Language Model Reasoning

Ihab Tabbara
Lev Rose
Zheyuan Wu

September 10, 2024

Agenda

- Chain of Thought
- Least-to-Most Prompting
- Self-Consistency
- Tree of Thought
- Graph of Thoughts

Chain of Thought

[Chain of Thought Prompting Elicits Reasoning in Large Language Models]

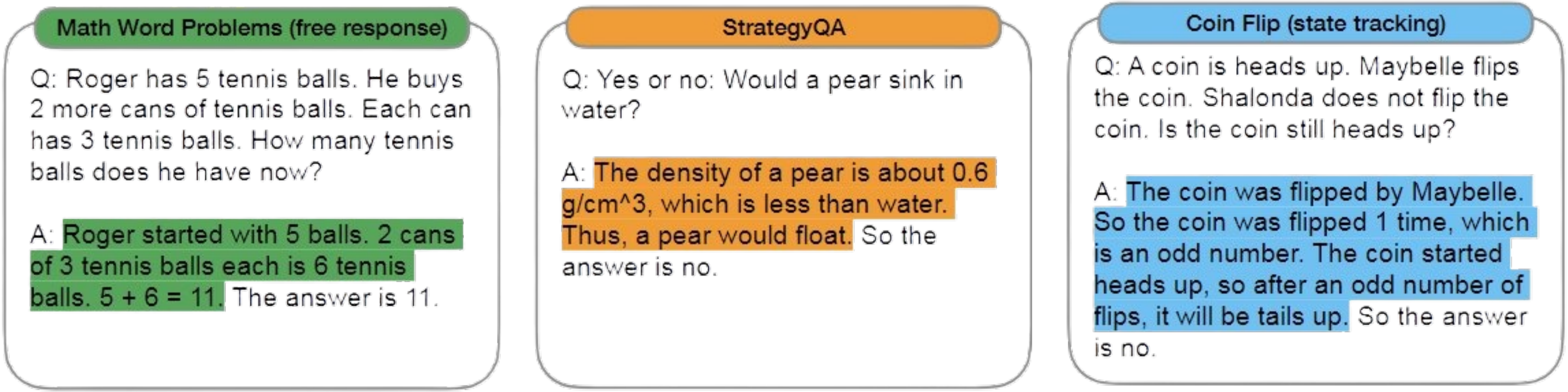
improves LLM to perform complex reasoning

Background

- Techniques for arithmetic reasoning can benefit from generating natural language rationales that lead to the final answer.
- LLMs offer the exciting prospect of in-context few-shot learning via prompting.

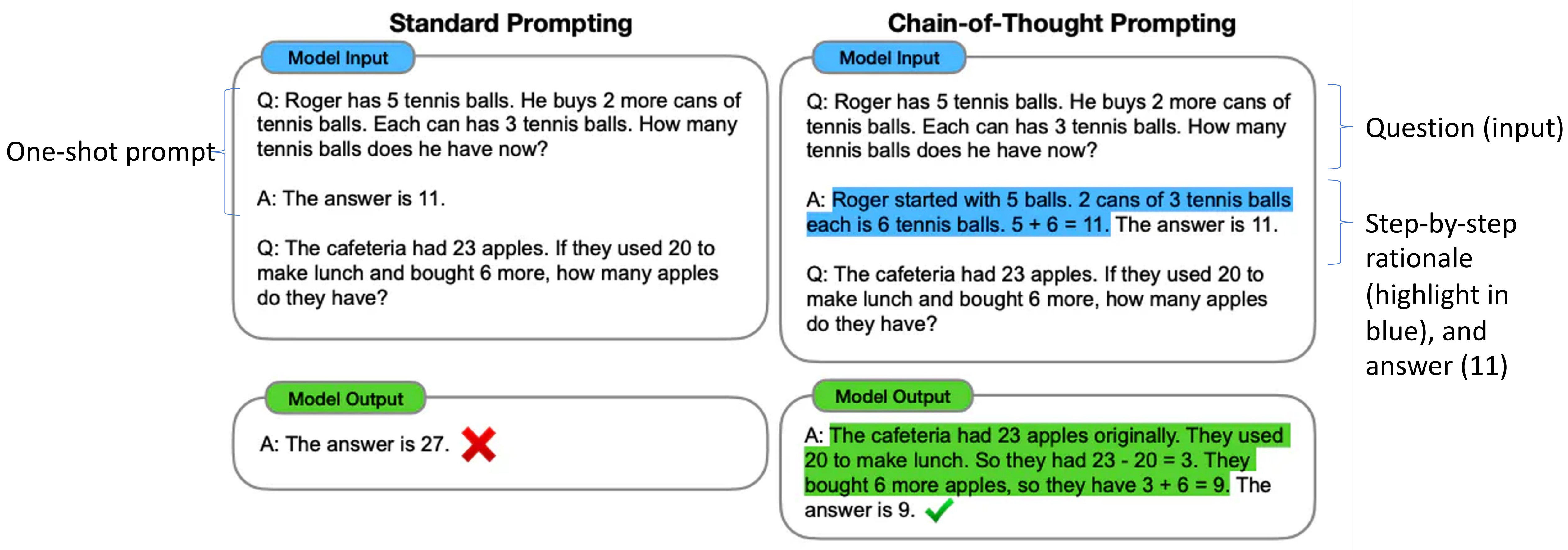
Proposed solution

- we explore the ability of language models to perform few-shot prompting for reasoning tasks, given a prompt that consists of triples:
  - ⟨input, chain of thought, output⟩



Chain of Thought Prompting (CoT)

Piecing together tokens to create longer answers



Model Choice for Experiment

GPT-3 *		LaMDA		PaLM	
Model Name	Model Param	Model Name	Model Param	Model Name	Model Param
text-ada-001	350M	LaMDA 442M	442M	PaLM 8B	8B
text-babbage-001	1.3B	LaMDA 2B	2B	PaLM 62B	62B
text-curie-001	6.7B	LaMDA 8B	8B	PaLM 137B	137B
text-davinci-002	175B	LaMDA 68B	68B		
		LaMDA 137B	137B		

- The parameter count for GPT-3 is estimated based on (Ouyang et al., 2022)
- Codex (Chen et al., 2021, code-davinci-002 in the OpenAI API) and UL2 20B model are also used in the experiment but not representative for showing the trends since they don't have models with different sizes, but the effects of CoT still applies.

Results (symbolic reasoning)

We use the following two toy tasks.

- Last letter concatenation.
  - This task asks the model to concatenate the last letters of words in a name
  - (e.g.,“Amy Brown” → “yn”).
- Coin flip.
  - This task asks the model to answer whether a coin is still heads up after people either flip or don’t flip the coin
  - (e.g., “A coin is heads up. Phoebe flips the coin. Osvaldo does not flip the coin. Is the coin still heads up?”→ “no”).

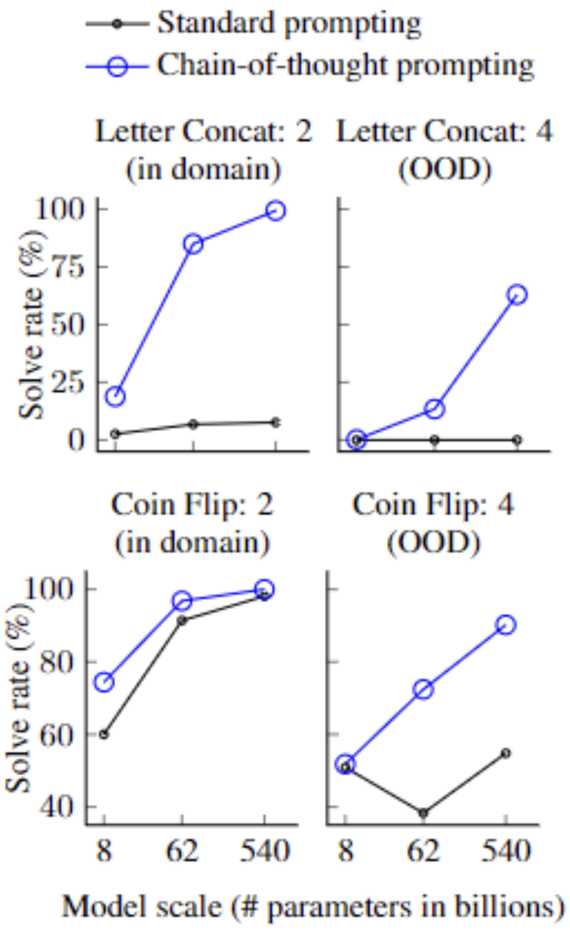


Figure 8: Using chain-of-thought prompting facilitates generalization to longer sequences in two symbolic reasoning tasks.

Results (math problems)

- Larger models gain more from the CoT prompting
- LLM gains more from CoT prompting for complex problems.

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).

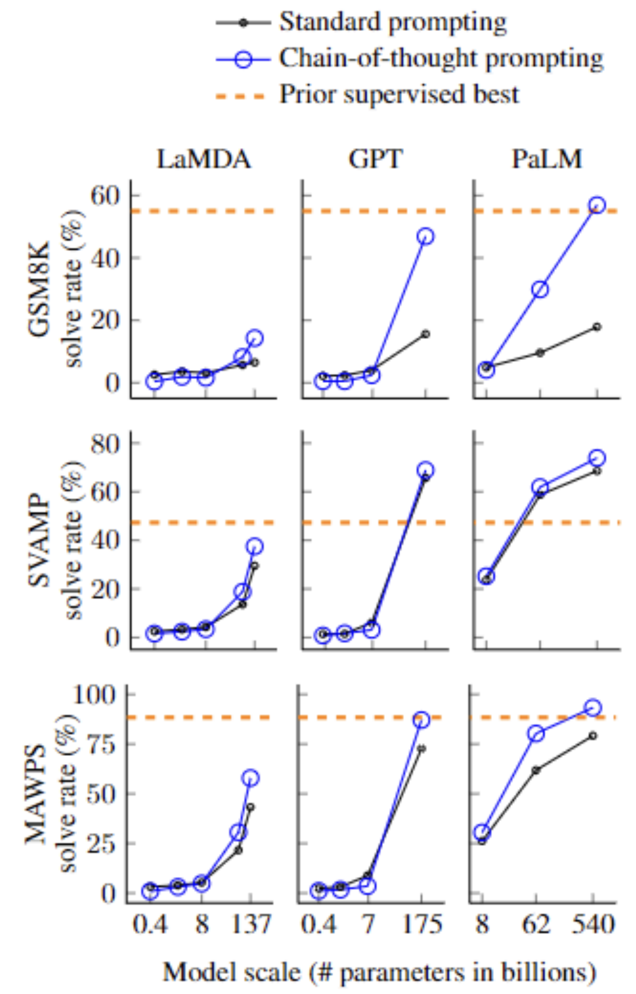


Figure 4: Chain-of-thought prompting enables large language models to solve challenging math problems. Notably, chain-of-thought reasoning is an emergent ability of increasing model scale. Prior best numbers are from Cobbe et al. (2021) for GSM8K, Jie et al. (2022) for SVAMP, and Lan et al. (2021) for MAWPS.

Results (commonsense)

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).

Date Understanding

Q: The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY?

A: One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.

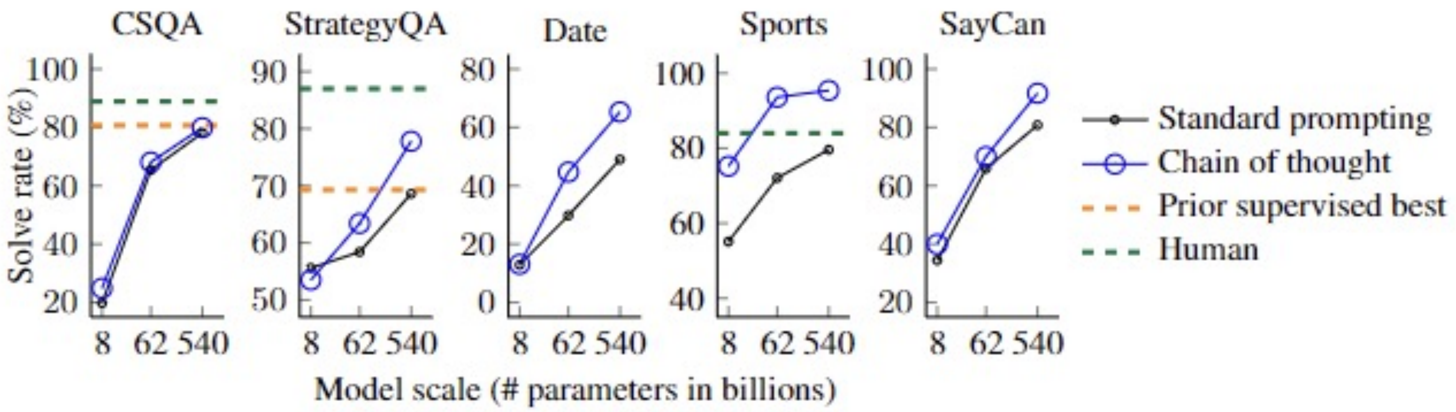


Figure 7: Chain-of-thought prompting also improves the commonsense reasoning abilities of language models. The language model shown here is PaLM. Prior best numbers are from the leaderboards of CSQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021) (single-model only, as of May 5, 2022). Additional results using various sizes of LaMDA, GPT-3, and PaLM are shown in Table 4.

[2] We sample examples ≤ 60 tokens to fit into our input context window, and also limit the examples to ≤ 2 steps to solve for a fair comparison with the eight exemplars that we composed.

Limitations

- We don’t know if LLM is actually “reasoning” like humans.
- Cost of prompting is high for human supervisors.
- Reasoning path can both leads to correct and incorrect answers.
- Don’t know if CoT can also scales to smaller models with less “emergence”.

Agenda

- Chain of Thought
- Least-to-Most Prompting
- Self-Consistency
- Tree of Thought
- Graph of Thoughts

Least-to-Most Prompting

[Least-to-Most Prompting Enables Complex Reasoning in Large Language Models]

Enabling Complex Reasoning in Language Models

Background & Problems to solve

- Large data required for training
- Explainability, machine learning is essentially a black box
- LMs can only solve problem typically at the same level of difficulty as the training sets
- Chain-of-thought prompting has a key limitation—it often performs poorly on tasks that require generalization

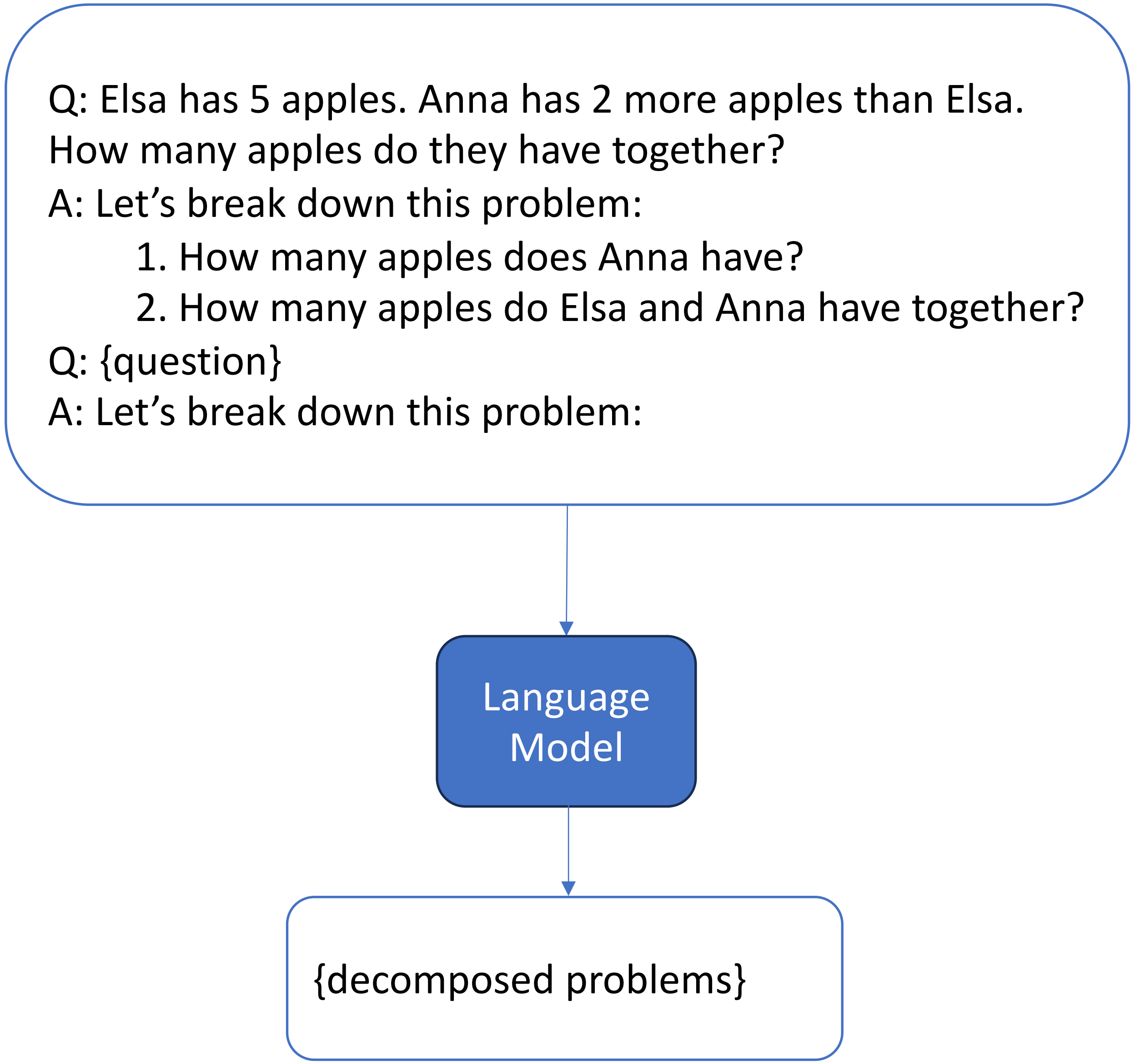
Proposed solution

Least-to-most prompting

- (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

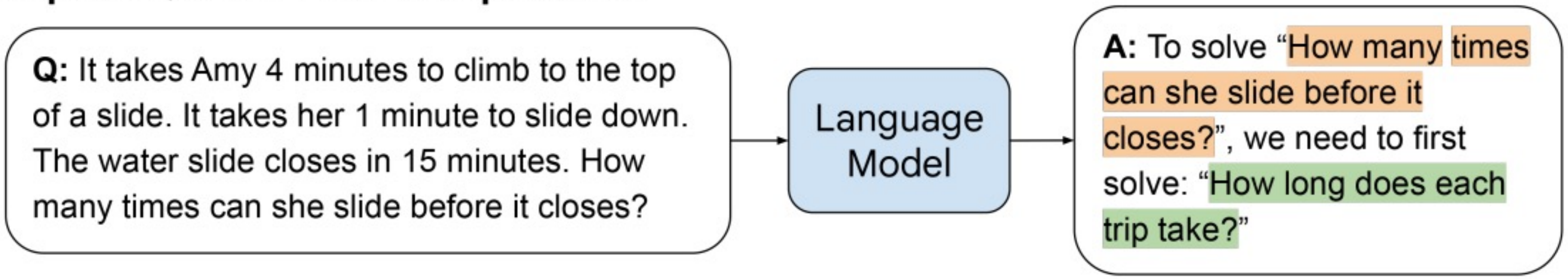
Decomposition

- First decomposing a complex problem into a list of easier subproblems.
- The prompt in this stage contains constant examples that demonstrate the decomposition, followed by the specific question to be decomposed.

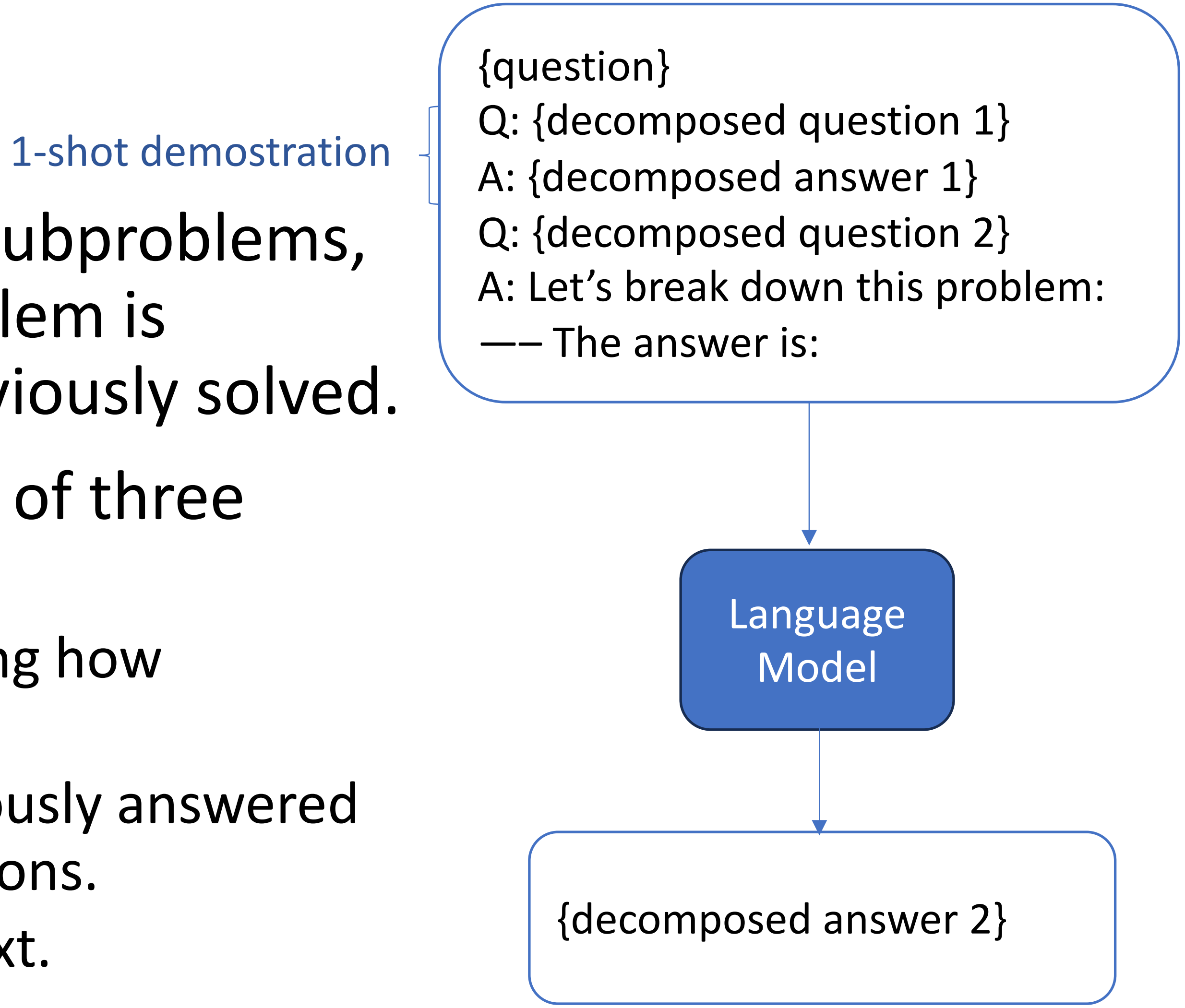


Decomposition

Stage 1: Decompose Question into Subquestions

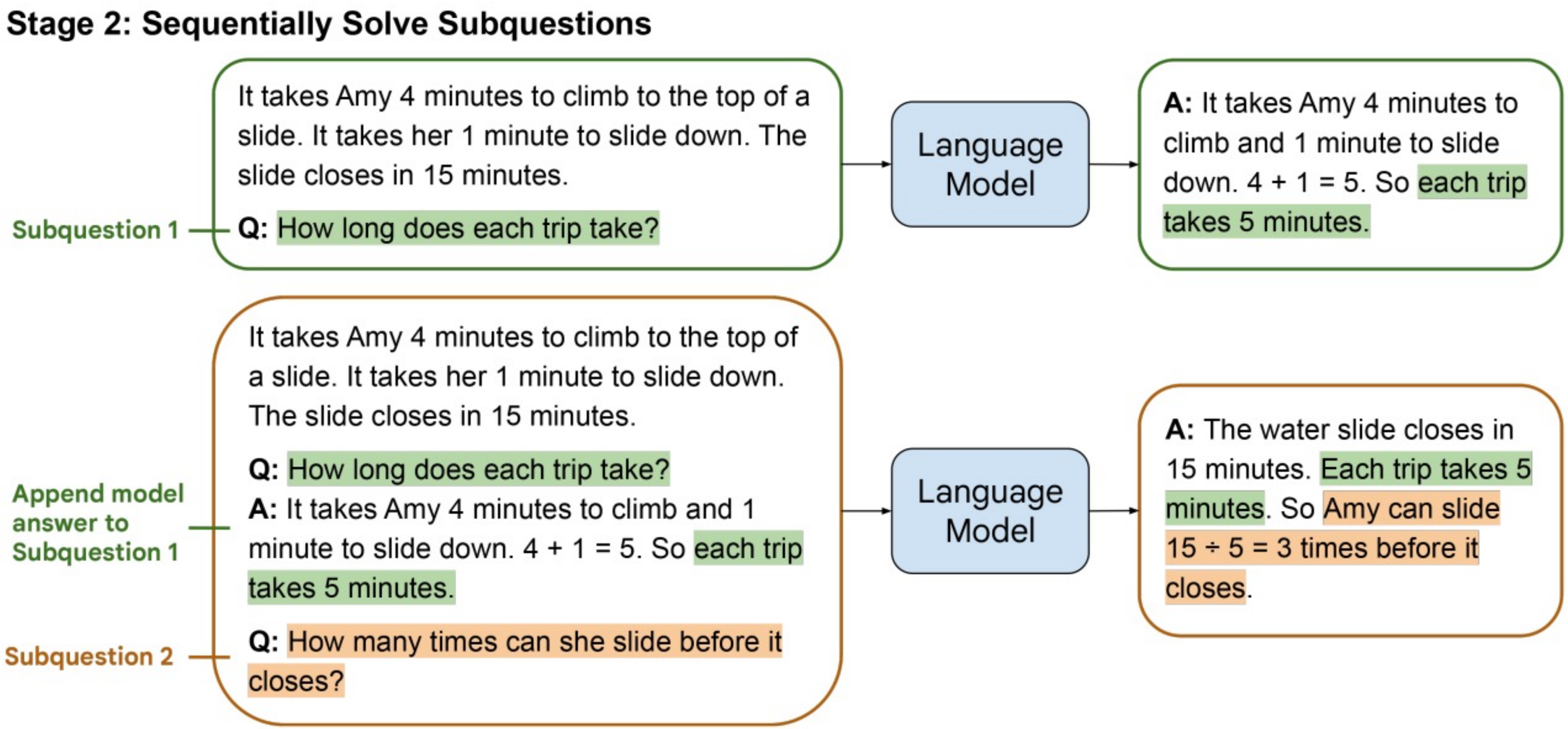


Subproblem solving



- Then sequentially solving these subproblems, whereby solving a given subproblem is facilitated by the answers to previously solved.
- The prompt in this stage consists of three parts:
  - 1. constant examples demonstrating how subproblems are solved (N-shots)
  - 2. a potentially empty list of previously answered subquestions and generated solutions.
  - 3. the question to be answered next.

Subproblem Solving



Results

- Symbolic manipuation

Decomposition

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%.

Subproblem solving

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.

	$L = 4$	$L = 6$	$L = 8$	$L = 10$	$L = 12$
Standard prompting	0.0	0.0	0.0	0.0	0.0
Chain-of-Thought	84.2	69.2	50.2	39.8	31.8
Least-to-Most	**94.0**	**88.4**	**83.0**	**76.4**	**74.0**

Results

- Compositional generalization

Decomposition

Subproblem solving

"jump around left thrice and walk opposite right"

Golden:
TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_LEFT JUMP TURN_RIGHT TURN_RIGHT WALK

Decomposition:
Q: "jump around left thrice and walk opposite right"
A: "jump around left thrice" can be solved by: "jump left", "jump around left", "jump around left thrice". "walk opposite right" can be solved by: "walk opposite right". So, "jump around left thrice and walk opposite right" can be solved by: "jump left", "jump around left", "jump around left thrice", "walk opposite right".

Solution:
Q: "jump left"
A: The output of "jump left" concatenates: the output of "turn left", the output of "jump". "turn left" outputs "TURN_LEFT". "jump" outputs "JUMP". So concatenating the output of "turn left" and the output of "jump" leads to "TURN_LEFT" + "JUMP". So the output of "jump left" is "TURN_LEFT" + "JUMP".

Q: "jump around left"
A: The output of "jump around left" concatenates: the output of "jump left", the output of "jump left", the output of "jump left", the output of "jump left". "jump left" outputs "TURN_LEFT" + "JUMP". So repeating the output of "jump around left" four times leads to ("TURN_LEFT" + "JUMP") * 4. So the output of "jump around left" is ("TURN_LEFT" + "JUMP") * 4.

...

Method	Standard prompting	Chain-of-Thought	Least-to-Most
`code-davinci-002`	16.7	16.2	**99.7**
`text-davinci-002`	6.0	0.0	**76.0**
`code-davinci-001`	0.4	0.0	**60.7**

Table 8: Accuracies (%) of different prompting methods on the test set of SCAN under length split. The results of `text-davinci-002` are based on a random subset of 100 commands.

Results

- Math reasoning

Decomposition

Q: Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice 30 years old, how old is Kody?
A: To answer the question "How old is Kody?", we need to know: "How old is Mohamed?", "How old was Mohamed four years ago?", "How old was Kody four years ago?".

Subproblem solving

Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice 30 years old, how old is Kody?

Q: How old was Mohamed four years ago?
A: We were told that Mohamed is currently twice 30 years old, so he is currently 30 * 2 = 60 years old. That means that four years ago he must have been 60 - 4 = 56 years old. The answer is 56.

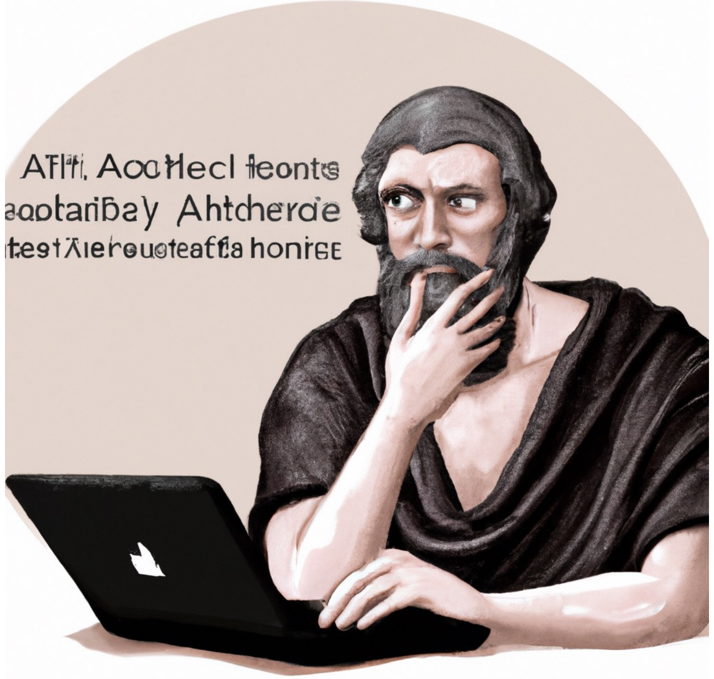
Q: How old is Kody?
A: Four years ago, Kody was half as old as Mohamed, so Kody must have been 56 / 2 = 28 years old then. Since Kody was 28 years old four years ago, she must now be 28 + 4 = 32 years old. The answer is 32.

Method	Non-football (DROP)	Football (DROP)	GSM8K
Zero-Shot	43.86	51.77	16.38
Standard prompting	58.78	62.73	17.06
Chain-of-Thought	74.77	59.56	60.87
Least-to-Most	**82.45**	**73.42**	**62.39**

Table 11: Accuracies (%) of different prompting methods on GSM8K and DROP (only the subset containing numerical problems). The base language model is `code-davinci-002`.

Limitations

- Decomposition prompts typically don't generalize well across different domains.
- Generalizing decomposition can even be difficult within the same domain.



https://blog.raph.ws/2023/05/aristotle-wouldnt-think-chatgpt-is-intelligent/

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \cdots$$

Prove that: The real part of every nontrivial zero of the Riemann zeta function is ½.

Take away

- Least to most promting is a useful technique for increasing the performance of LLM in questions that requires generalization and
  - Decompostion
  - Subproblem solving
- Decomposition prompts typically don’t generalize well across different domains.
- Generalizing decomposition can even be difficult within the same domain.

Agenda

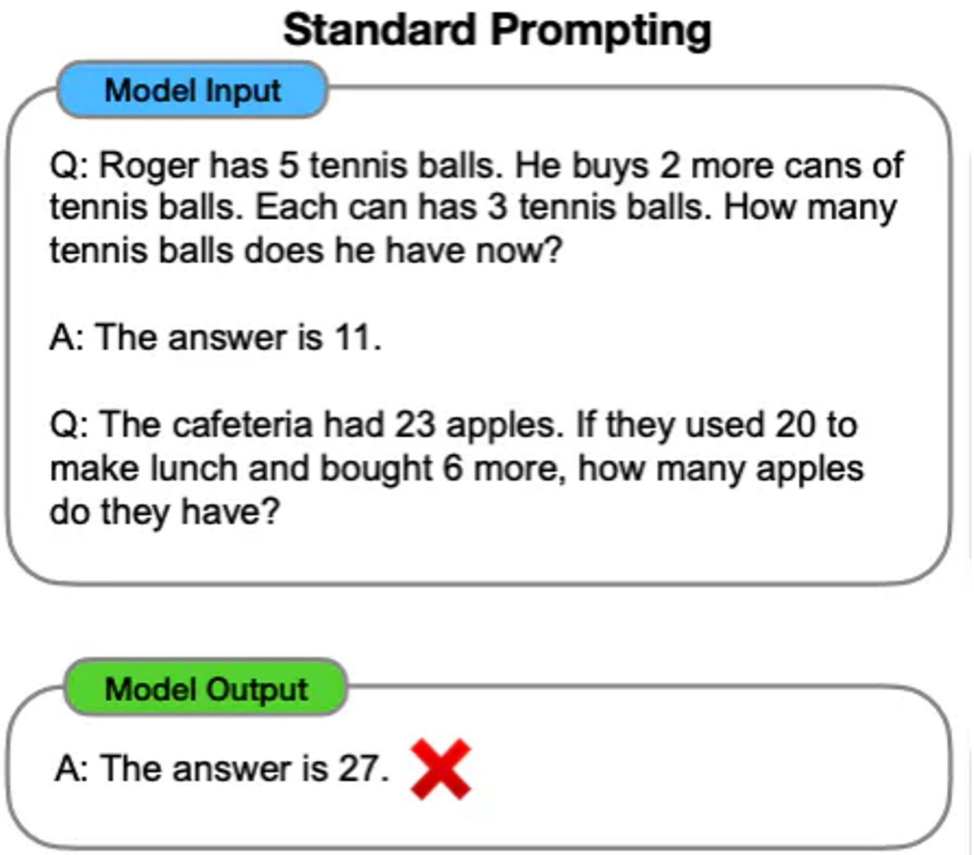
- Chain of Thought
- Least-to-Most Prompting
- Self-Consistency
- Tree of Thought
- Graph of Thoughts

Self-Consistency

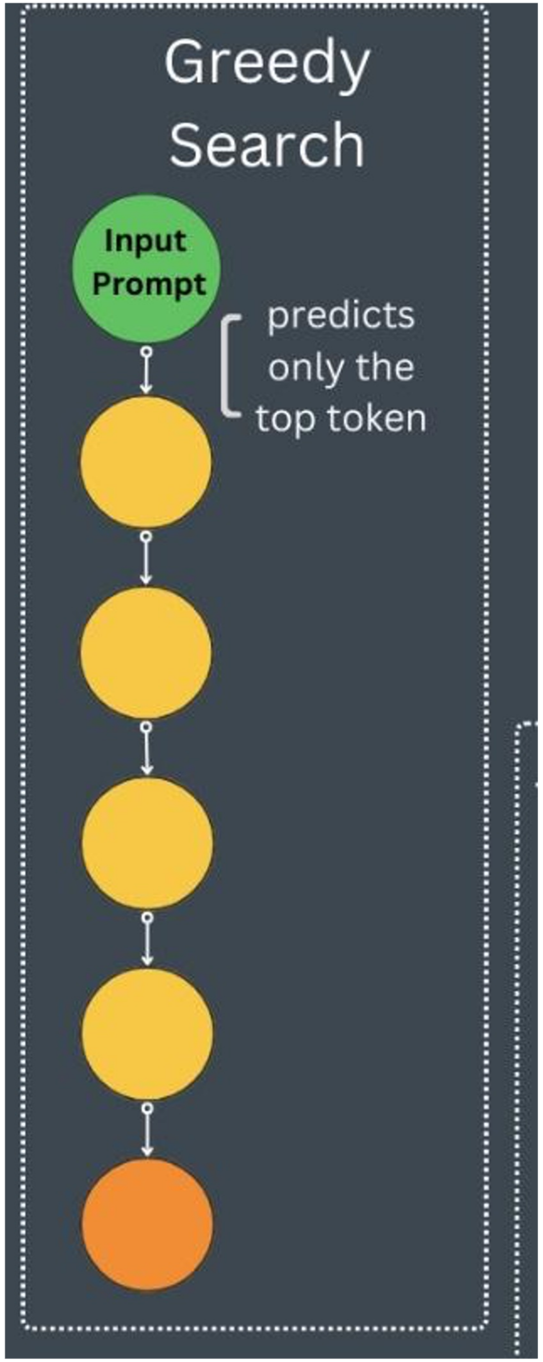
[Self-Consistency: Improvement of Chain of Thought Reasoning]

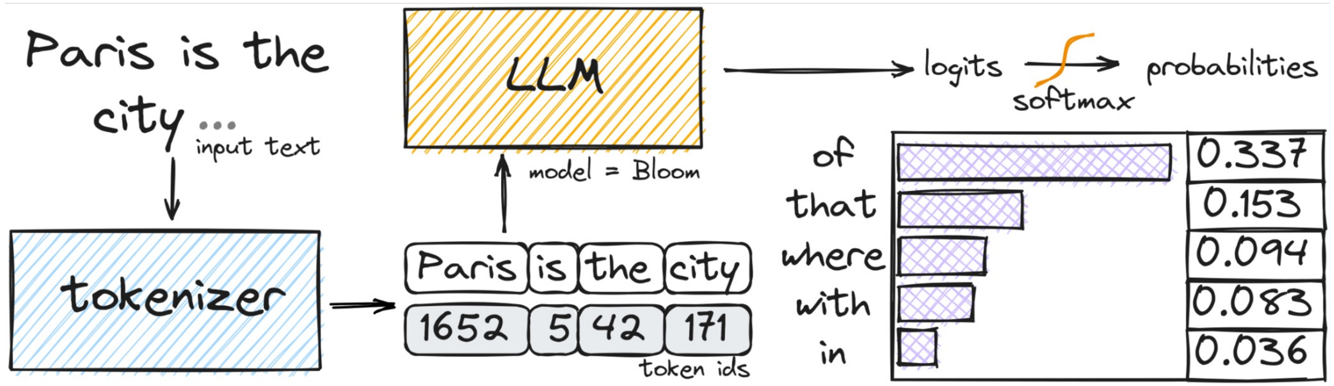
Improvement of Chain of Thought Reasoning

Background

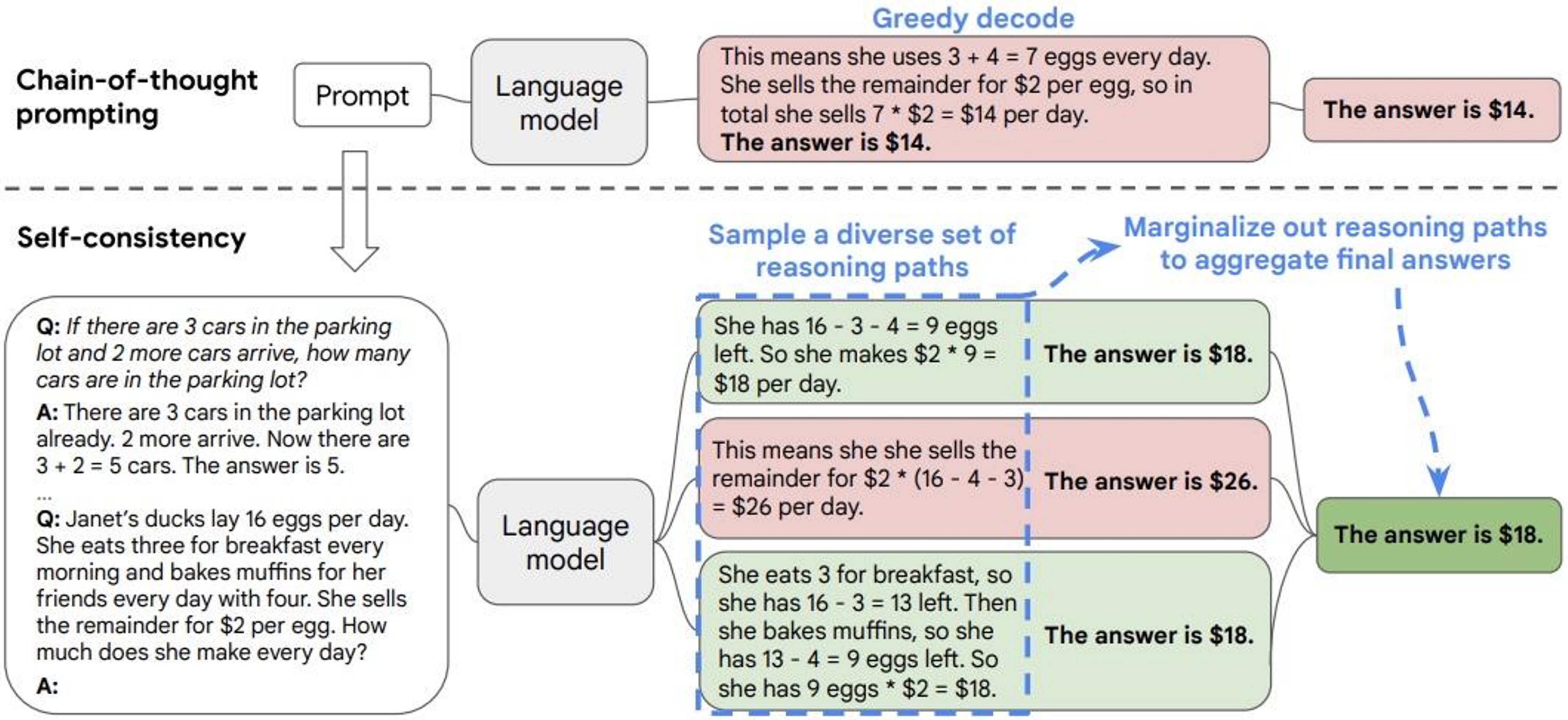


Standard Prompting
Model Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The answer is 27.

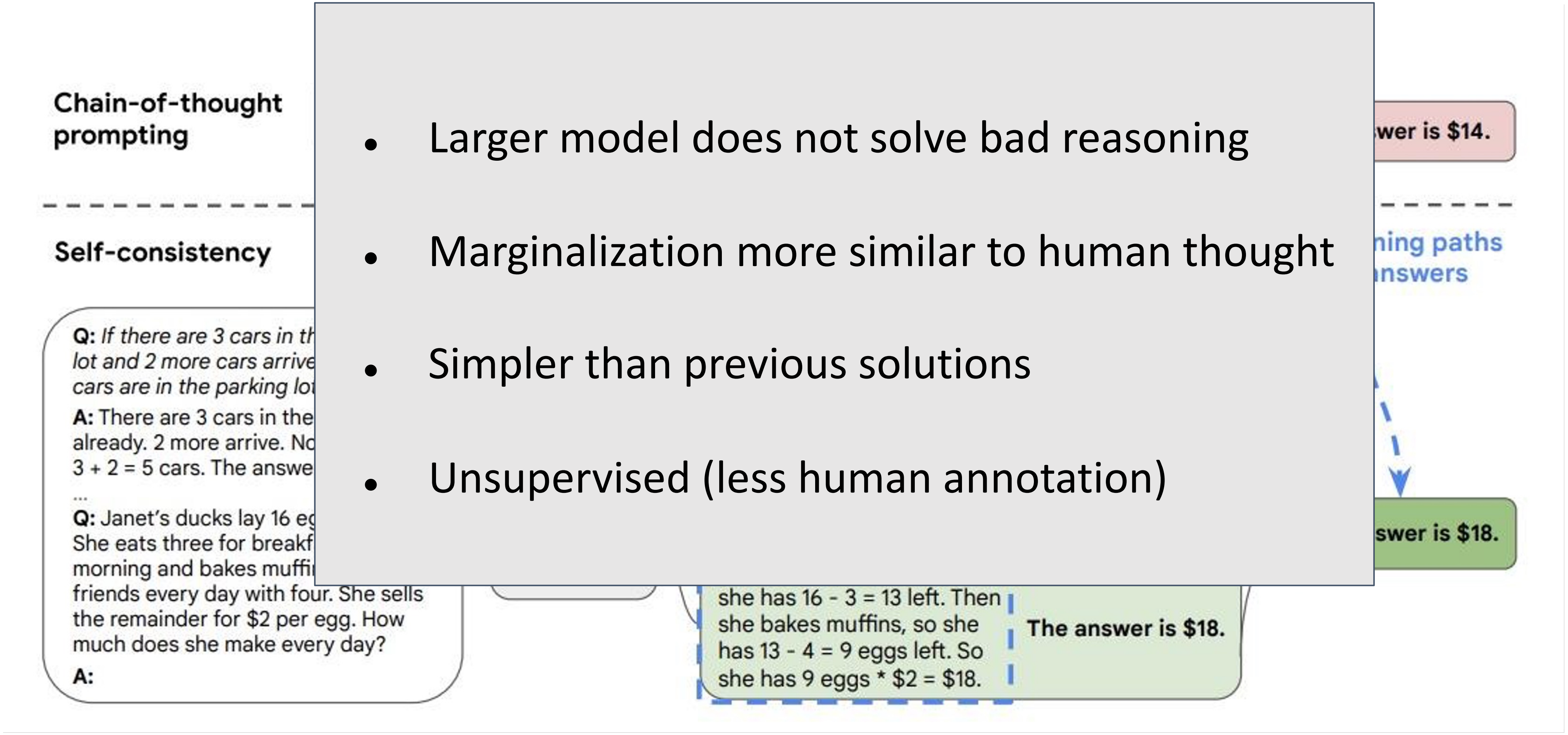

Greedy Search
Input Prompt
predicts only the top token


Paris is the city ...
input text
tokenizer
LLM
model = Bloom
Paris
is
the
city
1652
5
42
171
token ids
logits
softmax
probabilities
of
that
where
with
in
0.337
0.153
0.094
0.083
0.036

Self-Consistency



Self-Consistency



- Larger model does not solve bad reasoning
- Marginalization more similar to human thought
- Simpler than previous solutions
- Unsupervised (less human annotation)

Picking most consistent option from weights of different answers

- Hypothesis: lead to more correct answers

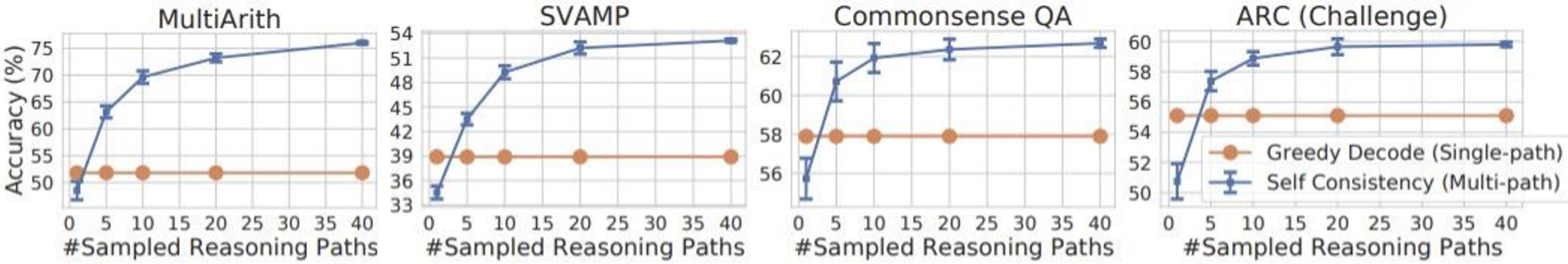


Figure 2: Self-consistency (blue) significantly improves accuracy over CoT-prompting with greedy decoding (orange) across arithmetic and commonsense reasoning tasks, over LaMDA-137B. Sampling a higher number of diverse reasoning paths consistently improves reasoning accuracy.

Process

r : thought path

a : answer

"Most Consistent"

$$(r_i, a_i)\ ,\ i\ =\ 1 \ldots m$$

$$argmax_a \sum_{i=1}^{m} \mathbb{1}(a_i\ =\ a)$$

Results - Arithmetic

	Method	AddSub	MultiArith	ASDiv	AQuA	SVAMP	GSM8K
	Previous SoTA	**94.9**[a]	60.5[a]	75.3[b]	37.9[c]	57.4[d]	35[e] / 55[g]
UL2-20B	CoT-prompting	18.2	10.7	16.9	23.6	12.6	4.1
	Self-consistency	24.8 (+6.6)	15.0 (+4.3)	21.5 (+4.6)	26.9 (+3.3)	19.4 (+6.8)	7.3 (+3.2)
LaMDA-137B	CoT-prompting	52.9	51.8	49.0	17.7	38.9	17.1
	Self-consistency	63.5 (+10.6)	75.7 (+23.9)	58.2 (+9.2)	26.8 (+9.1)	53.3 (+14.4)	27.7 (+10.6)
PaLM-540B	CoT-prompting	91.9	94.7	74.0	35.8	79.0	56.5
	Self-consistency	93.7 (+1.8)	99.3 (+4.6)	81.9 (+7.9)	48.3 (+12.5)	86.6 (+7.6)	74.4 (+17.9)
GPT-3 Code-davinci-001	CoT-prompting	57.2	59.5	52.7	18.9	39.8	14.6
	Self-consistency	67.8 (+10.6)	82.7 (+23.2)	61.9 (+9.2)	25.6 (+6.7)	54.5 (+14.7)	23.4 (+8.8)
GPT-3 Code-davinci-002	CoT-prompting	89.4	96.2	80.1	39.8	75.8	60.1
	Self-consistency	91.6 (+2.2)	**100.0** (+3.8)	**87.8** (+7.6)	**52.0** (+12.2)	**86.8** (+11.0)	**78.0** (+17.9)

Table 2: Arithmetic reasoning accuracy by self-consistency compared to chain-of-thought prompting (Wei et al., 2022). The previous SoTA baselines are obtained from: *a*: Relevance and LCA operation classifier (Roy & Roth, 2015), *b*: Lan et al. (2021), *c*: Amini et al. (2019), *d*: Pi et al. (2022), *e*: GPT-3 175B finetuned with 7.5k examples (Cobbe et al., 2021), *g*: GPT-3 175B finetuned plus an additional 175B verifier (Cobbe et al., 2021). The best performance for each task is shown in bold.

Results - Commonsense and Symbolic Reasoning

	Method	CSQA	StrategyQA	ARC-e	ARC-c	Letter (4)	Coinflip (4)
	Previous SoTA	**91.2**[a]	73.9[b]	86.4[c]	75.0[c]	N/A	N/A
UL2-20B	CoT-prompting	51.4	53.3	61.6	42.9	0.0	50.4
	Self-consistency	55.7 (+4.3)	54.9 (+1.6)	69.8 (+8.2)	49.5 (+6.8)	0.0 (+0.0)	50.5 (+0.1)
LaMDA-137B	CoT-prompting	57.9	65.4	75.3	55.1	8.2	72.4
	Self-consistency	63.1 (+5.2)	67.8 (+2.4)	79.3 (+4.0)	59.8 (+4.7)	8.2 (+0.0)	73.5 (+1.1)
PaLM-540B	CoT-prompting	79.0	75.3	95.3	85.2	65.8	88.2
	Self-consistency	80.7 (+1.7)	**81.6** (+6.3)	**96.4** (+1.1)	**88.7** (+3.5)	70.8 (+5.0)	91.2 (+3.0)
GPT-3 Code-davinci-001	CoT-prompting	46.6	56.7	63.1	43.1	7.8	71.4
	Self-consistency	54.9 (+8.3)	61.7 (+5.0)	72.1 (+9.0)	53.7 (+10.6)	10.0 (+2.2)	75.9 (+4.5)
GPT-3 Code-davinci-002	CoT-prompting	79.0	73.4	94.0	83.6	70.4	99.0
	Self-consistency	81.5 (+2.5)	79.8 (+6.4)	96.0 (+2.0)	87.5 (+3.9)	**73.4** (+3.0)	**99.5** (+0.5)

Table 3: Commonsense and symbolic reasoning accuracy by self-consistency compared to chain-of-thought prompting (Wei et al., 2022). The previous SoTA baselines are obtained from: *a*: DeBERTaV3-large + KEAR (Xu et al., 2021b), *b*: Chowdhery et al. (2022), *c*: UnifiedQA-FT (Khashabi et al., 2020). The best performance for each task is shown in bold.

Robust to Scaling

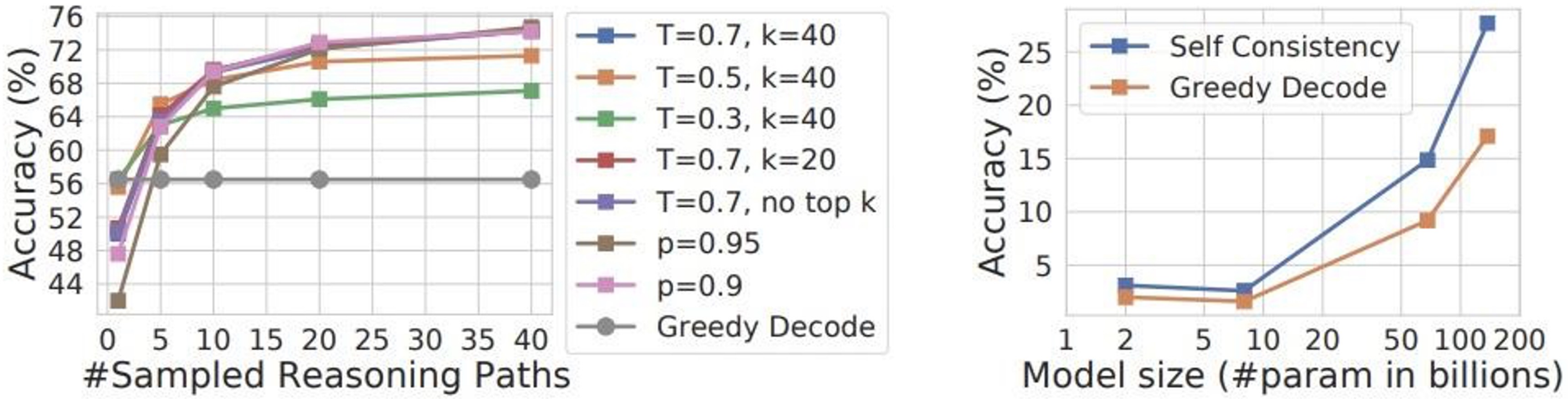


Figure 4: GSM8K accuracy. (Left) Self-consistency is robust to various sampling strategies and parameters. (Right) Self-consistency improves performance across language model scales.

Robust to Imperfect Prompts

LaMDA-137B	Prompt with correct chain-of-thought	17.1
	Prompt with imperfect chain-of-thought	14.9
	+ Self-consistency (40 paths)	**23.4**
	Prompt with equations	5.0
	+ Self-consistency (40 paths)	**6.5**
PaLM-540B	Zero-shot CoT (Kojima et al., 2022)	43.0
	+ Self-consistency (40 paths)	**69.2**

Table 8: Self-consistency works under imperfect prompts, equation prompts and zero-shot chain-of-thought for GSM8K.

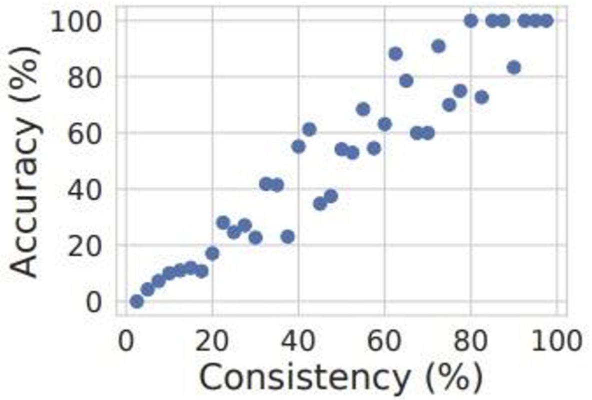


Figure 5: The consistency is correlated with model's accuracy.

Robust to Imperfect Prompts

LaMDA-137B	Prompt with correct chain-of-thought	17.1
	Prompt with imperfect chain-of-thought	14.9
	+ Self-consistency (40 paths)	**23.4**
	Prompt with equations	5.0
	+ Self-consistency (40 paths)	**6.5**
PaLM-540B	Zero-shot CoT (Kojima et al., 2022)	43.0
	+ Self-consistency (40 paths)	**69.2**

Table 8: Self-consistency works under imperfect prompts, equation prompts and zero-shot chain-of-thought for GSM8K.

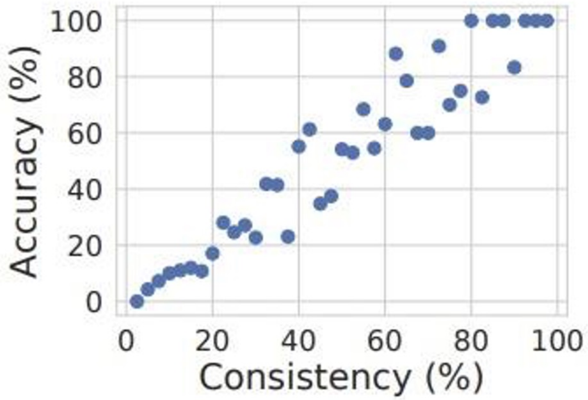


Figure 5: The consistency is correlated with model's accuracy.

Limitations

- Computationally expensive
- Sometimes create nonsensical reasoning paths

Take Away

- Improves both arithmetic and commonsense accuracy
- Improves collection of rationales and providing uncertainty estimates
- Improves responses to imperfect prompts
- Robust to scaling
- More expensive to check paths

Agenda

- Chain of Thought
- Least-to-Most Prompting
- Self-Consistency
- Tree of Thought
- Graph of Thoughts

Tree of Thought

[Tree of Thought: Problem Solving with Large Language Models]

Problem Solving with Large Language Models

Background and Motivation

How LLM’s think:

- LLMs use an autoregressive mechanism for text generation.
- They make token-level decisions one by one.
- Process occurs in a left-to-right fashion.

Background and Motivation

How LLM’s think:

- LLMs use an autoregressive mechanism for text generation.
- They make token-level decisions one by one.
- Process occurs in a left-to-right fashion.

What Are They Good For:

- Text Completion
- Translation
- Summarization
- Question Answering

Background and Motivation

How LLM’s think:

- LLMs use an autoregressive mechanism for text generation.
- They make token-level decisions one by one.
- Process occurs in a left-to-right fashion.

What Are They Good For:

- Text Completion
- Translation
- Summarization
- Question Answering

Current Limitations

- Struggle with tasks that require complex reasoning, planning, and problem-solving.
- Can not revisit or change any previous decisions

Is such a simple mechanism sufficient for a LM to be built toward a general problem solver? If not, what should be alternative mechanisms?

Background and Motivation

- Inspiration from human cognition?

Humans 2 systems of thinking

System 1: fast, automatic, unconscious

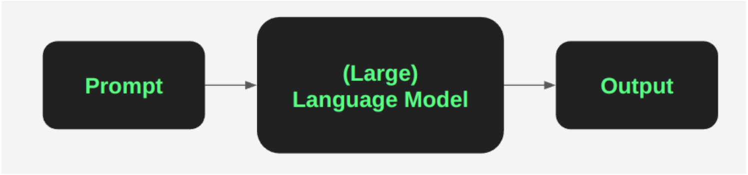
System 2: slow, deliberate, conscious

How do we allow a model to think like humans and solve complex tasks efficiently?

The authors propose a new way of prompting models in a tree like manner

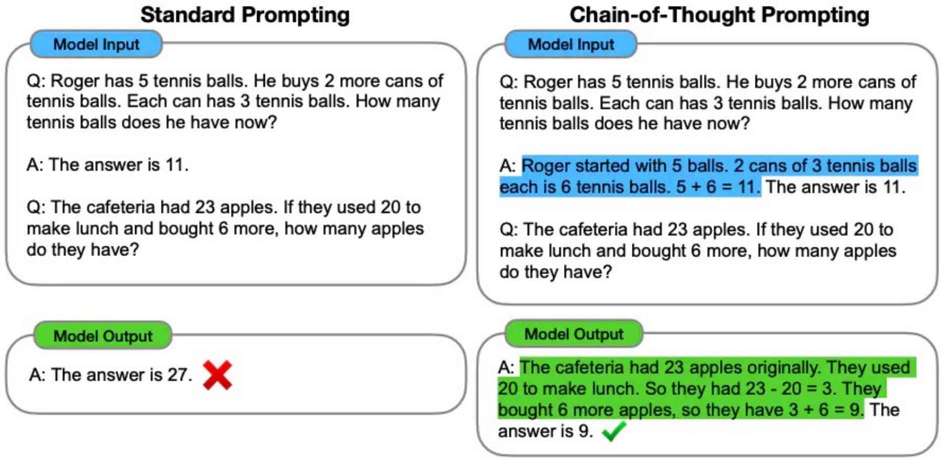
Background and Motivation

- Input-output (IO) prompting



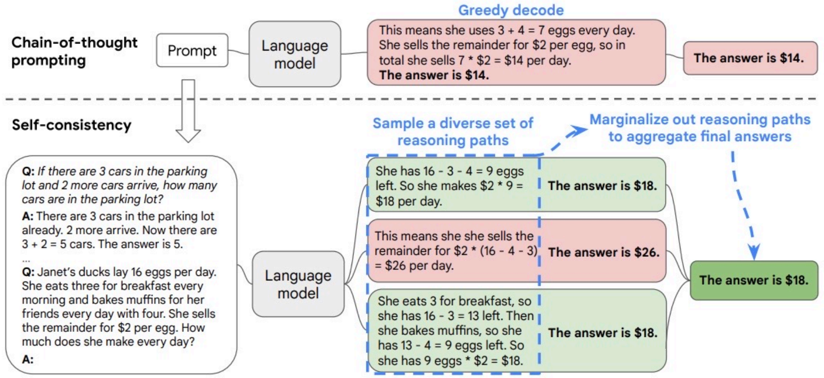
- Chain-of-thought (CoT) prompting

Breaks down reasoning into intermediate steps to solve non-trivial problems



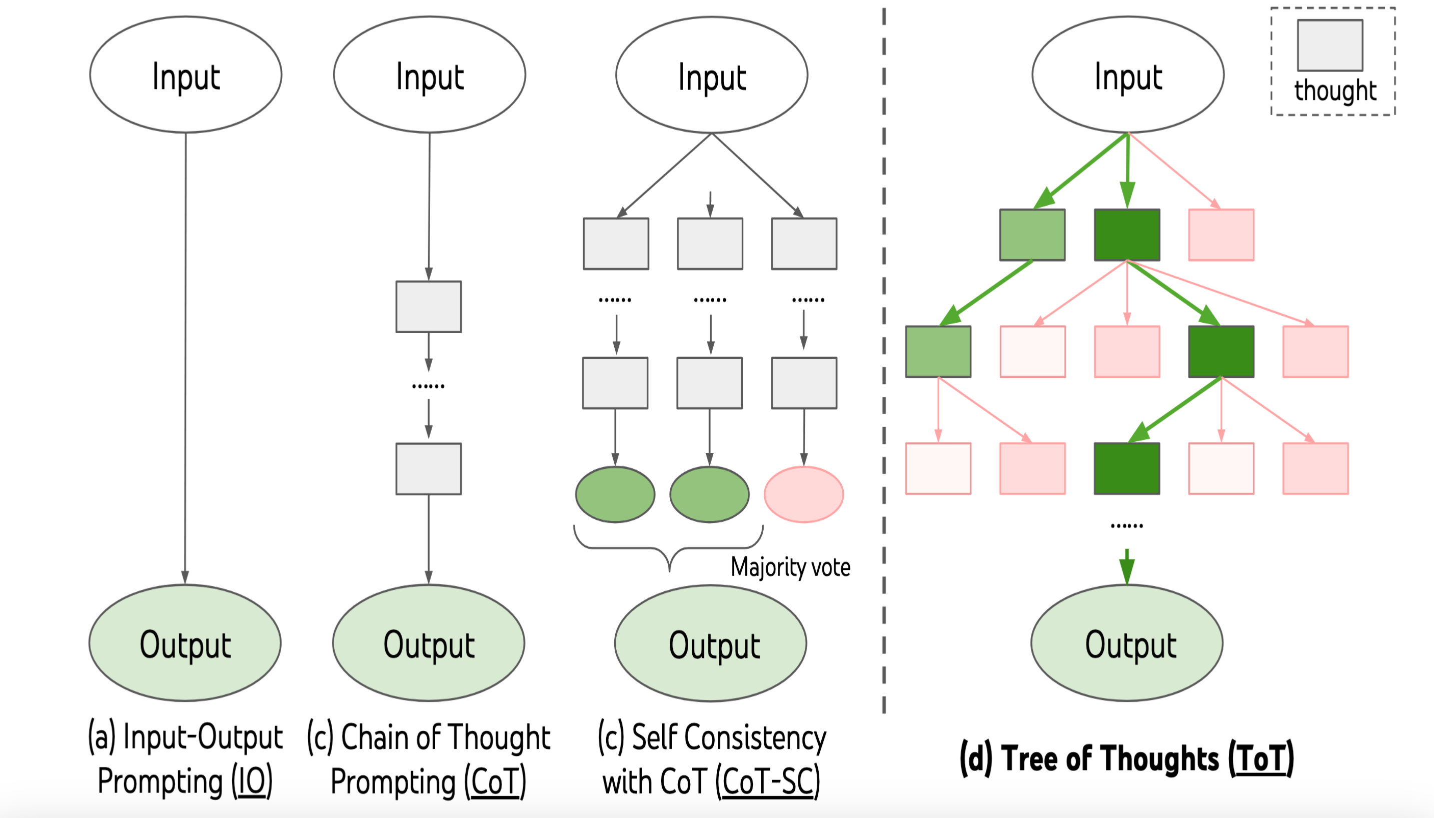
- Self-consistency with CoT

Samples multiple reasoning paths and selects the most frequent output



Tree of Thoughts

solutions are incrementally built by exploring different reasoning paths.



•**Nodes**: Each node represents a state, which is a partial solution made up of a collection of thoughts (intermediate reasoning steps).

•**Branches:** Each branch represents a potential step or decision that leads to a new state (a new collection of thoughts).

•**State evaluation:** States are evaluated using heuristics to estimate which paths are most promising for reaching a final solution.

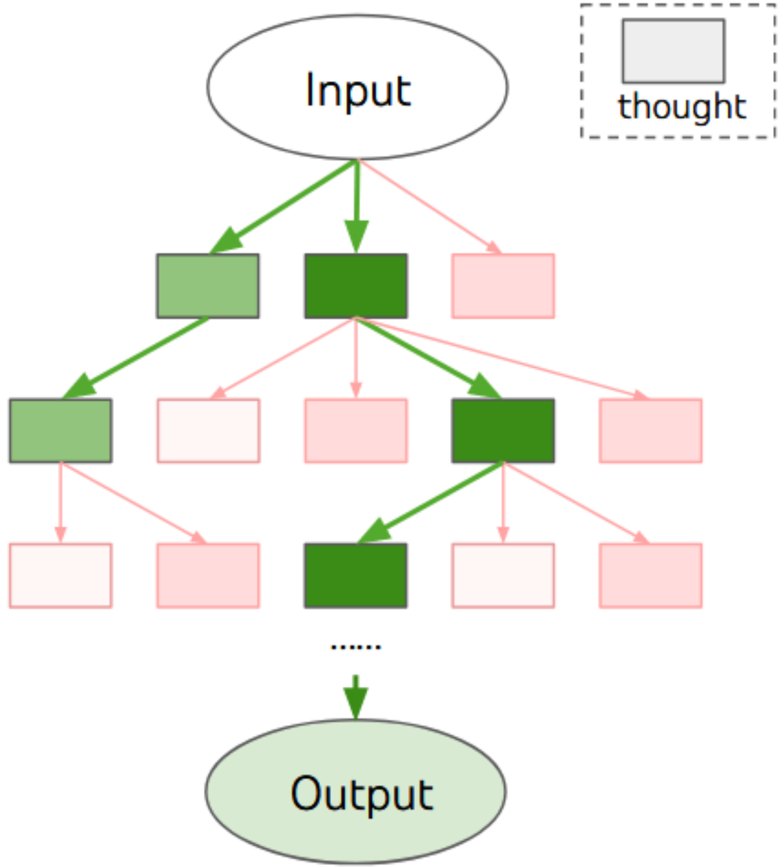
Tree of Thoughts - How does it work?

Thought Generation: Creating multiple potential solutions.

New state: Combine the current state with the new thoughts to form a new state

State Evaluation: Assessing the progress of each solution.

Search Algorithm: Choosing the best paths to follow.



Tree of Thoughts - Experiments

	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)

Tree of Thoughts - Experiments

	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)

1 G	2 R	3 A	4 D	5 E
6 L	O	V	E	D
7 A	M	O	N	G
8 R	A	I	S	E
9 E	N	D	E	D

Tree of Thoughts – Thought Generation

•**State S:** The current state is defined as S=[x,z1…zi] where x is the problem and zi are thoughts leading to the state.

Two strategies to generate thoughts based on the richness of the thought space.

<u>**Thought Generation from CoT Prompt:**</u>
Sample thoughts in i.i.d manner from CoT prompt. Works for rich thought spaces
Case of Creative Writing : In thought 1, LM makes a brief plan then write the passage, then for thought 2 LM Writes first paragraph…
Case of game of 24:
the thoughts could be given input “4 9 10 13”, “13 - 9 = 4 (left: 4 4 10); 10 - 4 = 6 (left: 4 6); 4 * 6 = 24 (left: 24)

Tree of Thoughts – Thought Generation

•**State S:** The current state is defined as S=[x,z1…zi] where x is the problem and zi are thoughts leading to the state.

Two strategies to generate thoughts based on the richness of the thought space.

Thought Generation from CoT Prompt:
Sample thoughts in i.i.d manner from CoT prompt. Works for rich thought spaces
Case of Creative Writing : In thought 1, LM makes a brief plan then write the passage, then for thought 2 LM writes first paragraph…
Case of game of 24:
the thoughts could be given input "4 9 10 13", "13 - 9 = 4 (left: 4 4 10); 10 - 4 = 6 (left: 4 6); 4 * 6 = 24 (left: 24)

Input-output (IO) prompting:
Case of creative writing: 1 prompt to write full passage
C**ase of crosswords:** each thought is 1 word

Tree of Thoughts – Evaluation of state

- Role: Helps the search algorithm determine which states to explore further and in which order.

- Two ways: (using LLMs)

1) Value each state independently:

- ❑An LLM prompt reasons about the state s and produces a value v (e.g., scalar or classification).
- ❑Heuristic Value: Scalar value or classification (e.g., sure/likely/impossible) assigned to each state,
- ❑Evaluation Basis: Few lookahead simulations combined with commonsense reasoning (e.g., checking if a number combination can reach a target or whether certain word parts make sense).

Tree of Thoughts – Evaluation of state

- Role: Helps the search algorithm determine which states to explore further and in which order.

- Two ways: (Using LLMs)

1) Value each state independently:

❑ An LLM prompt reasons about the state s and produces a value v (e.g., scalar or classification).
❑ Heuristic Value: Scalar value or classification (e.g., sure/likely/impossible) assigned to each state,
❑ Evaluation Basis: Few lookahead simulations combined with commonsense reasoning (e.g., checking if a number combination can reach a target or whether certain word parts make sense).

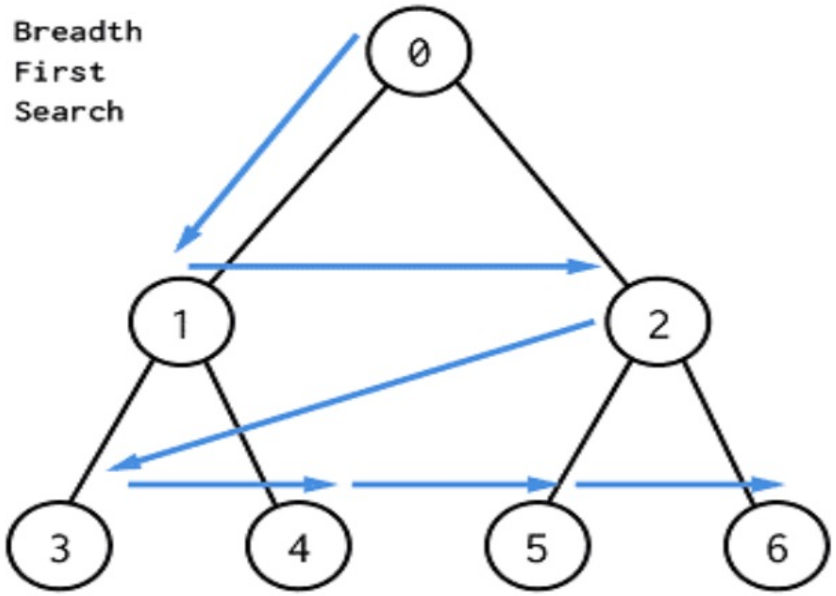
2)Vote across states:

❑ A frontier of states with values assigned to each, ranging from 1-10 or categorized as "good" or "bad" states.

Tree of Thoughts – Traversal

BFS: Maintains a set of the most promising states per step.

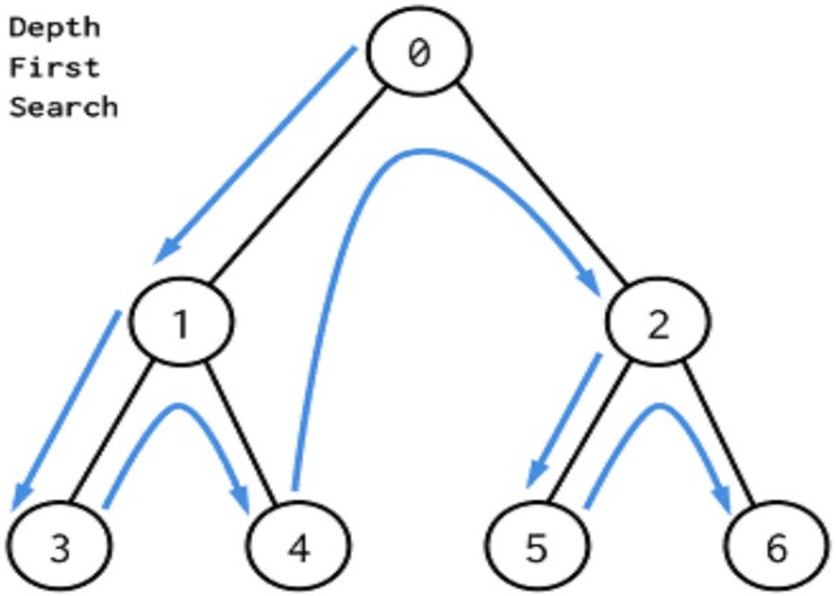
- When Used: Applied in tasks where the tree depth is small and manageable (e.g., Game of 24 and Creative Writing)
- Early thought steps are evaluated and pruned to a small number b≤5b



DFS: explores one path fully before backtracking to explore other possibilities.

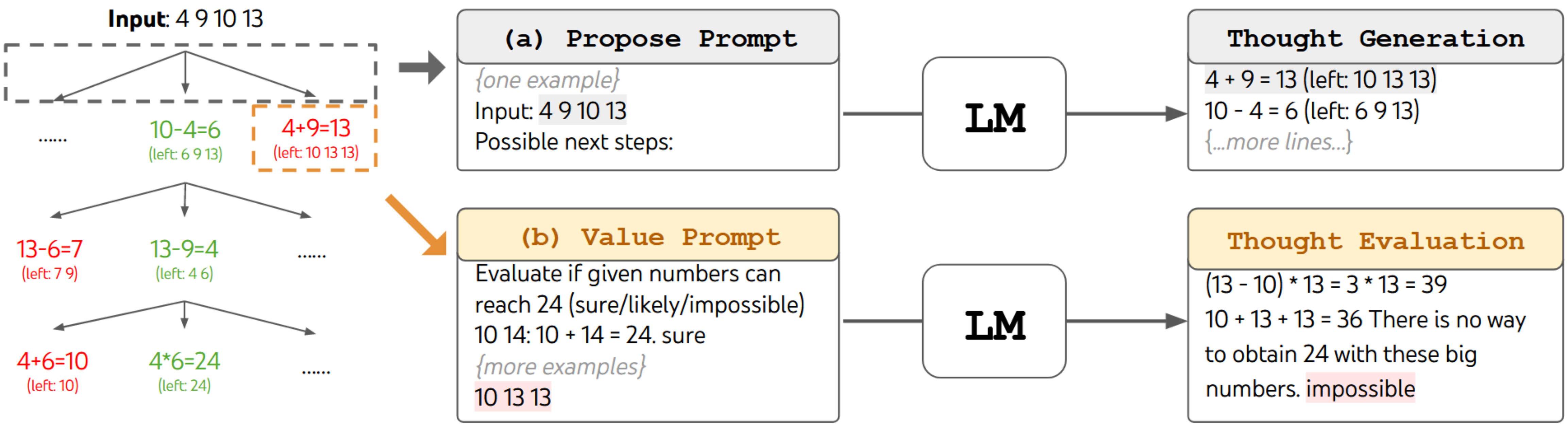
•Used for Complex Thought Sequences: (e.g., multi-step logic puzzles or crosswords).

•Backtracking on Failure: When a solution path proves unworkable, DFS backtracks to explore other branches, balancing deep exploration with pruning.



Game of 24

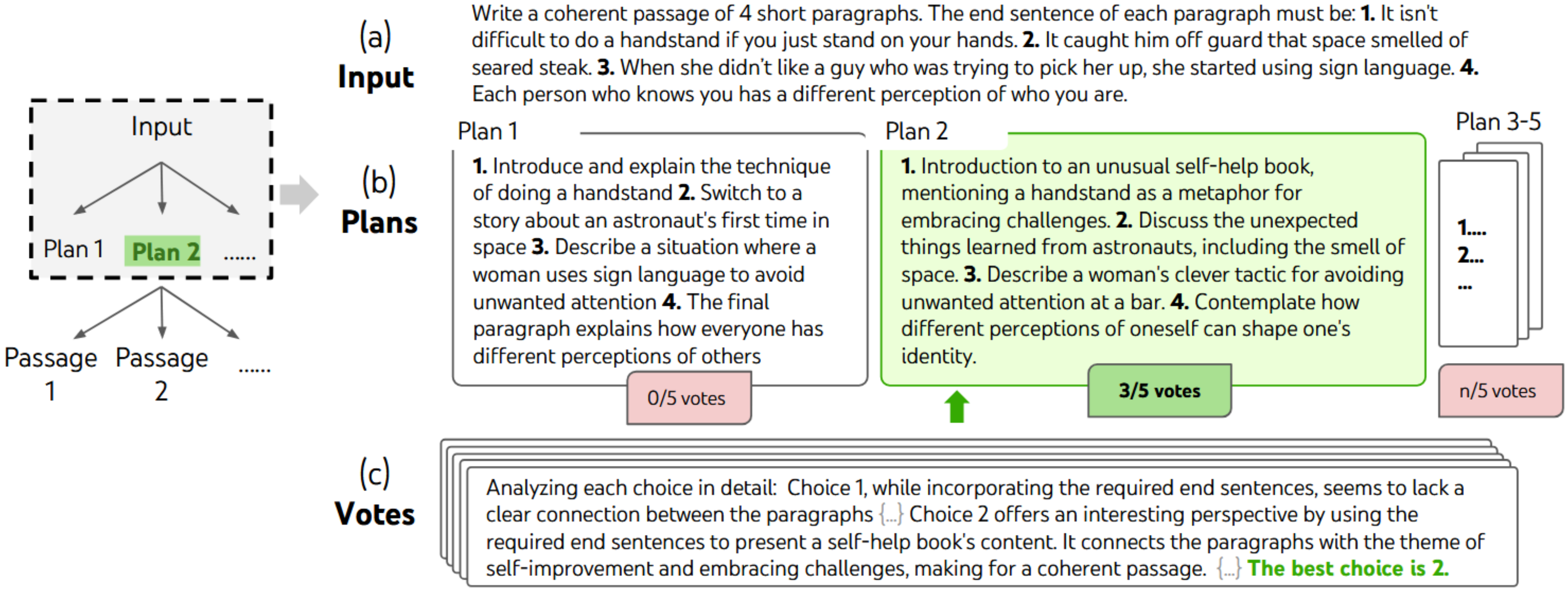
- Game of 24 is a mathematical reasoning challenge, where the goal is to use 4 numbers and basic arithmetic operations (+-*/) to obtain 24. For example, given input “4 9 10 13”, a solution output could be “(10 - 4) * (13 - 9) = 24”



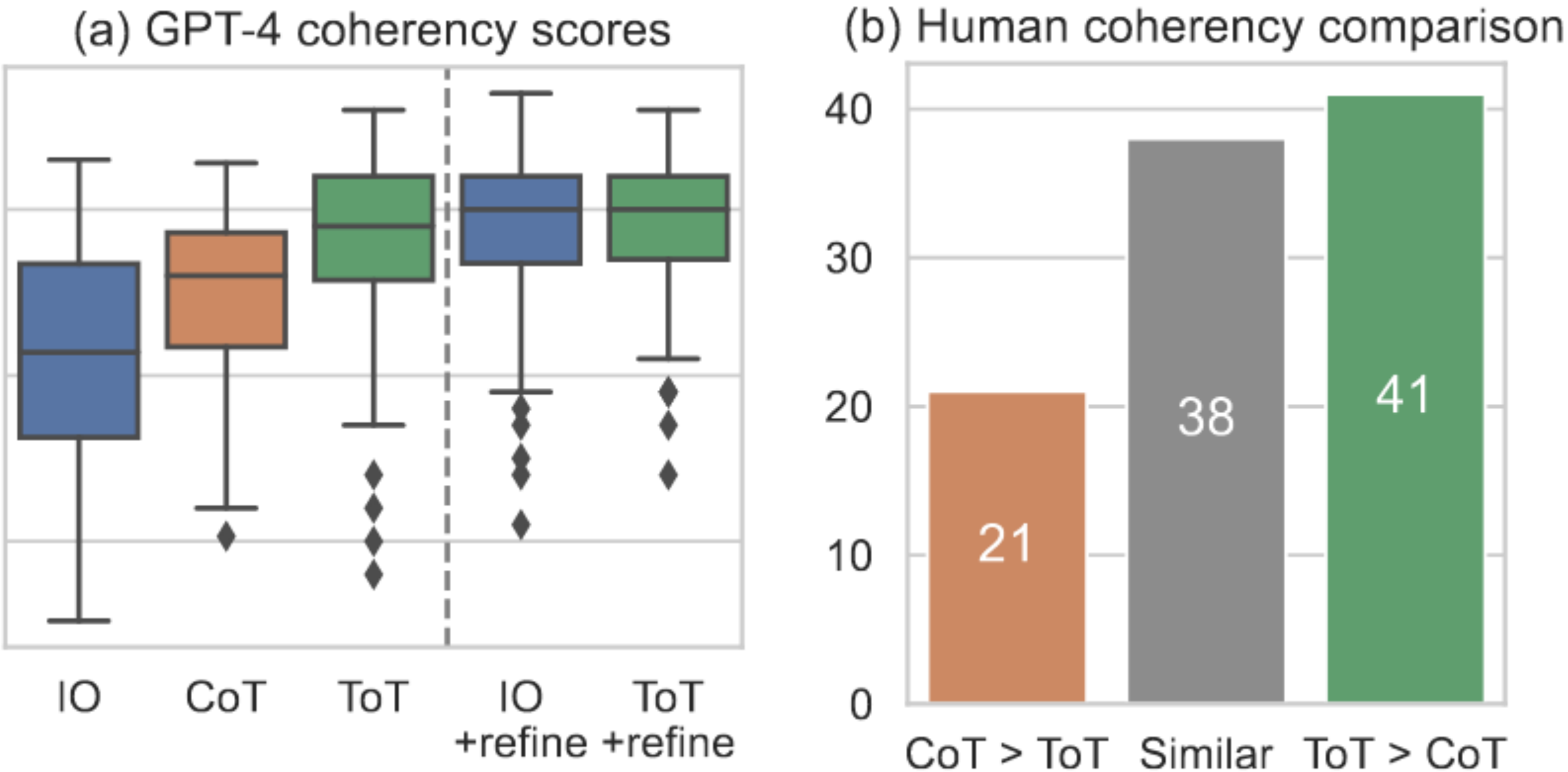
Game of 24 - Results

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%

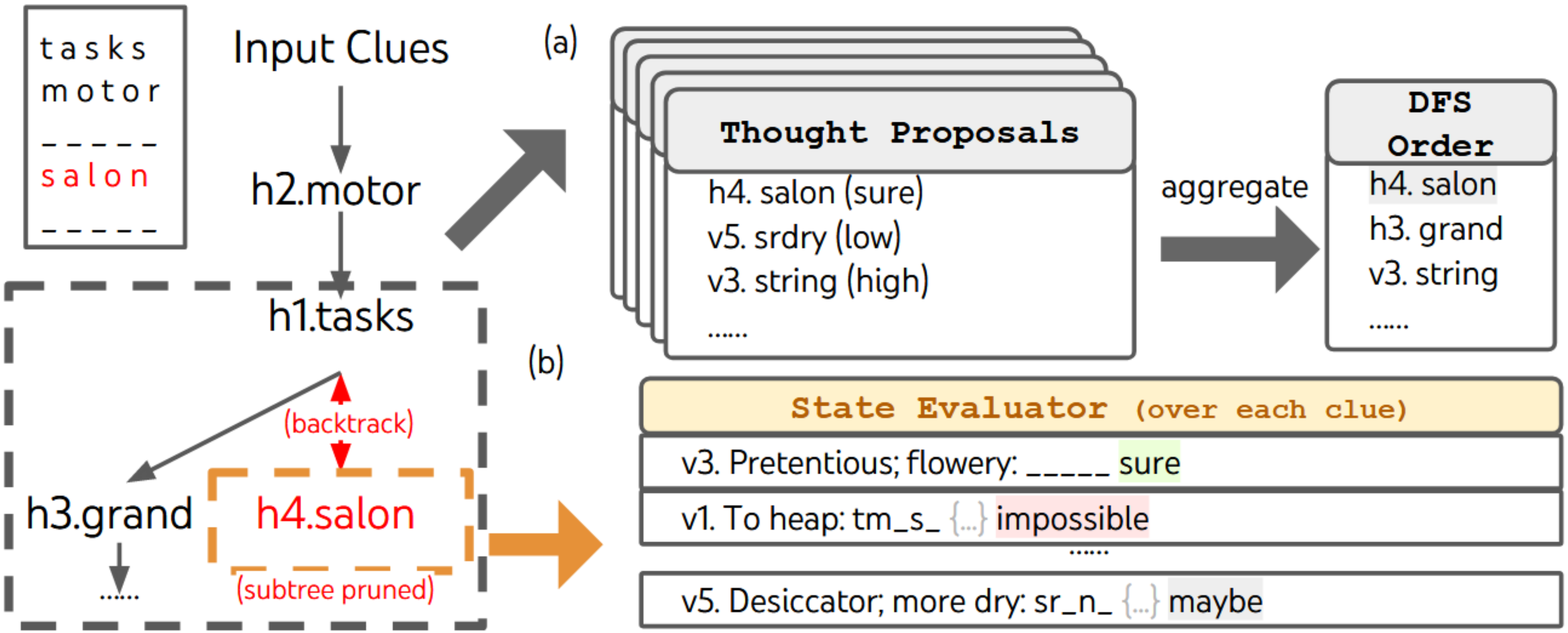
Creative Writing



Creative Writing - Results



Crosswords



tasks
motor
-----
salon

Input Clues
h2.motor
h1.tasks
h3.grand
h4.salon
(backtrack)
(subtree pruned)
......
(a)
(b)
Thought Proposals
h4. salon (sure)
v5. srdry (low)
v3. string (high)
......
aggregate
DFS Order
h4. salon
h3. grand
v3. string
......
State Evaluator (over each clue)
v3. Pretentious; flowery: _____ sure
v1. To heap: tm_s_ {...} impossible
......
v5. Desiccator; more dry: sr_n_ {...} maybe

Crosswords - 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

Conclusion

Augmentation of LMs: By searching a tree of possible paths, ToT enhances LMs' problem-solving capabilities, addressing tasks like creative writing and decision making.

Real-World Application: As LMs are deployed in real-world applications (e.g., coding, robotics, data analysis), ToT's search framework can address complex tasks that require deliberative thinking.

Improved Interpretability: ToT improves interpretability by offering high-level reasoning in natural language, making decision-making more transparent and aligned with human values.

Agenda

- Chain of Thought
- Least-to-Most Prompting
- Self-Consistency
- Tree of Thought
- Graph of Thoughts

Graph of Thoughts

Graph of Thoughts: Solving Elaborate Problems with Large Language Models

Solving Elaborate Problems with LLMs

Background

Scheme	**Sc?**	**Mc?**	**Tr?**	**Ag?**
Chain-of-Thought (CoT) [71]	🔋	✖	✖	✖
Self-Consistency with CoT [67]	🔋	🔋	✖	✖
Thought decomposition [75]	🔋	🔋	🪫	✖
Tree-of-Thought (ToT) [43]	🔋	🔋	🔋	✖
Tree of Thoughts (ToT) [77]	🔋	🔋	🔋	✖
Graph of Thoughts (GoT)	🔋	🔋	🔋	🔋

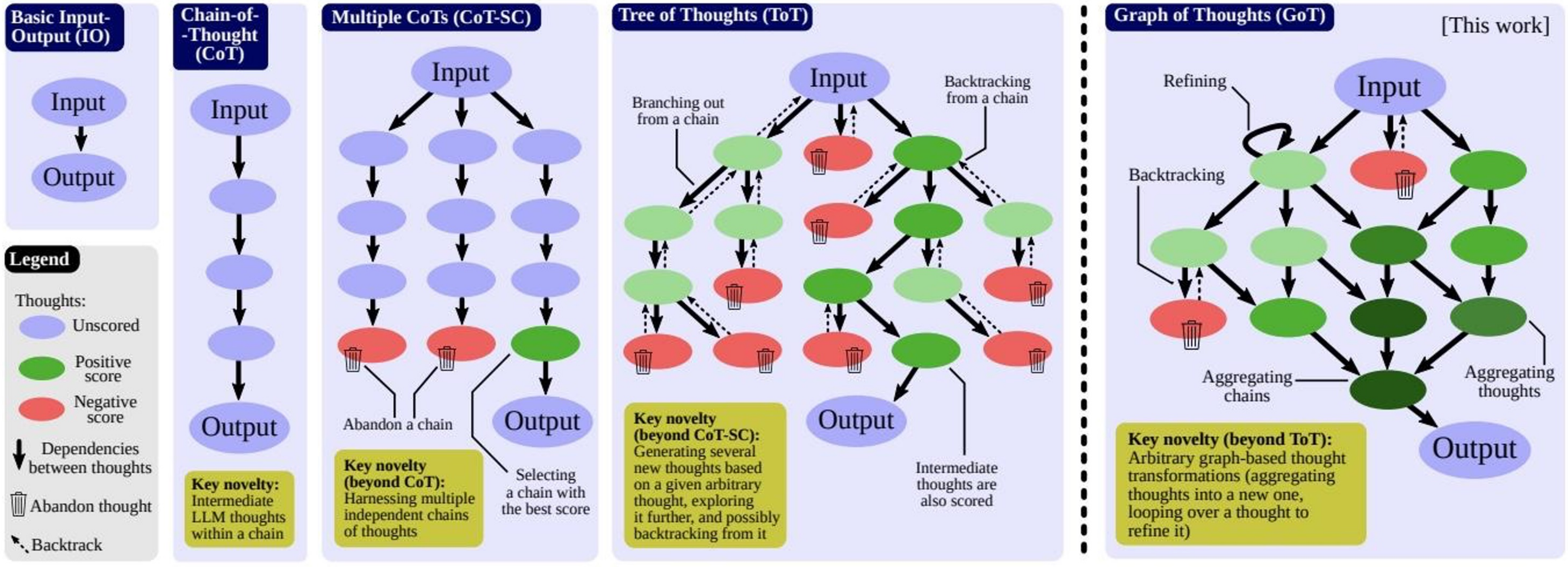
Table 1: Comparison of prompting schemes, with respect to the supported transformations of thoughts. **"Sc?"**: single chain of thoughts? **"Mc?"**: multiple chains of thoughts? **"Tr?"**: tree of thoughts? **"Ag?"**: arbitrary graph of thoughts? "🔋": full support, "🪫": partial support, "✖": no support.

Problem To Solve

- Rigid Tree Structure
  - Limits potential paths
  - Limits Problem solving possibilities

Proposed Solution

- Graph of Thought (GoT)
  - Arbitrary graph structure
  - Multiple chains adding and subtracting



GoT Framework - Reasoning Process

Graph: $G = (V, E)$

- $E \subseteq V \times V$

Directed edges: t, thought (not necessarily final)

- $(t1, t2)$, thought built upon another

Sometimes $G = (V,E,c) \rightarrow$ classes of thoughts

GoT Framework - Transformation of Thoughts

Graph-Enabled Transformations

pθ: LLM Used

E-/E+ and V-/V+: Additions or

subtractions of edges and vertices

$G' = T(G, p\theta) = (V', E')$

where $V' = (V \cup V^+) \setminus V^-$

and $E' = (E \cup E^+) \setminus E^-$

- Additions and Subtractions to pθ (current state)

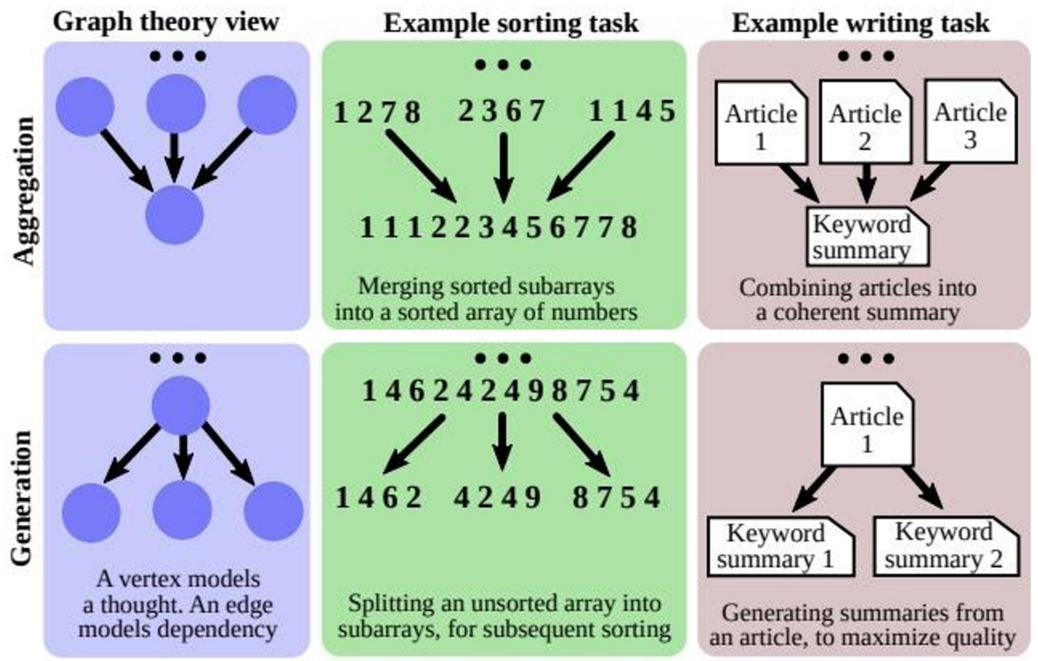


Figure 2: Examples of aggregation and generation thought transformations.

GoT Fram ts

Graph-Enabl

- Aggregation Transformations
- Refining Transformations
- Generation Transformations

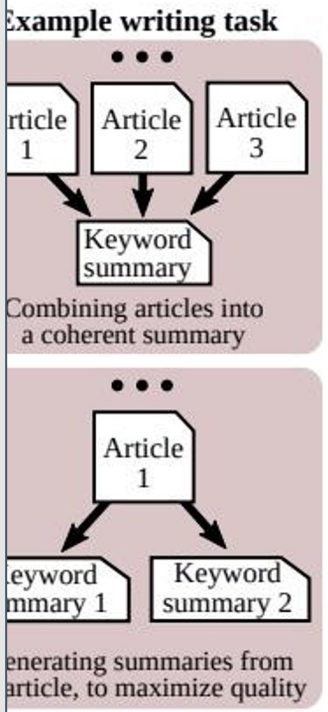
G′ = T (G, pθ

where V ′ = (

and E′ = (E U

- Addition

  To pθ (c



neration thought transformations.

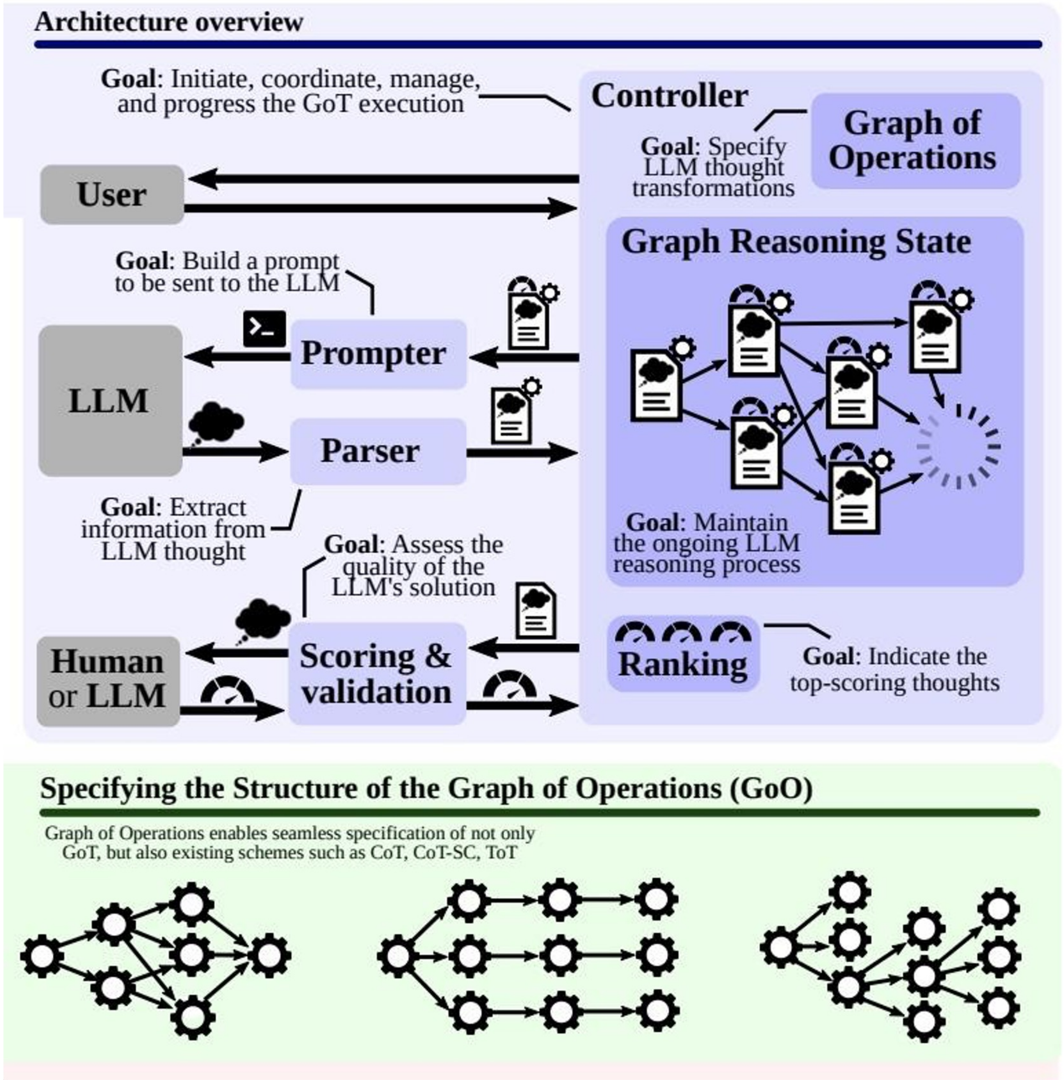
GoT Framework - Scoring and Ranking Thoughts

$E(v, G, p\theta)$, solution evaluation (v are thoughts to be evaluated)

$R(G, p\theta, h)$, thought evaluation (h are highest ranking thoughts)

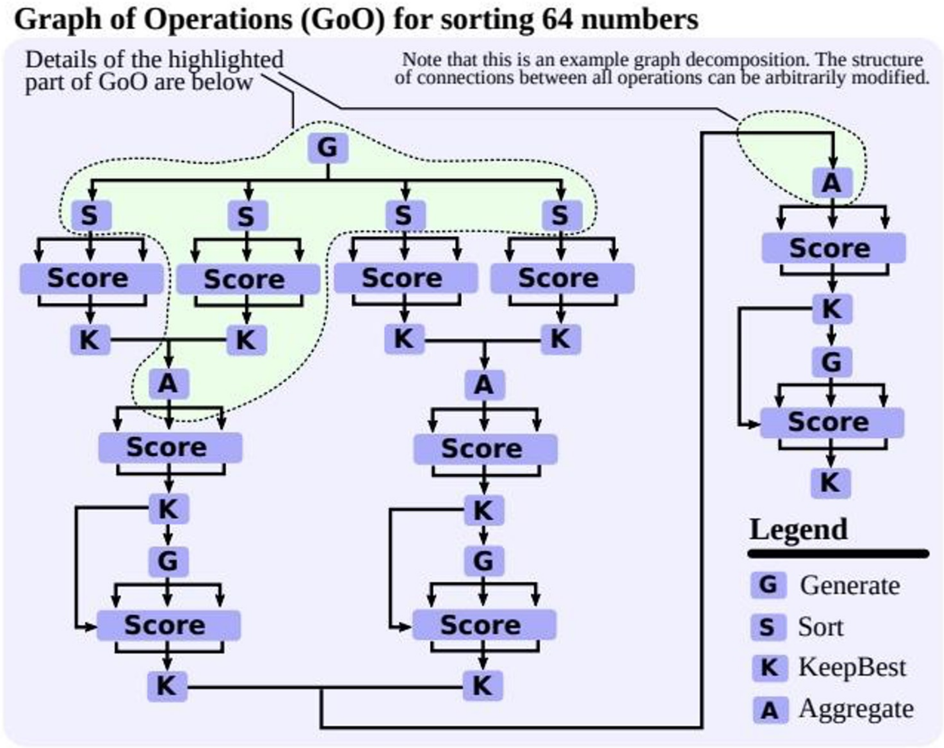
Architecture

- GoO (Graph of Operations)
- GRS (Graph Reasoning State)
- Prompter
- Parser
- Scording and validation
- Controller



Examples

Sorting



Keyword Counting

- Splits passage into smaller parts
- Aggregates subresults

Results

Sorting

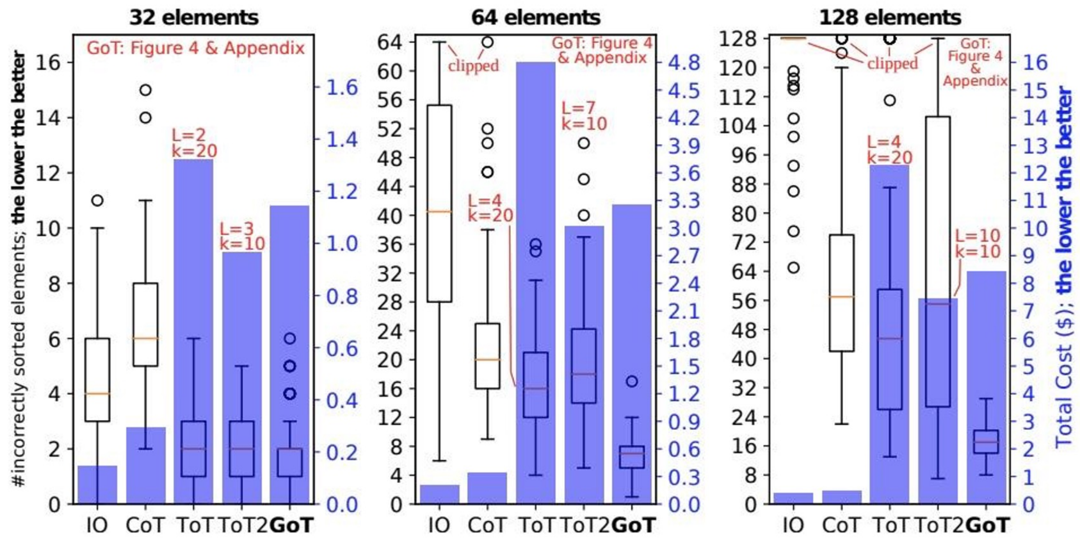


Figure 5: Number of errors and cost in sorting tasks with ChatGPT-3.5. L and k indicate the structure of ToT (see Sections 3.2 and 6).

Keyword Counting

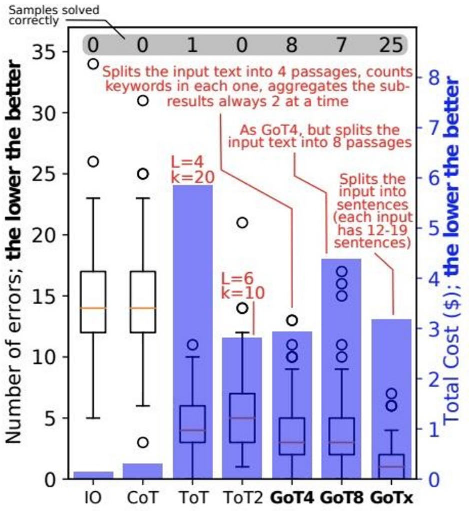


Figure 7: Number of errors and cost in keyword counting with ChatGPT-3.5. L and k indicate the structure of ToT (see Sections 3.2 and 6).

- IO: input-output (standard output)
- ToT2 (lower k, higher L)

Results

GoT improves on ToT,

- Reduce median error by 62%
- More costly than ToT2 variation

GoT provides answers better than CoT and IO

- 65% and 83%
- Significantly higher cost

GoT allows increased task complexity

Take Away

- Allows improvement without model update
- Outperforms other prompting schemes in solutions
- Reduces costs compared to other complex paradigms
- Excels at larger and more complex prompts

Future of LLM Reasoning

- Cannot self-correct reasoning
  - Struggle without human interaction
  - Could degrade quality with attempted self-correcting
  - https://arxiv.org/abs/2310.01798
- More to do in mathematical reasoning
  - Large field of metrics, datasets, and settings for rigorous logical reasoning
  - Lack of unified framework to determine successful models
  - https://arxiv.org/pdf/2402.00157
- LLMs are black-box mechanisms
  - Attention heads to discovery reasoning
  - https://arxiv.org/pdf/2409.03752

Q&A

Thank you for listening!