Chain-of-Thought Prompting

Mengzi Cheng (paper 3),
Yi Ping (paper 1),
Zhenghao He (paper 2)

Paper 1
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei et al. · Google Research (Brain Team)

arXiv:2201.11903

01

Background

From <input, output> to <input, chain-of-thought, output>

Background - Current Problem

Scaling ≠ Reasoning:
Scaling up improved performance and sample efficiency, but not enough for reasoning tasks.

Idea 1
Generate intermediate steps by training from scratch or fine-tuning a pretrained model.
Limitation
It is costly to create a large set of high quality rationales

Idea 2
Prompt with a few input-output exemplars.

Limitation
Work poorly on reasoning
Does not improve substantially with scale

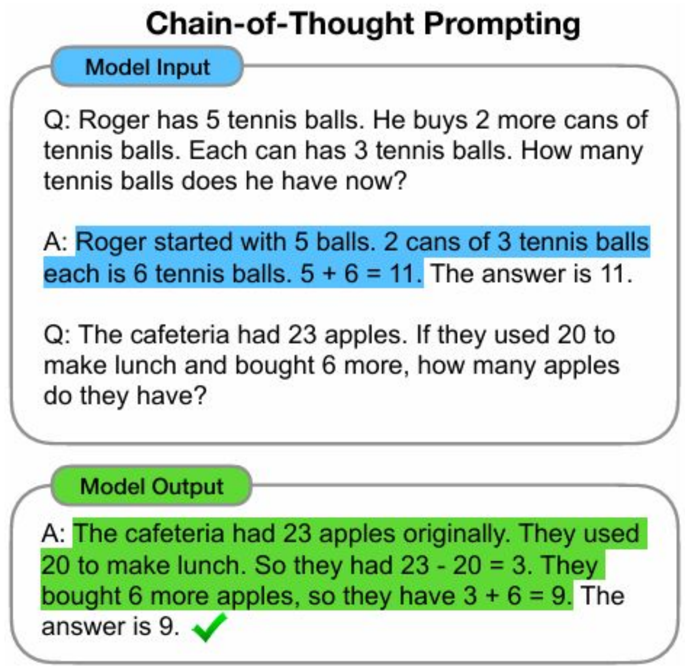
Background - Chain-of-Thought

Chain-of-Thought: intermediate natural language reasoning steps

No fine-tuning, only **Prompt**

Standard few-shot: exemplar pairs (input, output)

CoT prompting: exemplar triples (input, chain-of-thought, output)


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. ✓

https://arxiv.org/abs/2201.11903

Background - How CoT Work?

1

Break a problem into intermediate steps.
->
Allocate more computation to multi-step problem.

2

Intermediate reasoning steps
->
Gives an interpretable window into how an answer might be produced

3

Natural language reasoning steps
->
Can be used for arithmetic reasoning, commonsense reasoning, or symbolic reasoning

4

Elicitable via few-shot exemplars

02

Experiment

Arithmetical reasoning, commonsense reasoning, and symbolic reasoning

Experiment - Arithmetic Reasoning - Experiment Design

Evaluation: correctness of final answer (solve rate)

Benchmarks: GSM8K, SVAMP, ASDiv, AQuA, and MAWPS

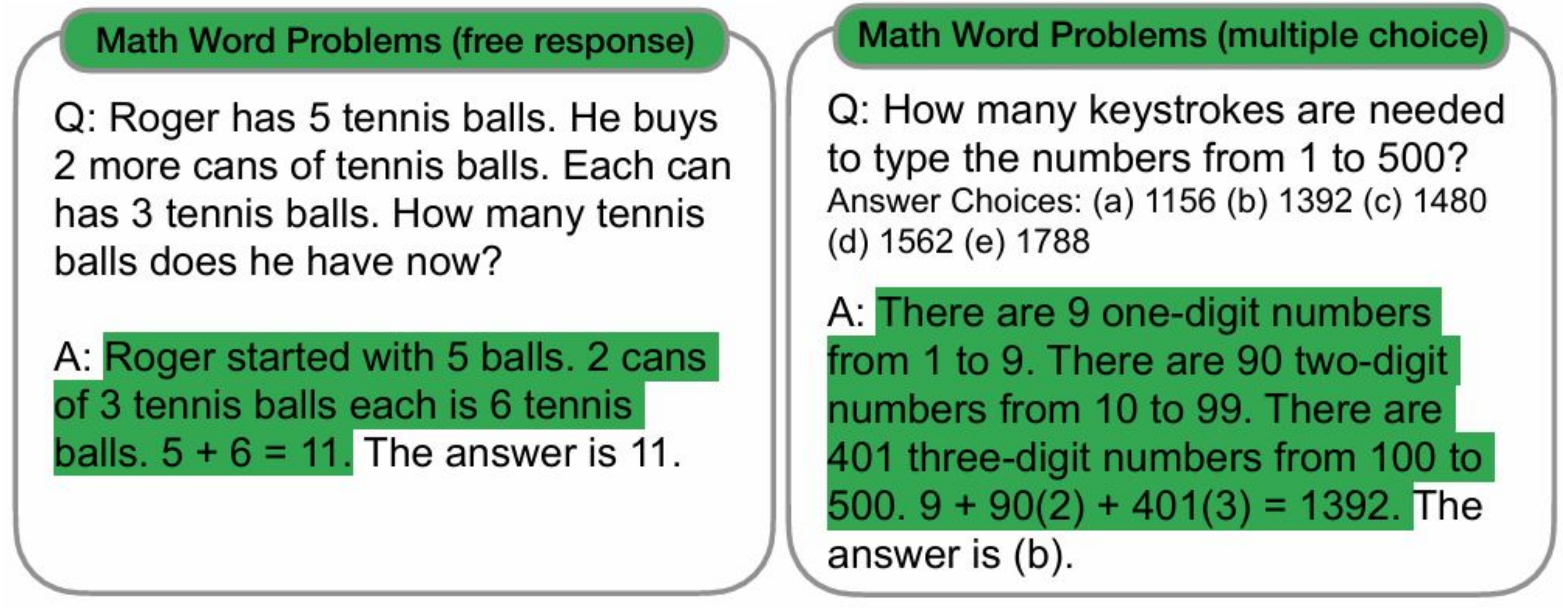
Math word problems -> semantic + multi-step reasoning

Baseline: standard prompting exemplars
Input -> output

Proposed: CoT exemplars
Input -> chain-of-thought -> output

Wrote and used 8 few-shot exemplars with CoT for GSM8K, SVAMP, ASDiv, and MAWPS

Experiment - Arithmetic Reasoning - Experiment Design



Math Word Problems (free response)

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.

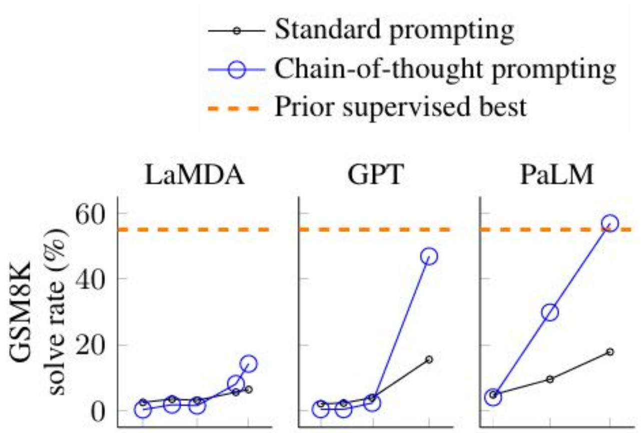
Math Word Problems (multiple choice)

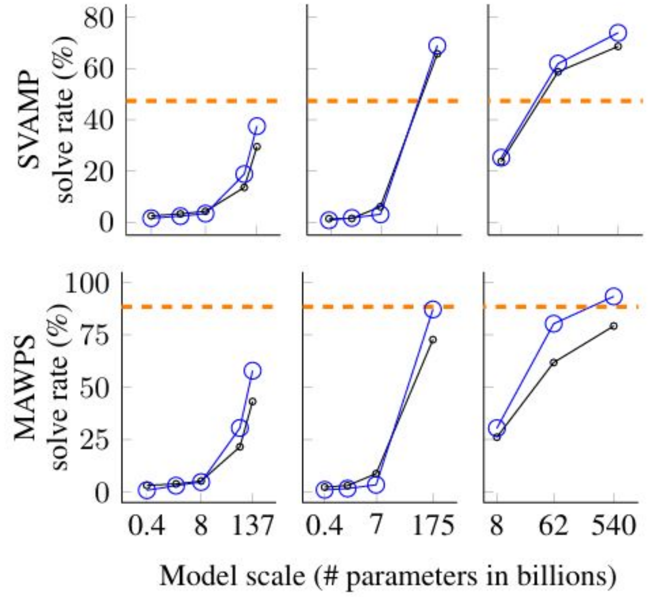
Q: How many keystrokes are needed to type the numbers from 1 to 500?
Answer Choices: (a) 1156 (b) 1392 (c) 1480 (d) 1562 (e) 1788

A: There are 9 one-digit numbers from 1 to 9. There are 90 two-digit numbers from 10 to 99. There are 401 three-digit numbers from 100 to 500. 9 + 90(2) + 401(3) = 1392. The answer is (b).

Experiment - Arithmetic Reasoning - Results

1. CoT's ability emergent with scale
2. Harder problems benefit more
3. For GPT-3 175B and PaLM 540B, CoT is comparable to task-specific finetuned models.

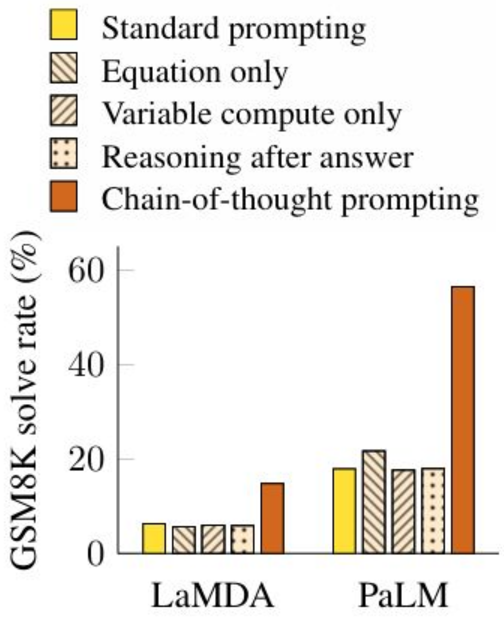




https://arxiv.org/abs/2201.11903

Experiment - Arithmetic Reasoning - Ablation Test

Equation Only	Variable compute only	Reasoning after answer
Replace reasoning with equation	Replace reasoning with string of dot with same length	Put the chain-of-though t after the final answer
Hard to map from text to equation; natural language matters.	Increasing generation length does not explain CoT.	CoT is not just activating knowledge.



https://arxiv.org/abs/2201.11903

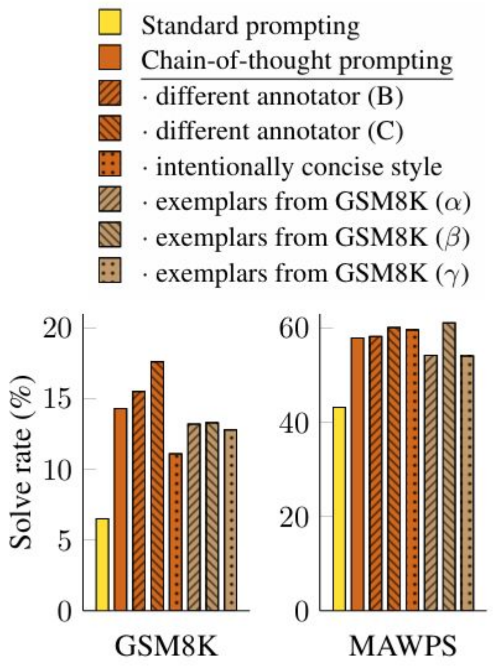
Experiment - Arithmetic Reasoning - Robustness

Changes:

- Different annotators for the same exemplars.
- Annotators write an intentionally concise CoT style.
- Different exemplars randomly sampled from GSM8K training set.

Finding:

- CoT variants outperform the standard baseline
- CoT success does not depend on style



https://arxiv.org/abs/2201.11903

Experiment - Commonsense Reasoning - Experiment Design

Evaluation: (solve rate)

Benchmarks: CSQA, StrategyQA, Date Understanding, Sports Understanding, SayCan

Baseline: standard prompting exemplars
 Input -> output

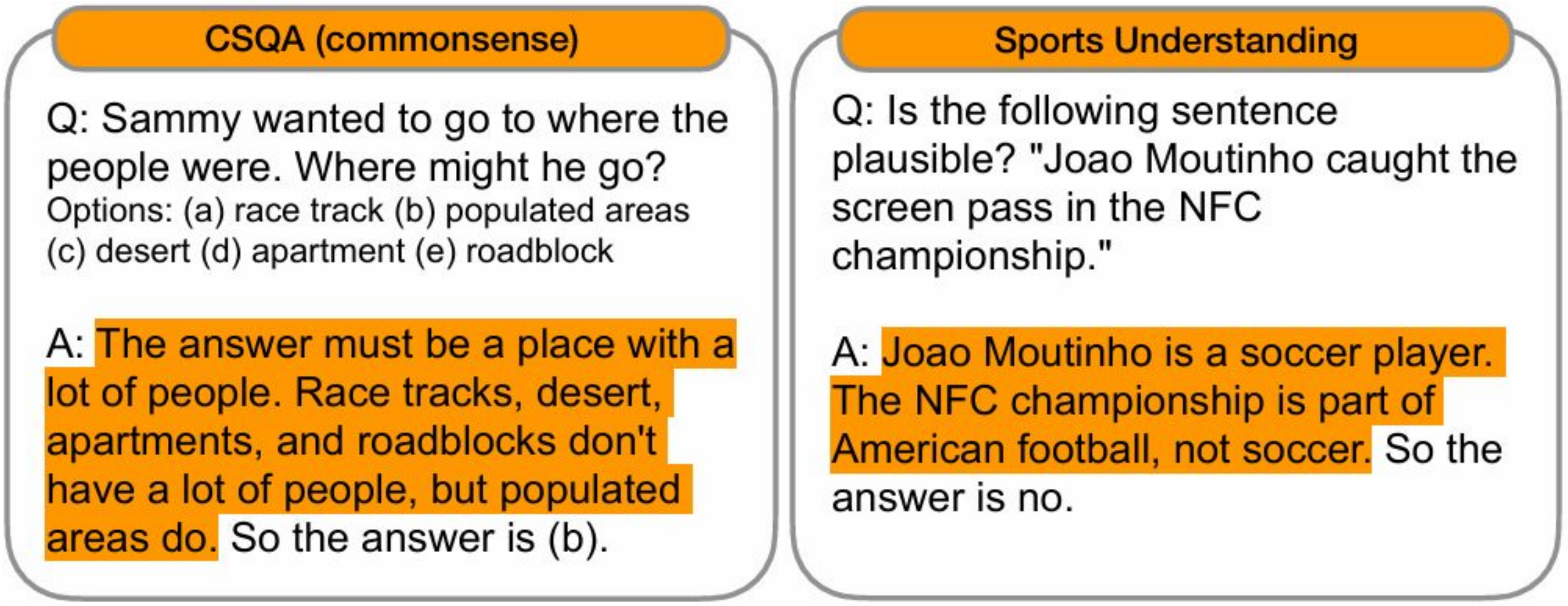
Proposed: CoT exemplars
Input -> chain-of-thought -> output
$\leq$ 60 tokens and $\leq$ 2 reasoning steps

CSQA & StrategyQA: Manually composed exemplars

Date Understanding& Sports Understanding: first 10 evaluation exemplars

SayCan: 6 training examples & manually composed exemplars

Experiment - Commonsense Reasoning - Experiment Design



CSQA (commonsense)

Q: Sammy wanted to go to where the people were. Where might he go?
Options: (a) race track (b) populated areas (c) desert (d) apartment (e) roadblock

A: The answer must be a place with a lot of people. Race tracks, desert, apartments, and roadblocks don't have a lot of people, but populated areas do. So the answer is (b).

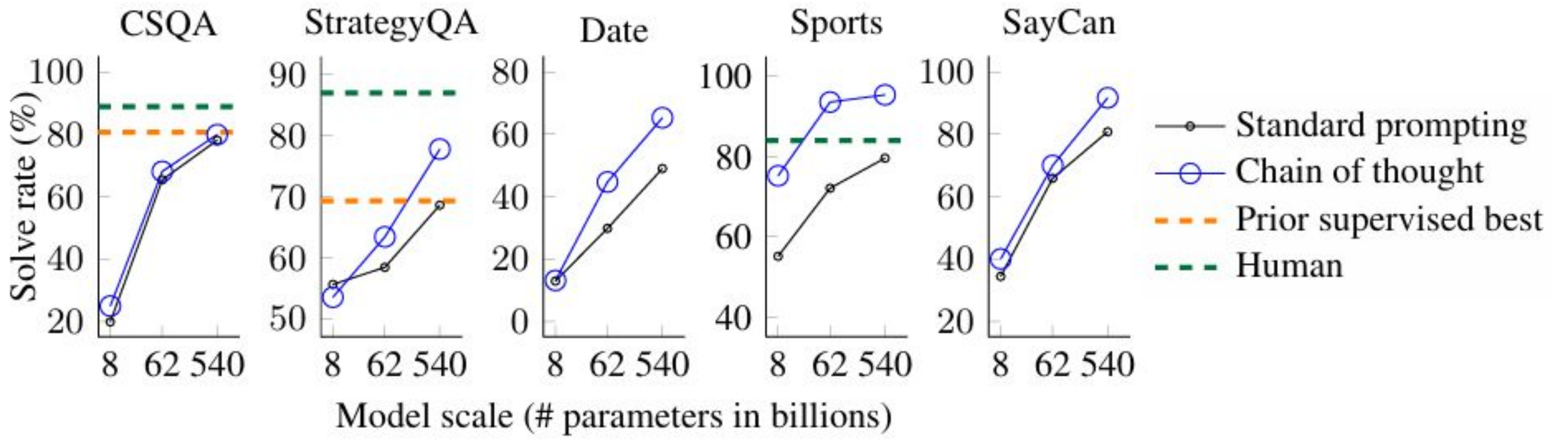
Sports Understanding

Q: Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."

A: Joao Moutinho is a soccer player. The NFC championship is part of American football, not soccer. So the answer is no.

Experiment - Commonsense Reasoning - Results

1. CoT helps beyond scaling
2. Strong across diverse tasks
3. Benefit is task-dependent



https://arxiv.org/abs/2201.11903

Experiment - Symbolic Reasoning - Experiment Design

Tasks:

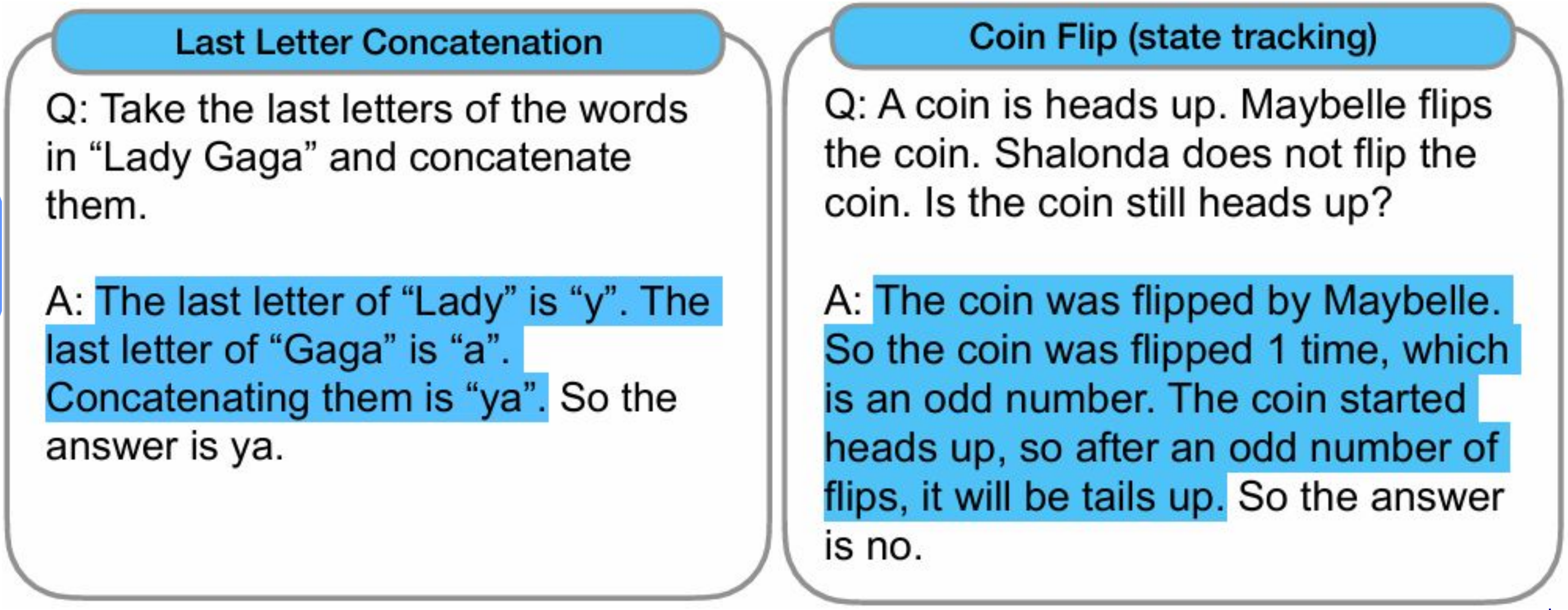
- Last letter concatenation: Concatenate last letters of each word
- Coin flip: Track whether a coin is heads up after a sequence of people flip

In-domain: same number of steps
Out-of-domain: more steps than exemplars

Test set include both in-domain and out-of-domain tests.

Manually compose exemplars

Experiment - Symbolic Reasoning - Experiment Design



Last Letter Concatenation

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.

Coin Flip (state tracking)

Q: A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?

A: The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.

Experiment - Symbolic Reasoning - Results

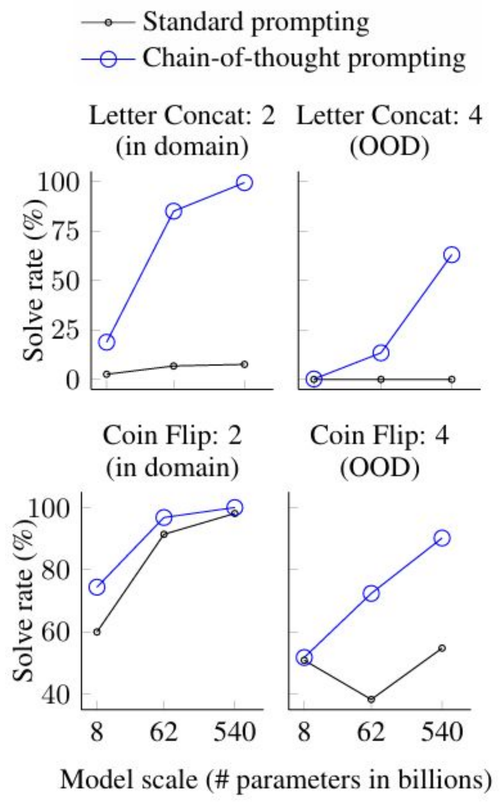
In-domain:

- PaLM 540B + CoT is near-perfect.
- Small models still fail

Out-of-domain:

- Improve with scale

CoT improves generalization to longer sequences



https://arxiv.org/abs/2201.11903

03

Major Contributions

Chain-of-thought prompting unlocks reasoning ability

Major Contributions

- Method:
  - CoT Prompting: <input, CoT, output>
  - Few-shot exemplars that include reasoning steps
- Demonstrate strong benefits on reasoning tasks
  - Arithmetic: huge improvements, stronger than ablations, and robust to exemplars
  - Commonsense: improve diverse tasks
  - Symbolic: generalize to longer sequence length
- CoT reasoning emerges with model scale.

04

Limitations & Future Directions

Limitations & Future Directions

Limitations

- CoT ≠ model is “reasoning”
- Human-written CoT is expansive at scale
- Reasoning path can be wrong
- CoT reasoning emerges mainly at large-scale models

Future Directions

- Improve factual generation
- Synthesize data generation
- Induce reasoning in smaller models
- Explore more prompting methods

Paper 2
Tree of Thoughts: Deliberate Problem Solving with Large Language Models

01

Background

Background - Motivation

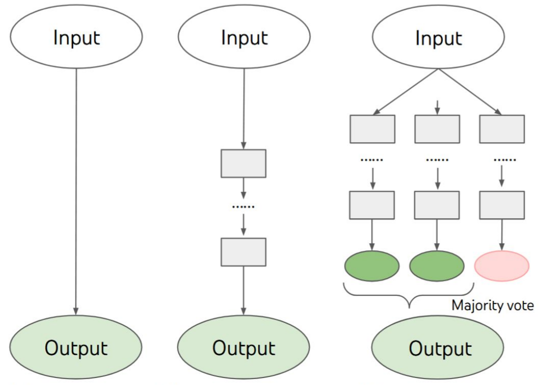
Large Language Models as Problem Solvers

- LLMs have shown strong performance on reasoning tasks via prompting methods such as Input–Output and Chain-of-Thought (CoT).
- These methods rely on left-to-right, single-path decoding during inference.

Background - Motivation

Limitation of Existing Inference Paradigms

- No explicit exploration of alternative reasoning paths.
- No lookahead or backtracking once an early decision is made.
- Performance degrades on tasks requiring planning, search, or global decisions.



(a) Input-Output Prompting (IO) (c) Chain of Thought Prompting (CoT) (c) Self Consistency with CoT (CoT-SC)

Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models, 2023." *URL https://arxiv. org/abs/2305.10601* 3 (2023): 1.

Background - Motivation

Key Observation

- Human problem solving resembles search in a combinatorial space, not a single linear chain.
- Existing prompting methods lack an explicit search structure.

Motivation

- Can we augment LLM inference with a deliberate, tree-structured reasoning process
- that enables exploration, evaluation, and backtracking without additional training?

02

Technical Detail

Technical Detail - Method Overview

Thought as a Semantic Unit

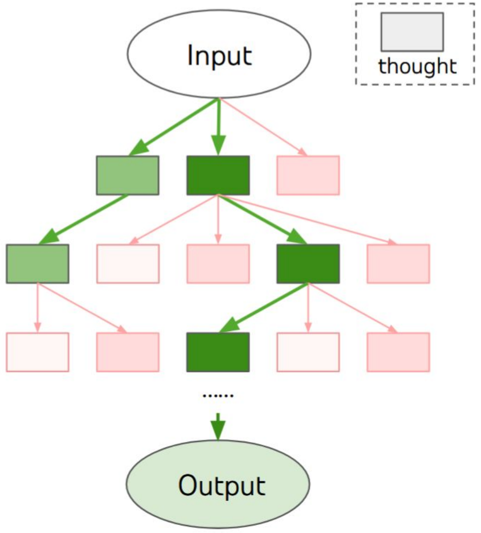
- A thought is defined as a coherent language sequence that serves as an intermediate step toward problem solving.
- The granularity of a thought is task-dependent (e.g., an equation line, a word, or a paragraph).

Tree of Thought Formulation

- Each node in the tree corresponds to a state

  $$s=[x,z1,\dots,zi]$$

  where *x* is the input and *z_i* are generated thoughts.
- Problem solving is framed as searching for a path from the root to a terminal state.



Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models, 2023." *URL https://arxiv. org/abs/2305.10601* 3 (2023): 1.

Technical Detail - Method

1. Thought Decomposition
   a. small enough to allow diverse generation,
   b. large enough to support meaningful evaluation.
2. Thought Generation
   a. Given a state s ToT generates k candidate thoughts using a thought generator $G(p\theta, s, k)$
   b. Sampling-based generation
      i. Sample i.i.d. thoughts from a CoT-style prompt.
   c. Proposal-based generation
      i. Propose multiple candidate thoughts sequentially in a single prompt to encourage diversity.

Technical Detail - Method

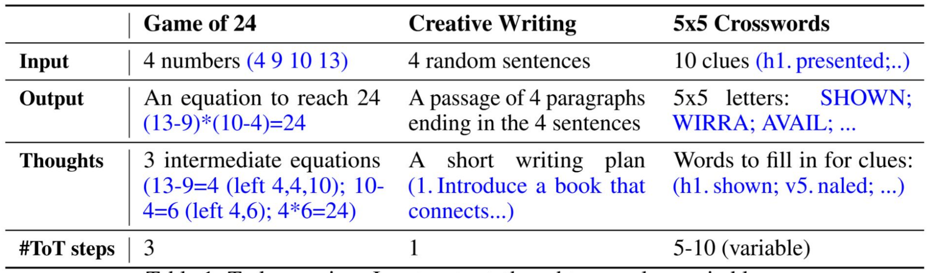
3. ToT uses a state evaluator $V(p\theta,S)$implemented by the LM itself.
    a. Value-based evaluation
    b. Vote-based evaluation
4. Search Algorithm
    a. Breadth-First Search (BFS): keeps the top-*b* states at each step.
    b. Depth-First Search (DFS): explores promising paths with pruning and backtracking.

This enables lookahead and backtracking, which are absent in standard prompting.

Technical Detail - Experiments

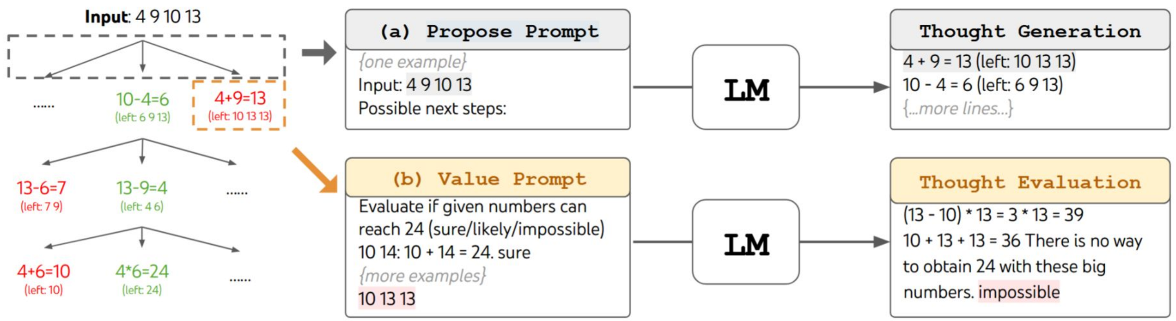
TASKS:

- Game of 24
  - Mathematical reasoning with exact constraints.
- Creative Writing
- 5×5 Mini Crosswords



	Game of 24	Creative Writing	5x5 Crosswords
Input	4 numbers (4 9 10 13)	4 random sentences	10 clues (h1. presented;..)
Output	An equation to reach 24 (13-9)*(10-4)=24	A passage of 4 paragraphs ending in the 4 sentences	5x5 letters: SHOWN; WIRRA; AVAIL; ...
Thoughts	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; ...)
#ToT steps	3	1	5-10 (variable)

Table 1: Task overview. Input, output, thought examples are in blue.



(a) Propose Prompt
{one example}
Input: 4 9 10 13
Possible next steps:

Thought Generation
4 + 9 = 13 (left: 10 13 13)
10 - 4 = 6 (left: 6 9 13)
{...more lines...}

(b) Value Prompt
Evaluate if given numbers can reach 24 (sure/likely/impossible)
10 14: 10 + 14 = 24. sure
{more examples}
10 13 13

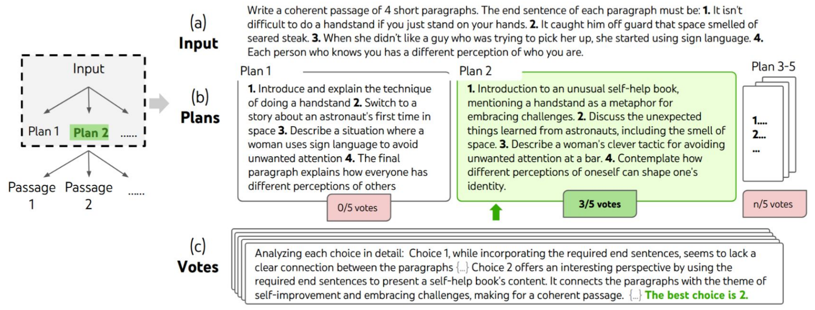
Thought Evaluation
(13 - 10) * 13 = 3 * 13 = 39
10 + 13 + 13 = 36 There is no way to obtain 24 with these big numbers. impossible

Figure 2: ToT in a game of 24. The LM is prompted for (a) thought generation and (b) valuation.

Technical Detail - Experiments

TASKS:

- Game of 24
- Creative Writing
  - Open-ended generation with global coherence constraints.
- 5×5 Mini Crosswords
  - Lexical reasoning with strong combinatorial structure.



Write a coherent passage of 4 short paragraphs. The end sentence of each paragraph must be: **1.** It isn't difficult to do a handstand if you just stand on your hands. **2.** It caught him off guard that space smelled of seared steak. **3.** When she didn't like a guy who was trying to pick her up, she started using sign language. **4.** Each person who knows you has a different perception of who you are.

1. Introduce and explain the technique of doing a handstand **2.** Switch to a story about an astronaut's first time in space **3.** Describe a situation where a woman uses sign language to avoid unwanted attention **4.** The final paragraph explains how everyone has different perceptions of others

1. Introduction to an unusual self-help book, mentioning a handstand as a metaphor for embracing challenges. **2.** Discuss the unexpected things learned from astronauts, including the smell of space. **3.** Describe a woman's clever tactic for avoiding unwanted attention at a bar. **4.** Contemplate how different perceptions of oneself can shape one's identity.

Analyzing each choice in detail: Choice 1, while incorporating the required end sentences, seems to lack a clear connection between the paragraphs {...} Choice 2 offers an interesting perspective by using the required end sentences to present a self-help book's content. It connects the paragraphs with the theme of self-improvement and embracing challenges, making for a coherent passage. {...} **The best choice is 2.**

Figure 4: A step of deliberate search in a randomly picked Creative Writing task. Given the input, the LM samples 5 different plans, then votes 5 times to decide which plan is best. The majority choice is used to consequently write the output passage with the same sample-vote procedure.

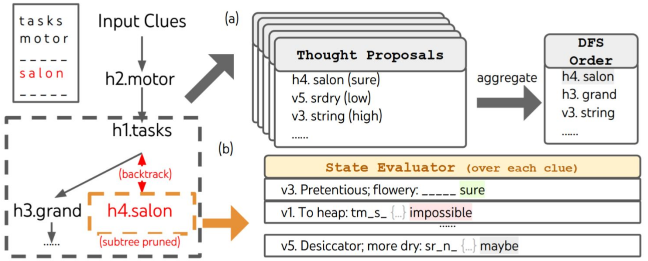


Figure 6: In Mini Crosswords, (a) how thoughts are proposed and aggregated in a priority queue for depth-first search (DFS), and (b) how a state is evaluated based on the possibility of filling in each remaining word clue, and pruned if any remaining clue is deemed not possible to fill by the LM. Then DFS backtracks to the parent state and explore the next promising thought for clue.

Technical Detail - Experiments

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)	**74%**
IO + Refine (k=10)	27%
IO (best of 100)	33%
CoT (best of 100)	49%

Table 2: Game of 24 Results.

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

Table 3: Mini Crosswords results.

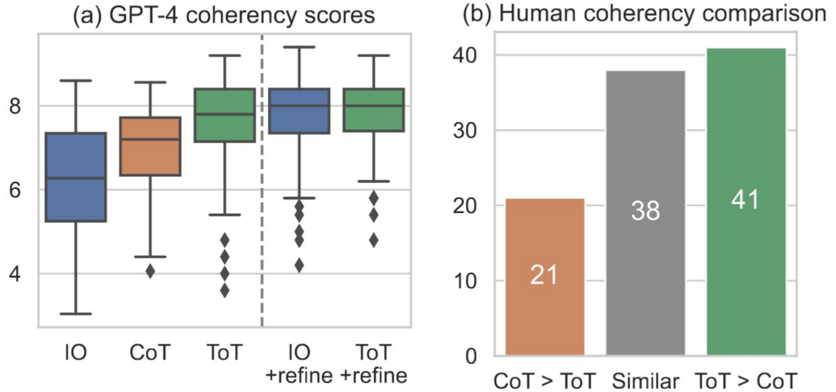


Figure 5: Creative Writing results.

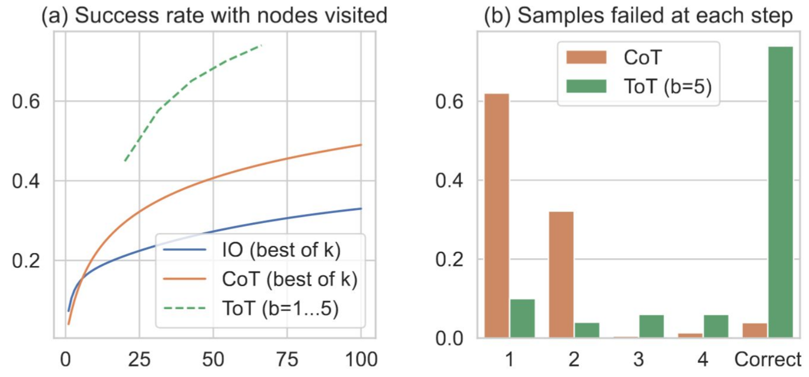


Figure 3: Game of 24 (a) scale analysis & (b) error analysis.

Yao, Shunyu, et al. "Tree of thoughts: Deliberate problem solving with large language models, 2023." *URL https://arxiv. org/abs/2305.10601* 3 (2023): 1.

03

Major Contributions

Major Contributions

- The paper introduces **Tree of Thoughts**, a tree-structured inference framework that enables deliberate search over intermediate thoughts, generalizing Chain-of-Thought prompting.
- It formulates problem solving as **thought-level generation and evaluation**, where the language model evaluates partial solutions to guide search.
- The approach demonstrates that **inference-time search alone**, without additional training, can substantially improve performance on tasks requiring planning or exploration.

04

Limitations

Limitations

- Inference-time cost:
  - Tree of Thoughts requires generating and evaluating multiple thoughts, which increases inference cost compared to linear decoding.
- Dependence on LM self-evaluation:
  - Search quality depends on the language model's ability to evaluate partial solutions, which can be noisy or imperfect.
- Task-specific design choices:
  - Effective use of ToT requires task-dependent choices for thought decomposition, evaluation strategy, and search algorithm.

05

Future Directions

Future work

- More advanced search algorithms:
    - Explore integrating ToT with A*, Monte Carlo Tree Search, or other heuristic search methods.
- Learning better heuristics:
    - Combine ToT with learned or hybrid evaluation strategies to improve state evaluation.
- Broader task coverage and efficiency:
    - Study how ToT scales to larger or more complex tasks, and how to reduce inference cost.

Paper 3

LEAST-TO-MOST PROMPTING ENABLES COMPLEX REASONING IN LARGE LANGUAGE MODELS

01

Background

Why CoT struggles with easy-to-hard generalization ?

Problem Setting: Easy-to-Hard Generalization

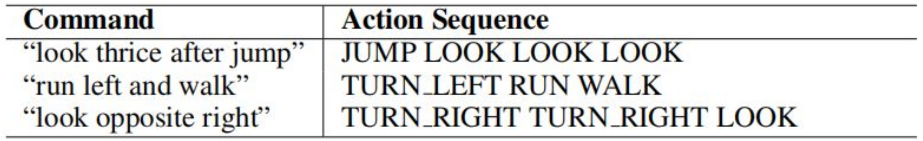
What works (often)

- Few-shot prompting can solve many tasks from a handful of demonstrations.
- Chain-of-Thought (CoT) adds intermediate rationales and improves reasoning on many benchmarks.
- Self-consistency can further boost CoT by sampling multiple reasoning paths.

What breaks

- CoT can fail when test problems are harder than prompt exemplars.
- Harder often means longer reasoning chains or more compositional structure.
- Goal: prompt-time method that extrapolates from simple exemplars to harder instances.

Why Decomposition Helps



Command	Action Sequence
"look thrice after jump"	JUMP LOOK LOOK LOOK
"run left and walk"	TURN_LEFT RUN WALK
"look opposite right"	TURN_RIGHT TURN_RIGHT LOOK

Table 5: Example commands in SCAN and their corresponding action sequences. An agent successfully executes a natural language command by performing its corresponding action sequence.

https://arxiv.org/abs/2205.10625

- Humans routinely solve complex problems by breaking them into simpler subproblems.
- SCAN maps natural-language commands to action sequences; the length split tests generalization to longer sequences.
- Subproblem answers become reusable "building blocks" for subsequent steps.
- Many benchmarks explicitly stress this ability: length generalization (SCAN), iterative symbolic tasks, and multi-step math.
- Least-to-most prompting adapts this educational principle to in-context learning (decompose → solve iteratively).

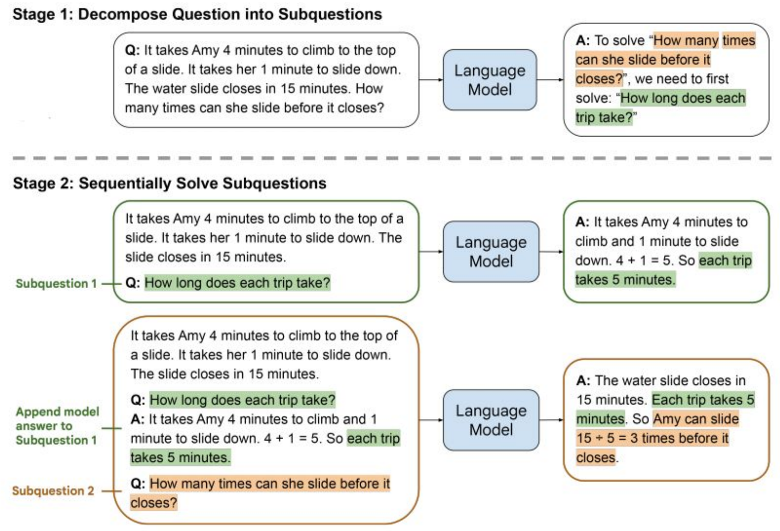
02

Technical Detail

Two-stage prompting: Decompose → Solve sequentially

Least-to-Most Prompting: Two Sequential Stages

- Stage 1 — Decomposition: prompt includes examples of how to decompose; model outputs a list of subproblems.
- Stage 2 — Subproblem solving: prompt includes (a) solution examples, (b) solved subproblems + answers so far, (c) next subproblem.
- Iterate Stage 2 until the original problem (appended last) is solved.



Stage 1: Decompose Question into Subquestions
Q: It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The water slide closes in 15 minutes. How many times can she slide before it closes?
Language Model
A: To solve "How many times can she slide before it closes?", we need to first solve: "How long does each trip take?"
Stage 2: Sequentially Solve Subquestions
It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.
Subquestion 1
Q: How long does each trip take?
Language Model
A: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.
It takes Amy 4 minutes to climb to the top of a slide. It takes her 1 minute to slide down. The slide closes in 15 minutes.
Q: How long does each trip take?
Append model answer to Subquestion 1
A: It takes Amy 4 minutes to climb and 1 minute to slide down. 4 + 1 = 5. So each trip takes 5 minutes.
Subquestion 2
Q: How many times can she slide before it closes?
Language Model
A: The water slide closes in 15 minutes. Each trip takes 5 minutes. So Amy can slide 15 ÷ 5 = 3 times before it closes.

Figure 1: Least-to-most prompting solving a math word problem in two stages: (1) query the language model to decompose the problem into subproblems; (2) query the language model to sequentially solve the subproblems. The answer to the second subproblem is built on the answer to the first subproblem. The demonstration examples for each stage's prompt are omitted in this illustration.

https://arxiv.org/abs/2205.10625

Key intuition: use earlier answers as context so later steps "inherit" solved structure (like recursion / iteration).

Prompt Design Patterns in L2M

Decomposition prompt

- Fixed exemplars show "how to decompose".
- Model outputs an ordered list of subproblems.
- Designed per task/domain (e.g., SCAN commands vs math word problems).

Solving prompt (iterative)

- Fixed exemplars show "how to solve subproblems".
- Append solved (subproblem, answer) pairs as running context.
- Ask the next subproblem; repeat until final answer.
- Some tasks can merge stages into one pass (e.g., GSM8K prompt).

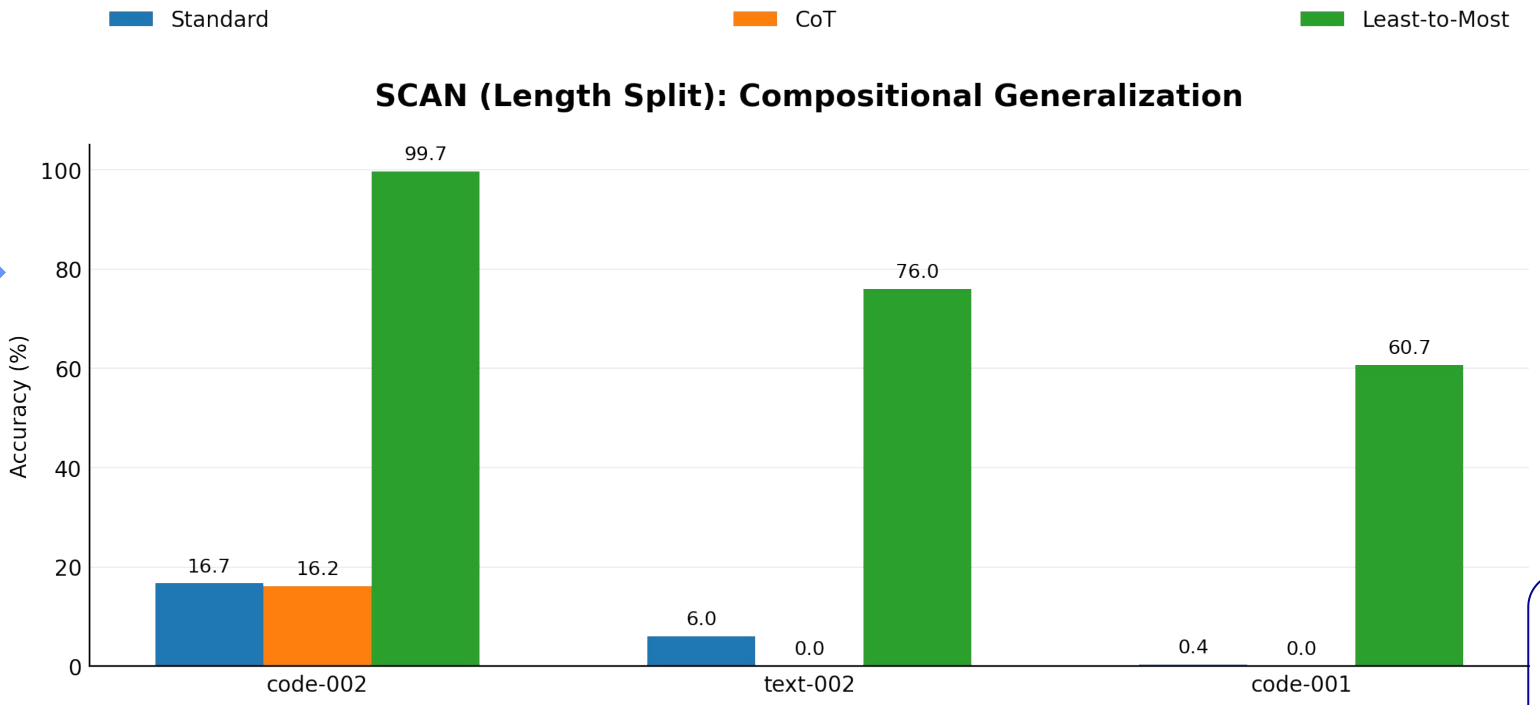
03

Major Contributions

Empirical gains on compositional & iterative reasoning

Compositional Generalization: SCAN (Length Split)

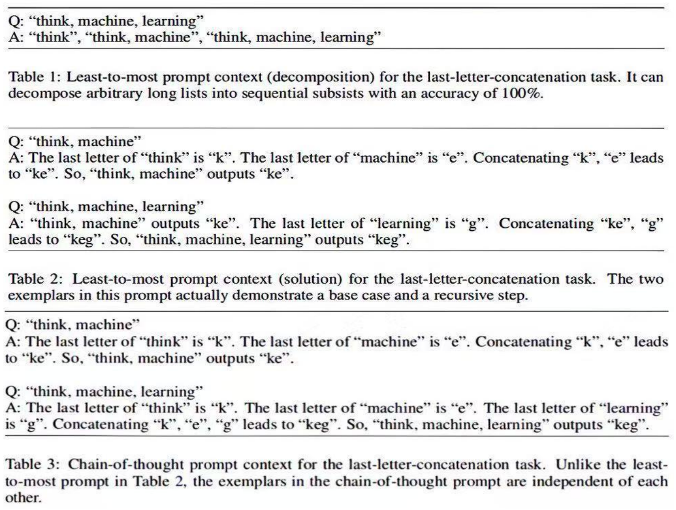
Least-to-most reaches 99.7% on code-davinci-002; standard/CoT ≈ 16%.



Beyond SCAN: Iteration & Multi-step Math

Symbolic manipulation (last-letter)

- Task: concatenate last letters across a word list (last-letter-concatenation).
- CoT degrades rapidly as list length grows; L2M stays robust at long lengths (e.g., L=12).
- Key idea: reuse intermediate answers via a base case + recursive step (Tables 1–3).



Q: “think, machine, learning”
A: “think”, “think, machine”, “think, machine, learning”

Table 1: Least-to-most prompt context (decomposition) for the last-letter-concatenation task. It can decompose arbitrary long lists into sequential subsists with an accuracy of 100%.

Q: “think, machine”
A: The last letter of “think” is “k”. The last letter of “machine” is “e”. Concatenating “k”, “e” leads to “ke”. So, “think, machine” outputs “ke”.

Q: “think, machine, learning”
A: “think, machine” outputs “ke”. The last letter of “learning” is “g”. Concatenating “ke”, “g” leads to “keg”. So, “think, machine, learning” outputs “keg”.

Table 2: Least-to-most prompt context (solution) for the last-letter-concatenation task. The two exemplars in this prompt actually demonstrate a base case and a recursive step.

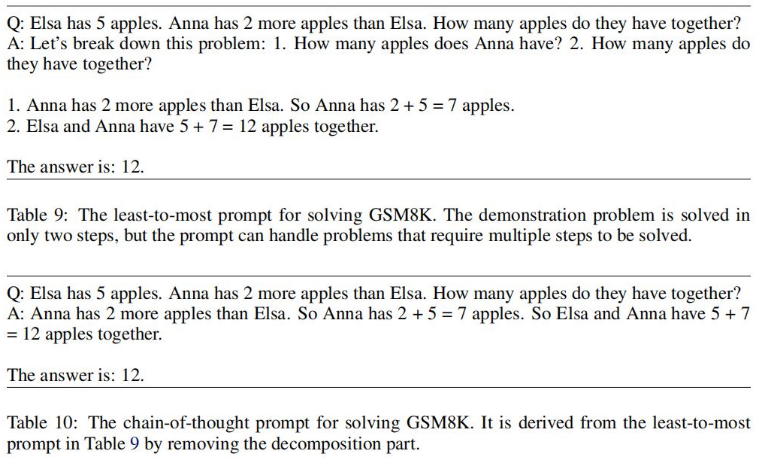
Q: “think, machine”
A: The last letter of “think” is “k”. The last letter of “machine” is “e”. Concatenating “k”, “e” leads to “ke”. So, “think, machine” outputs “ke”.

Q: “think, machine, learning”
A: The last letter of “think” is “k”. The last letter of “machine” is “e”. The last letter of “learning” is “g”. Concatenating “k”, “e”, “g” leads to “keg”. So, “think, machine, learning” outputs “keg”.

Table 3: Chain-of-thought prompt context for the last-letter-concatenation task. Unlike the least-to-most prompt in Table 2, the exemplars in the chain-of-thought prompt are independent of each other.

Math reasoning (GSM8K / DROP)

- Overall GSM8K: small gain vs CoT; largest gains on harder (≥5-step) problems.
- DROP numerical subset: L2M improves substantially on non-football / football subsets.
- L2M works as “decompose → solve sequentially”; CoT is derived by removing decomposition (Tables 9–10).



Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together?
A: Let’s break down this problem: 1. How many apples does Anna have? 2. How many apples do they have together?

1. Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples.
2. Elsa and Anna have 5 + 7 = 12 apples together.

The answer is: 12.

Table 9: The least-to-most prompt for solving GSM8K. The demonstration problem is solved in only two steps, but the prompt can handle problems that require multiple steps to be solved.

Q: Elsa has 5 apples. Anna has 2 more apples than Elsa. How many apples do they have together?
A: Anna has 2 more apples than Elsa. So Anna has 2 + 5 = 7 apples. So Elsa and Anna have 5 + 7 = 12 apples together.

The answer is: 12.

Table 10: The chain-of-thought prompt for solving GSM8K. It is derived from the least-to-most prompt in Table 9 by removing the decomposition part.

https://arxiv.org/abs/2205.10625

04

Limitations

Decomposition is the bottleneck

Limitation 1: Decomposition Does Not Transfer Easily

- Decomposition prompts often fail to generalize across domains.
- A math decomposition prompt may not help common sense questions (e.g., “Did Aristotle use a laptop?”).
- Even within a domain (e.g., GSM8K), the hard part is finding the right decomposition.
- Strong results on SCAN / last-letter partly come from “easy” decomposition structure in those tasks.

Limitation 2: Error Propagation & Typical Failure Modes

Iterative symbolic errors

- Last-letter task: errors dominated by concatenation mistakes (drop/add letters), not letter identification.
- Template confusion can occur (base vs extension step) but decreases with more exemplars.
- Longer sequences increase opportunities for copy/concatenation errors.

SCAN errors (length split)

- Misapplying "twice/thrice" to expressions, especially after "around".
- Interpreting "after" as "and" when composing two sub-expressions.
- Errors can arise in either decomposition or translation stages.

05

Future Directions

From one-way prompting to interactive learning

Direction 1: Toward Bidirectional, Feedback-Driven Prompting

- The paper argues prompting is often “one-way” instruction without model feedback.
- A natural next step is to evolve prompting into bidirectional conversations with immediate feedback.
- Least-to-most is a step in this direction: it decomposes, solves, and re-prompts using intermediate outputs.

Direction 2: Stronger Decomposition & Hybrid Prompting

Improve decomposition quality

- Design domain-specific decomposition exemplars more systematically.
- Use multiple candidate decompositions and select the best (prompt-level search).
- Reduce error propagation by validating intermediate sub-answers.

Hybridize with other prompting methods

- Combine L2M with chain-of-thought and/or self-consistency when helpful.
- For some tasks, merge the two stages into a single-pass prompt to reduce cost.
- Explore concise intermediate representations (e.g., SCAN uses Python notation).

Summary

Comparison between three papers

Title	Chain-of-Thought Prompting Elicits Reasoning in Large Language Models	Tree of Thoughts: Deliberate Problem Solving with Large Language Models	Least-to-Most Prompting Enables Complex Reasoning in Large Language Models
Year	2023	2023	2023
Main Idea	<input, CoT, output> few-shot exemplars	Treat reasoning as a search over a tree of “thoughts”	Two-stage prompting: decomposition + solve from easier to harder
Key Results	Improves arithmetic/ commonsense/ symbolic reasoning	ToT outperforms CoT on search/planning-heavy tasks	Improves compositional generalization
May Limitation	CoT ≠ “reasoning” Requires manual exemplar annotation Benefit for large models	Higher computation and API-call cost	Depends on producing correct decompositions

Sources

https://arxiv.org/abs/2201.11903
https://arxiv.org/abs/2305.10601
https://arxiv.org/abs/2205.10625

Q & A

Thank you

Mengzi Cheng (cdd6yg),
Yi Ping (efk4ps)
Zhenghao He(wff7ad)