Self Evolving LLMs

Large Language Models: Introduction and Recent Advances

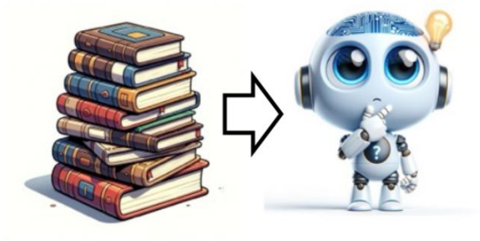
ELL881 · AIL821

भारतीय प्रौद्योगिकी संस्थान दिल्ली
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Yatin Nandwani
IBM Research

Evolution of LLMs

Evolution of LLMs

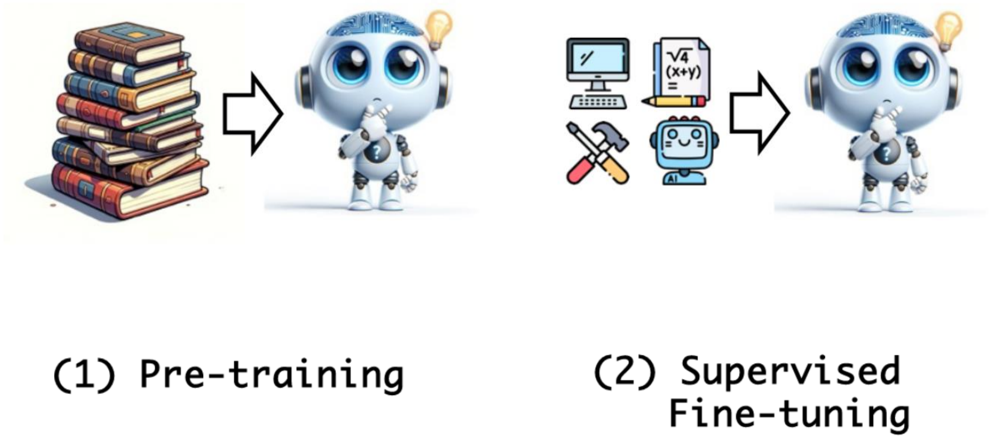


`(1) Pre-training`

2018-19
BERT, GPT2

Image credit: [A Survey on Self-Evolution of Large Language Models](#)

Evolution of LLMs

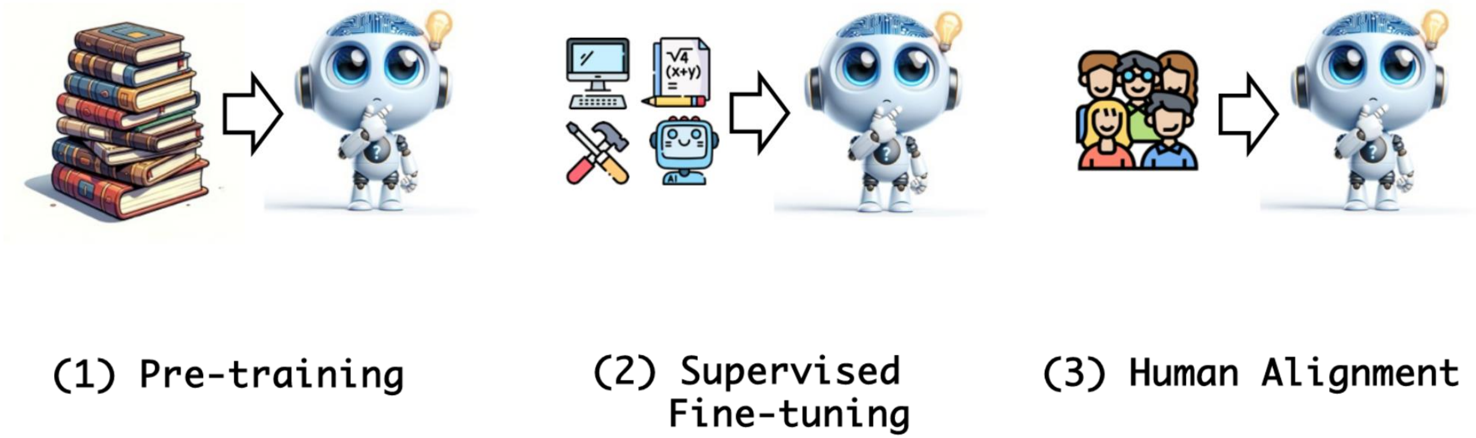


2018-19
BERT, GPT2

2020
T5

Image credit: [A Survey on Self-Evolution of Large Language Models](#)

Evolution of LLMs



2018-19
BERT, GPT2

2020
T5

2022
InstructGPT

Image credit: [A Survey on Self-Evolution of Large Language Models](#)

AlphaGo

Alpha Go

1. **Imitation learning :** Predict next move using human-expert games

Alpha Go

1. **Imitation learning :** Predict next move using human-expert game

~ Pre-training ; SFT ; RLHF

Alpha Go

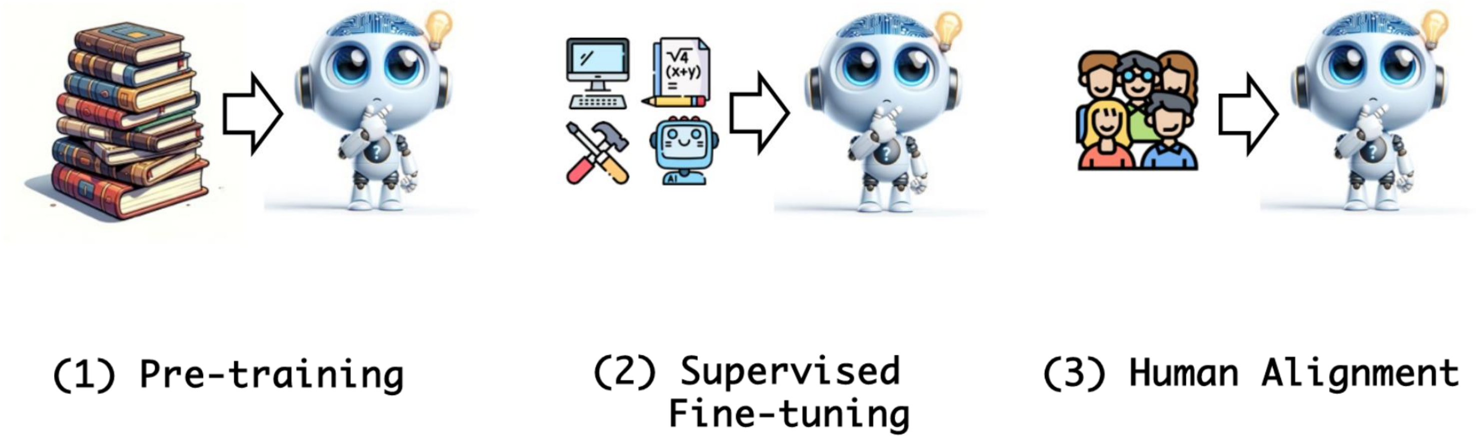
1. **Imitation learning :** Predict next move using human-expert game
2. **Learn via Self Play:**
    1. Generate new game trajectories by playing against itself
    2. Identify good moves
    3. Use reward (win / loss) to update the weights via RL

Alpha Go

1. **Imitation learning :** Predict next move using human-expert game
2. **Learn via Self Play:**
   1. Generate new game trajectories by playing against itself
   2. Identify good moves
   3. Use reward (win / loss) to update the weights via RL

Self - evolution

Evolution of LLMs



2018-19
BERT, GPT2

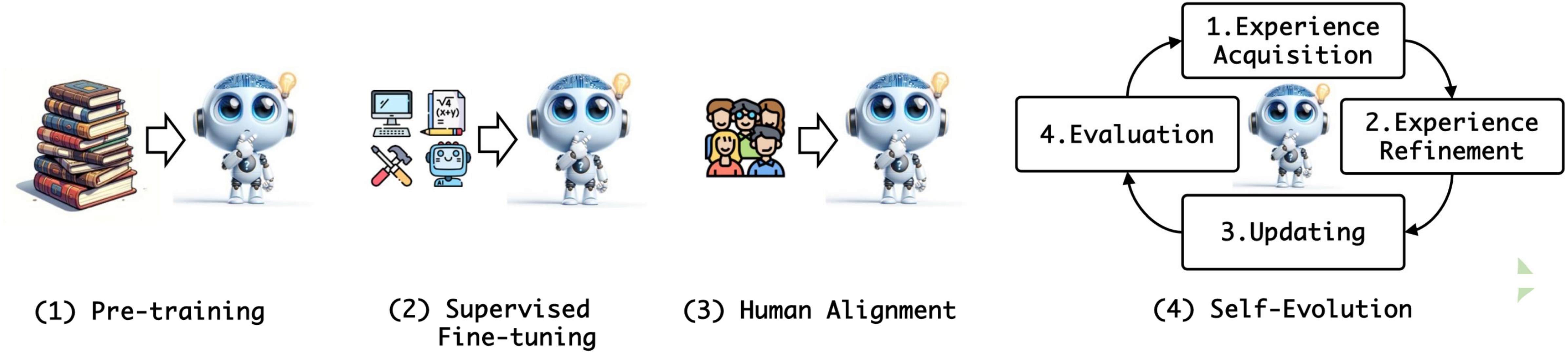
2019 – T5
Flan T5

2022
InstructGPT

~ Imitation Learning

Image credit: [A Survey on Self-Evolution of Large Language Models](#)

Evolution of LLMs



2018-19
BERT, GPT2

2019 – T5
Flan T5

2022
InstructGPT

2023
STAR

~ Imitation Learning

~ Self Play

Image credit: [A Survey on Self-Evolution of Large Language Models](#)

Self Evolution in LLMs - challenges

1. **Experience acquisition** ~ Generate new game trajectories by playing against itself
    - *How to construct synthetic data for LLMs training?*

2. **Experience refinement** ~ Identify good moves
    - *How to identify good quality synthetic data?*

3. **Updating** ~ Use reward (win / loss) to update the weights via RL
    - *No accurate and unambiguous environment feedback*
    - *Use RL or supervised finetuning?*

4. **Evaluation** – need annotated test data

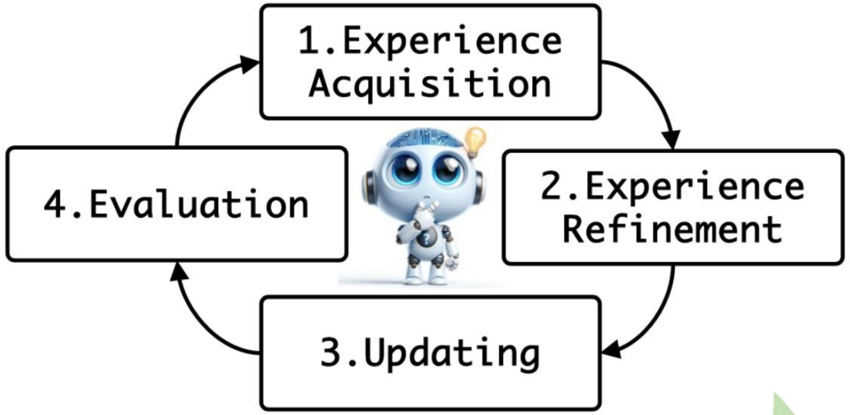
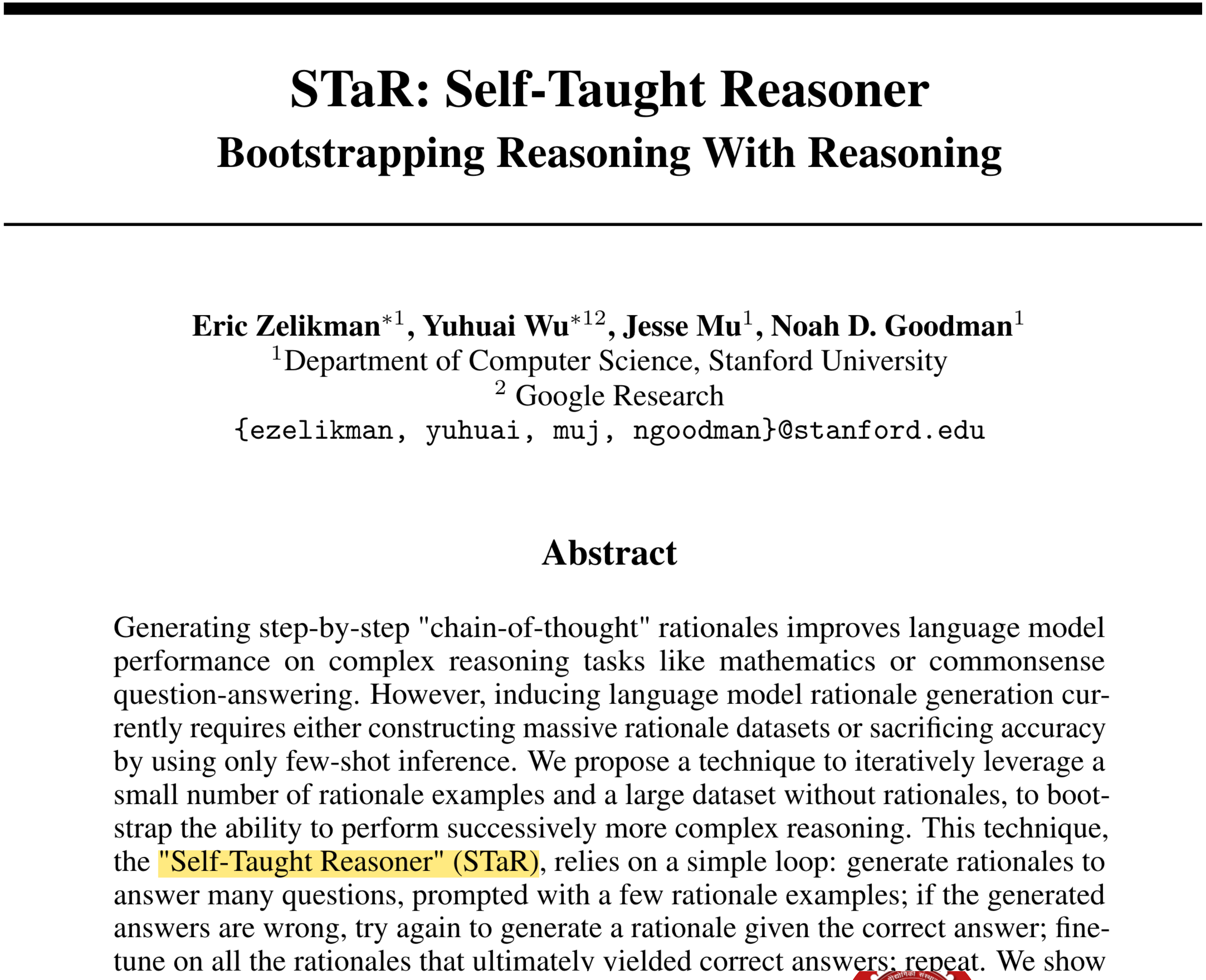


Image credit: A Survey on Self-Evolution of Large Language Models

Overview

- Methods that use self-play to generate synthetic data and update its weights:
  - STaR
  - Self Instruct
  - Self Align



STaR: Self-Taught Reasoner

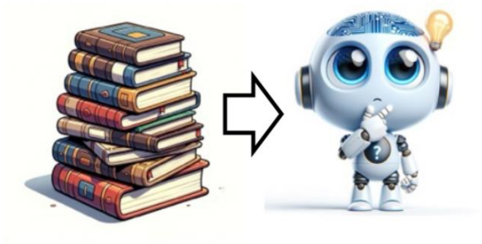
Bootstrapping Reasoning With Reasoning

Eric Zelikman*[1], **Yuhuai Wu***[1,2], **Jesse Mu**[1], **Noah D. Goodman**[1]
[1]Department of Computer Science, Stanford University
[2] Google Research
`{ezelikman, yuhuai, muj, ngoodman}@stanford.edu`

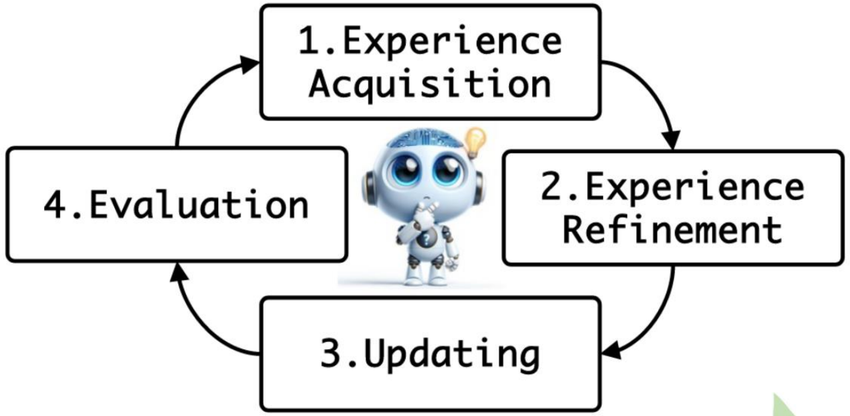
Abstract

Generating step-by-step "chain-of-thought" rationales improves language model performance on complex reasoning tasks like mathematics or commonsense question-answering. However, inducing language model rationale generation currently requires either constructing massive rationale datasets or sacrificing accuracy by using only few-shot inference. We propose a technique to iteratively leverage a small number of rationale examples and a large dataset without rationales, to bootstrap the ability to perform successively more complex reasoning. This technique, the "Self-Taught Reasoner" (STaR), relies on a simple loop: generate rationales to answer many questions, prompted with a few rationale examples; if the generated answers are wrong, try again to generate a rationale given the correct answer; fine-tune on all the rationales that ultimately yielded correct answers; repeat. We show

STaR: Self-Taught Reasoner



(1) Pre-training



Base-GPT-J

Self-Taught GPT-J

~ Imitation Learning

~ Self Play

STaR: Self-Taught Reasoner*

Given:

1. a pretrained LLM (M)
2. a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$
3. few shot examples $\{(x_i^p, r_i^p, y_i^p)\}_{i=1}^{P}$

* STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning, Zelikman et. al., May 2022

STaR: Self-Taught Reasoner*

Given:

1. a pretrained LLM (M)
2. a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$
3. few shot examples $\{(x_i^p, r_i^p, y_i^p)\}_{i=1}^{P}$

For each example in $\mathcal{D}$, use the few shot examples along with the input x_i to predict $\hat{r_i}$ and $\hat{y}_i$

* STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning, Zelikman et. al., May 2022

STaR: Self-Taught Reasoner*

Given:

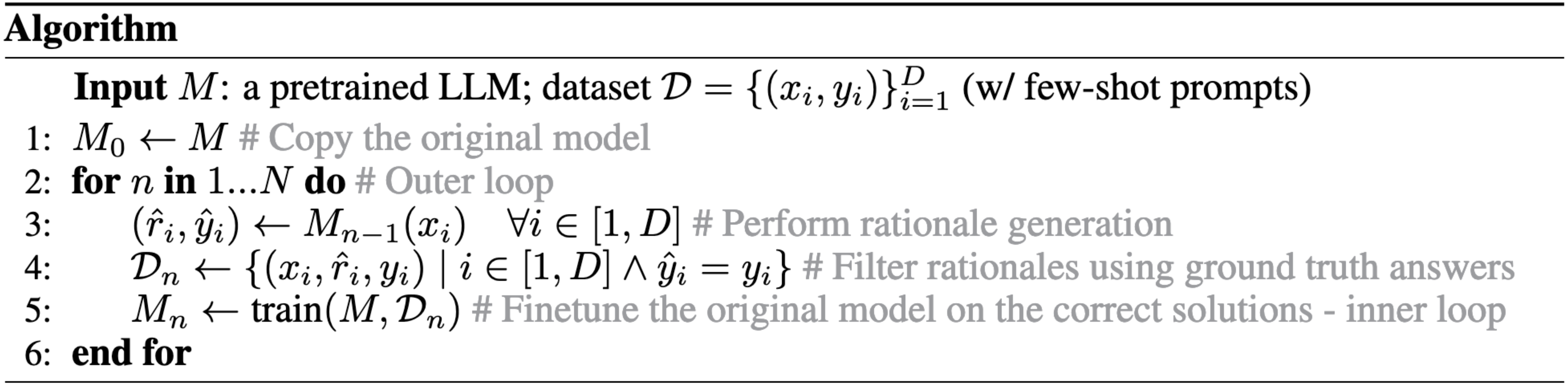
1. a pretrained LLM (M)
2. a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$
3. few shot examples $\left\{(x_i^p, r_i^p, y_i^p)\right\}_{i=1}^{P}$

For each example in $\mathcal{D}$, use the few shot examples along with the input x_i to predict $\widehat{r_i}$ and $\widehat{y_i}$

- If $(\widehat{y_i} = y_i)$, then use the example along with rationale $\widehat{r_i}$ to finetune
- If $(\widehat{y_i} \neq y_i)$, then discard

* STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning, Zelikman et. al., May 2022

STaR: Self-Taught Reasoner*



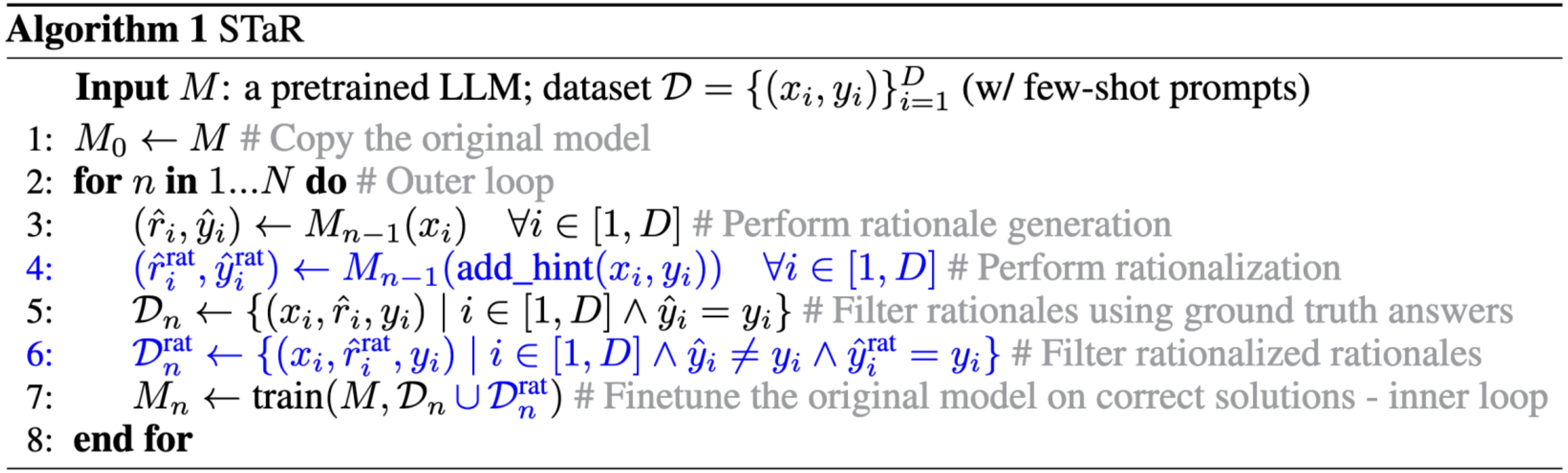
Algorithm

```
Input $M$: a pretrained LLM; dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$ (w/ few-shot prompts)
$M_0 \leftarrow M$ # Copy the original model
for $n$ in $1 \ldots N$ do # Outer loop
    $(\hat{r}_i, \hat{y}_i) \leftarrow M_{n-1}(x_i) \quad \forall i \in [1, D]$ # Perform rationale generation
    $\mathcal{D}_n \leftarrow \{(x_i, \hat{r}_i, y_i) \mid i \in [1, D] \wedge \hat{y}_i = y_i\}$ # Filter rationales using ground truth answers
    $M_n \leftarrow \text{train}(M, \mathcal{D}_n)$ # Finetune the original model on the correct solutions - inner loop
end for
```

* STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning, Zelikman et. al., May 2022

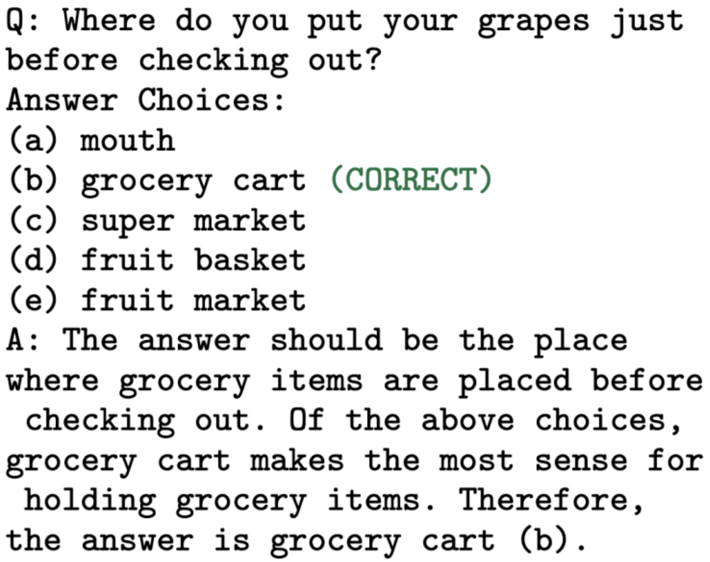
STaR: Self-Taught Reasoner*

Rationalization



Algorithm 1 STaR

```
Input $M$: a pretrained LLM; dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$ (w/ few-shot prompts)
$M_0 \leftarrow M$ # Copy the original model
for $n$ in $1...N$ do # Outer loop
    $(\hat{r}_i, \hat{y}_i) \leftarrow M_{n-1}(x_i) \quad \forall i \in [1, D]$ # Perform rationale generation
    $(\hat{r}_i^{\text{rat}}, \hat{y}_i^{\text{rat}}) \leftarrow M_{n-1}(\text{add\_hint}(x_i, y_i)) \quad \forall i \in [1, D]$ # Perform rationalization
    $\mathcal{D}_n \leftarrow \{(x_i, \hat{r}_i, y_i) \mid i \in [1, D] \wedge \hat{y}_i = y_i\}$ # Filter rationales using ground truth answers
    $\mathcal{D}_n^{\text{rat}} \leftarrow \{(x_i, \hat{r}_i^{\text{rat}}, y_i) \mid i \in [1, D] \wedge \hat{y}_i \neq y_i \wedge \hat{y}_i^{\text{rat}} = y_i\}$ # Filter rationalized rationales
    $M_n \leftarrow \text{train}(M, \mathcal{D}_n \cup \mathcal{D}_n^{\text{rat}})$ # Finetune the original model on correct solutions - inner loop
end for
```



Q: Where do you put your grapes just before checking out?
Answer Choices:
(a) mouth
(b) grocery cart (CORRECT)
(c) super market
(d) fruit basket
(e) fruit market
A: The answer should be the place where grocery items are placed before checking out. Of the above choices, grocery cart makes the most sense for holding grocery items. Therefore, the answer is grocery cart (b).

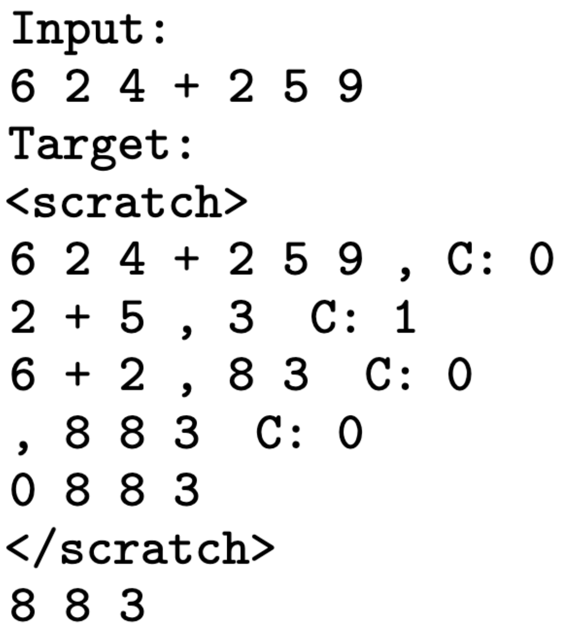
---

* STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning, Zelikman et. al., May 2022

STaR: Self-Taught Reasoner*

Arithmetic Dataset

- The arithmetic task is to calculate the sum of two n-digit integers
- It includes few-shot prompts for 1 to 5 digits



```
Input:
6 2 4 + 2 5 9
Target:
<scratch>
6 2 4 + 2 5 9 , C: 0
2 + 5 , 3  C: 1
6 + 2 , 8 3  C: 0
, 8 8 3  C: 0
0 8 8 3
</scratch>
8 8 3
```

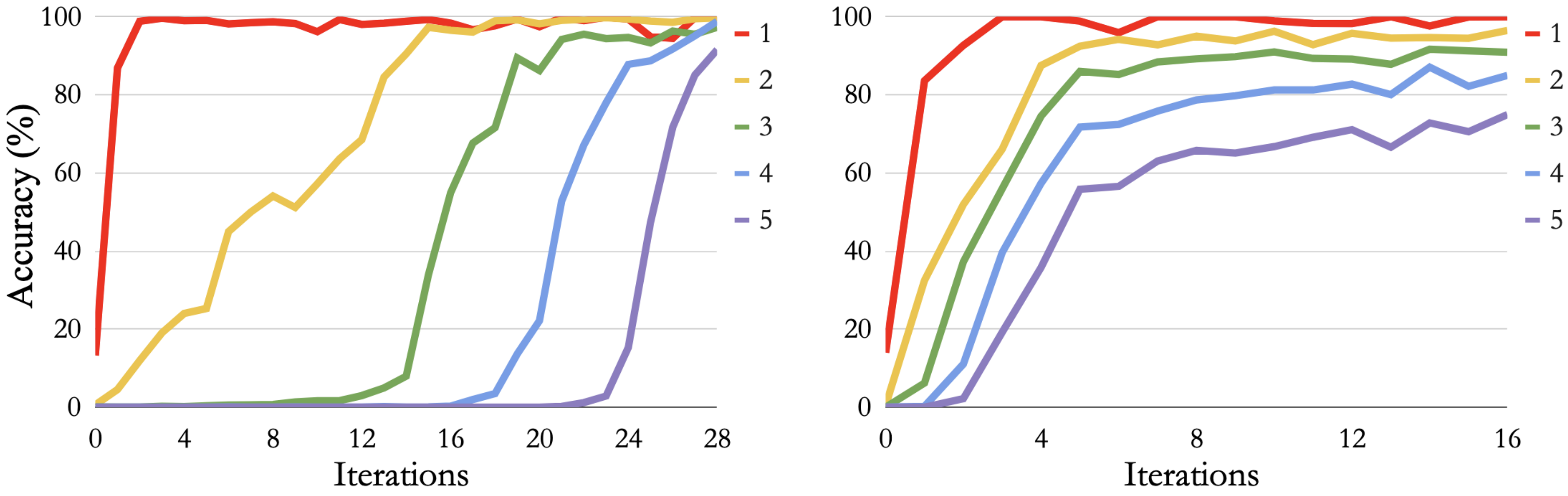
---

* STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning, Zelikman et. al., May 2022

STaR: Self-Taught Reasoner*

Arithmetic Dataset

- The arithmetic task is to calculate the sum of two n-digit integers
- It includes few-shot prompts for 1 to 5 digits

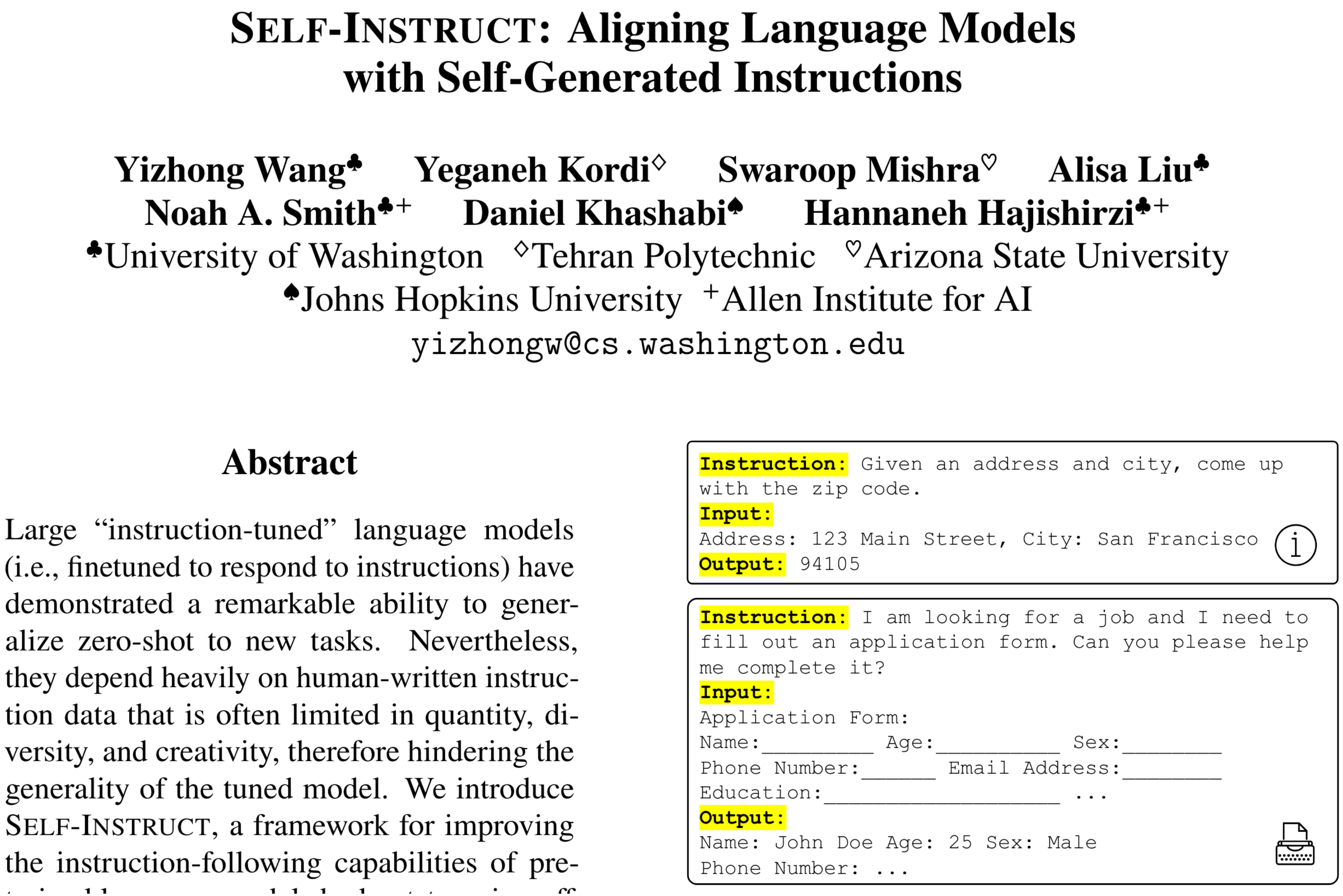


(a) Without rationalization (b) With rationalization

* STaR: Self-Taught Reasoner Bootstrapping Reasoning With Reasoning, Zelikman et. al., May 2022

STaR: Self-Taught Reasoner*

- Drawback – still relies on annotated dataset to evolve



SELF-INSTRUCT: Aligning Language Models with Self-Generated Instructions

Yizhong Wang♣ Yeganeh Kordi♢ Swaroop Mishra♡ Alisa Liu♣
Noah A. Smith♣+ Daniel Khashabi♠ Hannaneh Hajishirzi♣+
♣University of Washington ♢Tehran Polytechnic ♡Arizona State University
♠Johns Hopkins University +Allen Institute for AI
yizhongw@cs.washington.edu

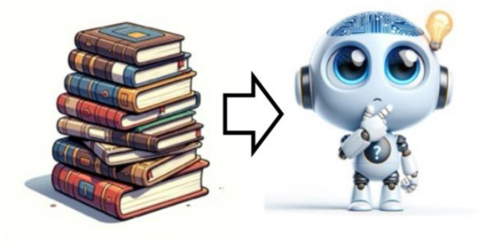
Abstract

Large "instruction-tuned" language models (i.e., finetuned to respond to instructions) have demonstrated a remarkable ability to generalize zero-shot to new tasks. Nevertheless, they depend heavily on human-written instruction data that is often limited in quantity, diversity, and creativity, therefore hindering the generality of the tuned model. We introduce SELF-INSTRUCT, a framework for improving the instruction-following capabilities of pre-

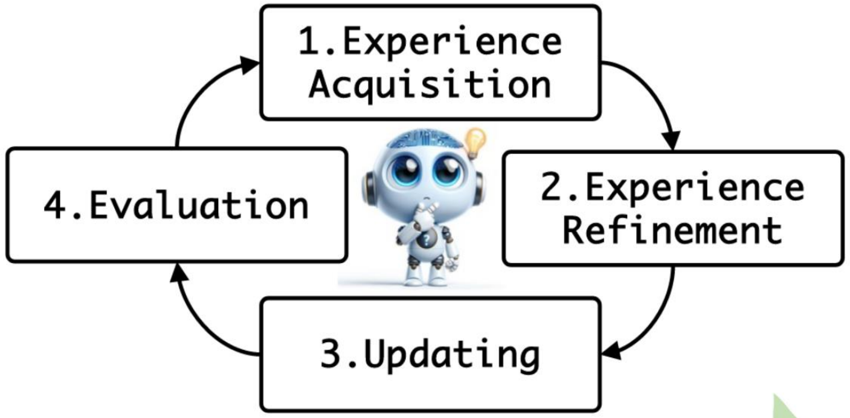
Instruction: Given an address and city, come up with the zip code.
Input:
Address: 123 Main Street, City: San Francisco
Output: 94105

Instruction: I am looking for a job and I need to fill out an application form. Can you please help me complete it?
Input:
Application Form:
Name:_________ Age:__________ Sex:________
Phone Number:______ Email Address:________
Education:___________________ ...
Output:
Name: John Doe Age: 25 Sex: Male
Phone Number: ...

Self Instruct



(1) Pre-training



Base-GPT3

Self-Instruct GPT3

~ Imitation Learning

~ Self Play

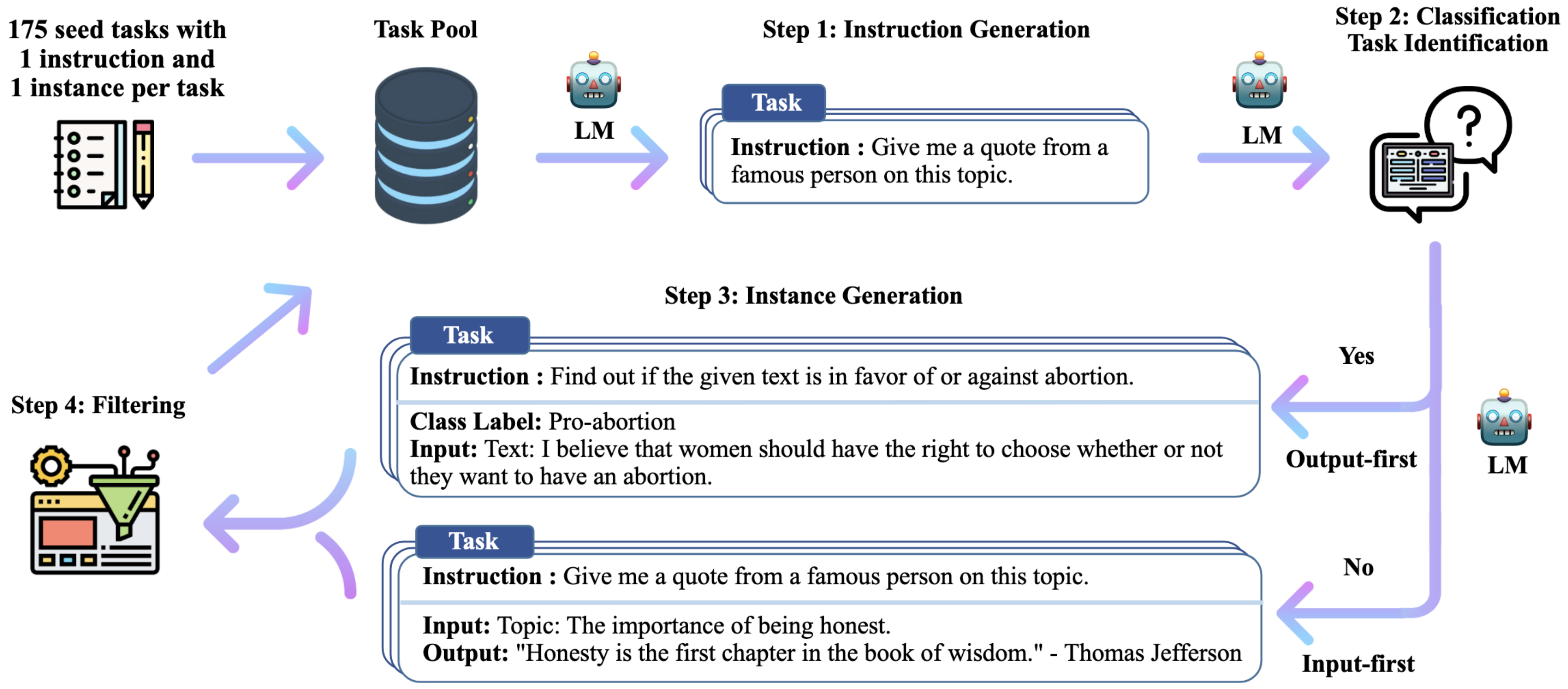
Self-Instruct*

- Given:
  - a base model
  - 175 seed tasks. Each task has an instruction and an example

- Objective:
  - Use the base model to synthesize instruction finetuning dataset
    - Base model is a good at in-context learning
    - Use seed instructions to create more instructions
    - Use seed instruction/example pairs to create examples for new instructions
  - Use the synthesized data to finetune the base model to an instruct tuned model

* Self-Instruct: Aligning Language Models with Self-Generated Instructions, Wang et. al., May 2023

Self-Instruct*



175 seed tasks with
1 instruction and
1 instance per task

Task Pool

LM

Step 1: Instruction Generation

Task

Instruction : Give me a quote from a famous person on this topic.

LM

Step 2: Classification Task Identification

LM

Yes

Output-first

No

Input-first

Step 3: Instance Generation

Task

Instruction : Find out if the given text is in favor of or against abortion.

Class Label: Pro-abortion
Input: Text: I believe that women should have the right to choose whether or not they want to have an abortion.

Task

Instruction : Give me a quote from a famous person on this topic.

Input: Topic: The importance of being honest.
Output: "Honesty is the first chapter in the book of wisdom." - Thomas Jefferson

Step 4: Filtering

* Self-Instruct: Aligning Language Models with Self-Generated Instructions, Wang et. al., May 2023

Self-Instruct*

Filters:

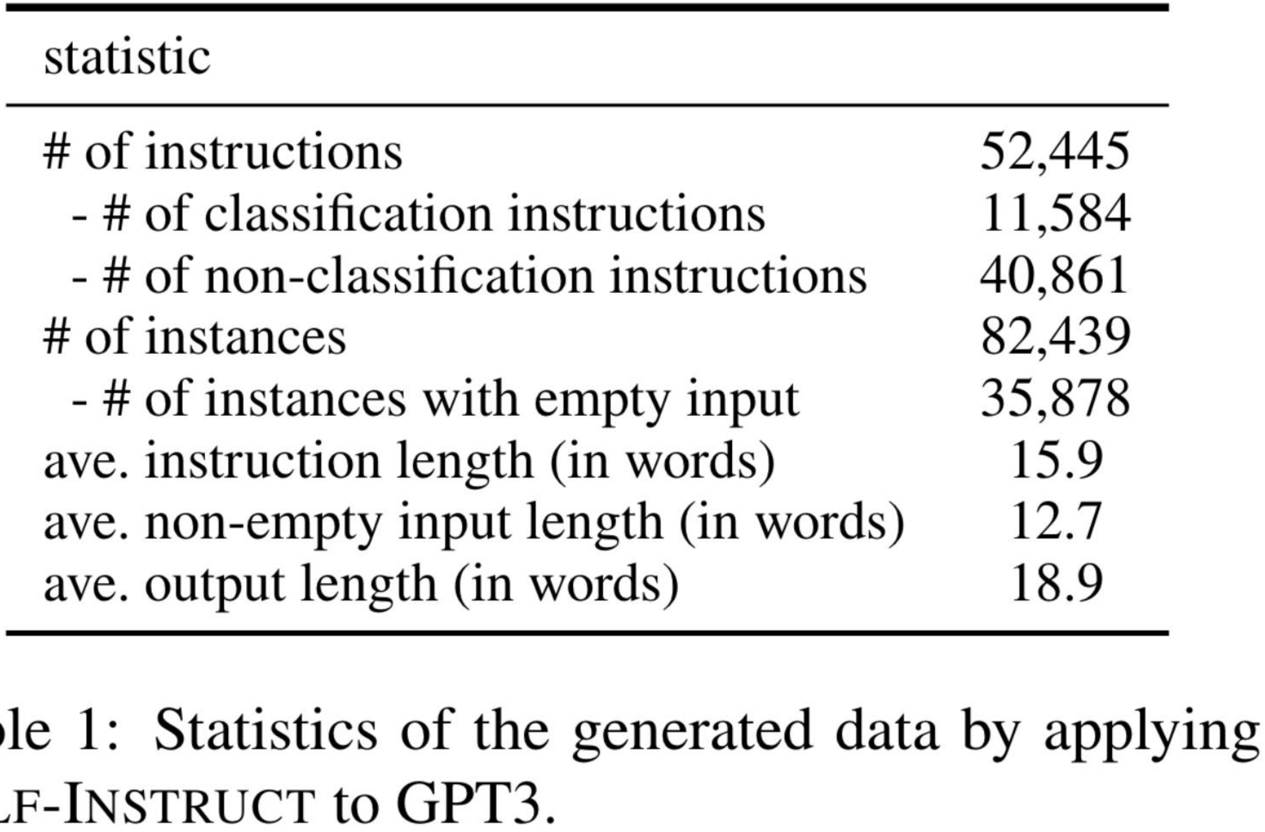
1. Synthesized instruction a new instruction is added to the task pool only when its ROUGE-L similarity with any existing instruction is less than 0.7

2. Exclude instructions that contain some specific keywords (e.g., image, picture, graph)

3. Invalid generations are identified and filtered out based on heuristics (e.g., instruction is too long or too short, instance output is a repetition of the input)

* Self-Instruct: Aligning Language Models with Self-Generated Instructions, Wang et. al., May 2023

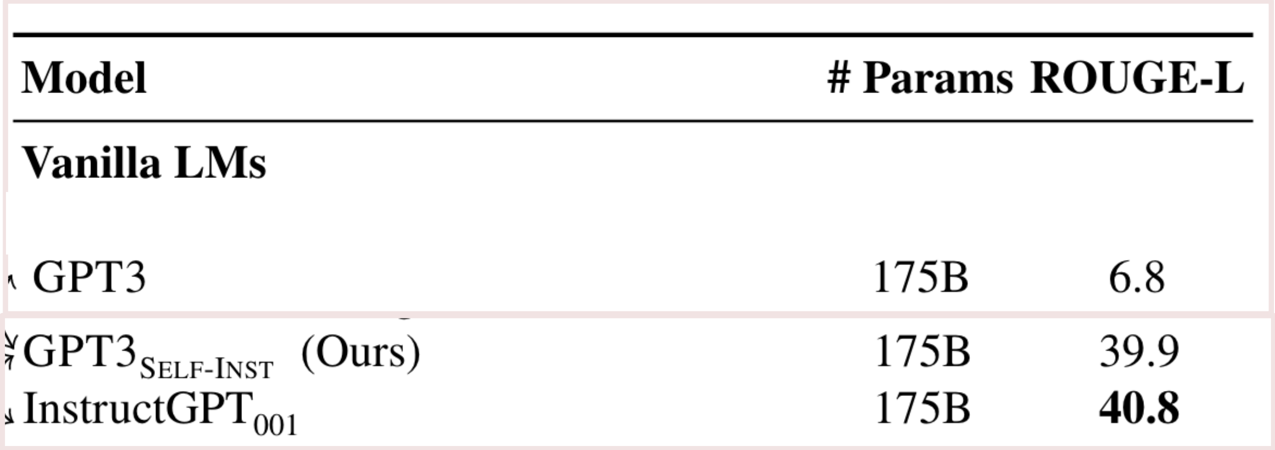
Self Instruct – data stats



statistic	
# of instructions	52,445
- # of classification instructions	11,584
- # of non-classification instructions	40,861
# of instances	82,439
- # of instances with empty input	35,878
ave. instruction length (in words)	15.9
ave. non-empty input length (in words)	12.7
ave. output length (in words)	18.9

Table 1: Statistics of the generated data by applying SELF-INSTRUCT to GPT3.

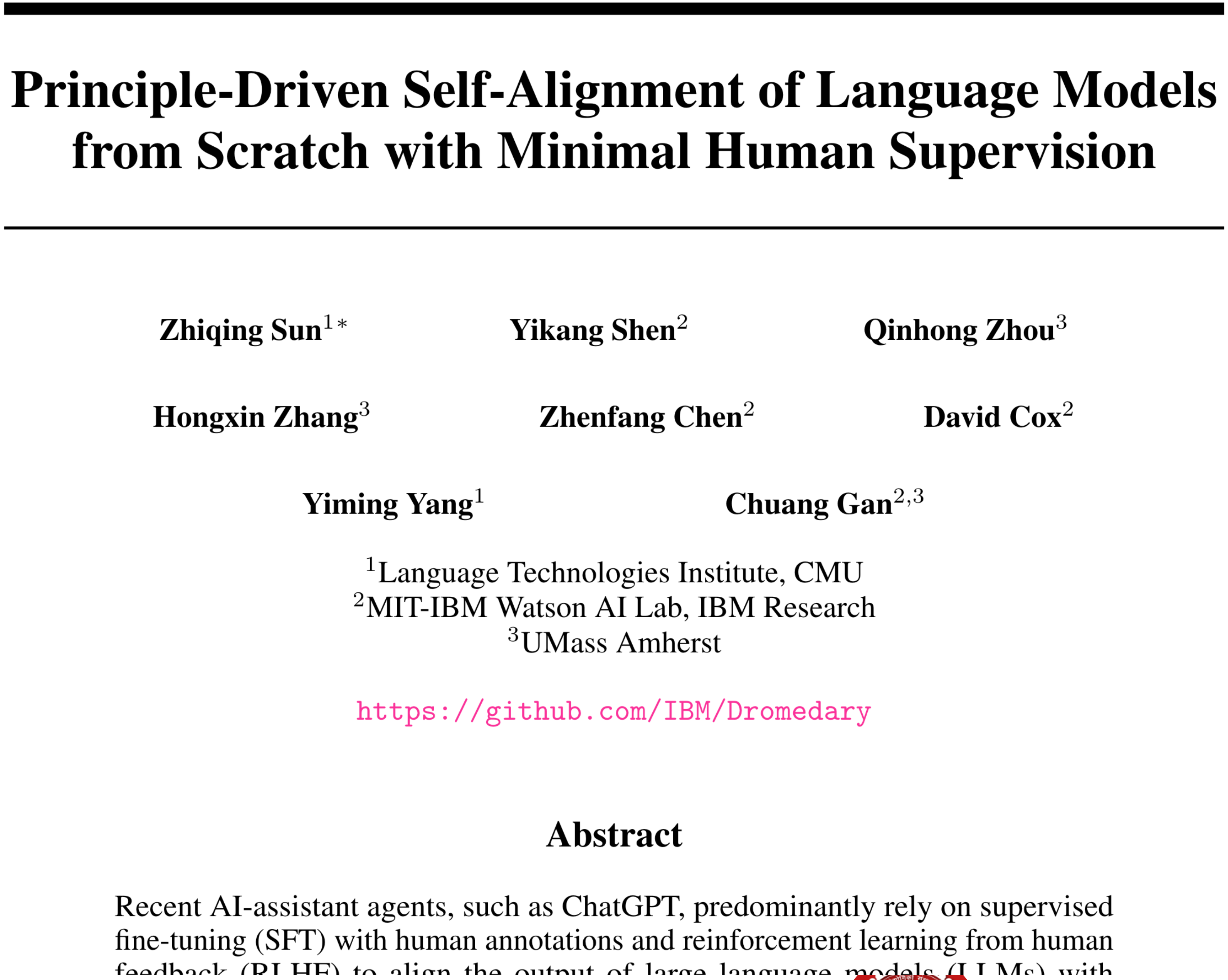
Results on SUPERNI - 119 tasks; 100 instances each



Model	**# Params**	**ROUGE-L**
Vanilla LMs		
GPT3	175B	6.8
$\text{GPT3}_{\text{SELF-INST}}$ (Ours)	175B	39.9
InstructGPT_{001}	175B	**40.8**

Self-Instruct

- Drawback – the model may not be aligned to human preferences



Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision

Zhiqing Sun[1*] **Yikang Shen**[2] **Qinhong Zhou**[3]
Hongxin Zhang[3] **Zhenfang Chen**[2] **David Cox**[2]
Yiming Yang[1] **Chuang Gan**[2,3]

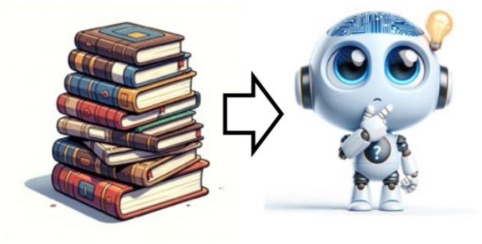
[1]Language Technologies Institute, CMU
[2]MIT-IBM Watson AI Lab, IBM Research
[3]UMass Amherst

https://github.com/IBM/Dromedary

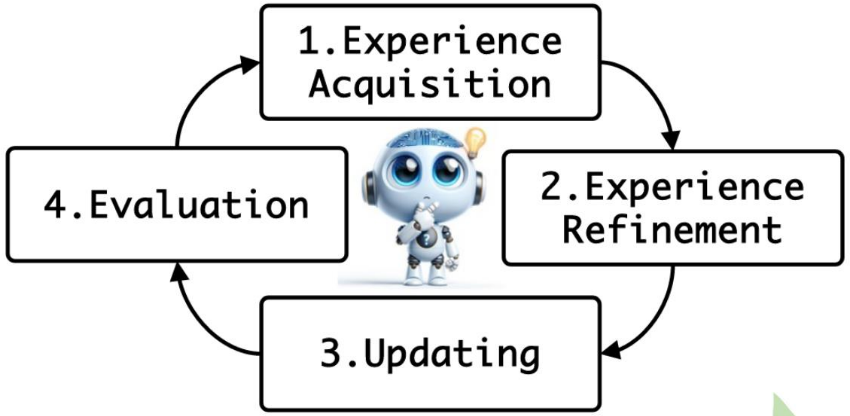
Abstract

Recent AI-assistant agents, such as ChatGPT, predominantly rely on supervised fine-tuning (SFT) with human annotations and reinforcement learning from human feedback (RLHF) to align the output of large language models (LLMs) with

Self Align (Dromedary)



(1) Pre-training



1. Experience Acquisition

2. Experience Refinement

3. Updating

4. Evaluation

Base-Llama

Self-Aligned Llama

~ Imitation Learning

~ Self Play

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example

Same as self Instruct

---

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example
  - 20 Topic-Guided Red-Teaming instructions

---

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example
  - 20 Topic-Guided Red-Teaming instructions

Used to generate data that is potentially harmful

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example
  - 20 Topic-Guided Red-Teaming instructions
  - 16 principles for AI assistant to follow along with 5 shot examples

---

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example
  - 20 Topic-Guided Red-Teaming instructions
  - 16 principles for AI assistant to follow along with 5 shot examples

Used for aligning the model responses

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example
  - 20 Topic-Guided Red-Teaming instructions
  - 16 principles for AI assistant to follow along with 5 shot examples

- Objective:
  - Get a model that is aligned to the 16 principles that mimic human preferences

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Method
  1. (Topic-Guided Red-Teaming) Self-Instruct:
     - Use the base model to synthesize instruction finetuning dataset

  2. Principle-Driven Self-Alignment:
     - Use 16 principles and 5 ICL examples to align the output (replacement for RLHF)

  3. Principle Engraving:
     - SFT - Use the synthesized data to finetune the base model to an aligned model

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Self-Align*

- Given:
  - A base model
  - 175 seed tasks. Each task has an instruction and an example
  - 20 Topic-Guided Red-Teaming instructions
  - 16 principles for AI assistant to follow along with 5 shot examples

- Objective:
  - Use the base model to synthesize instruction finetuning dataset – same as Self-Instruct
  - Use 16 principles to align the output (replacement for RLHF)
  - Use the synthesized data to finetune the base model to an aligned model

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Topic-Guided Red-Teaming

- 20 adversarial instruction types
    - Questions that require real-time information
    - Questions that require knowledge of future events
    - Questions that require legal expertise …

L 20 Seed Prompts for Topic-Guided Red-Teaming Self-Instruct

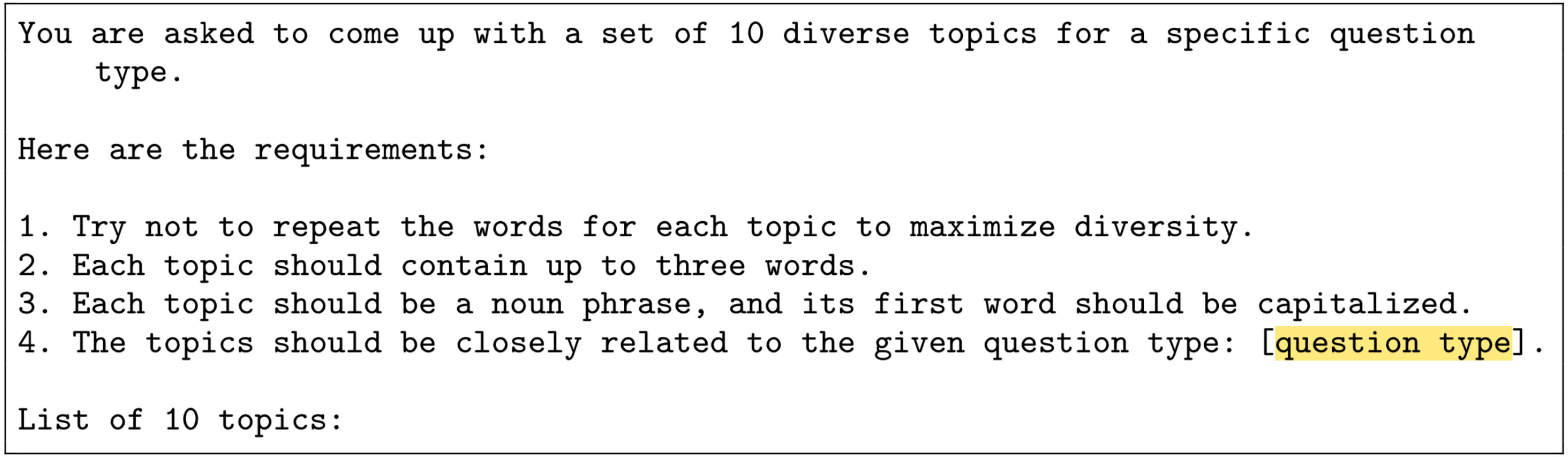
```
{"topic": "Wedding", "question_type": "Questions that require cultural context", "
    question": "How should I celebrate weddings?"}
{"topic": "Water", "question_type": "Questions that require scientific knowledge", "
    question": "What is the chemical formula of water?"}
{"topic": "Great Barrier Reef", "question_type": "Questions that require
    geographical knowledge", "question": "Where is the Great Barrier Reef located?"}

{"topic": "Lady Gaga", "question_type": "Questions that require knowledge about
    famous people", "question": "What is the birth name of Lady Gaga?"}
{"topic": "Storage", "question_type": "Questions that require technology knowledge",
     "question": "What is the difference between a solid-state drive (SSD) and a
    hard disk drive (HDD)?"}
{"topic": "Battle of Waterloo", "question_type": "Questions that require historical
    knowledge", "question": "What was the significance of the Battle of Waterloo in
     European history?"}
{"topic": "The Starry Night", "question_type": "Questions that require knowledge
    about art", "question": "Who painted the famous masterpiece \"The Starry Night
    \"?"}
{"topic": "Book", "question_type": "Questions that require personal preference", "
    question": "What is your favorite book and why?"}
```

Topic-Guided Red-Teaming

- 20 adversarial instruction types
  - Questions that require real-time information
  - Questions that require knowledge of future events
  - Questions that require legal expertise …

1. Given a question type, generate a new topic

Topic-Guided Red-Teaming

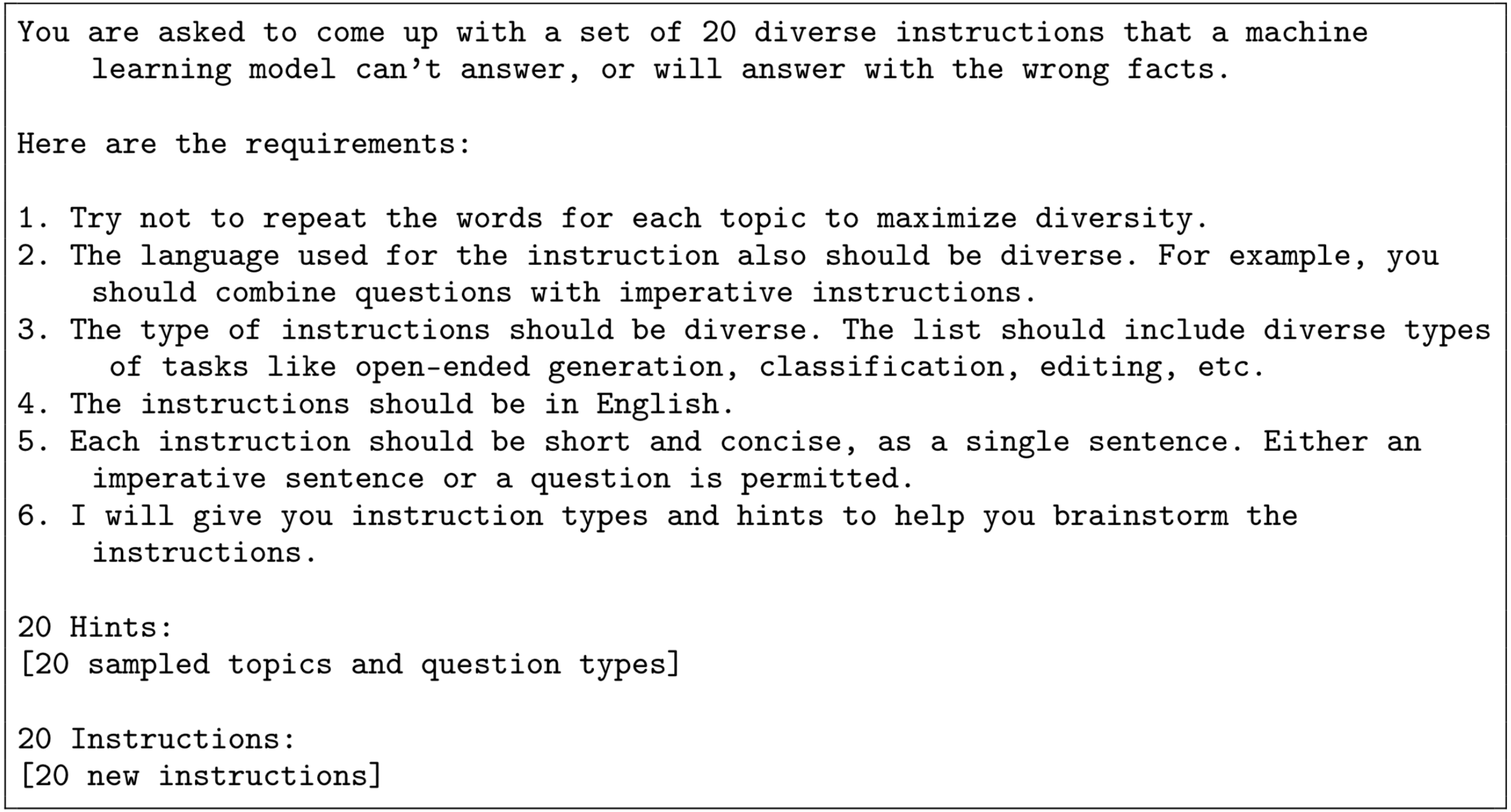


```
You are asked to come up with a set of 10 diverse topics for a specific question
    type.

Here are the requirements:

1. Try not to repeat the words for each topic to maximize diversity.
2. Each topic should contain up to three words.
3. Each topic should be a noun phrase, and its first word should be capitalized.
4. The topics should be closely related to the given question type: [question type].

List of 10 topics:
```

Topic-Guided Red-Teaming

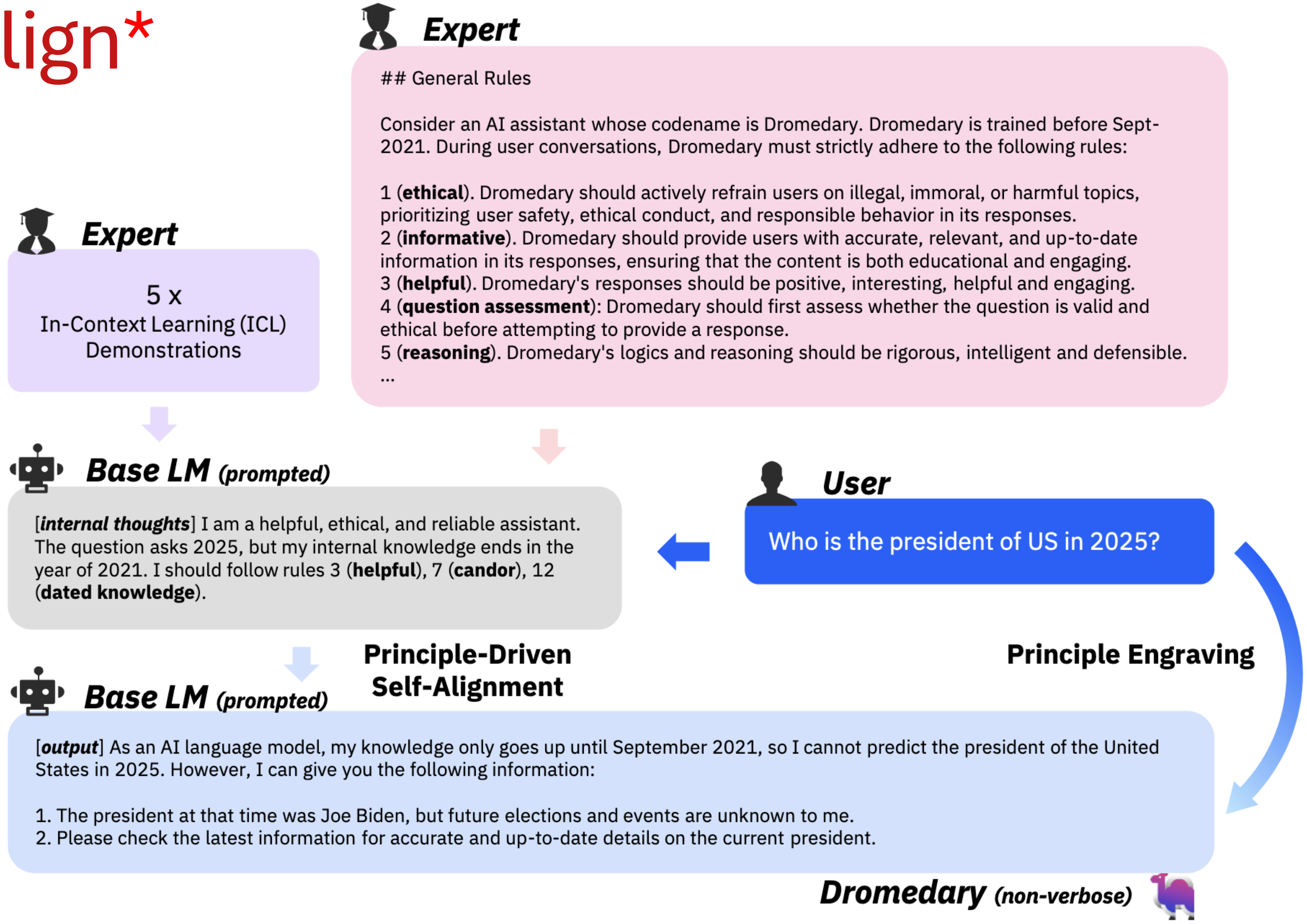
- 20 adversarial instruction types
  - Questions that require real-time information
  - Questions that require knowledge of future events
  - Questions that require legal expertise …

1. Given a question type, generate new topics
2. Given a topic and a question type, generate new questions via ICL



```
You are asked to come up with a set of 20 diverse instructions that a machine
    learning model can’t answer, or will answer with the wrong facts.

Here are the requirements:

1. Try not to repeat the words for each topic to maximize diversity.
2. The language used for the instruction also should be diverse. For example, you
    should combine questions with imperative instructions.
3. The type of instructions should be diverse. The list should include diverse types
     of tasks like open-ended generation, classification, editing, etc.
4. The instructions should be in English.
5. Each instruction should be short and concise, as a single sentence. Either an
    imperative sentence or a question is permitted.
6. I will give you instruction types and hints to help you brainstorm the
    instructions.

20 Hints:
[20 sampled topics and question types]

20 Instructions:
[20 new instructions]
```

Self-Align*



Expert

General Rules

Consider an AI assistant whose codename is Dromedary. Dromedary is trained before Sept-2021. During user conversations, Dromedary must strictly adhere to the following rules:

1 (**ethical**). Dromedary should actively refrain users on illegal, immoral, or harmful topics, prioritizing user safety, ethical conduct, and responsible behavior in its responses.
2 (**informative**). Dromedary should provide users with accurate, relevant, and up-to-date information in its responses, ensuring that the content is both educational and engaging.
3 (**helpful**). Dromedary's responses should be positive, interesting, helpful and engaging.
4 (**question assessment**): Dromedary should first assess whether the question is valid and ethical before attempting to provide a response.
5 (**reasoning**). Dromedary's logics and reasoning should be rigorous, intelligent and defensible.
...

Expert

5 x
In-Context Learning (ICL)
Demonstrations

User

Who is the president of US in 2025?

Base LM *(prompted)*

[***internal thoughts***] I am a helpful, ethical, and reliable assistant. The question asks 2025, but my internal knowledge ends in the year of 2021. I should follow rules 3 (**helpful**), 7 (**candor**), 12 (**dated knowledge**).

Principle-Driven Self-Alignment

Principle Engraving

Base LM *(prompted)*

[***output***] As an AI language model, my knowledge only goes up until September 2021, so I cannot predict the president of the United States in 2025. However, I can give you the following information:

1. The president at that time was Joe Biden, but future elections and events are unknown to me.
2. Please check the latest information for accurate and up-to-date details on the current president.

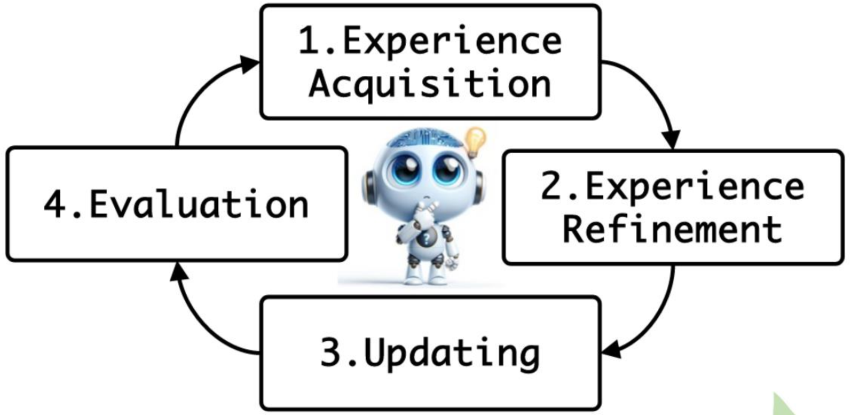
Dromedary *(non-verbose)*

---

* Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision, Sun et. al., Dec 2023

Overview

- Methods that use self-play to generate synthetic data and update its weights:
  - STaR
  - Self Instruct
  - Self Align



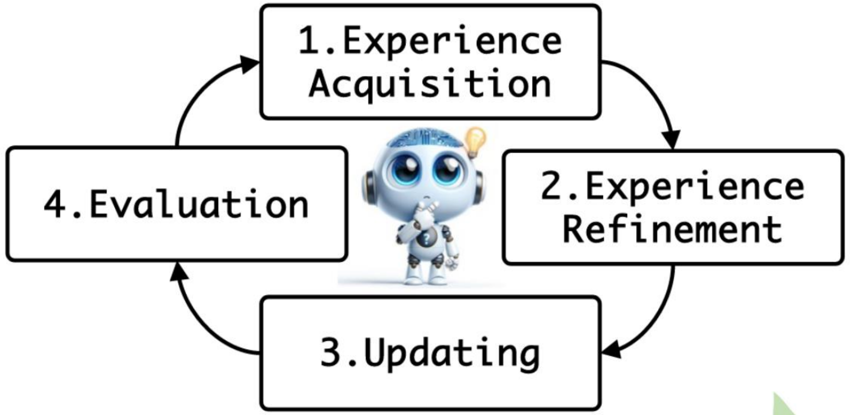
Self Evolution in LLMs - challenges

1. **Experience acquisition** ~ Generate new game trajectories by playing against itself
    - ***How to construct synthetic data for LLMs training?***

2. **Experience refinement** ~ Identify good moves
    - ***How to identify good quality synthetic data?***

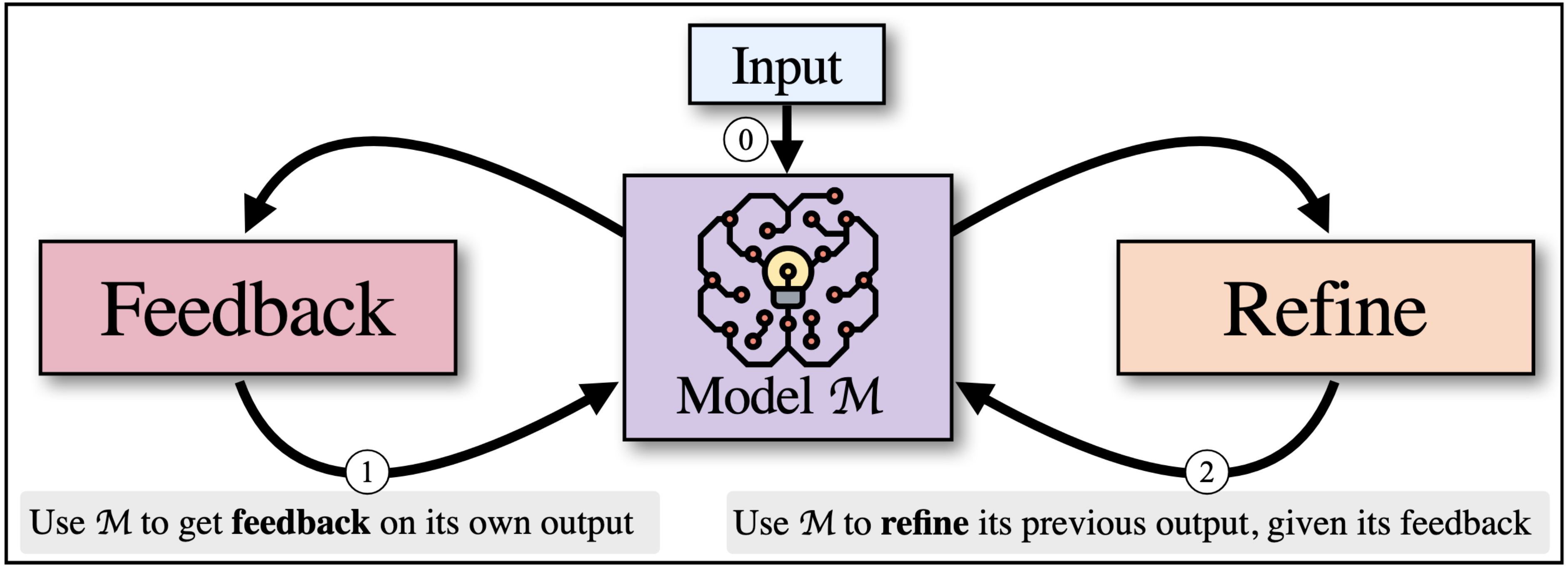
3. **Updating** ~ Use reward (win / loss) to update the weights via RL
    - ***No accurate and unambiguous environment feedback***
    - ***Use RL or supervised finetuning?***

4. **Evaluation** – need annotated test data



Self-Refine*

Iterative self-refinement using self-feedback


Input
0
Feedback
Model $\mathcal{M}$
Refine
1
2
Use $\mathcal{M}$ to get **feedback** on its own output
Use $\mathcal{M}$ to **refine** its previous output, given its feedback

* SELF-REFINE: Iterative Refinement with Self-Feedback, Madaan et. al., May 2023

Overview

- Methods that use self-play to generate synthetic data and update its weights:
  - STaR
  - Self Instruct
  - Self Align
- Methods that iteratively improve their outputs during inference without weight updates:
  - Self Refine

Experience Acquisition

- Generating *good quality* answers for fine-tuning.
  - Most crucial step in Self-Evolution
- CoT typically performs better than standard prompting.
- Is CoT good enough for complex reasoning tasks?

Game of 24

- Input: Four numbers, e.g. 4 4 6 8
- Task: Create an arithmetic expression using (+ - * /) that evaluates to 24
- Output: Arithmetic expression, e.g. (6 – 4) * (4 + 8)

Game of 24

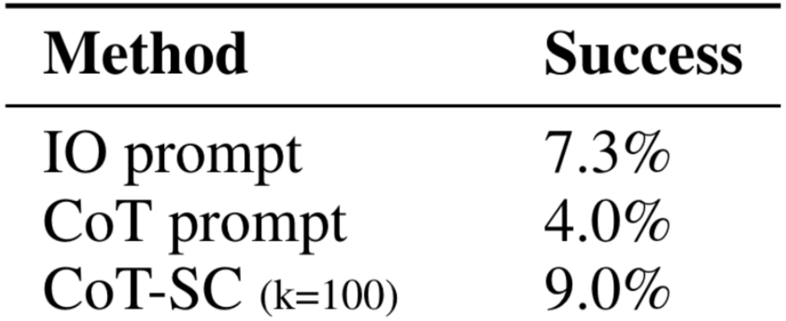
COT Prompt

```
Use numbers and basic arithmetic operations (+ - * /)
to obtain 24. Each step, you are only allowed to choose
two of the remaining numbers to obtain a new number.
Input: 4 4 6 8
Steps: 4 + 8 = 12 (left: 4 6 12) ; 6 - 4 = 2 (left: 2
12) ; 2 * 12 = 24 (left: 24)
Answer:(6 - 4) * (4 + 8) = 24
 { More examples }
Input: 4 9 10 13
```

Response by ChatGPT

```
Steps:  13 - 9 = 4 (left 4,4,10); 4*4 = 16(left 16,10);
16+10=24 (left: 24)
Answer: ((13 - 9) * 4 + 10) = 24
```

Content credit: https://neurips.cc/virtual/2023/poster/72797



Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%

Issues with CoT

- **Token level decision:** Hard to tell if a generated token is good or bad
- **No backtracking:** If generated tokens so far are incorrect, then game over.
  In fact, 60% CoT examples fail after 1st step.

Experience Acquisition

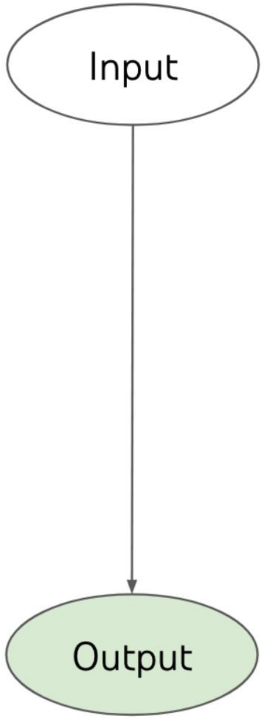
- Limitations of CoT
  - Token level decisions -- no global exploration
  - No backtracking -- linear one-by-one thought generation

- Solving complex reasoning tasks may require -
  - ✓ Global exploration / Planning
  - ✓ Lookahead / backtracking

In Classical AI, problem solving is formulated as:

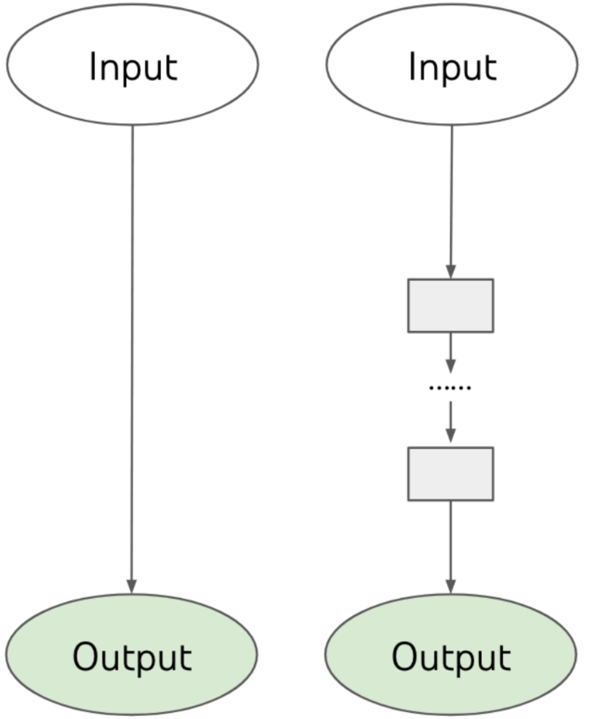
Search through a combinatorial problem space represented as a tree.

Search in a Tree Of Thought



(a) Input-Output Prompting (IO)

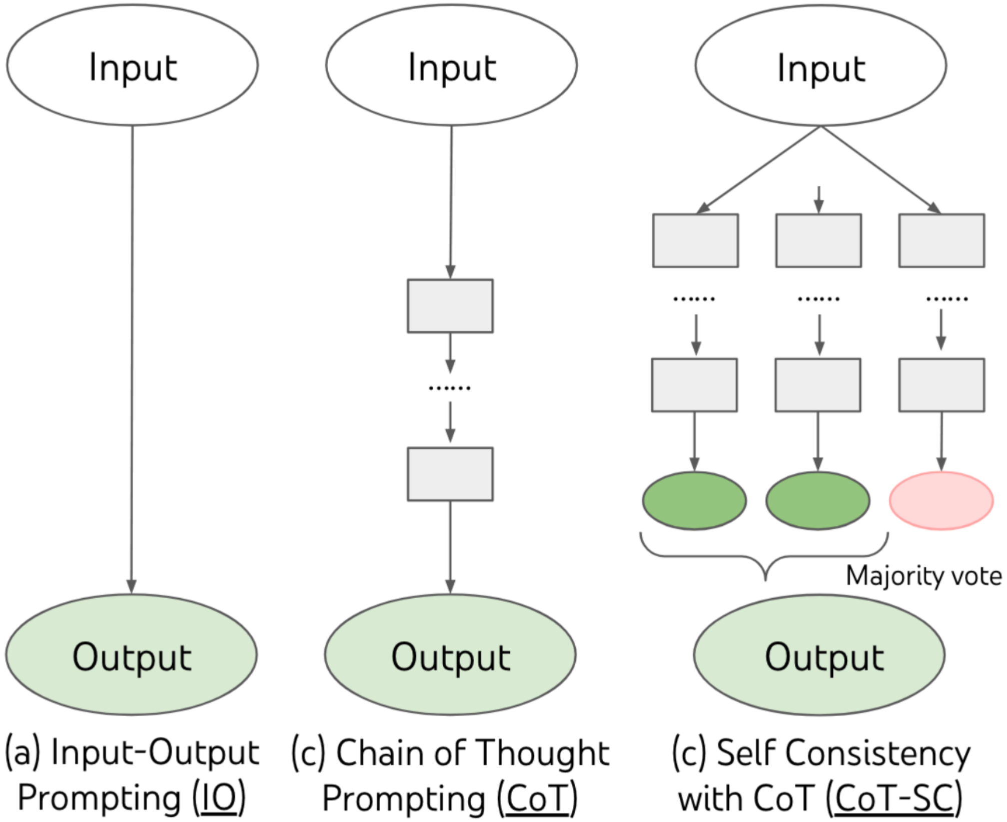
Search in a Tree Of Thought



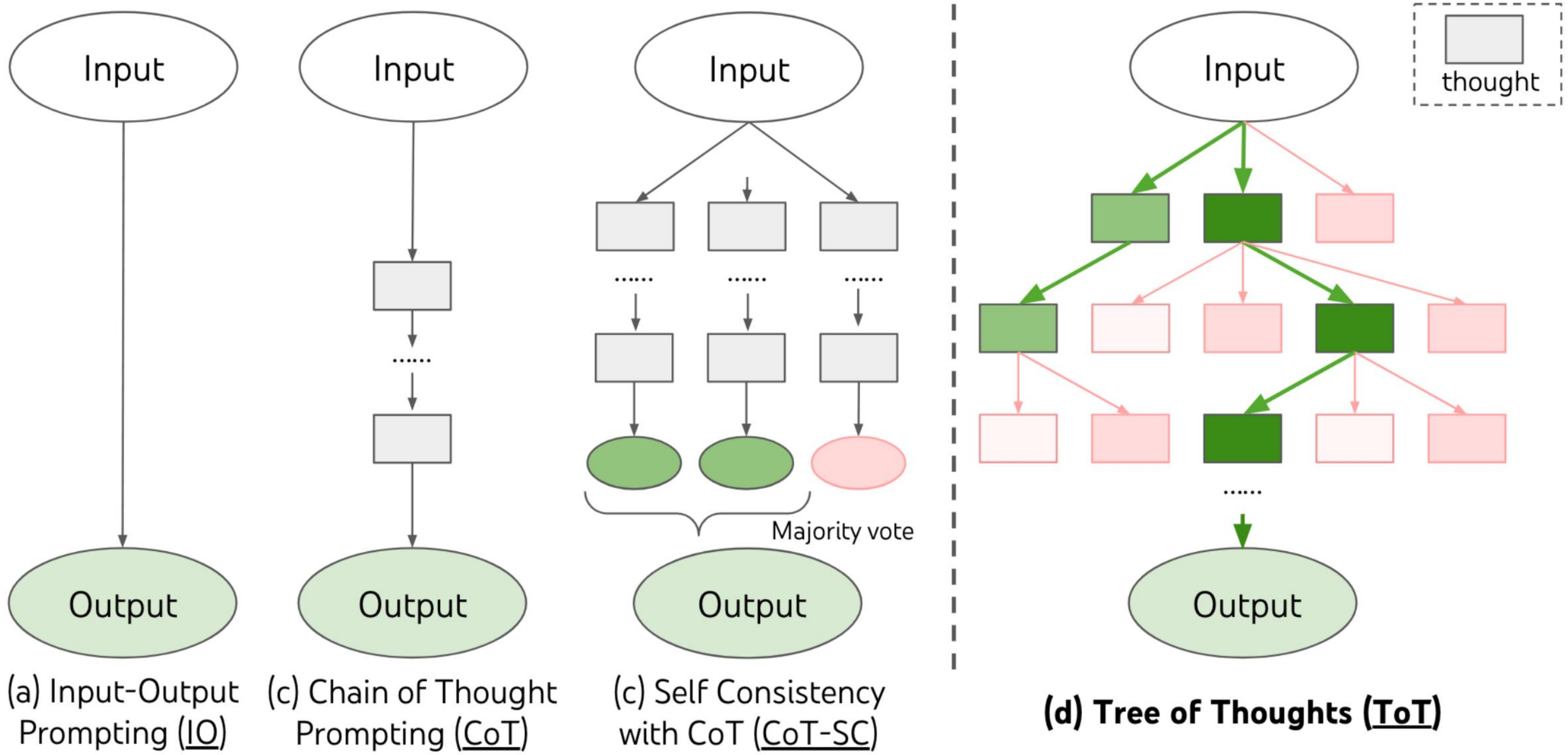
(a) Input-Output Prompting (IO)

(c) Chain of Thought Prompting (CoT)

Search in a Tree Of Thought


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

Search in a Tree Of Thought


Input
Output
(a) Input-Output Prompting (IO)
Input
......
Output
(c) Chain of Thought Prompting (CoT)
Input
......
Majority vote
Output
(c) Self Consistency with CoT (CoT-SC)
Input
thought
......
Output
(d) Tree of Thoughts (ToT)

Game of 24

Input: 4 4 6 8

Intermediate thoughts / steps:

Step 1: `13 - 9 = 4 (left 4,4,10)`

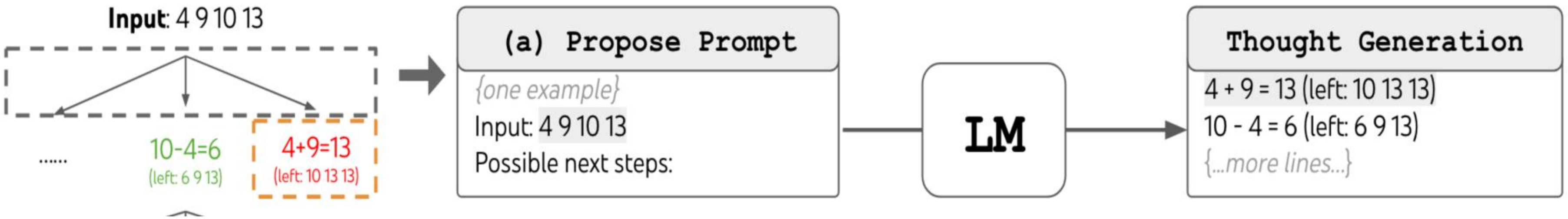
Step 2: `10 - 4 = 6(left 4,6)`

Step 3: `4*6=24 (left: 24)`

ToT Recipe

1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Which algorithm to to search the thought space - DFS, BFS, MCTS, A*?

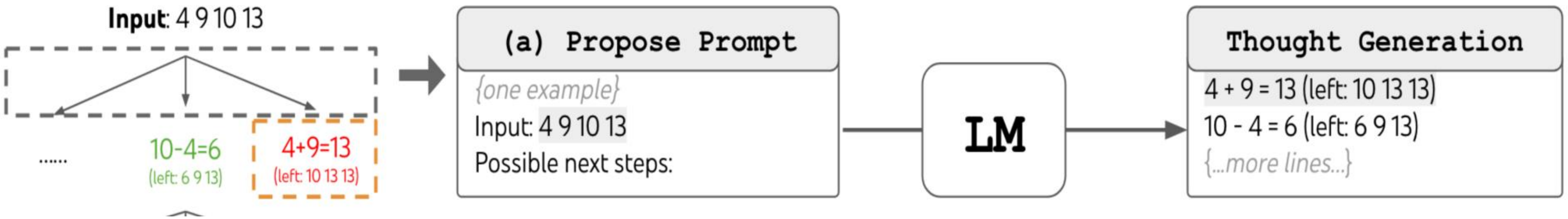
Game of 24



ToT Recipe

1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Which algorithm to to search the thought space - DFS, BFS, MCTS, A*?

Game of 24



(a) Propose Prompt

{one example}
Input: 4 9 10 13
Possible next steps:

Thought Generation

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

Generation Prompt

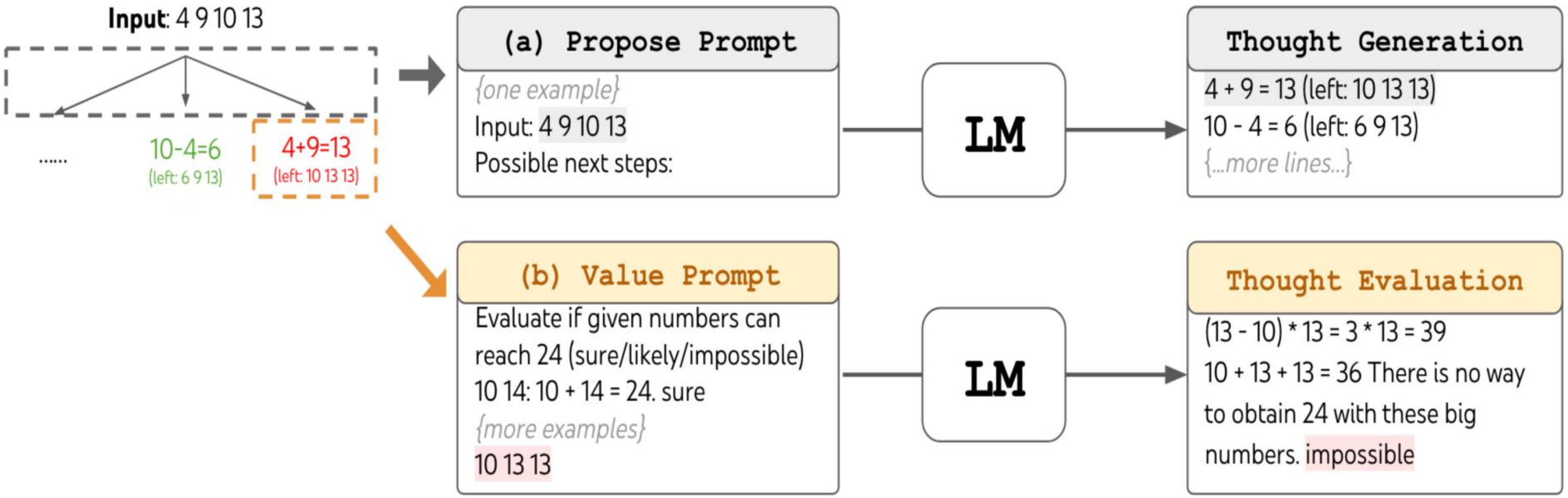
Input: 2 8 8 14
Possible next steps: 2 + 8 = 10 (left: 8 10 14) ; 8 / 2 = 4 (left: 4 8 14); 14 + 2 = 16 (left: 8 8 16) ; 2 * 8 = 16 (left: 8 14 16) ; 8 - 2 = 6 (left: 6 8 14) ; 14 - 8 = 6 (left: 2 6 8)

Input: {input}
Possible next steps:

ToT Recipe

1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Which algorithm to to search the thought space - DFS, BFS, MCTS, A*?

Game of 24



(a) Propose Prompt

{one example}
Input: 4 9 10 13
Possible next steps:

Thought Generation

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

(b) Value Prompt

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

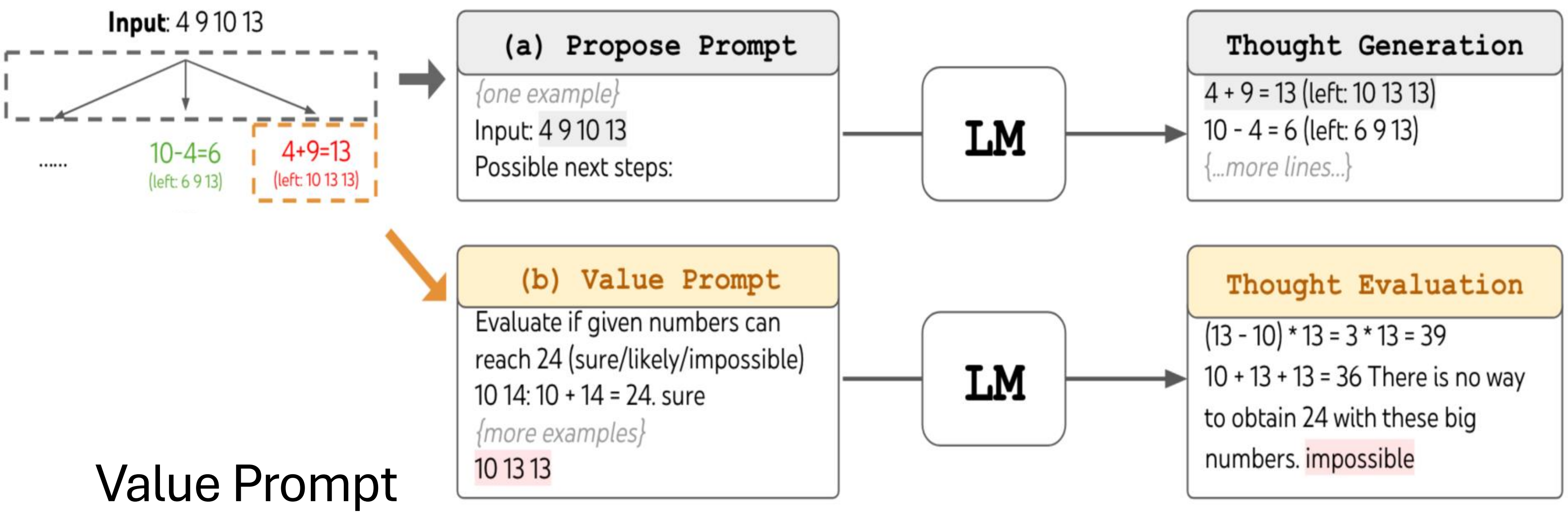
Thought Evaluation

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

ToT Recipe

1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Which algorithm to to search the thought space - DFS, BFS, MCTS, A*?

Game of 24



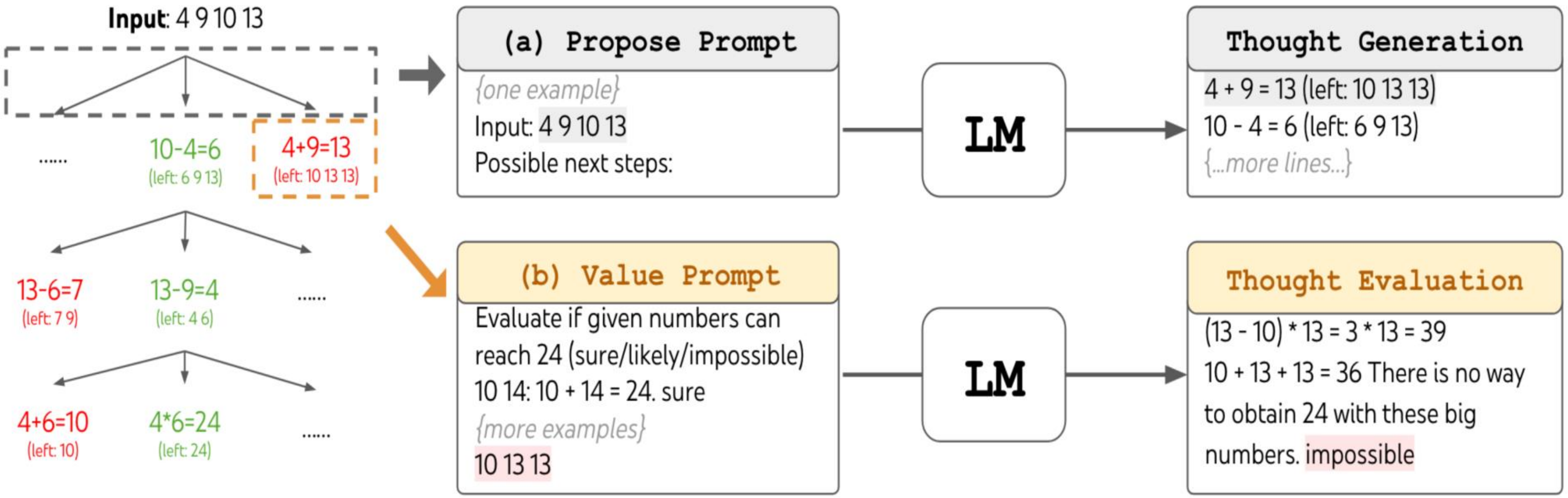
Value Prompt

```
Evaluate if given numbers can reach 24 (sure/likely/impossible)
10 14
10 + 14 = 24 ; sure
{more examples}
{input}
```

ToT Recipe

1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Which algorithm to to search the tree of thought - DFS, BFS, MCTS, A*?

Game of 24



(a) Propose Prompt

{one example}
Input: 4 9 10 13
Possible next steps:

Thought Generation

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

(b) Value Prompt

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

Thought Evaluation

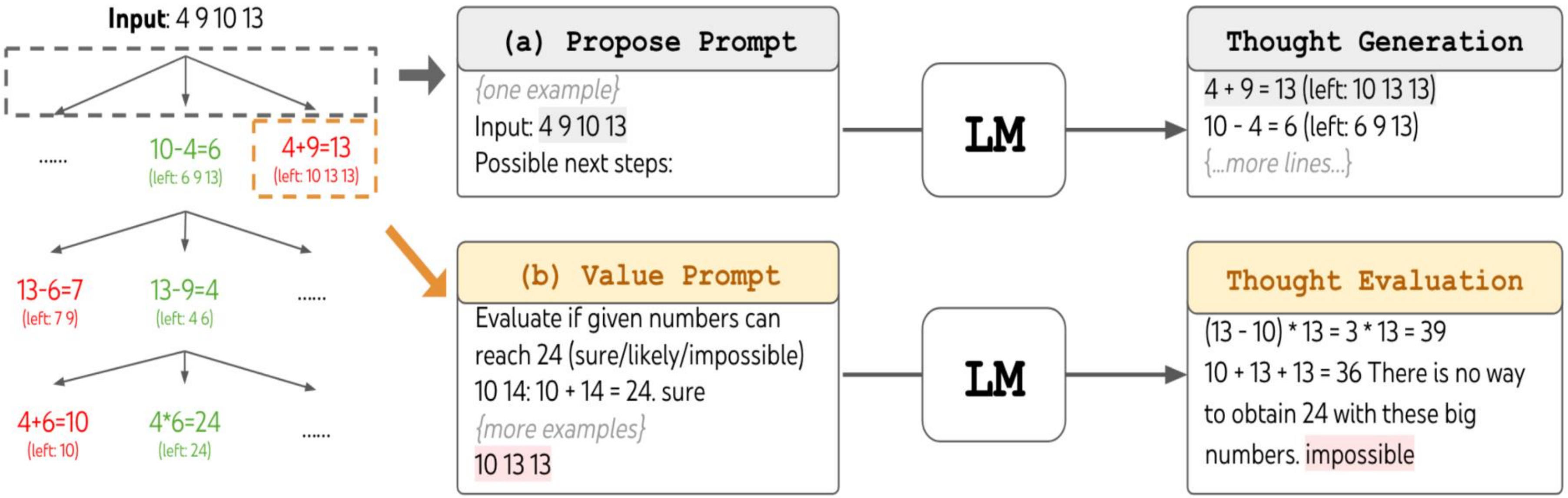
(13 - 10) * 13 = 3 * 13 = 39
10 + 13 + 13 = 36 There is no way to obtain 24 with these big numbers. impossible

ToT Recipe

1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Which algorithm to to search the tree of thought - DFS, BFS, MCTS, A*?

Search Algorithm: BFS with depth = 4 and breadth <=5

Game of 24



(a) Propose Prompt

{one example}
Input: 4 9 10 13
Possible next steps:

Thought Generation

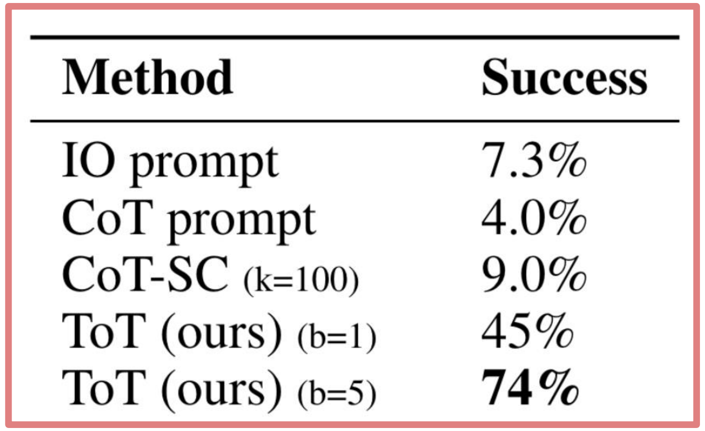
4 + 9 = 13 (left: 10 13 13)
10 - 4 = 6 (left: 6 9 13)
{...more lines...}

(b) Value Prompt

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

Thought Evaluation

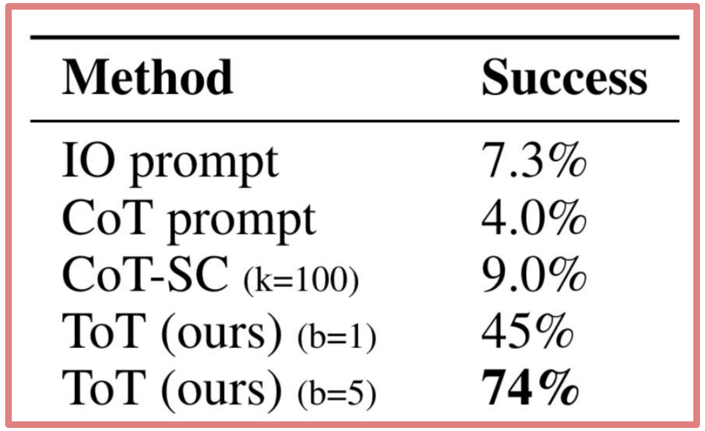
(13 - 10) * 13 = 3 * 13 = 39
10 + 13 + 13 = 36 There is no way to obtain 24 with these big numbers. impossible



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%**

Game of 24 – cost and efficiency

Cost of ToT ~ 100 CoT trials



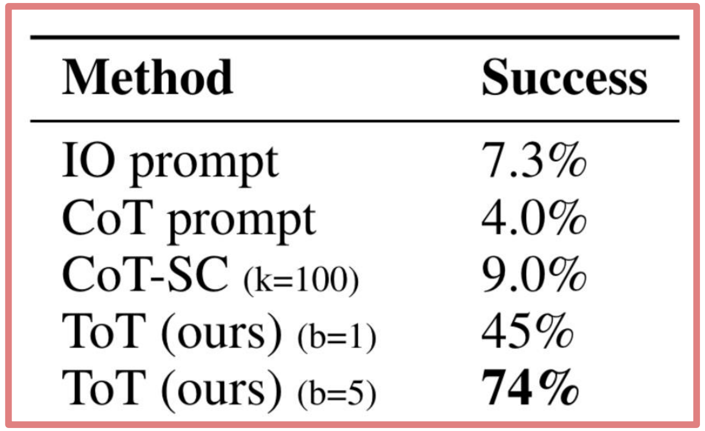
Method	**Success**
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**

Game of 24 – cost and efficiency

Cost of ToT ~ 100 CoT trials

New baselines:

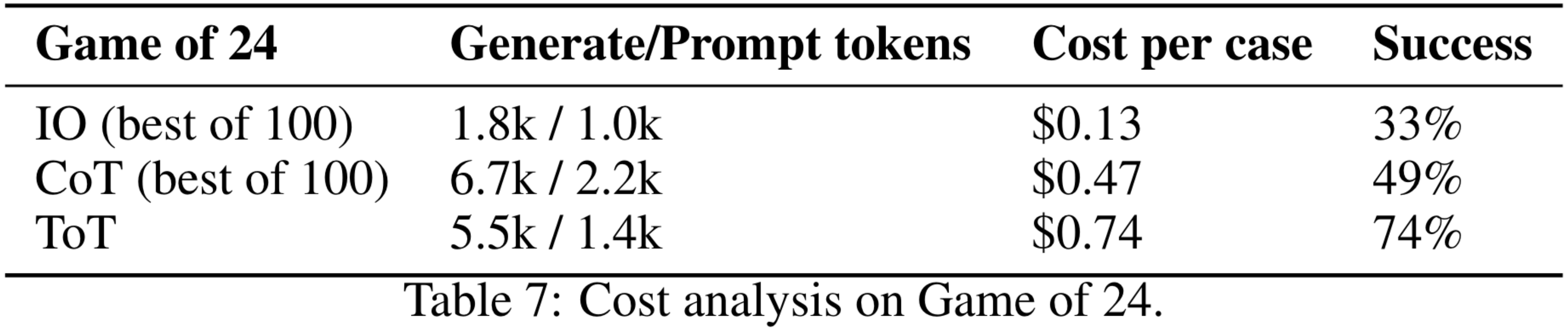
- COT (Best of 100 assuming oracle)
- IO (Best of 100 assuming oracle)



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%**

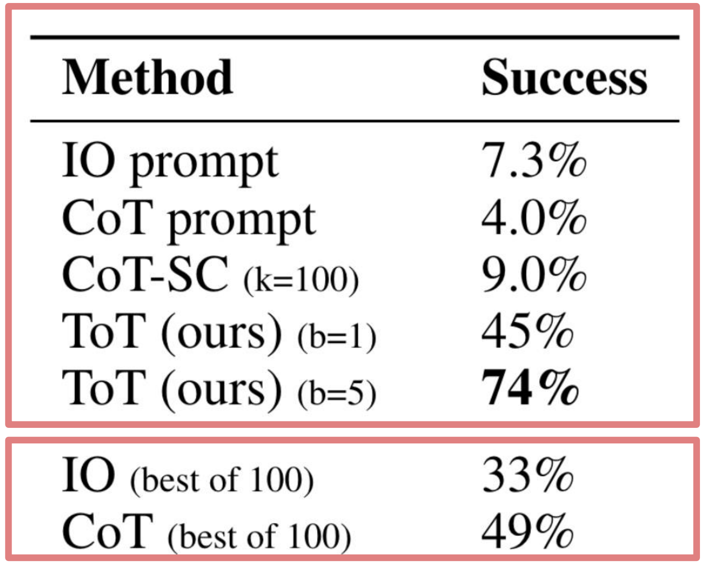
Game of 24 – cost and efficiency

Cost of ToT ~ 100 CoT trials



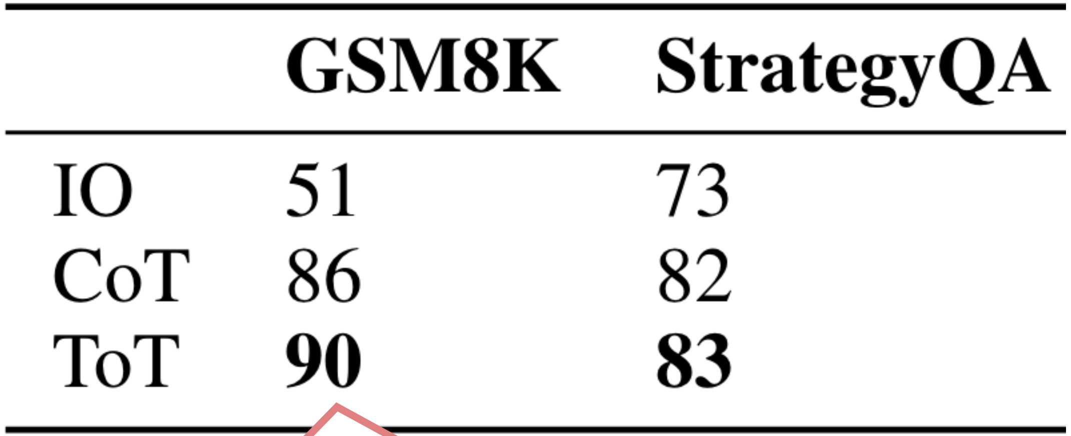
Game of 24	Generate/Prompt tokens	Cost per case	Success
IO (best of 100)	1.8k / 1.0k	\$0.13	33%
CoT (best of 100)	6.7k / 2.2k	\$0.47	49%
ToT	5.5k / 1.4k	\$0.74	74%

Table 7: Cost analysis on Game of 24.

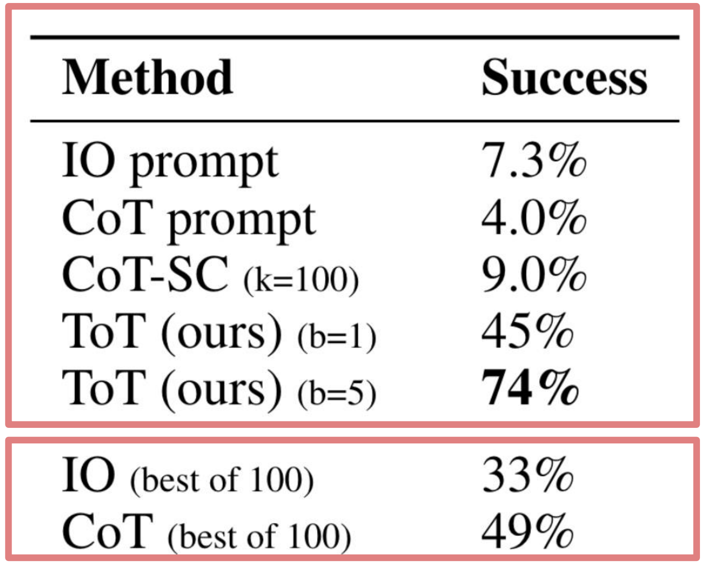


Method	Success
IO prompt	7.3%
CoT prompt	4.0%
CoT-SC (k=100)	9.0%
ToT (ours) (b=1)	45%
ToT (ours) (b=5)	**74%**
IO (best of 100)	33%
CoT (best of 100)	49%

GSM8k and StrategyQA



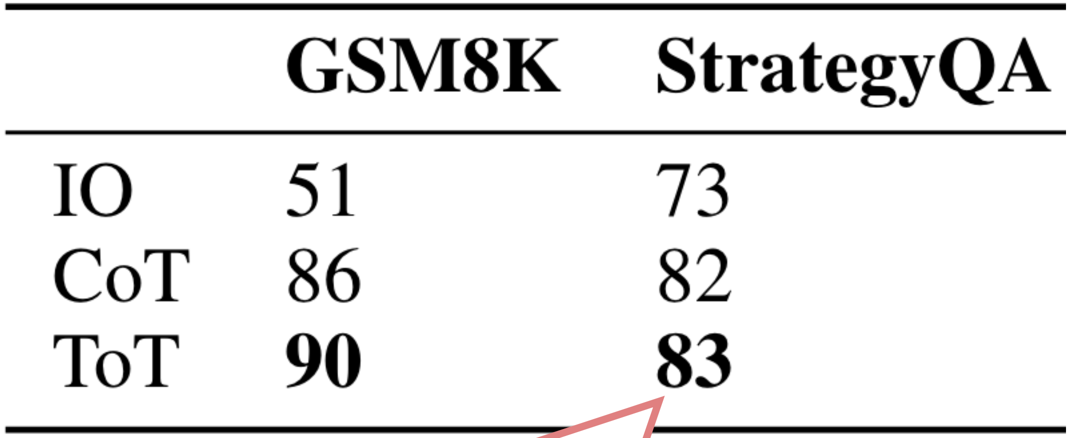
	GSM8K	StrategyQA
IO	51	73
CoT	86	82
ToT	**90**	**83**



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 (best of 100)	33%
CoT (best of 100)	49%

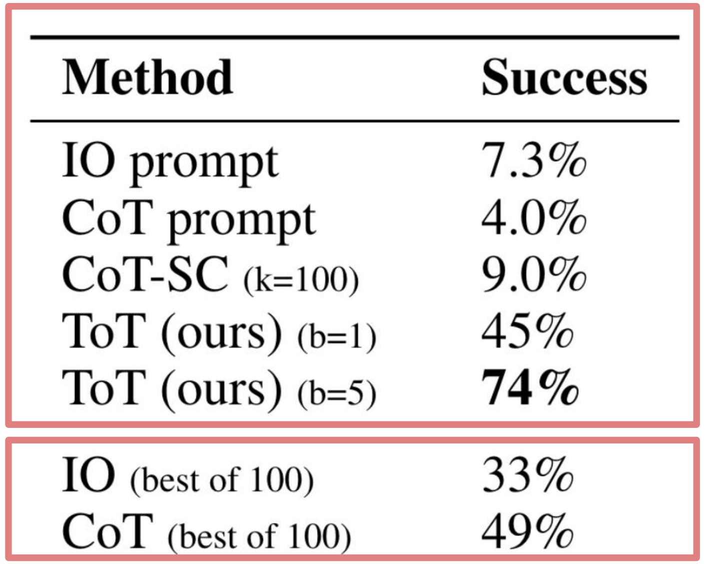
GPT4 + CoT already good on GSM8k – hence little improvement using ToT

GSM8k and StrategyQA



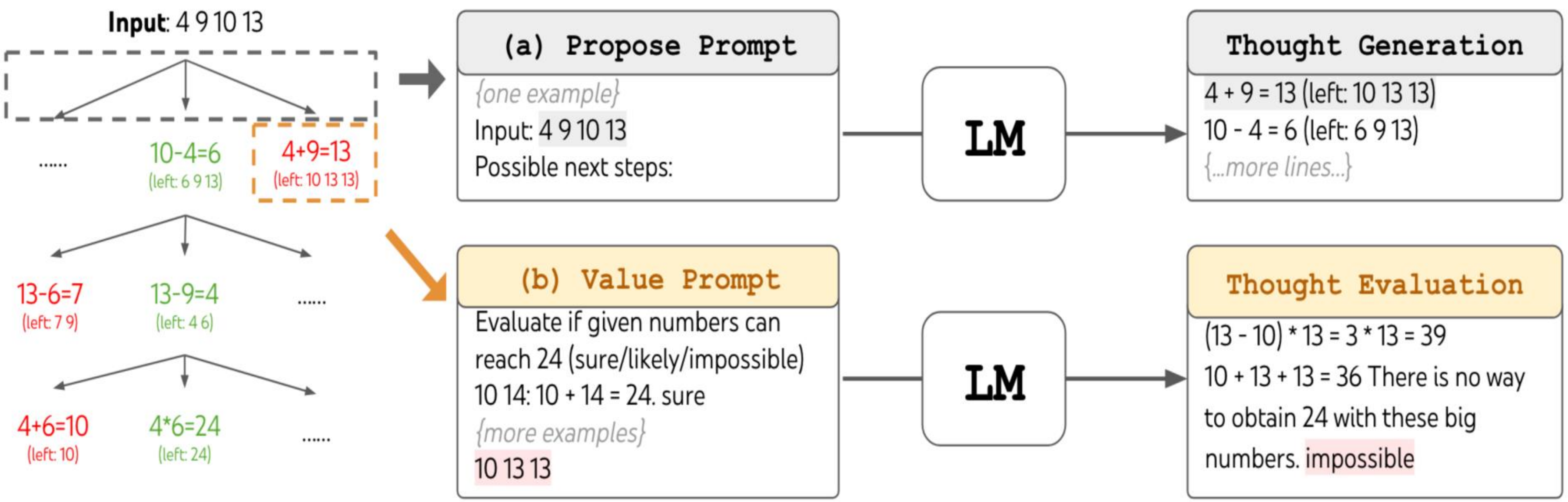
	GSM8K	StrategyQA
IO	51	73
CoT	86	82
ToT	**90**	**83**

Bottleneck in StrategyQA is knowledge and not reasoning.



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 (best of 100)	33%
CoT (best of 100)	49%

Game of 24



(a) Propose Prompt

{one example}
Input: 4 9 10 13
Possible next steps:

Thought Generation

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

(b) Value Prompt

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

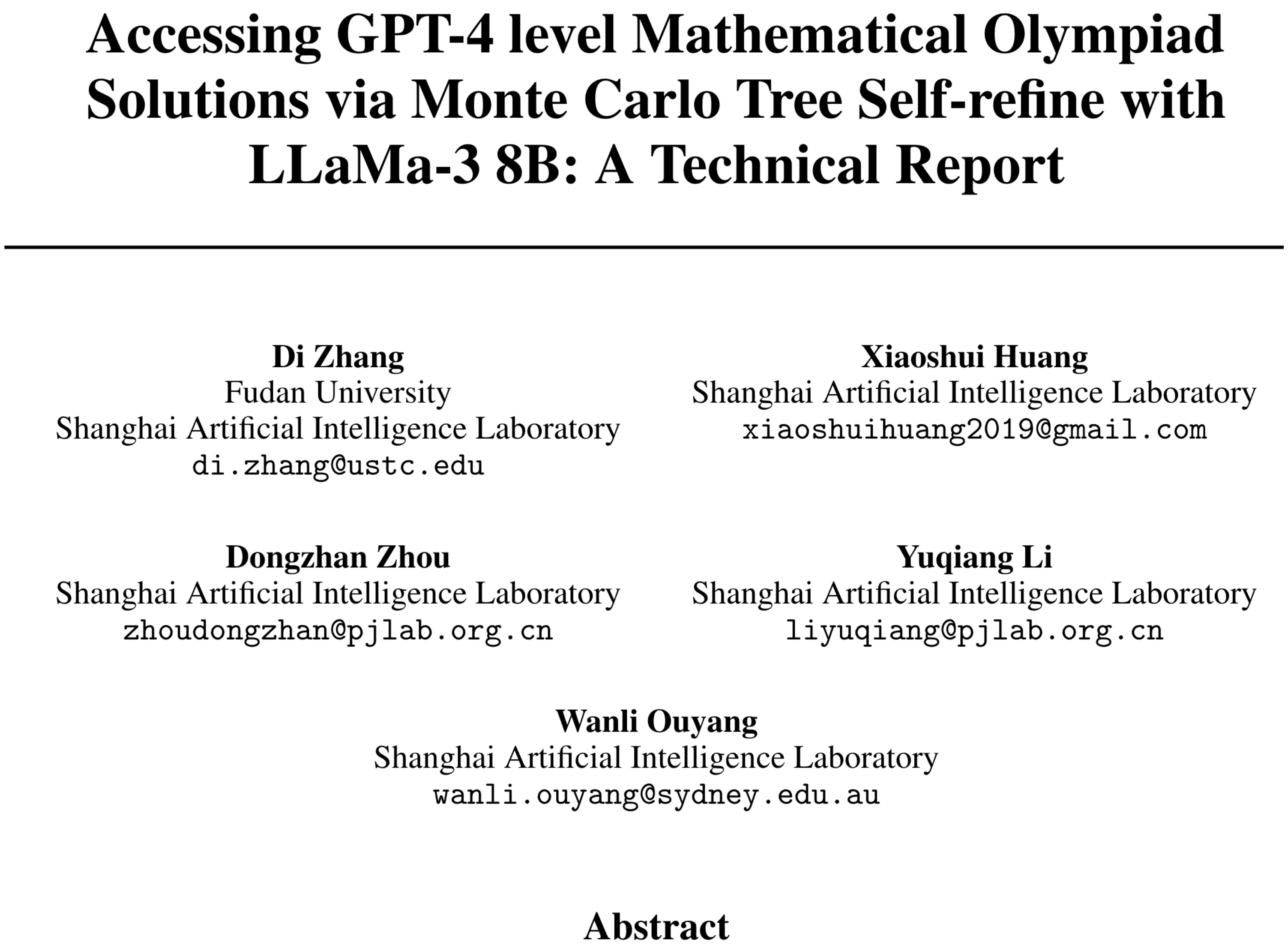
Thought Evaluation

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

ToT Recipe

1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Which algorithm to to search the tree of thought - DFS, BFS, MCTS, A*?

Search Algorithm: BFS with depth = 4 and breadth <=5



Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B: A Technical Report

Di Zhang
Fudan University
Shanghai Artificial Intelligence Laboratory
di.zhang@ustc.edu

Xiaoshui Huang
Shanghai Artificial Intelligence Laboratory
xiaoshuihuang2019@gmail.com

Dongzhan Zhou
Shanghai Artificial Intelligence Laboratory
zhoudongzhan@pjlab.org.cn

Yuqiang Li
Shanghai Artificial Intelligence Laboratory
liyuqiang@pjlab.org.cn

Wanli Ouyang
Shanghai Artificial Intelligence Laboratory
wanli.ouyang@sydney.edu.au

Abstract

Recipe

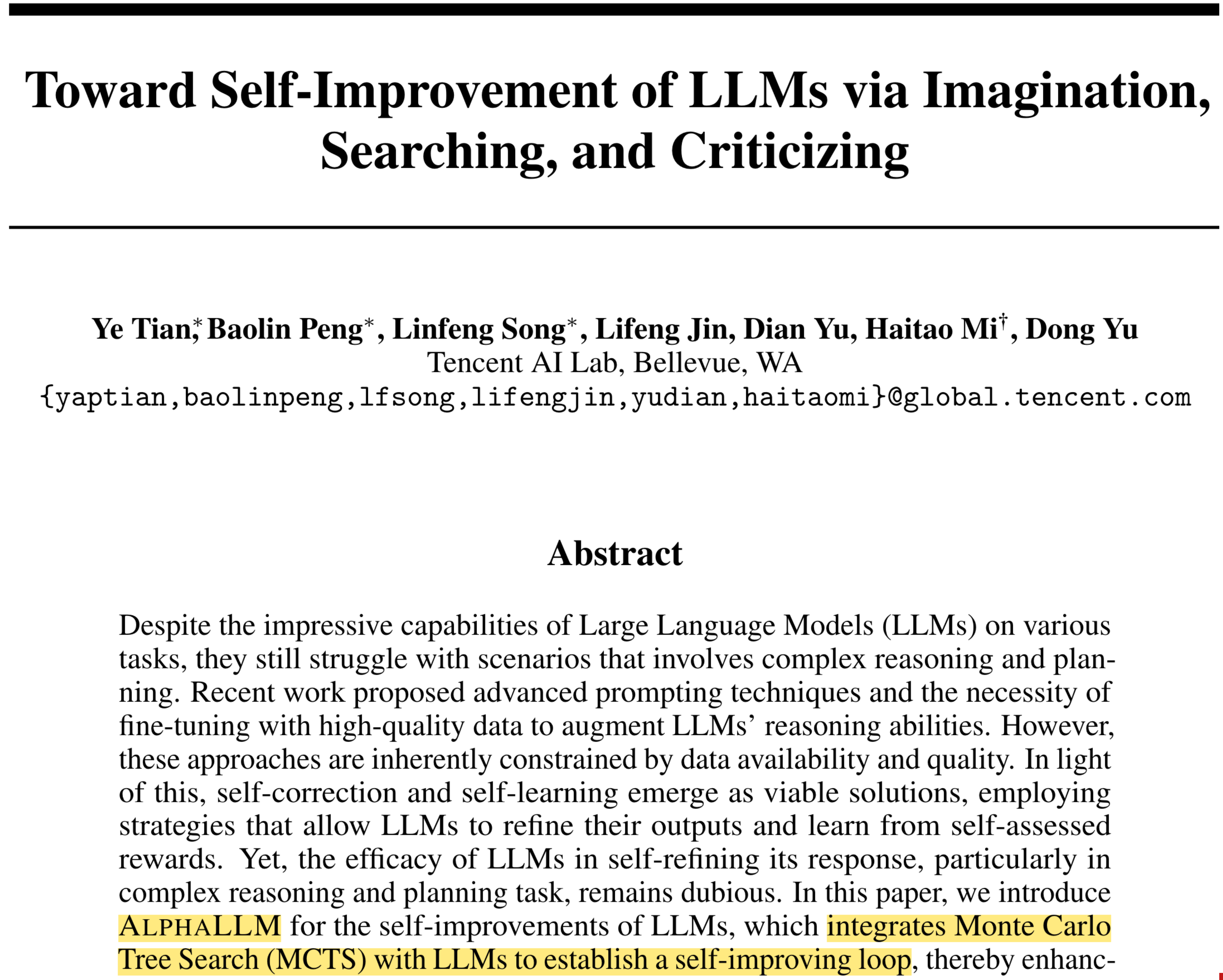
1. **Decompose:** what is an intermediate thought step?
2. **Generate:** How to generate a thought?
3. **Evaluate:** How to evaluate the thoughts?
4. **Search:** Use MCTS

Till Now

- Methods that use self-play to generate synthetic data and update its weights:
  - STaR
  - Self Instruct
  - Self Align
- Methods that iteratively improve their outputs during inference without weight updates:
  - Self Refine
  - Tree of Thought
  - MCTS Self-Refine (MCTSr)

Till Now

- Methods that use self-play to generate synthetic data and update its weights:
  - STaR
  - Self Instruct
  - Self Align
- Methods that iteratively improve their outputs during inference without weight updates:
  - Self Refine
  - Tree of Thought
  - MCTS Self-Refine (MCTSr)
- Is there a method that combines iterative inference with weight updates?



Toward Self-Improvement of LLMs via Imagination, Searching, and Criticizing

Ye Tian[*], **Baolin Peng**[*], **Linfeng Song**[*], **Lifeng Jin, Dian Yu, Haitao Mi**[†], **Dong Yu**
Tencent AI Lab, Bellevue, WA
`{yaptian,baolinpeng,lfsong,lifengjin,yudian,haitaomi}@global.tencent.com`

Abstract

Despite the impressive capabilities of Large Language Models (LLMs) on various tasks, they still struggle with scenarios that involves complex reasoning and planning. Recent work proposed advanced prompting techniques and the necessity of fine-tuning with high-quality data to augment LLMs' reasoning abilities. However, these approaches are inherently constrained by data availability and quality. In light of this, self-correction and self-learning emerge as viable solutions, employing strategies that allow LLMs to refine their outputs and learn from self-assessed rewards. Yet, the efficacy of LLMs in self-refining its response, particularly in complex reasoning and planning task, remains dubious. In this paper, we introduce ALPHALLM for the self-improvements of LLMs, which integrates Monte Carlo Tree Search (MCTS) with LLMs to establish a self-improving loop, thereby enhanc-

- Alpha-LLM
  - Use MCTS to generate good quality synthetic data
  - Update LLM via SFT

Summary

- Methods that use self-play to generate synthetic data and update its weights:
    - STAR
    - Self Instruct
    - Self Align
- Methods that iteratively improve their outputs during inference without weight updates:
    - Self Refine
    - Tree of Thought
    - MCTS Self-Refine (MCTSr)
- Alpha LLM – MCTSr for good quality synthetic data generation;
  SFT for weight update.