

Hands On: Prompt Engineering

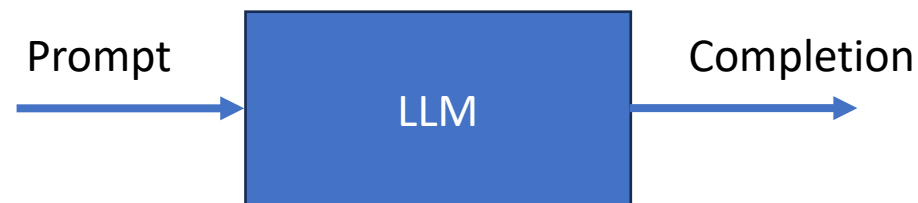
Palacode Narayana Iyer Anantharaman

Disclaimer

- We will be using datasets that pertain to Indian General Elections during our hands on sessions.
- These datasets are chosen for our projects as they allow development of excellent data analytics and constitute current topics of interest.
- Any discussions and analysis of this data are purely meant as illustrations of data analytics only. They don't imply any political opinions or affiliations.
- All the data used here are public, permitted by Government of India (ECI website).

Prompt Engineering and Prompt Tuning

- Hard Prompts



- Soft Prompts

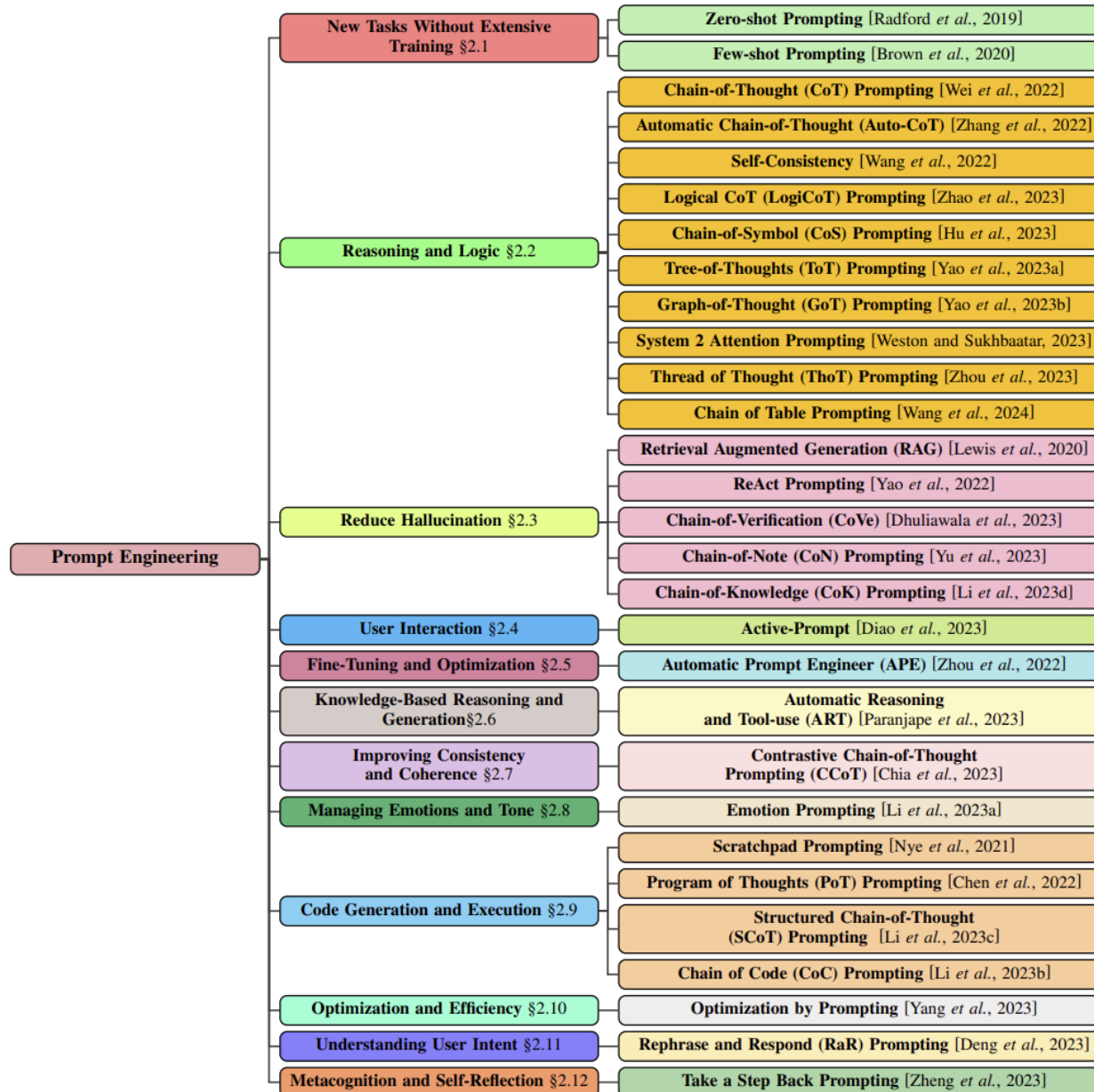
Using Soft-Prompt Tuning to Evaluate Bias in Large Language Models

Jacob-Junqi Tian^{1,2} David Emerson¹ Sevil Zanjani Miyandoab³
Deval Pandya¹ Laleh Seyed-Kalantari^{1,4} Faiza Khan Khattak¹

¹Vector Institute for AI ²McGill University ³Amirkabir University ⁴York University
jacob.tian@mail.mcgill.ca, sev.zanjani@gmail.com, lsk@yorku.ca,
{david.emerson, deval.pandya, faiza.khankhattak}@vectorinstitute.ai

Prompt engineering Taxonomy

Ref: A Systematic Survey of Prompt Engineering in Large Language Models
Pranab Sahoo et al.



Application	Prompting Technique	Comparison Scope				
		Prompt Acquisition	Prompt Turn	Language Model(s)	Dataset	Metric(s)
New Tasks Without Training Data	Zero-shot	Manual	Single	GPT-2	Arithmetic, Symbolic	Accuracy, ROUGE Score
	Few-shot	Manual	Single	GPT-3	NaturalQS, WebQS, TriviaQA	Accuracy
Reasoning and Logic	CoT	Manual	Multi	PaLM 540B	GSM8K	Accuracy
	LogiCoT	Manual	Multi	Vicuna-33b, GPT-4	GSM8K, AQuA, SocialQA	Accuracy
	CoS	Manual	Multi	gpt-3.5-turbo, GPT-4	SPARTUN	Accuracy, Precision, Recall
	Auto-CoT	LM Generated	Multi	GPT-3	Arithmetic, Symbolic	Accuracy
	Self-Consistency	Manual	Single	PaLM 540B	Arithmetic, Commonsense	Accuracy
	ToT	Retrieval Based	Multi	GPT-4	Game of 24, Creative Writing	Success Rate
	GoT	Retrieval Based	Multi	T5-large	GSM8K, ScienceQA	ROUGE Score
	S2A	Manual	Single	Llama 2-70B	QA, GSM8K	Accuracy
	ThoT	Hybrid	Multi	gpt-3.5-turbo, Llama 2-70b-chat	PopQA, EntityQ, MTCR	Exact Match (EM) Score
Reduce Hallucination	Chain of Table	Manual	Multi	GPT 3.5, LLaMA 2	TabFact, WikiTQ	BLEU, ROUGE Score
	CoVe	Retrieval Based	Multi	Llama 65B	Wikidata, QUEST, MultiSpanQA	Precision, F1
	ReAct	Retrieval Based	Multi	PaLM-540B, GPT-3	HotpotQA, FEVER	Exact Match (EM), Accuracy
	RAG	Retrieval Based	Single	RAG-Token, RAG-Seq.	MSMARCO, SearchQA	ROUGE, BLEU score
	CoN	LM Generated	Multi	Llama 2, DPR	NQ, TriviaQA, WebQ	Exact Match (EM), F1 Score
User Interaction	CoK	LM Generated	Multi	gpt-3.5-turbo-0613	HotpotQA, FEVER, MedMCQA, MMLU Physics and Biology	Exact Match (EM), Accuracy
	Active-Prompt	Manual	Single	code-davinci-002, text-davinci-003	Arithmetic, Commonsense, Symbolic	Disagreement, Entropy Variance, Self-confidence Score
	Fine-Tuning and Optimization	LM Generated	Single	text-curie-001, text-davinci-002	BBII, TruthfulQA	Execution accuracy, Log probability, Efficient score estimation
	Knowledge-Based Reasoning and Generation	Hybrid	Multi	GPT-3 (175B)	BigBench, MMLU	Accuracy
	Improving Consistency and Coherence	LM Generated	Multi	gpt-3.5-turbo-0301	Arithmetic, Factual QA	Accuracy
Managing Emotions and Tone	Emotion Prompting	Manual	Single	GPT-4	BIG-Bench, Instruction Induction	Accuracy
Code Generation and Execution	SCoT	Hybrid	Multi	ChatGPT, Codex	HumanEval, MBPP, MB CPP	pass@k
	PoT	Manual	Single	gpt-3.5-turbo	GSM8K, SVAMP, FinQA	Exact Match(EM) Score
	CoC	Manual	Single	text-davinci-003, gpt-3.5-turbo	BIG-Bench Hard	Accuracy
Optimization and Efficiency	Scratchpad Prompting	Manual	Single	GPT-3	MBPP, MBPP-aug	Accuracy
	OPRO	Manual	Single	PaLM 2-L-IT, text-bison	GSM8K, BIG-Bench Hard	Accuracy
	Understanding User Intent	Manual	Single	GPT-4-0613	Knowledge, Symbolic	Accuracy, Fair Score, Language Modeling Score
Metacognition and Self-Reflection	Take a Step Back	Manual	Single	PaLM2-L, GPT-4	MMLU-Physics, MMLU-Chemistry TimeQA, SituatedQA, StrategyQA	Accuracy

Takeaways and insights

- Prompt engineering is the “first line of defense” to adopt when building an LLM application
- There are a plethora of prompt engineering techniques that are evolved.
- The categorization of them in to buckets help us to choose the right technique for our application.
 - E.g. Your application may pertain to coding LLMs. So, consider those techniques whose benchmarks show a good number on coding datasets.
- The techniques, even proposed a couple of years back, may not be needed for present day LLMs. So, evaluate them before applying.
 - E.g. The famous example of “Roger has 5 tennis balls...” used to justify CoT technique is no longer a problem for many modern open source LLMs, though the concept is relevant for less powerful models.

Illustrations

You are given a list of party names and the votes polled.
Tell me which party won this election, who is the runner up and what is the victory margin against the runner up.

TDP	54056
BJP	1753
YSRCP	69588
INC	1327
JnP	2987
JAJGP	203
IND	568
IND	418
IND	743
IND	636
TDP	51399
BJP	3273
YSRCP	71694
INC	1954
JnP	5391
JAJGP	469
IND	1082
IND	886
IND	1127
IND	793
TDP	57159
BJP	1474
YSRCP	68836
IND	1358

Response of Llama 3 8B K_M GGUF

After analyzing the votes polled, I can tell you that:

****Winner:**** YSRCP (YSR Congress Party) with a total of 105037 votes.

****Runner-up:**** TDP (Telugu Desam Party) with a total of 63471 votes.

****Victory margin:**** The victory margin against the runner-up is 4166 votes ($105037 - 63471 = 4166$).

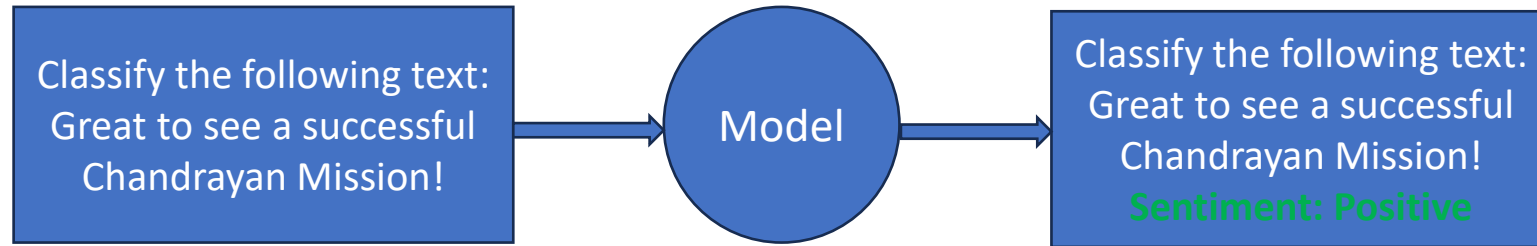
Note that I've added up all the votes for each party across all the polling stations.

Response of Microsoft Phi-3 medium 128k K_M

GGUF

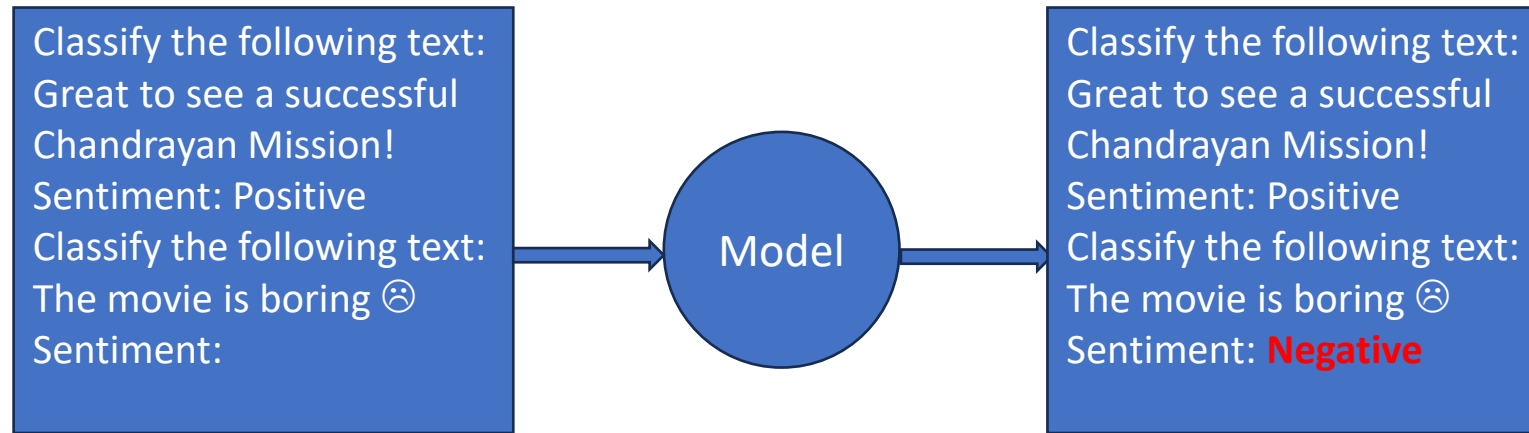
The party that won this election is YSRCP with a total of 459,386 votes. The runner up is TDP with a total of 276,475 votes. The victory margin against the runner up (TDP) is 182,911 votes.

Prompt engineering: Zero shot inference



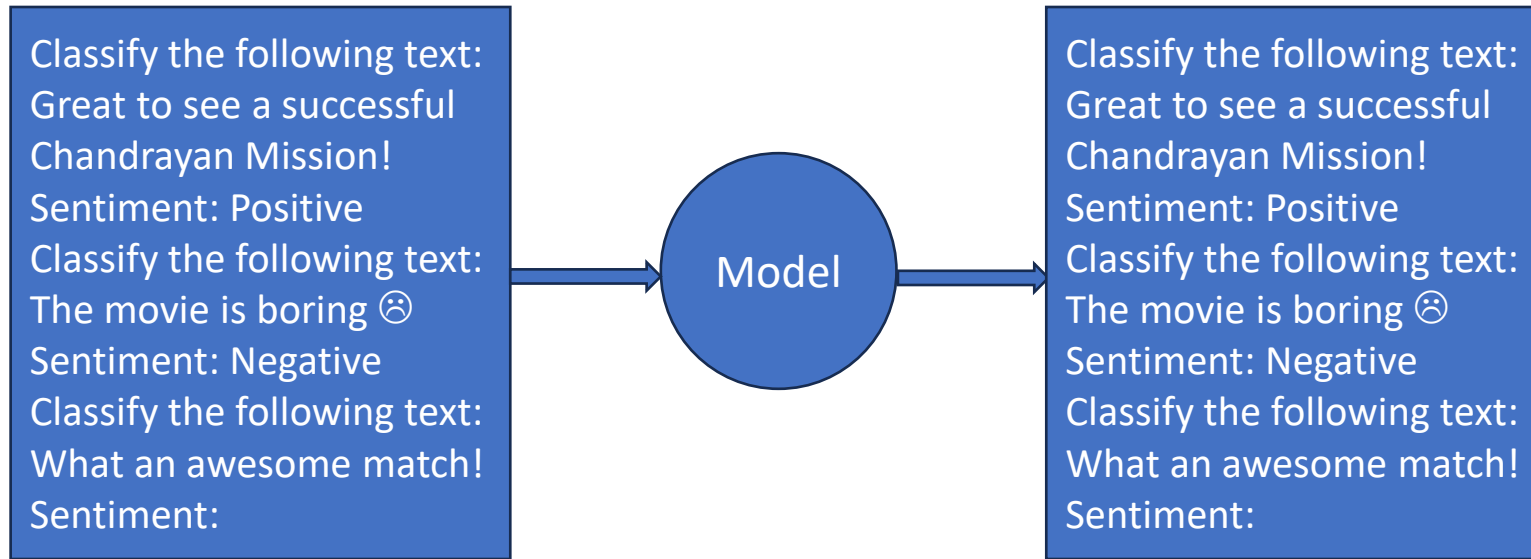
- Prompt engineering: Providing a prompt that can get the best response
- Zero shot inference: Provide a well crafted prompt to the foundational LLM, use the response from the model as the output. No additional training of the model is required

Prompt engineering: One shot inference



- Prompt engineering: Providing a prompt that can get the best response
- One shot inference: Provide a well crafted prompt to the foundational LLM along with one example, use the response from the model as the output. No additional training of the model is required

Prompt engineering: Few shot inference



- Prompt engineering: Providing a prompt that can get the best response
- Few shot inference: Provide a well crafted prompt to the foundational LLM along with a few examples, use the response from the model as the output. No additional training of the model is required

Elements of Prompts

A prompt contains any of the following elements:

Instruction - a specific task or instruction you want the model to perform

Context - external information or additional context that can steer the model to better responses

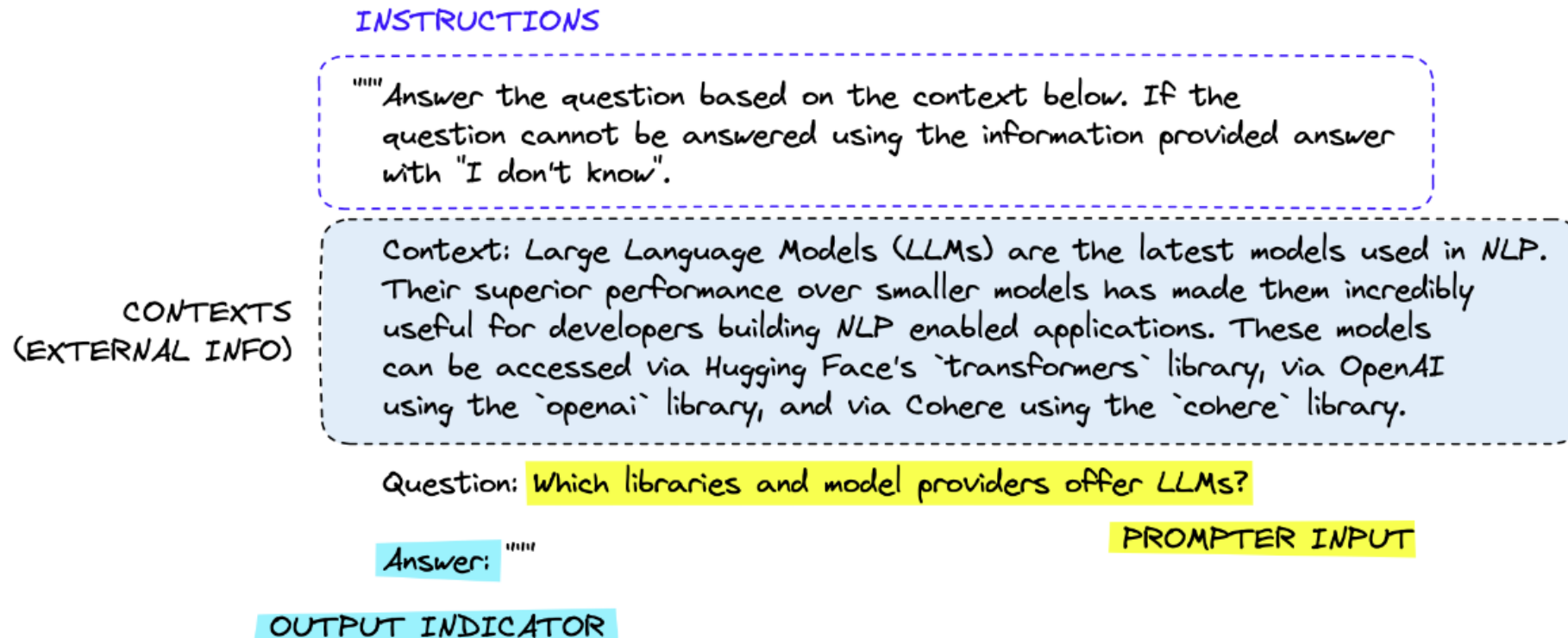
Input Data - the input or question that we are interested to find a response for

Output Indicator - the type or format of the output.

You do not need all the four elements for a prompt and the format depends on the task at hand.

Each LLM may specify a prompt template, refer to the model card in the Hugging Face

Example Prompt Structure



A typical prompt structure.

Langchain Prompt Templates Examples

Prompt templates

Prompt templates are pre-defined recipes for generating prompts for language models.

A template may include instructions, few-shot examples, and specific context and questions appropriate for a given task.

LangChain provides tooling to create and work with prompt templates.

LangChain strives to create model agnostic templates to make it easy to reuse existing templates across different language models.

Typically, language models expect the prompt to either be a string or else a list of chat messages.

Prompt template

Use `PromptTemplate` to create a template for a string prompt.

By default, `PromptTemplate` uses `Python's str.format` syntax for templating; however other templating syntax is available (e.g., `jinja2`).

```
from langchain import PromptTemplate

prompt_template = PromptTemplate.from_template(
    "Tell me a {adjective} joke about {content}."
)
prompt_template.format(adjective="funny", content="chickens")
```



```
"Tell me a funny joke about chickens."
```

Few shot learning

- The success of LLMs comes from their large size and ability to store “knowledge” within the model parameter, which is *learned* during model training.
- However, there are more ways to pass knowledge to an LLM.
- The two primary methods are:
 - **Parametric knowledge** — the knowledge mentioned above is anything that has been learned by the model during training time and is stored within the model weights (or *parameters*).
 - **Source knowledge** — any knowledge provided to the model at inference time via the input prompt.
- Langchain’s **FewShotPromptTemplate** caters to source knowledge input. The idea is to “train” the model on a few examples — we call this few-shot learning — and these examples are given to the model within the prompt

Demo – Few shot

Googling versus ChatGPT

How many elections has PM of India win? Just give me a number.

As of my last update, Narendra Modi had won two general elections as the Prime Minister of India.

No, PM has contested assembly elections as well.

Narendra Modi has won seven elections as the Prime Minister of India.

Election	House	Constituency	Result	Party
2002 ^[a]	Gujarat Legislative Assembly	Rajkot II	Won	BJP
2002		Maninagar	Won	
2007			Won	
2012			Won	
2014	Lok Sabha	Vadodara ^[b]	Won	
		Varanasi	Won	
2019			Won	
2024			Pending	

Chain of Thoughts

- The term PM of India must be referring to the incumbent Prime Minister of India
- “PM of India” should be “Narendra Modi”
- I need to find out the count of all elections he contested and pick those that he won.
- That would give me the required answer.

Using Few Shot Examples, it possible to get a more accurate answer.

Advanced Prompt Engineering

- Chain of Thoughts

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

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou
Google Research, Brain Team
{jasonwei,dennyzhou}@google.com

Published as a conference paper at ICLR 2023

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao^{*1}, Jeffrey Zhao², Dian Yu², Nan Du², Izhak Shafran², Karthik Narasimhan¹, Yuan Cao²

¹Department of Computer Science, Princeton University

²Google Research, Brain team

¹{shunyuy,karthikn}@princeton.edu

²{jeffreyzhao,dianyu,dunan,izhak,yuancao}@google.com

- Tree of Thoughts

Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Shunyu Yao Dian Yu Jeffrey Zhao Izhak Shafran
Princeton University Google DeepMind Google DeepMind Google DeepMind
Thomas L. Griffiths Yuan Cao Karthik Narasimhan
Princeton University Google DeepMind Princeton University

Published as a conference paper at ICLR 2023

- Graph of Thoughts

Abstract

Language models are increasingly being deployed for general problem solving across a wide range of tasks, but are still confined to token-level, left-to-right decision-making processes during inference. This means they can fall short in tasks that require exploration, strategic lookahead, or where initial decisions play a pivotal role. To surmount these challenges, we introduce a new framework for language model inference, “Tree of Thoughts” (ToT), which generalizes over the popular “Chain of Thought” approach to prompting language models, and enables exploration over coherent units of text (“thoughts”) that serve as intermediate steps toward problem solving. ToT allows LMs to perform deliberate decision making by considering multiple different reasoning paths and self-evaluating choices to decide the next course of action, as well as looking ahead or backtracking when necessary to make global choices. Our experiments show that ToT significantly enhances language models’ problem-solving abilities on three novel tasks requiring non-trivial planning or search: Game of 24, Creative Writing, and Mini Crosswords. For instance, in Game of 24, while GPT-4 with chain-of-thought prompting only solved 4% of tasks, our method achieved a success rate of 74%. Code repo with all prompts: <https://github.com/princeton-nlp/tree-of-thought-11m>.

SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

Xuezhi Wang^{†‡} Jason Wei[†] Dale Schuurmans[†] Quoc Le[†] Ed H. Chi[†]
Sharan Narang[†] Aakanksha Chowdhery[†] Denny Zhou^{†§}

[†]Google Research, Brain Team

[‡]xuezhw@google.com, [§]dennyzhou@google.com

- ReAct Agents

Chain-of-Thought (CoT) Prompting

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

Example

A

```
standard_prompt = f'''Use numbers and basic arithmetic operations (+ - * /) to obtain 24
```

```
Input: 4 4 6 8
```

```
Answer: (4 + 8) * (6 - 4) = 24
```

```
Input: 2 9 10 12
```

```
Answer: 2 * 12 * (10 - 9) = 24
```

```
Input: 4 9 10 13
```

```
Answer: (13 - 9) * (10 - 4) = 24
```

```
Input: 1 4 8 8
```

```
Answer: (8 / 4 + 1) * 8 = 24
```

```
Input: 5 5 5 9
```

```
Answer: 5 + 5 + 5 + 9 = 24
```

```
Input: 1 1 2 12
```

```
'''
```



It appears that you've provided a set of input numbers and specified arithmetic operations to obtain the result of 24 for each input. The provided examples demonstrate various mathematical expressions that yield the desired result. Here are the corresponding answers for the given inputs:

1. Input: 4 4 6 8

Answer: $(4 + 8) * (6 - 4) = 24$

2. Input: 2 9 10 12

Answer: $2 * 12 * (10 - 9) = 24$

3. Input: 4 9 10 13

Answer: $(13 - 9) * (10 - 4) = 24$

4. Input: 1 4 8 8

Answer: $(8 / 4 + 1) * 8 = 24$

5. Input: 5 5 5 9

Answer: $5 + 5 + 5 + 9 = 24$

6. Input: 1 1 2 12

Answer: $(1 + 1) * (2 + 12) = 24$

These are valid solutions that use basic arithmetic operations to reach the target number of 24 for each set of input numbers.

Chain of Thoughts – see `cot.py` in my code

Tree of Thoughts

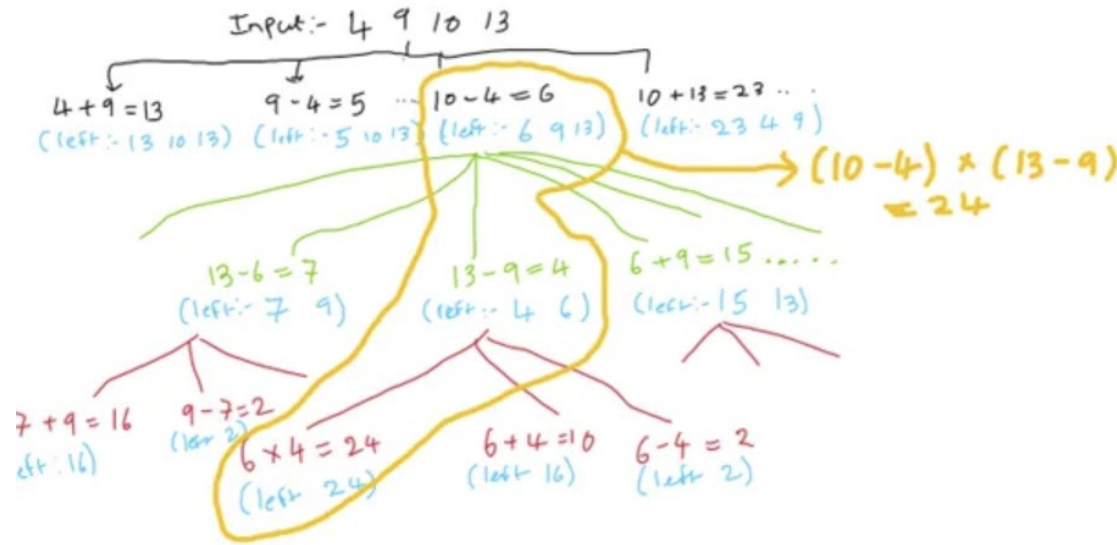


Fig 1: Solve a problem using a systematic way — Tree based approach

We can solve this in 3 steps:

Step 1 : Take 2 numbers and apply all the arithmetic operation at the first level for e.g. $4 + 9 = 13$ — sum of first 2 numbers

Step 2 : Identify left numbers as we need to use each number once. Once first 2 numbers are used we left with (left 13 10 13)

Step 3 : Continue to build the tree till we receive final sum.

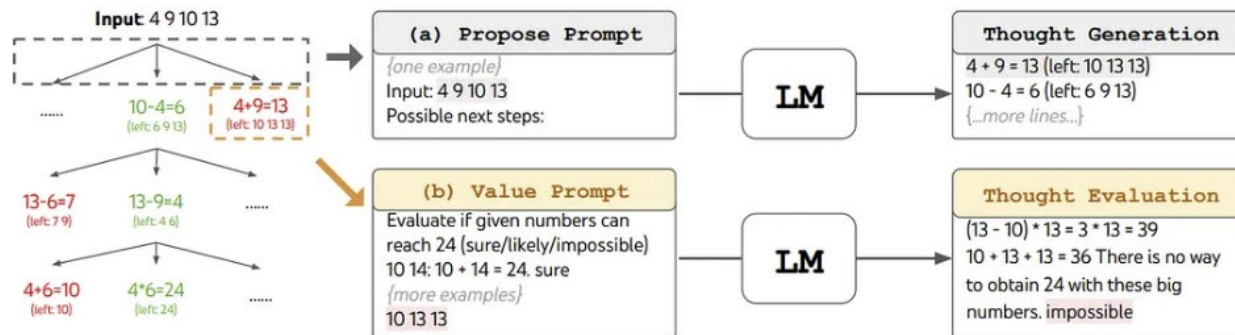
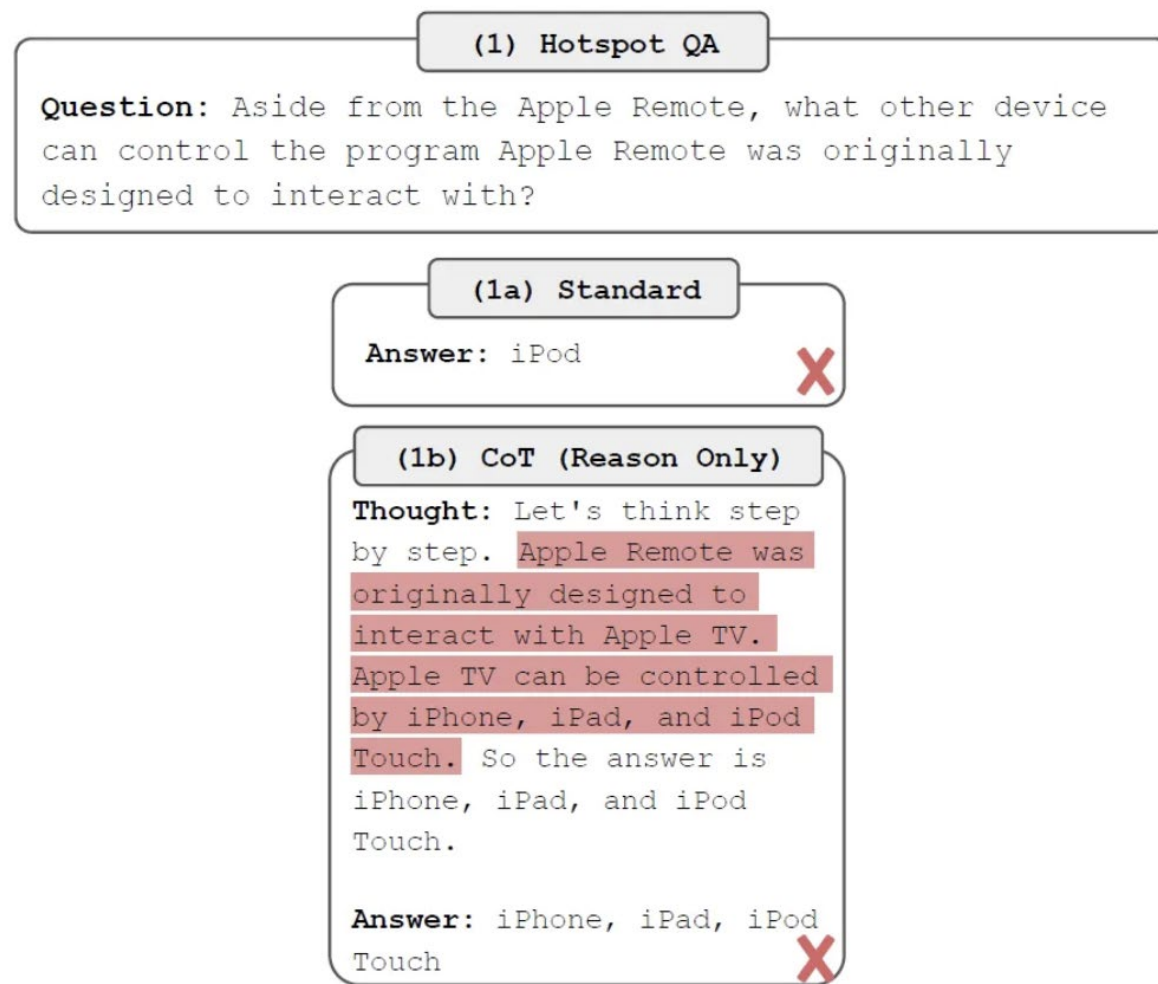


Fig 2: Source: Tree of Thoughts <https://arxiv.org/pdf/2305.10601.pdf>

Limitations of CoT Reasoning

- Chain-of-thought reasoning is intended to combat reasoning errors.
- Giving the LLM one or more examples (few-shot learning) and illustrating how to reason through examples, helps to solve a different problem in a more accurate way.
- But it still suffers from hallucination, and hallucinated “facts” can propagate through the reasoning, causing the model to come to the wrong conclusion regardless.
- ReAct aims to solve this issue by allowing the LLM to take actions such as searching Wikipedia so that it can find facts and reason from those.



Self Consistency

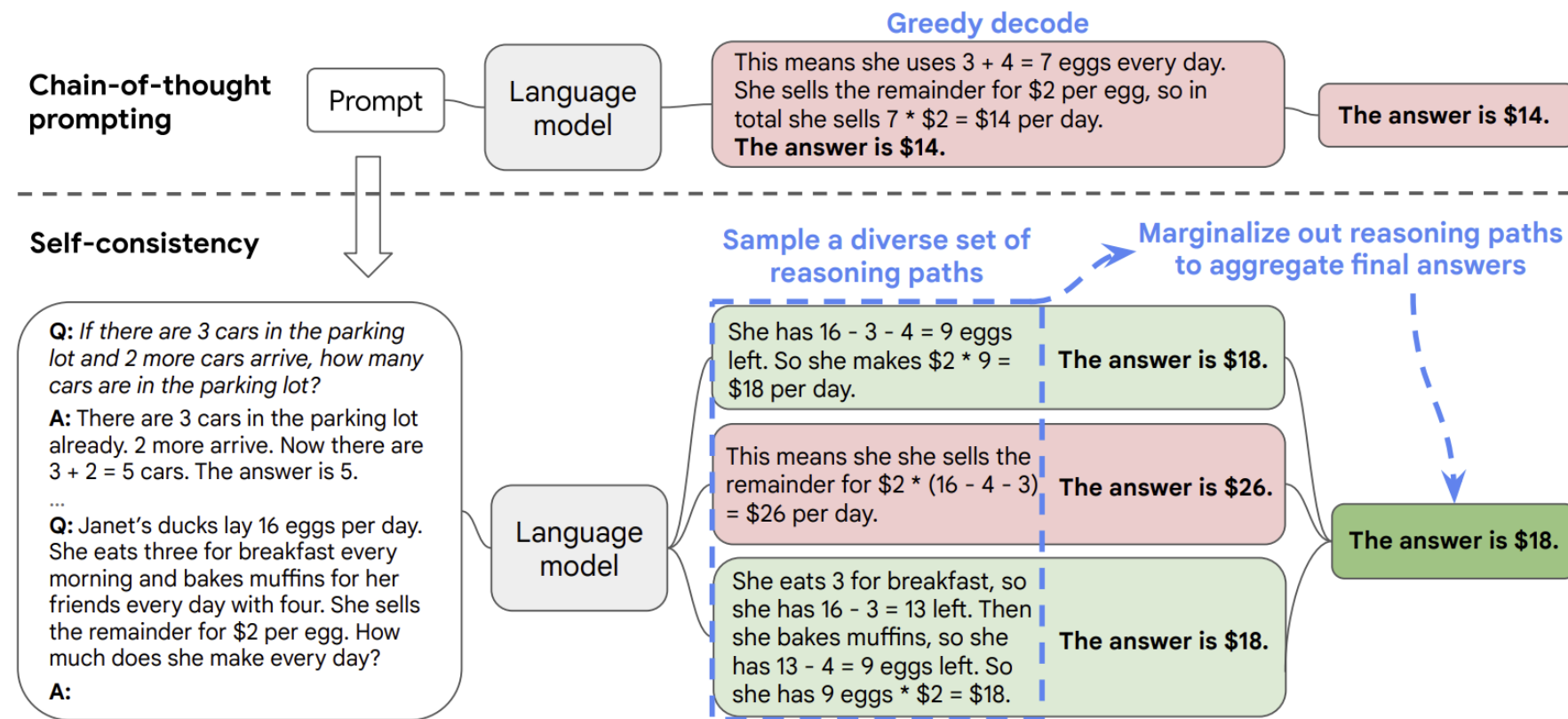
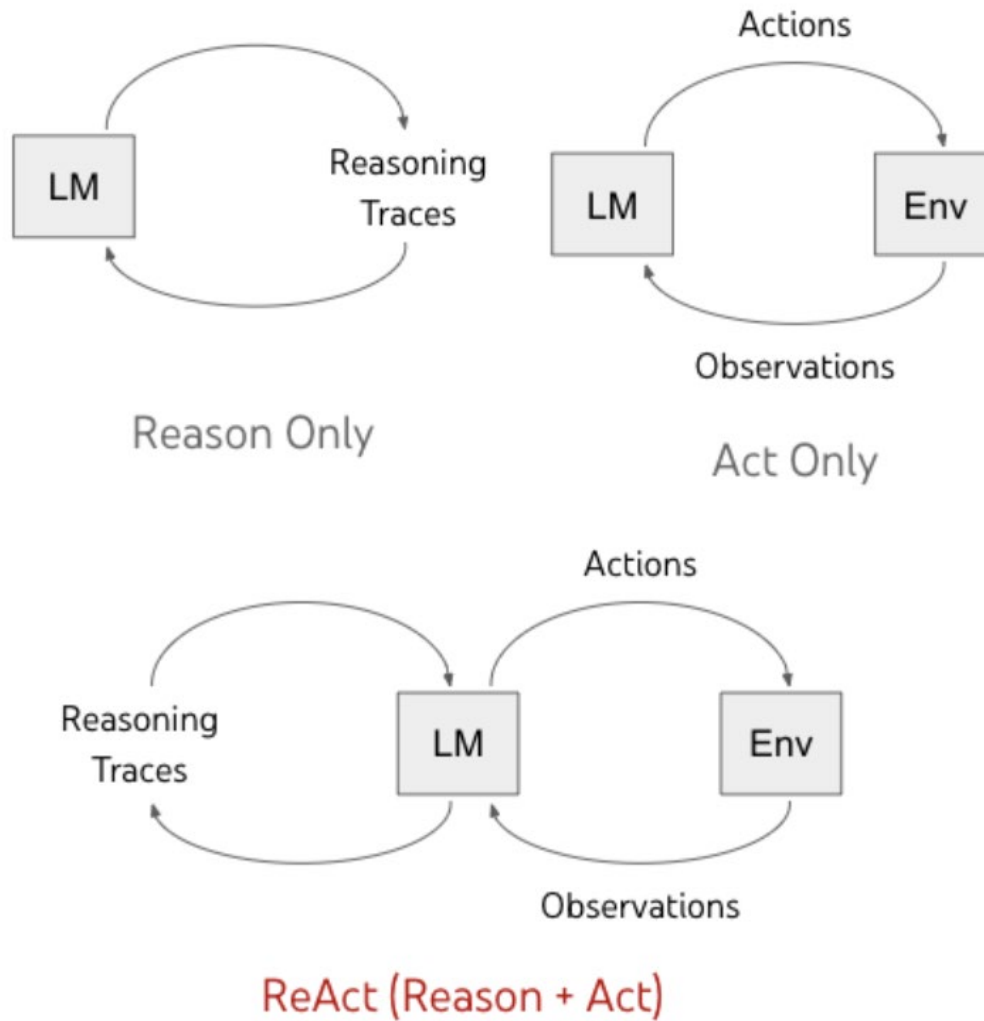


Figure 1: The self-consistency method contains three steps: (1) prompt a language model using chain-of-thought (CoT) prompting; (2) replace the “greedy decode” in CoT prompting by sampling from the language model’s decoder to generate a diverse set of reasoning paths; and (3) marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

ReAct Approach: Key Idea



Approach

1. An environment that takes a text action (out of a set of potential actions which can change based on the environment's internal state) and returns a text observation.
2. An output parser framework that stops the agent from generating text once it has written a valid action, executes that action in the environment, and returns the observation (appends it to the text generated so far and prompts the LLM with that).
3. Human-generated examples of intermixed thoughts, actions, and observations in the environment to use for few-shot learning.

Illustration

Question	What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?
Thought 1	I need to search Colorado orogeny, find the area that the eastern sector of the Colorado orogeny extends into, then find the elevation range of the area.
Action 1	Search[Colorado orogeny]
Observation 1	The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas.
Thought 2	It does not mention the eastern sector. So I need to look up eastern sector.
Action 2	Lookup[eastern sector]
Observation 2	(Result 1 / 1) The eastern sector extends into the High Plains and is called the Central Plains orogeny.
Thought 3	The eastern sector of Colorado orogeny extends into the High Plains. So I need to search High Plains and find its elevation range.
Action 3	Search[High Plains]

Yao et al. (2023)

Illustration#2

HotPotQA

Question: Seven Brief Lessons on Physics was written by an Italian physicist that has worked in France since what year?

(a) Standard

Answer: 1986

X

(b) Reason only

Thought: Let's think step by step. Seven Brief Lessons on Physics was written by Carlo Rovelli. Carlo Rovelli has worked in France since 1990.

Answer: 1990

X

(c) Act only

Act 1: Search[Seven Brief Lessons on Physics]

Obs 1: Seven Brief Lessons on Physics (Italian: Sette brevi lezioni di fisica) is a short book by the Italian physicist Carlo Rovelli. Originally published in Italian in...

Act 2: Lookup[Carlo Rovelli]

Obs 2: (Result 1 / 1) Seven Brief Lessons on Physics (Italian: Sette brevi lezioni di fisica) is a short book by the Italian physicist Carlo Rovelli.

Act 3: Finish[1983]

X

(d) ReAct

Thought 1: I need to search Seven Brief Lessons on Physics, find its author, then find when the author has worked in France since.

Act 1: Search[Seven Brief Lessons on Physics]

Obs 1: Seven Brief Lessons on Physics (Italian: Sette brevi lezioni di fisica) is a short book by the Italian physicist Carlo Rovelli. Originally published in Italian in...

Thought 2: The author of Seven Brief Lessons on Physics is Carlo Rovelli. I need to search Carlo Rovelli next and find when he has worked in France since.

Act 2: Search[Carlo Rovelli]

Obs 2: Carlo Rovelli is an Italian theoretical physicist and writer who has worked in Italy, the United States and, since 2000, in France.[1] He is also currently a Distinguished Visiting Research Chair at the Perimeter Institute...

Thought 3: Carlo Rovelli has worked in France since 2000. So the answer is 2000.

Act 3: Finish[2000]

✓

Illustration#3

ReAct

Claim: Reign Over Me is an American film made in 2010.
Thought 1: I need to search for Reign Over Me and find if it is an American film made in 2010.
Action 1: Search[Reign Over Me]
Observation 1: Reign Over Me ...
Thought 2: The observation says that it is an American film made in 2007, so it is not made in 2010.
Action 2: Finish[REFUTES]
Observation 2: Episode finished

CoT

Claim: Reign Over Me is an American film made in 2010.
Thought: Let's think step by step. First, is Reign Over Me an American film? Yes, it is. Second, was it made in 2010? Yes, it was.
Answer: SUPPORTS

*A comparison of the ReAct (**top**) and CoT (**bottom**) reasoning trajectories on an example from Fever (observation for ReAct is omitted to reduce space). In this case ReAct provided the right answer, and it can be seen that the reasoning trajectory of ReAct is more grounded on facts and knowledge, in contrast to CoT's hallucination behavior.*

Demo: ReAct with LangChain, Tavley