UNIVERSITY of VIRGINIA

Self-Exam LLM and Reasoning

Presented by Team 2

Jessie Chen (hc4vb)
Soneya Binta Hossain (sh7hv)
Ali Zafar Sadiq (mzw2cu)
Jeffrey Chen (fyy2ws)
Minjae Kwon (hbt9su)

UNIVERSITY of VIRGINIA | ENGINEERING

COMPUTER SCIENCE

Presentation Outline

Paper 1: Self-Consistency Improves Chain of Thought Reasoning in Language Models

Paper 2: Augmented Language Models: a Survey

Paper 3: If LLM Is the Wizard, Then Code Is the Wand: A Survey on How Code Empowers Large Language Models to Serve as Intelligent Agents

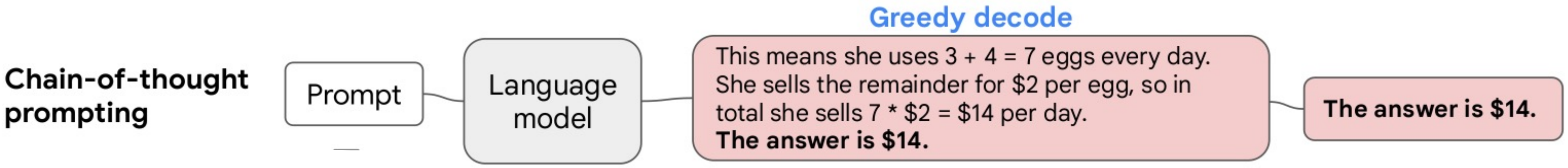
Paper 1

Self-Consistency Improves Chain of Thought Reasoning in Language Models

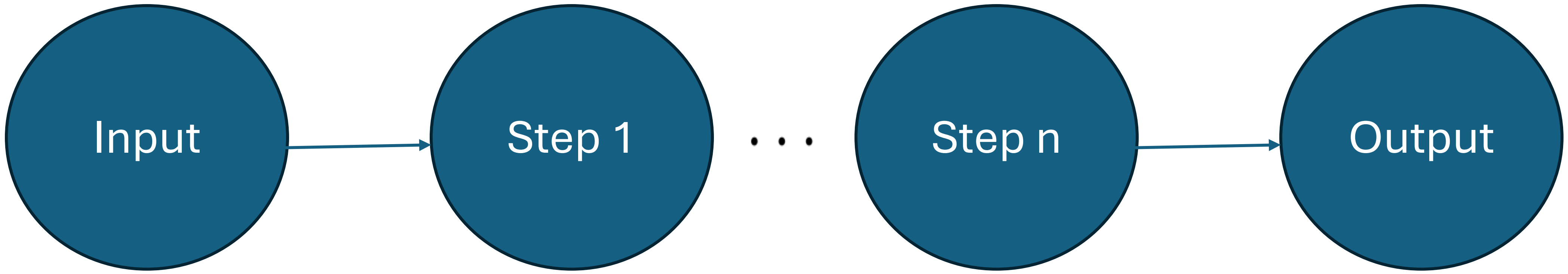
presenters: Jeffrey Chen (fyy2ws)
Minjae Kwon (hbt9su)

Presenter
Minjae Kwon (hbt9su)

Chain of Thought (CoT)



This means she uses 3 + 4 = 7 eggs every day.
She sells the remainder for $2 per egg, so in total she sells 7 * $2 = $14 per day.
The answer is $14.



Greedy Decoding !

Motivation

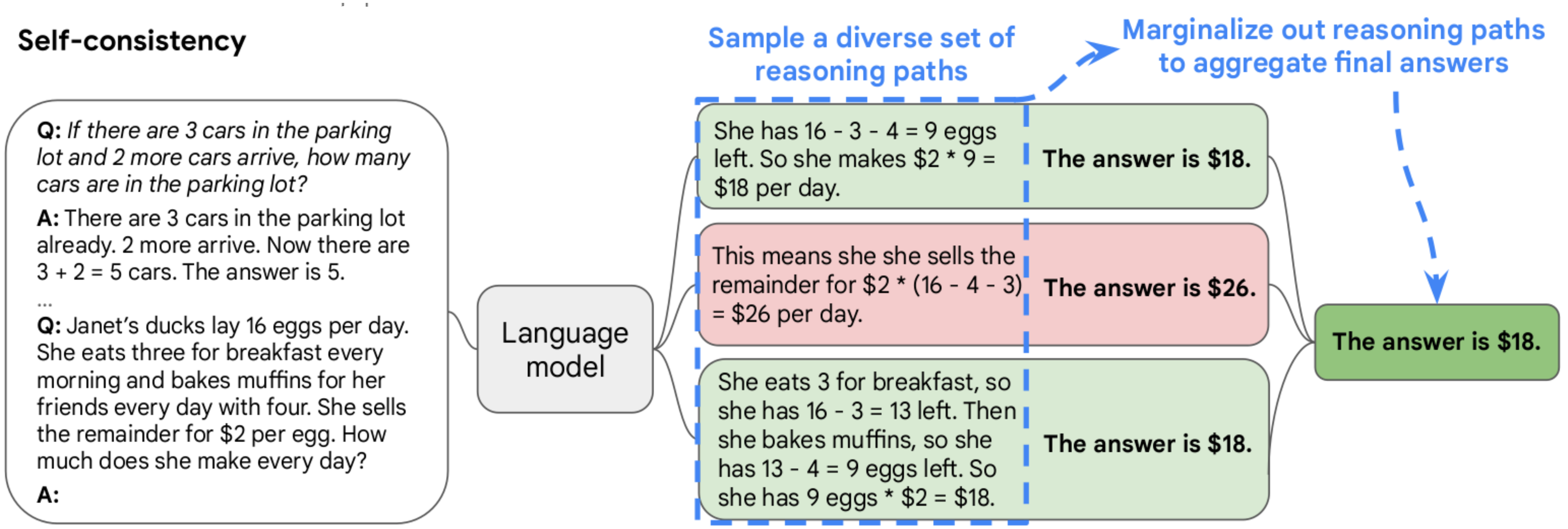
- Replace Greedy Decoding
- Multiple paths might exist to get an answer !

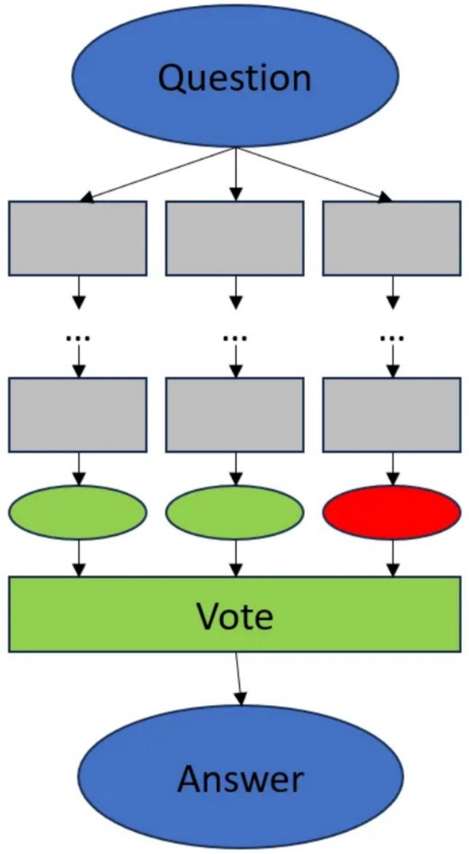
Example: Unscramble the word "LSTETRE" to form a valid English word.

Multiple options:

- Option 1: "LSTETRE" → "RETLSTE" → "LETTERS"
- Option 2: "LSTETRE" → "STLTERE" → "LETTERS"
- Option 3: "LSTETRE" → "TERLEST" → "TERSLET" → "LETTERS"

Self-Consistency


Self-consistency
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars. The answer is 5.
...
Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for $2 per egg. How much does she make every day?
A:
Language model
Sample a diverse set of reasoning paths
She has 16 - 3 - 4 = 9 eggs left. So she makes $2 * 9 = $18 per day.
The answer is $18.
This means she she sells the remainder for $2 * (16 - 4 - 3) = $26 per day.
The answer is $26.
She eats 3 for breakfast, so she has 16 - 3 = 13 left. Then she bakes muffins, so she has 13 - 4 = 9 eggs left. So she has 9 eggs * $2 = $18.
The answer is $18.
Marginalize out reasoning paths to aggregate final answers
The answer is $18.



Aggregation from Multiple Paths !
Unsupervised Method

Image source

Aggregation Strategy

Formulation

(1) Generated answers $a_i \in \mathbb{A}$ where $i \in [1, m]$

(2) Latent variables r_i: a sequence of tokens

(3) Majority vote with $\{(r_i, a_i)\}_{i \in [1,m]}$. For example, $\arg\max_a \sum_{i=1}^{m} \mathbb{I}(\mathbf{a}_i = a)$

(4) Aggregating with weight $P(\mathbf{r}_i, \mathbf{a}_i \mid \text{prompt}, \text{question})$

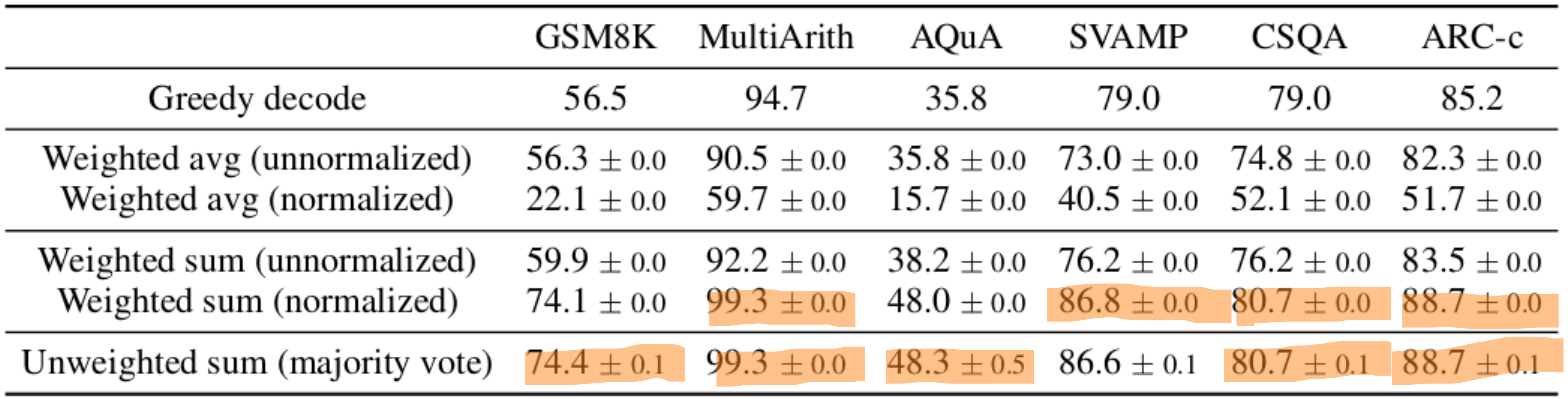
- Unnormalized:

$$\prod_{k=1}^{K} P(t_k \mid \text{prompt}, \text{ question}, t_1, \ldots, t_{k-1})$$

- Normalized:

$$\prod_{k=1}^{K} P(t_k \mid \text{prompt}, \text{ question}, t_1, \ldots, t_{k-1})^{\frac{1}{K}}$$

Aggregation Strategy

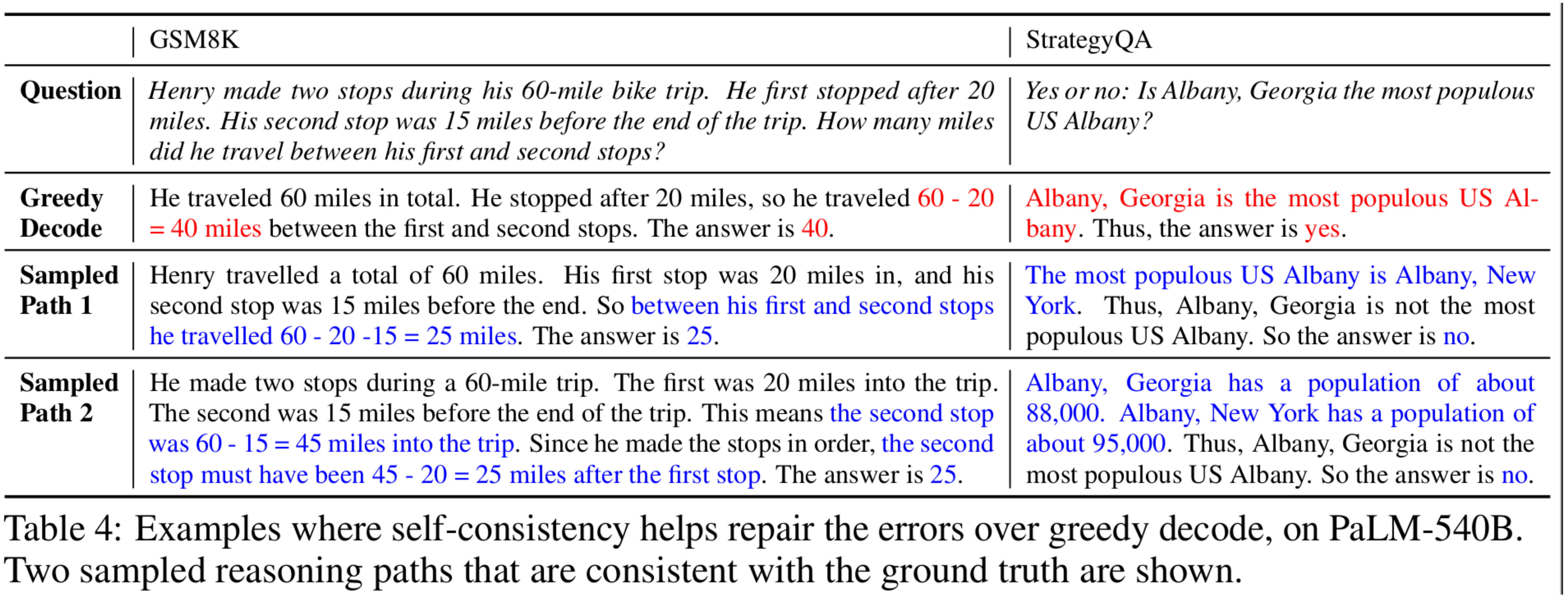


	GSM8K	MultiArith	AQuA	SVAMP	CSQA	ARC-c
Greedy decode	56.5	94.7	35.8	79.0	79.0	85.2
Weighted avg (unnormalized)	56.3 ± 0.0	90.5 ± 0.0	35.8 ± 0.0	73.0 ± 0.0	74.8 ± 0.0	82.3 ± 0.0
Weighted avg (normalized)	22.1 ± 0.0	59.7 ± 0.0	15.7 ± 0.0	40.5 ± 0.0	52.1 ± 0.0	51.7 ± 0.0
Weighted sum (unnormalized)	59.9 ± 0.0	92.2 ± 0.0	38.2 ± 0.0	76.2 ± 0.0	76.2 ± 0.0	83.5 ± 0.0
Weighted sum (normalized)	74.1 ± 0.0	99.3 ± 0.0	48.0 ± 0.0	86.8 ± 0.0	80.7 ± 0.0	88.7 ± 0.0
Unweighted sum (majority vote)	74.4 ± 0.1	99.3 ± 0.0	48.3 ± 0.5	86.6 ± 0.1	80.7 ± 0.1	88.7 ± 0.1

Table 1: Accuracy comparison of different answer aggregation strategies on PaLM-540B.

- Unweighted Sum Strategy is the best for Reasoning Dataset
- Weighted Sum: a + a + a, b + b, c
- Weighted Avg: (a +a + a)/3, (b + b)/2, c

Examples



	Dataset 1	Dataset 2
	GSM8K	StrategyQA
Question	*Henry made two stops during his 60-mile bike trip. He first stopped after 20 miles. His second stop was 15 miles before the end of the trip. How many miles did he travel between his first and second stops?*	*Yes or no: Is Albany, Georgia the most populous US Albany?*
Greedy Decode	He traveled 60 miles in total. He stopped after 20 miles, so he traveled 60 - 20 = 40 miles between the first and second stops. The answer is 40.	Albany, Georgia is the most populous US Albany. Thus, the answer is yes.
Sampled Path 1	Henry travelled a total of 60 miles. His first stop was 20 miles in, and his second stop was 15 miles before the end. So between his first and second stops he travelled 60 - 20 -15 = 25 miles. The answer is 25.	The most populous US Albany is Albany, New York. Thus, Albany, Georgia is not the most populous US Albany. So the answer is no.
Sampled Path 2	He made two stops during a 60-mile trip. The first was 20 miles into the trip. The second was 15 miles before the end of the trip. This means the second stop was 60 - 15 = 45 miles into the trip. Since he made the stops in order, the second stop must have been 45 - 20 = 25 miles after the first stop. The answer is 25.	Albany, Georgia has a population of about 88,000. Albany, New York has a population of about 95,000. Thus, Albany, Georgia is not the most populous US Albany. So the answer is no.

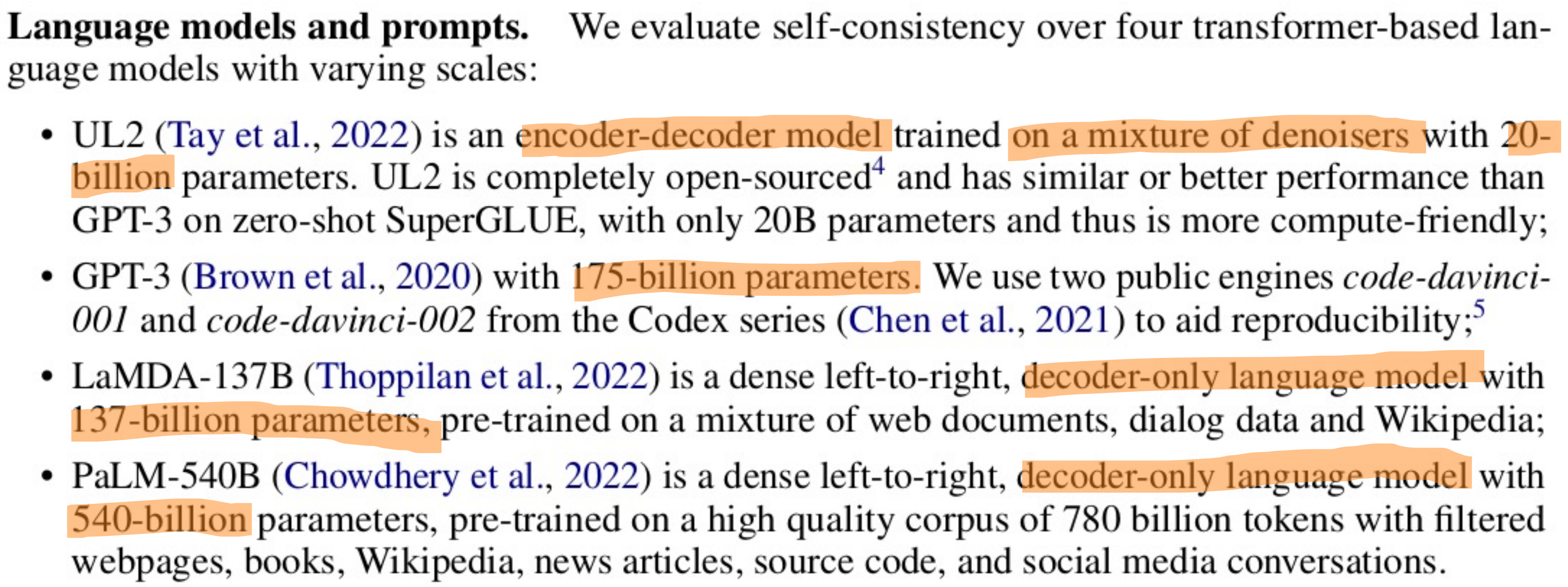
Table 4: Examples where self-consistency helps repair the errors over greedy decode, on PaLM-540B. Two sampled reasoning paths that are consistent with the ground truth are shown.

UNIVERSITY of VIRGINIA

Tasks and Datasets

- Arithmetic Reasoning
  - Math Word Problem Repository
  - AQUA-RAT
  - GSM8K
  - SVAMP
- Commonsense Reasoning
  - Commonsense-QA
  - Strategy-QA
  - AI2 Reasoning Challenge (ARC)
- Symbolic Reasoning
  - Last Letter Concatenation: Elon Musk -> nk
  - Coinflip: a coin is heads-up, after a few flips, is the coin still heads up?

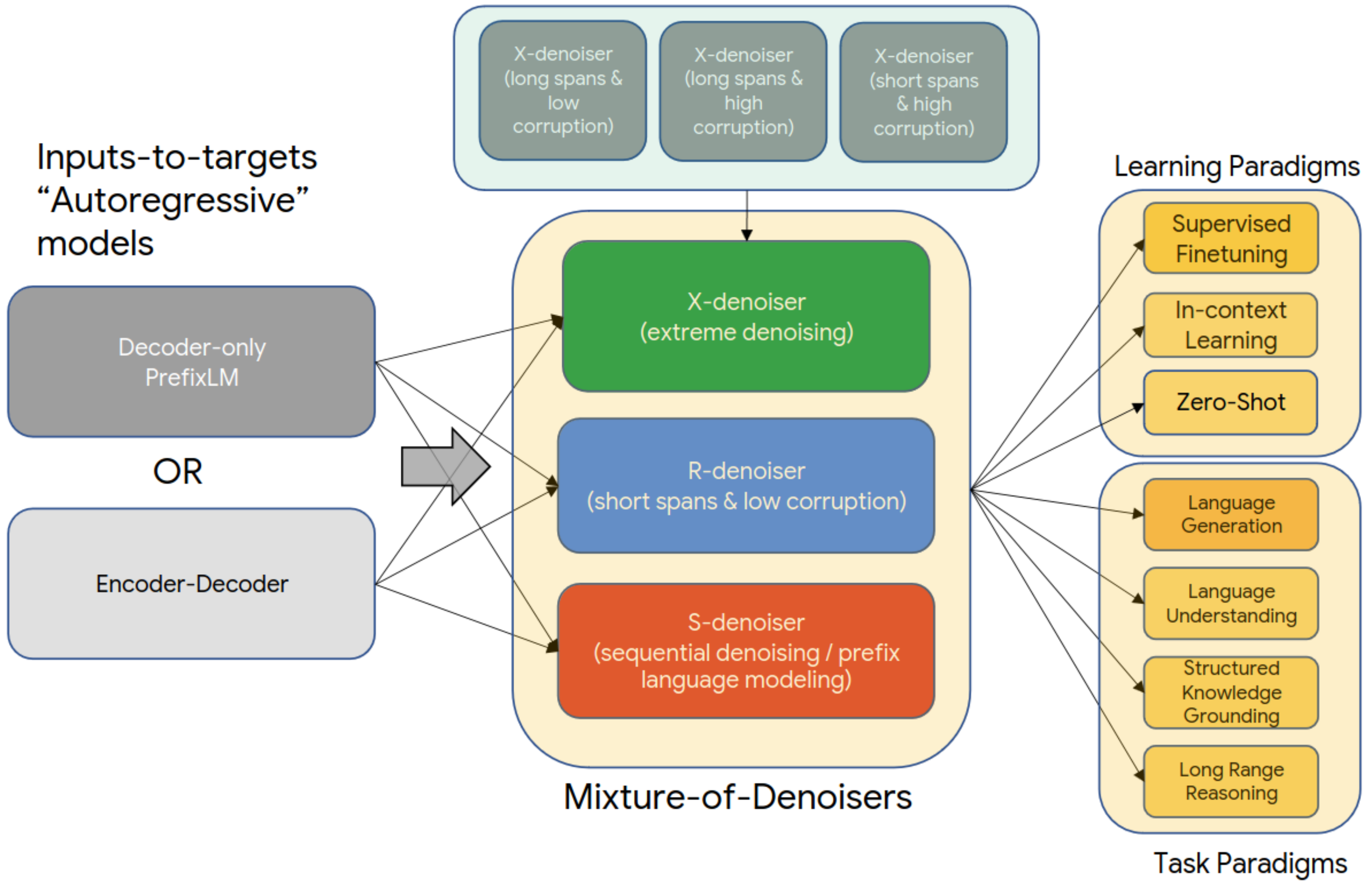
Evaluation Models



Language models and prompts. We evaluate self-consistency over four transformer-based language models with varying scales:

- UL2 (Tay et al., 2022) is an encoder-decoder model trained on a mixture of denoisers with 20-billion parameters. UL2 is completely open-sourced[4] and has similar or better performance than GPT-3 on zero-shot SuperGLUE, with only 20B parameters and thus is more compute-friendly;
- GPT-3 (Brown et al., 2020) with 175-billion parameters. We use two public engines *code-davinci-001* and *code-davinci-002* from the Codex series (Chen et al., 2021) to aid reproducibility;[5]
- LaMDA-137B (Thoppilan et al., 2022) is a dense left-to-right, decoder-only language model with 137-billion parameters, pre-trained on a mixture of web documents, dialog data and Wikipedia;
- PaLM-540B (Chowdhery et al., 2022) is a dense left-to-right, decoder-only language model with 540-billion parameters, pre-trained on a high quality corpus of 780 billion tokens with filtered webpages, books, Wikipedia, news articles, source code, and social media conversations.

UL2

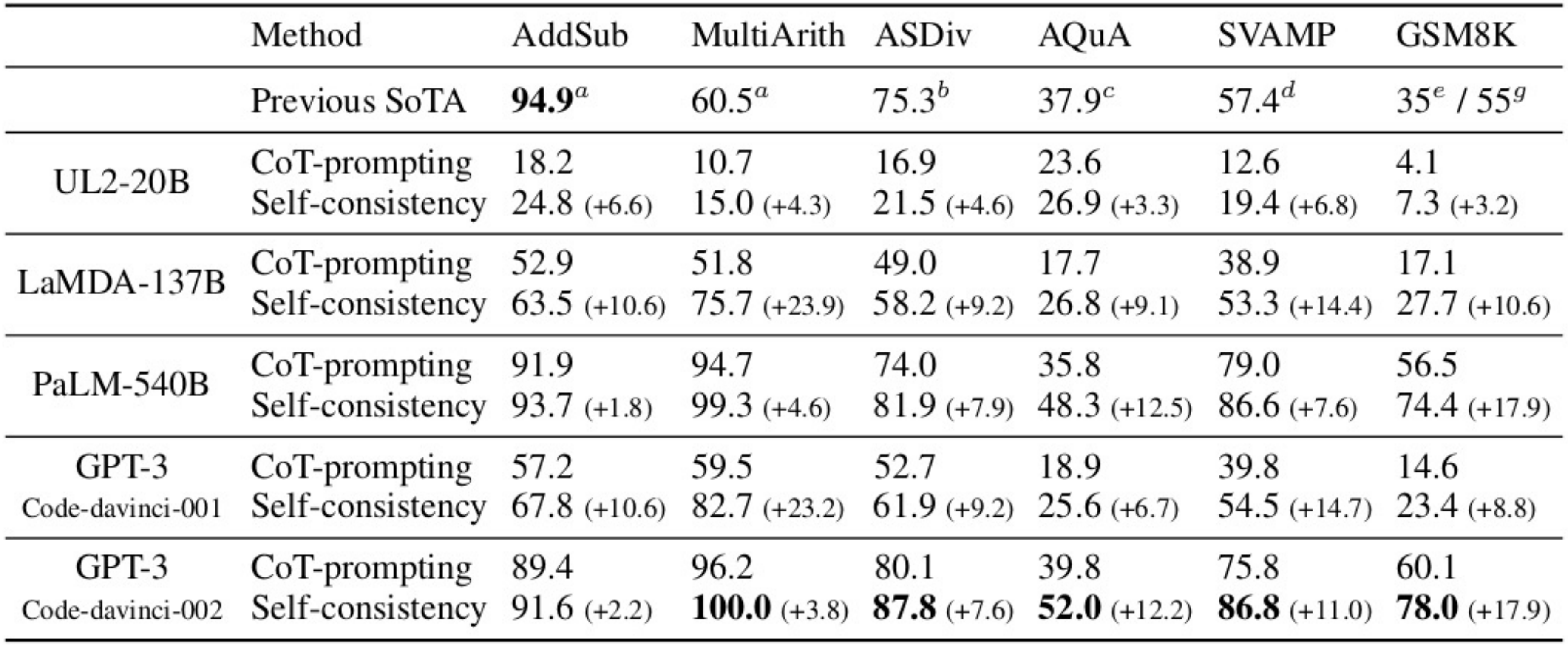


X-denoiser (long spans & low corruption)
X-denoiser (long spans & high corruption)
X-denoiser (short spans & high corruption)
Inputs-to-targets “Autoregressive” models
Decoder-only PrefixLM
OR
Encoder-Decoder
X-denoiser (extreme denoising)
R-denoiser (short spans & low corruption)
S-denoiser (sequential denoising / prefix language modeling)
Mixture-of-Denoisers
Learning Paradigms
Supervised Finetuning
In-context Learning
Zero-Shot
Language Generation
Language Understanding
Structured Knowledge Grounding
Long Range Reasoning
Task Paradigms

UNIVERSITY of VIRGINIA

Main Results

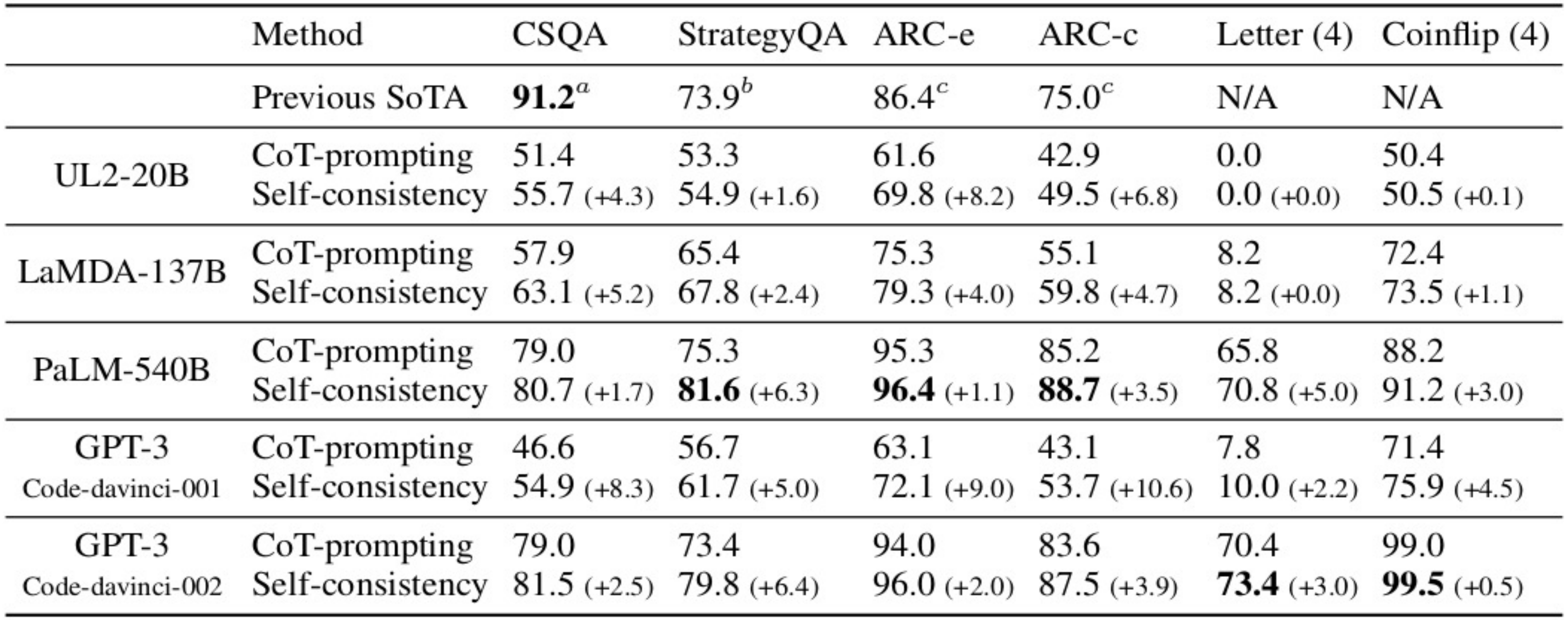
1. Arithmetic Reasoning



	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	CoT-prompting	57.2	59.5	52.7	18.9	39.8	14.6
Code-davinci-001	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	CoT-prompting	89.4	96.2	80.1	39.8	75.8	60.1
Code-davinci-002	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)

Main Results

2. Common Sense 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)

Main Results

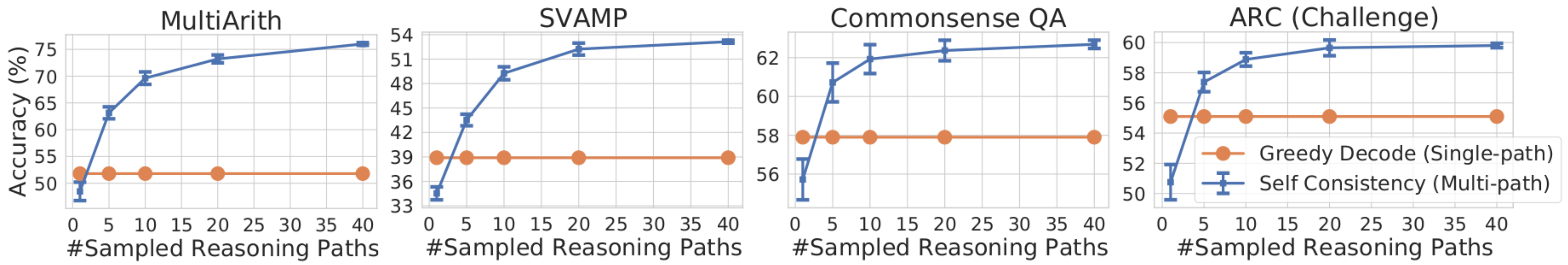
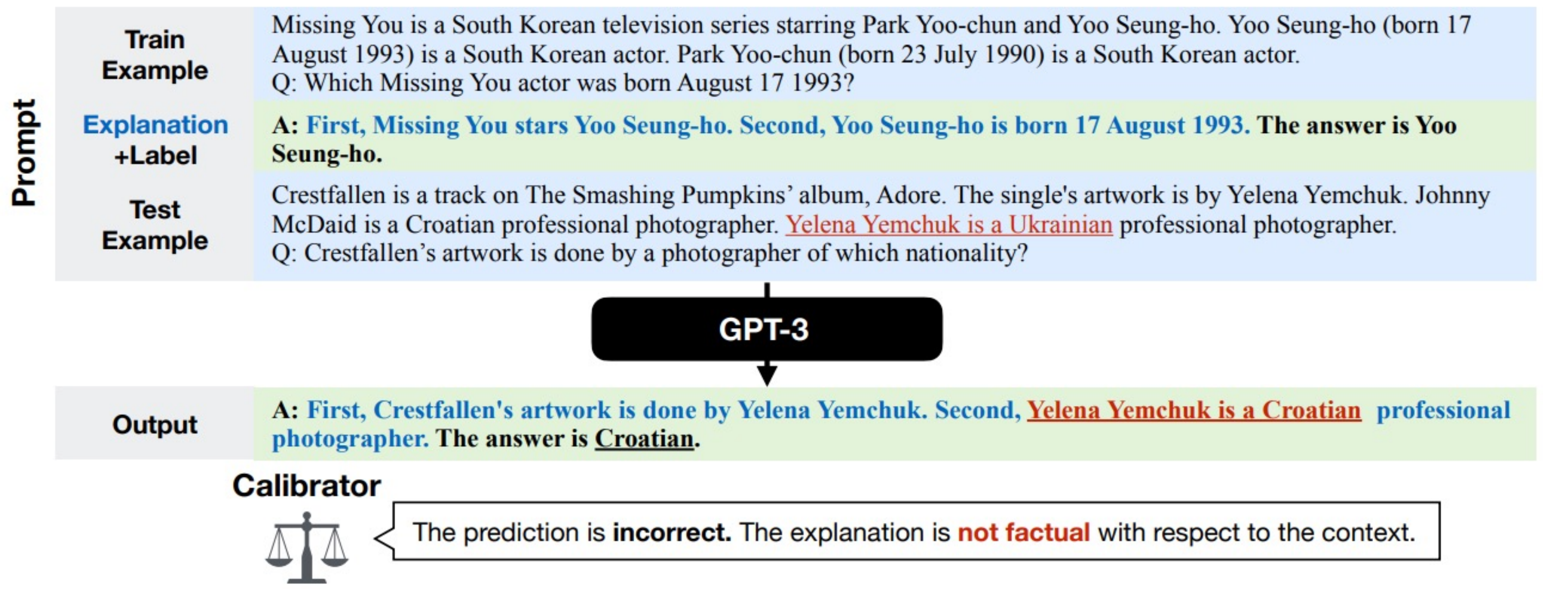


Figure 2: Self-consistency (blue) significantly improves accuracy over CoT-prompting with greedy decoding (orange) across arithmetic and commonsense reasoning tasks, over LaMDA-137B. Sampling a higher number of diverse reasoning paths consistently improves reasoning accuracy.

Presenter
Jeffrey Chen (fyy2ws)

Self-Consistency vs Chain of Thought

- Chain of thought can hurt performance
  - When compared to standard prompting
  - Few-shot in-context learning



Missing You is a South Korean television series starring Park Yoo-chun and Yoo Seung-ho. Yoo Seung-ho (born 17 August 1993) is a South Korean actor. Park Yoo-chun (born 23 July 1990) is a South Korean actor.
Q: Which Missing You actor was born August 17 1993?

A: First, Missing You stars Yoo Seung-ho. Second, Yoo Seung-ho is born 17 August 1993. The answer is Yoo Seung-ho.

Crestfallen is a track on The Smashing Pumpkins' album, Adore. The single's artwork is by Yelena Yemchuk. Johnny McDaid is a Croatian professional photographer. Yelena Yemchuk is a Ukrainian professional photographer.
Q: Crestfallen's artwork is done by a photographer of which nationality?

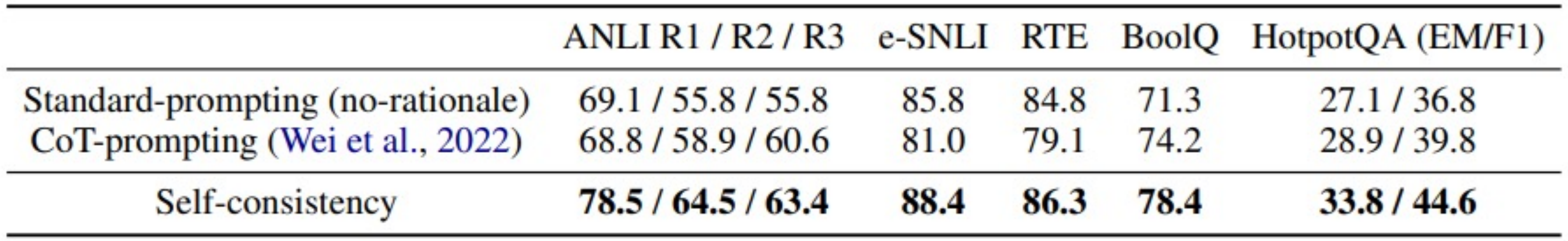
A: First, Crestfallen's artwork is done by Yelena Yemchuk. Second, Yelena Yemchuk is a Croatian professional photographer. The answer is Croatian.

The prediction is **incorrect.** The explanation is **not factual** with respect to the context.

Figure 1: Prompting GPT-3 with explanations. By including explanations in the in-context examples, we can cause GPT-3 to generate an explanation for the test example as well. In this case, the generated explanation is nonfactual, despite the simple reasoning involved here. However, we show this nonfactuality actually provides a signal that can help calibrate the model.

Self-Consistency vs Chain of Thought

- Self-consistency helps where chain of thought hurts
- Reliable way to add rationales



	ANLI R1 / R2 / R3	e-SNLI	RTE	BoolQ	HotpotQA (EM/F1)
Standard-prompting (no-rationale)	69.1 / 55.8 / 55.8	85.8	84.8	71.3	27.1 / 36.8
CoT-prompting (Wei et al., 2022)	68.8 / 58.9 / 60.6	81.0	79.1	74.2	28.9 / 39.8
Self-consistency	**78.5 / 64.5 / 63.4**	**88.4**	**86.3**	**78.4**	**33.8 / 44.6**

Table 5: Compare Standard/CoT prompting with self-consistency on common NLP tasks.

UNIVERSITY of VIRGINIA

Self-Consistency vs Sample-and-Rank

- Sample-and-Rank
  - Approach to improve generation quality
  - Multiple sequences sampled
  - Ranked according to log probability

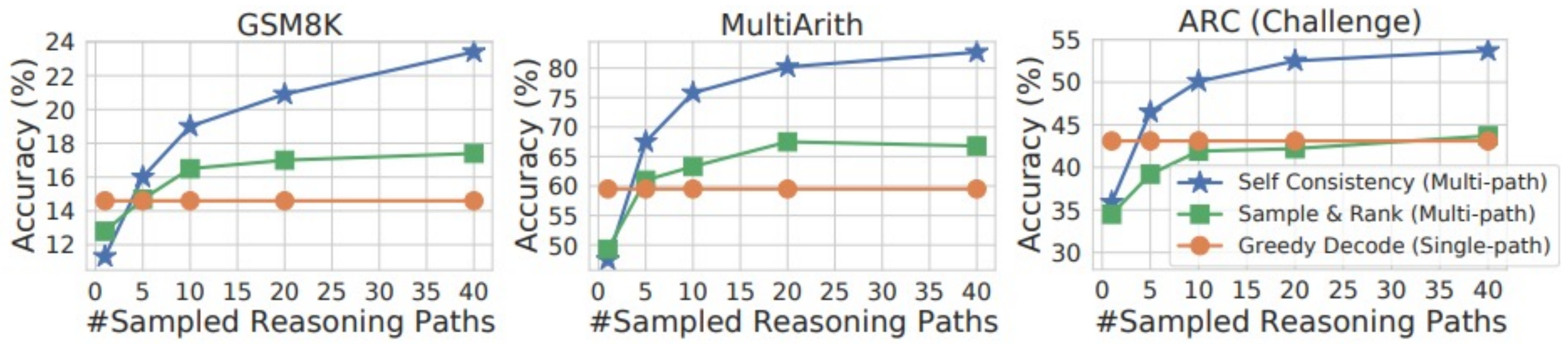
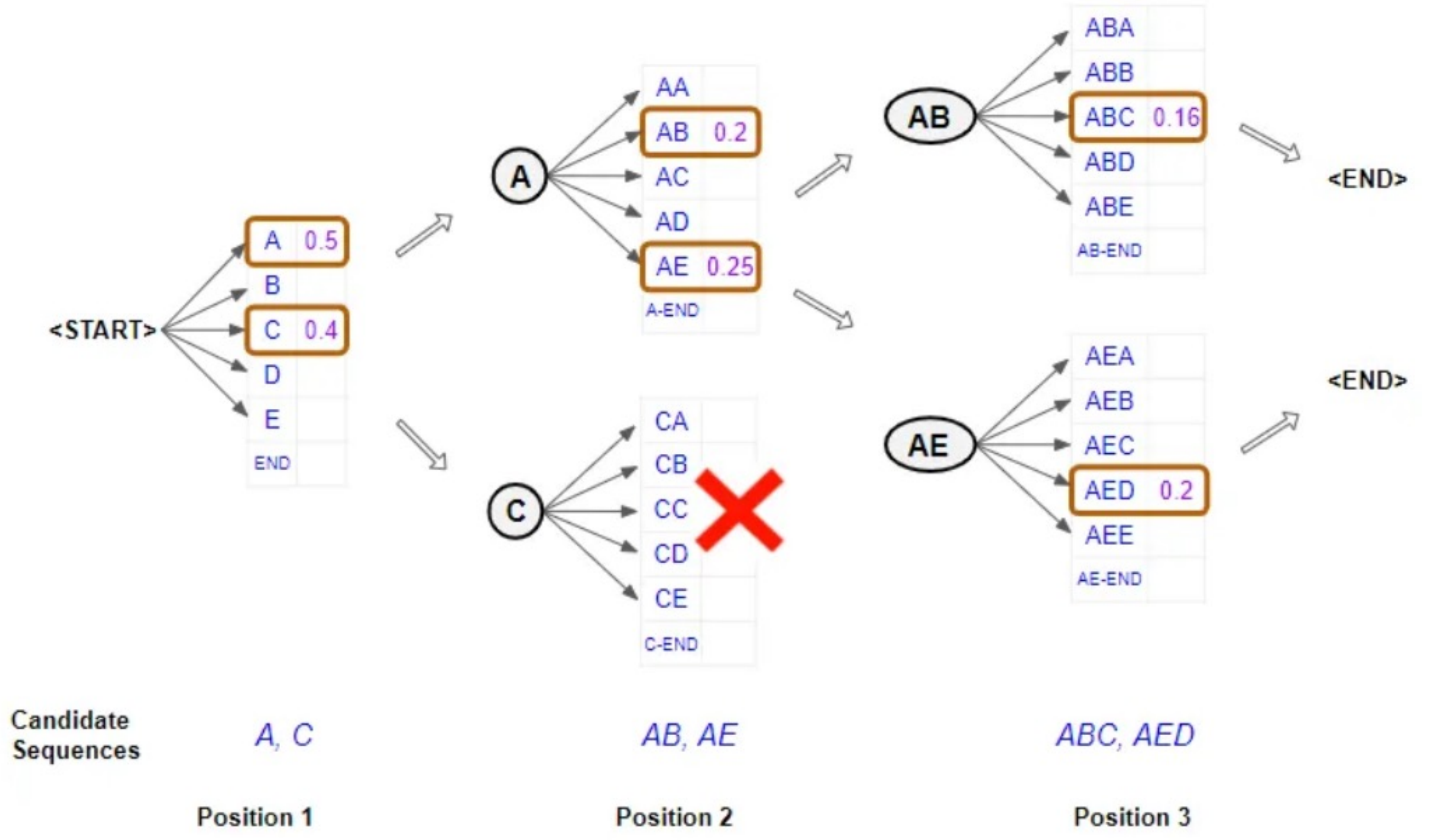


Figure 3: Self-consistency significantly outperforms sample-and-rank with the same # of samples.

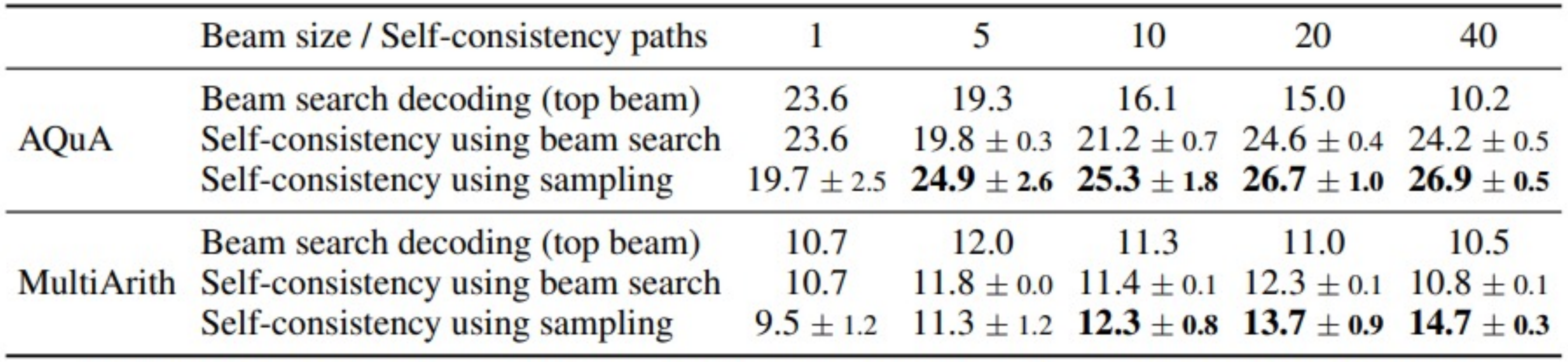
Self-Consistency vs Beam Search



Beam Search example, with width = 2 (Image by Author)

Self-Consistency vs Beam Search

- Accuracy reported on same number of beams and reasoning paths
- Self-consistency can adopt beam search
  - Worse performance than self-consistency
- Diversity is key



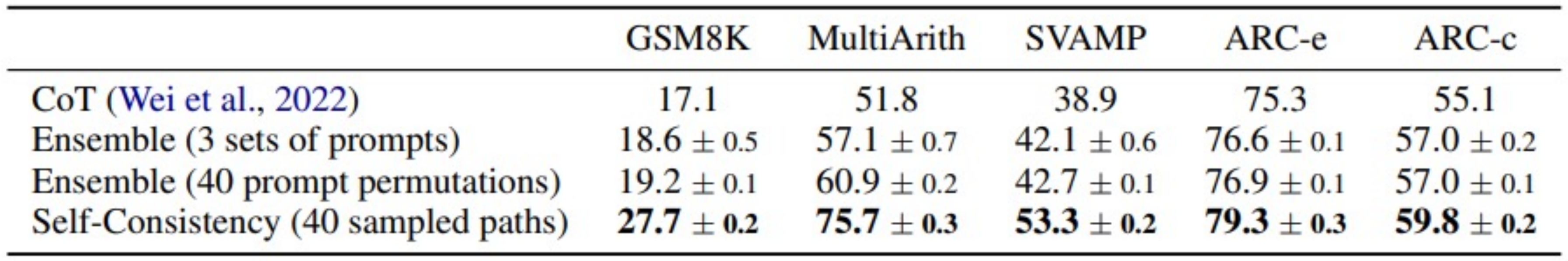
	Beam size / Self-consistency paths	1	5	10	20	40
AQuA	Beam search decoding (top beam)	23.6	19.3	16.1	15.0	10.2
	Self-consistency using beam search	23.6	19.8 ± 0.3	21.2 ± 0.7	24.6 ± 0.4	24.2 ± 0.5
	Self-consistency using sampling	19.7 ± 2.5	**24.9 ± 2.6**	**25.3 ± 1.8**	**26.7 ± 1.0**	**26.9 ± 0.5**
MultiArith	Beam search decoding (top beam)	10.7	12.0	11.3	11.0	10.5
	Self-consistency using beam search	10.7	11.8 ± 0.0	11.4 ± 0.1	12.3 ± 0.1	10.8 ± 0.1
	Self-consistency using sampling	9.5 ± 1.2	11.3 ± 1.2	**12.3 ± 0.8**	**13.7 ± 0.9**	**14.7 ± 0.3**

Table 6: Compare self-consistency with beam search decoding on the UL2-20B model.

UNIVERSITY of VIRGINIA

Self-Consistency vs Ensemble-Based Approaches

- Methods of ensembling
  - Prompt order permutation
  - Multiple sets of prompts
  - Majority vote used
- Self-consistency acts like a "self-ensemble"

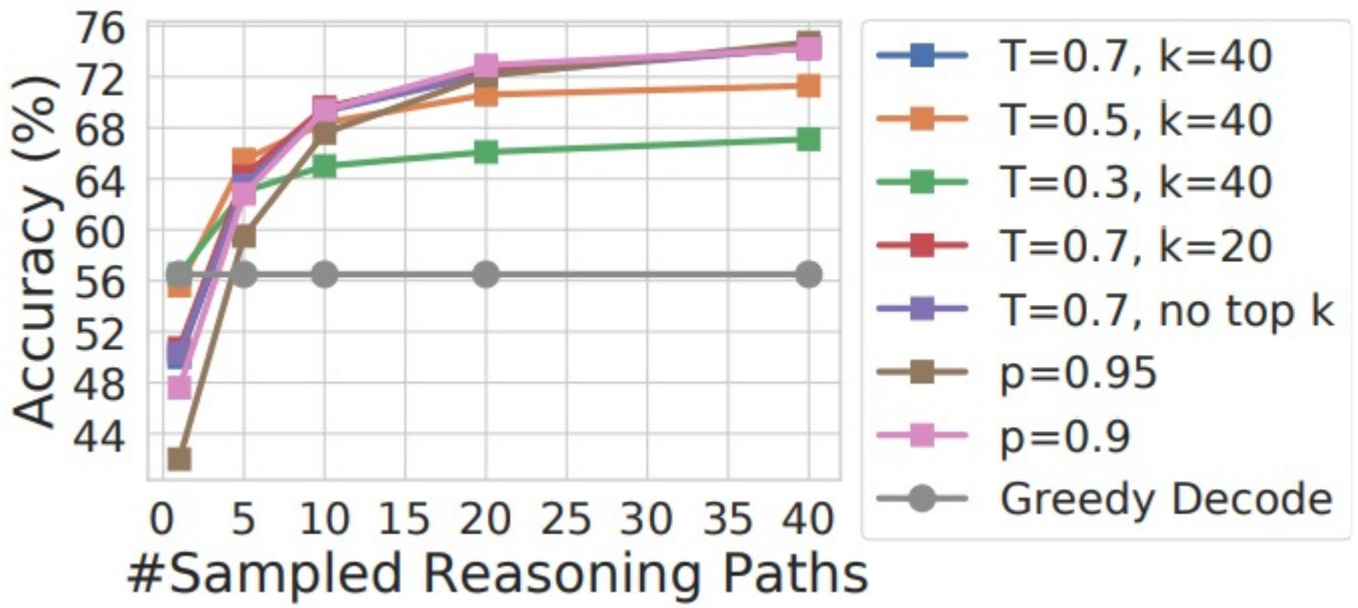


	GSM8K	MultiArith	SVAMP	ARC-e	ARC-c
CoT (Wei et al., 2022)	17.1	51.8	38.9	75.3	55.1
Ensemble (3 sets of prompts)	18.6 $\pm$ 0.5	57.1 $\pm$ 0.7	42.1 $\pm$ 0.6	76.6 $\pm$ 0.1	57.0 $\pm$ 0.2
Ensemble (40 prompt permutations)	19.2 $\pm$ 0.1	60.9 $\pm$ 0.2	42.7 $\pm$ 0.1	76.9 $\pm$ 0.1	57.0 $\pm$ 0.1
Self-Consistency (40 sampled paths)	**27.7** $\pm$ **0.2**	**75.7** $\pm$ **0.3**	**53.3** $\pm$ **0.2**	**79.3** $\pm$ **0.3**	**59.8** $\pm$ **0.2**

Table 7: Self-consistency outperforms prompt-order and multi-prompt ensembles on LaMDA-137B.

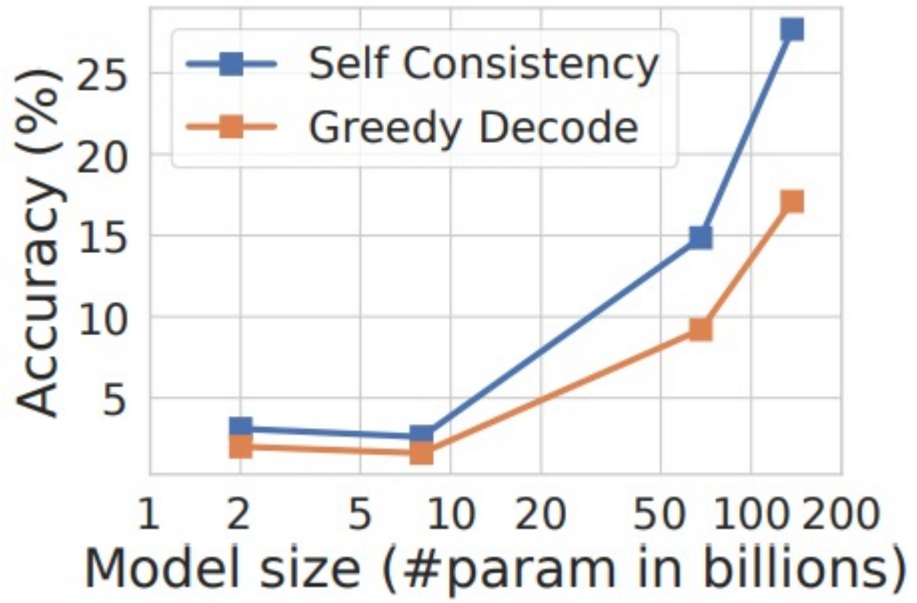
Robustness to Sampling Strategies

- Robust to sampling strategies and parameters
  - Temperature
  - k in top-k sampling
  - p in nucleus sampling



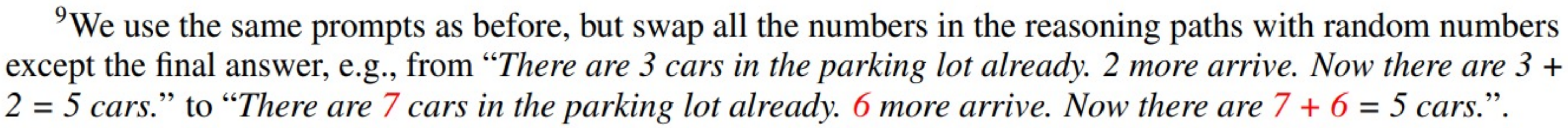
Robustness to Scaling

- Robust to scaling
- Gain relatively lower for smaller models
    - Certain abilities only emerge upon sufficient model scale

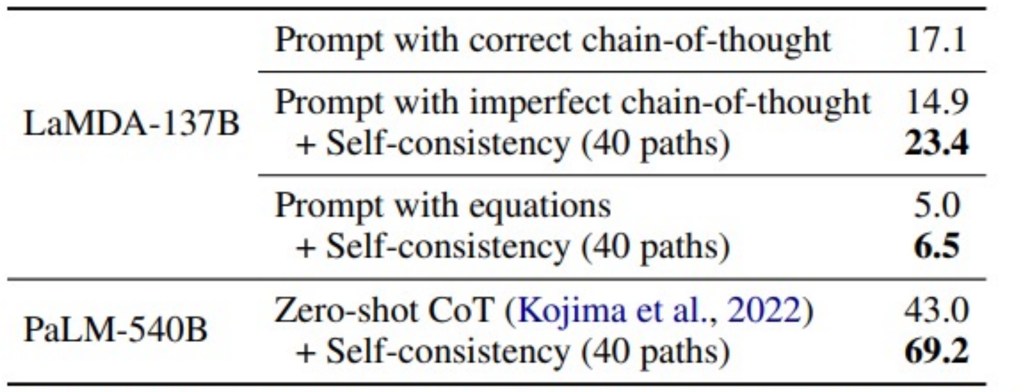


Prompt Robustness

- Improves robustness to imperfect prompts
  - Mistakes can lead to lower greedy accuracy
  - Self-consistency can fill in the gaps and improve results



[9]We use the same prompts as before, but swap all the numbers in the reasoning paths with random numbers except the final answer, e.g., from "*There are 3 cars in the parking lot already. 2 more arrive. Now there are 3 + 2 = 5 cars.*" to "*There are 7 cars in the parking lot already. 6 more arrive. Now there are 7 + 6 = 5 cars.*".



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.

Self-Consistency Robustness (cont.)

- Consistency highly correlated with accuracy
  - % of decodes agreeing with final aggregated answer
- Self-consistency can be used to provide uncertainty estimate of the model
  - Confers some ability for model to "know when it doesn't know"

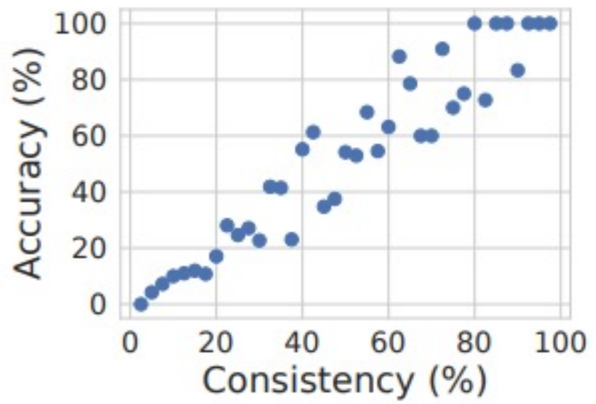
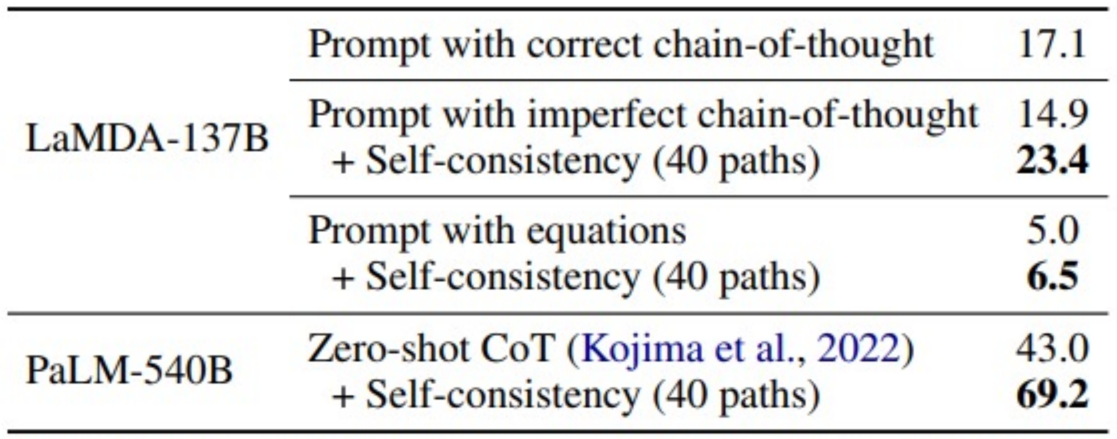


Figure 5: The consistency is correlated with model's accuracy.

Non NL Reasoning Paths

- Tested generality to non-natural-language paths
  - Such as equations
- Intermediate equations generated
- Gain is smaller when compared to natural language
  - Less opportunity for diversity



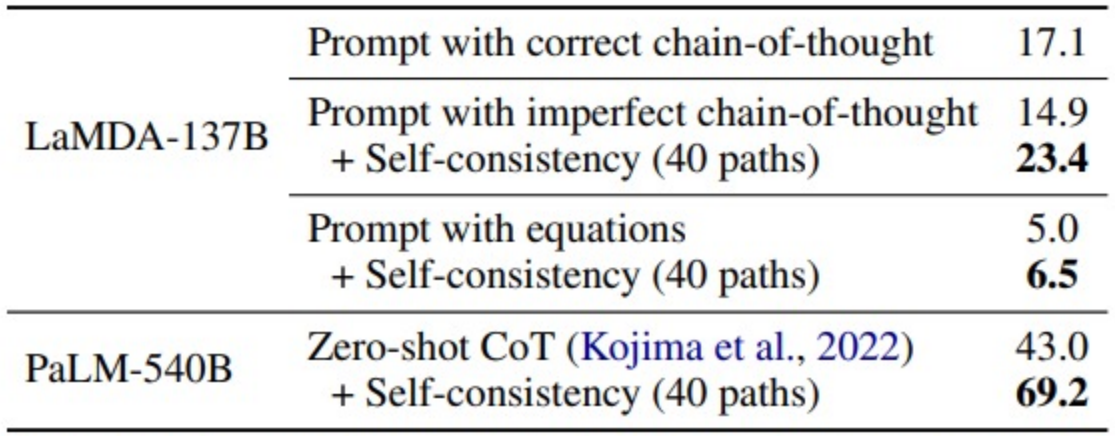
LaMDA-137B	Prompt with correct chain-of-thought	17.1
	Prompt with imperfect chain-of-thought	14.9
	+ Self-consistency (40 paths)	**23.4**
	Prompt with equations	5.0
	+ Self-consistency (40 paths)	**6.5**
PaLM-540B	Zero-shot CoT (Kojima et al., 2022)	43.0
	+ Self-consistency (40 paths)	**69.2**

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

UNIVERSITY of VIRGINIA

Zero-Shot Learning

- Improves performance for zero-shot chain of thought
- "Let's think step by step"



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.

Related Work

- Language models struggle with Type 2 tasks
  - Arithmetic, logical, commonsense reasoning
  - Previous work focused on specialized approaches
- Re-ranking
  - Requires training of additional ranker
- Self-consistency more widely applicable

Discussion

- Self-consistency improves task accuracy
  - Collect multiple reasoning rationales
  - Provide uncertainty estimates
- Limitations
  - Computational Cost
- Use self-consistency to generate better supervised data
  - Fine-tuning
- Language models sometimes generate nonsensical reasoning paths
  - Better ground models' rationale generations

Paper 2

Augmented Language Models: a Survey

Presenters:
Jessie Chen (hc4vb)
Soneya Hossain (sh7hv)

Jessie Chen (hc4vb)

Introduction -- Motivation

- LLMs suffer from limitations hindering a broader deployment
  - Hallucinations
  - Size and need for data can be impractical for training and maintenance

- LLMs are generally trained to perform statistical language modeling given a limited context.

- Augmenting LMs with both reasoning and tools may lead to more powerful agents. **Augmented Language Models (ALMs)**

Introduction -- Definitions

- **Reasoning**
  - the ability to make inferences using evidence and logic
  - e.g. Chain-of-Thought

- **Tool**
  - External module that is typically called using a rule or a special token
  - e.g. search engine

- **Act**
  - LM performs an action when the tool has an effect on the external world
  - e.g. robotic arm manipulation via LMs

Reasoning

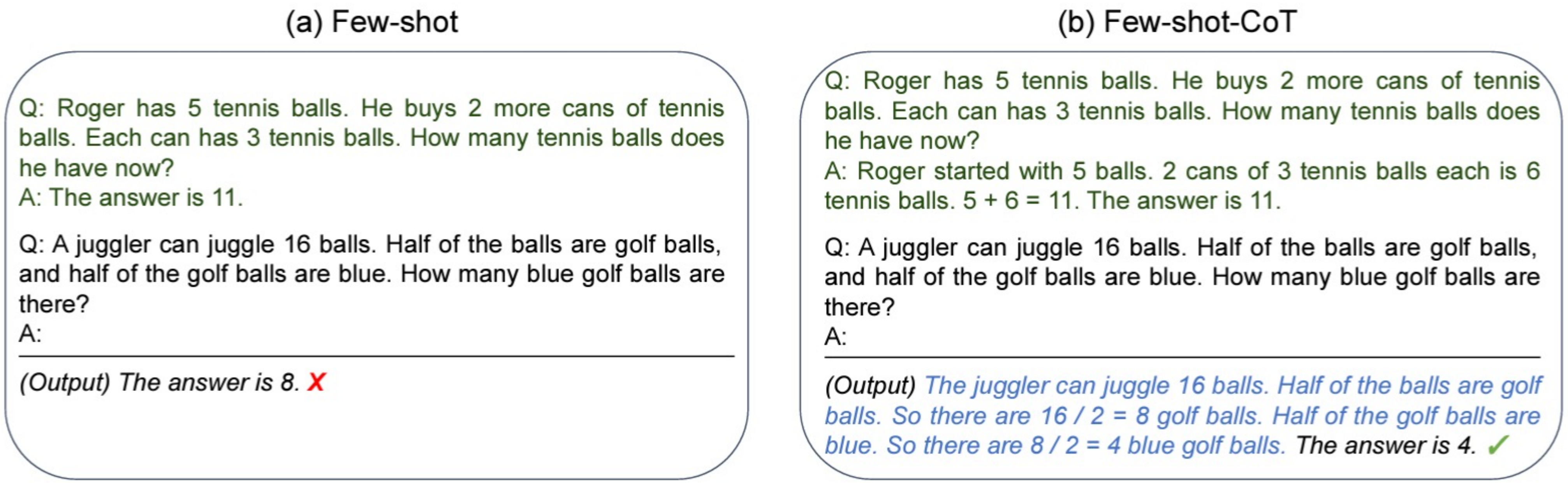
- **Eliciting reasoning with prompting**
  - elicitive prompts encourage LMs to solve tasks by following intermediate steps before predicting the output/answer

- **Recursive Prompting**
  - Explicitly decomposing problems into subproblems

- **Explicitly teaching language models to reason**
  - training LMs to use, as humans, a working memory when more than one step are required

Reasoning -- Eliciting reasoning with prompting

- **Few-shot Setting**
    - Prompt the model with a few input-output exemplars demonstrating the task.
    - Chain-of-Thought
        - prompt consists of <input, chain of thought, output>.



(a) Few-shot

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: 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:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

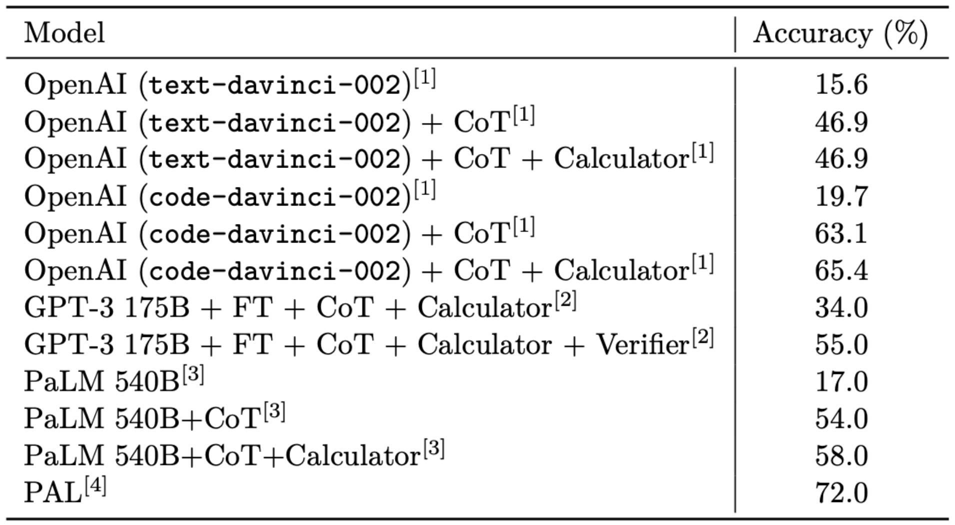
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: 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:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

Reasoning -- Eliciting reasoning with prompting

- CoT performance

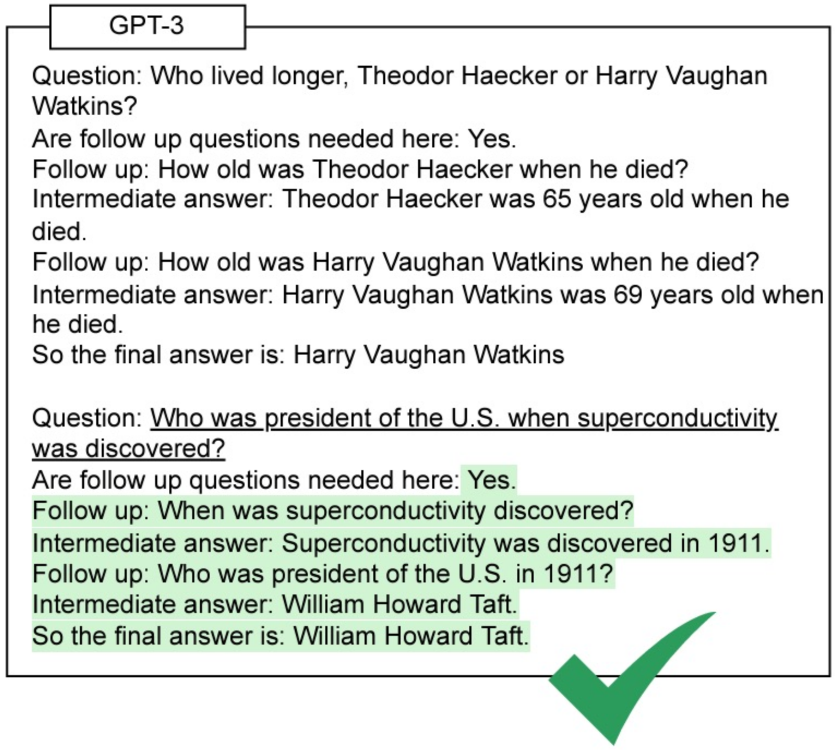


Model	Accuracy (%)
OpenAI (`text-davinci-002`)[1]	15.6
OpenAI (`text-davinci-002`) + CoT[1]	46.9
OpenAI (`text-davinci-002`) + CoT + Calculator[1]	46.9
OpenAI (`code-davinci-002`)[1]	19.7
OpenAI (`code-davinci-002`) + CoT[1]	63.1
OpenAI (`code-davinci-002`) + CoT + Calculator[1]	65.4
GPT-3 175B + FT + CoT + Calculator[2]	34.0
GPT-3 175B + FT + CoT + Calculator + Verifier[2]	55.0
PaLM 540B[3]	17.0
PaLM 540B+CoT[3]	54.0
PaLM 540B+CoT+Calculator[3]	58.0
PAL[4]	72.0

Table 1: Evaluation of different reasoning methods on GSM8K, a popular reasoning benchmark. FT denotes fine-tuning and CoT denotes chain-of-thought. The reported accuracies are based on [1]: (Wei et al., 2022c); [2]: (Cobbe et al., 2021); [3]: (Chowdhery et al., 2022); and [4]: (Gao et al., 2022).

- Other Variations
  - Self-Ask
  - Self-Consistency

Self-Ask



GPT-3

Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
Are follow up questions needed here: Yes.
Follow up: How old was Theodor Haecker when he died?
Intermediate answer: Theodor Haecker was 65 years old when he died.
Follow up: How old was Harry Vaughan Watkins when he died?
Intermediate answer: Harry Vaughan Watkins was 69 years old when he died.
So the final answer is: Harry Vaughan Watkins

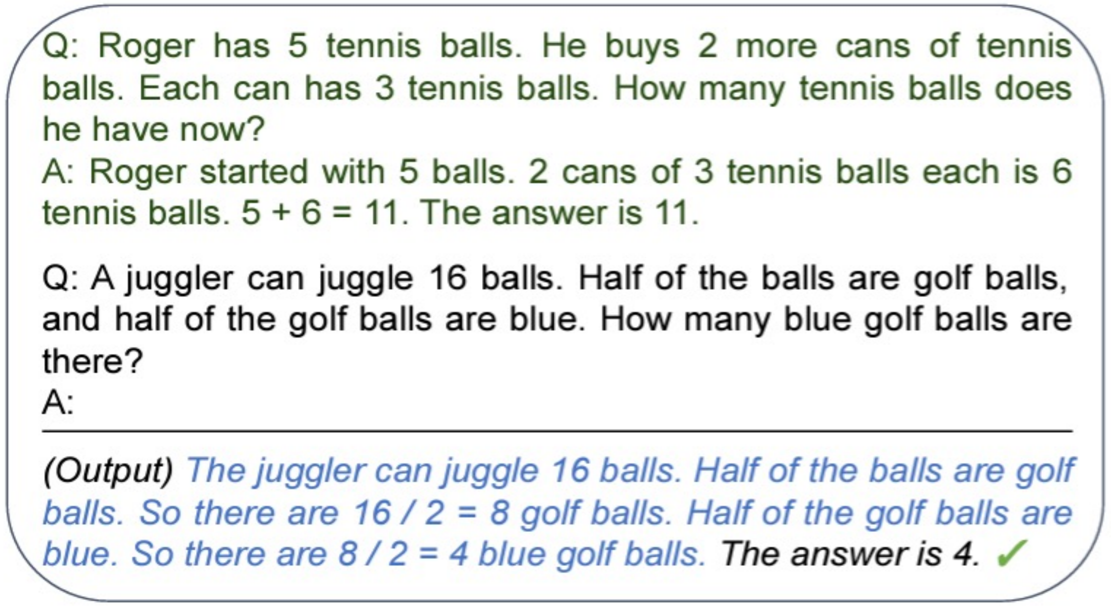
Question: Who was president of the U.S. when superconductivity was discovered?
Are follow up questions needed here: Yes.
Follow up: When was superconductivity discovered?
Intermediate answer: Superconductivity was discovered in 1911.
Follow up: Who was president of the U.S. in 1911?
Intermediate answer: William Howard Taft.
So the final answer is: William Howard Taft.

Reasoning -- Eliciting reasoning with prompting

- **Zero-shot Setting**
  - conditions the LM on a single prompt that is not an example

  - Zero-shot-CoT
    - Add "Let's think step by step" or a similar text before querying the model.

(b) Few-shot-CoT

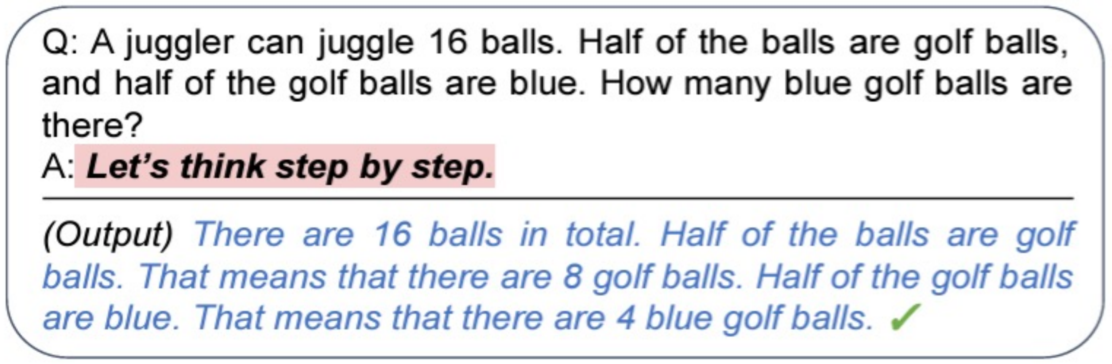


Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: 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:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(d) Zero-shot-CoT (Ours)



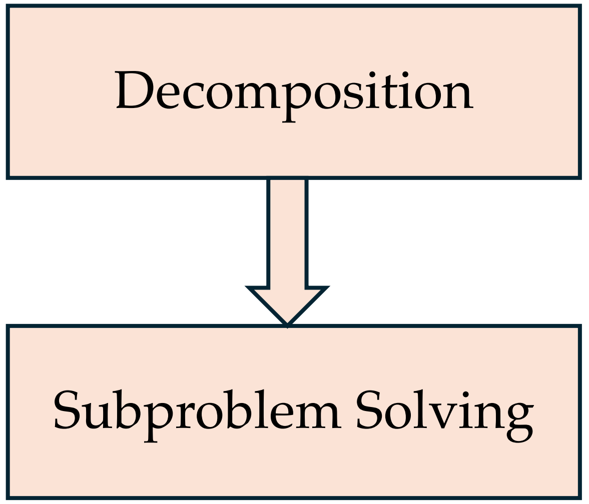
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

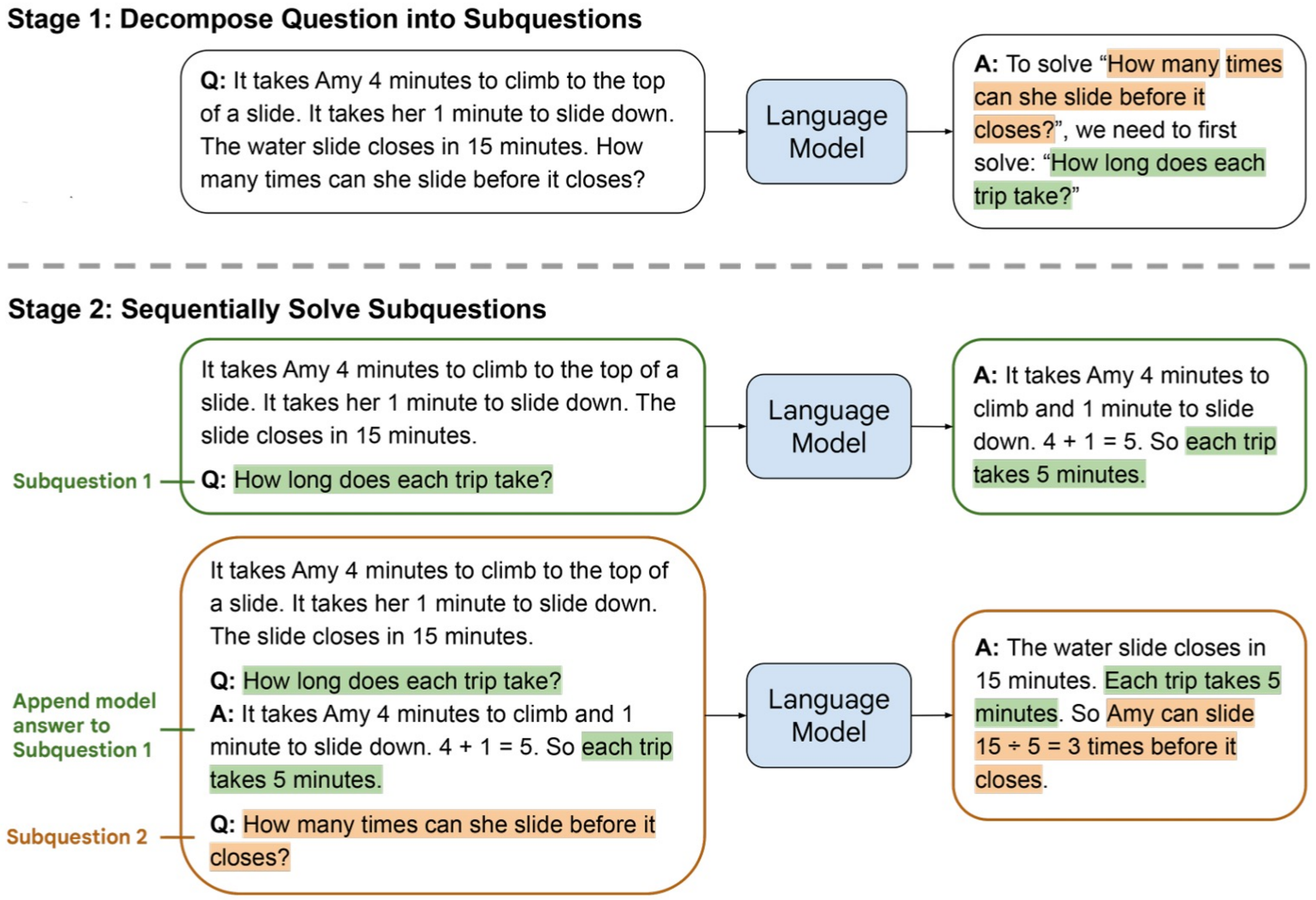
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

UNIVERSITY *of* VIRGINIA

Reasoning – Recursive Prompting

- **Problem Decomposition**
    - Least-to-most Prompting




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.

Reasoning – Recursive Prompting

- **Problem Decomposition**
  - Decomposed Prompting

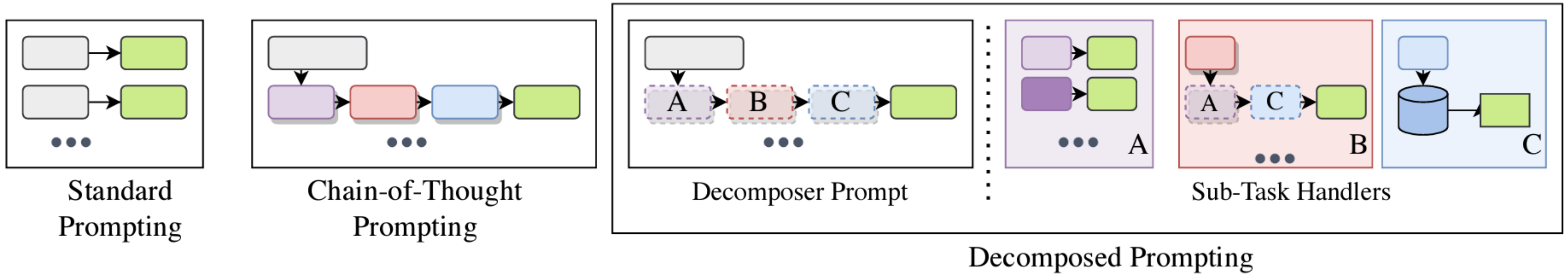
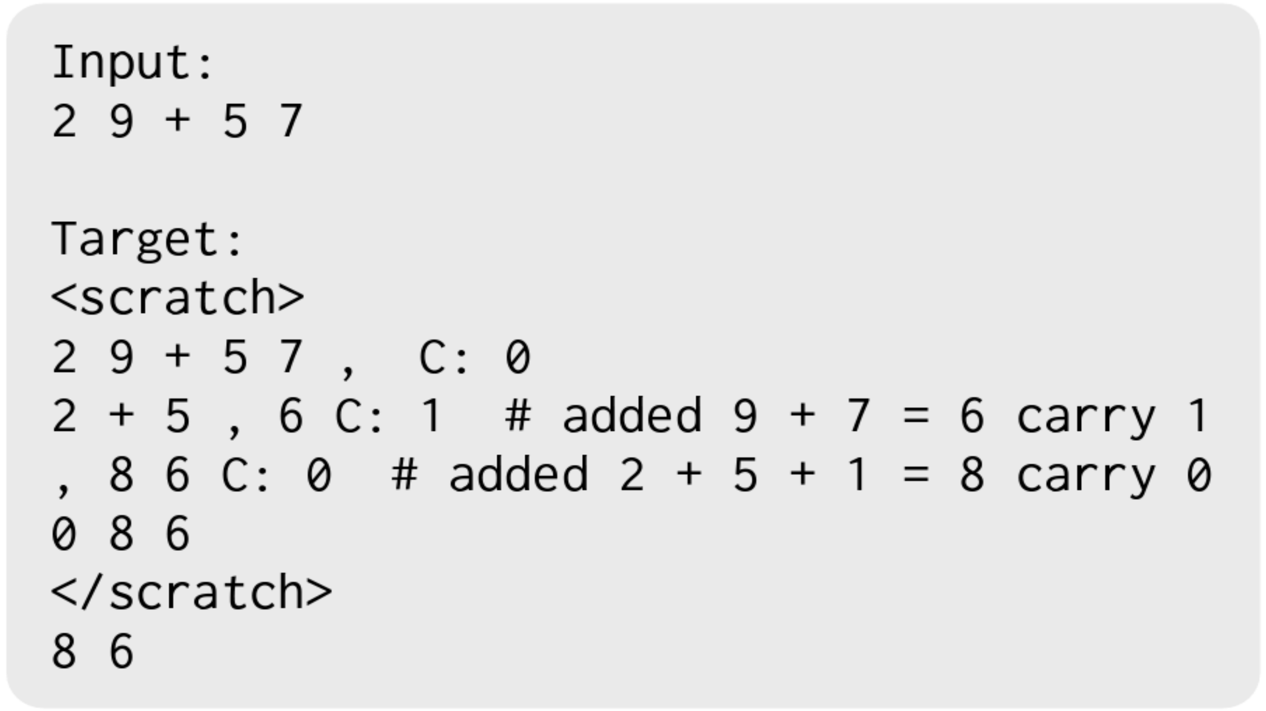


Figure 1: While standard approaches only provide labeled examples (shown as a grey input box with green label box), Chain-of-Thought prompting also describes the reasoning steps to arrive at the answer for every example in the prompt. Decomposed Prompting, on the other hand, uses the decomposer prompt to only describe the procedure to solve the complex tasks using certain sub-tasks. Each sub-task, indicated here with A, B and C is handled by sub-task specific handlers which can vary from a standard prompt (sub-task A), a further decomposed prompt (sub-task B) or a symbolic function such as retrieval (sub-task C)

Reasoning – Explicitly teaching language models to reason

- **Working Memory**
  - Scratchpad
  - train Transformers to perform multi-step computations by asking them to emit intermediate computation steps into a "scratchpad"



```
Input:
2 9 + 5 7

Target:
<scratch>
2 9 + 5 7 ,  C: 0
2 + 5 , 6 C: 1  # added 9 + 7 = 6 carry 1
, 8 6 C: 0  # added 2 + 5 + 1 = 8 carry 0
0 8 6
</scratch>
8 6
```

Figure 2: Example of input and target for addition with a scratchpad. The carry is recorded in the digit following "C:". Comments (marked by #) are added for clarity and are not part of the target.

Reasoning – Limitations

- To what extent LMs use the reasoning steps to support the final prediction remains poorly understood
- Reasoning steps may suffer from avoidable mistakes

Leverage external tools

Using Tool and Act

- Calling another Model

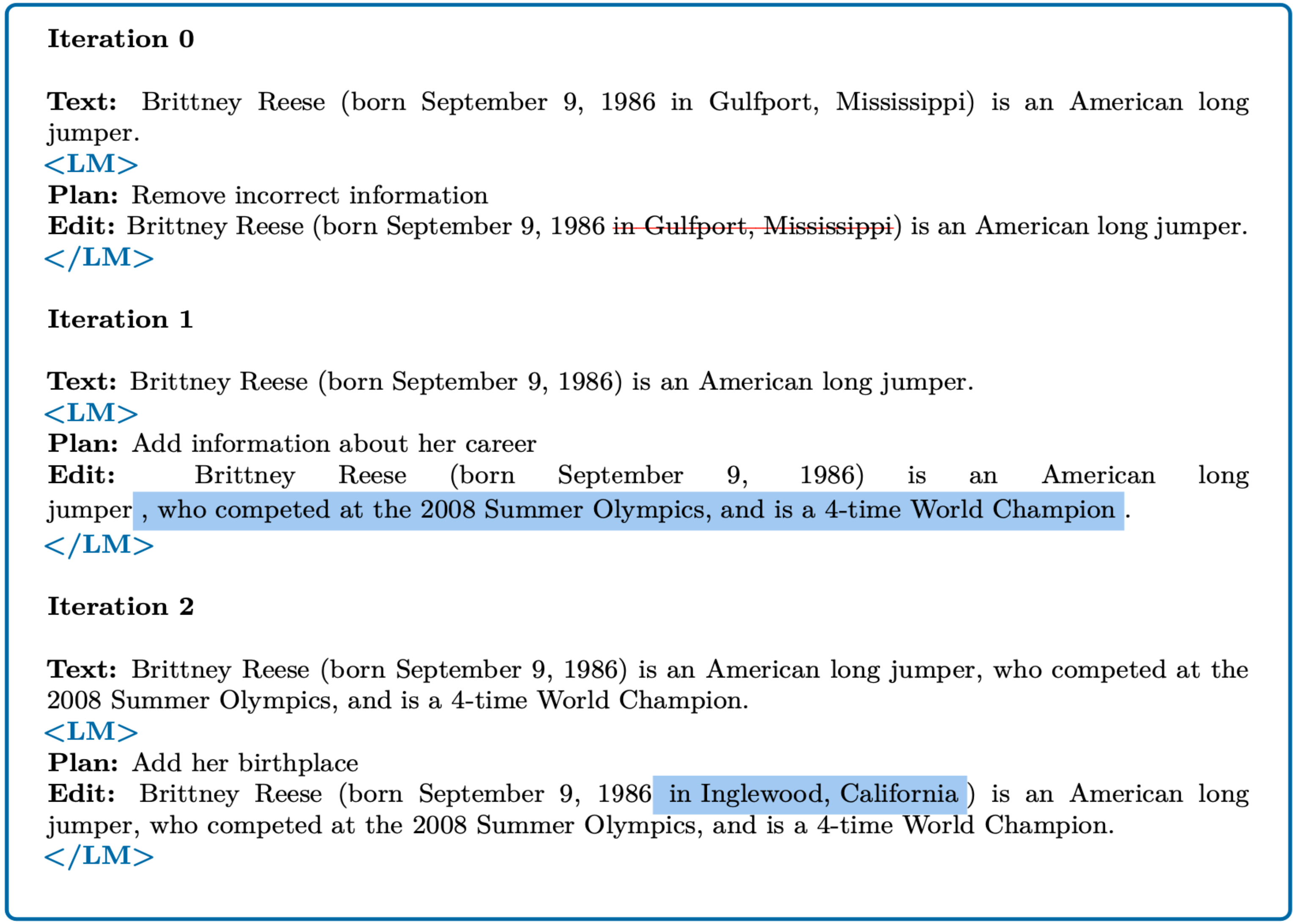
- Information Retrieval

- Computing via Symbolic Modules and Code Interpreters

- Acting on the Virtual and Physical World

Using Tools and Act – Calling another model

- **Iterative LM Calling**
  - PEER
  - an LM trained to produce a plan of action and edit the input text at each step.



Iteration 0

Text: Brittney Reese (born September 9, 1986 in Gulfport, Mississippi) is an American long jumper.
<LM>
Plan: Remove incorrect information
Edit: Brittney Reese (born September 9, 1986 ~~in Gulfport, Mississippi~~) is an American long jumper.
</LM>

Iteration 1

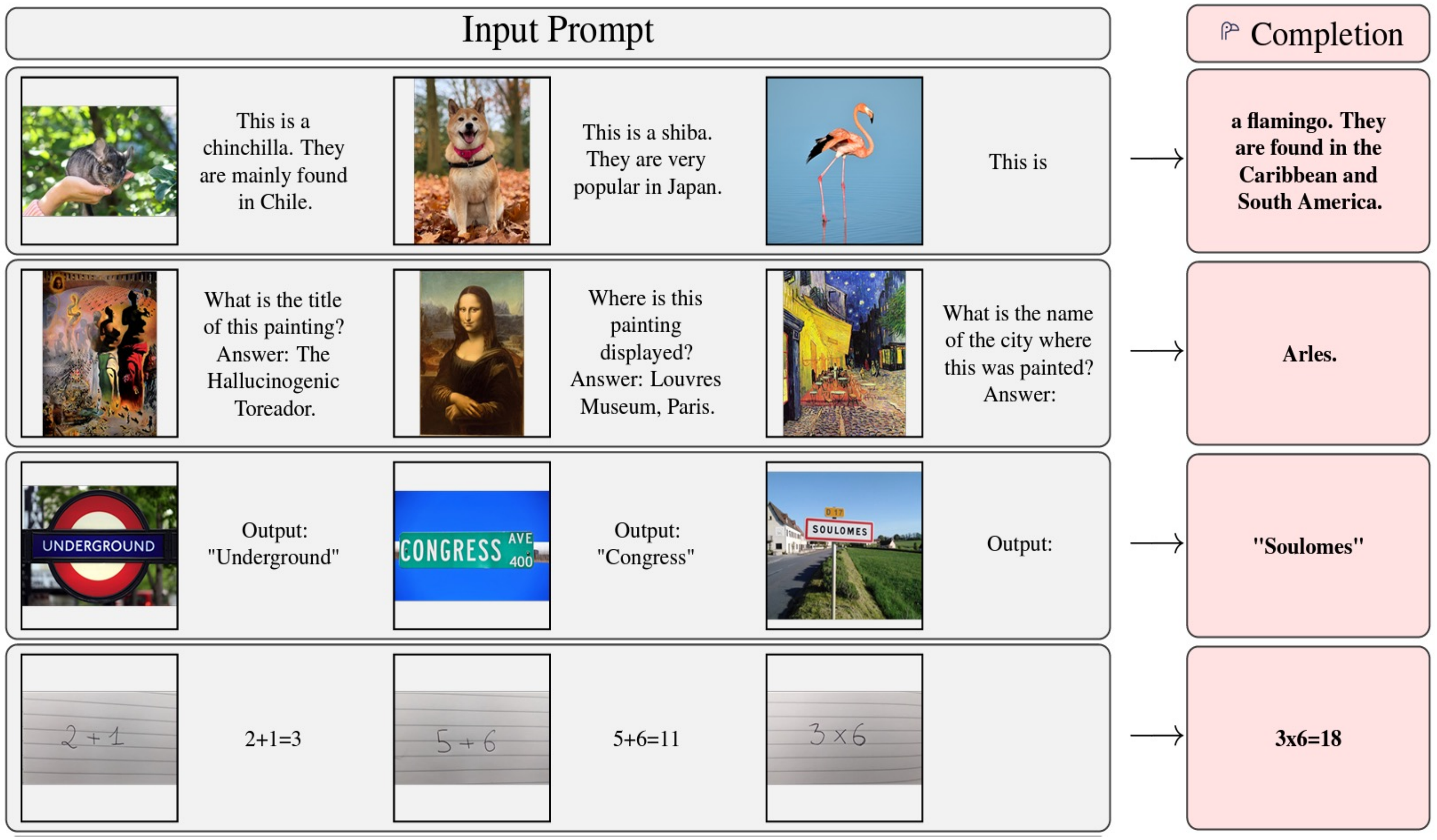
Text: Brittney Reese (born September 9, 1986) is an American long jumper.
<LM>
Plan: Add information about her career
Edit: Brittney Reese (born September 9, 1986) is an American long jumper , who competed at the 2008 Summer Olympics, and is a 4-time World Champion .
</LM>

Iteration 2

Text: Brittney Reese (born September 9, 1986) is an American long jumper, who competed at the 2008 Summer Olympics, and is a 4-time World Champion.
<LM>
Plan: Add her birthplace
Edit: Brittney Reese (born September 9, 1986 in Inglewood, California) is an American long jumper, who competed at the 2008 Summer Olympics, and is a 4-time World Champion.
</LM>

Using Tools and Act – Calling another model

- **Leveraging other modalities.**
  - Flamingo
  - Visual Language Models (VLMs) are trained on large-scale multimodal web corpora containing interleaved text and images, and they display few-shot learning capabilities of multimodal tasks.



Input Prompt						Completion
	This is a chinchilla. They are mainly found in Chile.		This is a shiba. They are very popular in Japan.		This is	a flamingo. They are found in the Caribbean and South America.
	What is the title of this painting? Answer: The Hallucinogenic Toreador.		Where is this painting displayed? Answer: Louvres Museum, Paris.		What is the name of the city where this was painted? Answer:	Arles.
	Output: "Underground"		Output: "Congress"		Output:	"Soulomes"
	2+1=3		5+6=11			3x6=18

Using Tools and Act – Information retrieval

- **Retrieval-Augmented Language models**
  - Conditioning LMs on retrieved documents.
  - RAG

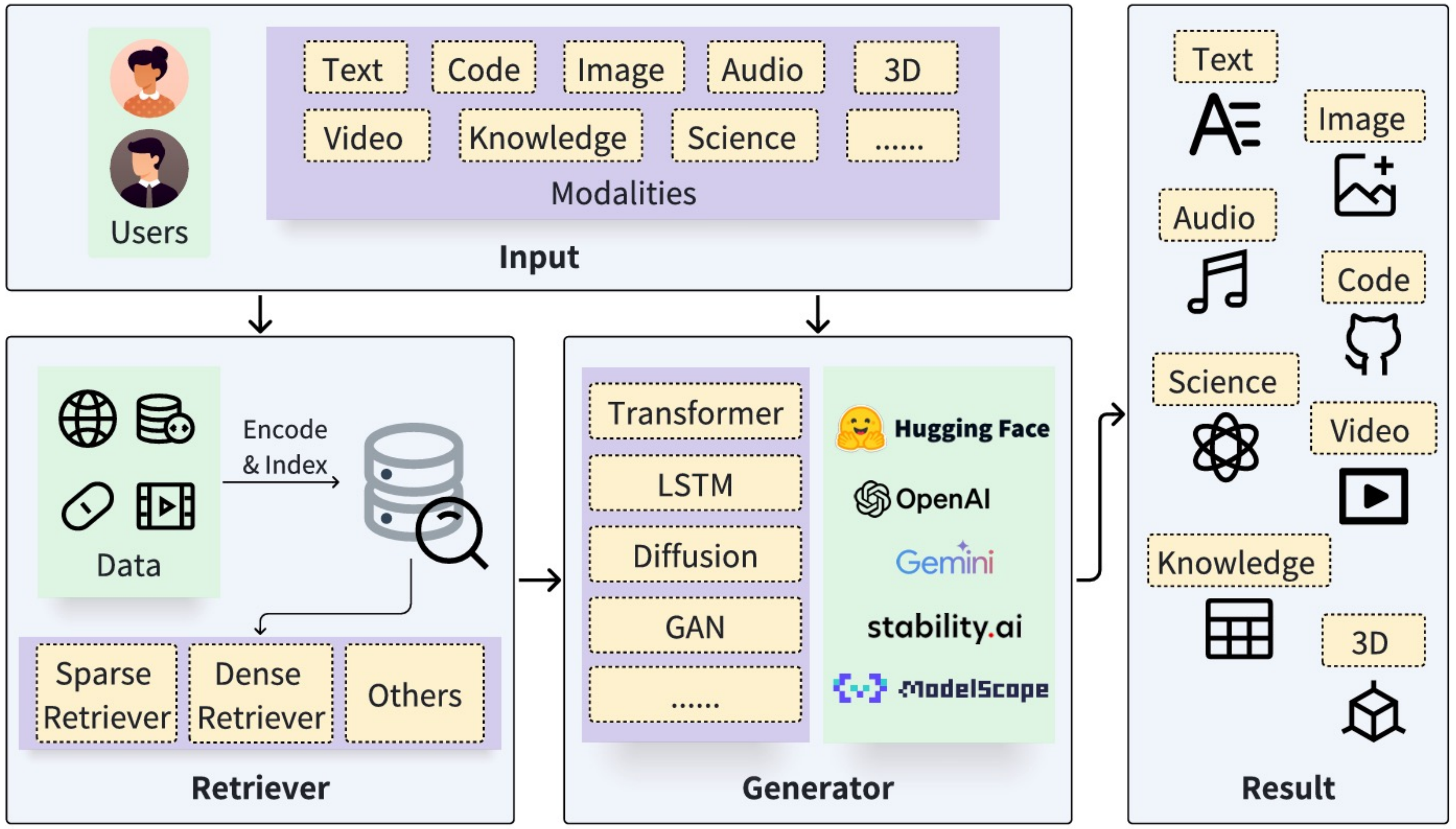


Fig. 1. A generic RAG architecture. The users' queries, which may be different modality, serve as input to both the retriever and the generator. The retriever searches for relevant data sources in storage, while the generator interacts with the retrieval results and ultimately produces results of various modalities.

Soneya Binta Hossain (sh7hv)

Using Tools and Act – Information retrieval

Querying search engines:

- passive entity ⟹ an active agent
- generates queries based on prompts, enlarging its action space.

Example Agents:

LaMDA:

- pre-trained on dialogues and web documents
- augmented with **retrieval** capabilities, a **calculator**, and a **translator** to enhance factual grounding.

Using Tools and Act – Information retrieval

Querying search engines:

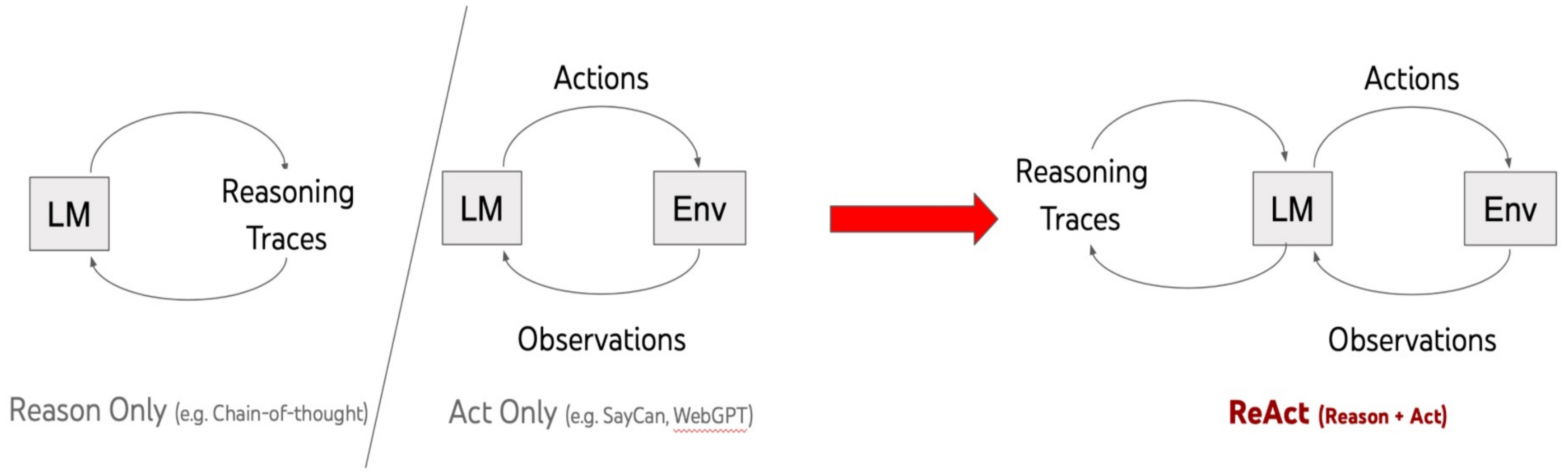
ReAct:

- Agent-like LM augmented with information retrieval abilities by querying search engines.
- Performs **reasoning** and **acting** in an **interleaved manner**, allowing greater synergy. Reasoning can help generate action plans **(reason to act)** and action can retrieve information from external sources, further improving the model's reasoning abilities **(act to reason)**.
- Few-shot prompting for teaching LM to use different tools: **search** and Wikipedia **lookup**
- Performs well on diverse language reasoning and decision-making tasks: question answering (HotPotQA), fact verification (Fever), text-based game (ALFWorld) and navigation (WebShop).

Using Tools and Act – Information retrieval

Querying search engines:

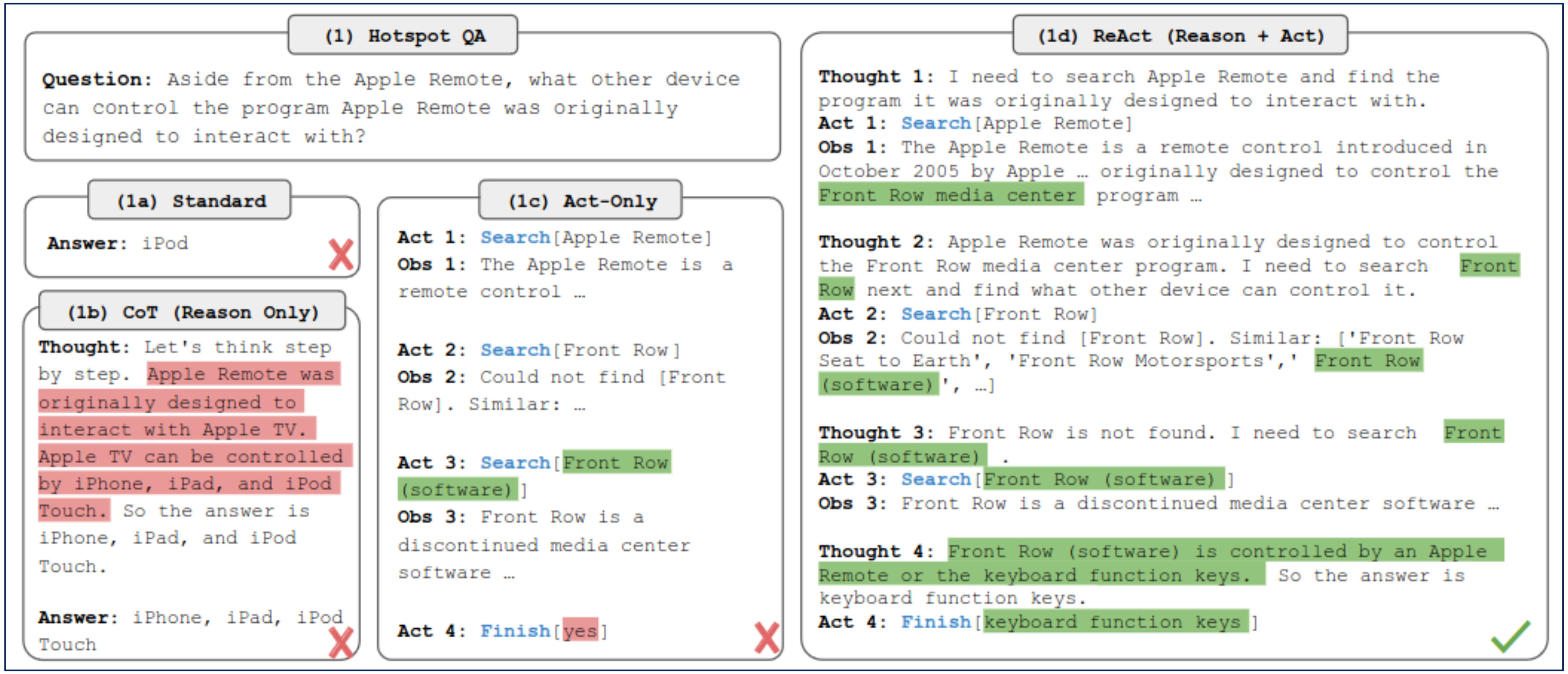
ReAct:



Using Tools and Act – Information retrieval

Querying search engines:

ReAct:



(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod ✗

(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch ✗

(1c) Act-Only

Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control …

Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: …

Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Act 4: Finish[yes] ✗

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software) ', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

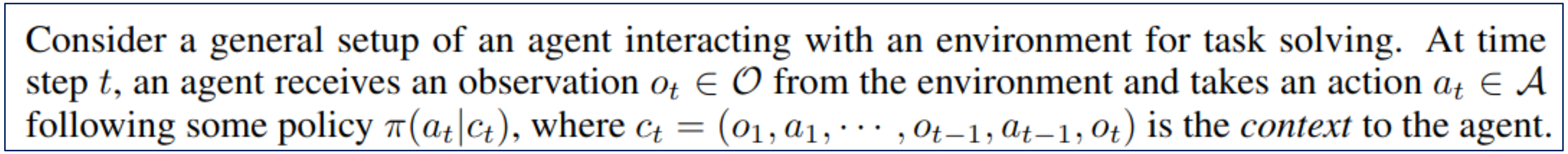
Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys] ✓

Figure: Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT, Reason Only), (c) Act-only, and (d) ReAct (Reason+Act), solving a HotpotQA (Yang et al., 2018) question.

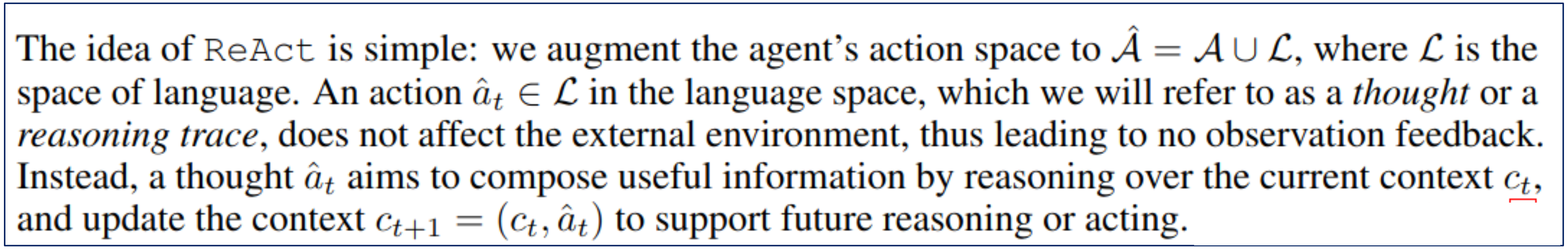
Using Tools and Act – Information retrieval

Querying search engines:

REACT : SYNERGIZING REASONING + ACTING



Consider a general setup of an agent interacting with an environment for task solving. At time step t, an agent receives an observation $o_t \in \mathcal{O}$ from the environment and takes an action $a_t \in \mathcal{A}$ following some policy $\pi(a_t|c_t)$, where $c_t = (o_1, a_1, \cdots, o_{t-1}, a_{t-1}, o_t)$ is the *context* to the agent.



The idea of `ReAct` is simple: we augment the agent's action space to $\hat{\mathcal{A}} = \mathcal{A} \cup \mathcal{L}$, where $\mathcal{L}$ is the space of language. An action $\hat{a}_t \in \mathcal{L}$ in the language space, which we will refer to as a *thought* or a *reasoning trace*, does not affect the external environment, thus leading to no observation feedback. Instead, a thought $\hat{a}_t$ aims to compose useful information by reasoning over the current context c_t, and update the context $c_{t+1} = (c_t, \hat{a}_t)$ to support future reasoning or acting.

Using Tools and Act – Information retrieval

Querying search engines:

REACT : SYNERGIZING RE ASONING + ACTING

- Used frozen large language model, PaLM-540B (Chowdhery et al., 2022)
- Prompted with few-shot in-context examples to generate both domain-specific actions and free-form language thoughts for task solving
- Each in-context example is a human trajectory of actions, thoughts, and environment observations to solve a task instance

Using Tools and Act – Information retrieval

REACT : SETUP

Domains :

- **HotPotQA**, a multi-hop question answering benchmark requiring reasoning over two or more Wikipedia passage
- **FEVER**, a fact verification benchmark where each claim is annotated SUPPORTS, REFUTES, or NOT ENOUGH INFO, based on if there exists a Wikipedia passage to verify the claim

Action Space: search[entity] from wiki page if it exists, **lookup**[string], which would return the next sentence in the page containing string, **finish**[answer]

METHODS:

- ReAct Prompting : Randomly select 6 and 3 cases from the training set and manually compose ReAct-format trajectories to use as few-shot exemplars in the prompts

- Baselines: standar, CoT, Act Only, CoT-SC

- ReAct → CoT-SC

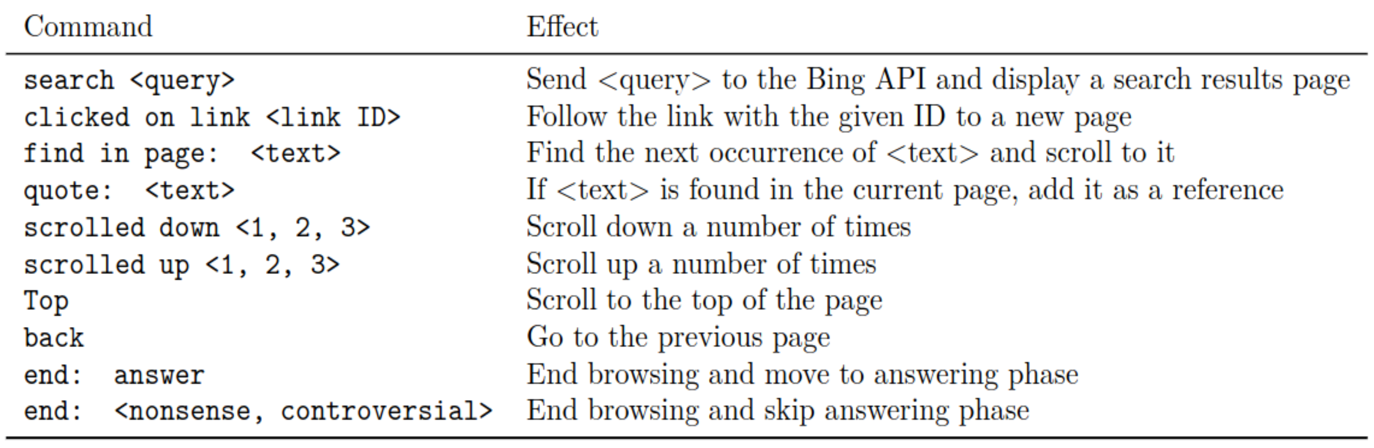
Finetuning : 1) Bootstrap approach 2) 3,000 correct trajectories via ReAct, 3) Finetune PaLM-8/62B, 4) Decode thoughts, actions, observations from questions/claims.

Using Tools and Act – Information retrieval

Searching and navigating the web:

WebGPT:

- LLM agent that can interact with open-ended internet to pursue different goals: searching information or purchasing items
- Navigating the web can further refine queries and additional actions beyond simple searches
- **Search** internet, **navigate** webpage, **follow** links and **cite** sources
- Fine-tuned GPT-3 model on human demonstration, reinforcement learning to predict human preferences
- Surpasses human question-answering abilities



Command	Effect
`search <query>`	Send <query> to the Bing API and display a search results page
`clicked on link <link ID>`	Follow the link with the given ID to a new page
`find in page: <text>`	Find the next occurrence of <text> and scroll to it
`quote: <text>`	If <text> is found in the current page, add it as a reference
`scrolled down <1, 2, 3>`	Scroll down a number of times
`scrolled up <1, 2, 3>`	Scroll up a number of times
`Top`	Scroll to the top of the page
`back`	Go to the previous page
`end: answer`	End browsing and move to answering phase
`end: <nonsense, controversial>`	End browsing and skip answering phase

Table 3: The actions *WebGPT* can perform, taken from Nakano et al. (2021).

Using Tools and Act – Computing via Symbolic Modules and Code Interpreters

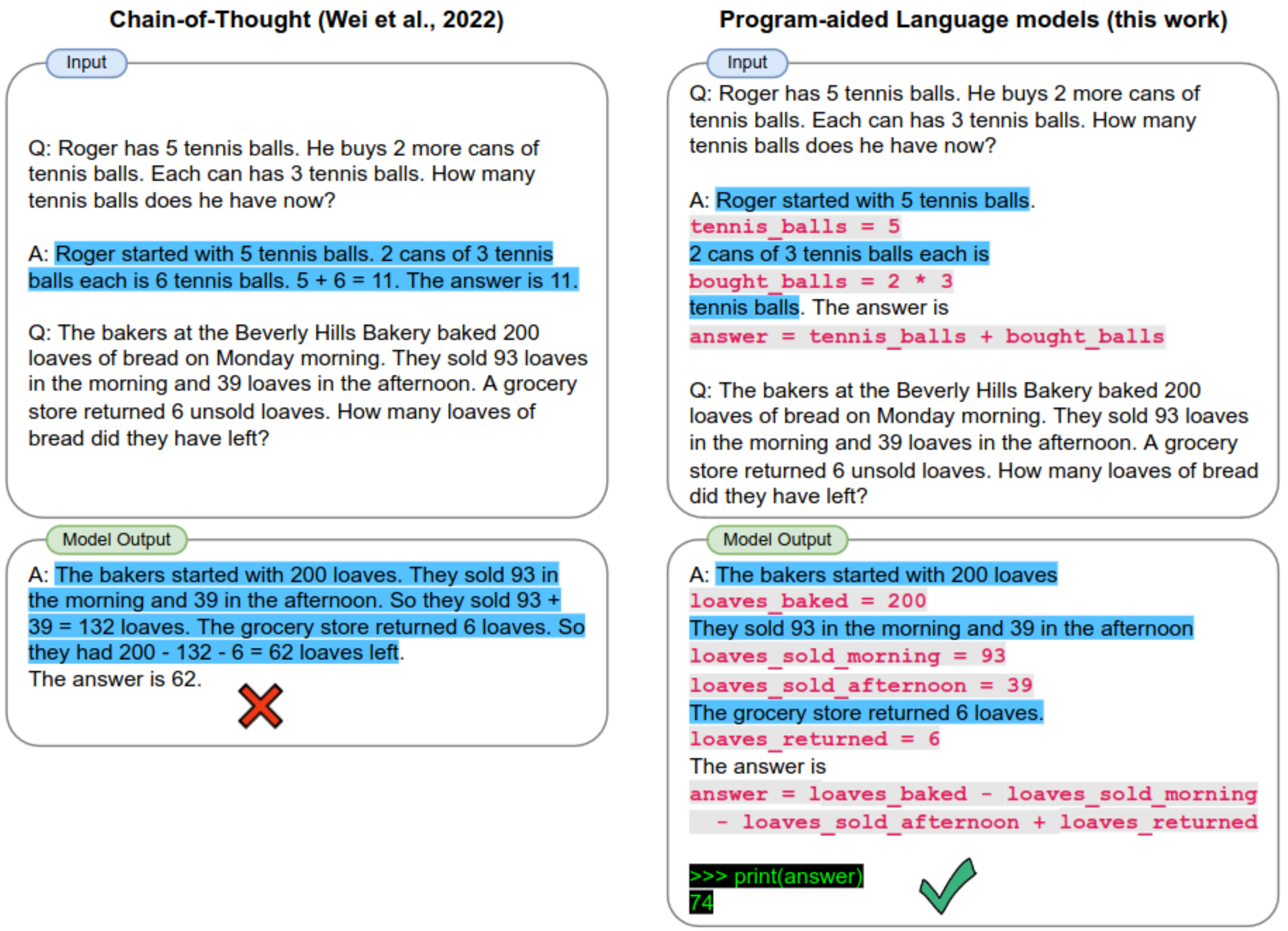
Limitations of Current LMs: LMs like GPT-3 struggles with complex arithmetic and out-of-distribution calculations

The action space of a transformer can be equipped with **Symbolic Modules** to perform arithmetic operation

⇨ **Physics Engine for Reasoning:** Mind's Eye uses a physics engine to ground physics reasoning, outperforming larger LMs in physical tasks with fewer parameters.

⇨ **CoT and Python for Reasoning:** PAL relies on CoT prompting of large LMs to decompose symbolic reasoning, mathematical reasoning, or algorithmic tasks into intermediate steps along with python code for each step

Using Tools and Act – Computing via Symbolic Modules and Code Interpreters


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 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
```
✓

Figure 1: A diagram illustrating PAL: Given a mathematical reasoning question, Chain-of-thought (left) generates intermediate reasoning steps of free-form text. In contrast, Program-aided Language models (PAL, right) generate intermediate steps *and* Python code. This shifts the role of *running* the reasoning steps from the language model to the Python interpreter. The final answer is obtained by running the generated reasoning chain. Chain-of-thought reasoning is highlighted in blue; PAL steps are `highlighted in gray and pink`; the Python interpreter run is highlighted in black and green.

Takeaway: Through innovative integrations of external tools/modules, LMs are overcoming their limitations, showcasing remarkable versatility and improved performance in complex reasoning and computational tasks.

Acting on the virtual world

LMs use tools to gather external information to improve their predictions or performance on a given task

Other tools allow the LM to act on the **virtual** or **physical** world

Acting on the physical world

Physical Robot Control:

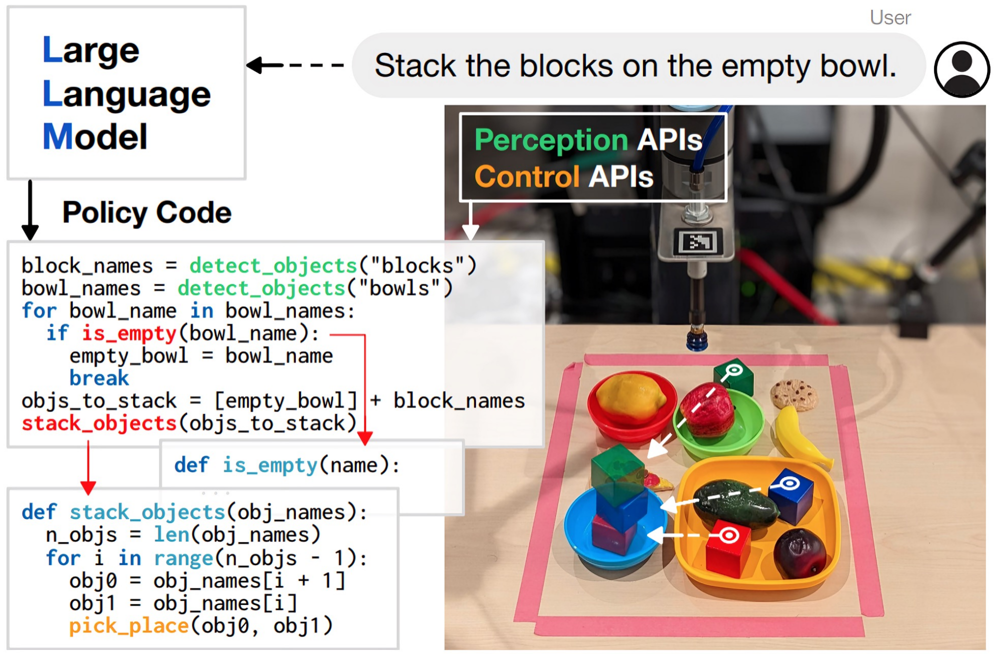
- LMs can write robot policy code given natural language commands by prompting the model with few demonstration

- LM generated policy code combines classic logic and external libraries, showcasing reasoning capabilities, generalization and precisions

- *LM lack contextual grounding for decision making.*

- **SayCan**: Teaching robots low-level skills and assessing their feasibility allows LMs to decompose complex commands into achievable tasks.

- **NLMap-SayCan for Grounding**: Integrates contextual info via a Visual Language Model (VLM), allowing for context-aware planning and task execution.



```
block_names = detect_objects("blocks")
bowl_names = detect_objects("bowls")
for bowl_name in bowl_names:
  if is_empty(bowl_name):
    empty_bowl = bowl_name
    break
objs_to_stack = [empty_bowl] + block_names
stack_objects(objs_to_stack)
```

```
def is_empty(name):
```

```
def stack_objects(obj_names):
  n_objs = len(obj_names)
  for i in range(n_objs - 1):
    obj0 = obj_names[i + 1]
    obj1 = obj_names[i]
    pick_place(obj0, obj1)
```

Learning to reason, use tools, and act

Supervision

- Few-shot prompting.
- Fine-tuning.
- **Prompt pre-training.**
- **Bootstrapping.**

Reinforcement learning

- Hard-coded reward functions.
- Human feedback.

Learning to reason, use tools, and act

Prompt pre-training:

- Mixes pre-training data with **labeled reasoning demonstrations** for a balanced approach
- Prevents deviation from the original data distribution
- Attempts to mitigate overfitting on fine-tuning examples
- Empirical gains from this method compared to separate fine-tuning are not yet clear

Learning to reason, use tools, and act

Bootstrapping:

- Some form of indirect supervision
- Typically works by prompting a LM to reason or act in a few-shot setup followed by a final prediction
- Examples for which the action or reasoning steps did not lead to a correct final prediction are then discarded
- Finally, either the original LM or a small model is fine-tuned on all correct examples
- Combines few-shot prompting's data efficiency with fine-tuning benefits
- Can be applied to teach the model to reason and act

Discussion

- Moving away from language modeling.
- A tradeoff between memorizing and querying tools.
- Generalizing the non-parametric framework.
- A path towards autonomous machine intelligence?
- Augmented Language Models benefits.
    - Truthfulness
    - Estimating and reducing uncertainty
    - Interpretability
    - Enhanced capabilities
- Ethical concerns.

Paper 3

If LLM Is the Wizard, Then Code Is the Wand: A Survey on How Code Empowers Large Language Models to Serve as Intelligent Agents

Presenters:
Ali Zafar Sadiq (mzw2cu)

Code Pretraining and Code Finetuning

Code Pretraining:

When the **code corpus** is **sourced from publicly accessible code repositories**, such as **GitHub**, it yields a **volume** comparable to that of **natural language pre-training**. We call **training with such an abundance of code as code pretraining.**

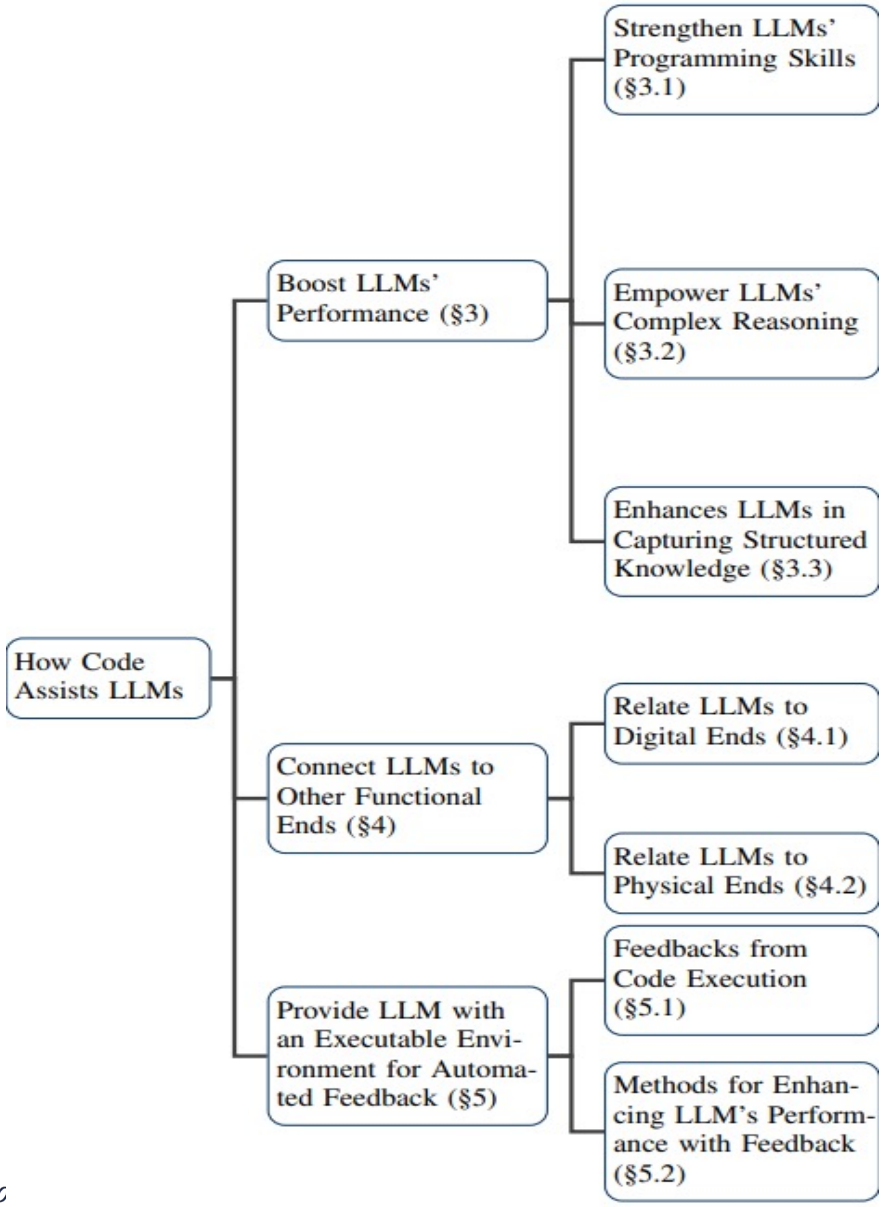
This process consists of training code on **a pre-trained natural language LLM** or or training a LLM from scratch with a blend of **natural language and code falls within code pretraining**.

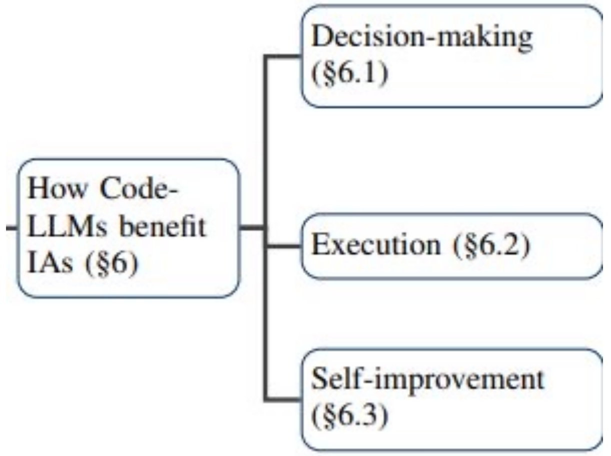
Code Finetuning:

When **dataset is smaller compared to the pre-trained natural language corpus**, we refer to such training process as **code fine-tuning**. The **objective** is to acquainting the model with **mathematical proof formulas , SQL** etc.

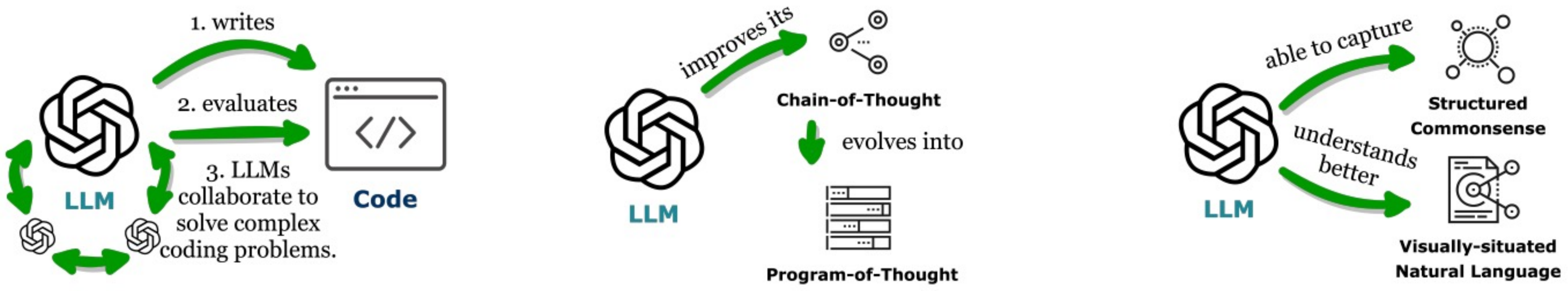
How Code Assists LLMs

How Code Benefit Intelligent Agents


How Code Assists LLMs
Boost LLMs' Performance (§3)
Strengthen LLMs' Programming Skills (§3.1)
Empower LLMs' Complex Reasoning (§3.2)
Enhances LLMs in Capturing Structured Knowledge (§3.3)
Connect LLMs to Other Functional Ends (§4)
Relate LLMs to Digital Ends (§4.1)
Relate LLMs to Physical Ends (§4.2)
Provide LLM with an Executable Envi- ronment for Automa- ted Feedback (§5)
Feedbacks from Code Execution (§5.1)
Methods for Enhan- cing LLM's Perform- ance with Feedback (§5.2)



How Code Pre-Training Boosts LLM Performance



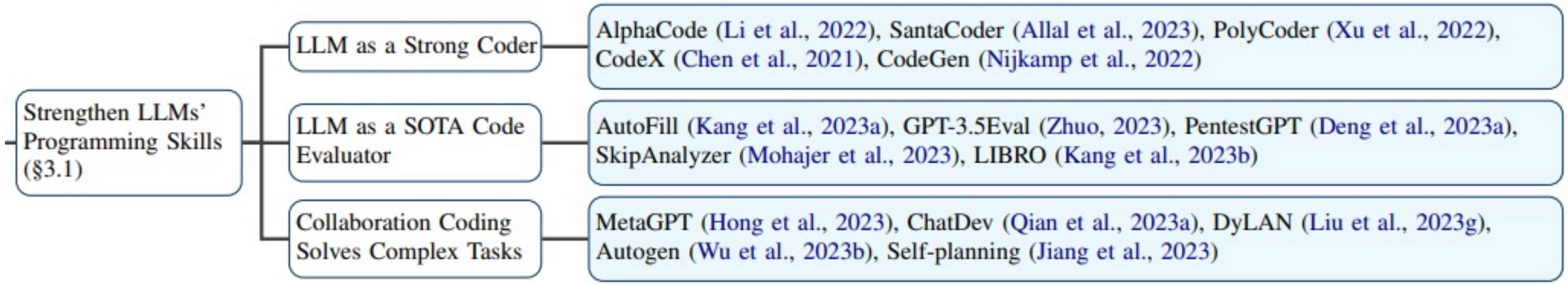
(a) Strengthen LLMs' programming and code evaluation skills (§3.1).

(b) Empower LLMs' complex reasoning, decoupling computation from language understanding (§3.2).

(c) Enable LLM to better capture structured knowledge and better understand complex multimedia data (§3.3).

Figure 3: How code pre-training boosts LLMs' performance.

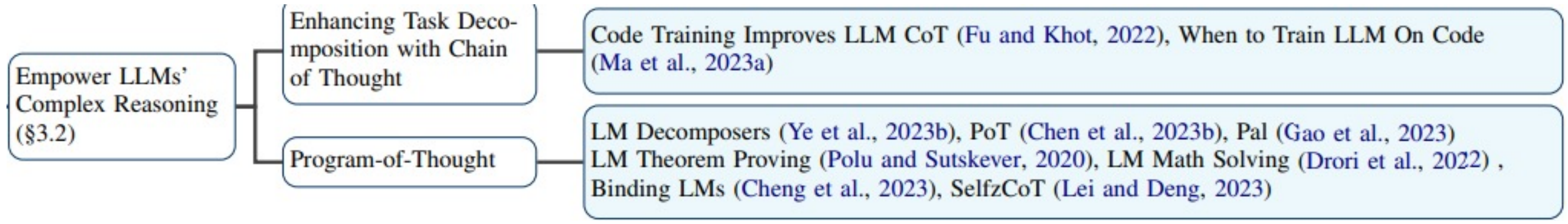
Strengthen LLMs' Programming Skills



Strengthen LLMs' Programming Skills (§3.1)
LLM as a Strong Coder
AlphaCode (Li et al., 2022), SantaCoder (Allal et al., 2023), PolyCoder (Xu et al., 2022), CodeX (Chen et al., 2021), CodeGen (Nijkamp et al., 2022)
LLM as a SOTA Code Evaluator
AutoFill (Kang et al., 2023a), GPT-3.5Eval (Zhuo, 2023), PentestGPT (Deng et al., 2023a), SkipAnalyzer (Mohajer et al., 2023), LIBRO (Kang et al., 2023b)
Collaboration Coding Solves Complex Tasks
MetaGPT (Hong et al., 2023), ChatDev (Qian et al., 2023a), DyLAN (Liu et al., 2023g), Autogen (Wu et al., 2023b), Self-planning (Jiang et al., 2023)

1. Strengthen LLMs' Programming Skills
   - **Coder**
     - PolyCoder master more than 10 languages
     - CodeX with 12 billion parameters that reads the entire GitHub database and is able to solve 72.31% of challenging Python program
   - **Evaluator**
     - Code fault localization
     - GPT-3.5 to evaluate the functional correctness and human preferences
   - **Collaborative Coding:**
     - Assigning three roles: analyst, coder, and tester to three distinct "GPT-3.5"s, which surpasses GPT-4 in code generation

Strengthen LLMs’ Programming Skills


Empower LLMs’ Complex Reasoning (§3.2)
Enhancing Task Decomposition with Chain of Thought
Program-of-Thought
Code Training Improves LLM CoT (Fu and Khot, 2022), When to Train LLM On Code (Ma et al., 2023a)
LM Decomposers (Ye et al., 2023b), PoT (Chen et al., 2023b), Pal (Gao et al., 2023)
LM Theorem Proving (Polu and Sutskever, 2020), LM Math Solving (Drori et al., 2022) ,
Binding LMs (Cheng et al., 2023), SelfzCoT (Lei and Deng, 2023)

2. Empower LLMs’ Complex Reasoning (Chain-of-thought, Program-of-thought)

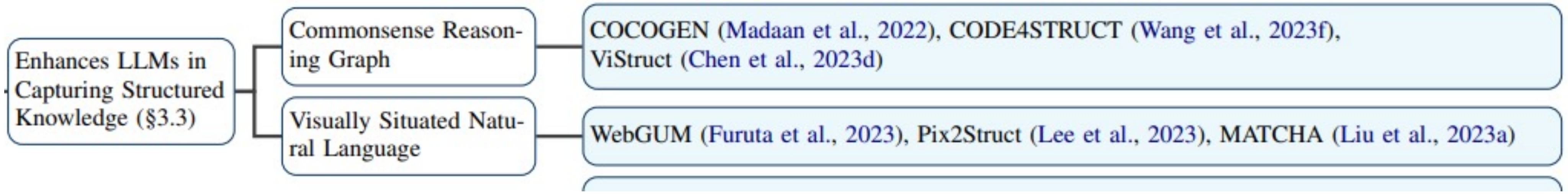
a) **Chain of Thought**

- LLMs pre-trained on code, such as GPT-3’s text-davinci-002 and Codex (Chen et al., 2021), see a dramatic performance improvement arising from CoT, with a remarkable accuracy increase of 15.6% to 46.9% and 19.7% to 63.1% respectively

b) **Program of Thought:**

- Enhances performance due to the precision and verifiability inherent in code
- Executing code and verifying outcomes post translation by LLMs, one can effectively mitigate the effects of incorrect reasoning in CoT

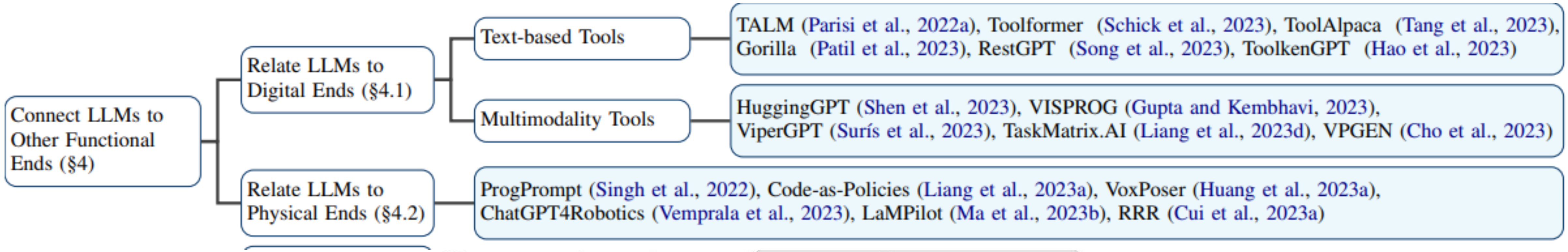
Enable LLMs to Capture Structured Knowledge



Enhances LLMs in Capturing Structured Knowledge (§3.3)
Commonsense Reasoning Graph
COCOGEN (Madaan et al., 2022), CODE4STRUCT (Wang et al., 2023f), ViStruct (Chen et al., 2023d)
Visually Situated Natural Language
WebGUM (Furuta et al., 2023), Pix2Struct (Lee et al., 2023), MATCHA (Liu et al., 2023a)

3. **Enable LLMs to Capture Structured Knowledge**
   - **Commonsense reasoning:**
     - Code possesses the **graph structure of symbolic representations**
     - **Leveraging** programming language for representing **visual structural information** and curriculum learning for enhancing the model's understanding of visual structures
   - **Markup code:**
     - Utilizing **markup code such as HTML and CSS** to for structured graphical information in graphical user interfaces
     - WebGUM showcased the **effectiveness of pre-training model with markup code**

Connecting LLMs to other Functional Ends



Connect LLMs to Other Functional Ends (§4)

- Relate LLMs to Digital Ends (§4.1)
  - Text-based Tools
    - TALM (Parisi et al., 2022a), Toolformer (Schick et al., 2023), ToolAlpaca (Tang et al., 2023), Gorilla (Patil et al., 2023), RestGPT (Song et al., 2023), ToolkenGPT (Hao et al., 2023)
  - Multimodality Tools
    - HuggingGPT (Shen et al., 2023), VISPROG (Gupta and Kembhavi, 2023), ViperGPT (Surís et al., 2023), TaskMatrix.AI (Liang et al., 2023d), VPGEN (Cho et al., 2023)
- Relate LLMs to Physical Ends (§4.2)
  - ProgPrompt (Singh et al., 2022), Code-as-Policies (Liang et al., 2023a), VoxPoser (Huang et al., 2023a), ChatGPT4Robotics (Vemprala et al., 2023), LaMPilot (Ma et al., 2023b), RRR (Cui et al., 2023a)

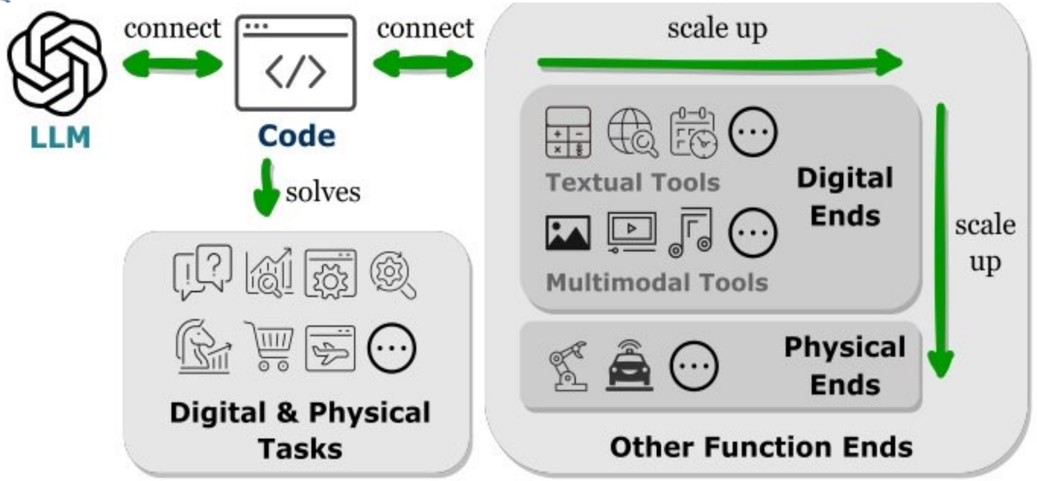
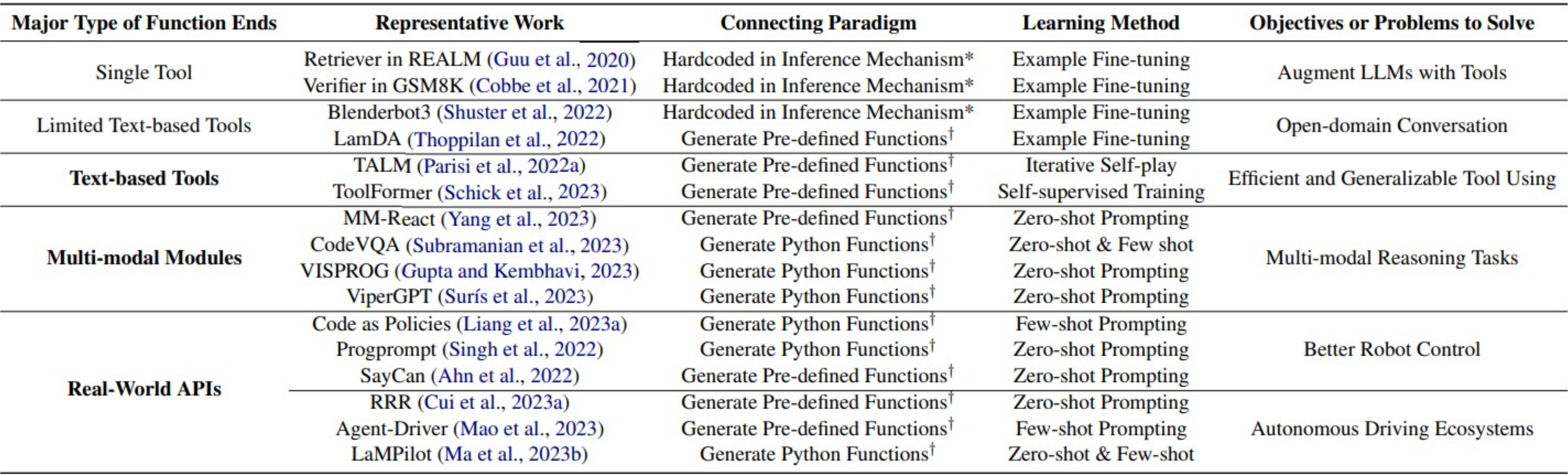


Figure 4: The code-centric tool-calling paradigm serves as a unified interface between LLMs and a large variety of functional ends, thus enabling many cross-modality and cross-domain tasks.

Connecting LLMs to other Functional Ends



Major Type of Function Ends	Representative Work	Connecting Paradigm	Learning Method	Objectives or Problems to Solve
Single Tool	Retriever in REALM (Guu et al., 2020)	Hardcoded in Inference Mechanism*	Example Fine-tuning	Augment LLMs with Tools
	Verifier in GSM8K (Cobbe et al., 2021)	Hardcoded in Inference Mechanism*	Example Fine-tuning	
Limited Text-based Tools	Blenderbot3 (Shuster et al., 2022)	Hardcoded in Inference Mechanism*	Example Fine-tuning	Open-domain Conversation
	LamDA (Thoppilan et al., 2022)	Generate Pre-defined Functions†	Example Fine-tuning	
Text-based Tools	TALM (Parisi et al., 2022a)	Generate Pre-defined Functions†	Iterative Self-play	Efficient and Generalizable Tool Using
	ToolFormer (Schick et al., 2023)	Generate Pre-defined Functions†	Self-supervised Training	
Multi-modal Modules	MM-React (Yang et al., 2023)	Generate Pre-defined Functions†	Zero-shot Prompting	Multi-modal Reasoning Tasks
	CodeVQA (Subramanian et al., 2023)	Generate Python Functions†	Zero-shot & Few shot	
	VISPROG (Gupta and Kembhavi, 2023)	Generate Python Functions†	Zero-shot Prompting	
	ViperGPT (Surís et al., 2023)	Generate Python Functions†	Zero-shot Prompting	
Real-World APIs	Code as Policies (Liang et al., 2023a)	Generate Python Functions†	Few-shot Prompting	Better Robot Control
	Progprompt (Singh et al., 2022)	Generate Python Functions†	Zero-shot Prompting	
	SayCan (Ahn et al., 2022)	Generate Pre-defined Functions†	Zero-shot Prompting	
	RRR (Cui et al., 2023a)	Generate Pre-defined Functions†	Zero-shot Prompting	Autonomous Driving Ecosystems
	Agent-Driver (Mao et al., 2023)	Generate Pre-defined Functions†	Few-shot Prompting	
	LaMPilot (Ma et al., 2023b)	Generate Python Functions†	Zero-shot & Few-shot	

UNIVERSITY of VIRGINIA

Embedding LLMs into Code Execution Environment

- LLMs demonstrate performance beyond the parameters of their training due to their ability to intake feedback
- Embedding LLMs into a code execution environment enables automated feedback

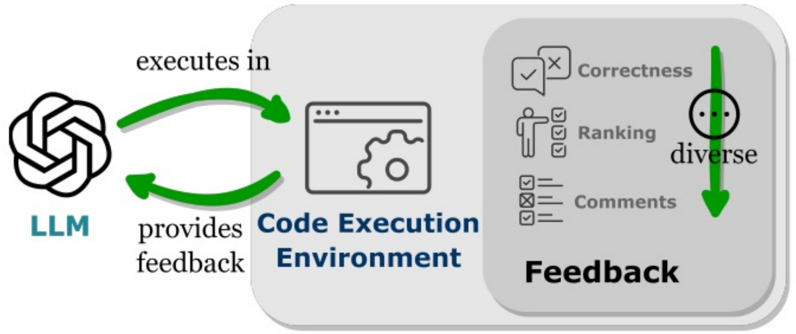


Figure 5: LLMs can be embedded into a code execution environment, where they collect faithful, automatic, and customizable feedback for self-improvement.

Automated Feedback

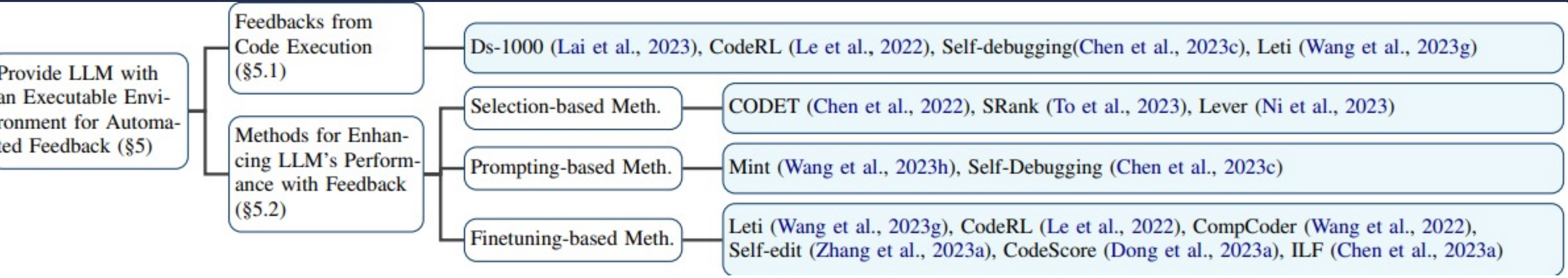
- Program execution outcomes and generating feedback include the
  - Creation of **unit tests**
  - Application of **exact result matching techniques**

- From these, feedback can be provided in two primary forms:
  - **Simple correctness feedback** and (whether a program is **correct or not**)
  - **Textual feedback** (**explanations** about the program or its **summarization**)

- Execution results can also be translated into **reward functions using predefined rules.** The rules map **execution results** into **scalar values** based on the severity of **different error types** suitable for reinforcement learning approaches.

- **Additional feedback** can be extracted by performing **static analysis** using **software engineering tools**

Enhancing LLM's Performance with Feedback

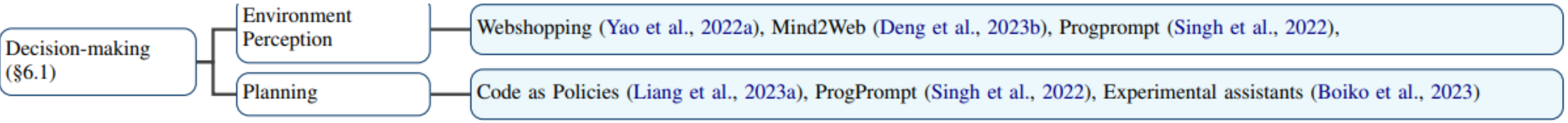

Provide LLM with an Executable Envi-ronment for Automa-ted Feedback (§5)
Feedbacks from Code Execution (§5.1)
Ds-1000 (Lai et al., 2023), CodeRL (Le et al., 2022), Self-debugging(Chen et al., 2023c), Leti (Wang et al., 2023g)
Methods for Enhan-cing LLM's Perform-ance with Feedback (§5.2)
Selection-based Meth.
CODET (Chen et al., 2022), SRank (To et al., 2023), Lever (Ni et al., 2023)
Prompting-based Meth.
Mint (Wang et al., 2023h), Self-Debugging (Chen et al., 2023c)
Finetuning-based Meth.
Leti (Wang et al., 2023g), CodeRL (Le et al., 2022), CompCoder (Wang et al., 2022), Self-edit (Zhang et al., 2023a), CodeScore (Dong et al., 2023a), ILF (Chen et al., 2023a)

- The **feedback** derived from **code execution** and **external evaluation** modules can **enhance LLMs through three major approaches**:
  - **Selection Based Method** (majority voting and re-ranking)
  - **Prompting Based Methods** and ("self-debugging" with in-context learning)
  - Finetuning Methods (improve the LLMs by updating their parameterized knowledge)
    - **Direct Finetuning from feedback**
    - **Generating Synthetic unit tests** to identify and **retain only correctly generated examples**, which are then composed into correct question-answer pairs
    - **RL with fixed reward values** for different execution result types based on unit tests

Applications

Improvements brought about by code training in LLMs are firmly rooted in their practical operational steps

These steps include



Webshopping (Yao et al., 2022a), Mind2Web (Deng et al., 2023b), Progprompt (Singh et al., 2022),

Code as Policies (Liang et al., 2023a), ProgPrompt (Singh et al., 2022), Experimental assistants (Boiko et al., 2023)

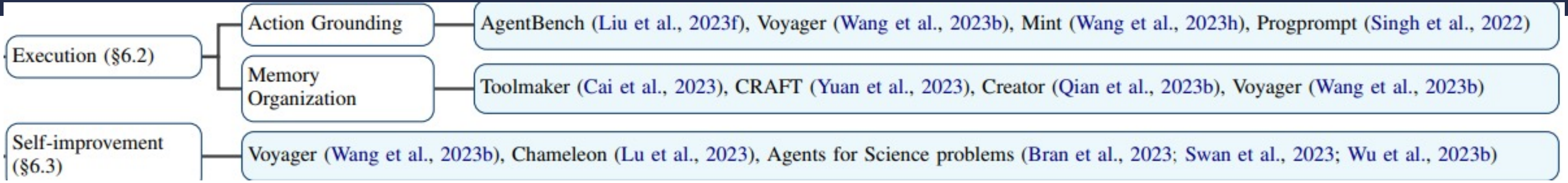
1. Enhancing the IA's decision-making in terms of
   - **Environment perception:**

     The **perceived information** needs to be **organized** in a **highly structured format**, ensuring that stimuli occurring at the same moment (e.g., coexisting multimodality stimuli) **influence the IA's perception** and decision.

   - **Planning:**

     Leveraging the synergized planning abilities of code-LLMs, IAs can generate **organized reasoning steps** using **modular and unambiguous code** alongside **expressive natural language.**

Applications



Execution (§6.2)

- Action Grounding — AgentBench (Liu et al., 2023f), Voyager (Wang et al., 2023b), Mint (Wang et al., 2023h), Progprompt (Singh et al., 2022)
- Memory Organization — Toolmaker (Cai et al., 2023), CRAFT (Yuan et al., 2023), Creator (Qian et al., 2023b), Voyager (Wang et al., 2023b)

Self-improvement (§6.3) — Voyager (Wang et al., 2023b), Chameleon (Lu et al., 2023), Agents for Science problems (Bran et al., 2023; Swan et al., 2023; Wu et al., 2023b)

2. Streamlining execution by
   - Actions grounding :

     IA **interfaces with external function ends** according to the planning, it must invoke action **primitives from a pre-defined set of actions**

   - Memory Organization :

     IA typically necessitates an memory organization module to manage exposed information, including original **planning**, **task progress**, **execution history**, available **tool set**, acquired **skills**, **augmented knowledge**, and **users' early feedback**

3. Optimizing performance through **feedback automatically derived from the code execution environment**

Challenges

1. The **Causality** between **Code Pre-training and LLMs' Reasoning Enhancement**
   - Gap persists in providing **explicit experimental evidence** that directly indicates the enhancement of LLMs' reasoning abilities through the acquisition of specific code properties
2. Acquisition of **Reasoning Beyond Code**:
   - Still lack the **human-like reasoning abilities**
3. Challenges of **Applying Code-centric Paradigm**:
   - Connect to different function ends is learning the **correct invocation of numerous functions**, including **selecting the right function** end and passing the **correct parameters at an appropriate time**

Thank You

Soneya Binta Hossain (sh7hv)
Jessie Chen (hc4vb)
Ali Zafar Sadiq (mzw2cu)
Jeffrey Chen (fyy2ws)
Minjae Kwon (hbt9su)

REFERENCES

https://arxiv.org/abs/2302.07842
https://arxiv.org/abs/2205.10625
https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html
https://proceedings.neurips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html
https://proceedings.mlr.press/v202/gao23f
https://arxiv.org/abs/2210.03350
https://arxiv.org/abs/2112.00114
https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html
https://proceedings.neurips.cc/paper_files/paper/2022/hash/960a172bc7fbf0177ccccbb411a7d800-Abstract-Conference.html

https://arxiv.org/abs/2203.11171
https://arxiv.org/abs/2401.00812