

Advanced Prompt Engineering Techniques are a set of methods for improving the performance of large language models on complex tasks. These techniques involve providing the LLM with more informative and structured prompts, as well as using prior knowledge and logical reasoning to guide the LLM's responses.

Chain-of-Thought (CoT) Prompting

[Chain-of-Thought prompting \(CoT\)](#) is a technique that provides the LLM with a sequence of intermediate steps that lead to the desired answer. It improves the reasoning abilities of large language models (LLMs). It allows the model to focus on solving one step at a time, rather than having to consider the entire problem all at once. It can be used for several reasoning tasks, including math word problems, commonsense reasoning, and symbolic manipulation. It can be readily implemented in sufficiently large language models without any special training or fine-tuning of the model. For example, CoT prompting in the PaLM model significantly enhanced performance in the GSM8K benchmark, improving it from 17.9% to 58.1%.

Few-shot CoT prompts LLMs with examples of similar problems to improve reasoning abilities. It is more effective than a few-shot baseline but can be more complex to implement. Zero-shot CoT involves adding **"Let's think step by step"** to the original prompt. This prompts the LLM to think about the question and come up with a chain of reasoning that leads to the answer. The reasoning is extracted from the LLM's response using a second prompt, "The answer is." Zero-shot CoT has been shown to outperform other methods for evaluating the zero-shot reasoning abilities of LLMs.

(a) Few-shot

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

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. **X**

(b) Few-shot-CoT

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

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

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf



Services

Blog

Contact

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 **X**

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

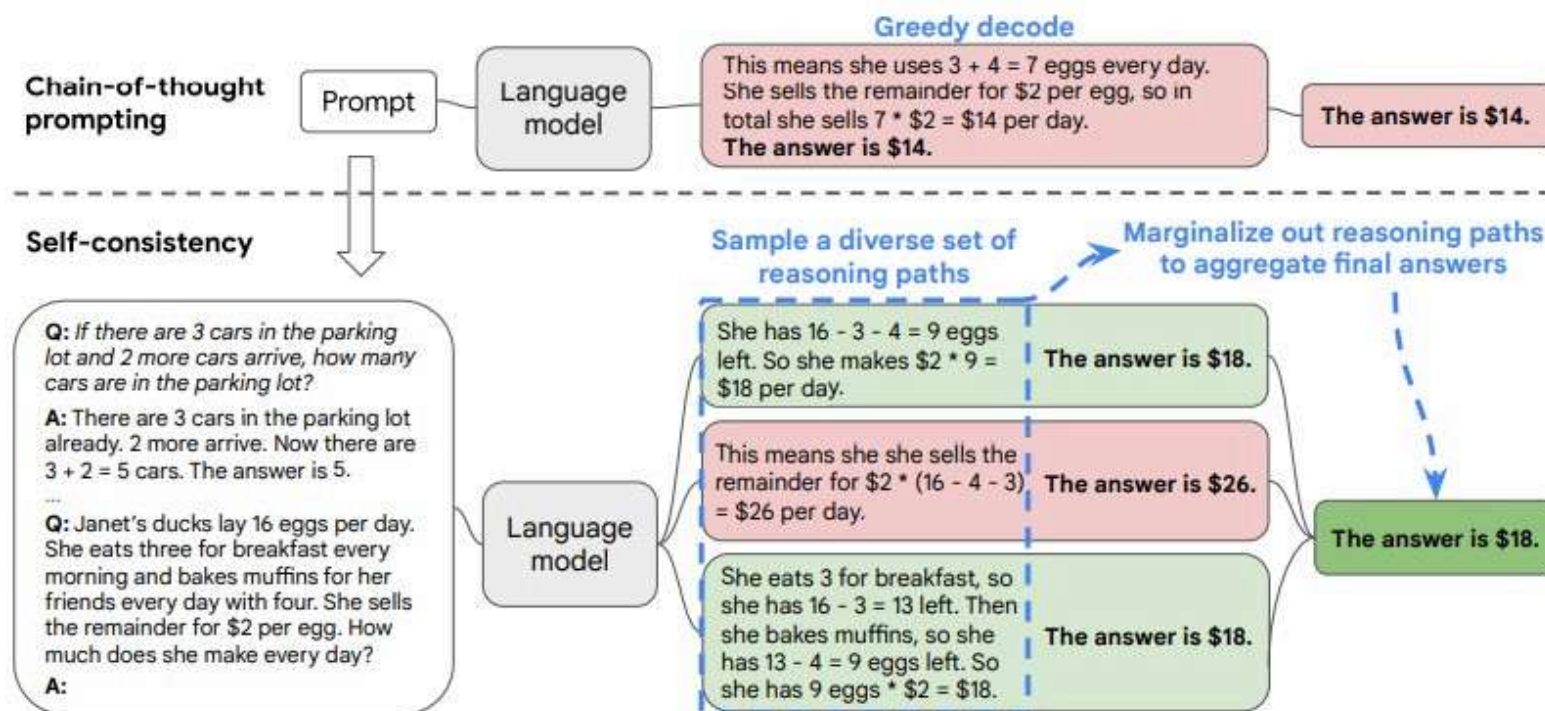
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. **✓**

CoT reasoning emerges in LLMs exceeding 100 billion parameters. This ability may stem from large LLMs' training on extensive datasets that include step-by-step reasoning. While instruction-following isn't essential for CoT, it might enhance its quality. Further research is required to fully understand the origins and potential of CoT reasoning in large LLMs. The researchers found that CoT prompting consistently outperformed standard baseline prompting across various linguistic styles, annotators, examples, and language models. It shows its robustness and effectiveness in enhancing language models' performance on diverse tasks. Sensitivity in CoT prompting pertains to how prompt design influences model performance. Well-matched, clear prompts are crucial, especially for complex tasks. Coherence in CoT ensures that reasoning steps follow a logical order. Later steps shouldn't depend on earlier ones, and vice versa. Removing coherence negatively affected system performance.

Self Consistency

Self-consistency is a technique for generating multiple diverse chains of thought for the same problem and then training the model to select the most consistent answer among these chains.

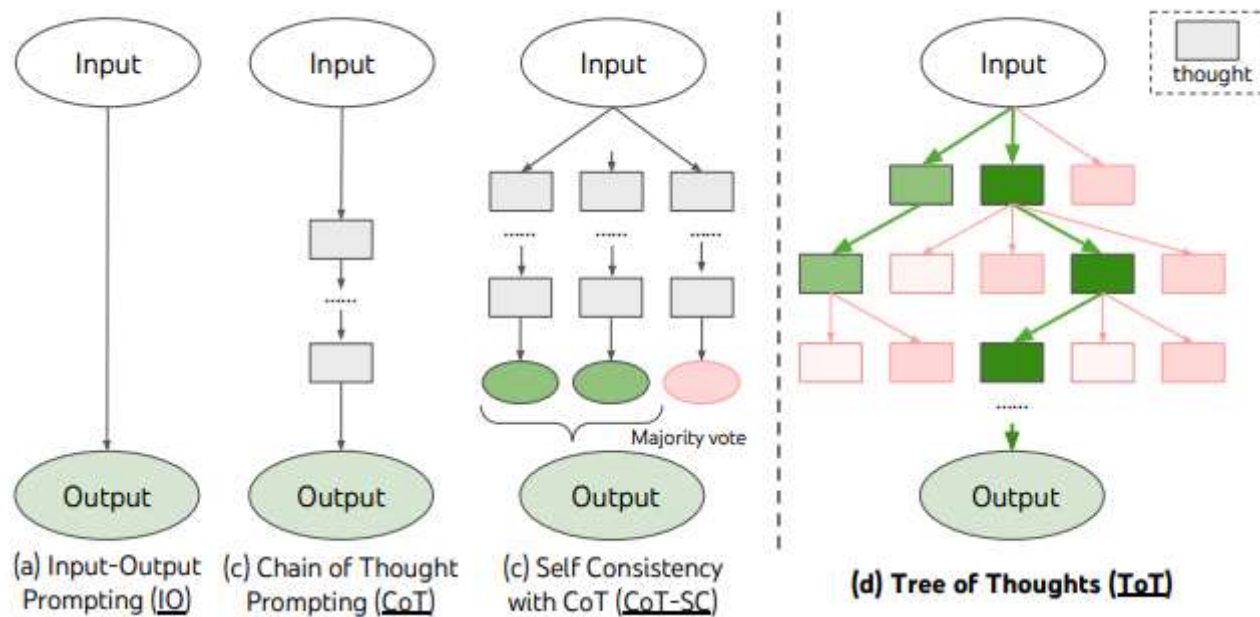
It is used to enhance the performance of language models, especially in tasks requiring multi-step reasoning, like chain-of-thought prompting.



It improves performance of CoT prompting across various benchmarks, such as GSM8K by 17.9%, SVAMP by 11.0%, and AQuA by 12.2%. It's an unsupervised technique that is compatible with pre-trained language models, requiring no extra human annotation, training, fine-tuning, or model changes. It remains robust across different sampling strategies and parameters, consistently enhancing performance. The benefits of self-consistency become more significant as language model scale increases. For example, it contributes up to +23% accuracy improvement for larger models like LaMDA137B and GPT-3. Even for large models that already perform well, self-consistency consistently offers additional gains, such as +12%-18% accuracy improvement on tasks like AQuA and GSM8K over PaLM-540B.

Tree-of-Thoughts (ToT) Prompting

Tree of Thoughts (ToT) is a new framework that extends the Chain-of-Thought approach by allowing language models to explore coherent units of text ("thoughts") as intermediate steps towards problem solving. ToT enables LMs to make deliberate decisions, consider multiple reasoning paths, and self-evaluate choices. It also allows LMs to look ahead or backtrack when necessary for making global decisions.



Tree of Thoughts enhances language models' problem-solving abilities on tasks like Game of 24, Creative Writing, and Mini Crosswords.

SYSTEM

Imagine three different experts having a conversation. They answer where the ball is now. If any expert realizes that they are wrong at any point they leave.

USER

The question is:

Bill is in the living room.

He walks to the kitchen, carrying a cup.

He puts a ball in the cup and carries the cup to the bedroom.

He turns the cup upside down, then walks to the garden.

He puts the cup down in the garden, then walks to the garage.

Where is the ball?

ASSISTANT

Expert 1: The ball is in the garden. Bill carried the cup to the garden and put it down there.

Expert 2: I disagree. The ball is in the bedroom. Bill turned the cup upside down in the bedroom, which would have caused the ball to fall out.

Expert 3: I concur with Expert 2. The ball is in the bedroom. When Bill turned the cup upside down, the ball would have fallen out.

Expert 1: Upon reconsideration, I agree with both of you. I was wrong. The ball is indeed in the bedroom. I will now leave the conversation.

⊖

Add message

For example, IO, CoT, and CoT-SC perform poorly on the task of solving Game of 24, achieving only 7.3%, 4.0%, and 9.0% success rates, respectively. ToT achieves much better results on this task. ToT with a breadth of $b = 1$ (meaning that it considers one possible solution at a time) already achieves a success rate of 45%, while $b = 5$ (meaning that it considers five possible solutions at a time) achieves 74%.

ToT is effective in tasks that require non-trivial planning or search. In the average GPT-4 scores for the three methods (ToT, IO, and CoT) across 100 tasks, ToT has the highest average score (7.56), followed by IO (6.19) and CoT (6.93). ToT is able to generate more coherent passages than IO and CoT on average.

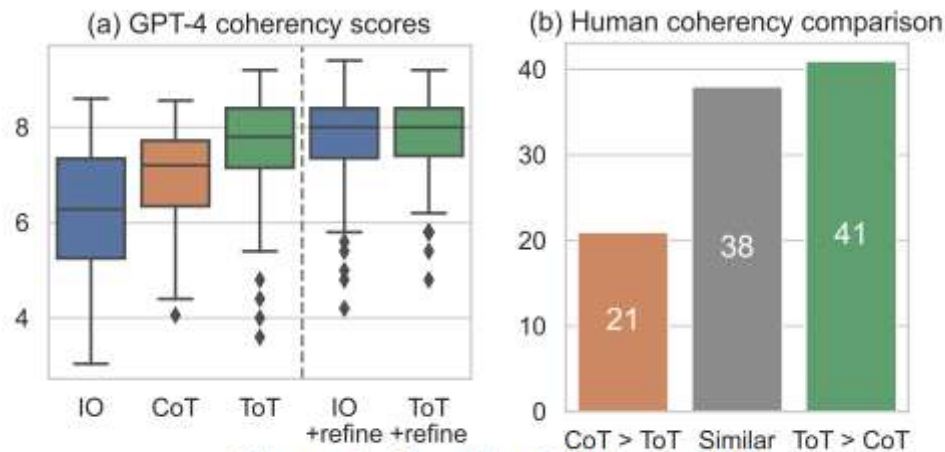


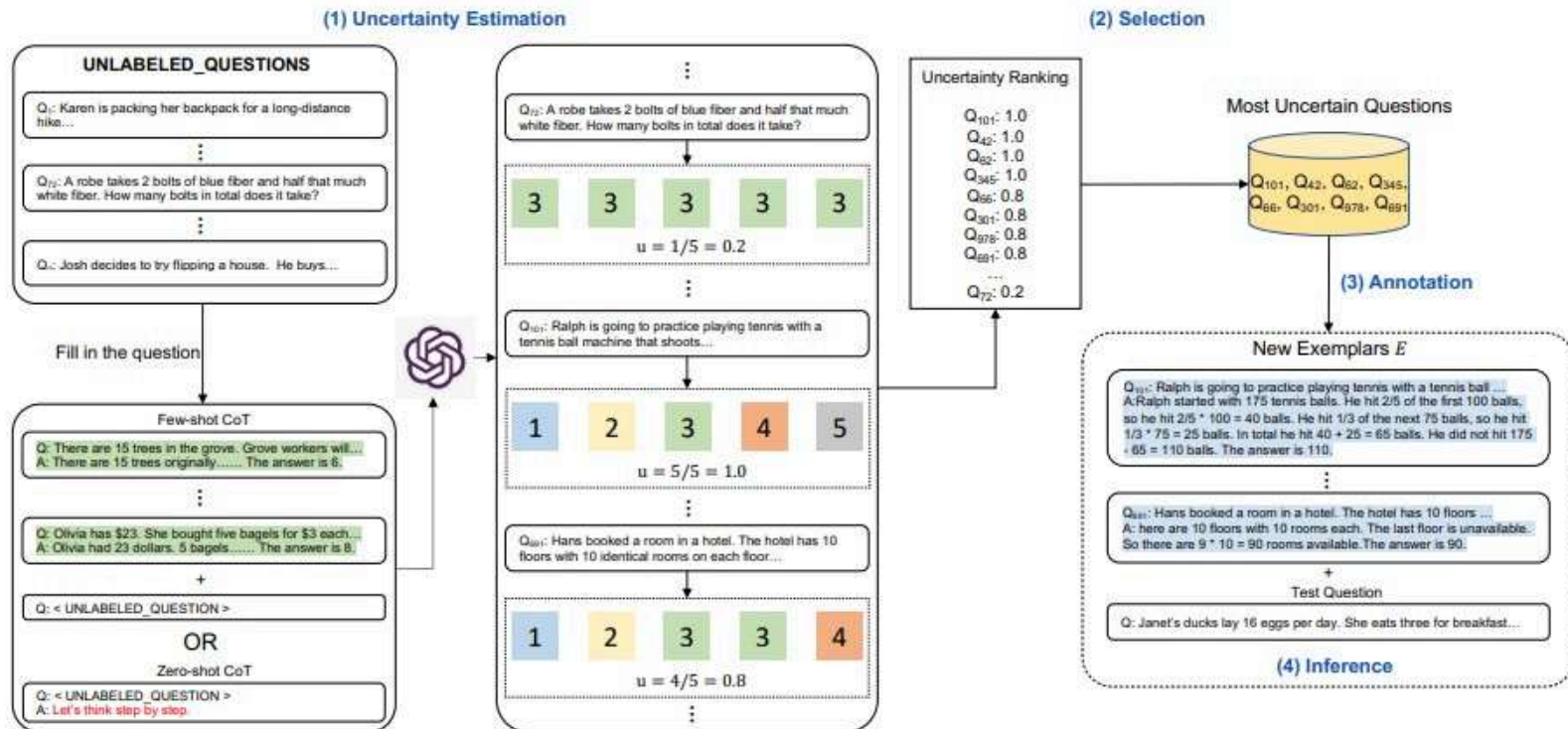
Figure 5: Creative Writing results.

Method	Success Rate (%)		
	Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	78	60	20
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

Active Prompting

Active prompting uses uncertainty-based active learning for adapting large language models (LLMs) to different tasks. It works in four stages. The first stage is uncertainty estimation. In this stage, the LLM is queried k times to generate possible answers with intermediate steps for a set of training questions. The uncertainty of each question is then calculated based on the k answers with a method called disagreement. Disagreement measures how much the k answers disagree with each other. The second stage is selection. The most uncertain questions are selected for annotation. The algorithm starts with the most uncertain question and then selects the next most uncertain question that is not already selected. The third stage is annotation. Humans annotate the selected questions with human-designed CoT reasoning. The CoT reasoning provides the LLM with additional information about how to answer the questions. The fourth stage is inference. The LLM is used to infer the answers to the questions. The LLM uses the new annotated exemplars to improve its performance on the questions.



Active prompt achieves the best performance compared with all baseline models. It is the most effective method for improving the performance of large language models (LLMs) on a variety of reasoning tasks.

USER

Q: As a freelancer, Baylor is paid for every finished work of a client he does on a freelance marketplace. Currently, he has \$4000 on his dashboard from previous work done. He is currently working for three clients, with the first client paying him half the amount of money he currently has on his dashboard once the job is done. The second client will pay him $\frac{2}{5}$ times more money than the first client once Baylor finishes his work. The third client will pay him twice the amount of money the first and second clients pay him together once he finishes the job. How much money will Baylor have in his dashboard after all the clients pay him for his work?

ASSISTANT

The first client will pay Baylor half the amount of money he currently has on his dashboard, a total of $\frac{1}{2} * \$4000 = \$4000 * 0.5 = 2000$

The total amount of money in Baylor's dashboard after the first client pays him is $\$4000 + \$2000 = \$4000 + 2000 = 6000$

The second client will pay Baylor $\frac{2}{5}$ times more money than the first client, a total of $\frac{2}{5} * \$2000 = \$2000 * \frac{2}{5} = 800$ more

The total amount of money the second client will pay Baylor is $\$2000 + \$800 = \$2000 + 800 = 2800$

The total amount of money in Baylor's dashboard after the second client pays him is $\$6000 + \$2800 = \$6000 + 2800 = 8800$

The third client will pay Baylor twice the amount of money the first and second clients pay him together, a total of $2 * (\$2000 + \$2800) = 2 * (2000 + 2800) = 9600$

The total amount of money in Baylor's dashboard after the third client pays him is $\$8800 + \$9600 = \$8800 + 9600 = 18400$. Answer: $\boxed{18400}$.

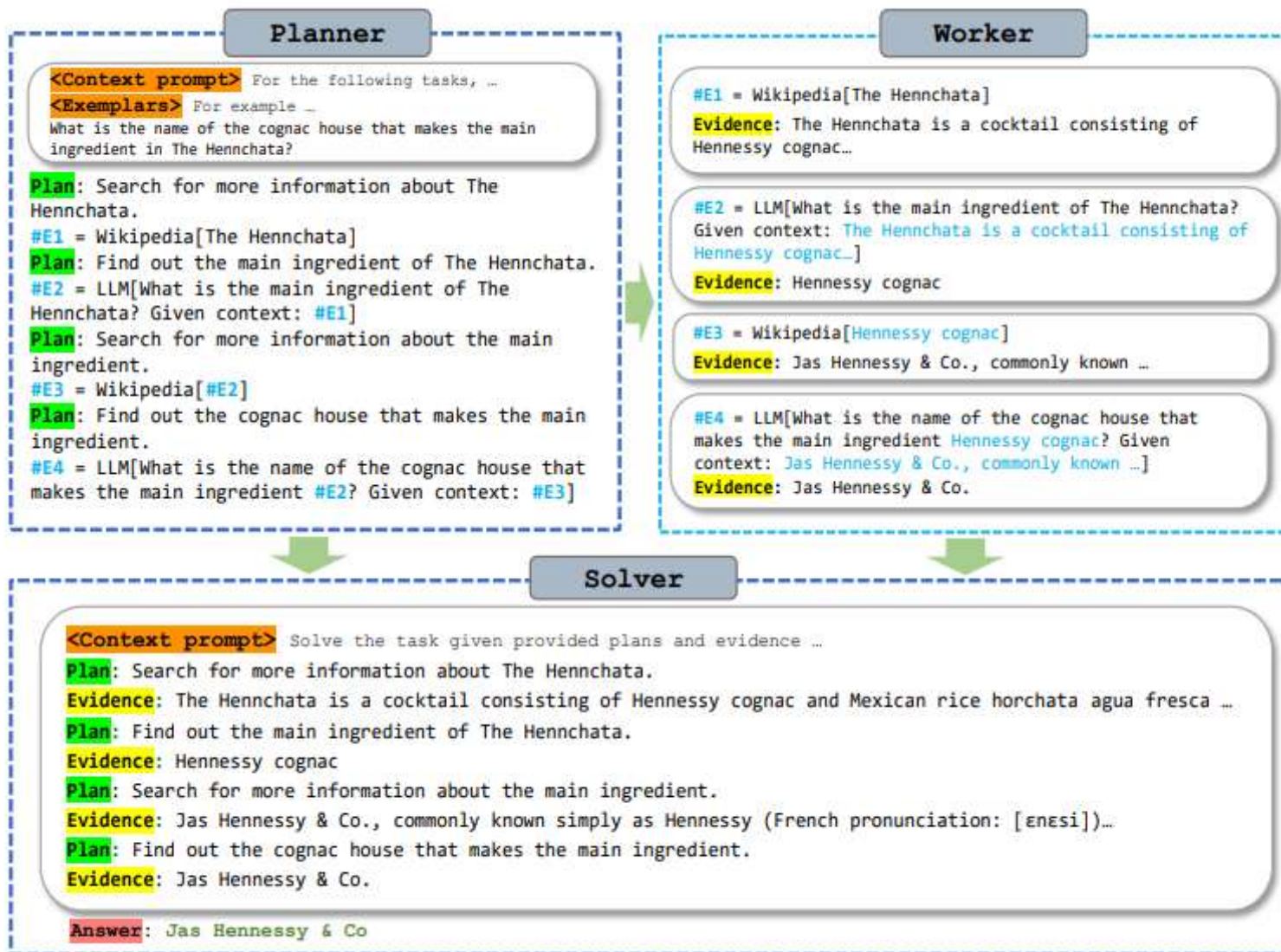
It outperforms self-consistency by an average of 2.1% with code-davinci-002 and 7.2% with text-davinci-002. This suggests that Active-Prompt is a more effective way to improve the performance of LLMs than self-consistency, which is a previous method for training LLMs. The largest improvement is observed in GSM8K (4.2%) and AQuA (3.1%). This suggests that Active-Prompt is particularly effective for tasks that do not require the transferability of CoT prompts.

METHOD	GSM8K	ASDiv	SVAMP	AQUA	SINGLEEQ	CSQA	STRATEGY	LETTER (4)	AVG.
Prior Best	55.0 ^a	75.3 ^b	57.4 ^c	37.9 ^d	32.5 ^e	91.2 ^f	73.9 ^g	-	-
<i>UL2-20B</i>									
CoT	4.4	16.9	12.5	-	-	51.4	53.3	0.0	-
SC	7.3	21.5	19.4	26.9	-	55.7	54.9	0.0	-
<i>LaMDA-137B</i>									
CoT	14.3	46.6	37.5	-	-	57.9	65.4	13.5	-
SC	27.7	58.2	53.3	26.8	-	63.1	67.8	8.2	-
<i>PaLM 540B</i>									
CoT	56.9	73.9	79.0	-	-	79.9	77.8	63.0	-
SC	74.4	81.9	86.6	48.3	-	80.7	81.6	70.8	-
<i>text-davinci-002</i>									
Auto-CoT	47.9	-	69.5	36.5	87.0	74.4	65.4	59.7	-
CoT	46.9	71.3	68.9	35.8	77.3	73.5	65.4	56.6	61.5
SC	58.2	76.9	78.2	41.8	87.2	72.9	70.7	57.6	67.9
Random-CoT	63.9	82.3	81.1	44.1	89.4	74.5	73.3	65.5	71.8
Active-Prompt (D)	73.2	83.2	82.7	48.4	90.6	76.6	76.9	67.7	74.9
Active-Prompt (E)	71.1	83.8	81.8	50.3	93.1	78.8	76.9	66.7	75.3
<i>code-davinci-002</i>									
Auto-CoT	62.8	-	-	-	-	-	-	-	-
CoT	63.1	80.4	76.4	45.3	93.1	77.9	73.2	70.4	72.5
SC	78.0	87.8	86.8	52.0	93.7	81.5	79.8	73.4	79.1
Random-CoT	78.6	87.1	88.0	53.1	94.0	82.1	79.4	73.3	79.4
Active-Prompt (D)	82.2	88.4	88.7	55.1	94.5	83.9	80.6	74.1	80.9
Active-Prompt (E)	83.4	89.3	87.5	57.0	95.5	82.6	80.6	76.7	81.6
<i>text-davinci-003</i>									
CoT	61.7	78.2	77.6	46.0	91.5	76.2	72.6	70.2	71.8
Active-Prompt (D)	65.6	79.8	80.5	48.0	93.1	78.9	74.2	71.2	73.9

Reasoning WithOut Observation (ReWOO)

ReWOO (Reasoning WithOut Observation) is a technique that detaches the reasoning process from external observations, such as the ability to access and process information from the real world. This detachment significantly reduces the amount of tokens that the LLM needs to consume, which in turn improves the efficiency of the LLM. ReWOO divided the workflow into three separate modules:

Planner, **Worker**, and **Solver**. The Planner takes a question as input and breaks it down into a sequence of steps. Each step is then formulated as a plan. The plans are interdependent, meaning that the output of one plan is used as the input to another plan. The Worker takes a plan as input and retrieves external knowledge from tools to provide evidence. The evidence can be anything from factual information to code snippets. The Solver takes the plans and evidence from the Worker module and synthesizes them to generate the ultimate answer to the initial question.



ReWOO was evaluated on six public NLP benchmarks and a curated dataset. It consistently outperformed the baseline methods on all of the benchmarks. For example, on HotpotQA, a multi-step reasoning benchmark, ReWOO achieved 5× token efficiency and 4% accuracy improvement. ReWOO also demonstrated robustness under tool-failure scenarios. It means that ReWOO is still able to perform well even when the external tools that it relies on are not available.

ReWOO outperforms ReAct. ReWOO was able to reduce token usage by 64% with an absolute accuracy gain of 4.4%. It is able to elicit more reasoning capabilities from LLMs than ReAct. ReWOO was also found to be more robust to tool failures than ReAct. When tools malfunction and return errors, ReAct-like ALM systems are highly fragile. ReWOO, on the other hand, is less compromised. ReWOO also performed well on the curated dataset, SOTUQA. SOTUQA is a document QA dataset that is more closely aligned with real-world ALM applications than previous public NLP benchmarks.

Dataset	Paradigm	#Tools	n	Acc	F1	EM	#Tokens	#Steps	\$Cost _{1k}
<i>HotpotQA</i> 1000	Direct	0	0	37.8	36.2	28.0	55.5	1.00	0.11
	CoT	0	1	41.6	30.8	22.4	481.9	1.79	0.96
	REACT	2	6	40.8	39.6	32.2	9795.1	4.97	19.59
	ReWOO	2	6	42.4	40.1	30.4	1986.2	4.45	3.97
<i>TriviaQA</i> 1000	Direct	0	0	<u>80.6</u>	<u>74.0</u>	<u>64.2</u>	43.4	1.00	0.09
	CoT	0	1	78.6	71.7	60.1	199.2	2.08	0.40
	REACT	2	1	59.4	53.2	47.4	4212.9	5.21	8.43
	ReWOO	2	1	66.6	60.6	51.8	1340.9	3.55	2.68
<i>GSM8K</i> 1000	Direct	0	0	26.8	14.4	—	101.1	1.00	0.20
	CoT	0	1	67.4	62.7	—	495.6	3.45	0.99
	REACT	3	1	62.0	37.3	—	1874.3	2.86	3.75
	ReWOO	3	1	62.4	36.2	—	1089.3	3.21	2.18
<i>StrategyQA</i> 300	Direct	0	0	64.6	64.6	64.6	41.8	1.00	0.08
	CoT	0	1 [†]	56.0	56.0	56.0	170.5	1.85	0.34
	REACT	5	1 [†]	64.6	64.6	64.6	1686.3	2.58	3.37
	ReWOO	5	1 [†]	66.6	66.6	66.6	1287.1	3.20	2.57
<i>PhysicsQA</i> 53	Direct	0	0	52.8	12.6	—	132.2	1.00	0.26
	CoT	0	1 [†]	62.2	15.2	—	346.8	3.07	0.69
	REACT	5	1 [†]	64.1	16.2	—	2163.3	2.77	4.33
	ReWOO	5	1 [†]	66.0	14.0	—	1225.7	2.56	2.45
<i>SportsU.</i> 300	Direct	0	0	<u>68.0</u>	<u>68.0</u>	<u>68.0</u>	47.63	1.00	0.10
	CoT	0	1 [†]	53.3	47.5	45.3	215.9	1.78	0.43
	REACT	5	1 [†]	58.6	51.9	49.3	1720.0	2.64	3.44
	ReWOO	5	1 [†]	61.3	55.8	55.3	854.2	3.04	1.71
<i>SOTUQA</i> Curated	Direct	0	0	52.7	15.3	—	52.2	1.00	0.10
	CoT	0	1 [†]	60.8	21.2	—	227.4	2.08	0.45
	REACT	5	1 [†]	64.8	42.7	—	1840.3	2.43	3.68
	ReWOO	5	1 [†]	70.2	44.8	—	1048.8	2.24	2.09

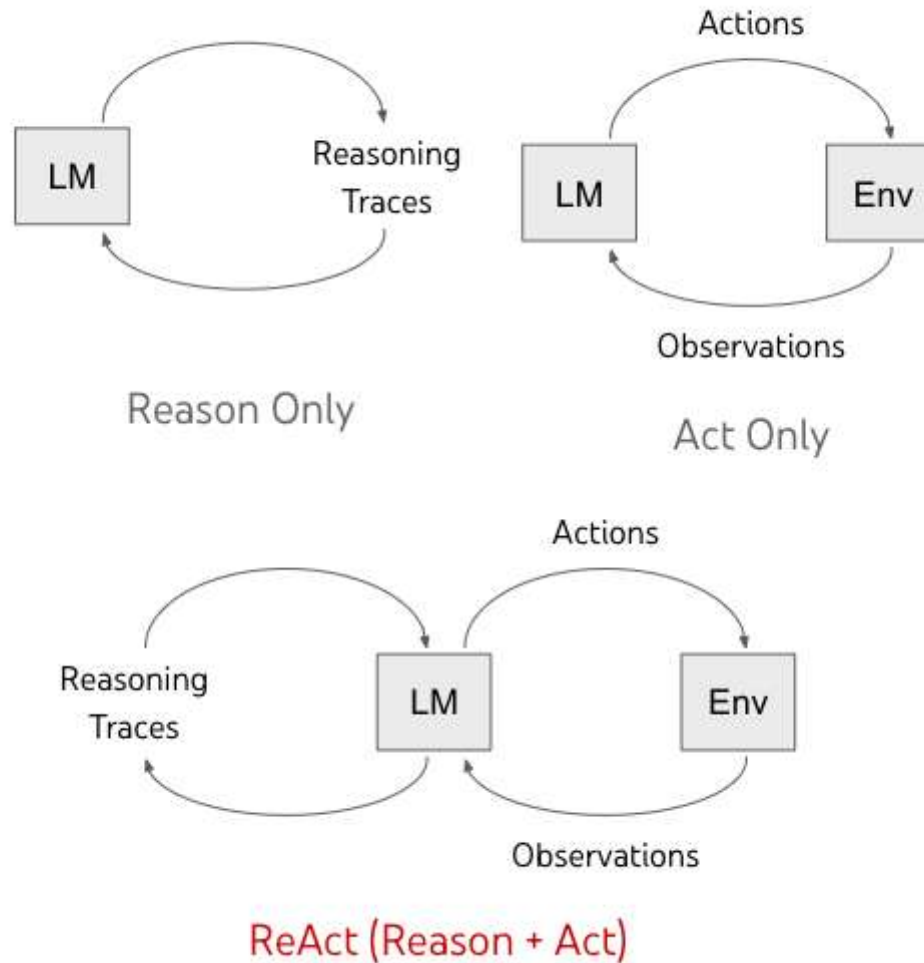
ReWOO decouples parametric modules from nonparametric tool calls. It means that the LLM can be fine-tuned to offload some of its reasoning ability to smaller language models. This offloading can substantially reduce the number of parameters that the LLM needs to store, which further improves

the efficiency of the LLM. ReWOO can offload reasoning ability from a 175B GPT3.5 model to a 7B LLaMA model. It has the potential to create truly efficient and scalable ALM systems.

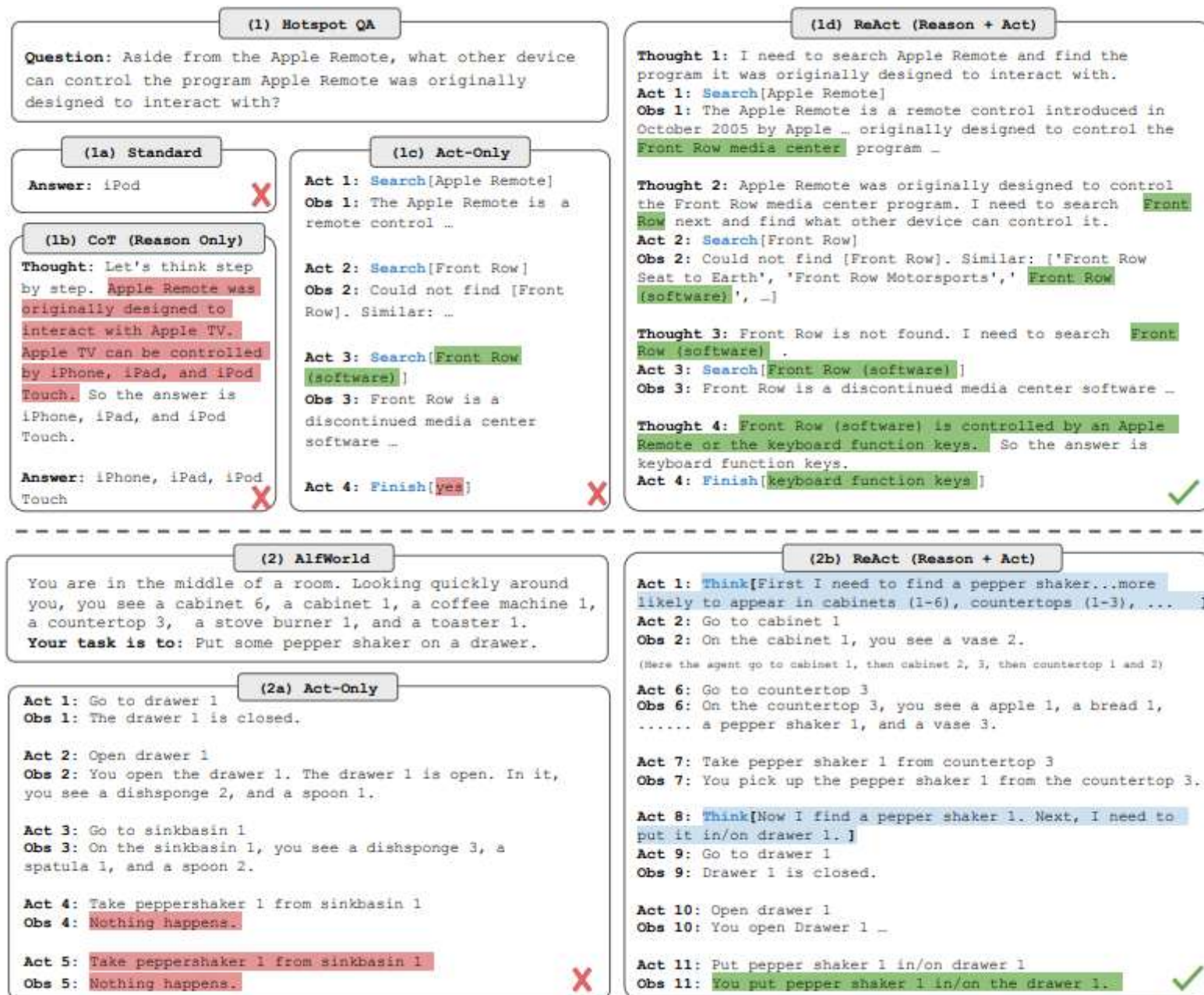
Reason and Act (ReAct)

ReAct is a technique that combines reasoning and acting with language models for solving various language reasoning and decision-making tasks. It prompts language models to generate both verbal reasoning traces and actions. It enables dynamic reasoning, high-level planning for acting, and interaction with external environments.

Here is our [extensive guide on ReAct Prompting](#).



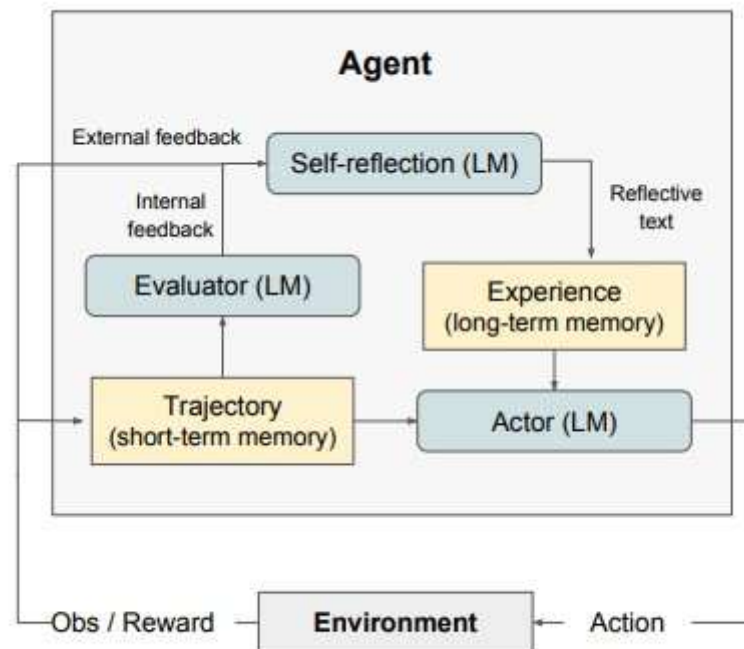
It is evaluated on four diverse benchmarks, including question answering (HotPotQA), fact verification (Fever), text-based games (ALFWorld), and web page navigation (WebShop). On HotpotQA and Fever, ReAct was able to overcome prevalent issues of hallucination and error propagation in chain-of-thought reasoning. It also outperformed imitation and reinforcement learning methods with an improved 34% and 10% on ALFWorld and WebShop. This is because ReAct is able to learn from human examples and apply that knowledge to new situations.



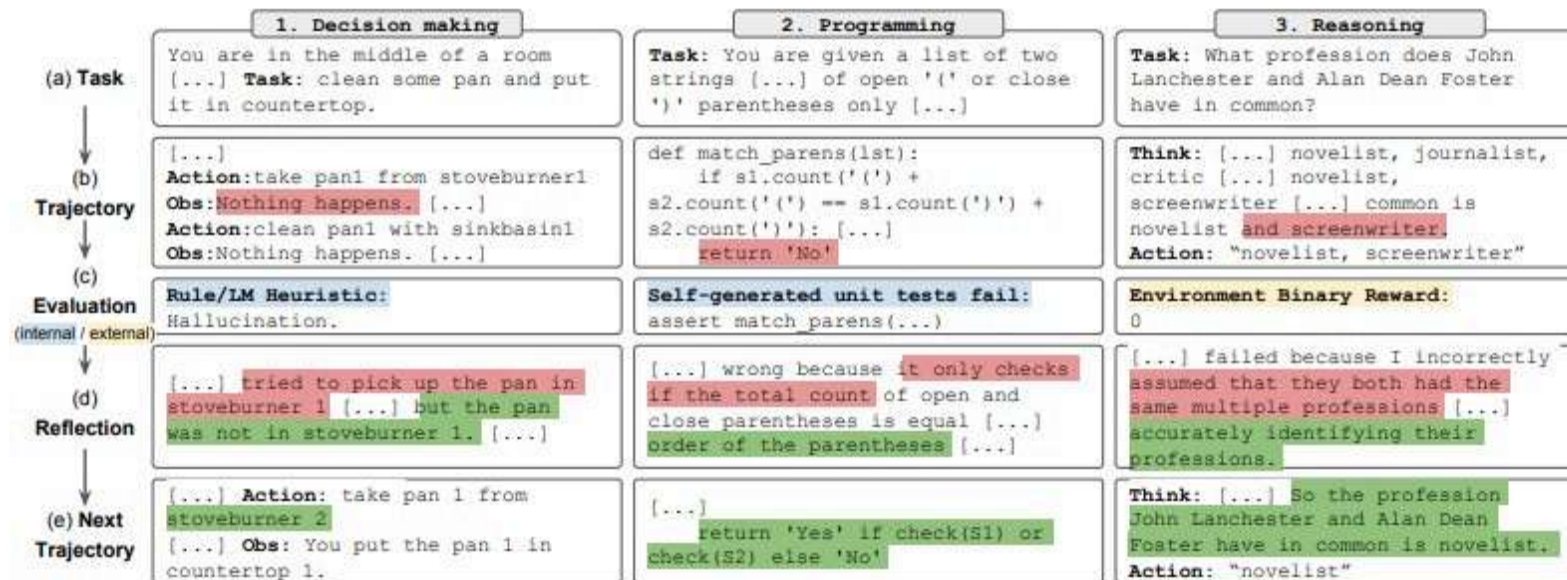
interpretable decision-making and reasoning process, allowing humans to inspect reasoning, factual correctness, and even control or correct the agent's behavior during task execution.

Reflection

Reflexion is a framework that uses linguistic feedback to reinforce language agents. Linguistic feedback is feedback that is expressed in natural language. Reflexion agents learn to reflect on task feedback signals, and then maintain their own reflective text in an episodic memory buffer. This reflective text is then used to induce better decision-making in subsequent trials. The Reflexion framework uses self-reflection. It generates verbal self-reflections to provide more informative feedback. These self-reflections are then stored in the agent's memory. The agent can then use this information to improve its performance on future trials.



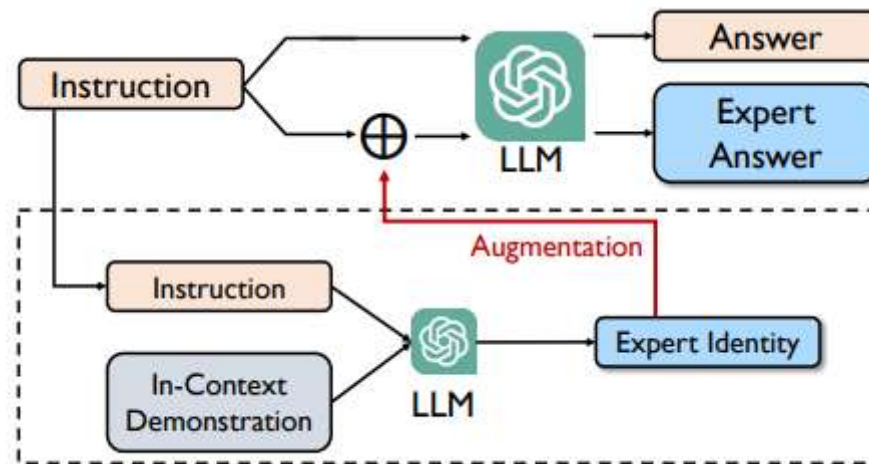
Reflexion is flexible enough to incorporate various types and sources of feedback signals. For example, feedback signals can be scalar values (such as rewards or punishments), or they can be free-form language. Feedback signals can also be external (provided by a human or another agent), or they can be internally simulated (generated by the agent itself).



Reflexion agents outperforms strong baseline approaches in decision-making tasks, reasoning tasks, and programming tasks. In decision-making tasks (AIWorld), Reflexion agents improve by 22% over 12 iterative learning steps. In reasoning questions (HotPotQA), Reflexion agents show a 20% improvement. In Python programming tasks (HumanEval), Reflexion agents achieve an improvement of up to 11%. It achieves a 91% pass@1 accuracy on the HumanEval, surpassing the previous state-of-the-art GPT-4 that achieves 80%.

Expert Prompting

Expert Prompting is an augmented strategy for instructing Large Language Models (LLMs). It envisions a distinguished expert agent tailored to each specific instruction. LLMs are asked to answer instructions conditioned on the identity of the envisioned expert. It is an automatic prompting method. Expert identities are generated using In-Context Learning. It requires writing several instruction-expert pair exemplars. The generated expert identities are found to be satisfactory.



Expert Prompting is a generalized prompting method. Expert identities are defined with detailed and elaborate descriptions. It can match instructions in various domains or genres. It's adaptable to different areas, such as nutrition or physics. It is simple to implement. It doesn't require complex crafting of prompt templates or iterative processes. Writing good expert identity is critical. It should be specialized, detailed, and comprehensive for each instruction. The descriptions must be automatically generated to be practical and efficient.

Instruction: Describe the structure of an atom.

Expert Identity:

You are a physicist specialized in atomic structure and the properties of matter at the atomic level. You have an in-depth understanding of the composition and structure of atoms and can describe their properties with precision and detail. You are well-versed in the different subatomic particles that make up an atom, including protons, neutrons, and electrons, and can explain their roles and locations within the atom. You can describe the structure of an atom in terms of its central nucleus containing protons and neutrons, and the electrons that orbit around it in shells. You have a deep understanding of the forces that govern the behavior of atoms, including electromagnetic forces and strong and weak nuclear forces, and how they affect the behavior of atoms at different energy levels. Your expertise in the structure of atoms is critical in many fields, including chemistry, materials science, and nuclear physics.

gpt-3.5 answer:

An atom consists of a nucleus, which is made up of positively charged protons and neutral neutrons, surrounded by negatively charged electrons that orbit the nucleus. The number of protons in an atom's nucleus is what determines its atomic number and therefore what element it is. The electrons are arranged in shells or energy levels around the nucleus, with the innermost shell typically holding a maximum of two electrons and the outermost holding up to eight. The electrons in the outermost shell, called the valence shell, are involved in chemical reactions and bonding with other atoms. The size of an atom is determined by the distance between the nucleus and the outermost electrons.

gpt-3.5 answer w/ expert identity:

The structure of an atom can be thought of as comprising of three main components: a central nucleus containing protons and neutrons, and electrons that orbit around it in distinct energy levels or shells. The nucleus is the dense, positively charged center of the atom, while the electrons are negatively charged particles that orbit around the nucleus at varying distances. The number of protons present in the nucleus determines the atomic number of the atom, which in turn defines its unique chemical properties and place in the periodic table of elements. The number of neutrons present in the nucleus can vary, giving rise to isotopes of the atom. The electrons occupy distinct energy levels or shells around the nucleus, with those closest to the nucleus having lower energy levels than those further away. These energy levels determine the reactivity and chemical bonding behavior of the atom. The overall structure of an atom is governed by the electromagnetic force that holds the negatively charged electrons in orbit around the positively charged nucleus, and the strong nuclear force that binds the protons and neutrons together in the nucleus.

Automatic Prompt Engineering (APE)

APE is a technique that treats the instruction as the “program,” and it optimizes the instruction by searching over a pool of instruction candidates proposed by an LLM. The LLM candidates are scored using a chosen score function, and the instruction with the highest score is selected. APE is inspired by classical program synthesis and the human approach to prompt engineering. Program synthesis is the task of automatically generating code from a natural language description of the desired behavior. The human approach is the process of manually crafting instructions effective at using LLMs to produce desired outputs.



Keep the high score candidates



Discard the low score candidates



Final selected prompt with highest score

LLMs as Inference Models

Professor Smith was given the following instructions: <INSERT>

Here are the Professor's responses:

Demonstration Start

Input: prove **Output:** disprove

Input: on **Output:** off

...

Demonstration End

[Optional]

LLMs as Resampling Models

Generate a variation of the following instruction while keeping the semantic meaning.

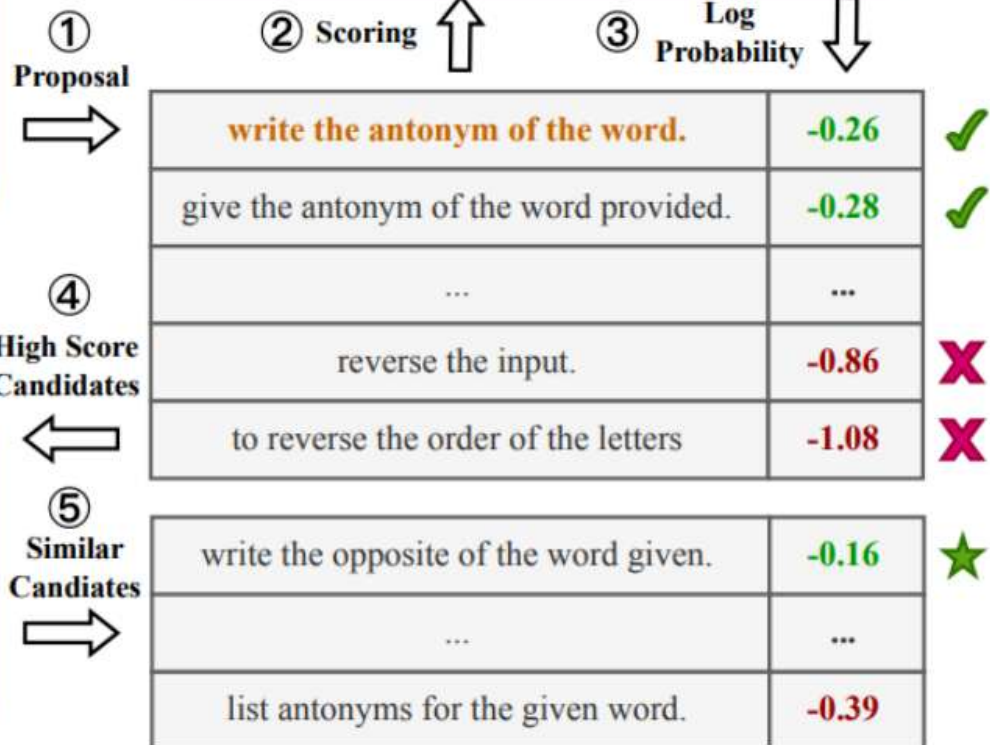
Input: write the antonym of the word.

Output: <COMPLETE>

LLMs as Scoring Models

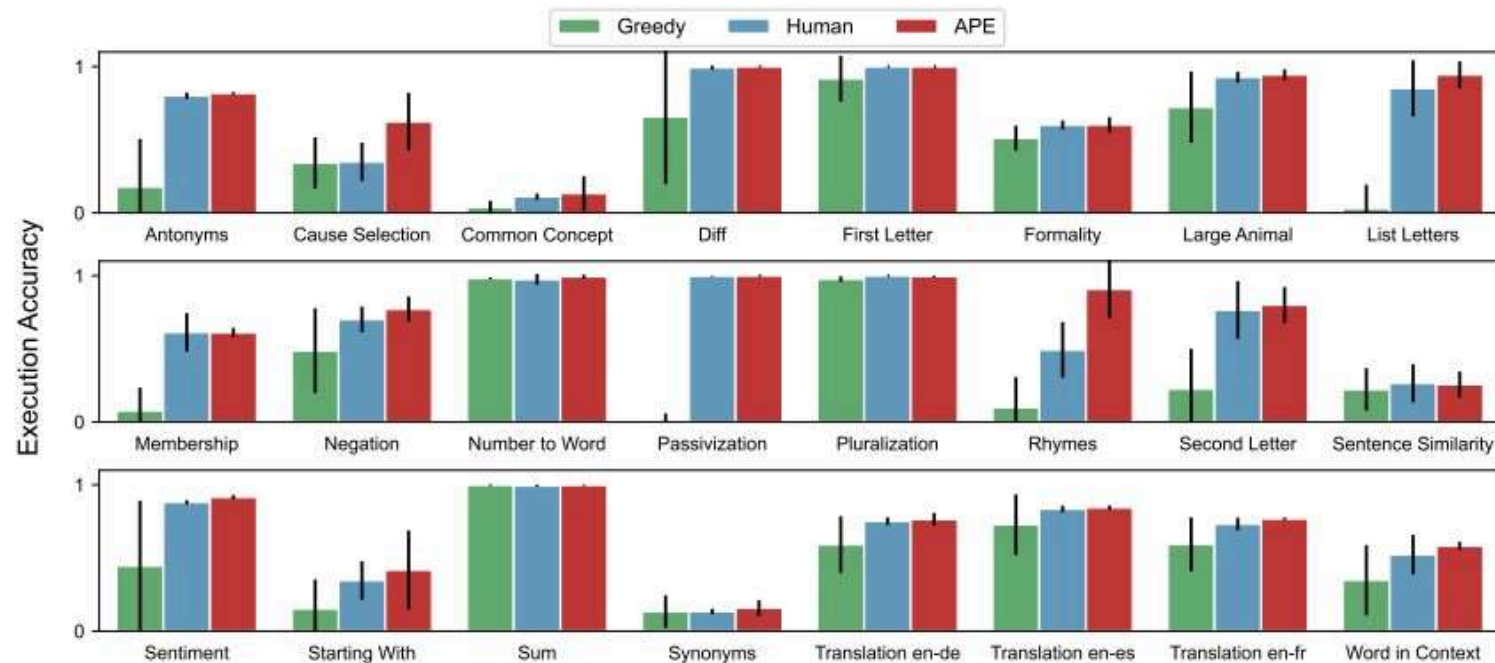
Instruction: write the antonym of the word. <LIKELIHOOD>

Input: direct **Output:** indirect



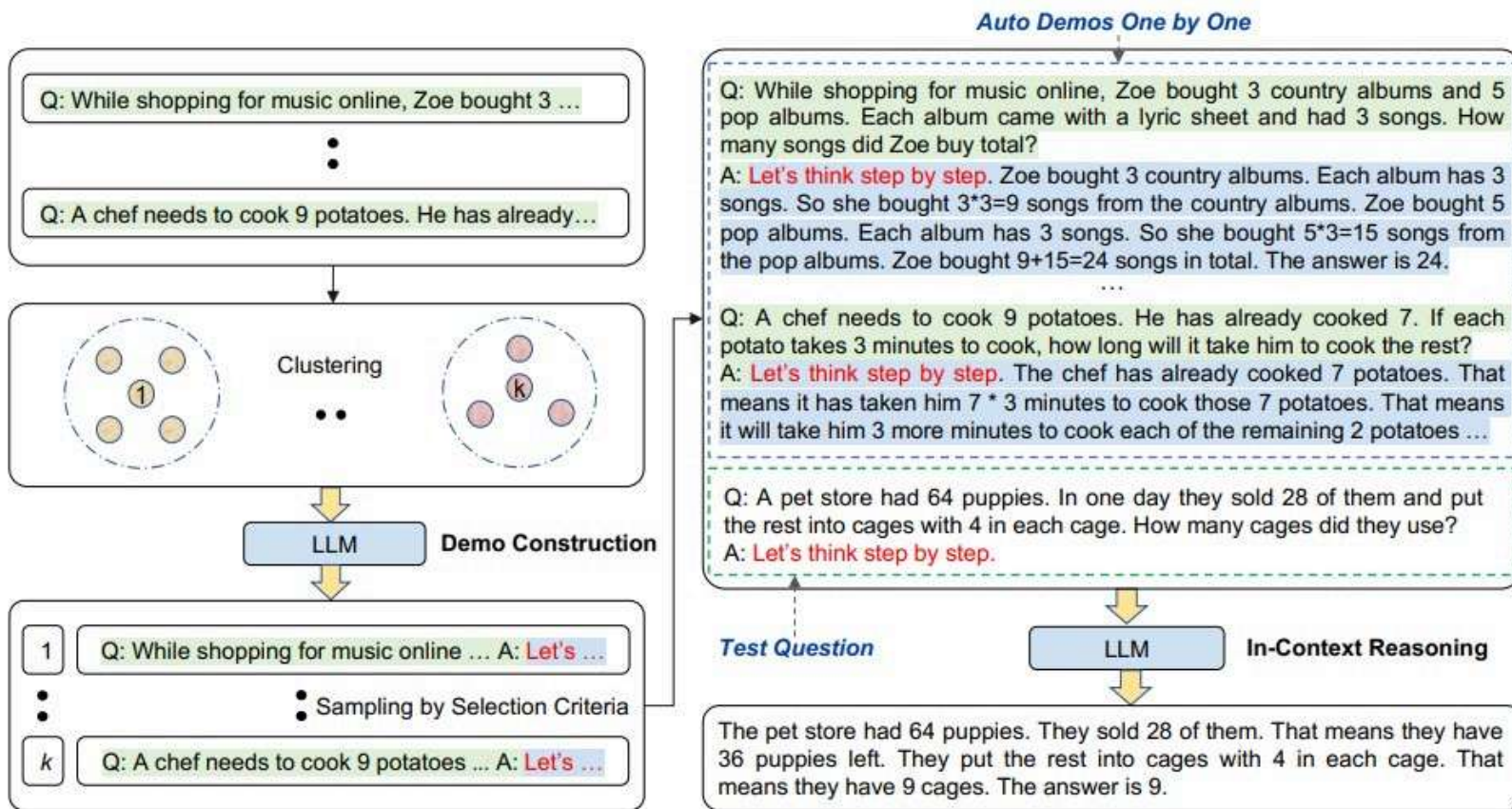
APE achieves human-level performance on zero-shot learning with model-generated instructions on 24/24 Instruction Induction and 17/21 Big-Bench tasks. It surpasses human performance with the InstructGPT model, obtaining an IQM of 0.810 compared to humans' 0.749. To achieve this, a dataset of questions and reasoning steps is generated using InstructGPT with the prompt "Let's think step by step." Then any data points that had incorrect answers were removed. Finally, APE was used to find a

prompt starting with "Let's" that maximized the likelihood of these correct reasoning steps. APE produced the prompt "Let's work this out in a step-by-step way to be sure we have the right answer." This generated prompt further improved performance on two tasks: MultiArith from 78.7 to 82.0, and GSM8K from 40.7 to 43.0.



Auto-CoT

Auto-CoT is a process of automatically constructing demonstrations with questions and reasoning chains. It first clusters the questions in a dataset into a few clusters. Then, it selects a representative question from each cluster and generates its reasoning chain using Zero-Shot-CoT with simple heuristics. The Auto-CoT method has several advantages over other methods. It is automatic, scalable, and effective, which means that it generates demonstrations that are accurate and informative.

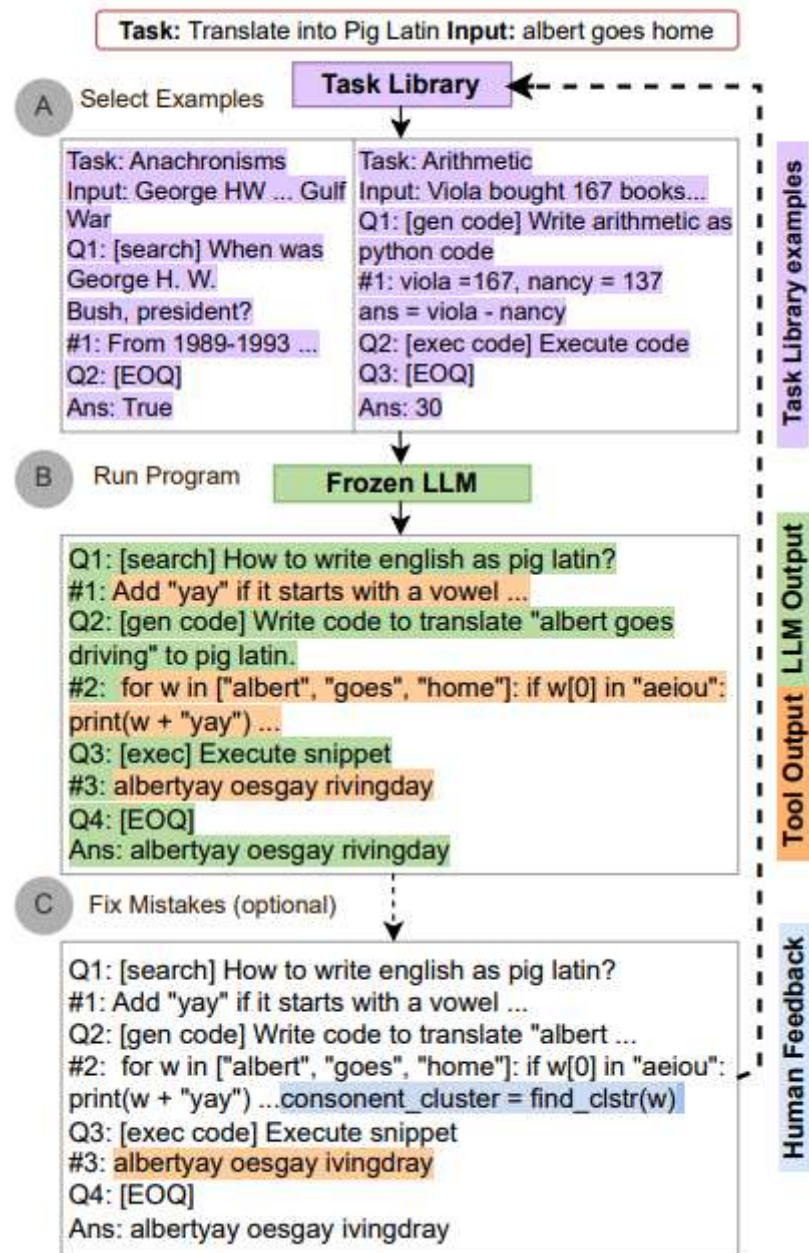


On comparing the accuracy of Auto-CoT with the four baseline methods on ten datasets from three categories of reasoning tasks, Auto-CoT consistently matches or exceeds the performance of the CoT that requires manual designs of demonstrations. The reason for this is that Auto-CoT is able to generate demonstrations that are task-adaptive. It means that the demonstrations are tailored to the specific dataset and reasoning task. In contrast, Manual-CoT may use the same demonstrations for multiple datasets, which can lead to lower accuracy.

Automatic Multi-step Reasoning and Tool-use (ART)

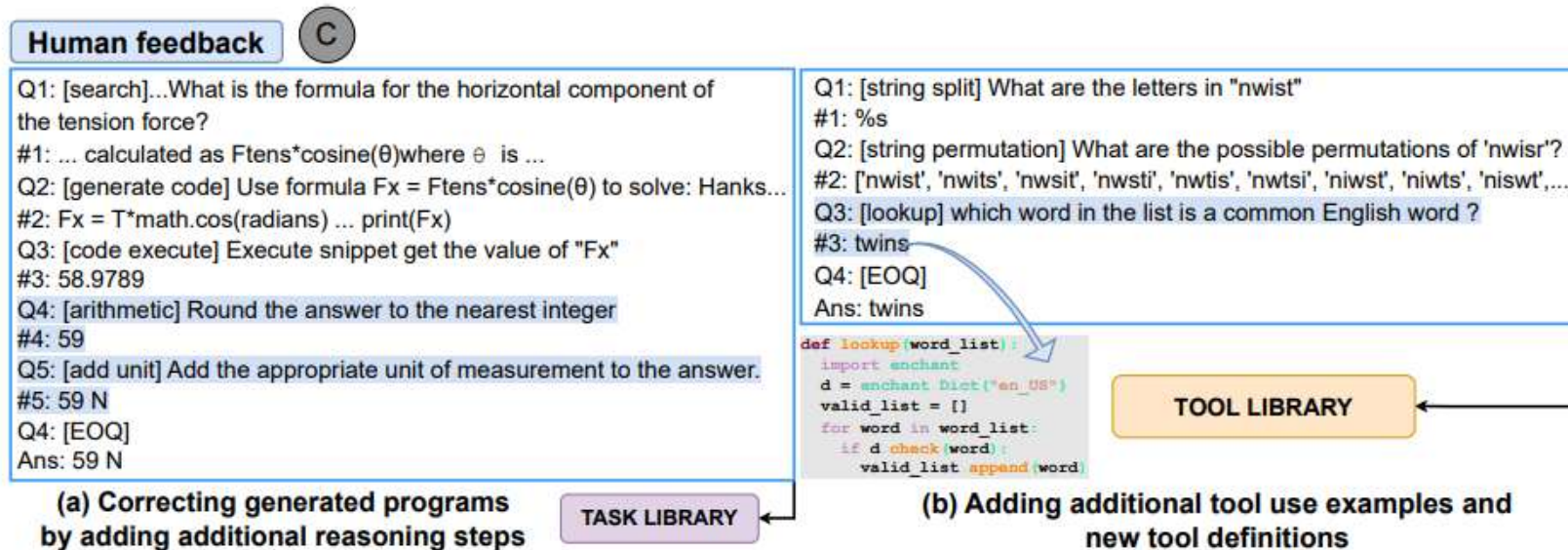
ART is a framework that uses large language models to automatically generate intermediate reasoning steps for a new task. The LLMs are frozen, which means that they are not updated during

the reasoning process. It allows ART to be more efficient and scalable than frameworks that use trainable LLMs. ART selects demonstrations of multistep reasoning and tool use from a task library. A decomposition is a high-level description of the steps involved in solving a task. ART then selects and uses tools in the tool library alongside LLM generation to complete the intermediate reasoning steps. At test time, ART seamlessly pauses generation whenever external tools are called, and integrates their output before resuming generation. This allows ART to leverage the capabilities of external tools to solve complex tasks.



ART has been shown to be effective on a variety of tasks, including natural language inference, question answering, and code generation. It outperforms previous approaches to few-shot reasoning and tool-use, and it is able to solve tasks that were previously thought to be impossible for LLMs.

Humans can optionally edit decompositions to improve performance. For example, they can correct errors in code or incorporate new tools. ART is extensible, which means that it can be easily extended to include new tasks and tools.



ART consistently matched or outperformed automatically generated CoT reasoning chains on 32 out of 34 BigBench tasks and all MMLU tasks. On average, it achieved an improvement of over 22 percentage points. The use of tools in ART significantly enhanced performance on test tasks, with an average improvement of over 12.3 percentage points compared to scenarios where no tools were allowed. ART also improved over direct few-shot prompting by an average of 10.8 percentage points across unseen BigBench and MMLU tasks. Its improvements were particularly remarkable in tasks requiring arithmetic and algorithmic reasoning, where it outperformed direct few-shot prompting by 12.5%. ART also surpassed previous best-known results for GPT3, which use supervision for decomposition and/or tool use, by 6.1 percentage points. ART allows for human intervention and performance improvement by updating the task and tool libraries with new demonstrations. With additional human feedback, ART surpassed the best-known results for GPT3 by an average of over 20 percentage points on 12 test tasks.