Carnegie Mellon University
Language Technologies Institute

Complex Few-shot Reasoning with LLMs

Aman Madaan

Complex Few-shot Reasoning with LLMs

Goal Today:

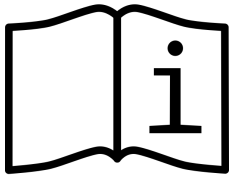
- Try to cover the details of all possible exciting recent works 
- Provide a high-level summary of a *class* of few-shot prompting techniques 

I will definitely miss interesting works:

- Please reach out! We will add it to our website

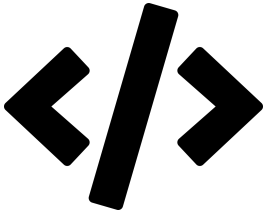
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration

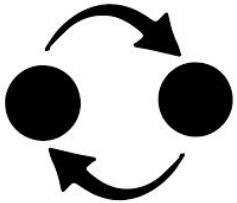


Instructions

Tool Augmentation



Structured Generation



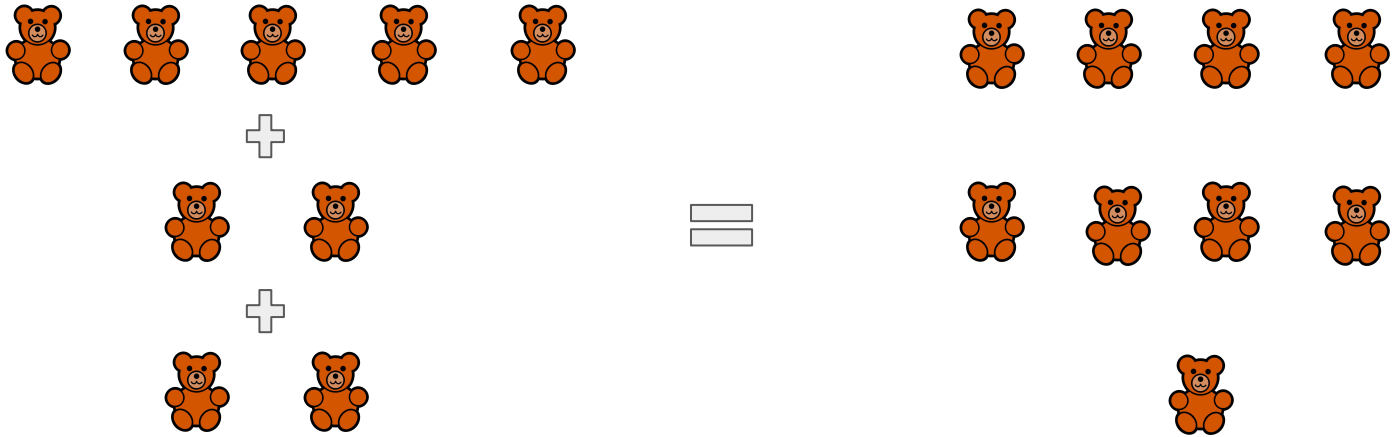
Feedback

Memory

Background

Solving middle-school math problems

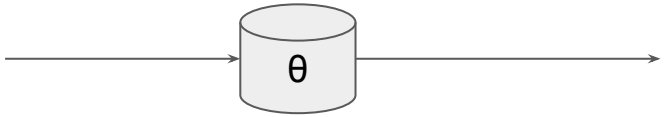
Q: Shawn has 5 toys. For Christmas, he got 2 toys each from his mom and dad. How many toys does he have now?



A: The answer is 9 toys

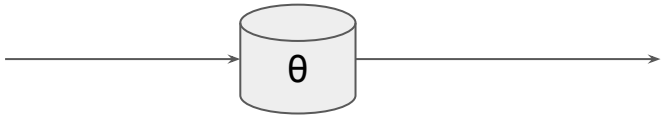
Fine-tuning

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?



A: The answer is 5 cars.

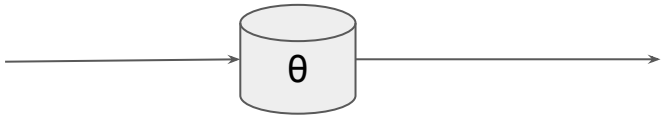
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?



A: The answer is 39 pieces.

Train/Fine-tune

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?



A: The answer is 9 toys

Test

Few-shot prompting

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

A: *The answer is 5 cars.*

Prompt

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?

A:

Test Example

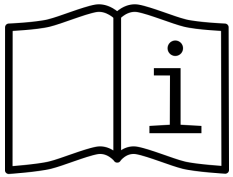
The answer is 9 toys

Rest of the slides

- General idea of the technique
- Representative Work

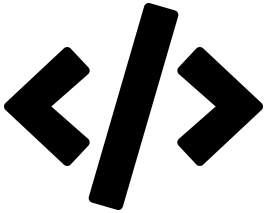
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration

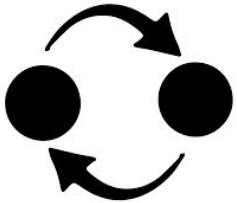


Instructions

Tool Augmentation



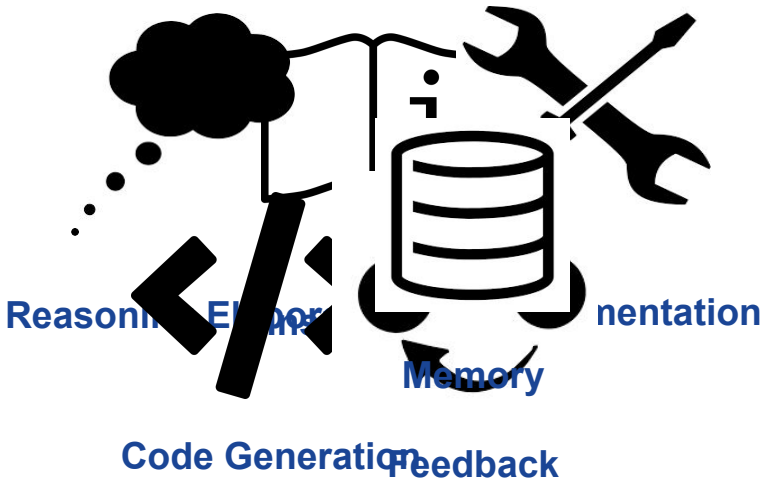
Structured Generation



Feedback

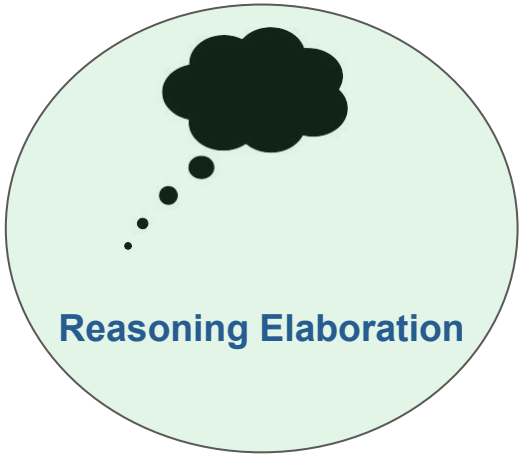
Memory

Complex Few-shot Reasoning with LLMs: Key Techniques

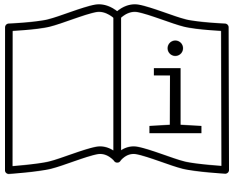


Techniques overlap in practice – focus on the main contribution

Complex Few-shot Reasoning with LLMs: Key Techniques

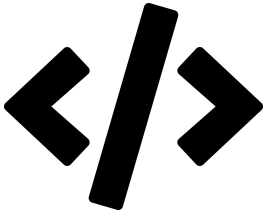


Reasoning Elaboration

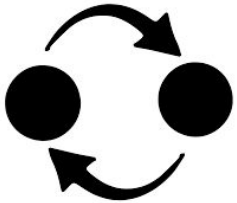


Instructions

Tool Augmentation



Structured Generation



Feedback

Memory

Few-shot prompting

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

A: *The answer is 5 cars.*

Prompt

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?

A:

Test Example

The answer is 9 toys

Few-shot prompting

Direct Prompt

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

A: *The answer is 5 cars.*

Chain-of-thought Prompting (Wei et al. 2022)

Direct Prompt

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

A: *The answer is 5 cars.*

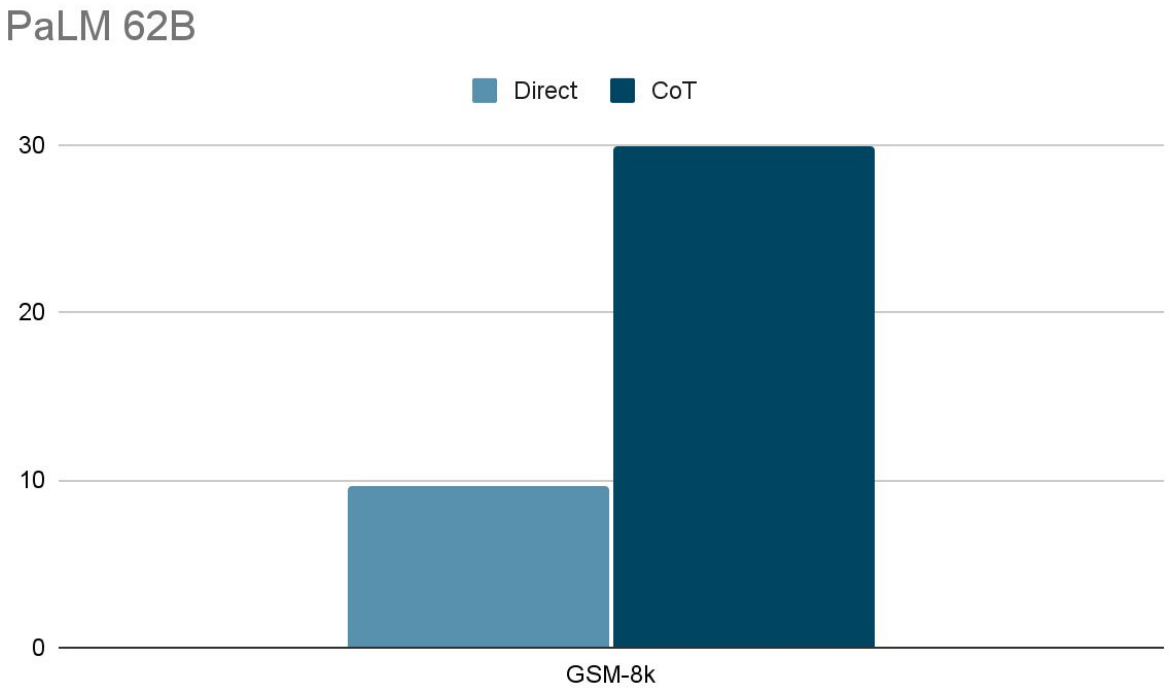
Chain-of-Thought Prompt

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

Thought (T): There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5.

A: *The answer is 5 cars.*

Chain of thought prompting is extremely effective



Chain-of-thought Prompting (CoT)

- General idea:
  - Standard prompt:
    - Q → A
  - Chain-of-thought prompt:
    - Q → Reasoning Process, A
- Many existing prompting techniques can be seen as an improvement over the general CoT strategy
  - The thought can be: Text, Code, API calls
- Similar ideas:
  - Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems (Ling et al. 2017)
  - Think about it! Improving defeasible reasoning by first modeling the question scenario (Madaan et al. 2021).
  - Show your work: Scratchpads for Intermediate Computation with Language Models (Nye et al. 2021)

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

Thought (T): There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5.

A: *The answer is 5 cars.*

Least-to-Most Prompting (Zhou et al. 2022)

- Breakdown the reasoning process into steps
  - Decompose the problem into simpler sub-problems
  - Solve simpler sub-problems

Least-to-Most Prompting

Question

Question Decomposition

Subproblem Solving

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

Q: How many cars are in the parking lot after the first car arrives?
Q: How many cars are in the parking lot after the second car arrives?

Q: How many cars are in the parking lot after the first car arrives?
A: There are originally 3 cars. After the first car arrives, we have 3 + 1 = 4 cars.
Q: How many cars are in the parking lot after the second car arrives?
A: After the first car arrives, we have 3 + 1 = 4 cars. After the second car arrives, we have 4 + 1 = 5 cars.

Least-to-Most Prompting

Least-to-most prompting (solving stage)

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

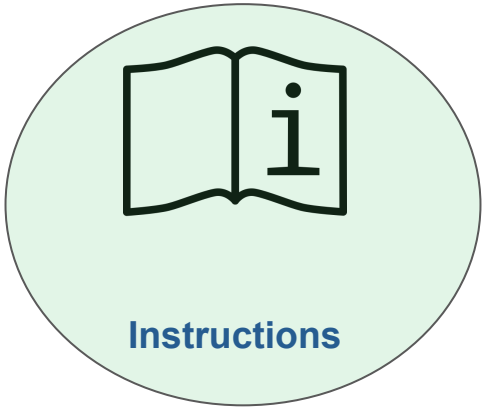
Q: "think, machine, learning"
A: "think, machine" outputs "ke". The last letter of "learning" is "g". Concatenating "ke", "g" leads to "keg". So, "think, machine, learning" outputs "keg".

Table 2: A test case of least-to-most prompting for the last-letter-concatenation task. Generated with `code-davinci-002` in GPT-3. The prompt context is shown on the right column of Table 1.

Method	L = 4	L = 6	L = 8	L = 10	L = 12
Standard prompting	0.0	0.0	0.0	0.0	0.0
Chain-of-Thought	89.4	75.0	51.8	39.8	33.6
Least-to-Most	**94.0**	**88.4**	**83.0**	**76.4**	**74.0**

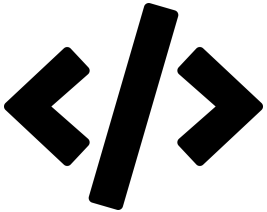
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration: Spell out the reasoning process before generating the answer

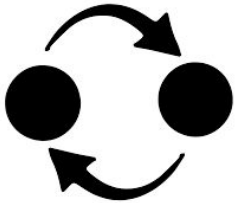


Instructions

Tool Augmentation



Structured Generation



Feedback

Memory

Instructions

- Answer the following question: Does water exist in a liquid state at room temperature?
  - Answer the following question: Does water exist in a liquid state at room temperature? Choose yes or no.

- Write a summary of the history of computer science
  - Summarize the history of computer science, highlighting key developments.
  - Provide a summary of the history of computer science in this format:
    - Inventors and innovations
    - Evolution of programming languages
    - Impact of internet
  - Summarize the history of computer science in no more than 200 words, keeping the language simple and easy to understand
  - Assume you're writing for a tech blog aimed at beginners. Summarize the history of computer science in a way that sparks interest in the subject.

Same end goal – different phrasing

- Implicit vs. Explicit
- Abstract vs. Concrete

Reframing Instructional Prompts to GPTk's Language (Mishra et al., 2022)

	Raw task definitions and their reframed counterpart
PATTERN REFRAMING	*Raw Task: Craft a question which requires commonsense to be answered. Based on the given context, craft a common-sense question, especially those that are LONG, INTERESTING, and COMPLEX. The goal is to write questions that are easy for humans and hard for AI machines! To create such questions, here are some suggestions: A. What may (or may not) be the plausible reason for an event? B. What may (or may not) happen before (or after, or during) an event? C. What may (or may not) be a plausible fact about someone (or something)? D. What may (or may not) happen if an event happens (or did not happen)? You can also create other types of questions.* **Input**: Context:<> **Expected Output**: Question:<>
	Reframed Task: Use 'what may happen', 'will ...?', 'why might', 'what may have caused', 'what may be true about', 'what is probably true about', 'what must' and similar phrases in your question based on the input context. **Input**: Context:<> **Expected Output**: Question:<>

Instructions

RESTRAINING

REFRAMING

Raw Task: *... What is the type of the answer corresponding to the given question? Number, Date, or Span?...*
Input: Passage: <>. Question: <> **Expected Output**: <Number/Date/Span> ...

Reframed Task: *... What is the type of the answer corresponding to the given question? Number, Date, or Span?...*
Input: Passage: <> Question: <> *Answer either Number, Date or Span?* **Expected Output**:<Number/Date/Span>

Mishra, Swaroop, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. "Reframing Instructional Prompts to GPTk's Language." *ACL 2022 Findings*.

Instructions

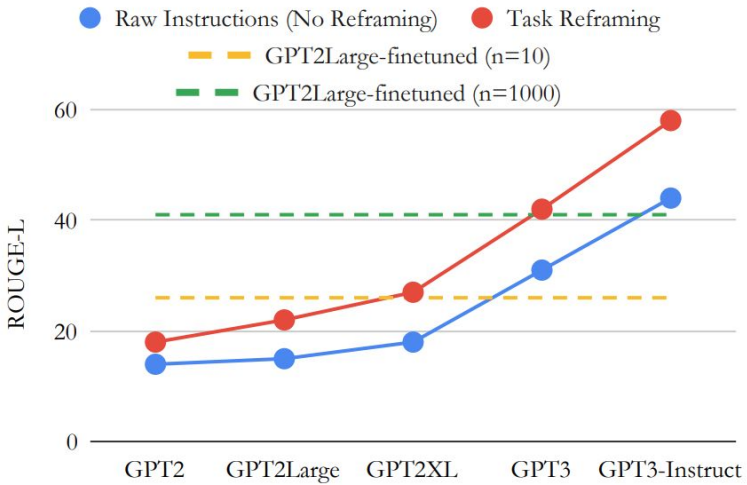


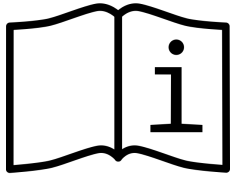
Figure 2: Across a variety of model sizes, reframed prompts consistently show considerable performance gain over raw task instructions (no reframing) in a few-shot learning setup. Since fine-tuning GPT3 is

- Explicit is better than implicit
- Be aware of instances where you might be expecting the model to read your mind!

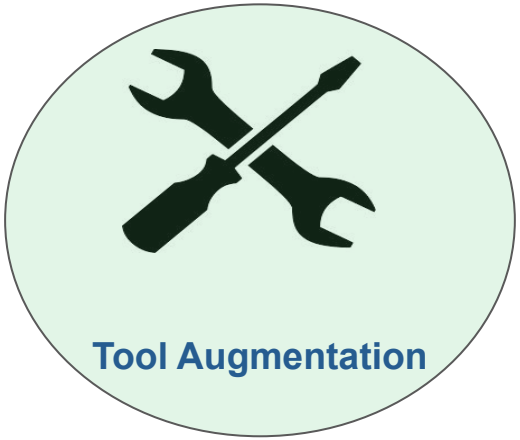
Mishra, Swaroop, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. "Reframing Instructional Prompts to GPTk's Language." *ACL 2022 Findings*.

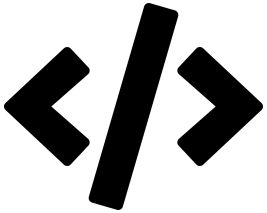
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration: Spell out the reasoning process before generating the answer

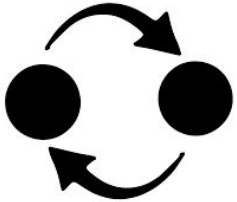


Instructions Provide explicit instructions to the LLM, capture all expectations





Structured Generation



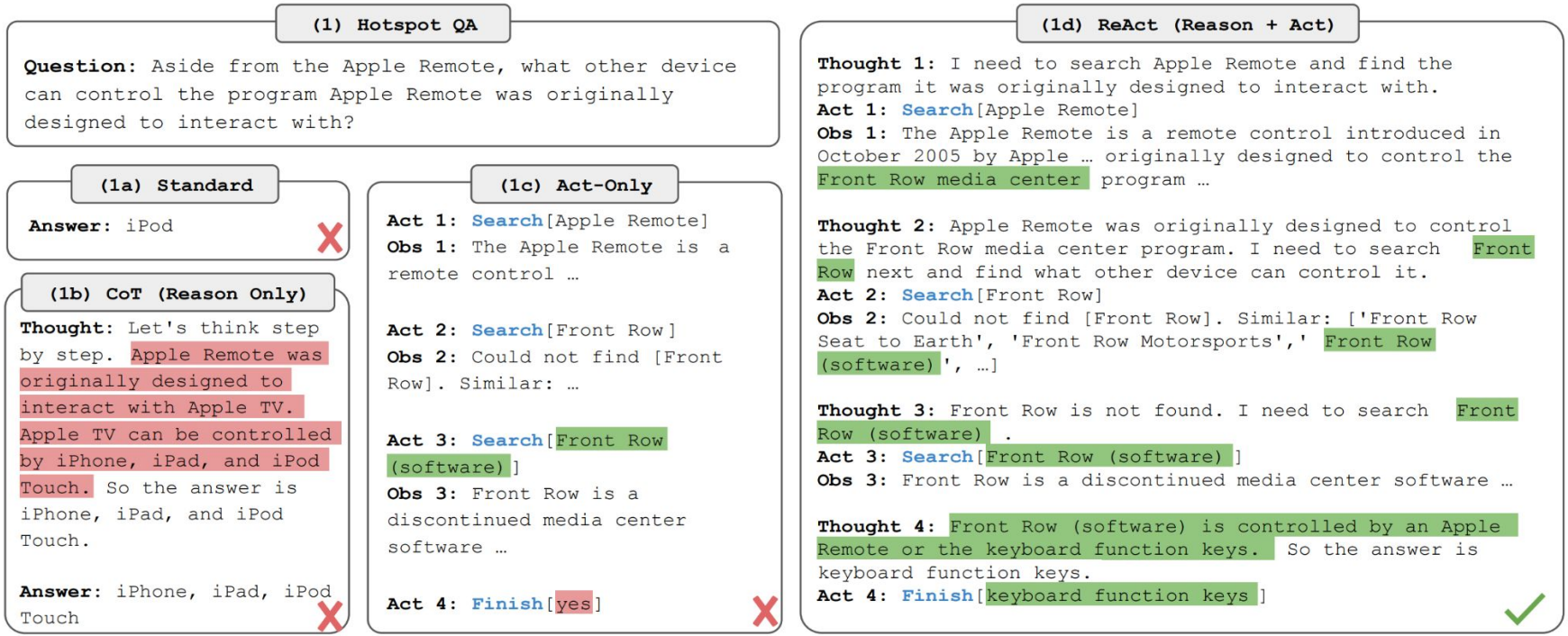
Feedback

Memory

Augmenting LLMs with Tools

- Typical language generation setup: LLMs handle *everything*
  - `There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5.`
    - *What has to be done?*
      - Add `3 + 2`
    - *How to do it?*
      - Perform the calculation and produce the answer `= 5`
- Leverage external tools to separate 'what' needs to be done from 'how' to do it.
- Calculator as a tool (T):
  - `There are originally 3 cars.`
  - `2 more cars arrive.`
  - `Number of cars after the first car comes = calculator(3, 1) = 4.`
  - `Number of cars after the second car comes = calculator(4, 1) = 5.`
  - `The answer is 5.`

React (Yao et al. 2022)



React (Yao et al. 2022)

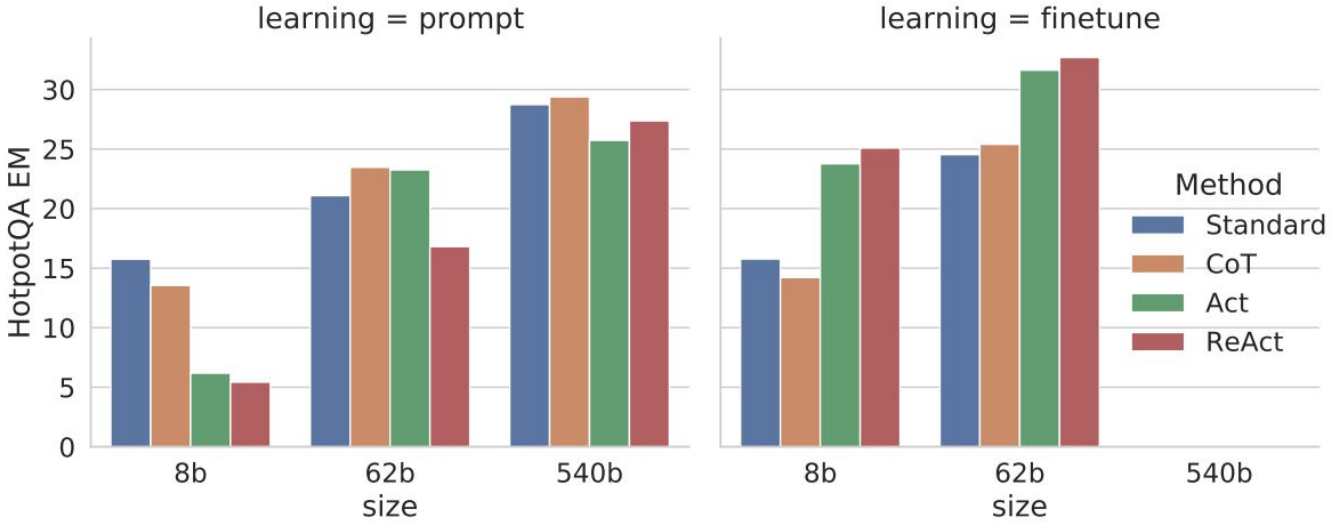


Figure 3: Scaling results for prompting and finetuning on HotPotQA with `ReAct` (ours) and baselines.

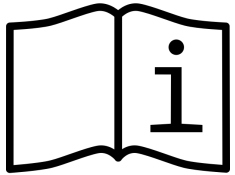
Augmenting LLMs with Tools

- Obvious use cases
  - **Real-time information:** Stock market updates, Temperature monitoring, Traffic reports.
  - **Specialization:** Solving Ordinary Differential Equations (ODEs)
  - **Multimodal**: Performing clicks, Generating image captions.

- Future Directions
  - Combine multiple tools within the same language model framework for increased functionality and adaptability.
  - [2304.09842] Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models
  - [2304.08354] Tool Learning with Foundation Models
  - [2302.04761] Toolformer: Language Models Can Teach Themselves to Use Tools
  - [2303.09014] ART: Automatic multi-step reasoning and tool-use for large language models

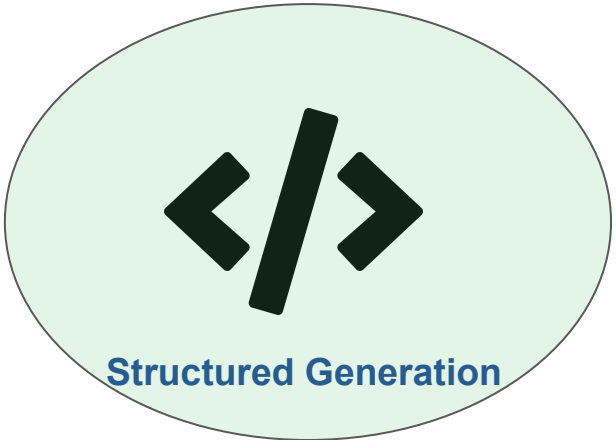
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration: Spell out the reasoning process before generating the answer

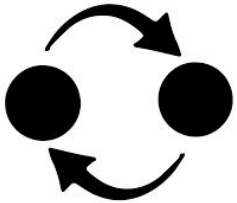


Instructions Provide explicit instructions to the LLM, capture all expectations

Tool Augmentation: Enhance LLMs for specialized tasks



Structured Generation



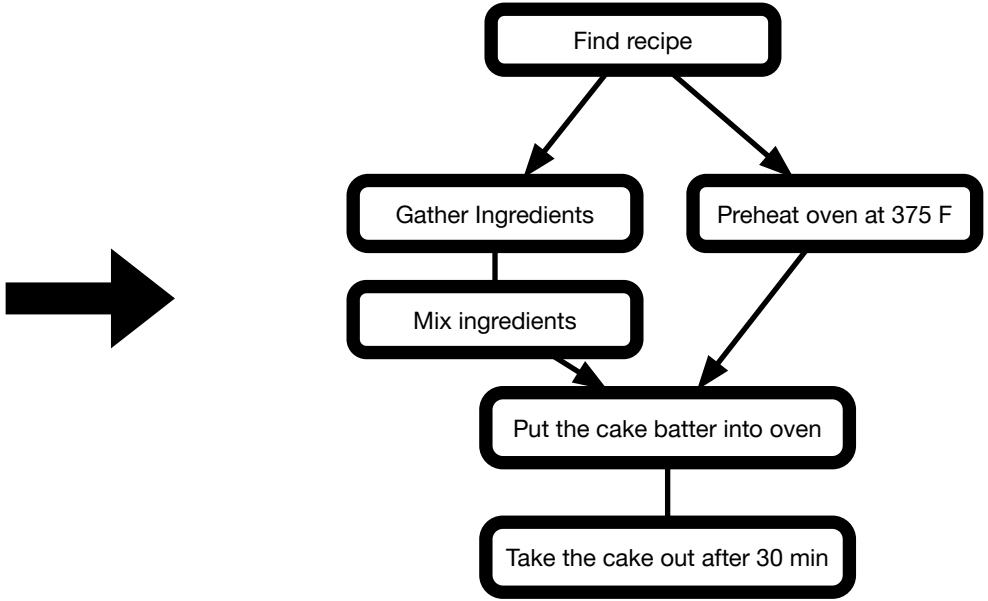
Feedback

Memory

Structured Commonsense Reasoning

- Natural language input (e.g., scenario)
- Structured output (e.g., plan graph, reasoning graph)

Bake a cake



https://proscript.allenai.org/

Leveraging Language Models for Structured commonsense Reasoning

- Need to generate a graph but … language models can only generate strings
- Workaround
  - *Flatten* the graph as a string
  - Train a seq2seq model

Goal: Bake a cake

```
"find recipe" -> "gather ingredients”;
"gather ingredients" -> "mix ingredients”;
"gather ingredients" -> "preheat oven at 375F”;
"find recipe" -> "preheat oven at 375F”;
"preheat oven at 375F" -> "put the cake batter into oven";
"mix ingredients" -> "put the cake batter into oven";
"put the cake batter into oven" -> "take the cake out after 30 min"
```

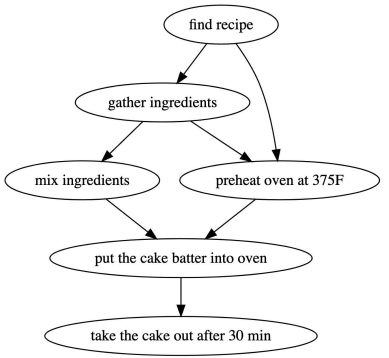
Intermediate Representation

Neural Language Modeling for Contextualized Temporal Graph Generation

Aman Madaan, Yiming Yang

proScript: Partially Ordered Scripts Generation

Keisuke Sakaguchi,[1] Chandra Bhagavatula,[1] Ronan Le Bras,[1]
Niket Tandon,[1] Peter Clark,[1] Yejin Choi[1,2]
[1]Allen Institute for Artificial Intelligence
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington

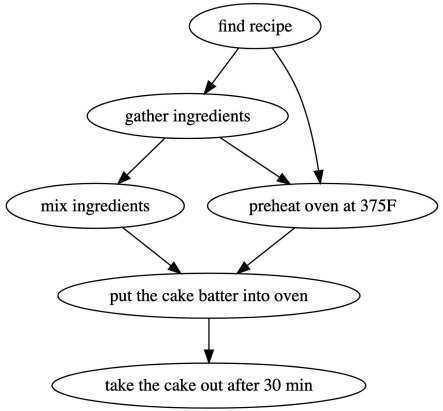


Recovered Graph

Leveraging Language Models for Structured commonsense Reasoning

- Issues with the workaround

"find recipe" → "gather ingredients";
“gather ingredients" → "mix ingredients”;
"find recipe" → "preheat oven at 375F”;
"preheat oven at 375F" → "put the cake batter into oven";
"mix ingredients" → "put the cake batter into oven";
"put the cake batter into oven" → "take the cake out after 30 min"



- We want ***structures, not strings***

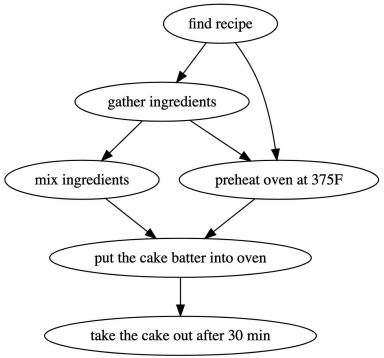
⁉ Are the two mix ingredients the same?

Language Models of Code Are Few-shot Commonsense Learners (Madaan et al. 2022)

Goal: Bake a cake

```
class BakeACake:
    def __init__(self) -> None:
        self.find_recipe = Node()
        self.gather_ingredients = Node()
        self.mix_ingredients = Node()
        self.find_recipe = Node()
        self.preheat_oven_at_375f = Node()
        self.put_cake_batter_into_oven = Node()
        self.take_cake_out_after_30_min = Node()

        self.find_recipe.children = [self.gather_ingredients,
self.preheat_oven_at_375f]
        self.gather_ingredients.children = [self.mix_ingredients]
        self.mix_ingredients.children = [self.put_cake_batter_into_oven]
        self.preheat_oven_at_375f.children =
[self.put_cake_batter_into_oven]
        self.put_cake_batter_into_oven.children =
[self.take_cake_out_after_30_min]
```

Intermediate Representation



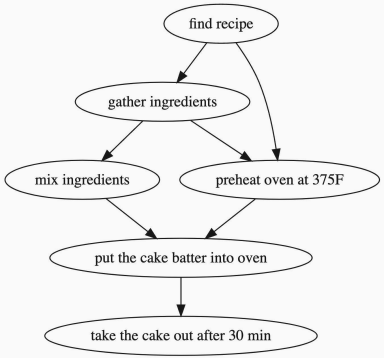
Recovered Graph

Language Models of Code Are Few-shot Commonsense Learners (Madaan et al. 2022)

Goal: Bake a cake

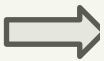
```
"find recipe" -> "gather ingredients";
"gather ingredients" -> "mix ingredients";
"gather ingredients" -> "preheat oven at 375F";
"find recipe" -> "preheat oven at 375F";
"preheat oven at 375F" -> "put the cake batter into oven";
"mix ingredients" -> "put the cake batter into oven";
"put the cake batter into oven" -> "take the cake out after 30 min"
```

Intermediate Representation



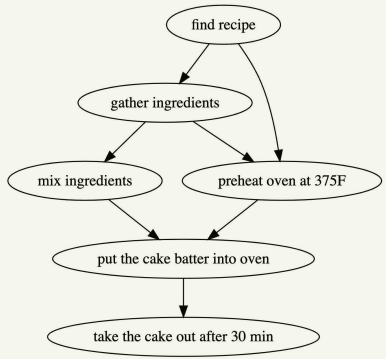
Recovered Graph

Goal: Bake a cake



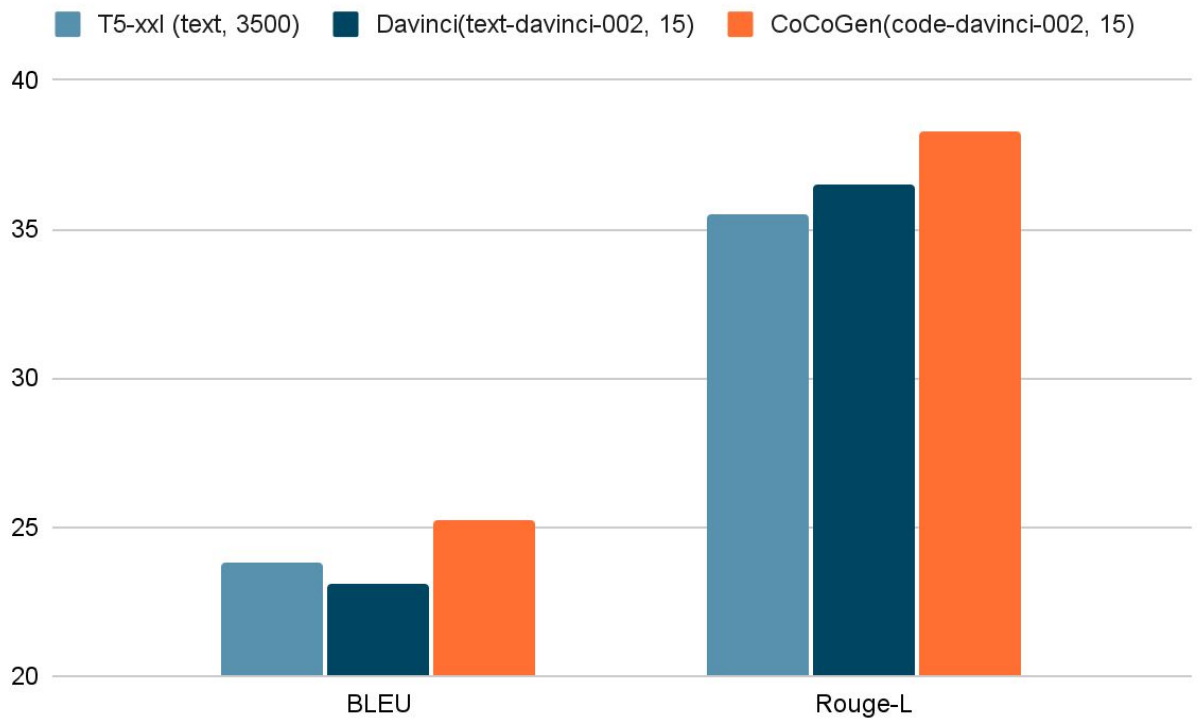
```
class BakeACake:
    def __init__(self) -> None:
        self.find_recipe = Node()
        self.gather_ingredients = Node()
        self.mix_ingredients = Node()
        self.find_recipe = Node()
        self.preheat_oven_at_375f = Node()
        self.put_cake_batter_into_oven = Node()
        self.take_cake_out_after_30_min = Node()

        self.find_recipe.children = [self.gather_ingredients,
self.preheat_oven_at_375f]
        self.gather_ingredients.children = [self.mix_ingredients]
        self.mix_ingredients.children = [self.put_cake_batter_into_oven]
        self.preheat_oven_at_375f.children =
[self.put_cake_batter_into_oven]
        self.put_cake_batter_into_oven.children =
[self.take_cake_out_after_30_min]
```

Intermediate Representation



Recovered Graph

Script Generation Results on ProScript



PAL: Program-aided Language Models (Gao*, Madaan*, Zhou* et al. 2023)

- Olivia has $23. She bought five bagels for $3 each. How much money does she have left?

Olivia had 23 dollars. 5 bagels for 3 dollars each will be dollars. So she has dollars left.

CoT

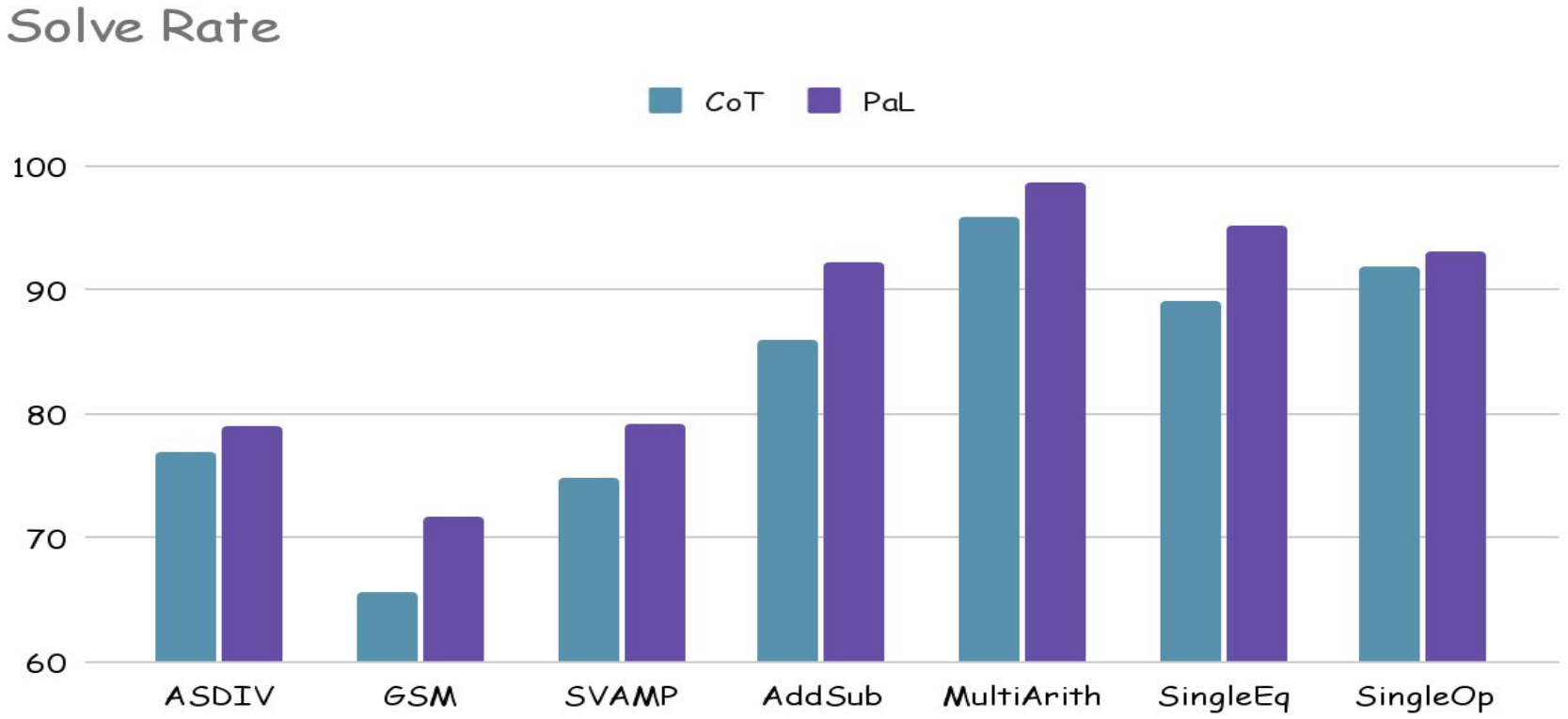
```
def solution():
    money_initial = 23
    bagels = 5
    bagel_cost = 3
    money_spent = bagels * bagel_cost
    money_left = money_initial - money_spent
    result = money_left
    return result
```

PaL

Comparison with CoT:

- The language model is responsible for generating a high-level plan that is executed to derive the answer
- The results are obtained after running the program

Improves Solve Rate for Multiple Maths Reasoning Tasks

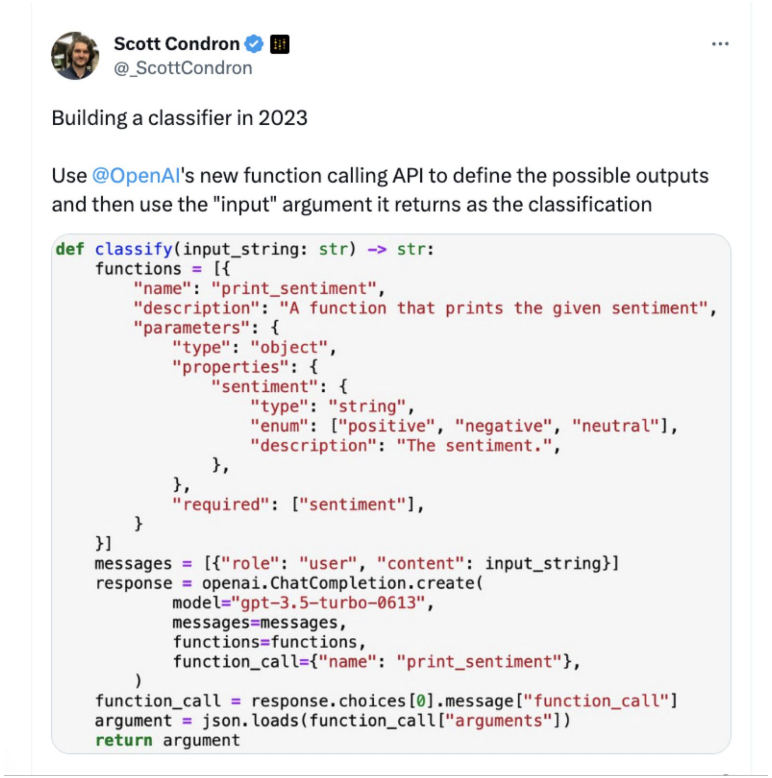


Few-shot SOTA

Why should code help?

```
class BakeACake:
    def __init__(self) -> None:
        self.find_recipe = Node()
        self.gather_ingredients = Node()
        self.mix_ingredients = Node()
        self.find_recipe = Node()
        self.preheat_oven_at_375f = Node()
        self.put_cake_batter_into_oven = Node()
        self.take_cake_out_after_30_min = Node()

        self.find_recipe.children = [self.gather_ingredients,
self.preheat_oven_at_375f]
        self.gather_ingredients.children = [self.mix_ingredients]
        self.mix_ingredients.children = [self.put_cake_batter_into_oven]
        self.preheat_oven_at_375f.children =
[self.put_cake_batter_into_oven]
        self.put_cake_batter_into_oven.children =
[self.take_cake_out_after_30_min]
```

Structured Generation



Scott Condron
@_ScottCondron

Building a classifier in 2023

Use @OpenAI's new function calling API to define the possible outputs and then use the "input" argument it returns as the classification

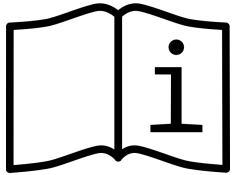
```
def classify(input_string: str) -> str:
    functions = [{
        "name": "print_sentiment",
        "description": "A function that prints the given sentiment",
        "parameters": {
            "type": "object",
            "properties": {
                "sentiment": {
                    "type": "string",
                    "enum": ["positive", "negative", "neutral"],
                    "description": "The sentiment.",
                },
            },
            "required": ["sentiment"],
        }
    }]
    messages = [{"role": "user", "content": input_string}]
    response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo-0613",
            messages=messages,
            functions=functions,
            function_call={"name": "print_sentiment"},
        )
    function_call = response.choices[0].message["function_call"]
    argument = json.loads(function_call["arguments"])
    return argument
```

https://twitter.com/_ScottCondron/status/1670827747684364288

[More on future directions]

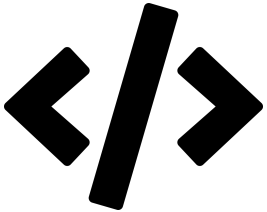
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration: Spell out the reasoning process before generating the answer

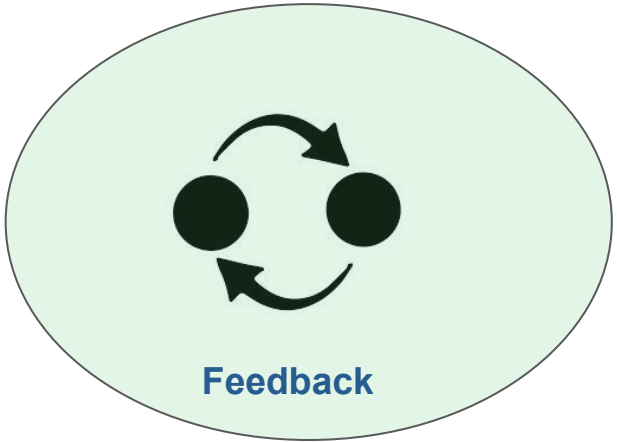


Instructions Provide explicit instructions to the LLM, capture all expectations

Tool Augmentation: Enhance LLMs for specialized tasks



Structured Generation: Use structure to guide model



Memory

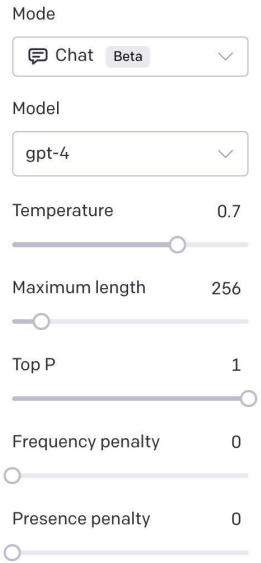
Feedback

- Humans typically generate content iteratively:
  - Generate → Feedback → Refine

- LLMs, with the right prompt, have recently shown promise at mimicking this process
  - Generate an output
  - Gather feedback
  - Improve output with feedback, repeat

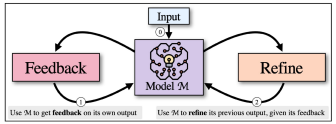
Self-Refine (Madaan et al. 2023)

USER	Give me an acronym for a paper titled "Politeness transfer: A tag and generate approach" Make it easy to pronounce. Relevant to the title.
ASSISTANT	PTAGA (Puh-TAG-uh)
USER	That is not a great acronym either by relevance or pronunciation
ASSISTANT	My apologies for that. Here's another acronym: P-TAG (Pee-Tag)
USER	Same problem!
ASSISTANT	Apologies for the confusion. Here's another attempt: TRAPG (Transferring Respect and Politeness: Generate Approach) which can be pronounced as "trap-gee".



GPT-3.5/4 is good!
But often requires a few rounds of talking through the problem

Let LLM Self-Refine



Generate Acronym for: *Politeness transfer: A tag and generate approach*

Model $\mathcal{M}$

GENERATE

PTATAGA

Model $\mathcal{M}$

FEEDBACK

Model $\mathcal{M}$

REFINE

POLTAG

Model $\mathcal{M}$

FEEDBACK

Relevance of PTATAGA: The acronym includes the first letters of each major word in the title but doesn't fully capture the meaning of the title. <3/5>

Memorability of PTATAGA: The acronym is not the most memorable. <2/5>

Pronunciation of PTATAGA: The acronym is pronounceable but not very fluid. <3/5>

Relevance of POLTAG: A tag and generate approach: The acronym POLTAG is a good representation of the title as it combines the first few letters of Politeness and Tag. <4/5>

Memorability of POLTAG: The acronym POLTAG is easy to remember as it is short, simple, and has a clear structure. <4/5>

→ **STOP**

Pronunciation of POLTAG: The acronym is easily pronounced as it follows the common structure of acronyms. <5/5>

Results

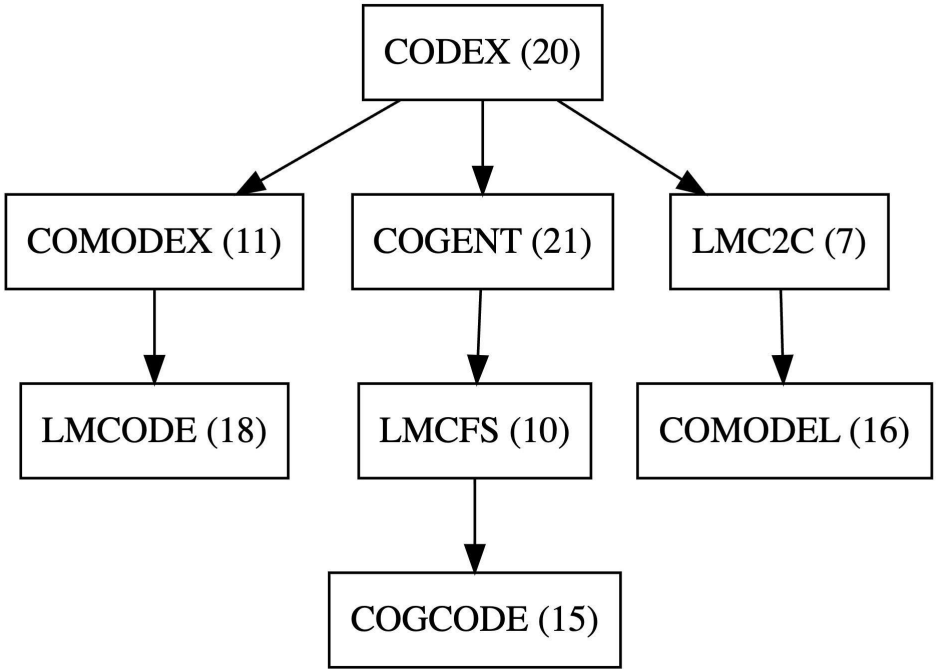
	GPT-3.5		ChatGPT		GPT-4	
Task	Base	+SELF-REFINE	Base	+SELF-REFINE	Base	+SELF-REFINE
Sentiment Reversal	8.8	**30.4** (↑21.6)	11.4	**43.2** (↑31.8)	3.8	**36.2** (↑32.4)
Dialogue Response	36.4	**63.6** (↑27.2)	40.1	**59.9** (↑19.8)	25.4	**74.6** (↑49.2)
Code Optimization	14.8	**23.0** (↑8.2)	23.9	**27.5** (↑3.6)	27.3	**36.0** (↑8.7)
Code Readability	37.4	**51.3** (↑13.9)	27.7	**63.1** (↑35.4)	27.4	**56.2** (↑28.8)
Math Reasoning	**64.1**	**64.1** (0)	74.8	**75.0** (↑0.2)	92.9	**93.1** (↑0.2)
Acronym Generation	41.6	**56.4** (↑14.8)	27.2	**37.2** (↑10.0)	30.4	**56.0** (↑25.6)
Constrained Generation	28.0	**37.0** (↑9.0)	44.0	**67.0** (↑23.0)	15.0	**45.0** (↑30.0)

Table 1: SELF-REFINE results on various tasks using GPT-3.5, ChatGPT, and GPT-4 as base LLM. SELF-REFINE consistently improves LLM. Metrics used for these tasks are defined in Section 3.2.

- Larger gains where feedback is obvious + easy
- Lots of room for improvement!

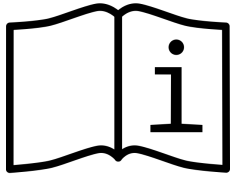
Future Directions: Feedback + Planning

- Monte-Carlo Tree Search + Self-Refine
  - Extend Self-Refine Beyond Linear Search
- Other exciting work:
  - Tree-of-thought
  - Reflexion
  -



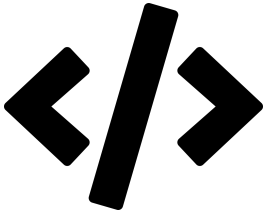
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration: Spell out the reasoning process before generating the answer

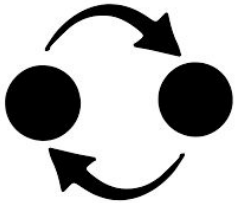


Instructions Provide explicit instructions to the LLM, capture all expectations

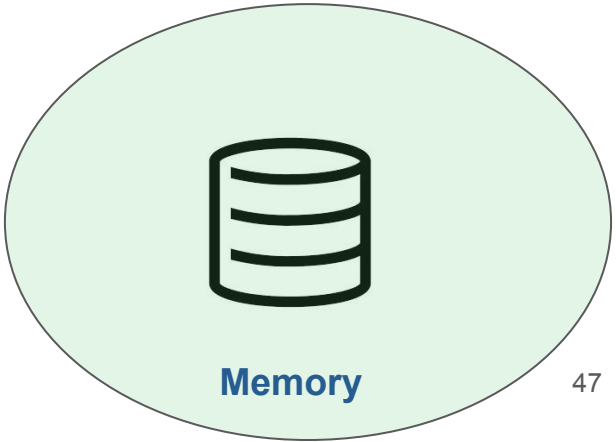
Tool Augmentation: Enhance LLMs for specialized tasks



Structured Generation: Use structure to guide model



Feedback: Refine model outputs during inference



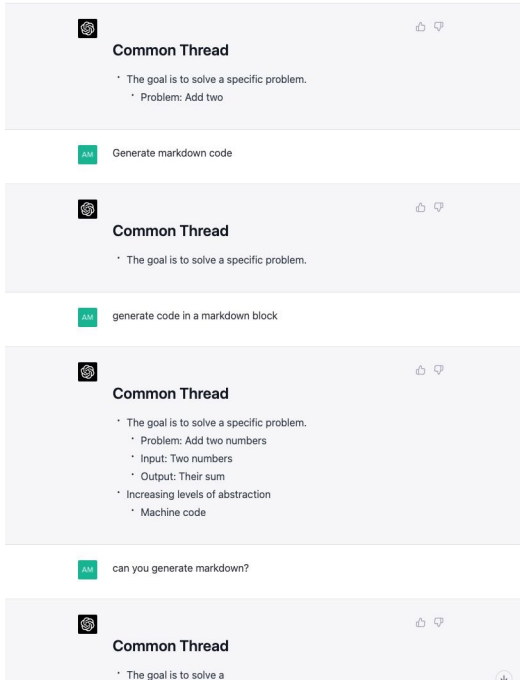
Memory

- Standard few-shot prompting setup
  - Prompt P: $[X_i, Y_i]$
  - Test examples X_1, X_2
    - $P + X_1 \rightarrow Y'_1$
    - $X_2 \rightarrow Y'_2$

- Idea: maintain a memory of examples seen so far, and any feedback

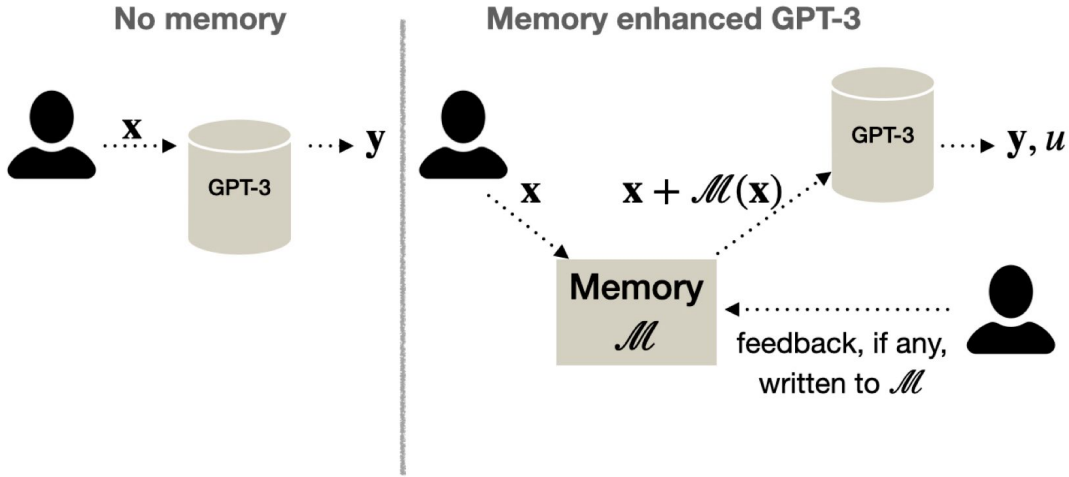
MemPrompt (Madaan et al. 2022)

Models repeat mistakes



GPT-4

MemPrompt


No memory
x
GPT-3
y
Memory enhanced GPT-3
x
x + 𝓜(x)
GPT-3
y, u
Memory
𝓜
feedback, if any,
written to 𝓜

Workflow

- **Step 1:** User asks a question
- **Step 2:** Check if the same question has been asked before, and a clarification is present in memory
  - Step 2.1: If a clarification is present, add question + clarification
  - Step 2.2: If not, just ask a question
- **Step 3:** Model generates an answer
- **Step 4:** Take clarification on answer if needed, add clarification to memory

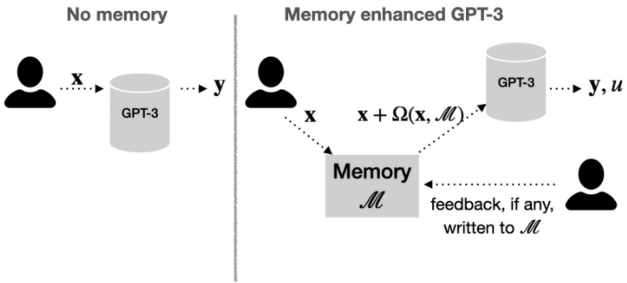
Question	Feedback
A word pronounced as fellow ?	I want a word that sounds similar!
What is dissimilar to delicious ?	Give me the reverse of delicious
What is a word like great ?	**Wrong! I want something similar** ✓
How do I use melancholy ?	No…I wanted a sample sentence
What is on the lines of pretty ?	I was looking for a similar word
Could you expand on browser ?	I actually wanted a definition

1. *Query the memory*

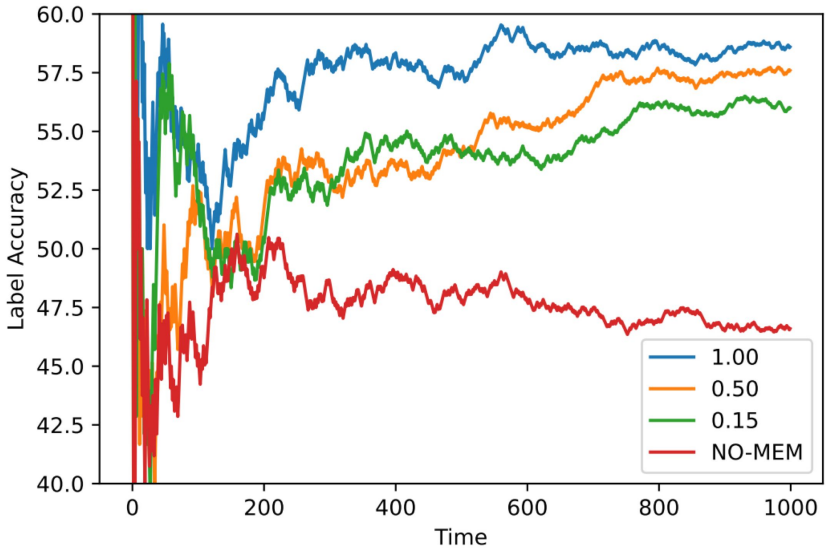
2. *Retrieve relevant feedback*

q : What is akin to quick ?

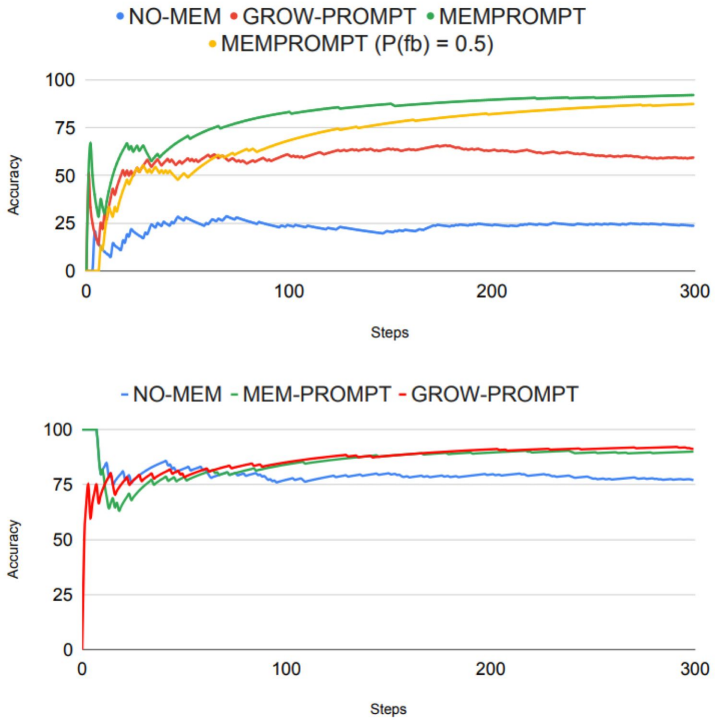
fb: Wrong! when I mention like, I want something similar



Memprompt Results



Ethical Reasoning with Categorical Feedback



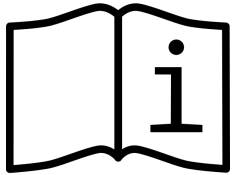
Lexical tasks (top) and word scrambling tasks (bottom)

Memory for Few-shot Prompting

- Simpler but effective variants
  - KATE: Store the training set in a database, retrieve most relevant examples on the fly

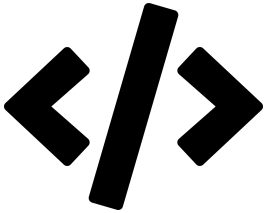
Complex Few-shot Reasoning with LLMs: Key Techniques

Reasoning Elaboration: Spell out the reasoning process before generating the answer

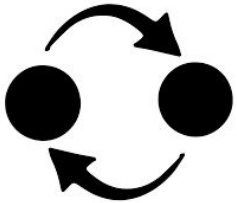


Instructions Provide explicit instructions to the LLM, capture all expectations

Tool Augmentation: Enhance LLMs for specialized tasks



Structured Generation: Use structure to guide model

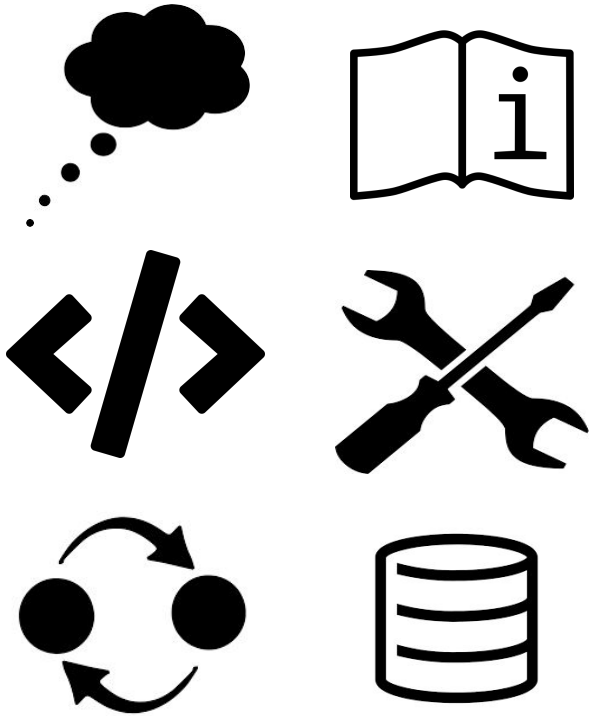


Feedback: Refine model outputs during inference

Memory: maintain a history of interactions with LLM

Future Directions

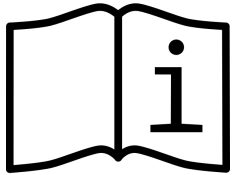
- LLMs are getting better at following instructions, exciting new possibilities
    - Planning, Search – resurgence of classical AI techniques
    - Generating actionable feedback from tools
    - Storing a memory of interactions across multiple users
    - Enhanced Structured Generation Capabilities



Complex Few-shot Reasoning with LLMs: Key Techniques

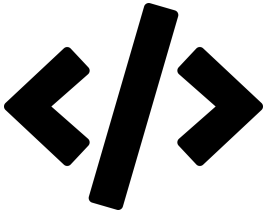
Questions?

- **Reasoning Elaboration: Spell out the reasoning process before generating the answer**

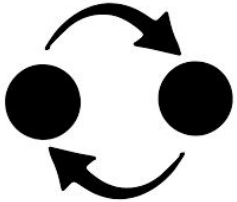


- **Instructions Provide explicit instructions to the LLM, capture all expectations**

- **Tool Augmentation: Enhance LLMs for specialized tasks**



- **Structured Generation: Use structure to guide model**

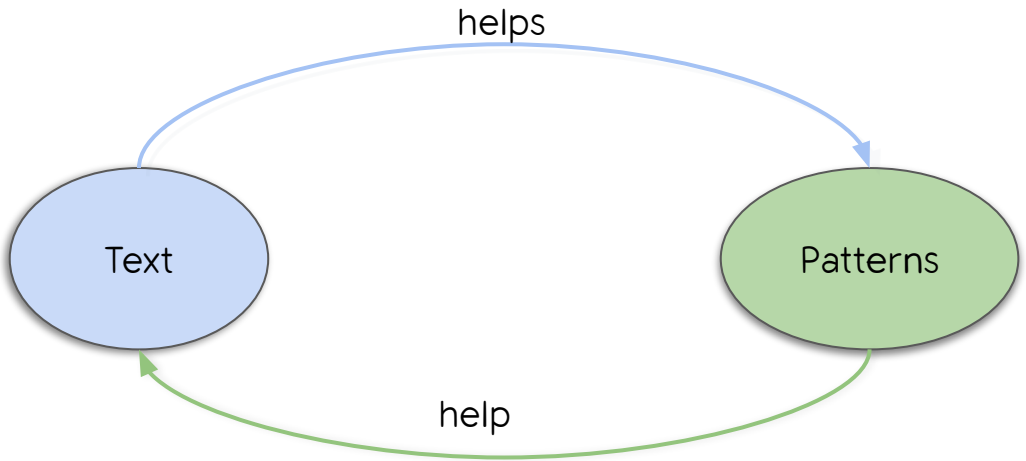


- **Feedback: Refine model outputs during inference**

- **Memory: maintain a history of interactions with LLM**

Appendix

Text and Patterns: For Effective Chain of Thought, It Takes Two to Tango



Aman Madaan* **and Amir Yazdanbakhsh**♦
Carnegie Mellon University ♦Google Research, Brain Team
amadaan@cs.cmu.edu, ayazdan@google.com
(Equal Contribution)

Paper

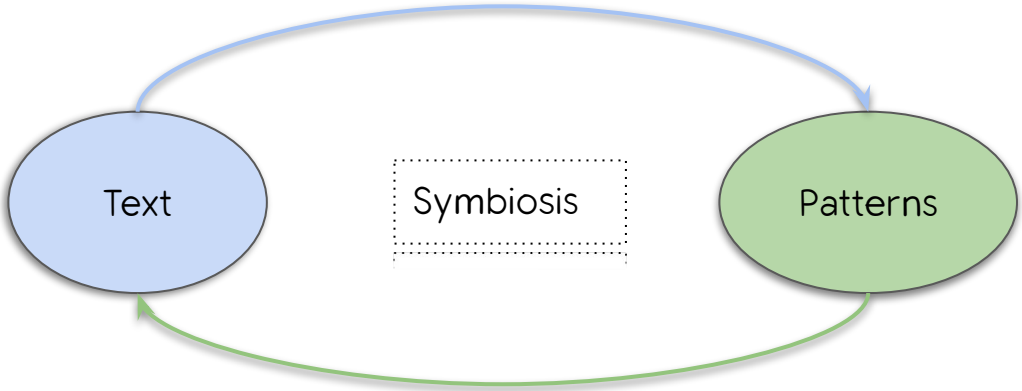
*Work done when Aman was a student researcher at Google Research, Brain Team.

What makes the chain of thought prompting so effective?

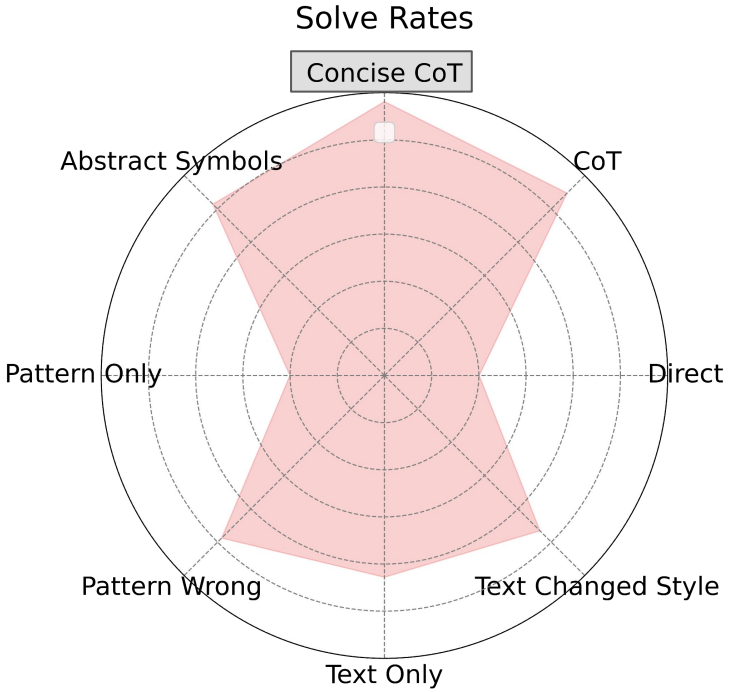
- *The thought makes the model think about the problem?*
- *The thought helps the model learn better*
- *The thought serves as an additional example of the task*
- *The thought helps the model remind of the task*
- *The thought helps extract relevant information for solving the task*

What makes chain of thought prompting so effective

Text: Extracting common sense, copy mechanism



Patterns: Task understanding, final answer generation



Approach

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

T: There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5.

A: The answer is 5 cars.

Symbols | Patterns | Text

- ***Counterfactual prompting:***
  - Change one *knob* at a time (symbol, patterns, text)

What if? prompting (counterfactual prompting)

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

Thought (T): There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5.

A: The answer is 5 cars.

What if we don't have actual numbers?

Q: If there are α cars in the parking lot and β more cars arrive, how many cars are in the parking lot?

Thought (T): There are originally α cars. β more cars arrive. α + β = λ.

A: The answer is λ cars.

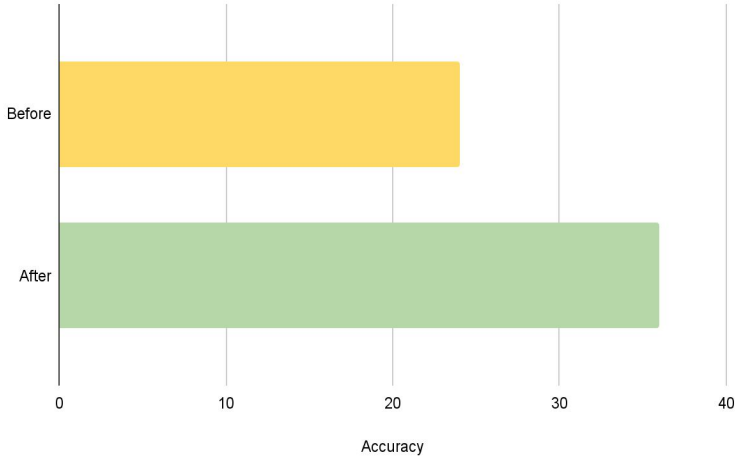
What if the prompt is misleading?

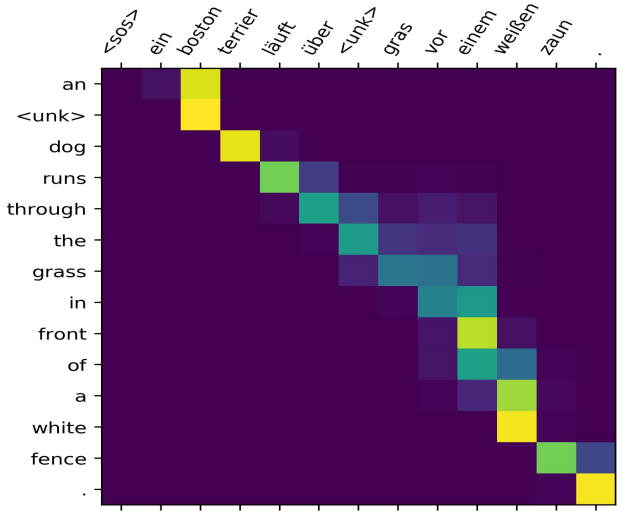
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

Thought (T): There are originally 3 cars. 2 more cars arrive. 3 + 2 = 7.

A: The answer is 5 cars.

Evaluating counterfactual prompts





Empirical results: difference in final <u>outcome</u>

Attention patterns: difference in <u>mechanism</u>

Complex Reasoning with LLMs

- Complex Reasoning with LLMs
- Few-shot prompting
  - Why does it work?
- Augmenting Complex Reasoning
  - **Better Chain-of-thought**
    - **Least2Most, Algorithmic Prompt, Self-ask**
  - Programs
    - PaL, ReACT
  - Memory
    - Memprompt, KATE
  - Structures
    - Tree-of-thought
  - Sampling
    - Re-prompting

Complex Reasoning with LLMs

- Few-shot prompting
  - Why does it work?
- Class of Techniques for Few-shot Prompting
  - Decomposition
    - Least2Most, Algorithmic Prompt, Self-ask, Complex Prompting
  - Code-Assisted Reasoning
    - PaL, PoT, ReACT
  - Structures
  -
    - Memory: Memprompt, KATE
    - Trees: Tree-of-thought
  - Sampling
    - Re-prompting, Self-Refine