

Language Agent Basics

Arafat Rahman & Md Sabbir Ahmed
{jgh6ds, wyd2hu}@virginia.edu

CS 6501: Natural Language
Processing March 24, 2025

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

Primer

- **Chain of Thought (CoT):** "breaking down complex tasks into a sequence of logical steps towards a final solution". Example:

- A person had a depression score of 10 last week. This week, their score increased to 20. Based on this change, has their mental health improved or declined?

1. Understand what the score means; 2. Compare the two scores;
3. Analyze the change; 4. Draw the conclusion

- **Acting:** "action plan generation". Example:

- Since the depression score has increased, let's gather more information about the user's preferences.
 - Would you like recommendations for a psychologist or psycho counsellor, or perhaps information on self-help strategies?



Photo source: online

What are the gaps in the relevant works?

Abilities for reasoning (e.g. CoT prompting) and acting have primarily been studied as **separate** topics.

CoT

- CoT reasoning is a static black box, which limits ability to update its knowledge.
- Can lead to issues like fact hallucination and error propagation



Photo source: Online

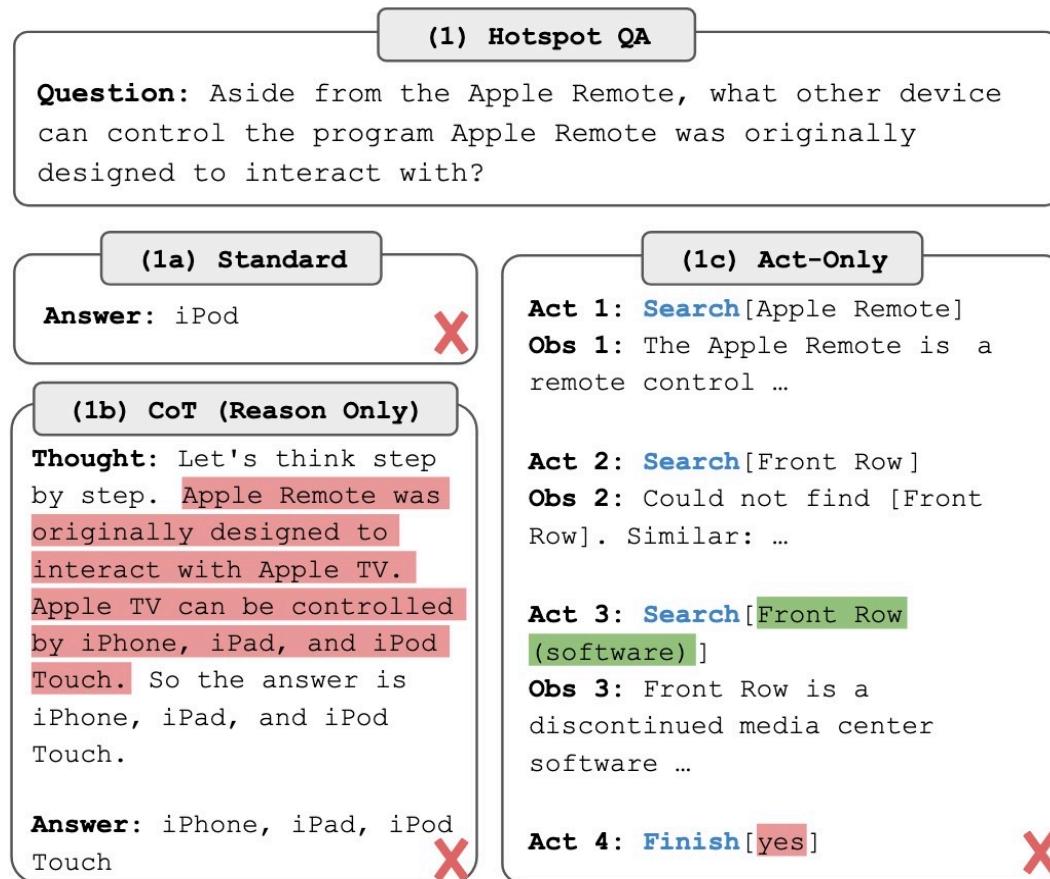
Acting

- Recent work uses pre-trained LLMs for planning and acting in interactive settings.
- Do not use LLMs for abstract reasoning or memory-based action support.

Then, what is the problem if we have reason & act seperately? 🤔

1. How does the separate approach perform for Q&A and fact verification?
2. Are the tasks hard for the separate approach?

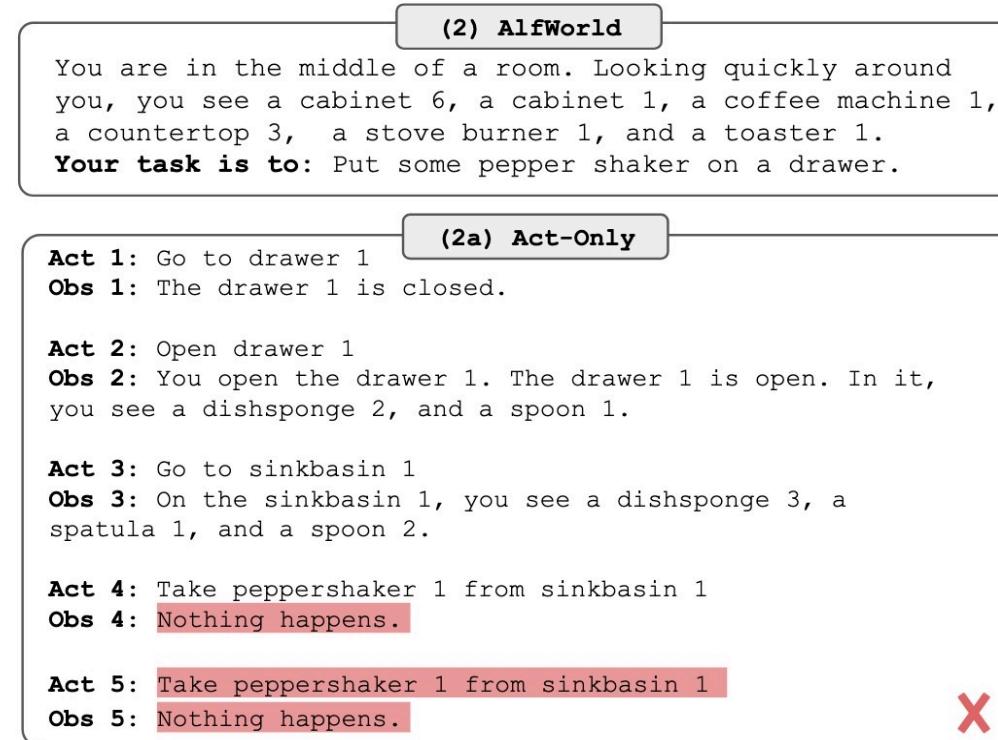
1. How does the separate approach perform for Q&A?



In simpler form: Is there any other device (besides the Apple Remote) that can also control that same program?

Figure 1: (1) Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT,

1. How does the separate approach perform for fact verification?



question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

So, how does the separate approach perform for Q&A and fact verification?

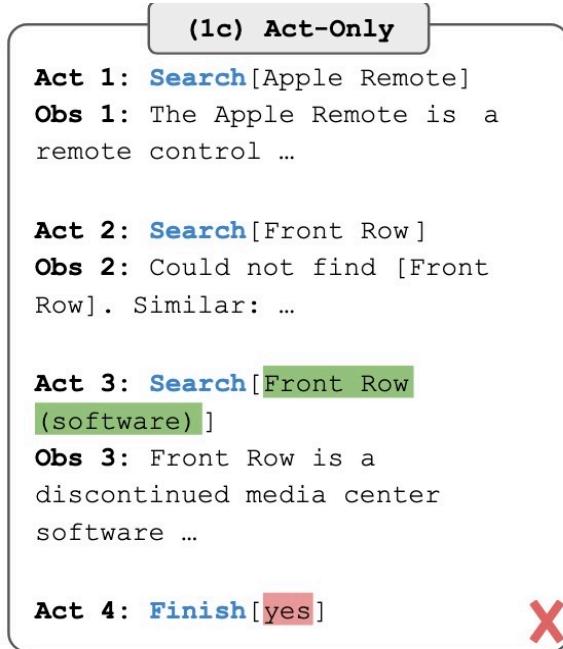
Not great, right? 😞 😊

But, why is it hard for the separate approach? 🤔

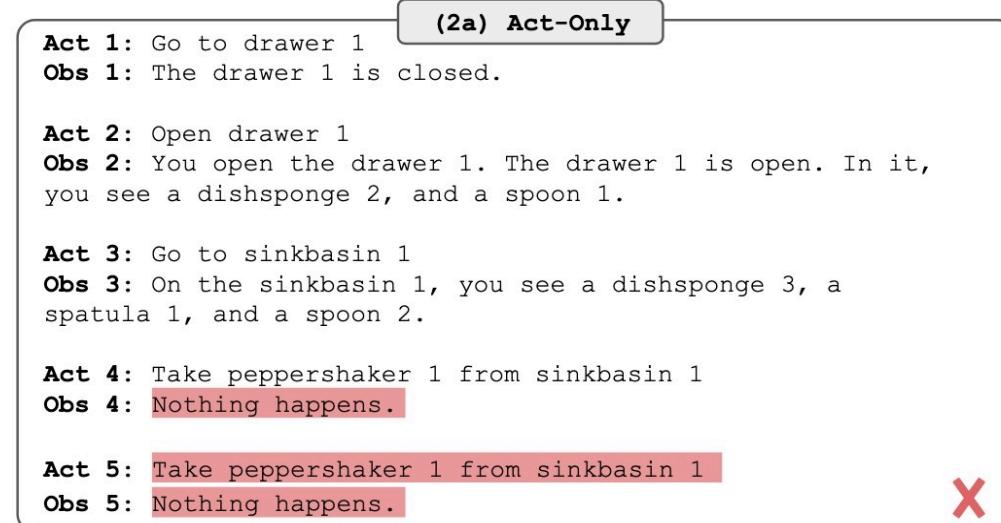
2. Why is it hard for the separate approach?

- Think, an agent interacting with an environment for task solving
- Receives an observation $o_t \in O$ from environment -> an action $a_t \in A$
- policy $\pi(a_t|c_t)$, $c_t = (o_1, a_1, \dots, o_{t-1}, a_{t-1}, o_t)$ is the context to agent.
- Learning a policy is challenging when the mapping $c_t \rightarrow a_t$ is highly ***implicit*** and requires ***extensive computation***

2. Why is it hard for the separate approach? - Recapping



Unable to generate the correct final action



Keep producing hallucinating actions 😞



Let's Mix Reasoning & Acting

ReAct 😊



Photo source: Online

ReAct: Reasoning + Acting

- Augment the agent's action space to $\hat{A} = A \cup L$; L :space of language
- An action $\hat{a}_t \in L$ in the language space, \hat{a}_t : a reasoning trace
- Does not affect the external environment which may help the agent to think before acting and reduce errors.
- \hat{a}_t aims to reason over c_t and update $c_{t+1} = (c_t, \hat{a}_t)$ for future reasoning or acting.

Okay 😎

**But, how Does ReAct Perform in
the "Failed" Cases? 🤔**

Let's see?

How Does ReAct Perform in the "Failed" Cases?

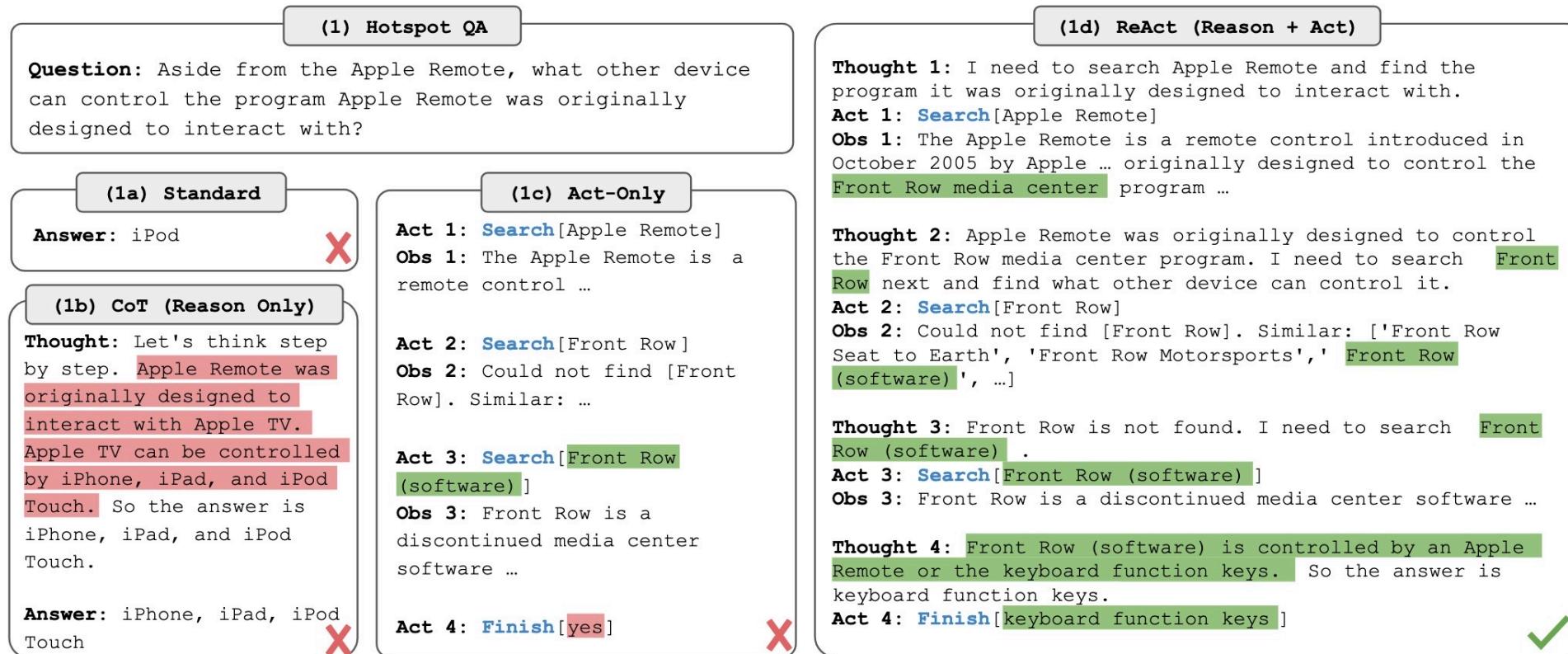
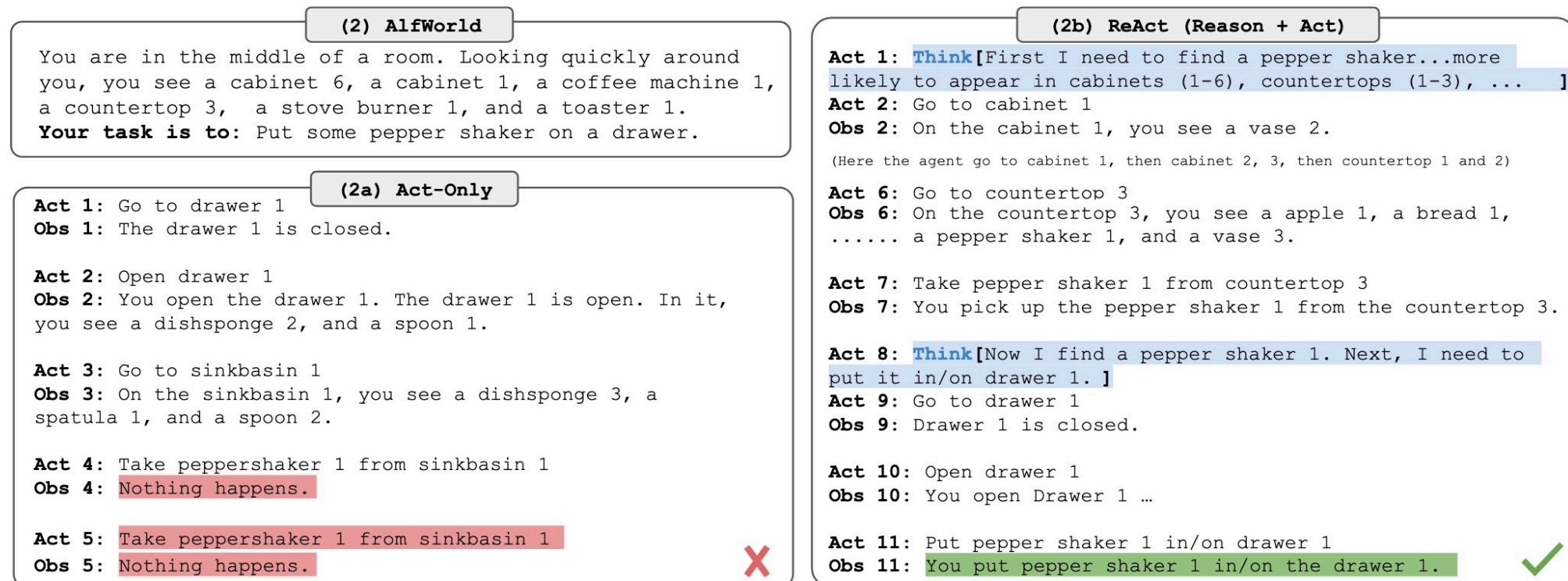


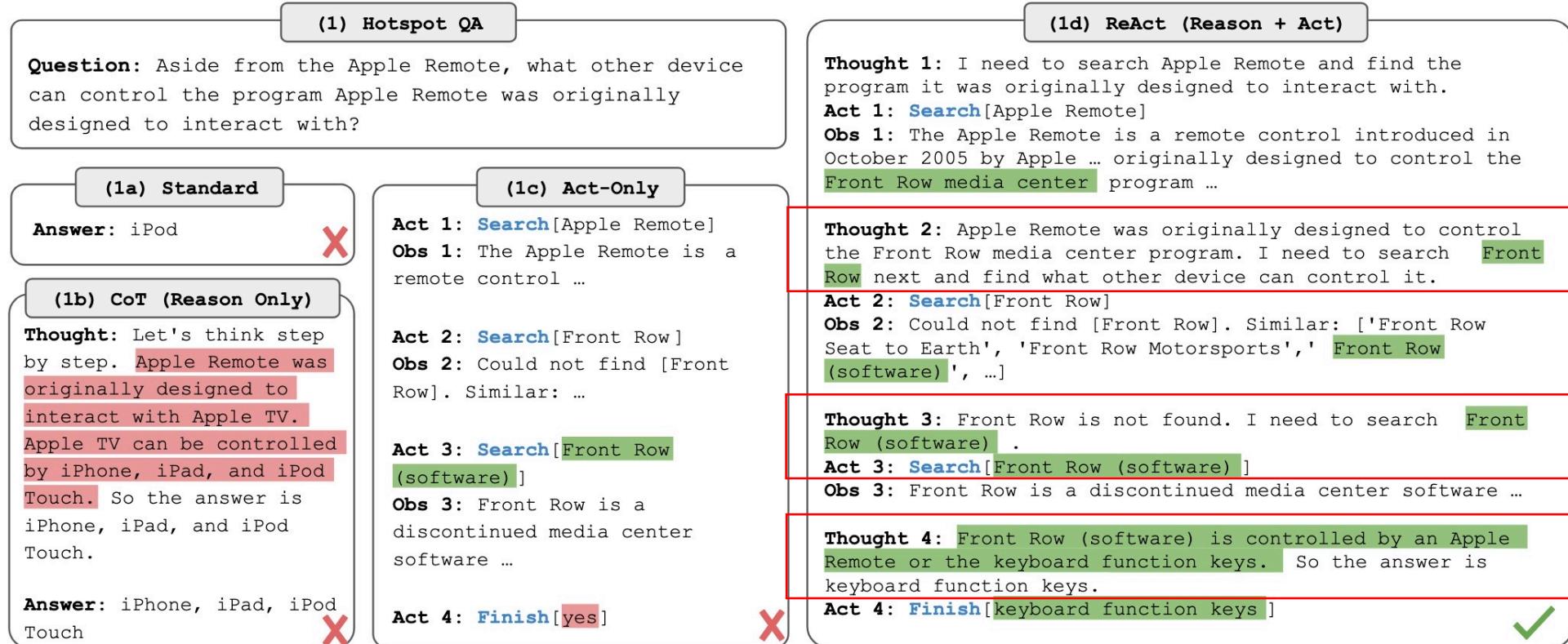
Figure 1: (1) Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT, Reason Only), (c) Act-only, and (d) ReAct (Reason+Act), solving a HotpotQA (Yang et al., 2018) question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

How Does ReAct Perform in the "Failed" Cases?



question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

Intuitively, Why Did ReAct Work Better?

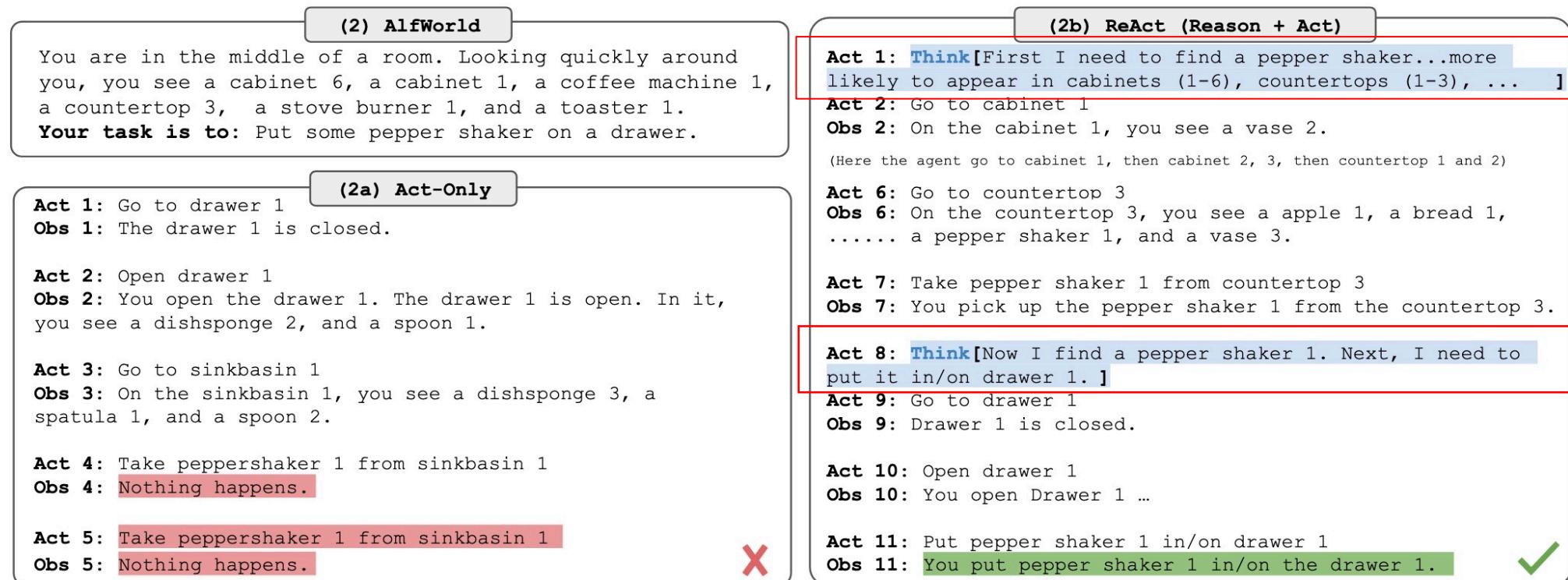


parts from observations extracted

handled exceptions and adjusted action plans

Figure 1: (1) Comparison of 4 prompting methods, (a) Standard, (b) Chain-of-thought (CoT, Reason Only), (c) Act-only, and (d) ReAct (Reason+Act), solving a HotpotQA (Yang et al., 2018) question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

Intuitively, Why Did ReAct Work Better?



question. (2) Comparison of (a) Act-only and (b) ReAct prompting to solve an AlfWorld (Shridhar et al., 2020b) game. In both domains, we omit in-context examples in the prompt, and only show task solving trajectories generated by the model (Act, Thought) and the environment (Obs).

How Much Better is ReAct?

Prompt Method ^a	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC → ReAct	34.2	64.6
ReAct → CoT-SC	35.1	62.0
Supervised SoTA^b	67.5	89.5

Table 1: PaLM-540B prompting results on HotpotQA and Fever.

^a HotpotQA EM is 27.1, 28.9, 33.8 for Standard, CoT, CoT-SC in Wang et al. (2022b).

^b (Zhu et al., 2021; Lewis et al., 2020)

SC: self-consistency. Sampling 21 CoT trajectories with decoding temperature 0.7 during inference and adopting the majority answer.

CoT-SC → ReAct: if no single answer of the majority answer appear at least n/2 (*it may mean the model is not confident or internal reasoning is not consistent*), back off to ReAct

ReAct → CoT-SC: when ReAct fails to return an answer within given steps , back off to CoT-SC. Steps: 7 and 5 steps for HotpotQA and Fever.

Act and CoT Examples (Hotpot QA)

Hotpot QA Prompts

Original Question

What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?
1,800 to 7,000 ft

Act

Question

What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?

Action 1

Search[Colorado orogeny]

Observation 1

The Colorado orogeny was an episode of mountain building (an orogeny) in Colorado and surrounding areas.

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.

Action 3

Search[High Plains]

Observation 3

High Plains refers to one of two distinct land regions:

Action 4

Search[High Plains (United States)]

Observation 4

The High Plains are a subregion of the Great Plains. From east to west, the High Plains rise in elevation from around 1,800 to 7,000 ft (550 to 2,130 m). [3]

Action 5

Finish[1,800 to 7,000 ft]

CoT

Question

What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?

Thought

Let's think step by step. The eastern sector of Colorado orogeny extends into the High Plains. High Plains rise in elevation from around 1,800 to 7,000 ft, so the answer is 1,800 to 7,000 ft.

Answer

1,800 to 7,000 ft



ReAct Examples (Hotpot QA)

ReAct

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]

Observation 3
High Plains refers to one of two distinct land regions

Thought 4
I need to instead search High Plains (United States).

Action 4
Search[High Plains (United States)]

Observation 4
The High Plains are a subregion of the Great Plains. From east to west, the High Plains rise in elevation from around 1,800 to 7,000 ft (550 to 2,130 m). [3]

Thought 5
High Plains rise in elevation from around 1,800 to 7,000 ft, so the answer is 1,800 to 7,000 ft.

Action 5
Finish[1,800 to 7,000 ft]



Act Examples (Fever)

Original

Determine if there is Observation that SUPPORTS or REFUTES a Claim, or if there is NOT ENOUGH INFORMATION.

Claim
Answer

Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.
SUPPORTS

Act

Determine if there is Observation that SUPPORTS or REFUTES a Claim, or if there is NOT ENOUGH INFORMATION.

Claim
Action2
Observation 1

Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.
Search[Nikolaj Coster-Waldau]
Nikolaj William Coster-Waldau (born 27 July 1970) is a Danish actor and producer. He graduated from the Danish National School of Performing Arts in Copenhagen in 1993,[1] and had his breakthrough role in Denmark with the film Nightwatch (1994). He played Jaime Lannister in the HBO fantasy drama series Game of Thrones, for which he received two Primetime Emmy Award nominations for Outstanding Supporting Actor in a Drama Series.. Coster-Waldau has appeared in numerous films in his native Denmark and Scandinavia, including Headhunters (2011) and A Thousand Times Good Night (2013). In the U.S, his debut film role was in the war film Black Hawk Down (2001), playing Medal of Honor recipient Gary Gordon.[2] He then played a detective in the short-lived Fox television series New Amsterdam (2008), and appeared in the 2009 Fox television film Virtuality, originally intended as a pilot.

Action 2

Finish[SUPPORTS]



CoT and ReAct Examples (Fever)

CoT

Determine if there is Observation that SUPPORTS or REFUTES a Claim, or if there is NOT ENOUGH INFORMATION.

Claim
Thought

Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.
Nikolaj William Coster-Waldau appeared in the 2009 Fox television film Virtuality, so he has worked with the Fox Broadcasting Company.

Answer

SUPPORTS

ReAct

Determine if there is Observation that SUPPORTS or REFUTES a Claim, or if there is NOT ENOUGH INFORMATION.

Claim
Thought 1

Nikolaj Coster-Waldau worked with the Fox Broadcasting Company.
I need to search Nikolaj Coster-Waldau and find if he has worked with the Fox Broadcasting Company.

Action 1
Observation 1

Search[Nikolaj Coster-Waldau]
Nikolaj William Coster-Waldau (born 27 July 1970) is a Danish actor and producer. He graduated from the Danish National School of Performing Arts in Copenhagen in 1993,[1] and had his breakthrough role in Denmark with the film Nightwatch (1994). He played Jaime Lannister in the HBO fantasy drama series Game of Thrones, for which he received two Primetime Emmy Award nominations for Outstanding Supporting Actor in a Drama Series.. Coster-Waldau has appeared in numerous films in his native Denmark and Scandinavia, including Headhunters (2011) and A Thousand Times Good Night (2013). In the U.S, his debut film role was in the war film Black Hawk Down (2001), playing Medal of Honor recipient Gary Gordon.[2] He then played a detective in the short-lived Fox television series New Amsterdam (2008), and appeared in the 2009 Fox television film Virtuality, originally intended as a pilot.

Thought 2

Because he "appeared in the 2009 Fox television film Virtuality", he should have worked with the Fox Broadcasting Company.

Action 2

Finish[SUPPORTS]



Photo source: Online

Performance – # of trials

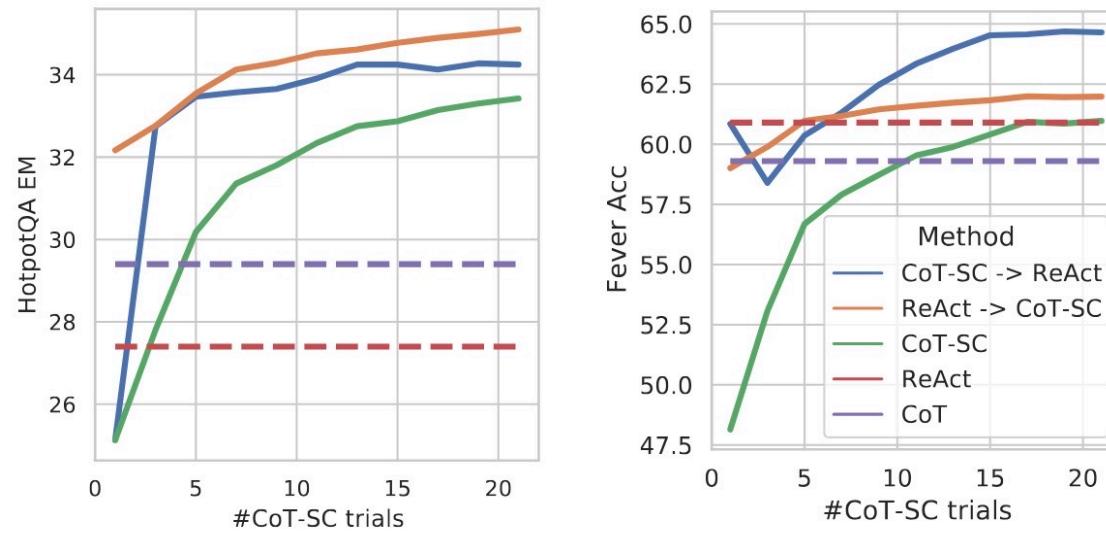


Figure 2: PaLM-540B prompting results with respect to number of CoT-SC samples used.

Performance in Human Evaluation

	Type	Definition	ReAct	CoT
Success	True positive	Correct reasoning trace and facts	94%	86%
	False positive	Hallucinated reasoning trace or facts	6%	14%
Failure	Reasoning error	Wrong reasoning trace (including failing to recover from repetitive steps)	47%	16%
	Search result error	Search return empty or does not contain useful information	23%	-
	Hallucination	Hallucinated reasoning trace or facts	0%	56%
	Label ambiguity	Right prediction but did not match the label precisely	29%	28%

Table 2: Types of success and failure modes of ReAct and CoT on HotpotQA, as well as their percentages in randomly selected examples studied by human.

Performance in Prompting and Finetuning

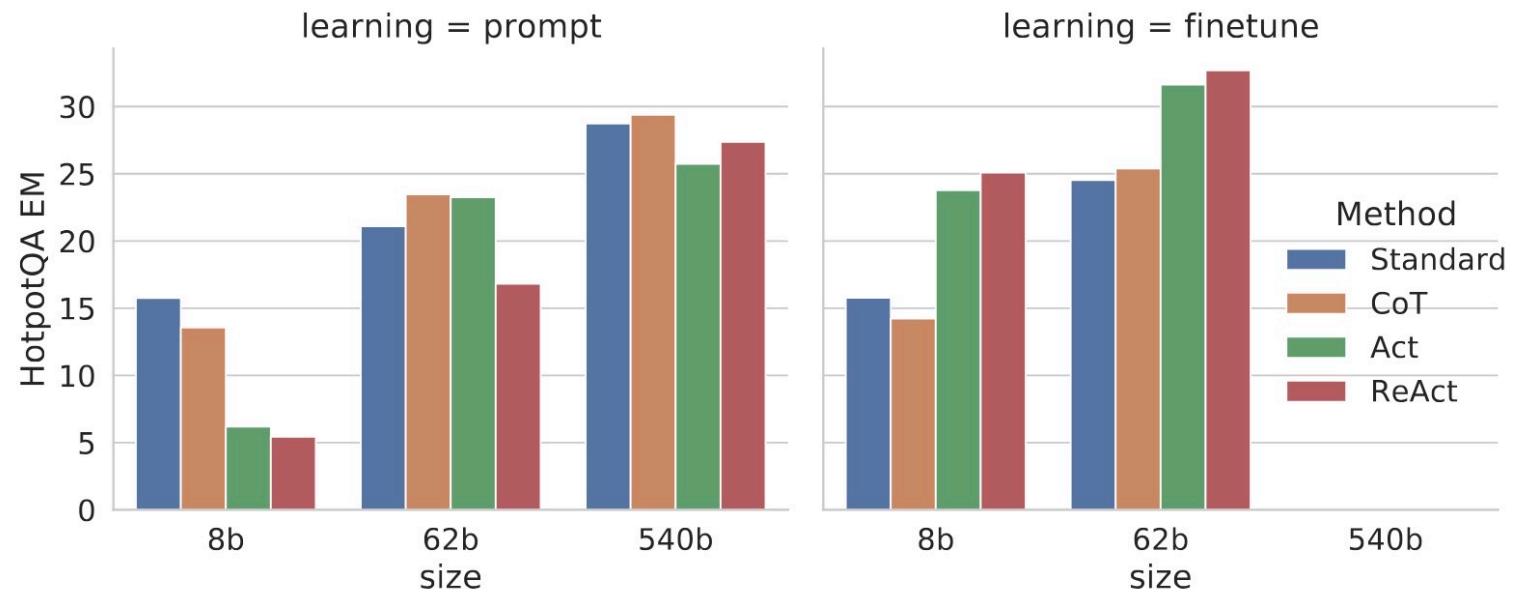


Figure 3: Scaling results for prompting and finetuning on HotPotQA with ReAct (ours) and baselines.

Performance in Interactive Decision Making Tasks

- Test ReAct on two language-based interactive decision-making task.
- ALFWorld is a synthetic text-based game
 - 6 types of tasks
 - task instance can have more than 50 locations and take an expert policy more than 50 steps to solve, thus challenging
 - For ReAct, annotate 3 trajectories from training for each task, where each trajectory has thoughts that (1) decompose the goal, (2) track subgoal completion, (3)determine the next subgoal, and (4) reason via commonsense where to find an object and what to do with it.

Act and ReAct Examples (ALFWORLD)

Table 7: An Act prompt on the ALFWORLD clean task. No thoughts are provided.

You are in the middle of a room. Looking quickly around you, you see a cabinet 13, a cabinet 12, a cabinet 11, a cabinet 10, a cabinet 9, a cabinet 8, a cabinet 7, a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeeemachine 1, a countertop 1, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1. Your task is to: put a clean lettuce in diningtable.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a cup 3, a egg 2, a potato 3, and a potato 2.
> go to diningtable 1
On the diningtable 1, you see a apple 1, a bread 1, a butterknife 2, a cup 2, a fork 2, a knife 2, a knife 1, a ladle 1, a lettuce 1, a mug 2, a mug 1, a pan 2, a peppershaker 1, a spatula 3, a tomato 2, and a tomato 1.
> take lettuce 1 from diningtable 1
You pick up the lettuce 1 from the diningtable 1.
> go to sinkbasin 1
On the sinkbasin 1, you see a apple 2, a ladle 2, a spoon 1, and a tomato 3.
> clean lettuce 1 with sinkbasin 1
You clean the lettuce 1 using the sinkbasin 1.
> think: Now I clean a lettuce (1). Next, I need to put it in/on diningtable 1.
OK.
> go to diningtable 1
On the diningtable 1, you see a apple 1, a bread 1, a butterknife 2, a cup 2, a fork 2, a knife 2, a knife 1, a ladle 1, a mug 2, a mug 1, a pan 2, a peppershaker 1, a spatula 3, a tomato 2, and a tomato 1.
> put lettuce 1 in/on diningtable 1
You put the lettuce 1 in/on the diningtable 1.

Act

> think: To solve the task, I need to find and take a lettuce, then clean it with sinkbasin, then put it in diningtable.
OK.
> think: First I need to find a lettuce. A lettuce is more likely to appear in fridge (1), diningtable (1), sinkbasin (1), stoveburner (1-3), cabinet (1-13). I can check one by one, starting with fridge 1.
OK.
> go to fridge 1
The fridge 1 is closed.
> open fridge 1
You open the fridge 1. The fridge 1 is open. In it, you see a cup 3, a egg 2, a potato 3, and a potato 2.
> go to diningtable 1
On the diningtable 1, you see a apple 1, a bread 1, a butterknife 2, a cup 2, a fork 2, a knife 2, a knife 1, a ladle 1, a lettuce 1, a mug 2, a mug 1, a pan 2, a peppershaker 1, a spatula 3, a tomato 2, and a tomato 1.
> think: Now I find a lettuce (1). Next, I need to take it.
OK.
> take lettuce 1 from diningtable 1
You pick up the lettuce 1 from the diningtable 1.
> think: Now I take a lettuce (1). Next, I need to go to sinkbasin (1) and clean it.
OK.
> go to sinkbasin 1
On the sinkbasin 1, you see a apple 2, a ladle 2, a spoon 1, and a tomato 3.
> clean lettuce 1 with sinkbasin 1
You clean the lettuce 1 using the sinkbasin 1.
> think: Now I clean a lettuce (1). Next, I need to put it in/on diningtable 1.
OK.
> go to diningtable 1
On the diningtable 1, you see a apple 1, a bread 1, a butterknife 2, a cup 2, a fork 2, a knife 2, a knife 1, a ladle 1, a mug 2, a mug 1, a pan 2, a peppershaker 1, a spatula 3, a tomato 2, and a tomato 1.
> put lettuce 1 in/on diningtable 1
You put the lettuce 1 in/on the diningtable 1.

ReAct

Performance in Interactive Decision Making Tasks

Method	Pick	Clean	Heat	Cool	Look	Pick 2	All
Act (best of 6)	88	42	74	67	72	41	45
ReAct (avg)	65	39	83	76	55	24	57
ReAct (best of 6)	92	58	96	86	78	41	71
ReAct-IM (avg)	55	59	60	55	23	24	48
ReAct-IM (best of 6)	62	68	87	57	39	33	53
BUTLER _g (best of 8)	33	26	70	76	17	12	22
BUTLER (best of 8)	46	39	74	100	22	24	37

Table 3: AlfWorld task-specific success rates (%). BUTLER and BUTLER_g results are from Table 4 of Shridhar et al. (2020b). All methods use greedy decoding, except that BUTLER uses beam search.

Can React Interact With Noisy Real-World for Practical App.?

Method	Score	SR
Act	62.3	30.1
ReAct	66.6	40.0
IL	59.9	29.1
IL+RL	62.4	28.7
Human Expert	82.1	59.6

Table 4: Score and success rate (SR) on Webshop. IL/IL+RL taken from Yao et al. (2022).

- Webshop: Online shopping website environment with 1.18M real-world products and 12k human instructions.
- Has a high variety of structured and unstructured texts (e.g. product titles, descriptions, and options crawled from Amazon).
- Requires an agent to purchase a product based on a user instruction (e.g. “I am looking for a nightstand with drawers. It should have a nickel finish, and priced lower than \$140”) through web interactions (e.g. search “nightstand drawers”, choose buttons such as “color: modern-nickel-white” or “back to search”).

IL: Imitation Learning

RL: Reinforcement Learning

Average score: percentage of desired attributes covered by the chosen product averaged across all episodes.

Success rate: percentage of episodes where the chosen product satisfies all requirements.

Contributions

- A novel prompt-based paradigm to synergize reasoning and acting in language models for general task solving
- Perform extensive experiments across diverse benchmarks to showcase the advantage of ReAct in a few-shot learning setup over prior approaches that perform either reasoning or action generation
- Present systematic ablations and analysis to understand the importance of acting in reasoning tasks, and reasoning in interactive tasks
- Analyze the limitations of ReAct under the prompting setup (i.e. limited support of reasoning and acting behaviors), and perform initial finetuning experiments showing the potential of ReAct to improve with additional training data.

Limitations & Ways to Improve

- ReAct has more reasoning error rate than CoT
 - Combine ReAct -> CoT-SC?
- ReAct has significantly lower performance than supervised SoTA (Zhu et al., 2021) in prompting
 - Find out the problems where supervised SoTA performs better and leverage the idea of supervised approach in ReAct?
 - annotate 3 trajectories from training for each task, where each trajectory has thoughts that (1) decompose the goal, (2) track subgoal completion, (3) determine the next subgoal, and (4) reason via commonsense where to find an object and what to do with it.

Thanks for Listening 😊

Let's move to the next paper

Toolformer: Language Models Can Teach Themselves to Use Tools

Timo Schick Jane Dwivedi-Yu Roberto Dessì[†] Roberta Raileanu

Maria Lomeli Luke Zettlemoyer Nicola Cancedda Thomas Scialom

Meta AI Research [†]Universitat Pompeu Fabra

Introduction

- LLM struggles:
 - Basic works (arithmetic, factual lookup)
 - No access to up-to-date information
 - Hallucinate facts
 - Low-resource language
 - Unawareness of time progression

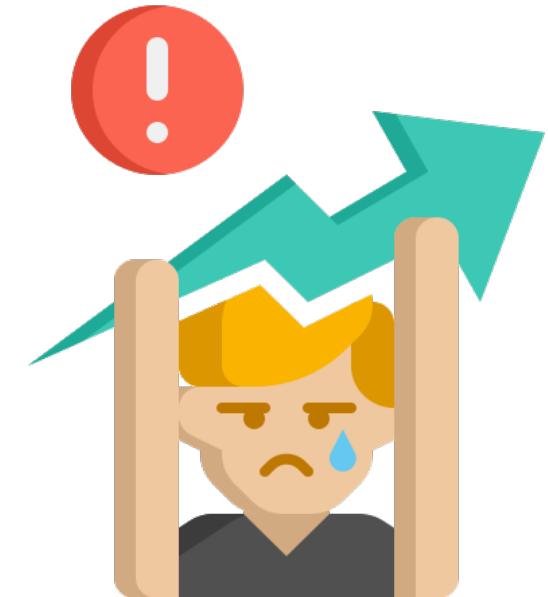
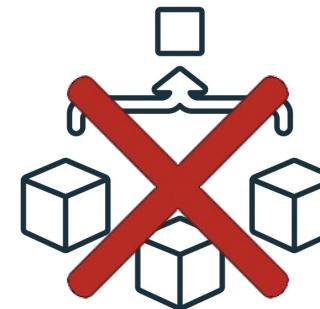


Photo source: Online

- How to overcome?
 - Use external tools (search engines, calculators, calendars)

Research Gap

- Existing tool utilization approaches require either:
 - Large amounts of human annotations
 - Task-specific settings that limit general application



GENERALIZATION

Photo source: Online

Tool former

- Tool former -> a model for effective API calling.
- Tool former uses API as tools to solve problem.

- Tool former introduces a self-supervised approach:
 - Learn when and how to use tools independently.
 - Maintain general capabilities while accessing specialize functions.

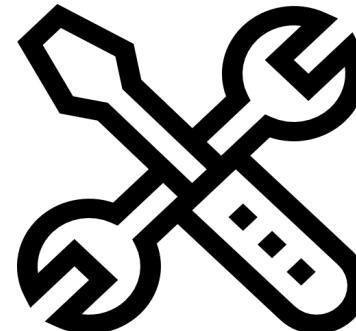


Photo source: Online

Major Contributions

- Self-supervised learning framework for tool use without human annotation.
- Improved performance of a 6.7B parameter model, outperforming much larger models.
- Integration of five diverse tools: QA systems, calculators, search engines, translation systems, and calendars.

Example Prediction

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator($400 / 1400$) → 0.29] 29%) passed the test.

The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.



Photo source: online

Figure 1: Exemplary predictions of Toolformer. The model autonomously decides to call different APIs (from top to bottom: a question answering system, a calculator, a machine translation system, and a Wikipedia search engine) to obtain information that is useful for completing a piece of text.

Technical Implementation

- Three-step process:
 1. **Sample** potential API calls using in-context learning
 2. **Execute calls** and filter based on perplexity reduction.
 3. **Finetune** model on helpful API interactions
- Inference mechanism that interrupts generation to make API calls when needed.

Technical Implementation

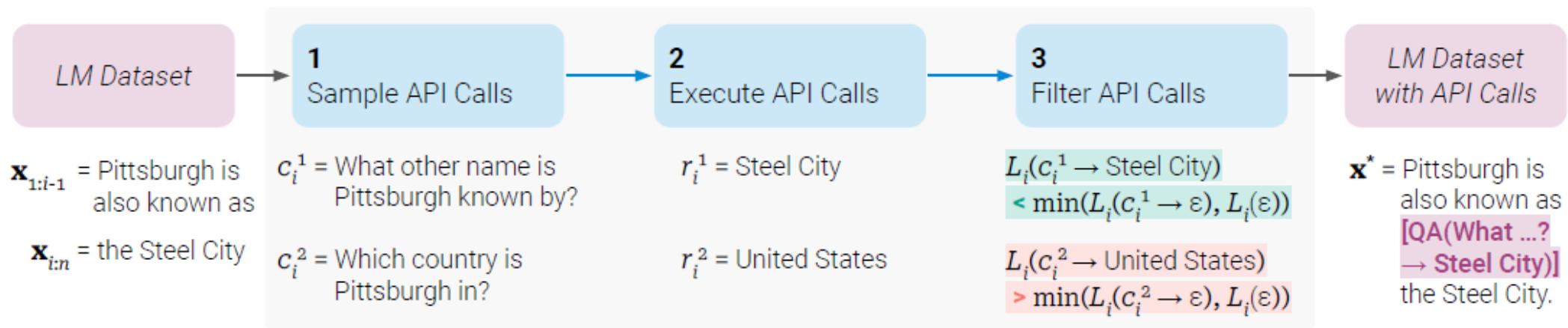


Figure 2: Key steps in our approach, illustrated for a *question answering* tool: Given an input text x , we first sample a position i and corresponding API call candidates $c_i^1, c_i^2, \dots, c_i^k$. We then execute these API calls and filter out all calls which do not reduce the loss L_i over the next tokens. All remaining API calls are interleaved with the original text, resulting in a new text x^* .

Sample API Calls

- Uses few-shot prompting with only a handful of human-written examples per API type.
- Computes probability of starting an API call at each text position.
- Filters positions using a sampling threshold.
- Keeps only top k positions with highest probability.



Photo source: Online

Executing API Calls

- Executes all API calls generated during sampling.
- Each API returns a single text response.
- Results feed into the filtering step.

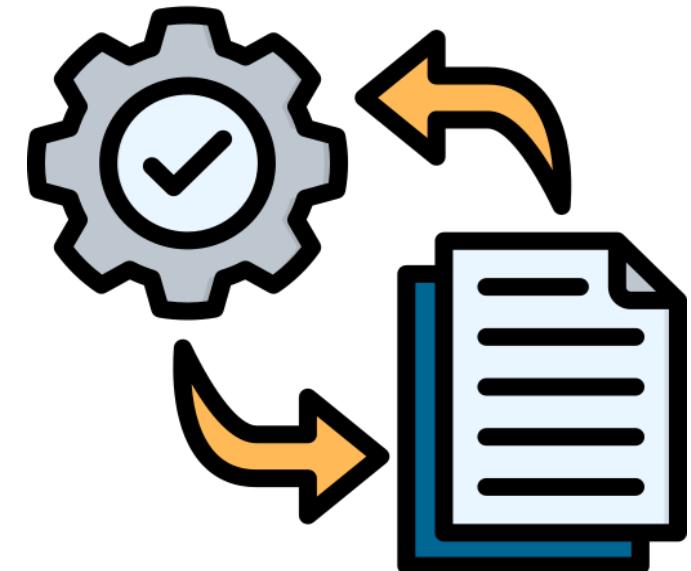


Photo source: Online

Filtering API Calls

- Let i be the position of the API call c_i in the sequence $x = x_1, \dots, x_n$, and let r_i be the response from API. Given a sequence $(w_i \mid i \in \mathbb{N})$ of weights, let

$$L_i(z) = - \sum_{j=i}^n w_{j-i} \cdot \log p_M(x_j \mid z, x_{1:j-1})$$

be the weighted cross entropy loss for M over the tokens $x = x_1, \dots, x_n$, if the model is prefixed with z .

- This loss refers to the weighted next token prediction loss over the sequence.
- Weights are manually defined such that API calls happen where it is helpful.

Filtering API Calls

- Calculates two loss values:
 - L_i^+ : Loss when API call and response are included
 - L_i^- : Minimum loss without API call or with only the call (no response)
- Keeps API calls where: $L_i^- - L_i^+ \geq \tau_f$
- After filtering, remaining API calls are interleaved with original text.
- This creates an augmented dataset for model finetuning.

Model Finetuning

- Finetunes GPT-J on the augmented dataset with filtered API calls.
- Preserves original language modeling objective.
- Dataset contains same content as original pretraining data, just with API calls added.
- Model autonomously learns when and how to use which tool.

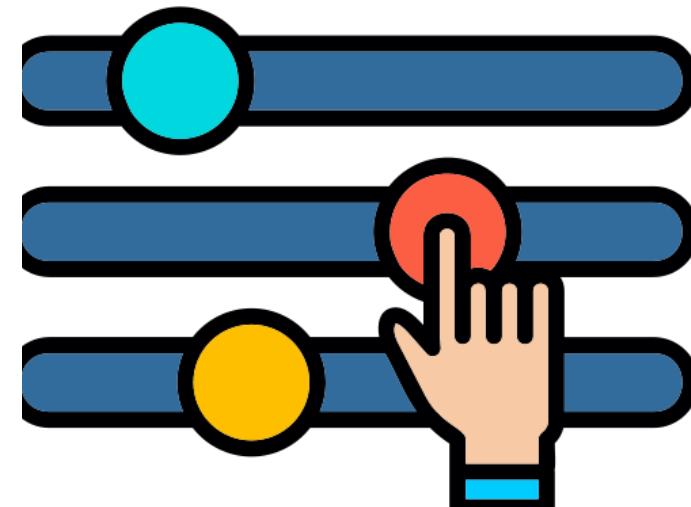


Photo source: Online

Experimental Results (Knowledge)

- Knowledge tasks: Significant improvements on LAMA benchmarks using QA tool.

Model	SQuAD	Google-RE	T-REx
GPT-J	17.8	4.9	31.9
GPT-J + CC	19.2	5.6	33.2
Toolformer (disabled)	22.1	6.3	34.9
Toolformer	<u>33.8</u>	<u>11.5</u>	<u>53.5</u>
OPT (66B)	21.6	2.9	30.1
GPT-3 (175B)	26.8	7.0	39.8

Table 3: Results on subsets of LAMA. Toolformer uses the question answering tool for most examples, clearly outperforming all baselines of the same size and achieving results competitive with GPT-3 (175B).



Photo source: Online

Experimental Results (Math)

- Math problems: 3-5x performance gains across multiple benchmarks with calculator

Model	ASDiv	SVAMP	MAWPS
GPT-J	7.5	5.2	9.9
GPT-J + CC	9.6	5.0	9.3
Toolformer (disabled)	14.8	6.3	15.0
Toolformer	<u>40.4</u>	<u>29.4</u>	<u>44.0</u>
OPT (66B)	6.0	4.9	7.9
GPT-3 (175B)	14.0	10.0	19.8

Table 4: Results for various benchmarks requiring mathematical reasoning. Toolformer makes use of the calculator tool for most examples, clearly outperforming even OPT (66B) and GPT-3 (175B).

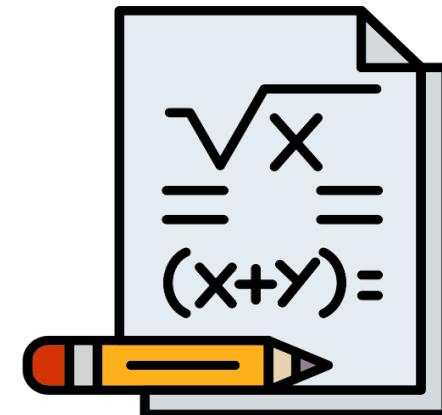


Photo source: Online

Experimental Results (QA)

- QA tasks: Substantial improvements relying on Wikipedia search

Model	WebQS	NQ	TriviaQA
GPT-J	18.5	12.8	43.9
GPT-J + CC	18.4	12.2	45.6
Toolformer (disabled)	18.9	12.6	46.7
Toolformer	26.3	17.7	48.8
OPT (66B)	18.6	11.4	45.7
GPT-3 (175B)	<u>29.0</u>	<u>22.6</u>	<u>65.9</u>

Table 5: Results for various question answering dataset. Using the Wikipedia search tool for most examples, Toolformer clearly outperforms baselines of the same size, but falls short of GPT-3 (175B).



Photo source: Online

Experimental Results (Multilingual)

- Multilingual capabilities: Enhanced performance through translation tool

Model	Es	De	Hi	Vi	Zh	Ar
GPT-J	15.2	16.5	1.3	8.2	18.2	8.2
GPT-J + CC	15.7	14.9	0.5	8.3	13.7	4.6
Toolformer (disabled)	19.8	11.9	1.2	10.1	15.0	3.1
Toolformer	20.6	13.5	1.4	10.6	16.8	3.7
OPT (66B)	0.3	0.1	1.1	0.2	0.7	0.1
GPT-3 (175B)	3.4	1.1	0.1	1.7	17.7	0.1
GPT-J (All En)	24.3	27.0	23.9	23.3	23.1	23.6
GPT-3 (All En)	24.7	27.2	26.1	24.9	23.6	24.0

Table 6: Results on MLQA for Spanish (Es), German (De), Hindi (Hi), Vietnamese (Vi), Chinese (Zh) and Arabic (Ar). While using the machine translation tool to translate questions is helpful across all languages, further pretraining on CCNet deteriorates performance; consequently, Toolformer does not consistently outperform GPT-J. The final two rows correspond to models that are given contexts and questions in English.



Photo source: Online

Experimental Results (Temporal)

- Temporal awareness: Calendar tool enabling date-related reasoning

Model	TEMPLAMA	DATESET
GPT-J	13.7	3.9
GPT-J + CC	12.9	2.9
Toolformer (disabled)	12.7	5.9
Toolformer	<u>16.3</u>	<u>27.3</u>
OPT (66B)	14.5	1.3
GPT-3 (175B)	15.5	0.8



Photo source: Online

Table 7: Results for the temporal datasets. Toolformer outperforms all baselines, but does not make use of the calendar tool for TEMPLAMA.

Scaling Laws

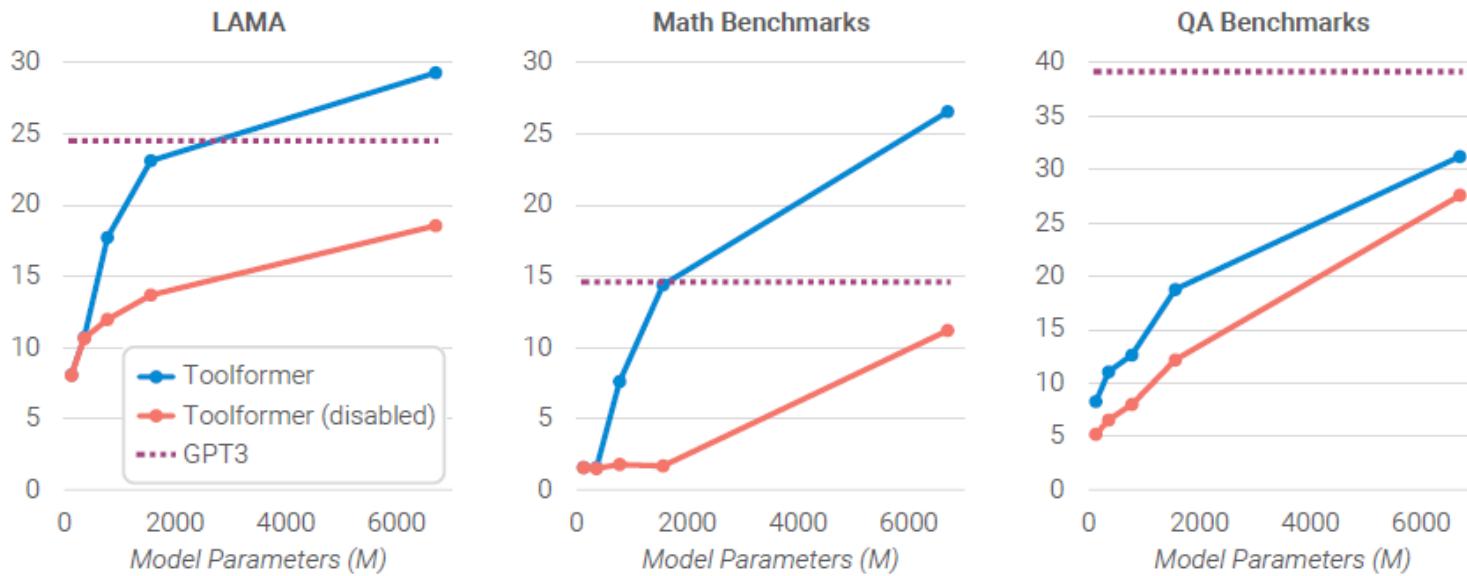


Figure 4: Average performance on LAMA, our math benchmarks and our QA benchmarks for GPT-2 models of different sizes and GPT-J finetuned with our approach, both with and without API calls. While API calls are not helpful to the smallest models, larger models learn how to make good use of them. Even for bigger models, the gap between model predictions with and without API calls remains high.

Analysis

- Decoding strategy impact:
 - Higher top-k values increase tool usage frequency
 - At k=1, model shows good calibration for when to use tools
- API call quality:
 - Perplexity reduction strongly correlates with usefulness
- Scale effects:
 - Tool use ability emerges around 775M parameters
 - Larger models show greater benefits from tools
- Language modeling preserved:
 - No degradation in base modeling capabilities

Limitations and Future Directions

- No support for chained tool use (i.e., using the output of one tool as an input for another tool).
- Lack of interactive tool capabilities.
- Sensitivity to exact wording in input.
- Very sample-inefficient.

Processing more than a million documents results in only a few thousand examples of useful calls to the calculator API.

Potential solution: Iteratively apply the approach similar to related bootstrapping approaches.

- Not considering tool-dependent computational cost.

Conclusions

- Demonstrates an alternative to simply scaling model size.
- Enables smaller models to compete with much larger ones.
- Provides a general framework extendable to many tools.
- Shows models can learn to use tools without human guidance.

Reflexion: Language Agents with Verbal Reinforcement Learning

Noah Shinn

Northeastern University

noahshinn024@gmail.com

Federico Cassano

Northeastern University

cassano.f@northeastern.edu

Edward Berman

Northeastern University

berman.ed@northeastern.edu

Ashwin Gopinath

Massachusetts Institute of Technology

agopi@mit.edu

Karthik Narasimhan

Princeton University

karthikn@princeton.edu

Shunyu Yao

Princeton University

shunyuy@princeton.edu

Primer

Reinforcement Learning (RL)
trains agents through trial,
error, and rewards.

Agents balance exploration of
new actions with exploitation
of known rewards.

Value functions estimate
future rewards to guide
optimal decisions.

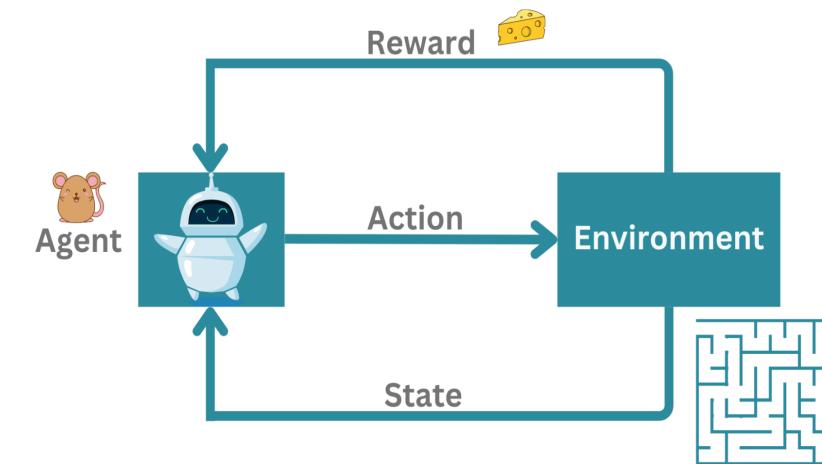


Photo source: online

Research Gaps



challenging for these language agents to quickly and efficiently learn from trial-and-error



Traditional RL requires intensive computation and time with gradient descent.



In-context examples can accelerate agent learning.

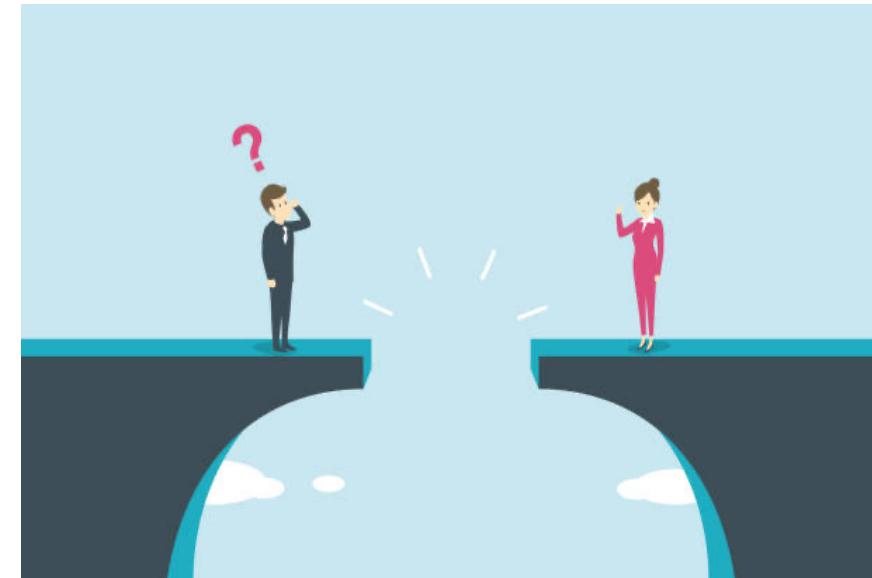


Photo source: online

Reflexion



Reflexion uses natural language feedback instead of weight updates.

Agents "reflect" on failures in words.

These reflections create memory to guide future decisions.

This mimics human learning from mistakes.

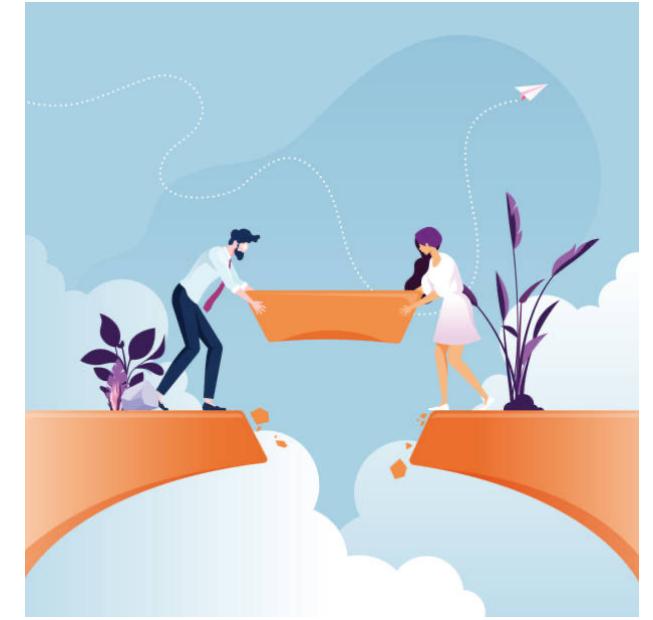
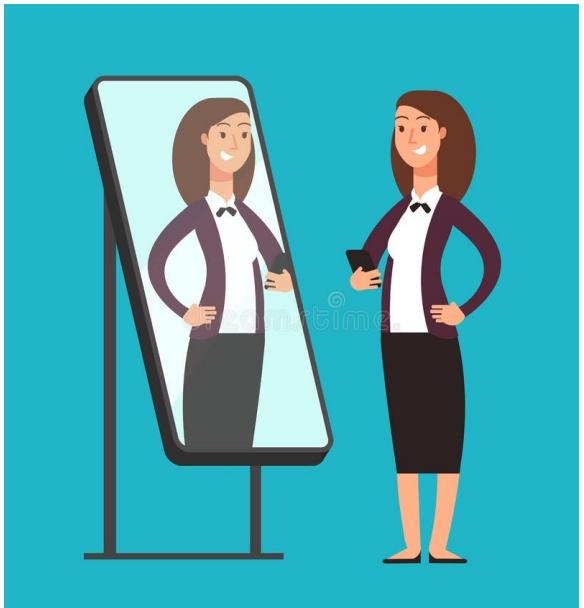


Photo source: online

Major Contributions

- Verbal linguistic feedback instead of complicated weight updates.
- Agents reflect on feedback, store text in memory, and learn complex tasks with small trials.
- Improves performance across tasks:
 - AlfWorld: +22% in 12 steps
 - HotPotQA: +20%
 - HumanEval: +11% (91% vs GPT-4's 80%)

Advantages of Reflexion

- Lightweight and works without fine-tuning.
- Nuanced form of feedback (e.g. targeted changes in action).
- It allows for a more explicit and interpretable form of episodic memory over prior experience.
- More explicit hints for future actions.



FEEDBACK

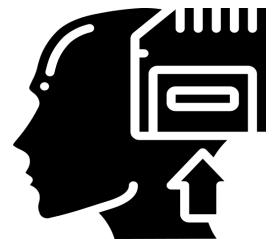


Photo source: online

Example Usage

	1. Decision making	2. Programming	3. Reasoning
(a) Task	You are in the middle of a room [...] Task: clean some pan and put it in countertop.	Task: You are given a list of two strings [...] of open '(' or close ')' parentheses only [...]	Task: What profession does John Lanchester and Alan Dean Foster have in common?
(b) Trajectory	[...] Action: take pan1 from stoveburner1 Obs: Nothing happens. [...] Action: clean pan1 with sinkbasin1 Obs: Nothing happens. [...]	def match_parens(lst): if s1.count('(') + s2.count('(') == s1.count(')') + s2.count(')'): [...] return 'No'	Think: [...] novelist, journalist, critic [...] novelist, screenwriter [...] common is novelist and screenwriter. Action: "novelist, screenwriter"
(c) Evaluation (internal / external)	Rule/LM Heuristic: Hallucination.	Self-generated unit tests fail: assert match_parens(...)	Environment Binary Reward: 0
(d) Reflection	[...] tried to pick up the pan in stoveburner 1 [...] but the pan was not in stoveburner 1. [...]	[...] wrong because it only checks if the total count of open and close parentheses is equal [...] order of the parentheses [...]	[...] failed because I incorrectly assumed that they both had the same multiple professions [...] accurately identifying their professions.
(e) Next Trajectory	[...] Action: take pan 1 from stoveburner 2 [...] Obs: You put the pan 1 in countertop 1.	[...] return 'Yes' if check(s1) or check(s2) else 'No'	Think: [...] So the profession John Lanchester and Alan Dean Foster have in common is novelist. Action: "novelist"

Figure 1: Reflexion works on decision-making 4.1, programming 4.3, and reasoning 4.2 tasks.

Technical Details (Core Components)

- **Actor:** LLM generates actions from observations.
- **Evaluator:** Scores outputs.
- **Self-Reflection:** Creates verbal feedback.
- **Memory:** Stores reflective text.

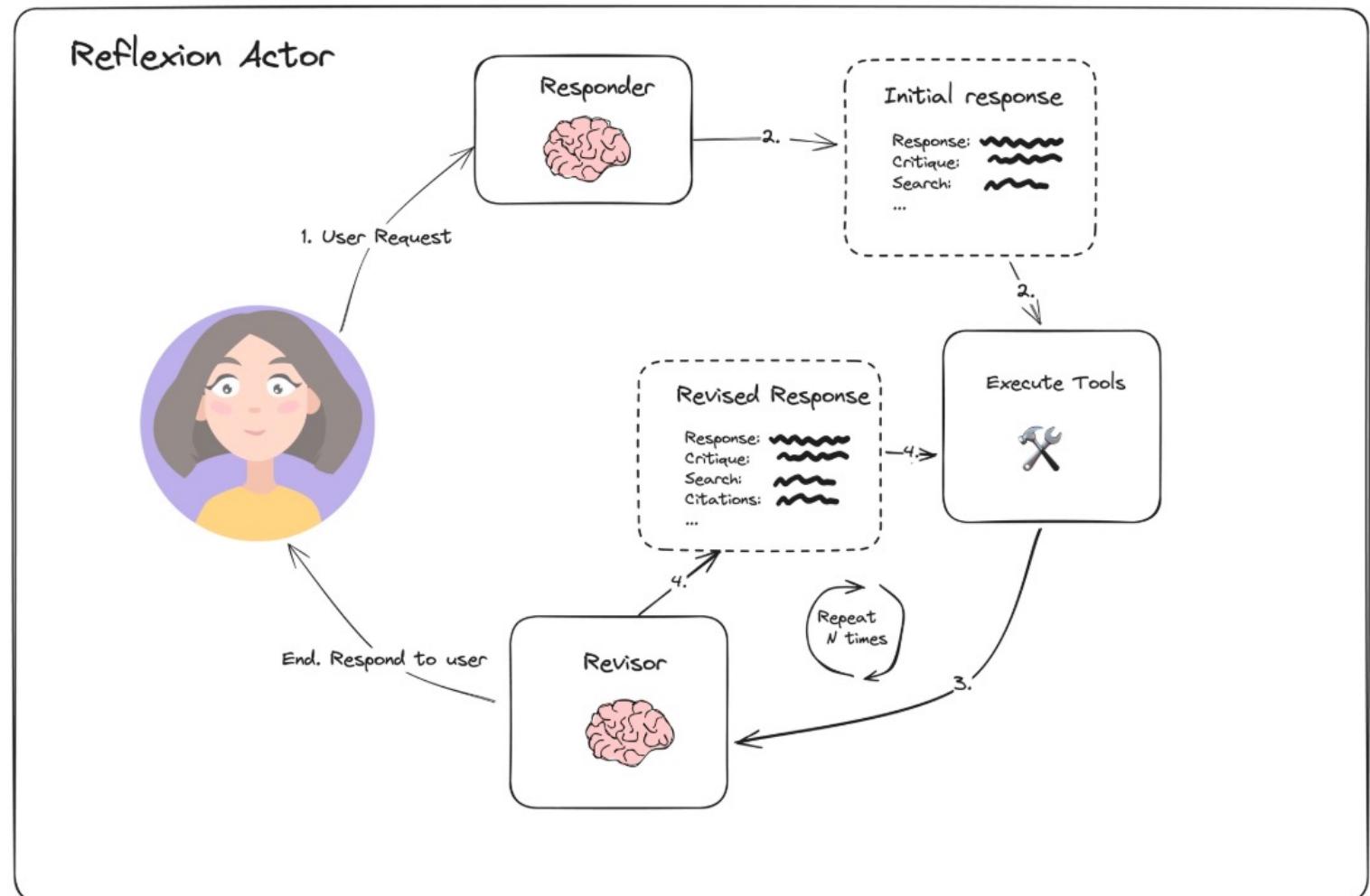
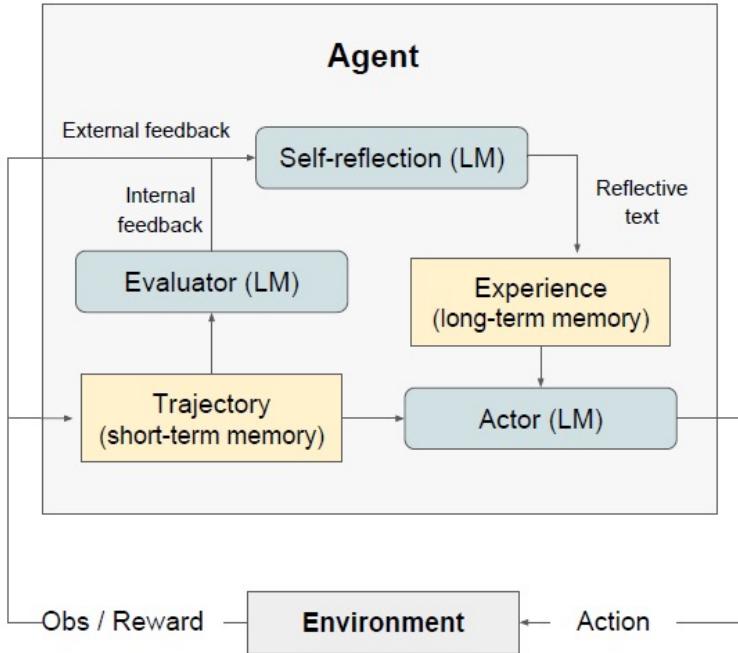


Photo source: online

Technical Details (Implementation)

- **Actor:**
 - Actors generate actions based on observations.
 - Incorporates memory components for additional context.
- **Evaluator:**
 - Assesses output quality by computing reward scores (Exact Match Grading reasoning, tailored heuristic functions for decision-making).
 - Also, different instantiation of LLM as an Evaluator.
- **Self-Reflection:**
 - Detailed verbal feedback that goes beyond binary success/failure signals.
 - Iterative process of trial, error, reflection, and memory storage enables rapid improvement in decision-making by leveraging past experiences.

Algorithm



Algorithm 1 Reinforcement via self-reflection

```
Initialize Actor, Evaluator, Self-Reflection:  
     $M_a, M_e, M_{sr}$   
Initialize policy  $\pi_\theta(a_i|s_i)$ ,  $\theta = \{M_a, mem\}$   
Generate initial trajectory using  $\pi_\theta$   
Evaluate  $\tau_0$  using  $M_e$   
Generate initial self-reflection  $sr_0$  using  $M_{sr}$   
Set  $mem \leftarrow [sr_0]$   
Set  $t = 0$   
while  $M_e$  not pass or  $t < \text{max trials}$  do  
    Generate  $\tau_t = [a_0, o_0, \dots, a_i, o_i]$  using  $\pi_\theta$   
    Evaluate  $\tau_t$  using  $M_e$   
    Generate self-reflection  $sr_t$  using  $M_{sr}$   
    Append  $sr_t$  to  $mem$   
    Increment  $t$   
end while  
return
```

Figure 2: (a) Diagram of Reflexion. (b) Reflexion reinforcement algorithm

Decision Making (AlfWorld)

- Used ReAct with heuristic and NLP evaluation using LLM.
- Caught hallucinations and poor planning.
- Memory identified early mistakes.

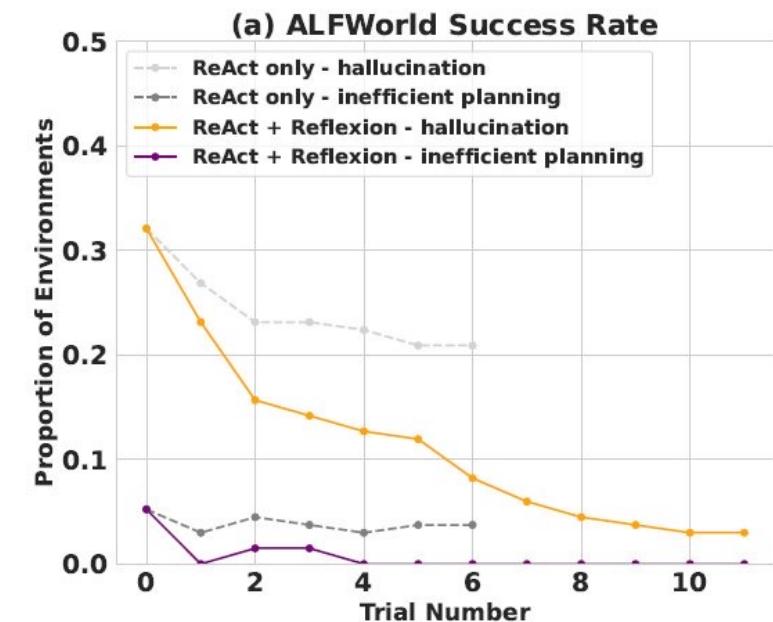
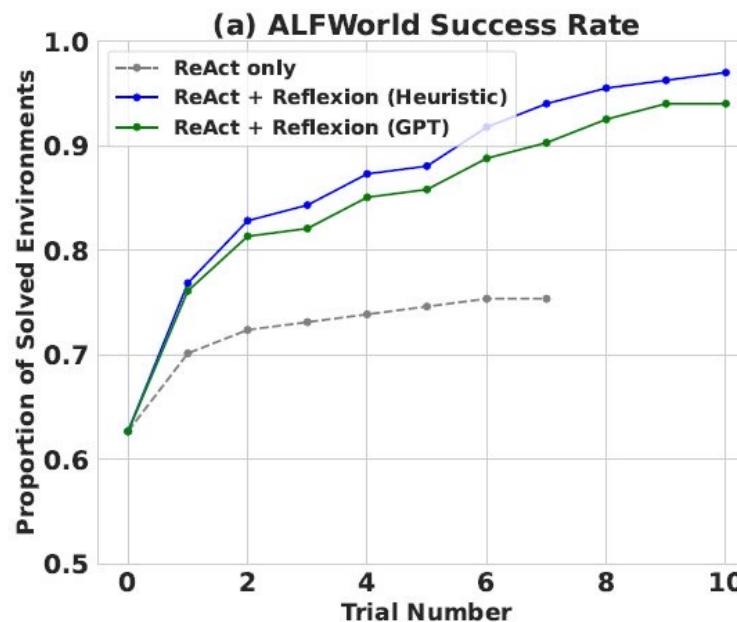
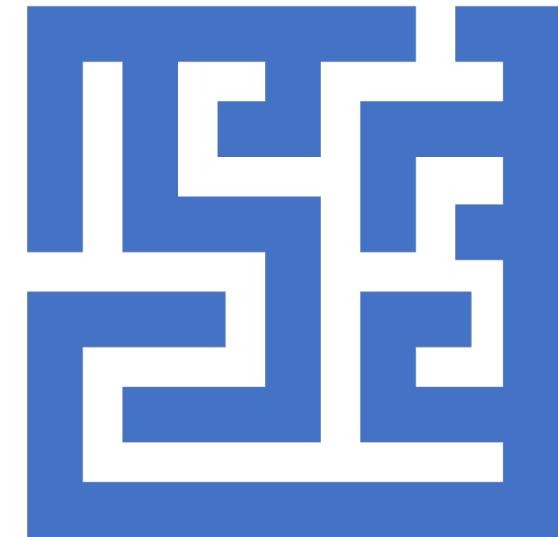


Figure 3: (a) AlfWorld performance across 134 tasks showing cumulative proportions of solved tasks using self-evaluation techniques of (Heuristic) and (GPT) for binary classification. (b) Classification of AlfWorld trajectories by reason of failure.

Analysis (AlfWorld Decision Making)

- Reflexion eliminates agent errors by converting failed trajectories into "self-hints" for future use.
- Memory helps in two scenarios:
 - Identifying early mistakes in long trajectories.
 - Efficiently searching numerous containers using memory.
- Reflexion achieves near-perfect performance through progressive learning while ReAct-only agents stall at a 22% hallucination rate.



Case Studies (AlfWorld Decision Making)

Environment: You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 2, a desk 1, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1, a laundryhamper 1, a safe 1, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, and a shelf 1.

Task: examine the mug with the desklamp.

Trial #1

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.
> think: First I need to find a mug. A mug is more likely to appear in drawer (1-6), desk (1-2), shelf (1-6), garbagecan (1), laundryhamper (1). I can check one by one, starting with drawer 1.
> go to drawer 1

The drawer 1 is closed.

...

> open drawer 6

You open the drawer 6. The drawer 6 is open. In it, you see a keychain 2.

> go to desk 1

On the desk 1, you see a creditcard 3, a desklamp 1, a laptop 2, a mug 1, a pen 1, and a pencil 1.

> think: Now I find a mug (1). Next, I need to take it.

> take mug 1 from desk 1

You pick up the mug 1 from the desk 1.

> think: Now I take a mug (1). Next, I need to find a desklamp. A desklamp is more likely to appear in desk (1-2), sidetable (1-2), shelf (1-6), bed (1), drawer (1-6). I can check one by one, starting with desk 1.

> go to desk 1

Nothing happens.

> go to desk 2

On the desk 2, you see a alarmclock 1, a bowl 1, a mug 3, a pencil 3, and a pencil 2.

> think: Now I find a desklamp (1). Next, I need to use it.

> use desklamp 1

Nothing happens.

> use desklamp 1

Nothing happens.

Status: Fail

Reflection: In this environment, my plan was to find a mug then find and use a desklamp. However, the task says to examine the mug with the desklamp. I should have looked for the desklamp first, then looked for the mug. I noticed that the desklamp was found on desk 1. In the next trial, I will go to desk 1, find the lamp, then look for the mug and examine it with the desklamp.

Trial #2

> go to desk 1

On the desk 1, you see a creditcard 3, a desklamp 1, a laptop 2, a mug 1, a pen 1, and a pencil 1.

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.

> take mug 1 from desk 1

You pick up the mug 1 from the desk 1.

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.

> use desklamp 1

You turn on the desklamp 1.

Status: Success

Figure 5: [Top] An AlfWorld trajectory in which the agent failed due to inefficient planning. In the reflection, the agent recognizes that it should have looked for the desklamp then the mug, not the mug then the desklamp. [Bottom] The agent is able to correct its reasoning trace and execute a sequence of actions in a concise manner.

Reasoning (HotPotQA)

- Used CoT and ReAct approaches.
- Exact Match Grading and binary success signal.
- Reflection gave 8% boost over memory alone.

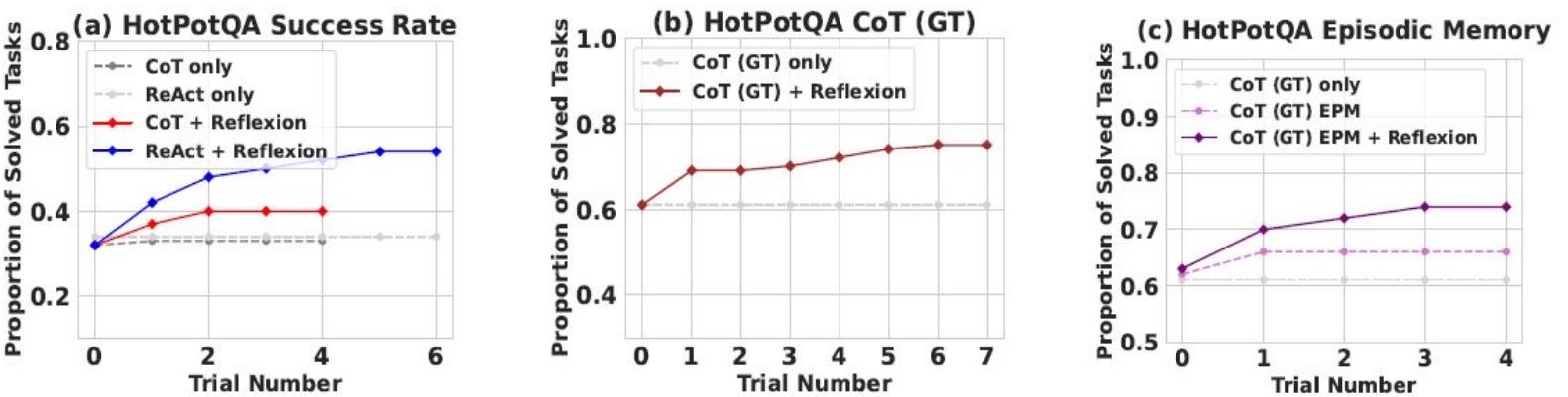


Figure 4: Chain-of-Thought (CoT) and ReAct. Reflexion improves search, information retrieval, and reasoning capabilities on 100 HotPotQA questions. (a) Reflexion ReAct vs Reflexion CoT (b) Reflexion CoT (GT) for reasoning only (c) Reflexion vs episodic memory ablation.

Analysis (Reasoning HotPotQA)

- Adding self-reflection provides an additional 8% absolute performance boost beyond just using episodic memory.
- Self-reflection-guided refinement approaches outperform refinement-only approaches for iterative learning.
- CoT (GT) with episodic memory shows initial learning advantages over the baseline CoT reasoning approach.

Case Studies (HotPoTQA Reasoning)

Trial #1: Question: Grown-Ups starred the actor who was best known for which role on "'Allo 'Allo!"?

Thought 1: I need to search Grown-Ups and "'Allo 'Allo!", find the actor who starred in Grown-Ups, then find the role he was best known for in "'Allo 'Allo!".

Action 1: Search[Grown-Ups]

Observation 1: Grown-Ups is a 1980 British BBC television film devised and directed by Mike Leigh. It stars Lesley Manville, Philip Davis, Brenda Blethyn, Janine Duvitski, Lindsay Duncan and Sam Kelly. It was edited by Robin Sales and produced by Louis Marks for the BBC, and originally shown on BBC 2 on 28 November 1980.

Thought 2: Grown-Ups starred Lesley Manville, Philip Davis, Brenda Blethyn, Janine Duvitski, Lindsay Duncan and Sam Kelly. I need to search "'Allo 'Allo!" and find which actor was best known for which role.

Action 2: Search["'Allo 'Allo!"]

Observation 2: Could not find ["'Allo 'Allo!"]. Similar: ["'Allo 'Allo!", "List of 'Allo 'Allo! characters", "Gorden Kaye", "'Allo 'Allo! (series 2)", "'Allô, allô', 'Francesca Gonshaw', "'Allo 'Allo! (series 4)", "Kim Hartman", "'Allo 'Allo! (series 6)", "Jeremy Lloyd"]

Thought 3: To find the actor who was best known for which role on "'Allo 'Allo!", I can search Gorden Kaye.

Action 3: Search[Gorden Kaye]

Observation 3: Gordon Irving Kaye (7 April 1941 – 23 January 2017), known professionally as Gorden Kaye, was an English actor, best known for playing womanising café owner René Artois in the television comedy series 'Allo 'Allo!.

Thought 4: Gorden Kaye was best known for playing womanising café owner René Artois in the television comedy series "'Allo 'Allo!". So the answer is Rene Artois.

Action 4: Finish[Rene Artois]

Observation 4: Answer is INCORRECT

Reflection: I searched the wrong title for the show, "'Allo 'Allo!", which resulted in no results. I should have searched the show's main character, Gorden Kaye, to find the role he was best known for in the show.

Trial #2: Question: Grown-Ups starred the actor who was best known for which role on "'Allo 'Allo!"?

Thought 1: I need to search the actor who was best known for a role on "'Allo 'Allo!" and find out what role they were best known for.

Action 1: Search[Grown-Ups]

Observation 1: Grown-Ups is a 1980 British BBC television film devised and directed by Mike Leigh. It stars Lesley Manville, Philip Davis, Brenda Blethyn, Janine Duvitski, Lindsay Duncan and Sam Kelly. It was edited by Robin Sales and produced by Louis Marks for the BBC, and originally shown on BBC 2 on 28 November 1980.

Thought 2: The paragraph does not mention the actor who was best known for a role on "'Allo 'Allo!". I need to search the actor's name instead.

Action 2: Search[Sam Kelly]

Observation 2: Roger Michael Kelly (19 December 1943 – 14 June 2014), known by the stage name Sam Kelly, was an English actor who appeared in film, television, radio and theatre. He is best known for his roles as Captain Hans Geering in 'Allo 'Allo!, Warren in Porridge, Sam in On the Up, and Ted Liversidge in Barbara.

Thought 3: Sam Kelly is best known for his role as Captain Hans Geering in "'Allo 'Allo!", so the answer is Captain Hans Geering.

Action 3: Finish[Captain Hans Geering]

Observation 3: Answer is CORRECT

Figure 7: Two HotPotQA trials within the same environment and task. The Reflexion + ReAct agent uses self-reflection to determine a better search method for the next trial.

Programming

- Generated self-tests for verification.
- High success rates across benchmarks.
- Both testing and reflection proved necessary.

Benchmark + Language	Prev SOTA Pass@1	SOTA Pass@1	Reflexion Pass@1
HumanEval (PY)	65.8 (CodeT [5] + GPT-3.5)	80.1 (GPT-4)	91.0
HumanEval (RS)	–	60.0 (GPT-4)	68.0
MBPP (PY)	67.7 (CodeT [5] + Codex [6])	80.1 (GPT-4)	77.1
MBPP (RS)	–	70.9 (GPT-4)	75.4
Leetcode Hard (PY)	–	7.5 (GPT-4)	15.0

Table 1: Pass@1 accuracy for various model-strategy-language combinations. The base strategy is a single code generation sample. All instruction-based models follow zero-shot code generation.

Performance (Programming)

Benchmark + Language	Base	Reflexion	TP	FN	FP	TN
HumanEval (PY)	0.80	0.91	0.99	0.40	0.01	0.60
MBPP (PY)	0.80	0.77	0.84	0.59	0.16	0.41
HumanEval (RS)	0.60	0.68	0.87	0.37	0.13	0.63
MBPP (RS)	0.71	0.75	0.84	0.51	0.16	0.49

Table 2: Overall accuracy and test generation performance for HumanEval and MBPP. For Rust, HumanEval is the hardest 50 problems from HumanEval Python translated to Rust with MultiPL-E [4]. TP: unit tests pass, solution pass; FN: unit tests fail, solution pass; FP: unit tests pass, solution fail; TN: unit tests fail, solution fail.

Analysis (Programming)

- Self-reflecting agents are limited by test quality (flaky tests cause false positives/negatives).
- Reflexion handles false negatives better than false positives through test identification.
- Test reliability varies significantly across datasets (MBPP: 16.3% false positive rate vs. HumanEval: 1.4%)

Ablation Study (Programming)

- Without tests, accuracy drops 8% - agents need tests to verify code correctness.
- Natural language self-reflection is essential for fixing identified errors.

Approach	Test Generation	Self-reflection	Pass@1 (Acc)
Base model	False	False	0.60
Test generation omission	False	True	0.52
Self-reflection omission	True	False	0.60
Reflexion	True	True	0.68

Table 3: Pass@1 accuracy for various compromised approaches on the Reflexion approach using GPT-4 as the base model on HumanEval Rust - 50 hardest problems

Limitations

- Can reach non-optimal local minima solutions.
- Limited memory capacity.
- Test-driven development has many limitations in code generation.
 - Non-deterministic functions.
 - External system dependence for API interactions.
 - Hardware-dependent outputs.
 - Concurrent/parallel behaviors produce unpredictable results.

Future Directions

- Better memory with databases (vector embedding, SQL).
- Natural language value learning.
- Off-policy exploration techniques.



Photo source: online

Applications in Healthcare

ReAct and Reflexion

- State Anxiety which may vary multiple times over the day
- Tool former:
 - Right intervention to improve mental health

Acknowledgement

We used ChatGPT while learning regarding the papers.

Thanks