Natural Language Processing

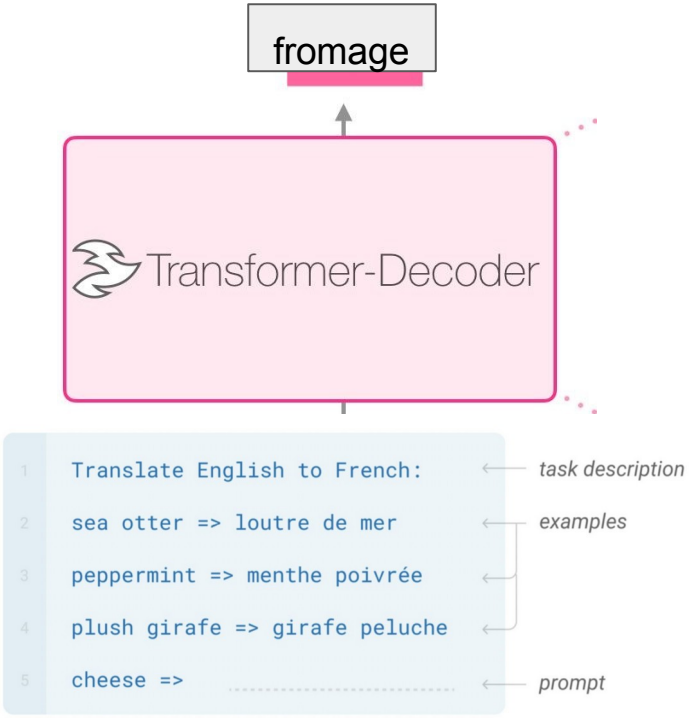
Prompting

Instructor: Yangqiu Song

Slides credits: Jesse Mu, Jason Wei, Danqi Chen, Chris Pan and Hongjie Wang, Wenhu Chen, Linda Zeng and Xiang Li

Background: In-context Learning

- Pretrain a language model on task (LM)
- Manually design a “prompt” that demonstrates how to formulate a task as a generation task.
- **No need to update the model weights at all!**



Brown et al. 2020

New methods of “prompting” LMs

Zero/few-shot prompting

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

Traditional fine-tuning

```
sea otter => loutre de mer
```

gradient update

```
peppermint => menthe poivrée
```

gradient update

```
cheese =>
```

[Brown et al., 2020]

Chain-of-Thought Prompting

Chain-of-Thought Prompting

- Introduces chain-of-thought prompting as a novel method.
- Significantly improves LLMs' complex reasoning tasks performance.
  - Arithmetic reasoning
  - Commonsense reasoning
  - Symbolic reasoning

Traditional Prompting vs Chain-of-Thought

- Traditional Prompting
  - In-context few-shot learners
  - Works poorly on tasks that require reasoning abilities
  - Not improving with increasing language model scale

- Chain-of-thought
  - Series of intermediate reasoning steps
  - Mimic human step-by-step thought

Comparision

Standard Prompting

Model 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: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model 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 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

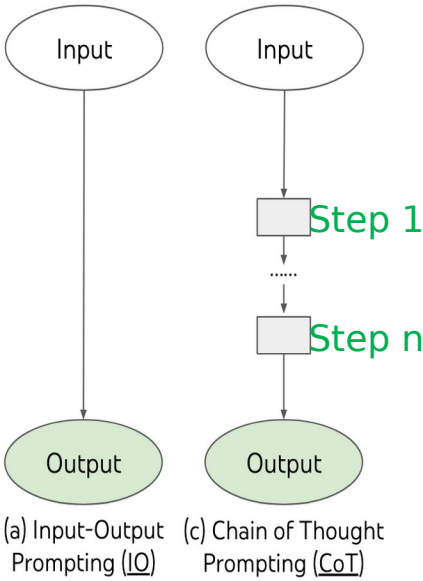
Chain-of-thought Prompting Principals

- Step-by-Step Guidance - Breaking complex problems into smaller steps
  - “Let’s do it step by step.”

- Explicit Reasoning Chains - Creating prompts that detail the reasoning process step by step
  - “Consider the dangers of lightning. Next, evaluate the safety of being outdoors vs. indoors. Conclude with the safest option.”

- Use of Examples - Providing solved examples with detailed reasoning step
  - “If A is bigger than B and B is bigger than C, then A is the biggest. For example, if A = 5, B = 3, and C = 2, then 5 is the biggest.”

- Iterative Refinement - Refining prompts based on model output for clarity and effectiveness
  - "Consider the relationship between A, B, and C step by step, starting with comparing A and B.”



Attractive Properties of CoT

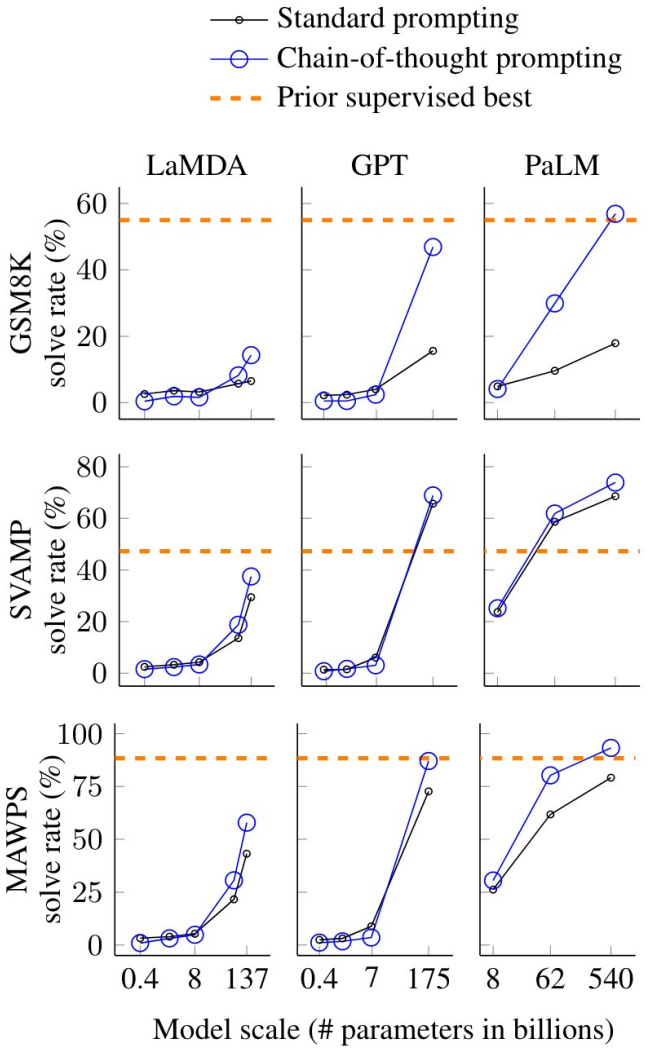
- Enables decomposition of multi-step problems.
- Provides interpretable insights into LLM's reasoning.
  - How was the answer derived?
  - Easy to debug when reasoning path went wrong
- Broad applicability across diverse reasoning tasks.
  - (Any) problems solvable via language.

Evaluation - Arithmetic Reasoning

- Benchmark:
  - Five math word problems: GSM8K, SVAMP, ASDiV, AQuA, MAWPS
  - “Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?” (GSM8K)
- Baseline: Standard prompting with in-context exemplars
- CoT: 8 few-shot exemplars with chains of thought for prompting

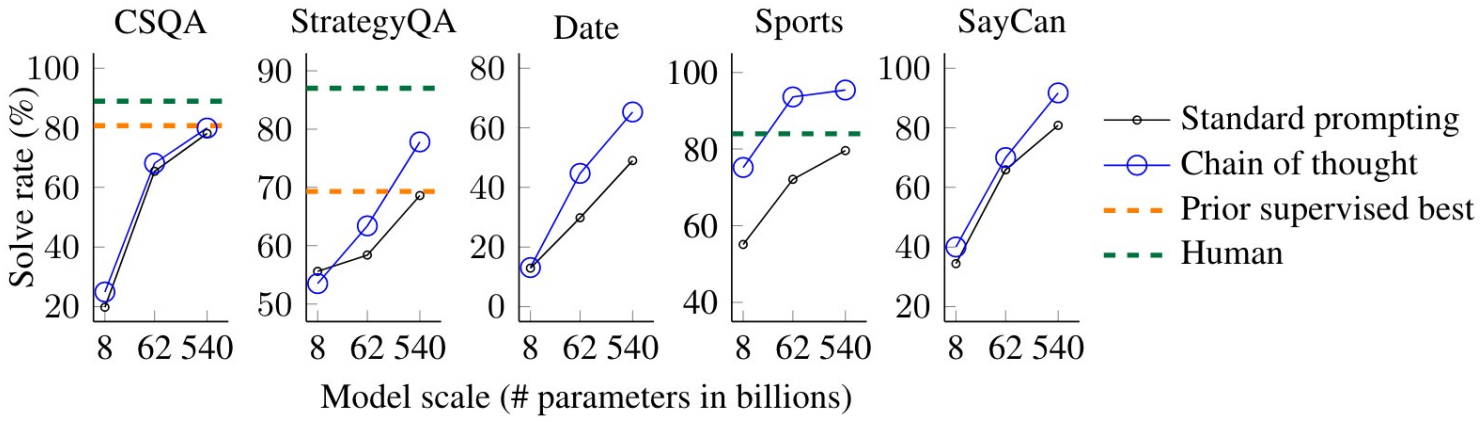
Evaluation - Arithmetic Reasoning

- CoT is an emergent ability of the model scale
  - No effect on small models
- Harder problems have a greater improvement
  - Performance doubled on GSM8K
- PaLM 62B -> PaLM 540B



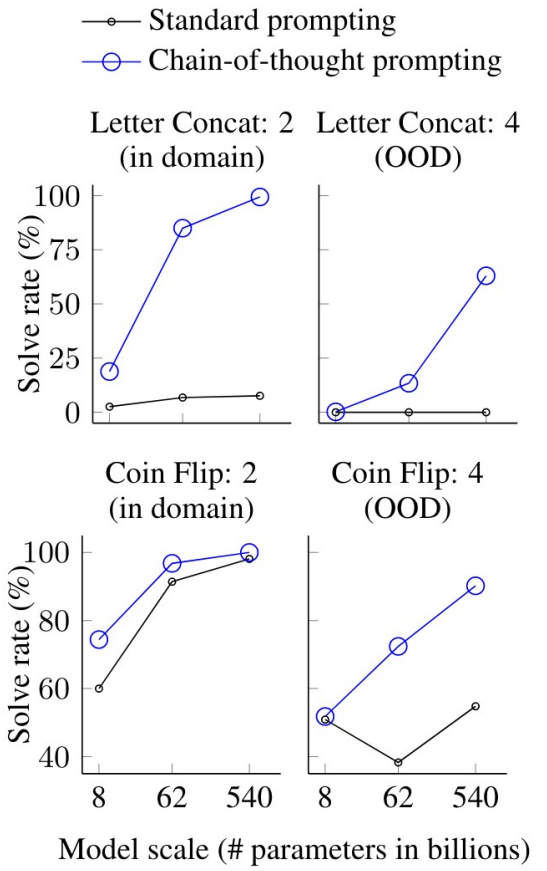
Evaluation - Commonsense Reasoning

- Benchmark: Five common sense reasoning datasets
  - Q: Yes or no: Would a pear sink in water?
- CoT: Same as arithmetic reasoning
  - Q: Yes or no: Would a pear sink in water? A: The density of a pear is about 0.6 g/cm^3, which is less than water. Thus, a pear would float. So the answer is no.
- Re



Evaluation - Symbolic Reasoning

- Tasks
  - Last letter concatenation. (e.g., “Amy Brown” -> “yn”).
  - Coin flip. (e.g., “A coin is heads up. Alice flips the coin. Bob does not flip the coin. Is the coin still heads up?”)
- CoT prompting: same as in common sense
  - Q: Take the last letters of the words in “Lady Gaga” and concatenate them. A: The last letter of “Lady” is “y”. The last letter of “Gaga” is “a”. Concatenating them is “ya”. So the answer is ya.
- Results for PaLM
  - In domain vs OOD
  - Standard prompting provide lower bound of LLMs



For last letter concatenation, the model only sees exemplars of names with two words, and then performs last letter concatenation on names with 3 and 4 words. We do the same for the number of potential flips in

Chain-of-thought prompting

Model 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 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Do we even need examples of reasoning?

Can we just ask the model to reason through things?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

[Wei et al., 2022; also see Nye et al., 2021]

Zero-shot chain-of-thought prompting

Model 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 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: ***Let's think step by step.*** *There are 16balls in total. Half of the balls are golf balls. That means there are 8 golf balls. Half of the golf balls are blue. That means there are 4 blue golf balls.*

✓

[Kojima et al., 2022]

Zero-shot chain-of-thought prompting

	MultiArith	GSM8K
Zero-Shot	**17.7**	**10.4**
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
Zero-Shot-CoT	**78.7**	**40.7**
Few-Shot-CoT (2 samples)	84.8	41.3
Few-Shot-CoT (4 samples : First) (*1)	89.2	-
Few-Shot-CoT (4 samples : Second) (*1)	90.5	-
Few-Shot-CoT (8 samples)	93.0	48.7

Greatly outperforms zero-shot →

Manual CoT still better →

[Kojima et al., 2022]

Zero-shot chain-of-thought prompting

No.	Category	Zero-shot CoT Trigger Prompt	Accuracy
1	APE	Let's work this out in a step by step way to be sure we have the right answer.	**82.0**
2	Human-Designed	Let's think step by step. (*1)	78.7
3		First, (*2)	77.3
4		Let's think about this logically.	74.5
5		Let's solve this problem by splitting it into steps. (*3)	72.2
6		Let's be realistic and think step by step.	70.8
7		Let's think like a detective step by step.	70.3
8		Let's think	57.5
9		Before we dive into the answer,	55.7
10		The answer is after the proof.	45.7
-		(Zero-shot)	17.7

APE: Automatic Prompt Engineer

[Zhou et al., 2022; Kojima et al., 2022]

Limitation of CoT

- Emulate the thought process, but not necessarily reasoning
- Cost of manually augmenting exemplars with CoTs
- Emergence of CoT reasoning only at large model scale
- No guarantee of correct reasoning paths

Self-Consistency Improves Chain of Thought Reasoning in Language Models

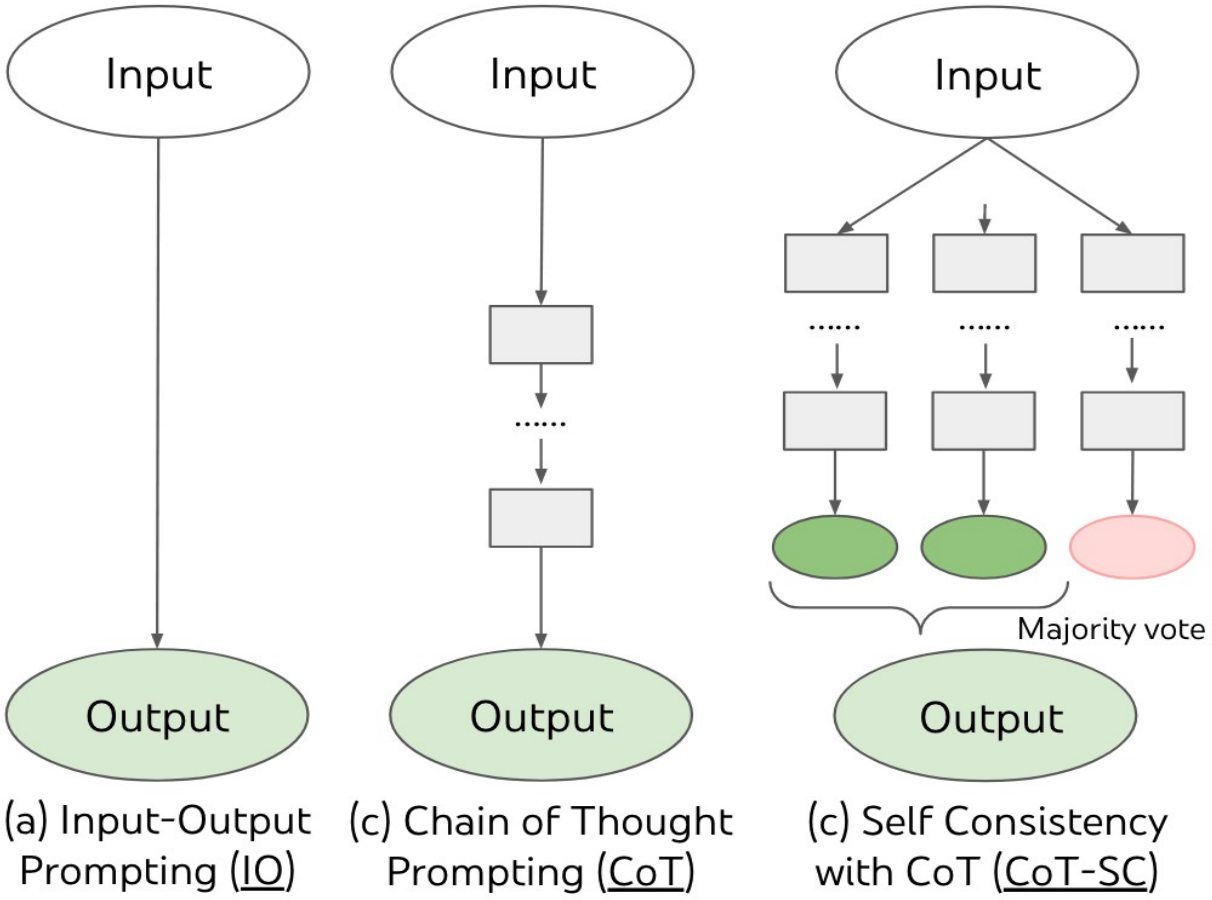
Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, Denny Zhou. 2022

Motivation

- Analogous to the human way of thinking:
  - If multiple different ways of thinking lead to the same answer, one has greater confidence that the final answer is correct.
  - Complex reasoning tasks typically admit multiple reasoning paths that reach a correct answer.

- Correct reasoning processes tend to have greater agreement in their final answer than incorrect processes.

Traditional CoT vs CoT w/ Self Consistency



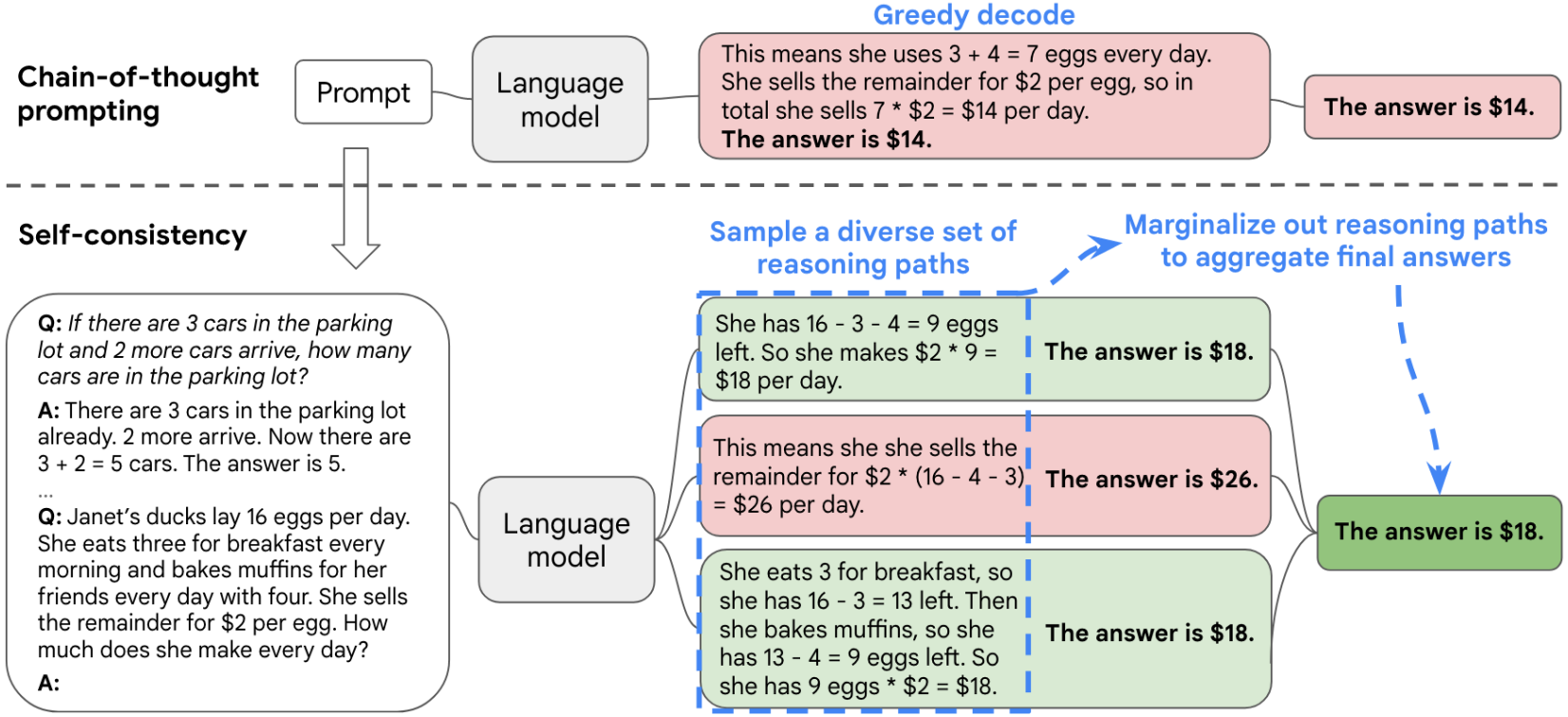
(a) Input-Output Prompting (IO)

(c) Chain of Thought Prompting (CoT)

(c) Self Consistency with CoT (CoT-SC)

3 steps of Self Consistency

1. Prompting LLM with CoT
2. Sampling from the LLM decoder to get diverse reasoning paths
3. Selecting the most consistent answers (e.g. majority



Properties of Self Consistency

- Self-ensemble
  - No additional training required
  - No additional human annotation
  - No auxiliary models
  - No fine-tuning
  - ...
- Off-the-shelf method
- Robust to sampling strategies and imperfect prompts.
  - Sometimes CoT hurts performance , while CoT-SC dose not

Evaluation

- Benchmarks:
  - Arithmetic reasoning: same as CoT paper
  - Commonsense reasoning: CommonsenseQA, StrategyQA, ARC(new)
  - Symbolic Reasoning: same as CoT paper
- Baseline: 8 few-shot exemplars with CoT prompting
- Language models: GPT3, LaMDA, PaLM, UL2 20B, ~~Codex~~

Results

- Self-consistency improves all three kinds of reasoning performance over all four language models significantly over chain-of-thought prompting.
- The more reasoning path, the better
- Self-Consistency Improves Robustness to Imperfect Prompts

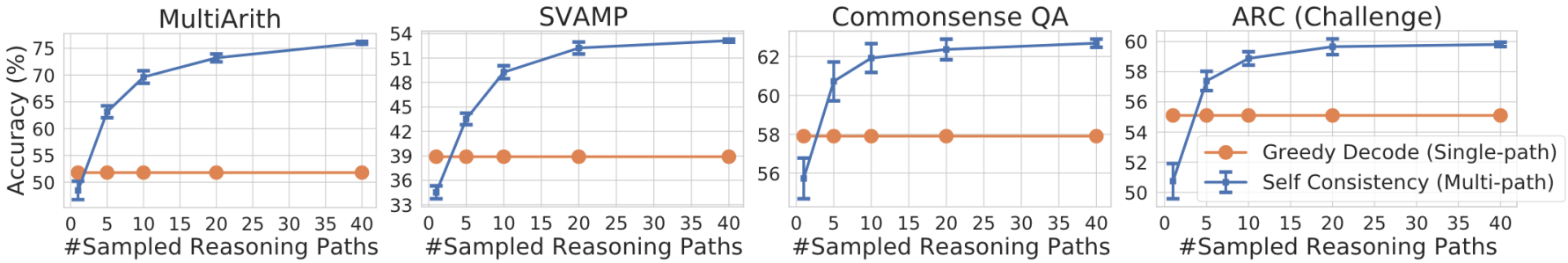


Figure 2: Self-consistency (blue) significantly improves accuracy over CoT-prompting with greedy decoding (orange) across arithmetic and commonsense reasoning tasks, over LaMDA-137B. Sampling a higher number of diverse reasoning paths consistently improves reasoning accuracy.

SIFT: Grounding LLM Reasoning in Contexts via Stickers

Zeng, Xuyao Huang*, Boxiu Li*, Zhijie Deng†.
2025

Amazing Performance

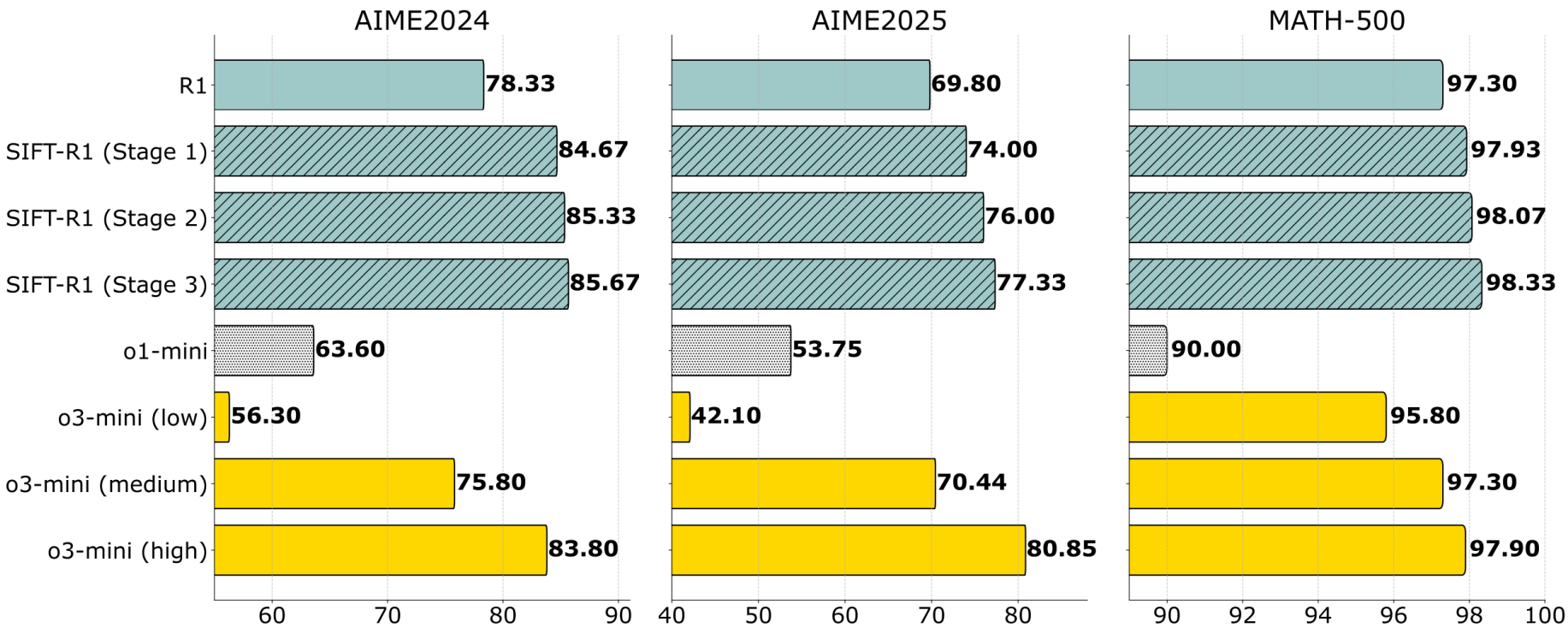


Figure 1: Applying SIFT to DeepSeek-R1 demonstrates highly competitive reasoning performance on AIME2024, AIME2025, and MATH-500 (pass@1 accuracy). The results for o1-mini and o3-mini on AIME are referenced from Ye et al. (2025).

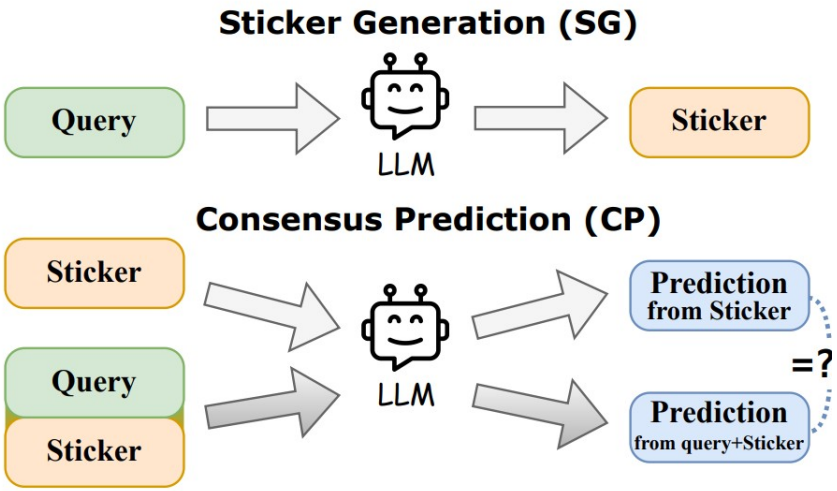


Figure 5: Four core operations in SIFT: (i) Sticker Gen Optimization (FO), (iv) Inverse Generation (IG).

Algorithm 2: Consensus Prediction (CP)

Input : Query Q, Sticker S
Output : Prediction from Q & S, or $\rightsquigarrow$ (unequal)

$P_S \leftarrow \text{LLM}(S)$; // Sticker-only
$P_{Q,S} \leftarrow \text{LLM}(Q, S)$; // Query+Sticker
if $\text{EQUIVALENT}(P_S, P_{Q,S})$ **then**
  // Consensus validation
  return $P_{Q,S}$
else
  return $\rightsquigarrow$
end

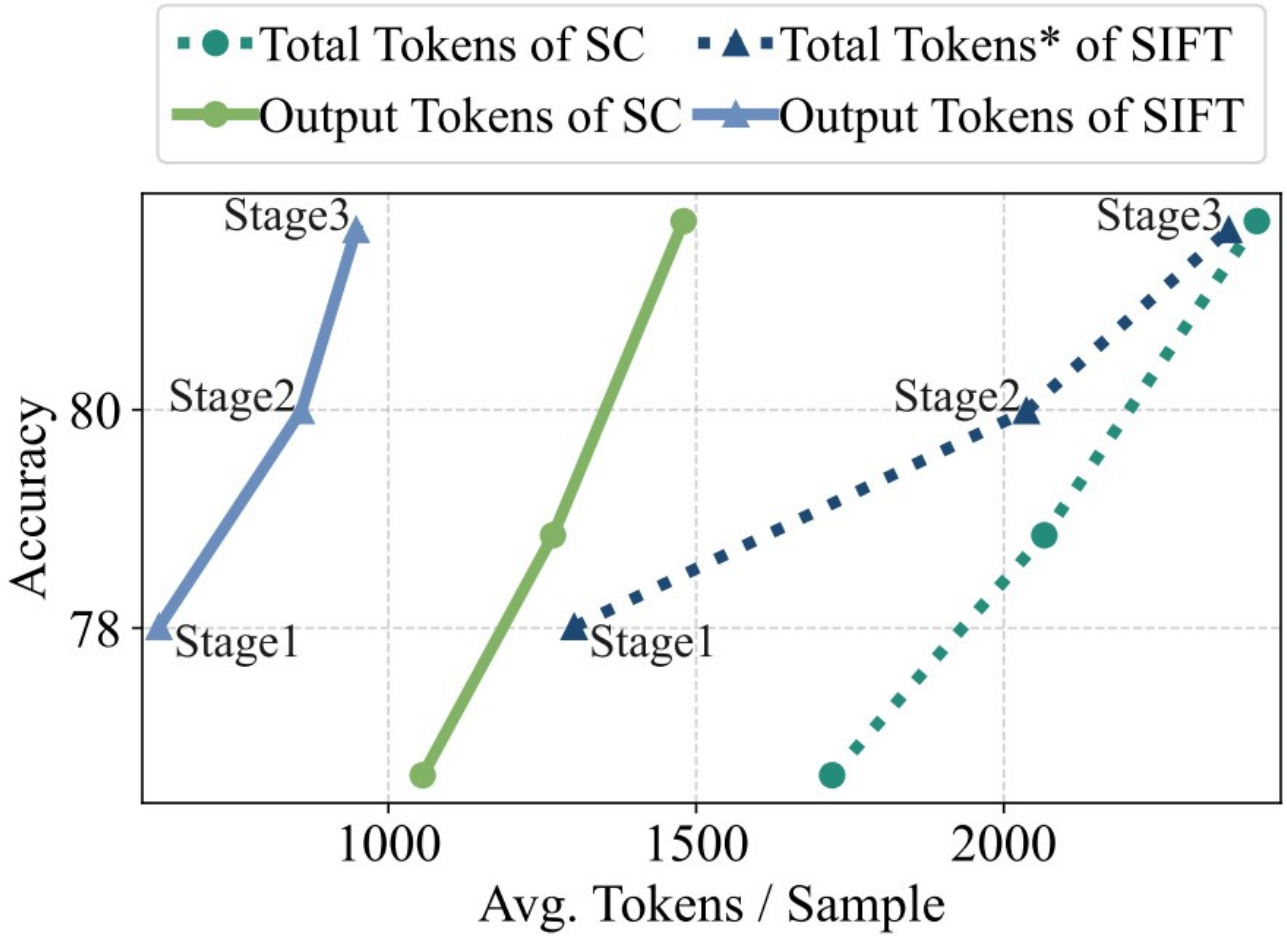
Algorithm 1: LLM reasoning with SIFT

Input : Query Q
Output : Final result of Q

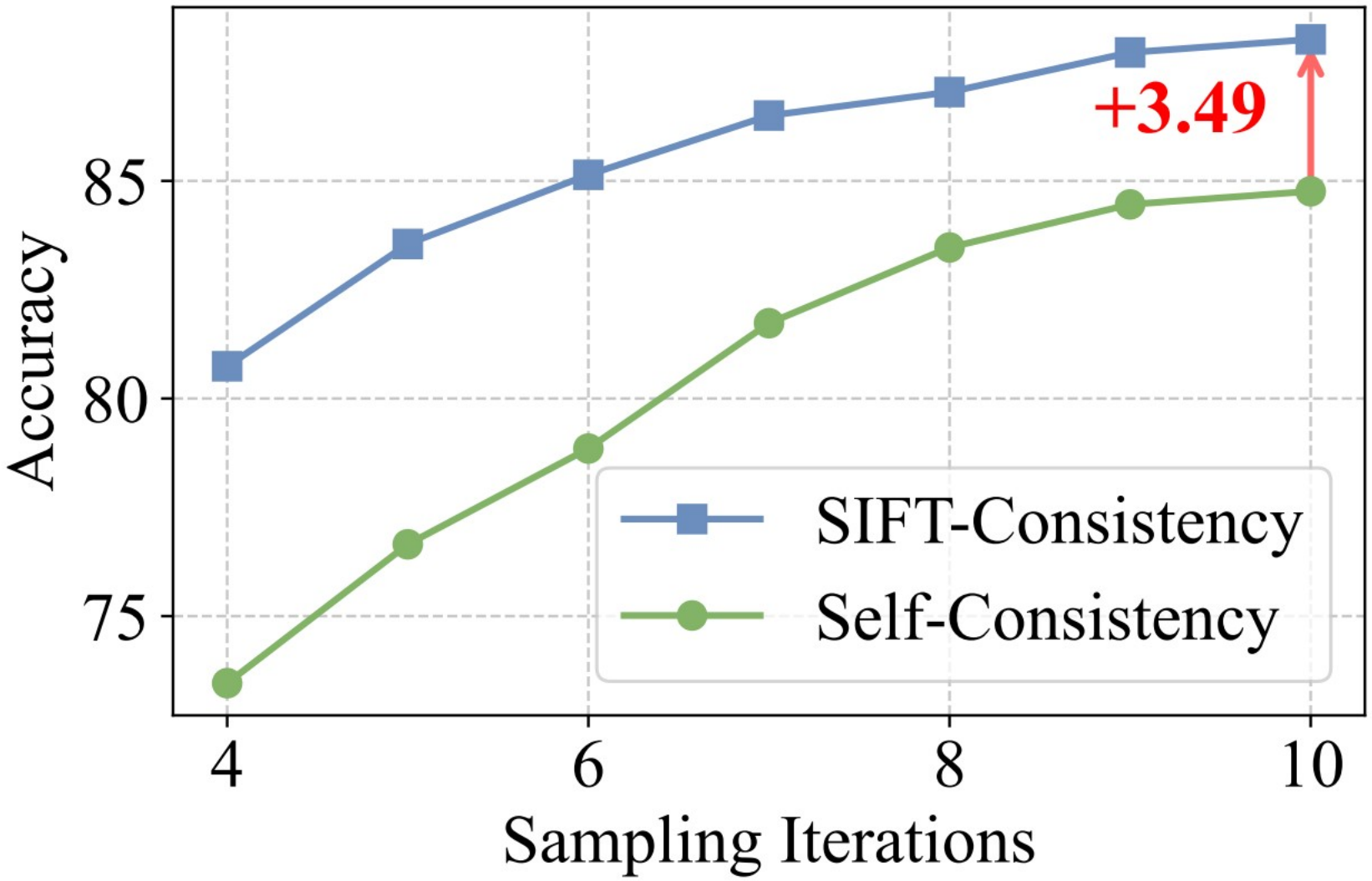
$S_1 \leftarrow \text{SG}(Q)$; // Sticker generation
$P_1 \leftarrow \text{CP}(Q, S_1)$;
if $P_1 \neq\rightsquigarrow$ **then**
  return P_1 ; // Exit if consensus
else
  // Forward
  $S_2 \leftarrow \text{FO}(Q, S_1)$, $P_2 \leftarrow \text{CP}(Q, S_2)$;
  if $P_2 \neq\rightsquigarrow$ **then**
    return P_2
  else
    // Inverse
    $S_3 \leftarrow \text{FO}(Q, \text{IG}(P_{Q,S_2}))$;
    $P_3 \leftarrow \text{CP}(Q, S_3)$;
    return P_3 **if** $P_3 \neq\rightsquigarrow$ **else** $\text{LLM}(Q)$
  end
end

Tokens vs Performance

- Compared to Standard SelfConsistency



SIFT+Consistency



Program of Thoughts Prompting

Wenhu Chen, Xueguang Ma, Xinyi Wang, William W. Cohen. 2022

https://arxiv.org/pdf/2211.12588.pdf

Motivation

- Let LM do the math if you have a calculator
- Let LM execute the program if you have an interpreter?

So…

- Delegate computation to program interpreter
- Decoupling complex computation from reasoning and language understanding.

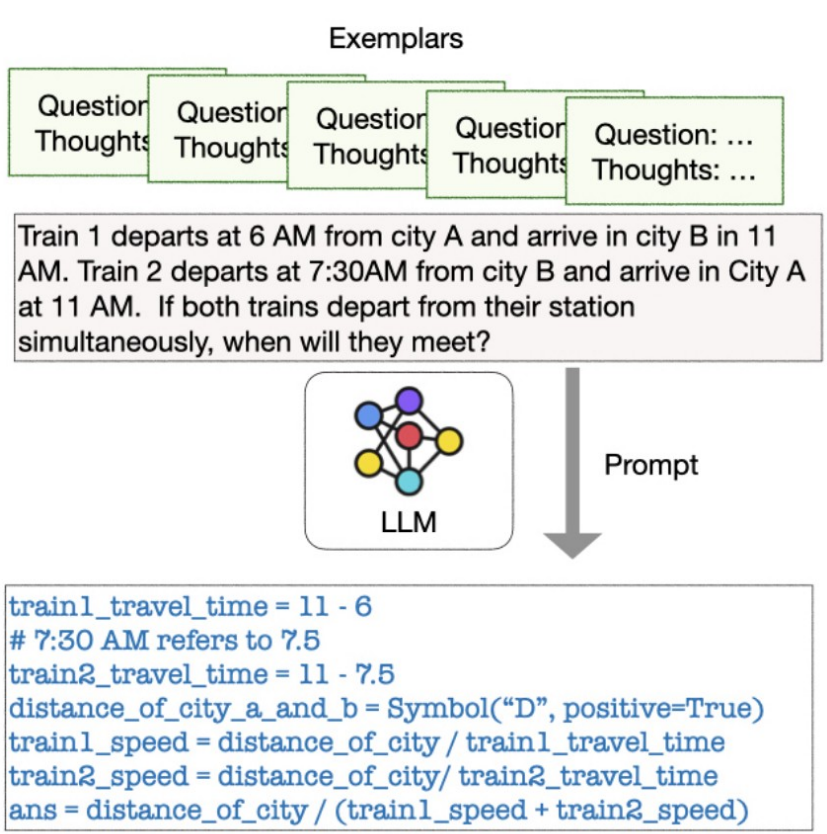
Understanding the gap between Cot and PoT

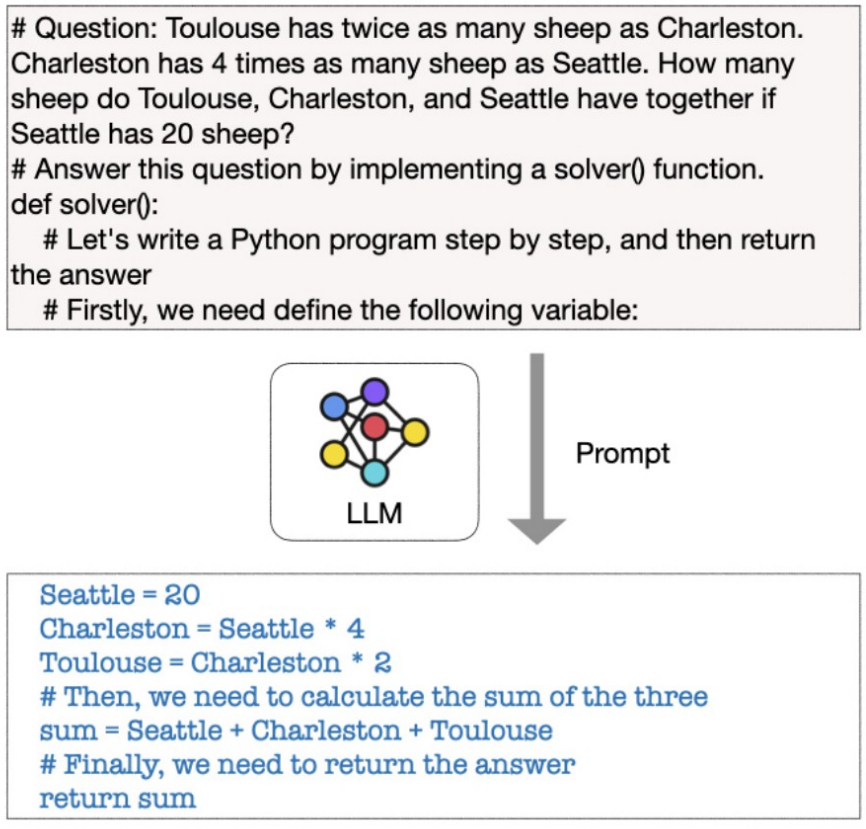
- Why LM are not ideal for solving expressions
  - LLMs often make errors in arithmetic calculations, particularly with large numbers
  - They struggle to solve complex mathematical problems like polynomial or differential equations
  - LLMs are inefficient at handling iterative processes, especially with a high number of iterations

Understanding the gap between Cot and PoT

- CoT uses LLM for both reasoning and computation
- PoT uses code interpreter for external computation
- PoT breaks down complex equations into multi-step thought processes
- PoT assigns semantic meanings to variables

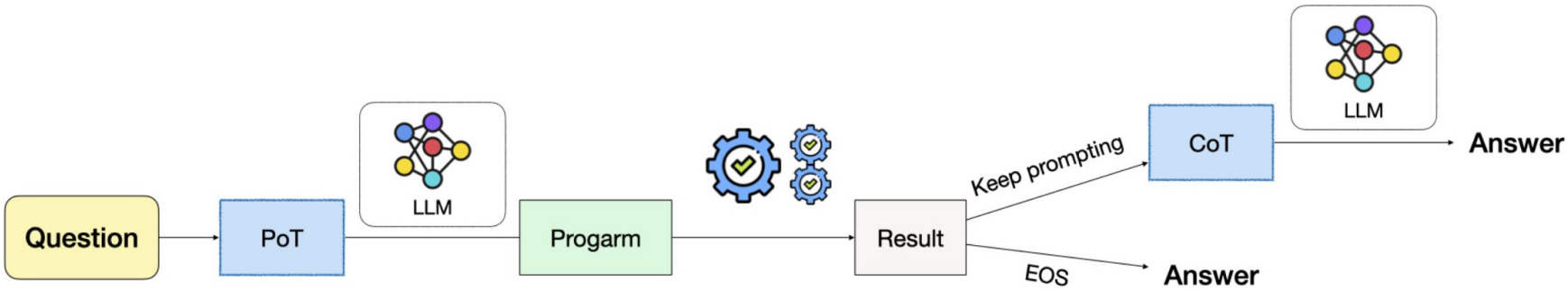
Few-shot PoT vs Zero-shot PoT





PoT as an Intermediate Step

- Program of Thoughts (PoT) for computations in problems needing textual reasoning
- Execute PoT-generated programs for intermediate results, then use Chain of Thoughts (CoT) for the final answer.



Chain of Thoughts vs Program of Thoughts

Question: In Fibonacci sequence, it follows the rule that each number is equal to the sum of the preceding two numbers. Assuming the first two numbers are 0 and 1, what is the 50th number in Fibonacci sequence?

The first number is 0, the second number is 1, therefore, the third number is 0+1=1. The fourth number is 1+1=2. The fifth number is 1+2=3. The sixth number is 2+3=5. The seventh number is 3+5=8. The eighth number is 5+8=13.
….. (Skip 1000 tokens)
The 50th number is 32,432,268,459.

CoT

32,432,268,459 ✗

```
length_of_fibonacci_sequence = 50
fibonacci_sequence = np.zeros(length_of_)
fibonacci_sequence[0] = 0
fibonacci_sequence[1] = 1
For i in range(3, length_of_fibonacci_sequence):
    fibonacci_sequence[i] = fibonacci_sequence[i-1] +
    fibonacci_sequence[i-2]
ans = fibonacci_sequence[-1]
```

PoT

python → 12,586,269,025 ✓

Question: Ketty saves 20000 dollars to the bank. After three years, the sum with compound interest rate is 1000 dollars more than the sum with simple interest rate. What is the interest rate of the bank?

Assuming the interest rate is x. The sum after two years with simple interest rate is 20000 + x * 20000 * 3 = 20000 + 60000x. The sum after two years with compoud interest rate is 20000 * (1 + x) ^ 3 = 200000 + 60000 * x + 60000x^2 + 20000x^3. The difference can be written as 60000x^2 + 20000x^3 = 1000. In order to solve x, we can use the quadratic formula. x = (-b +- sqrt(b^2 - 4ac)) / 2a, …. , x = (-20000 +- 6160) / 120000, x = -0.051333.

CoT

-0.051333 ✗

```
interest_rate = Symbol('x')
sum_in_two_years_with_simple_interest= 20000 +
interest_rate * 20000 * 3
sum_in_two_years_with_compound_interest = 20000 * (1 +
interest_rate)**3
# Since compound interest is 1000 more than simple interest.
ans = solve(sum_after_in_yeras_with_compound_interest -
sum_after_two_years_in_compound_interest - 1000,
interest_rate)
```

PoT

python SymPy → x = 0.24814 ✓

Evaluation

- Benchmarks:
  - Five Math problem datasets, GSM8K, AQuA, SVAMP, TabMWP, MultiArith
  - Three financial datasets, FinQA, ConvFinQA, and TATQA

Dataset	Split	Example	Domain	Input	Output
GSM8K (Cobbe et al., 2021)	Test	1318	MWP	Question	Number
AQuA (Ling et al., 2017)	Test	253	MWP	Question	Option
SVAMP (Patel et al., 2021)	Test	1000	MWP	Question	Number
MultiArith (Roy & Roth., 2015)	Test	600	MWP	Question	Number
TabMWP (Lu et al., 2022)	Test	7861	MWP	Table + Question	Number + Text
FinQA (Chen et al., 2021b)	Test	1147	Finance	Table + Text + Question	Number + Binary
ConvFinQA (Chen et al., 2022)	Test	421	Finance	Table + Text + Conversation	Number + Binary
TATQA (Zhu et al., 2021)	Dev	1668	Finance	Table + Text + Question	Number + Text

Evaluation

- Implementation details
  - Model: mainly used Codex (code-davinci-002), also used GPT-3, PaLM, LaMDA
  - Few-shot settings: 4-8 shots for all datasets
- Baselines
  - CoT
  - CoT-SC (with majority voting)

Results (Compared to CoT)

- Under few-shot: 8% gain on math datasets; 15% gain on financial datasets
- Under zero-shot: 12% gain on math dataset
- On average, the performance of PoT-SC was 10% better than

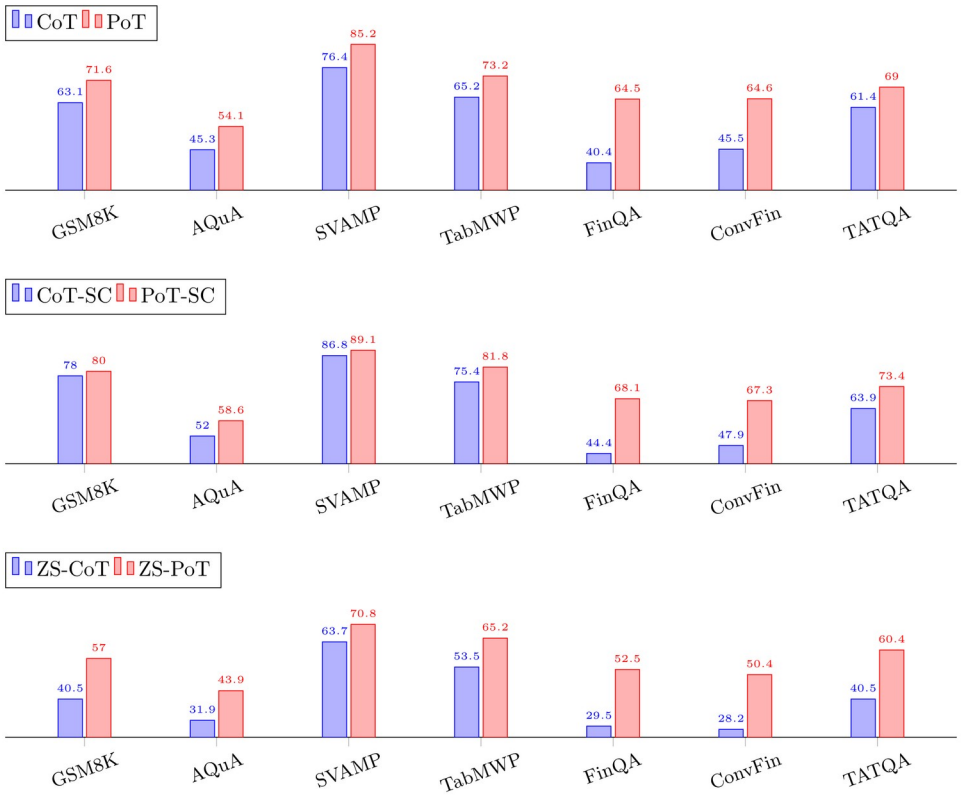


Figure 2: Few-shot (upper), Few-shot + SC (middle) and Zero-Shot (lower) Performance overview of Codex PoT and Codex CoT across different datasets.

Summary

- Prompt engineering has become a new technology for NLP in the era of LLMs
- There are still a lot of unknown things about it
  - Bias,
  - Privacy,
  - etc.