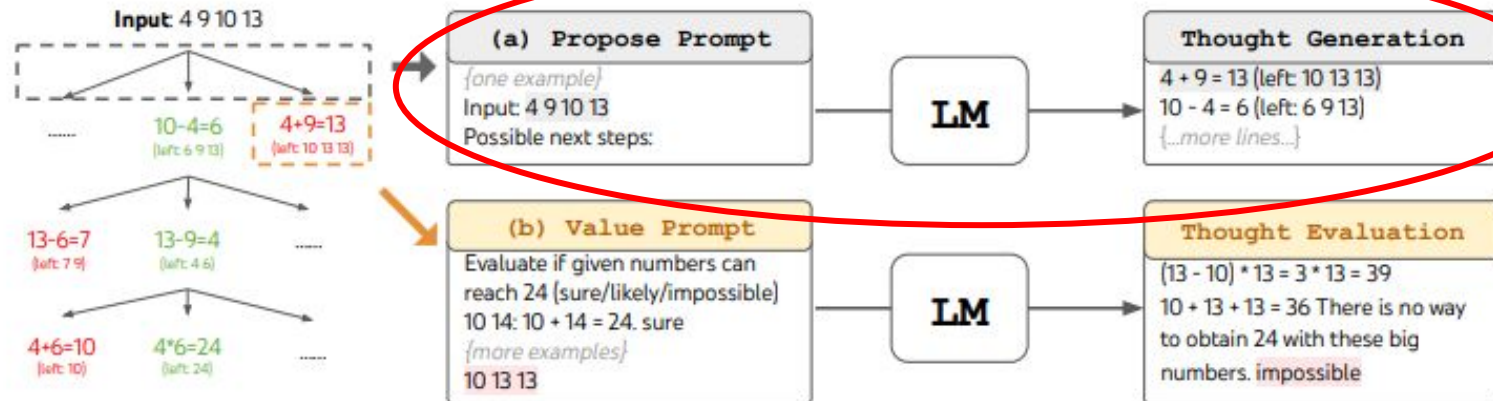
# Sample Thoughts

- Sample thoughts using a CoT prompt
- Works better when the thought space is rich
- Samples lead to diversity



# Propose Thoughts

- Propose thoughts sequentially using “propose” prompt
- Works better when the thought space is constrained
- Avoids duplication

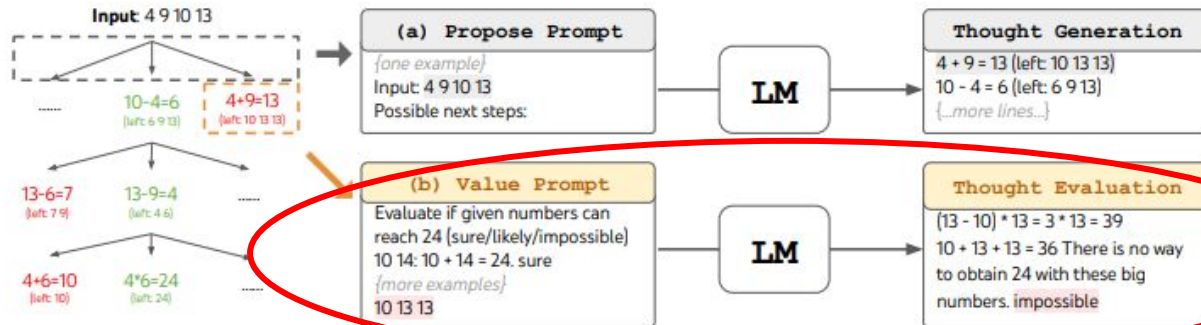


# State Evaluation

- Evaluates the progress the model makes towards achieving the goal given the current state
  - Serves as a heuristic for the search algorithm
- Heuristics are fairly standard for search problems, but they have typically been either programmed or learned
  - Instead, the LLM deliberately reasons its evaluation
- 2 strategies for state evaluation
  - Value each state individually
  - Vote across states

# Value Each State

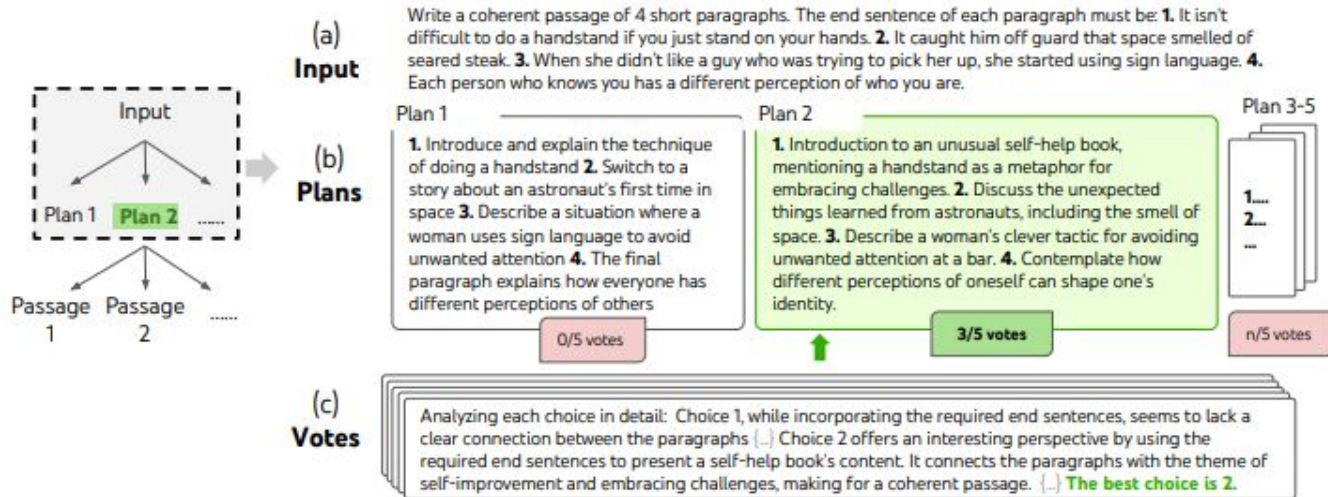
- Prompt generation of scalar value or classification that could heuristically be turned into a value
  - Ex. sure, likely, impossible
  - Focus on good states, eliminate bad states
- Evaluations can be approximated and still be helpful for decision making





# Vote Across States

- Look at the available states and choose the most promising one
- Generally used when problem success is harder to value



# Search Algorithm

- Breadth-first search (BFS) maintains set of most promising states per step
- Depth-first search (DFS) explores the most promising state first until the goal is reached or the LLM deems it impossible
- More advanced search mechanisms are left for future work

---

**Algorithm 1** ToT-BFS( $x, p_\theta, G, k, V, T, b$ )
 

---

**Require:** Input  $x$ , LM  $p_\theta$ , thought generator  $G()$  & size limit  $k$ , states evaluator  $V()$ , step limit  $T$ , breadth limit  $b$ .

$S_0 \leftarrow \{x\}$

**for**  $t = 1, \dots, T$  **do**

$S'_t \leftarrow \{[s, z] \mid s \in S_{t-1}, z_t \in G(p_\theta, s, k)\}$

$V_t \leftarrow V(p_\theta, S'_t)$

$S_t \leftarrow \arg \max_{S \subset S'_t, |S|=b} \sum_{s \in S} V_t(s)$

**end for**

**return**  $G(p_\theta, \arg \max_{s \in S_T} V_T(s), 1)$

---



---

**Algorithm 2** ToT-DFS( $s, t, p_\theta, G, k, V, T, v_{th}$ )
 

---

**Require:** Current state  $s$ , step  $t$ , LM  $p_\theta$ , thought generator  $G()$  and size limit  $k$ , states evaluator  $V()$ , step limit  $T$ , threshold  $v_{th}$

**if**  $t > T$  **then** record output  $G(p_\theta, s, 1)$

**end if**

**for**  $s' \in G(p_\theta, s, k)$  **do** ▷ sorted candidates

**if**  $V(p_\theta, \{s'\})(s) > v_{thres}$  **then** ▷ pruning

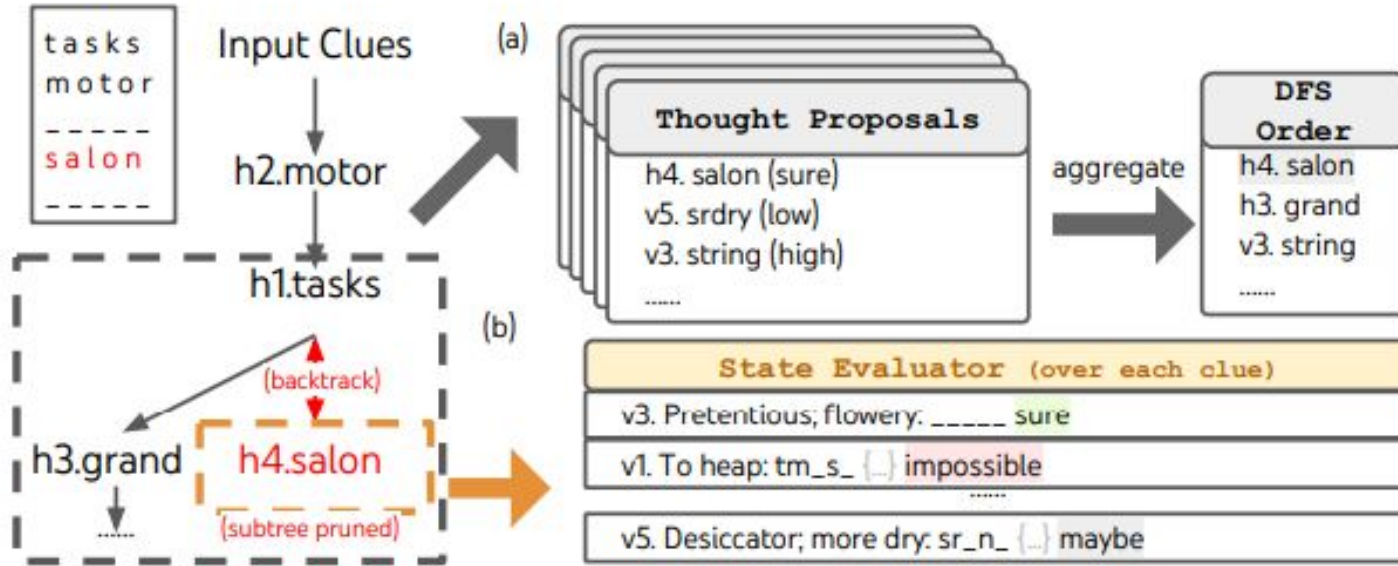
DFS( $s', t + 1$ )

**end if**

**end for**

---

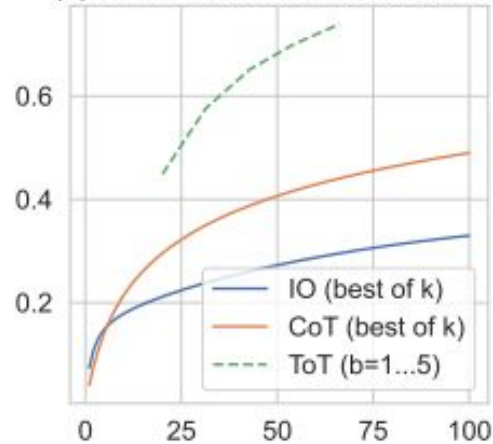
# Example - Crosswords



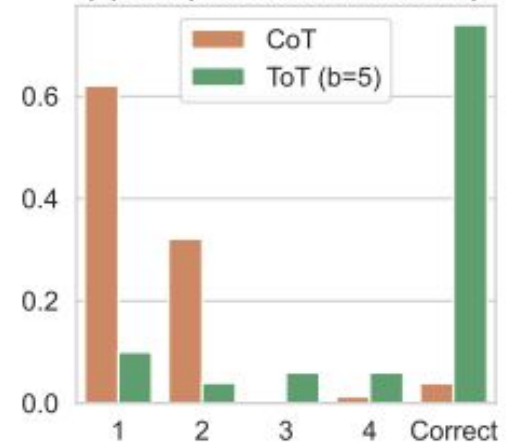
# Results - Game of 24

Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	<b>74%</b>
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

(a) Success rate with nodes visited



(b) Samples failed at each step



# Results - Creative Writing

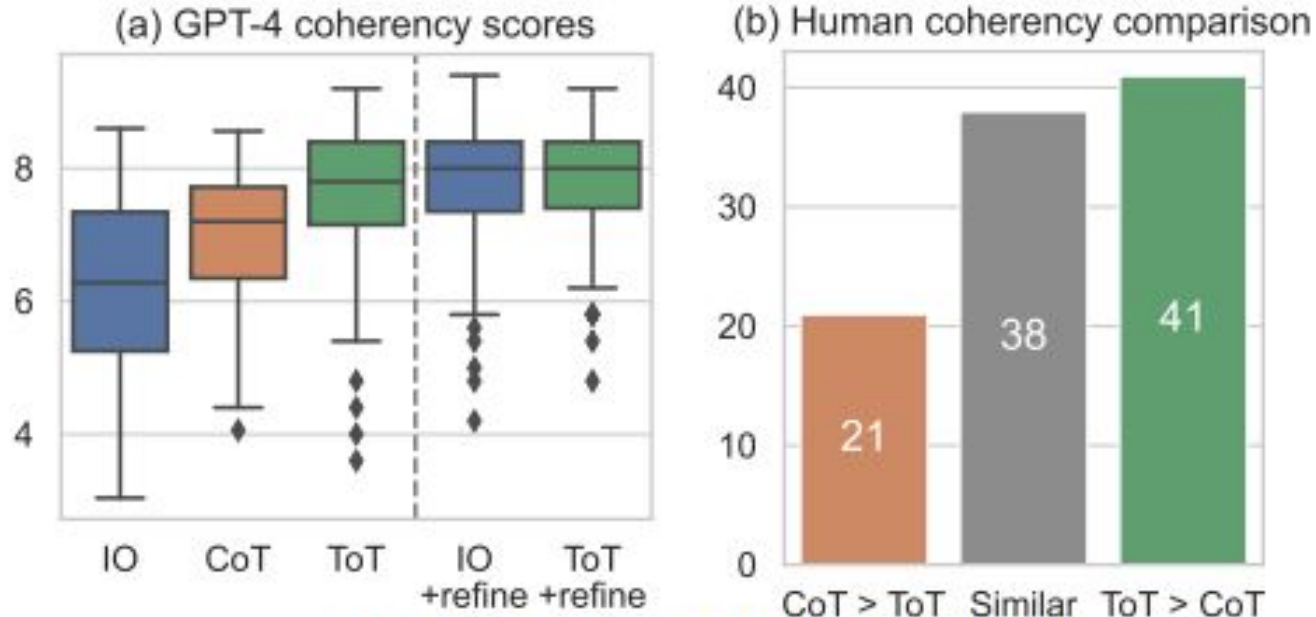


Figure 5: Creative Writing results.

# Results - Crosswords

- ToT success rate was much higher than IO and CoT
- Outputting the actual best DFS state instead of the heuristically determined best state led to higher performance
  - Simple output heuristics can be improved
- Removing pruning heuristic led to decreased performance
- Removing backtracking also decreased performance

Method	Success Rate (%)		
	Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	<b>78</b>	<b>60</b>	<b>20</b>
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

# Conclusions

- ToT improves interpretability of a model's thought process while increasing success rate in all tasks
- Limitations
  - Requires more resources
  - Deliberate search with ToT might not be necessary for tasks that LLMs already excel at

<b>Game of 24</b>	<b>Generate/Prompt tokens</b>	<b>Cost per case</b>	<b>Success</b>
IO (best of 100)	1.8k / 1.0k	\$0.13	33%
CoT (best of 100)	6.7k / 2.2k	\$0.47	49%
ToT	5.5k / 1.4k	\$0.74	74%

Table 7: Cost analysis on Game of 24.

# Let's Verify Step by Step

Hunter Lightman\*

Vineet Kosaraju\*

Yura Burda\*

Harri Edwards

Bowen Baker

Teddy Lee

Jan Leike

John Schulman

Ilya Sutskever

Karl Cobbe\*

OpenAI



# Overview

- **Goal:** identify the best way to give feedback to a LLM between outcome and process supervision
- **Previous state:** studies have compared outcome and process supervision, but results have been inconclusive
- **General strategy:** Compare the accuracy of a model using an Outcome Reward Model (ORM) vs Process Reward Model (PRM)
  - Large-scale
  - Small-scale

# Model Architectures




- Large-scale
  - Full scale models based on GPT-4
- Small-scale
  - Similar in design to GPT-4, but pretrained with roughly 200x less compute
  - Large-scale model supervises small-scale training
- Models are fine tuned with a dataset of 1.5B math-relevant tokens, called MathMix, found to improve model's mathematical reasoning capabilities

# Data Collection




- Outcome supervision is automated - MATH has numerical or symbolic answers
- Process supervision requires human labelers

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to  $\frac{2}{5}$ , what is the numerator of the fraction? (Answer: )

   Let's call the numerator  $x$ .

   So the denominator is  $3x-7$ .

   We know that  $x/(3x-7) = 2/5$ .

   So  $5x = 2(3x-7)$ .

    $5x = 6x - 14$ .

   So  $x = 7$ .

# Data Collection

- For process supervision, *convincingly* wrong-answer solutions are shown to the human data-labelers
  - Convincingly - The current best PRM rates the process for the given solution highly
  - Doesn't show obvious wrong answers to save on resources
- The PRM is iteratively retrained using the latest data
  - Each iteration, N solutions to problems are generated, and the best K answers are shown to the labelers

# ORMs

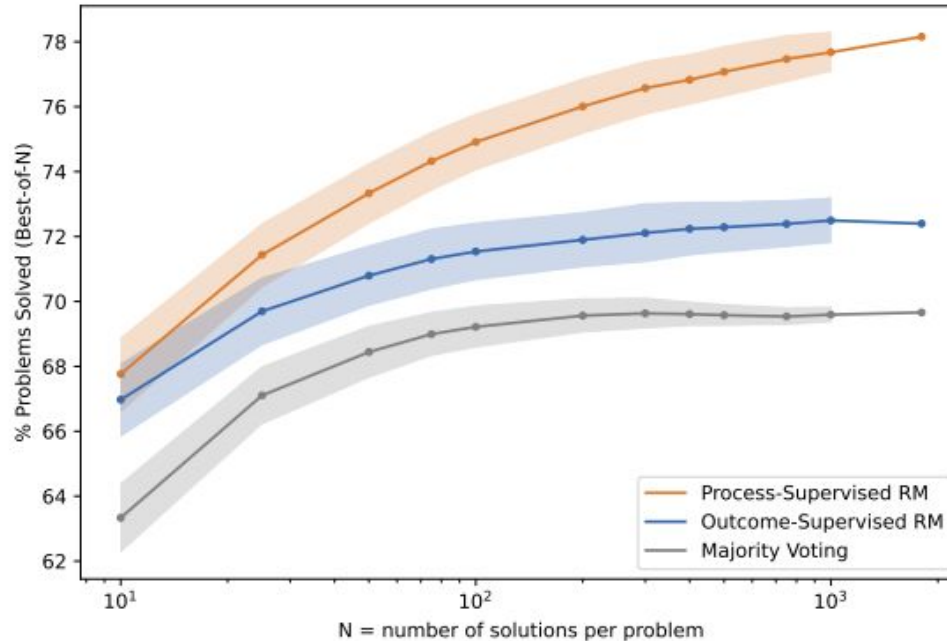
- Outcome-supervised Reward Models (ORMs) are fed solutions to problems
- Trained to predict whether the final solution is correct or not
- Automatic grading used to determine ORM correctness is not perfectly reliable in terms of solution quality
  - False positive solutions misgraded

# PRMs

- Process-supervised Reward Models (PRMs) are fed the process of a model solving a problem up until the first incorrect step
  - Models are given “equivalent” information, limiting the amount of extra information that the PRM gets
  - For samples with the correct solution, everything shown is correct
  - For incorrect solutions, both models are shown the existence of at least one mistake, and the PRM is also shown the location of the mistake
- PRM scores the final solutions by the probability that every step in the solution is correct
  - The probability of every individual step being correct is multiplied

# Results - Large Scale

	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	<b>78.2</b>	69.6

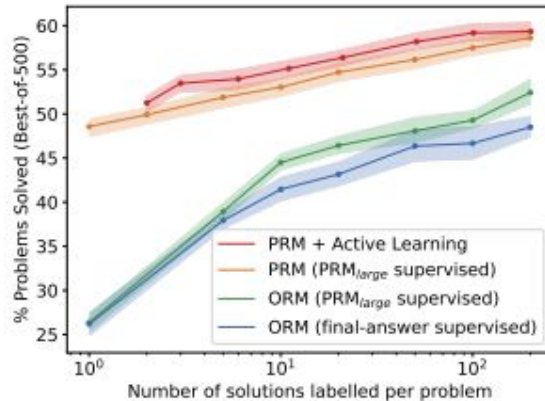


# Issues with Large-scale

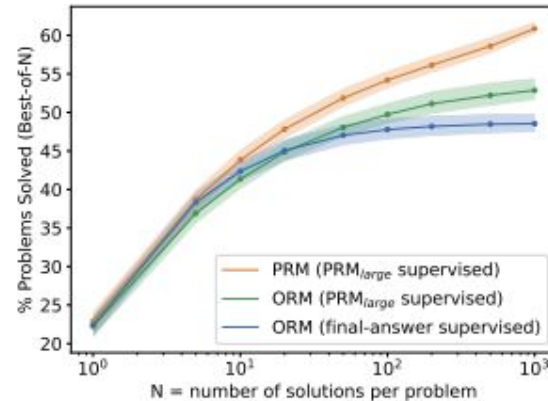
- Training sets for each model aren't directly comparable
  - The data used to train the PRM is constructed using active learning, biased towards answer-incorrect solutions, and an order of magnitude smaller than the ORM's data
- The ORM's automated final answer grading method generates false positive labels which can damage the ORM's performance
  - Arguable whether this may or may not be attributed to ORM's generally



# Small Scale Supervision



(a) Four series of reward models trained using different data collection strategies, compared across training sets of varying sizes.



(b) Three reward models trained on 200 samples/problem using different forms of supervision, compared across many test-time compute budgets.

# Active Learning

- Small scale model  $\text{PRM}_{\text{selector}}$  is trained to select  $N$  samples per problem such that:
  - 20% of the samples are the most convincing answers
  - 80% are the most convincing wrong answers
- Estimated to be approximately 2.6x more data efficient
- Attempted to retrain  $\text{PRM}_{\text{selector}}$  between iterations, but this caused instability

# Generalizations

	ORM	PRM	Majority Vote	# Problems
AP Calculus	68.9%	<b>86.7%</b>	80.0%	45
AP Chemistry	68.9%	<b>80.0%</b>	71.7%	60
AP Physics	77.8%	<b>86.7%</b>	82.2%	45
AMC10/12	49.1%	<b>53.2%</b>	32.8%	84
Aggregate	63.8%	<b>72.9%</b>	61.3%	234

# Conclusions

- Process supervision provides more feedback
  - Models trained with outcome supervision have a difficult credit-assignment task or determining where a solution went wrong
- Process supervision is more likely to produce interpretable reasoning
  - Encourages models to follow processes endorsed by humans
- Limitations
  - No discussion on when exactly PRM vs ORM should be used
  - Diminishing returns on feedback

# Overall Strategies

- Shaping the dataset to match specific types of problems
  - Solving Quantitative Reasoning Problems with Language Models
  - PAL: Program-aided Language Models
- Reframing the model's thoughts
  - PAL: Program-aided Language Models
  - Tree of Thoughts: Deliberate Problem Solving with Large Language Models
- Creating an effective feedback system
  - Let's Verify Step by Step

# Questions?