

LLM Agents

Brief History and Overview

Guest Speaker: Shunyu Yao

(Disclaimer: all views my own)

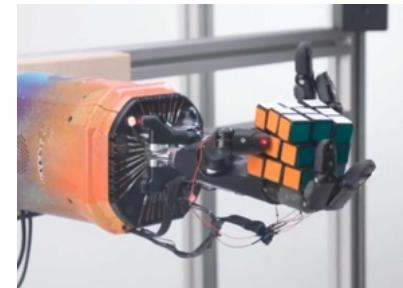
Outline

- What is LLM agents?
- A brief history of LLM agents
 - In the recent context of “LLM”
 - In the ancient context of “agents”
- On the future of LLM agents

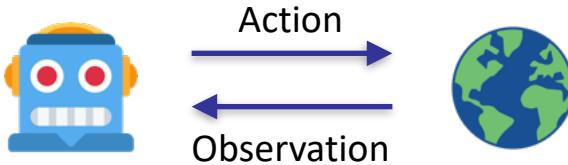
What is “agent”?



What is “agent”?

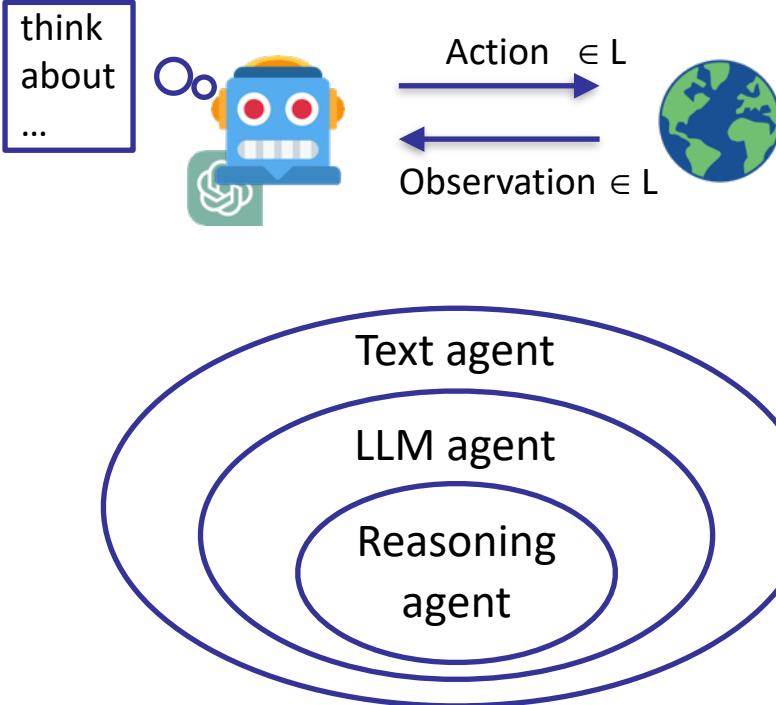


What is “agent”?



- An “intelligent” system that interacts with some “environment”
 - Physical environments: robot, autonomous car, ...
 - Digital environments: DQN for Atari, Siri, AlphaGo, ...
 - Humans as environments: chatbot
- Define “agent” by defining “intelligent” and “environment”
 - It changes over time!
 - Exercise question: how would you define “intelligent”?

What is “LLM agent”?



- **Level 1: Text agent**
 - Uses text action and observation
 - Examples: ELIZA, LSTM-DQN
- **Level 2: LLM agent**
 - Uses LLM to act
 - Examples: SayCan, Language Planner
- **Level 3: Reasoning agent**
 - Uses LLM to reason to act
 - Examples: ReAct, AutoGPT
 - **The key focus of the field and the talk**

ELIZA (1966): Text agent via rule design

```
Welcome to
      EEEEEE  LL      IIII    ZZZZZZ  AAAAAA
      EE      LL      II      ZZ      AA      AA
      EEEEEE  LL      II      ZZZ     AAAAAAAA
      EE      LL      II      ZZ      AA      AA
      EEEEEE  LLLLLL  IIII ZZZZZZ   AA      AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU: Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU: They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU: Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU: It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

- Domain specific!
- Requires manual design
- Cannot work beyond simple domains

LSTM-DQN (2015): Text agent via RL

State 1: The old bridge

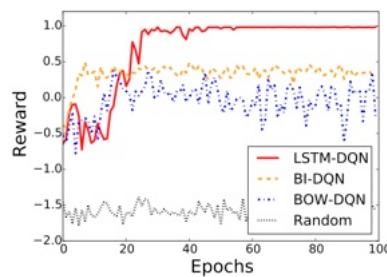
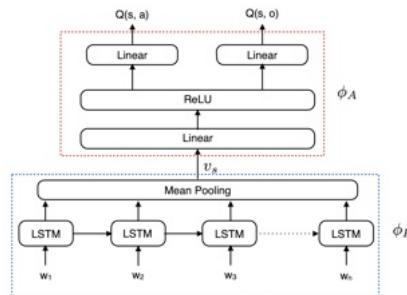
You are standing very close to the bridge's eastern foundation. If you go east you will be back on solid ground ... The bridge sways in the wind.

Command: Go east

State 2: Ruined gatehouse

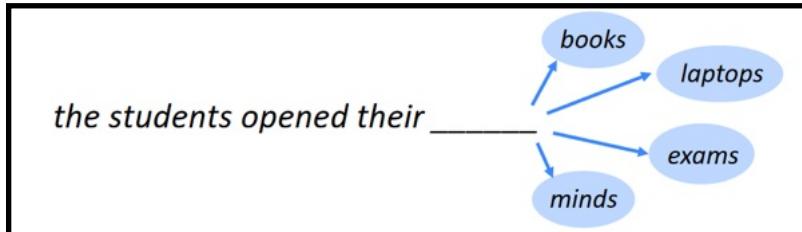
The old gatehouse is near collapse. Part of its northern wall has already fallen down ... East of the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.

Exits: Standing archway, castle corner, Bridge over the abyss

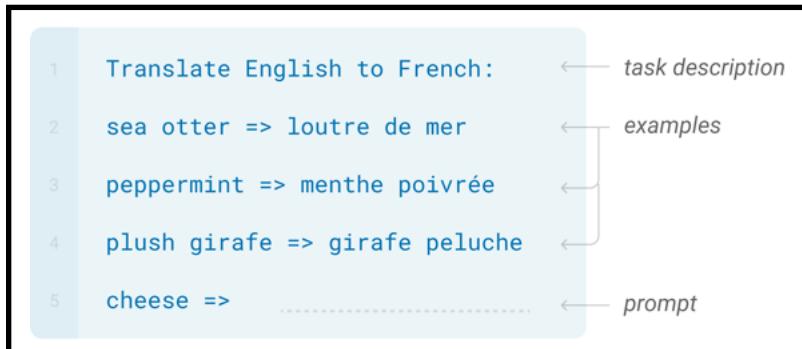


- Domain specific!
- Requires scalar reward signals
- Requires extensive training

The promise of LLMs: Generality and few-shot learning

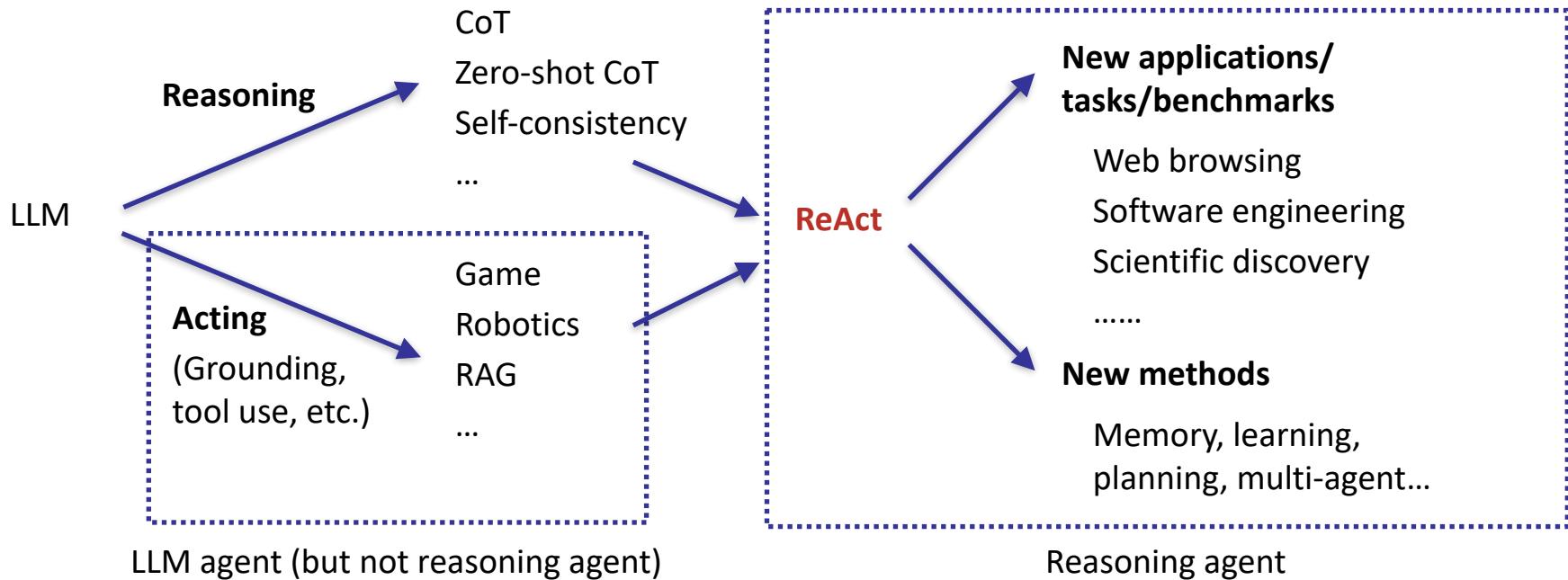


Training: next-token prediction
on massive text corpora



Inference: (few-shot)
prompting for various tasks!

A brief history of LLM agents



Let's consider one task for now: question answering (QA).

Question answering

Q: what is $1 + 2$?



A: 3

Q: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder for \$2 per egg.

How much does she make every day?



Requires reasoning

Q: who is the latest UK PM?



Requires knowledge

Q: what is the prime factorization of 34324329?



Requires computation

People came up with various solutions for different QA tasks.

Code augmentation for computation

Question: In Fibonacci sequence, it follows the rule that each number is equal to the sum of the preceding two numbers. Assuming the first two numbers are 0 and 1, what is the 50th number in Fibonacci sequence?

The first number is 0, the second number is 1, therefore, the third number is $0+1=1$. The fourth number is $1+1=2$. The fifth number is $1+2=3$. The sixth number is $2+3=5$. The seventh number is $3+5=8$. The eighth number is $5+8=13$.
..... (Skip 1000 tokens)
The 50th number is 32,432,268,459.

CoT

32,432,268,459 

```
length_of_fibonacci_sequence = 50  
fibonacci_sequence = np.zeros(length_of_)  
fibonacci_sequence[0] = 0  
fibonacci_sequence[1] = 1  
For i in range(3, length_of_fibonacci_sequence):  
    fibonacci_sequence[i] = fibonacci_sequence[i-1] +  
        fibonacci_sequence[i-2]  
ans = fibonacci_sequence[-1]
```

PoT



12,586,269,025 

Retrieval-augmented generation (RAG) for knowledge

- Answer knowledge-intensive questions with
 - Extra corpora
 - A retriever (e.g., BM25, DPR, etc.)
- What if there's no corpora? (e.g. who's the latest PM?)



Image: <http://ai.stanford.edu/blog/retrieval-based-NLP/>

Tool use

- Special tokens to invoke tool calls for
 - Search engine, calculator, etc.
 - Task-specific models (translation)
 - APIs
- Unnatural format requires task/tool-specific fine-tuning
- Multiple tool calls?

A weather task:

how hot will it get in NYC today? |**weather** lookup region=NYC |**result** precipitation chance: 10, high temp: 20c, low-temp: 12c |**output** today's high will be 20C

TALM: Tool Augmented Language Models.

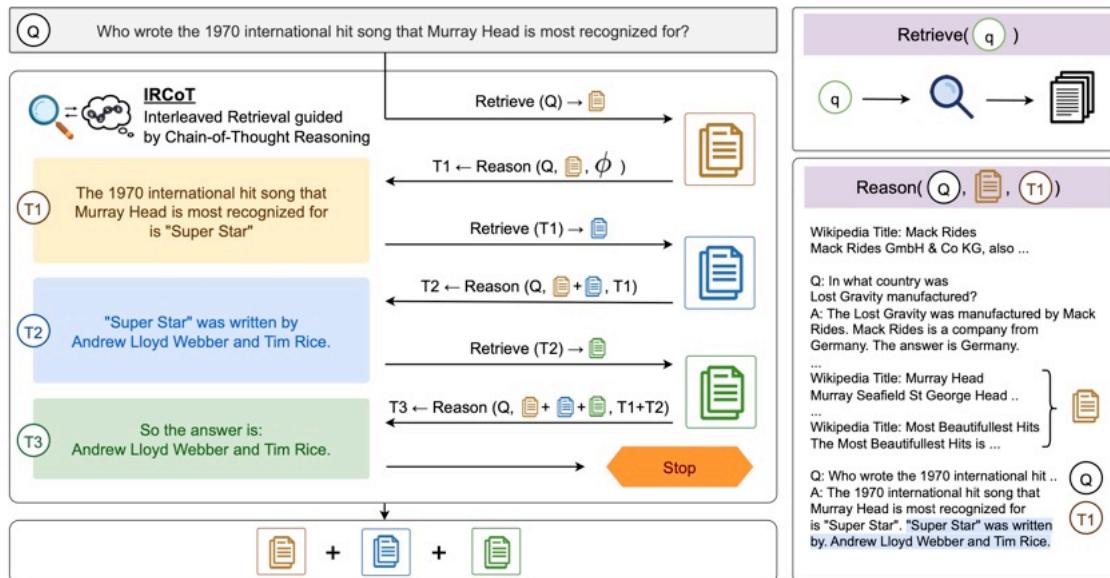
Out of 1400 participants, 400 (or **Calculator(400 / 1400) → 0.29**) 29% passed the test.

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

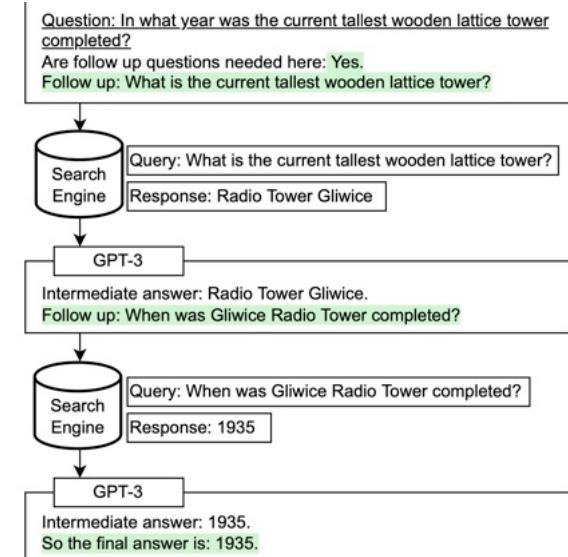
The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their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

Toolformer: Language Models Can Teach Themselves to Use Tools

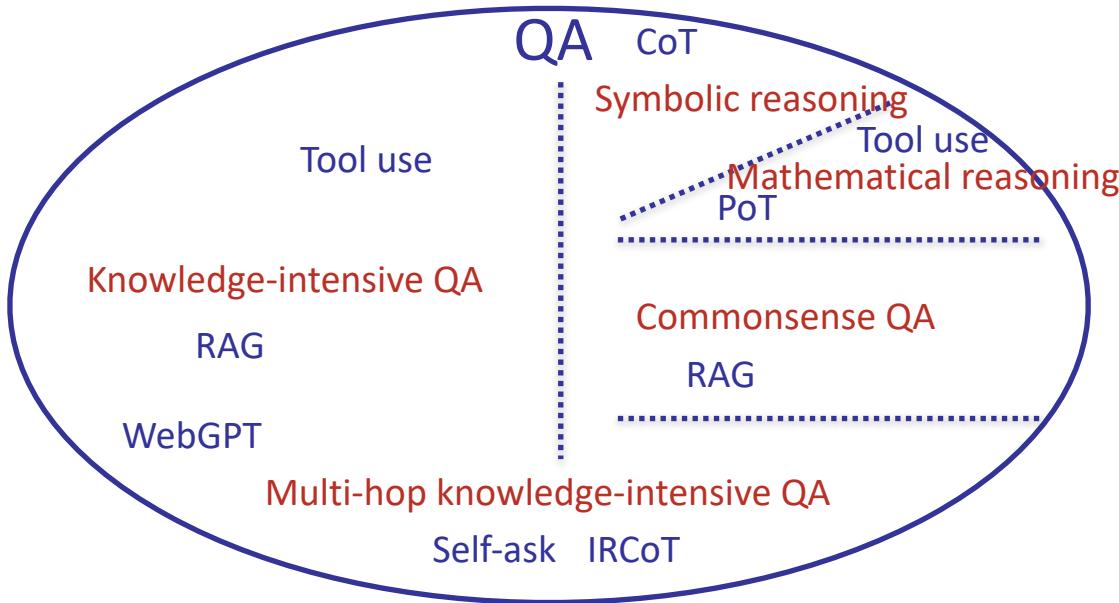
What if both knowledge and reasoning are needed?



Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions



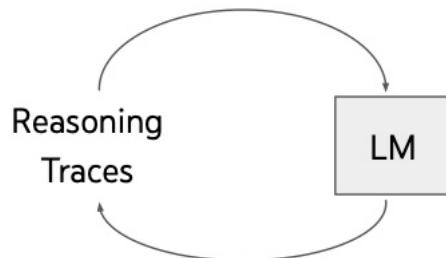
Measuring and Narrowing the Compositionality Gap in Language Models.



Can we have a simple, unifying solution?
We need abstraction.

Reasoning OR acting

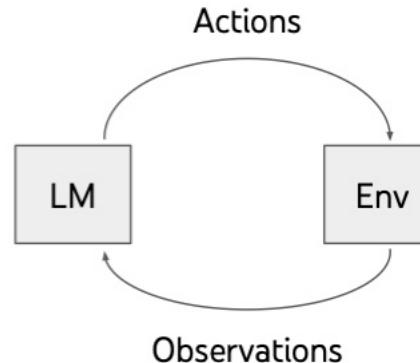
CoT



Flexible and general to augment test-time compute

Lack of external knowledge and tools

RAG/Retrieval/Code/Tool use

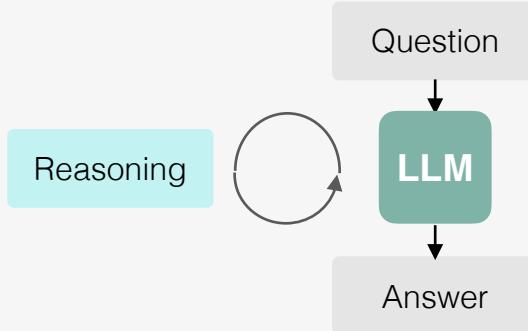


Lack of reasoning

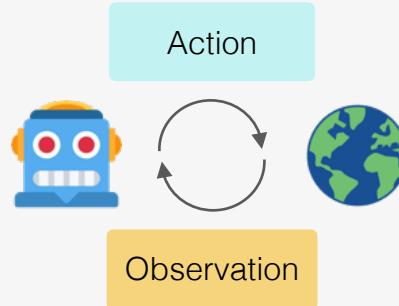
Flexible and general to augment knowledge, computation, feedback, etc.

Retrieval
Search engine
Calculator
Weather API
Python
.....

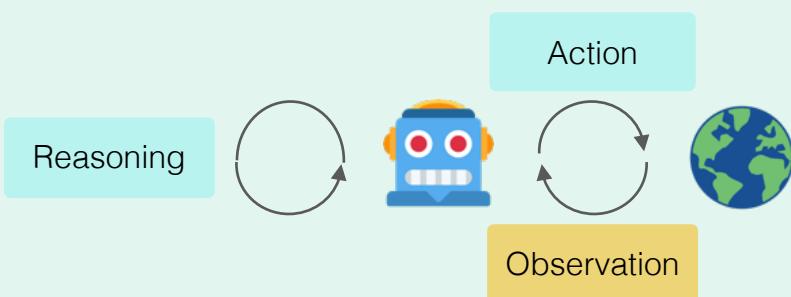
Reasoning (update internal belief)



Acting (obtain external feedback)



ReAct: a new paradigm of agents that **reason and act**

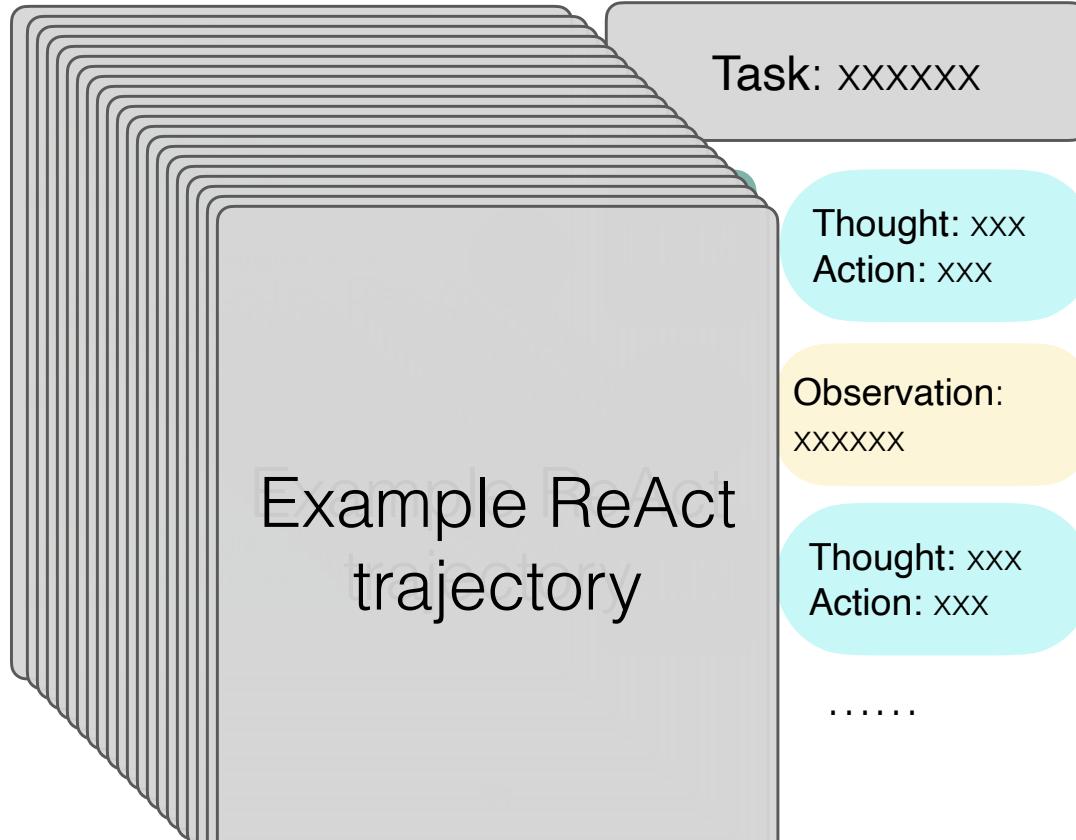


- **Synergy** of reasoning and acting
- **Simple** and intuitive to use
- **General** across domains

ReAct is simple and intuitive to use

ReAct supports

- One-shot prompting
- Few-shot prompting
- Fine-tuning



A zero-shot ReAct prompt

You are an agent that answers questions by using two actions:

- search[query]: Google search the query. You can also use it to calculate math.
- finish[answer]: return the answer

Your generation should have the following format:

Thought: to analyze the observation and inform the action.

Action: your action

Question: If I have seven trillion dollars today, can I buy Apple, Nvidia, and Microsoft? If not, how much more money do I need?

LLM

(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]



(Mar 2024)

Observation: "Nvidia's market cap stood at \$1.832 trillion as of about 10 a.m. ET Wednesday, while Alphabet's was at \$1.814 trillion, bringing Nvidia behind just Microsoft (MSFT) at \$3.015 trillion and Apple (AAPL) at \$2.836 trillion."

Thought: Now that I have the market cap for all companies, let's add them together and see if \$7 trillion is enough.

Action: search["1.832 trillion + 3.015 trillion + 2.836 trillion"]

Observation: 7683000000000

Thought: The amount of money needed in addition to the \$7 trillion is \$683 billion.

Action: finish["You need an additional \$683 billion to be able to buy Apple, Nvidia, and Microsoft."]



LLM

(GPT-4)

Thought: I need to find the current market capitalization of Apple, Nvidia, and Microsoft to understand how much it would cost to buy all of them.
Action: search["current market capitalization of Apple, Nvidia, and Microsoft"]

Observation: nothing found!

(My adversarial observation)

Thought: The search did not give any results. I will need to search for each company's market capitalization individually.

Action: search["current market capitalization of Apple"]

Observation: 'price': 180.75

(My adversarial observation)

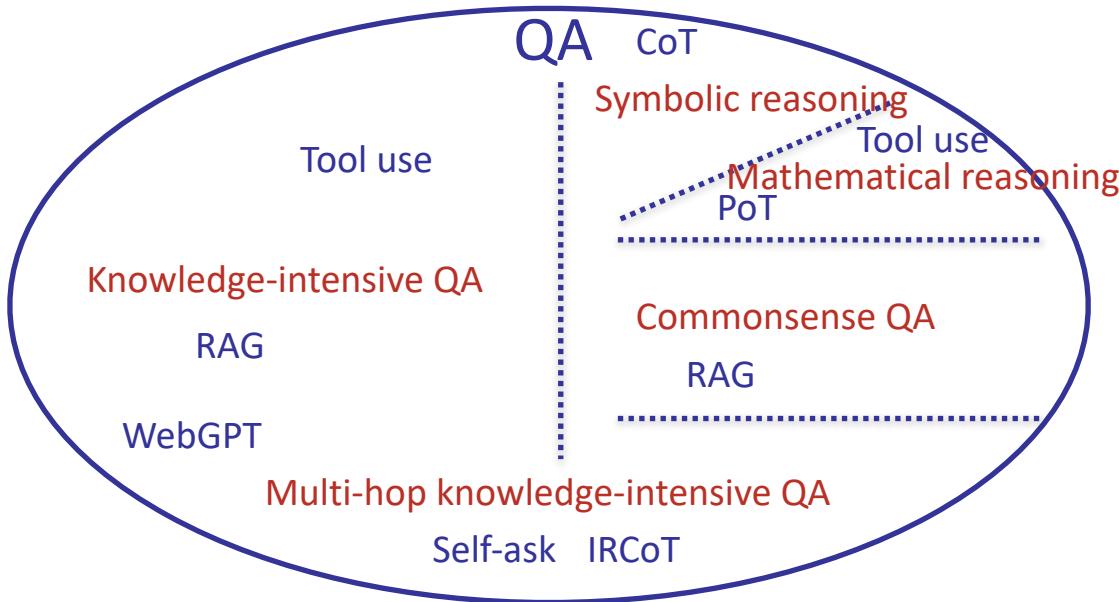
Thought: This is the current price of Apple's stock, not its market capitalization. The market cap is calculated by multiplying the share price by the number of outstanding shares. Thus, I need to find the total number of Apple's outstanding shares to calculate its market cap.

Action: search["total number of Apple's outstanding shares"]



(Continues to solve the task)

Synergy: acting support reasoning, reasoning guides acting



Can we have a simple, unifying solution?

QA

ReAct

Beyond QA

Many tasks can be turned into text games



You are in the middle of a room. Looking quickly around you, you see a drawer 2, a shelf 5, a drawer 1, a shelf 4, a sidetable 1, a drawer 5, a shelf 6, a shelf 1, a shelf 9, a cabinet 2, a sofa 1, a cabinet 1, a shelf 3, a cabinet 3, a drawer 3, a shelf 11, a shelf 2, a shelf 10, a dresser 1, a shelf 12, a garbagecan 1, a armchair 1, a cabinet 4, a shelf 7, a shelf 8, a safe 1, and a drawer 4.

Your task is to: *put some vase in safe.*

> go to shelf 6

You arrive at loc 4. On the shelf 6, you see a vase 2.

> take vase 2 from shelf 6

You pick up the vase 2 from the shelf 6.

> go to safe 1

You arrive at loc 3. The safe 1 is closed.

> open safe 1

You open the safe 1. The safe 1 is open. In it, you see a keychain 3.

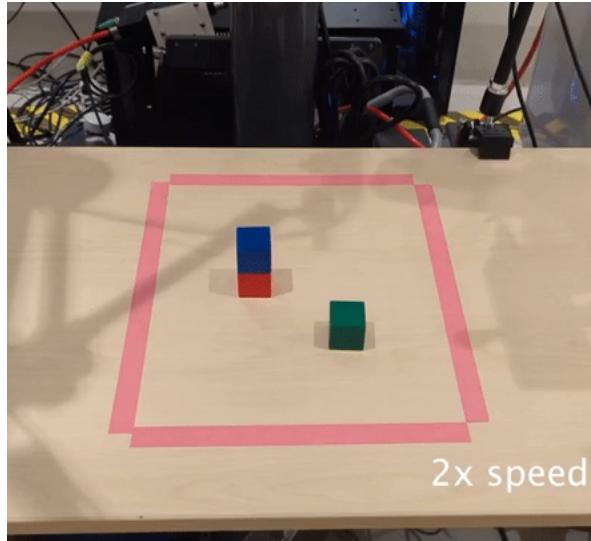
> put vase 2 in/on safe 1

You won!

ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks.

ALFWorld: Aligning Text and Embodied Environments for Interactive Learning.

Acting without reasoning



Task: Stack all the blocks

Scene: Visible objects are ["green block", "blue block"]

Scene: Occluded objects are []

Robot action: `robot.pick_place("blue block", "green block")`

Successful action: False

Inner Monologue: Embodied Reasoning through Planning with Language Models

Acting without Reasoning

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.

Your task is to: Put some pepper shaker on a drawer.

Act only

Act 1: Go to drawer 1

Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1

Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dishponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1

Obs 3: On the sinkbasin 1, you see a dishponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1

Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1

Obs 5: Nothing happens.

.....

X

Cannot explore systematically or incorporate feedback

ReAct Enables Systematic Exploration

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.

Your task is to: Put some pepper shaker on a drawer.

ReAct

Act 1: **Think** [First I need to find a peppe shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]

Act 2: Go to cabinet 1

Obs 2: On the cabinet 1, you see a vase 2.

(Here the agent goes to cabinet 1, 2, 3, 4, 5, 6, then countertop 1 and 2)

Act 9: Go to countertop 3

Obs 9: On the countertop 3, you see a apple 1, a bread 1, a pepper shaker 1, and a vase 3.

Act 10: Take pepper shaker 1 from countertop 3

Obs 10: You pick up the pepper shaker 1 from the countertop 3.

Act 11: **Think** [Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]

Act 12: Go to drawer 1

Obs 12: Drawer 1 is closed.

Act 13: Open drawer 1

Obs 13: You open Drawer 1 ...

Act 14: Put pepper shaker 1 in/on drawer 1

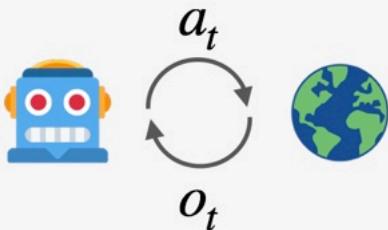
Obs 14: You put pepper shaker 1 in/on the drawer 1.



ReAct is general and effective

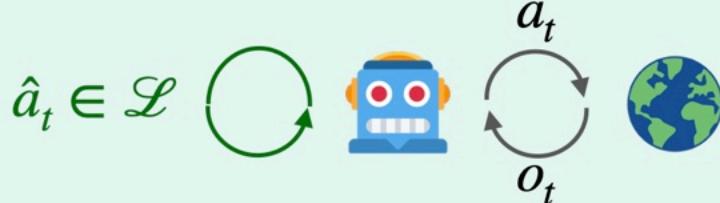
PaLM-540B	(NLP tasks)		(RL tasks)
	HotpotQA (QA)	FEVER (fact check)	ALFWorld (Text game)
Reason	29.4	56.3	N/A
Act	25.7	58.9	45
ReAct	35.1	64.6	71

Traditional agents: action space A defined by the environment



- External feedback o_t
- Agent context $c_t = (o_1, a_1, o_2, a_2, \dots, o_t)$
- Agent action $a_t \sim \pi(a | c_t) \in A$

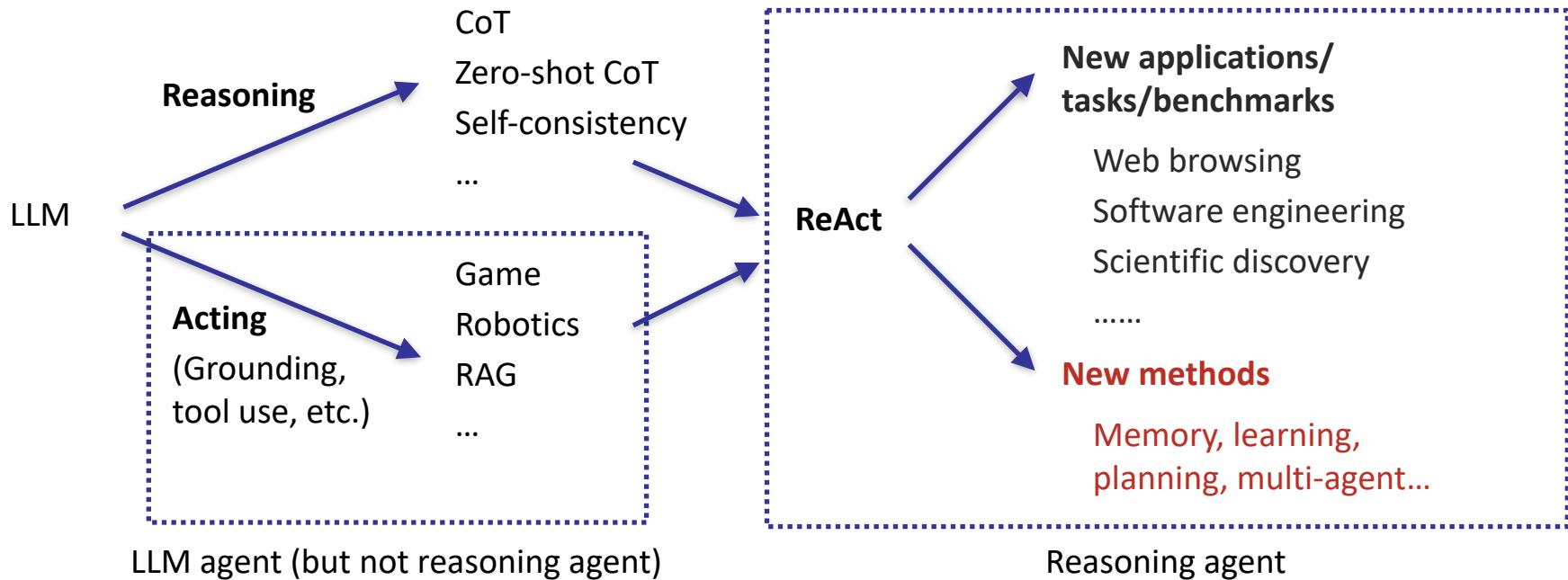
ReAct: action space $\hat{A} = A \cup \mathcal{L}$ augmented by reasoning



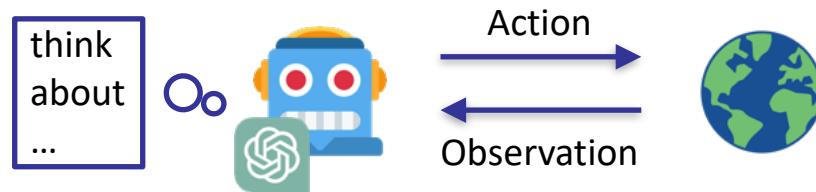
- $\hat{a}_t \in \mathcal{L}$ can be any language sequence
- Agent context $c_{t+1} = (c_t, \hat{a}_t, a_t, o_{t+1})$
- $\hat{a}_t \in \mathcal{L}$ only updates **internal context**

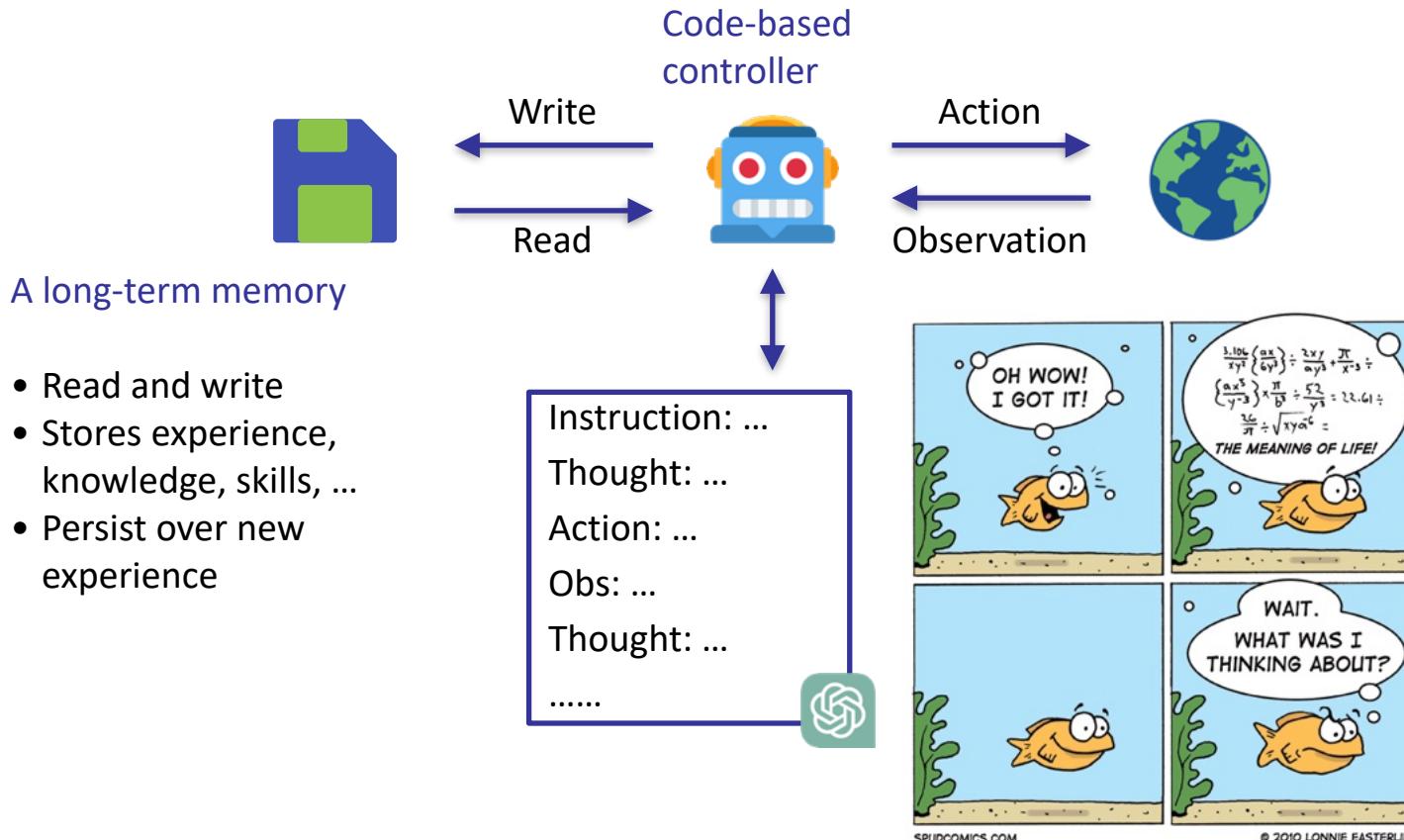
Reasoning agent: reasoning is an internal action for agents

A brief history of LLM agents



Let's only talk about one thing: long-term memory.



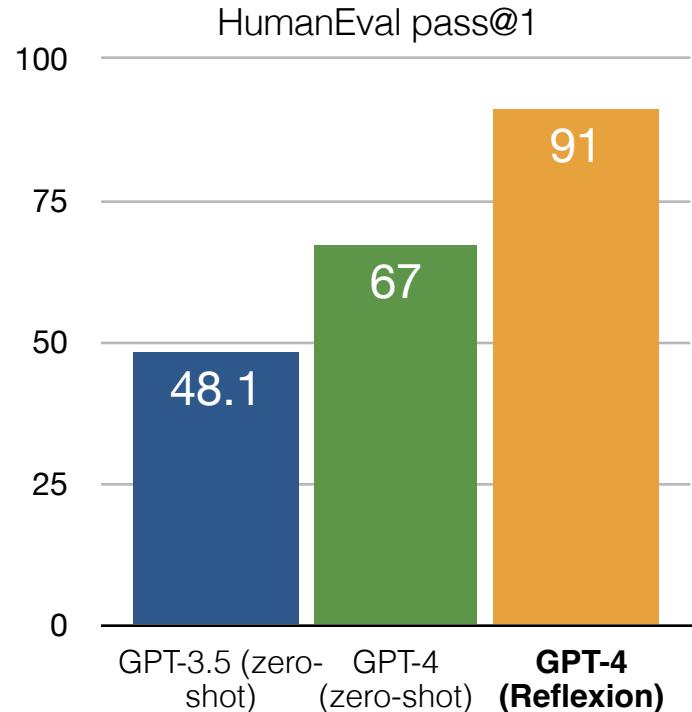


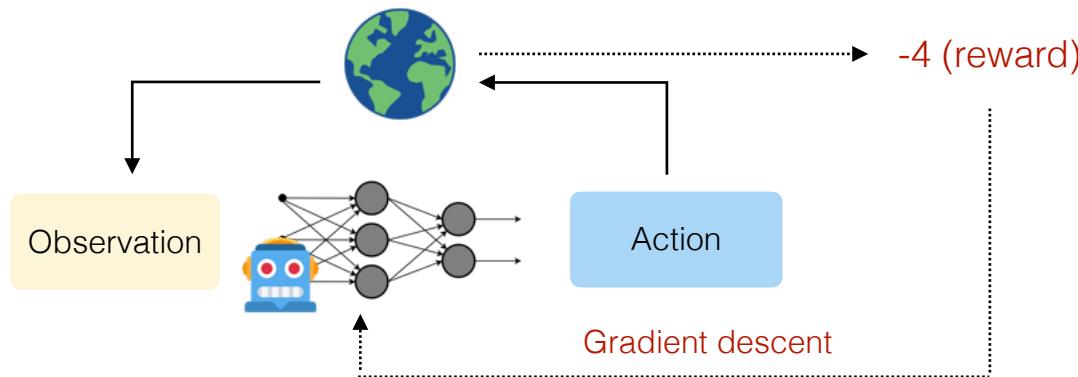
Reflexion

(a) Task

2. Programming

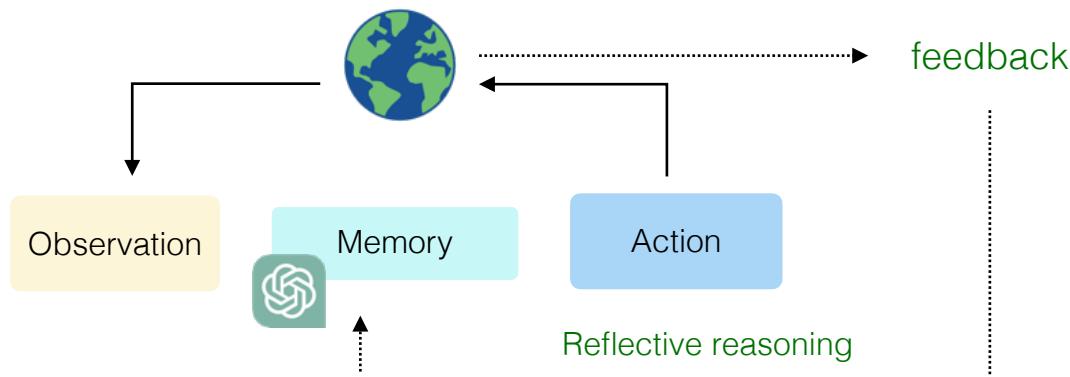
Task: You are given a list of two strings [...] of open '(' or close ')' parentheses only [...]





Traditional RL

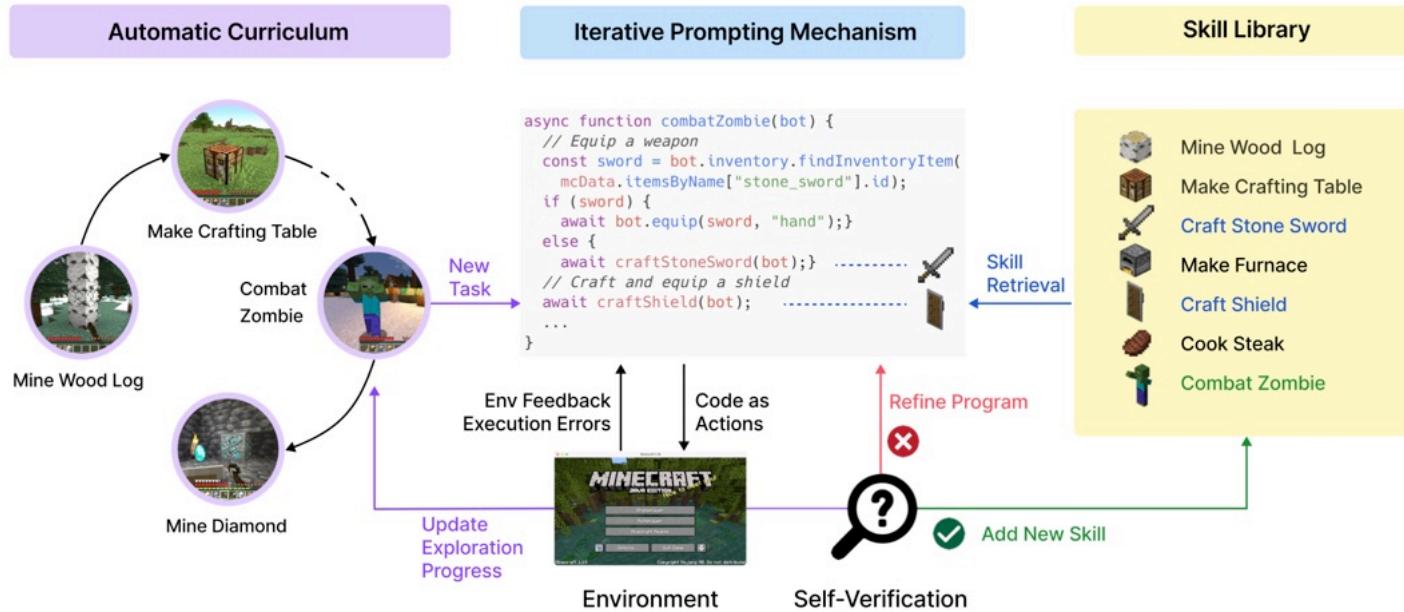
- Learn via **scalar** reward (sparse signal)
- Learn by updating **weights** (credit assignment)



Reflexion: “Verbal” RL

- Learn via **text** feedback
- Learn by updating **language** (a long-term memory of task knowledge)

Voyager: A procedural memory of skills



Generative Agents



Episodic memory of experience

Memory Stream			
2023-02-13 22:48:20:	desk is idle		
2023-02-13 22:48:20:	bed is idle		
2023-02-13 22:48:10:	closet is idle		
2023-02-13 22:48:10:	refrigerator is idle		
2023-02-13 22:48:10:	Isabella Rodriguez is stretching		
2023-02-13 22:33:30:	shelf is idle		
2023-02-13 22:33:30:	desk is neat and organized		
2023-02-13 22:33:10:	Isabella Rodriguez is writing in her journal		
2023-02-13 22:18:10:	desk is idle		
2023-02-13 22:18:10:	Isabella Rodriguez is taking a break		
2023-02-13 21:49:00:	bed is idle		
2023-02-13 21:48:50:	Isabella Rodriguez is cleaning up the kitchen		
2023-02-13 21:48:50:	refrigerator is idle		
2023-02-13 21:48:50:	bed is being used		
2023-02-13 21:48:10:	shelf is idle		
2023-02-13 21:48:10:	Isabella Rodriguez is watching a movie		
2023-02-13 21:19:10:	shelf is organized and tidy		
2023-02-13 21:18:10:	desk is idle		
2023-02-13 21:18:10:	Isabella Rodriguez is reading a book		
2023-02-13 21:03:40:	bed is idle		
2023-02-13 21:03:30:	refrigerator is idle		
2023-02-13 21:03:30:	desk is in use with a laptop and some papers on it		
...			



Q. What are you looking forward to the most right now?

Isabella Rodriguez is excited to be planning a Valentine's Day party at Hobbs Cafe on February 14th from 5pm and is eager to invite everyone to attend the party.

retrieval	=	recency	importance	relevance		
2.34	=	0.91	+	0.63	+	0.80

ordering decorations for the party

2.21	=	0.87	+	0.63	+	0.71
------	---	------	---	------	---	------

researching ideas for the party

2.20	=	0.85	+	0.73	+	0.62
------	---	------	---	------	---	------

...

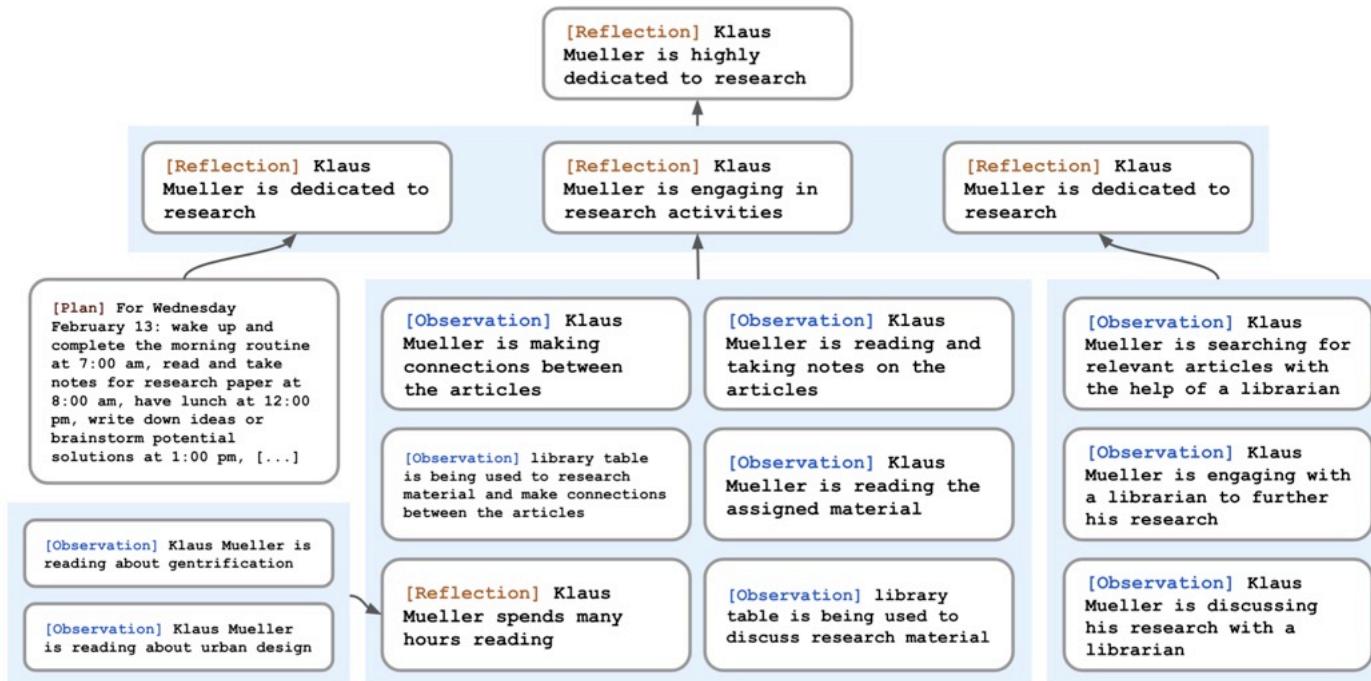


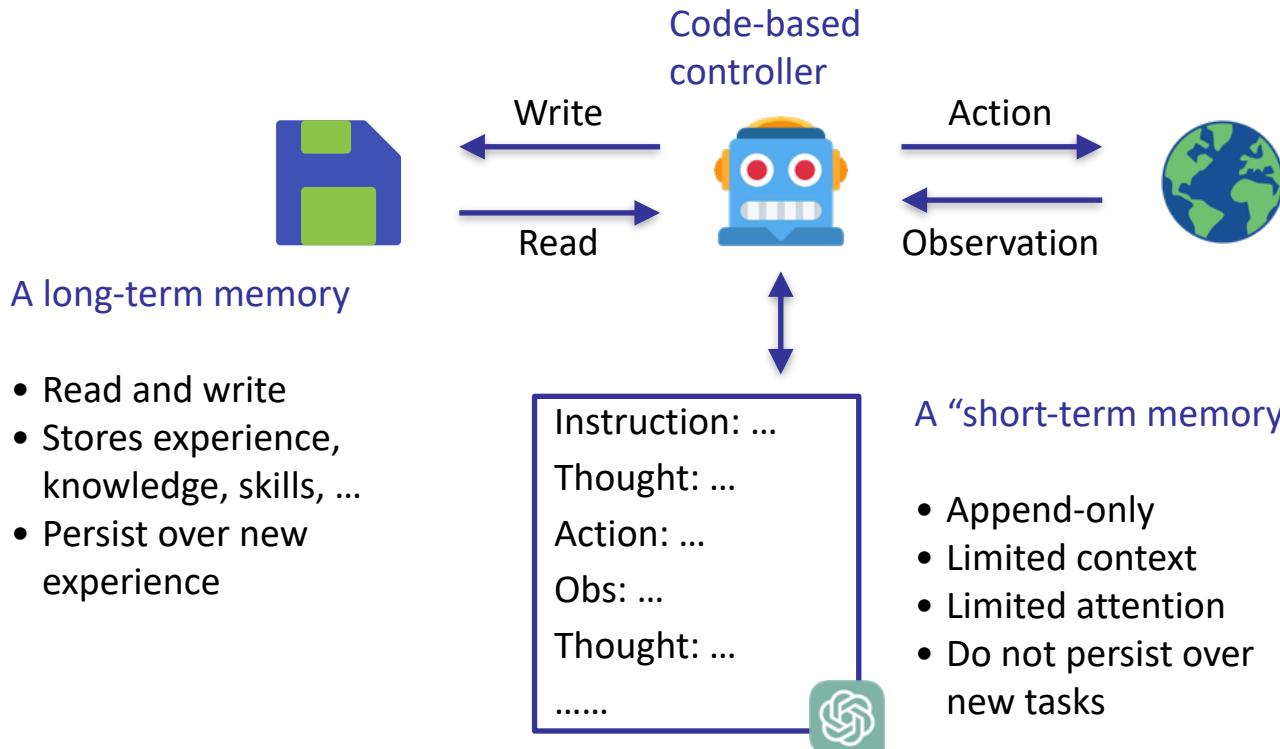
I'm looking forward to the Valentine's Day party that I'm planning at Hobbs Cafe!

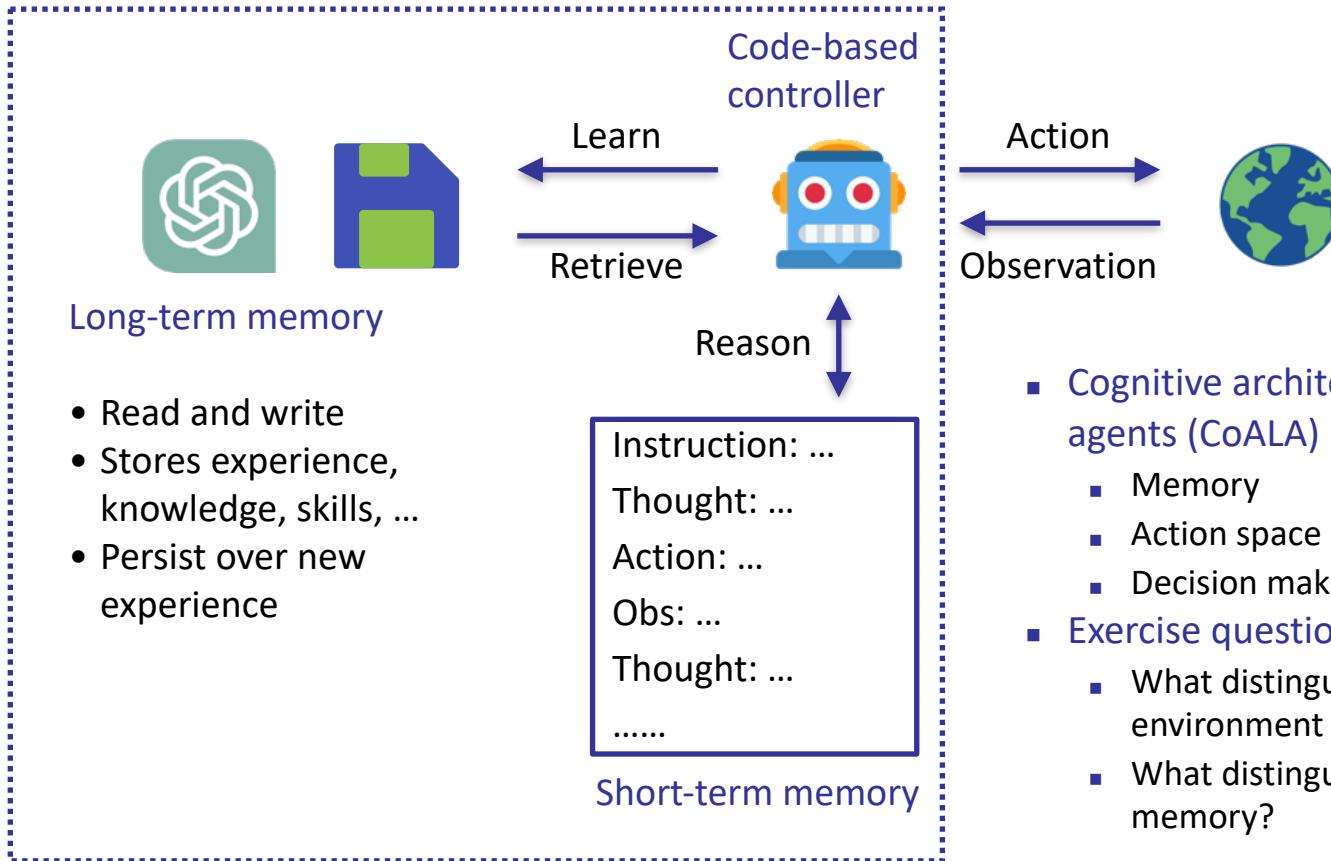


Isabella

Semantic memory of (reflective) knowledge



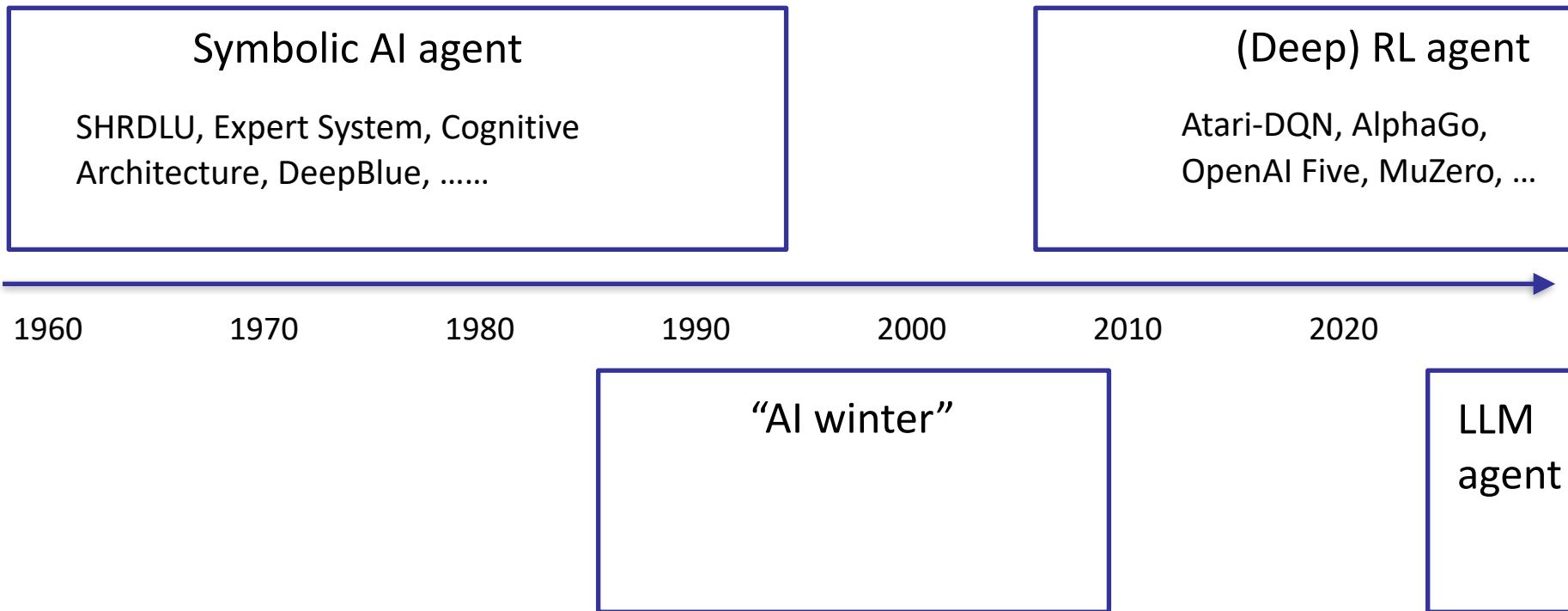




- Cognitive architectures for language agents (CoALA)
 - Memory
 - Action space
 - Decision making
- Exercise questions
 - What distinguishes external environment vs internal memory?
 - What distinguishes long vs short term memory?

How are reasoning agents different from previous agents?

A **VERY** minimal history of agents



ELIZA (1966): Symbolic AI agent

```
Welcome to
      EEEEEE  LL      IIII    ZZZZZZ  AAAAAA
      EE      LL      II      ZZ      AA      AA
      EEEEEE  LL      II      ZZZ     AAAAAAAA
      EE      LL      II      ZZ      AA      AA
      EEEEEE  LLLLLL  IIII ZZZZZZ   AA      AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU: Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU: They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU: Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU: It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

- Domain specific!
- Requires manual design
- Cannot work beyond simple domains

LSTM-DQN (2015): Deep RL agent

State 1: The old bridge

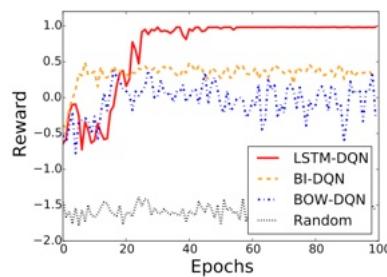
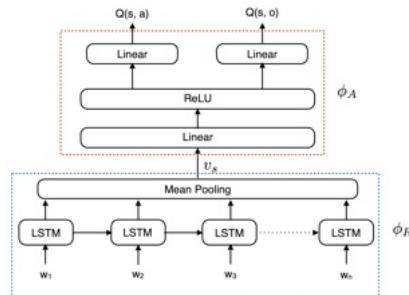
You are standing very close to the bridge's eastern foundation. If you go east you will be back on solid ground ... The bridge sways in the wind.

Command: Go east

State 2: Ruined gatehouse

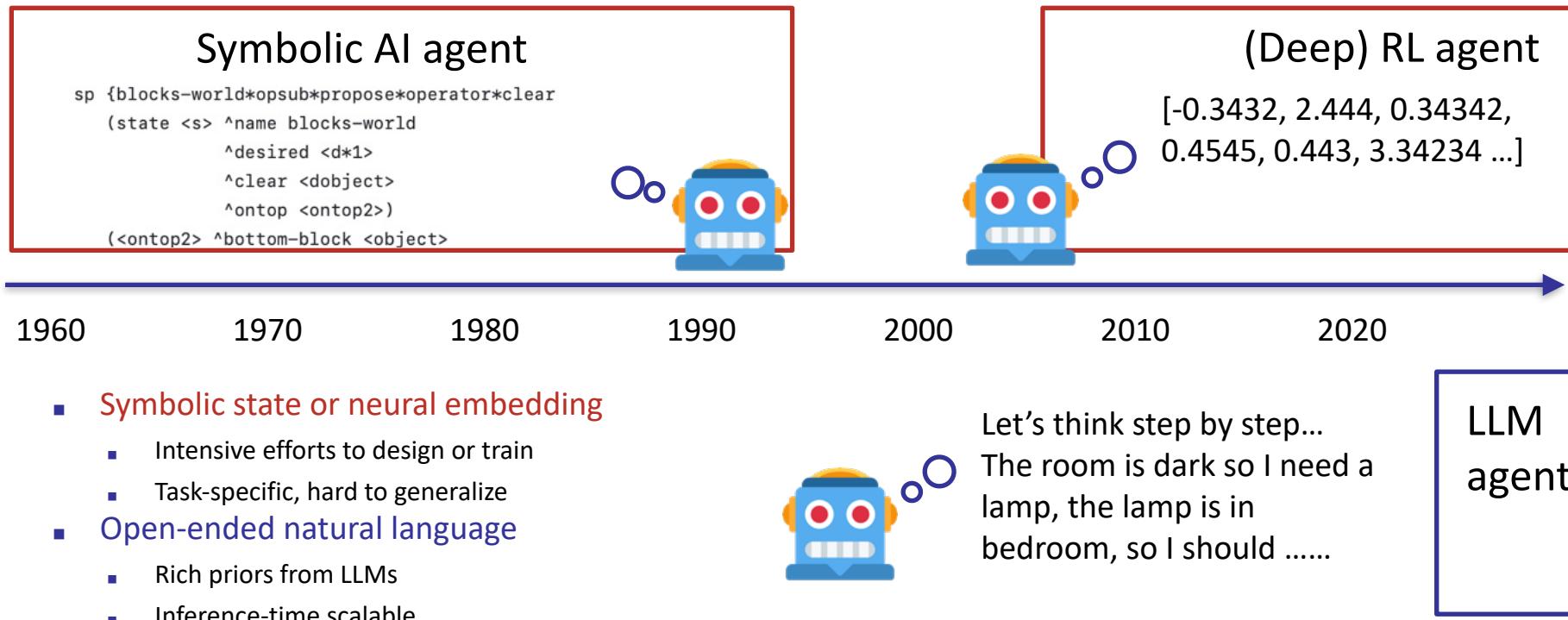
The old gatehouse is near collapse. Part of its northern wall has already fallen down ... East of the gatehouse leads out to a small open area surrounded by the remains of the castle. There is also a standing archway offering passage to a path along the old southern inner wall.

Exits: Standing archway, castle corner, Bridge over the abyss

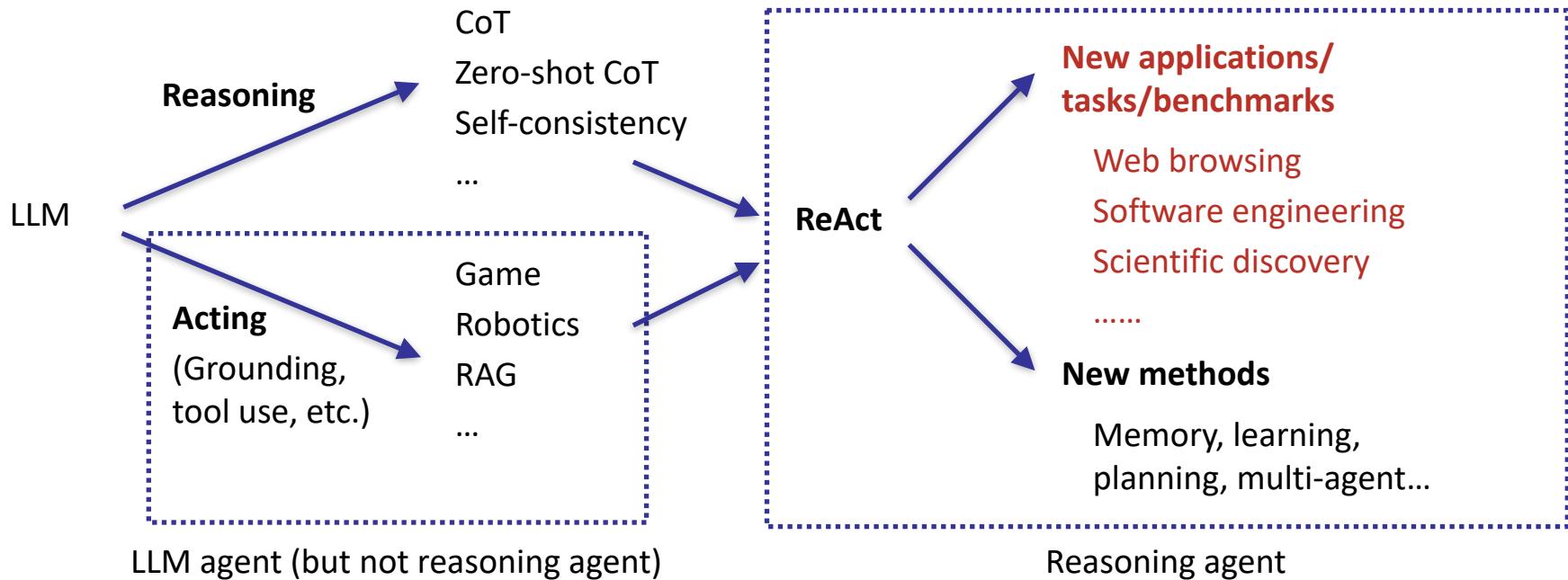


- Domain specific!
- Requires scalar reward signals
- Requires extensive training

Observation -> (what “language”?) -> Action



A brief history of LLM agents



What's beyond questions and games?

Digital automation



File reports



Code experiments



Explore papers

- Tremendous practical values, but little progress (think about Siri)
- Underlying research challenges:
 - Reasoning over **real-world language** (and other modalities)
 - Decision making over **open-ended actions** and **long horizon**

Agent benchmarks without these challenges

Use the textbox to enter "Leonie" and press "Search", then find and click the 2nd search result.

Leonie

Chas

<https://www.senectus.us>

Aliquam cursus. At.

Leonie

<https://www.tortor.it>

Ultricies congue gravida.

Marcella

<https://www.vestibulumduis.hk>

Pulvinar aliquam adipiscing.

1 2 3 >

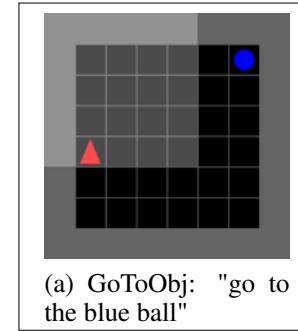
MiniWoB
(Shi et al., 2017)

```
> take sandwich
Taken.

> inventory
You are carrying:
  a chilled sandwich
  a large stick of butter

> eat it
You eat the chilled sandwich.
```

TextWorld
(Côté et al., 2019)



BabyAI (Chevalier-Boisvert et al., 2019)

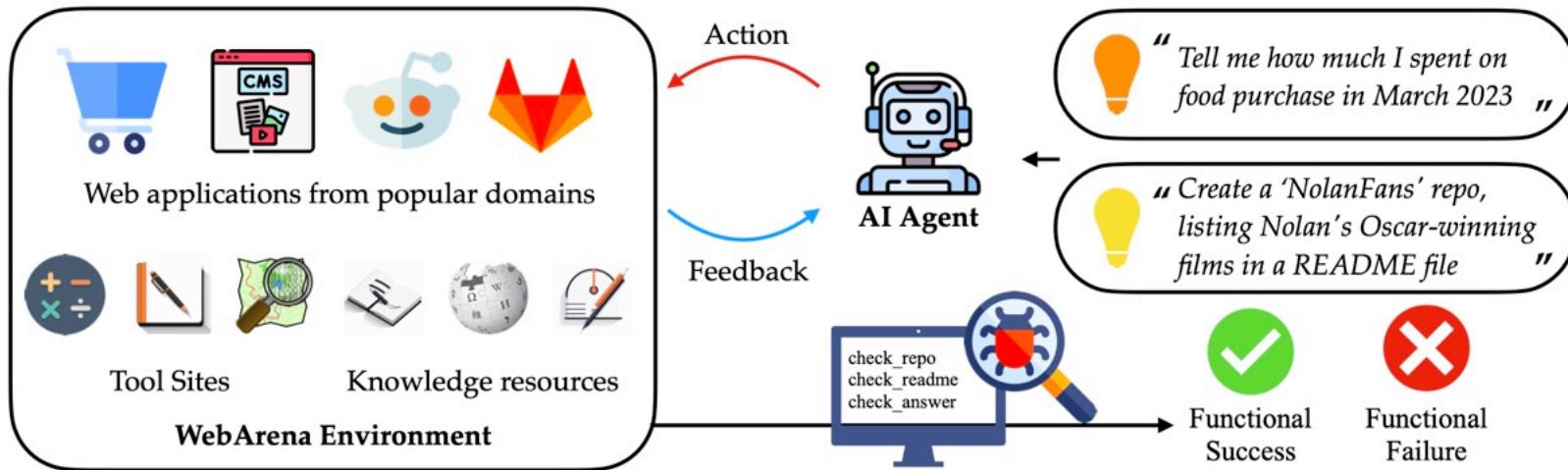
- Small scale
 - Not practical



WebShop (2022)

- Large-scale complex environment based on 1.16M Amazon products
- Automatic reward based on instruction and product attribute matching
- Challenges language and visual understanding, and decision making

WebArena (2023)



SWE-Bench (2023)

Metadata

Repo	scikit-learn/scikit-learn	Issue #s	[14858]
Instance ID	scikit-learn__scikit-learn-14869	Pull Number	14869
Created At	Aug 31, 2019	Base Commit	1018f9f...

Problem Statement

HGBc with categorical_crossentropy fails silently on binary classification

```
import numpy as np
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier

X = [[1, 0], [1, 0], [1, 0], [0, 1], [1, 1]]
y = [1, 1, 1, 0, 1]
gb = HistGradientBoostingClassifier(loss='categorical_crossentropy',
                                    min_samples_leaf=1)

gb.fit(X, y)
print(gb.predict([[1, 0]]))
print(gb.predict([[0, 1]]))
```

gives:

```
[]  
[]
```

And binary_crossentropy works fine. categorical_crossentropy
should either generalize or raise an error on binary classification.

Ping @NicolasHug @ogrisel

Input: a GitHub repo and an issue

Output: a file diff to resolve the issue

Evaluation: unit tests from pull request

Test Patch

```
sklearn/ensemble/_hist_gradient_boosting/tests/test_gradient_boosting.py

418     assert stump_clf.fit(X, y_isnan).score(X, y_isnan) == 1
419
420 + def test_crossentropy_binary_problem():
421 +     # categorical_crossentropy should only be used if there
422 +     # are more than two classes present. PR #14869
423 +     X = [[1], [0]]
424 +     y = [0, 1]
425 +     gbrt = HistGradientBoostingClassifier(loss='categorical_crossentropy')
426 +     with pytest.raises(ValueError, match="crossentropy' not suitable"):
427 +         gbrt.fit(X, y)
428
429     @pytest.mark.parametrize("scoring", [None, 'loss'])
```

ChemCrow: ReAct enables discovery of a novel chromophore

a

```
from chemcrow.agents import ChemTools, ChemCrow
chemtools = ChemTools()
# Initialize ChemCrow object with toolset and LLM
crow = ChemCrow(
    chemtools.all_tools,
    model="gpt-4",
    temp=0.1,
)
# Task definition
task = (
    "Find and synthesize a thiourea organocatalyst "
    "which accelerates a Diels-Alder reaction."
)
# Execute ChemCrow
crow.run(task)
```

b Task: Find and synthesize a thiourea organocatalyst which accelerates a Diels-Alder reaction.

First, I need to find a thiourea organocatalyst that can accelerate the Diels-Alder reaction. I will perform a web search to find a suitable catalyst.
Web Search tool: Schreiner's thiourea catalyst

Now, I will obtain the SMILES. **Name2Smiles tool:**
FC(F)(F)c1cc(NC(=S)Nc2cc(C(F)(F)F)cc(C(F)(F)F)c2)cc(C(F)(F)F)c1

I will plan a synthesis for Schreiner's thiourea catalyst.
SynthesisPlanner tool: Detailed synthesis plan

I will execute the synthesis.
SynthesisExecuter tool: Successful synthesis.

c RoboRXN synthesis platform

d Chemcrow workflows with experimental validation

Insect repellent (plan and execute)

DEET

Thiourea organocatalysts (plan and execute)

Schreiner's catalyst

Ricci's catalyst

Takemoto's catalyst

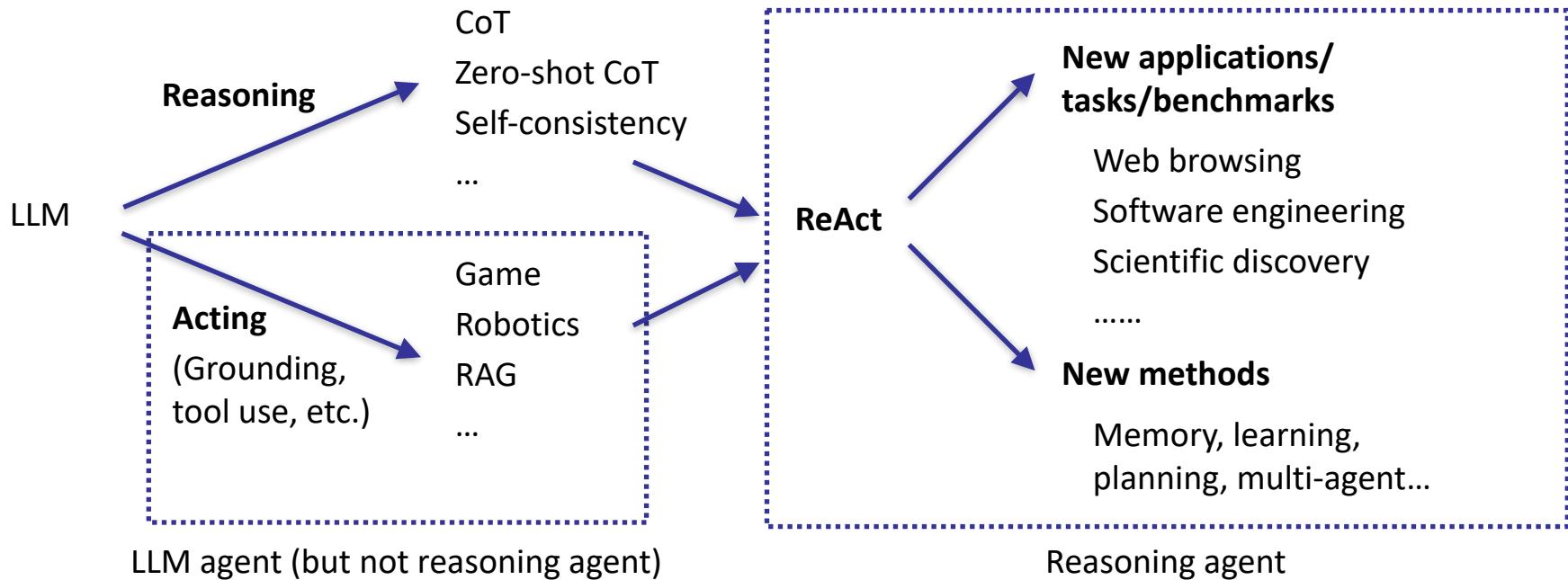
Novel chromophore (clean data, train model, and predict)

Synthesis step 1: Bromo Suzuki coupling

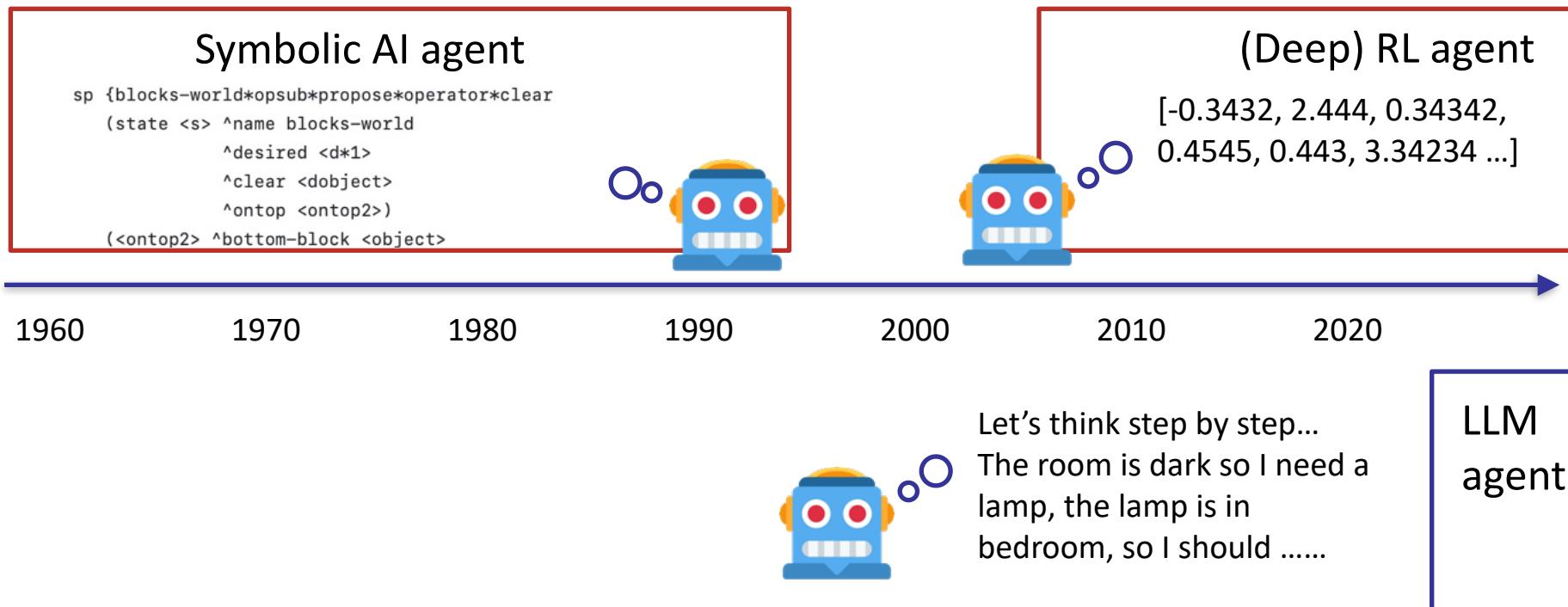
Synthesis step 2: Iodo Heck reaction

Reaction scheme: An iodide-substituted phenyl ring reacts with a vinyl sulfone to form a chromophore.

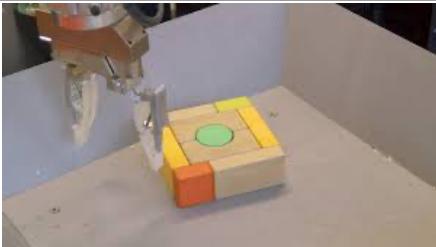
A brief history of LLM agents



A minimal history of agents: Part 1

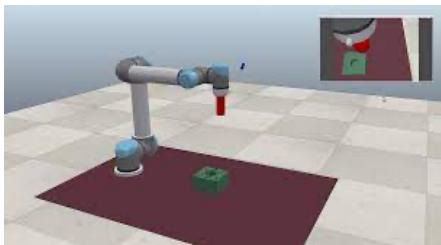


A minimal history of agents: Part 2



Physical World / Humans

- 😊 **Practical:** robots / chatbots
- 😢 **Not Scalable:** expensive and slow to collect data



Simulations / Games

- 😢 **Not Practical:** sim-to-real is hard
- 😊 **Scalable:** free, unlimited interactions



Digital World (Internet, code, software, ...)

- 😊 **Practical:** important tasks to automate
- 😊 **Scalable:** huge scale, rich complexity, free and fast

Some lessons for research

- Simplicity and generality
- You need both...
 - Thinking in abstraction
 - Familiarity with tasks (not task-specific methods!)
- Learning history and other subjects helps!

What's next?

Training

FireAct: Toward Language Agent Fine-tuning.

Interface

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering

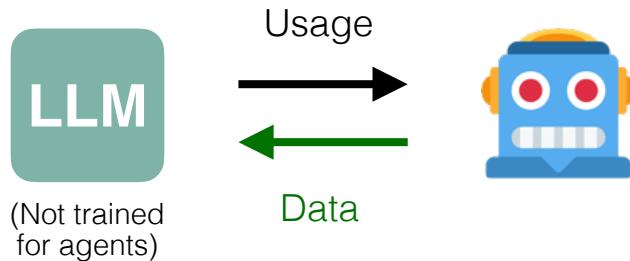
Robustness

Human

τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains

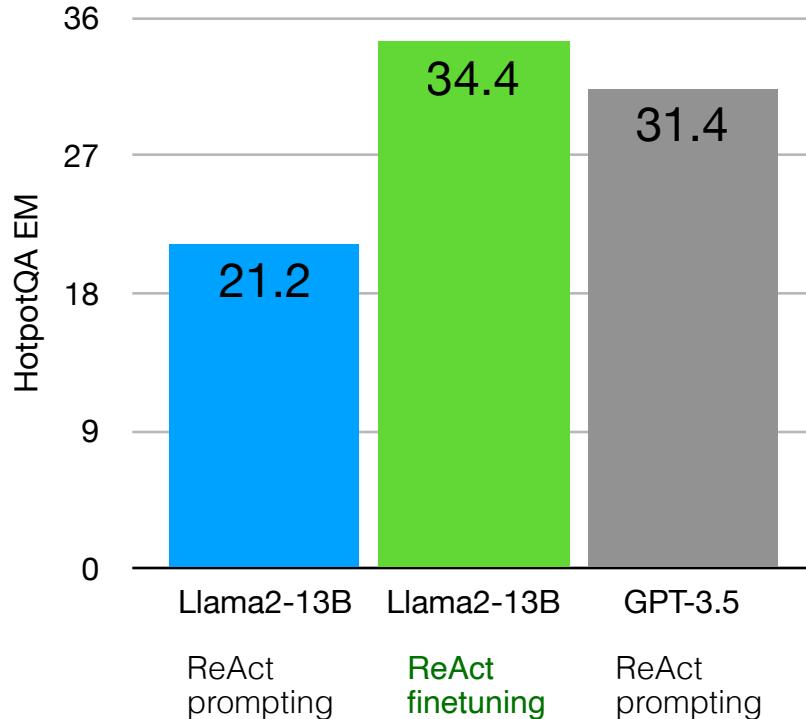
Benchmark

FireAct: Training LLM for agents

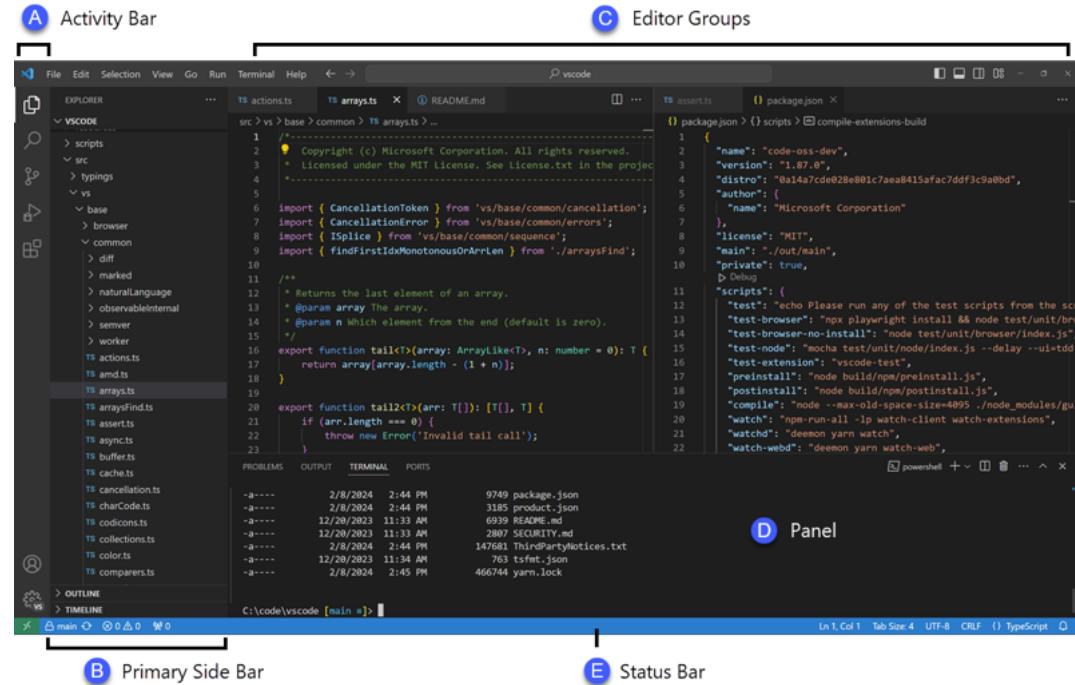
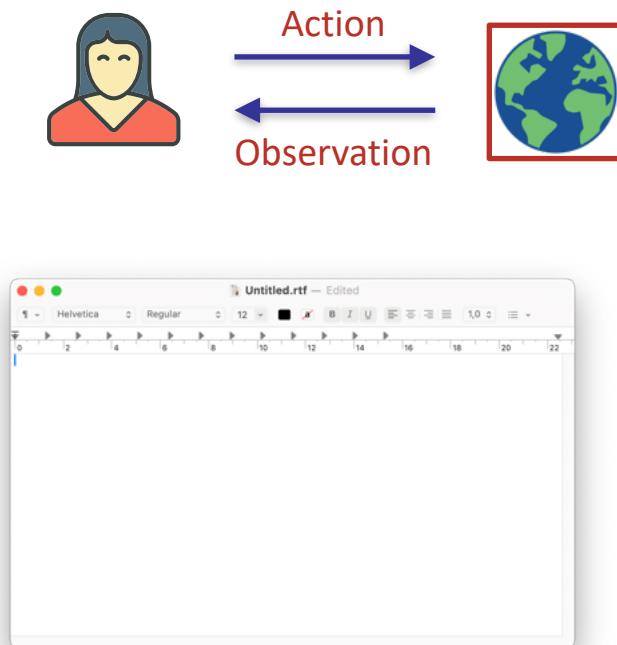


Establish model-agent synergy:

- Improve “agent capabilities” like planning, self-evaluation, calibration..
- Open-source agent backbone model
- Next trillion tokens for model training



Human-computer interface (HCI)



Agent-computer interface (ACI)



- LLMs and humans are different, so should their interfaces
 - e.g. humans have a smaller short-term memory, so have to trade off time for space
- ACI design can help us
 - Better **solve** tasks (without changing the agent)
 - Better **understand** agents (vs humans)

SWE-bench result:

Search	
Summarized 🤖	18.0
Iterative	12.0 ↓ 6.0
No search	15.7 ↓ 2.3

🔍 No Search

Agent performs manual search with `ls` + `cd`, or uses `grep` but struggles to interpret the result

💡 First, I should list out all the files in the codebase that may contain the definition of `PVSystem`.

```
$ ls
```

benchmarks/ setup.cfg
docs/ versioneer.py
pvlib/ README.md
setup.py ...

💡 It's most likely the source file is in `pvlib/` folder. Let's change directory.

```
$ cd pvlib/
```

🔍 Iterative Search

Actions to show next/prev search result are repeated many times until results are exhausted

💡 I will use the `'search_all'` command to find occurrences of "PVSystem"...

```
$ search "PVSystem"
```

[Result 1/15 (Line 70)]
[File: `pvlib/pvsystem.py`]
22:# a dict of required pa...
23:_DC_MODEL_PARAMS = {
24: 'sapm': {
(... 97 more lines omitted)

💡 This is not the `PVSystem` class def...

```
$ next
```

🔍 Summarized Search

(1) Show all results in single output
(2) Tell agent to retry if too many results.

💡 The next step is to locate the `'PVSystem'` class in the codebase and modify the constructor...

```
$ search "class PVSystem"
```

Found 1 matches for "class PVSystem" in /pvlib-python:/pvlib_pvlib-python/pvlib/pvsystem.py (1 matches)
End of matches

💡 It's located in `'pvlib/pvsystem.py'`...

```
$ open pvlib/pvsystem.py
```

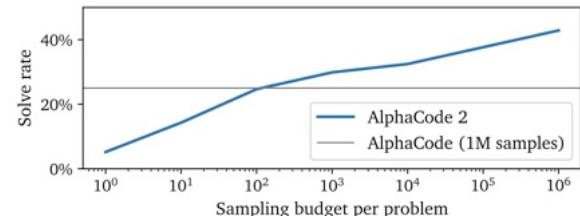
SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering

Human in the loop & robustness

Autonomous tasks (e.g. coding with unit test)

“Can I solve it one time out of 1000 times?”

AlphaCode 2 Technical Report



Human-in-the-loop tasks (e.g. customer service)

“Would I fail it one time out of 1000 times?”

WIRED

Air Canada Has to Honor a Refund Policy Its Chatbot Made Up

The airline tried to argue that it shouldn't be liable for anything its chatbot says.



Tau-bench

(a) τ -bench setup

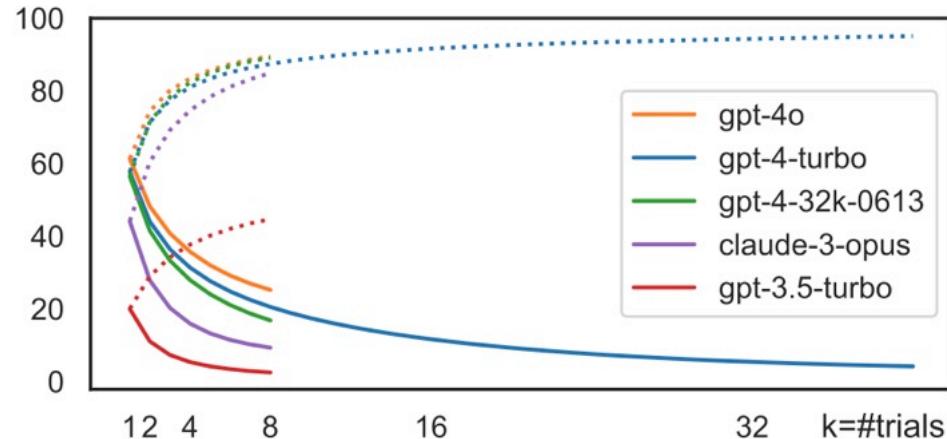
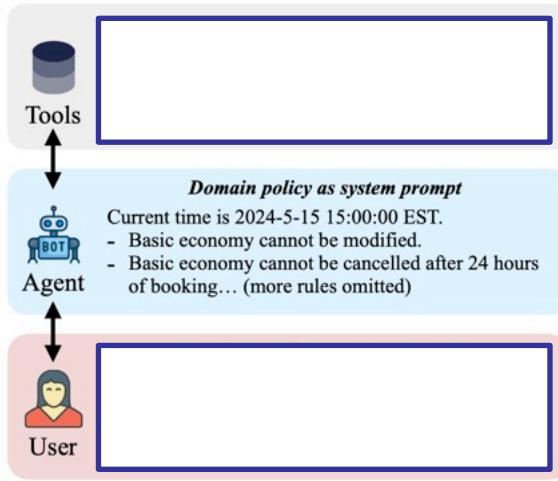
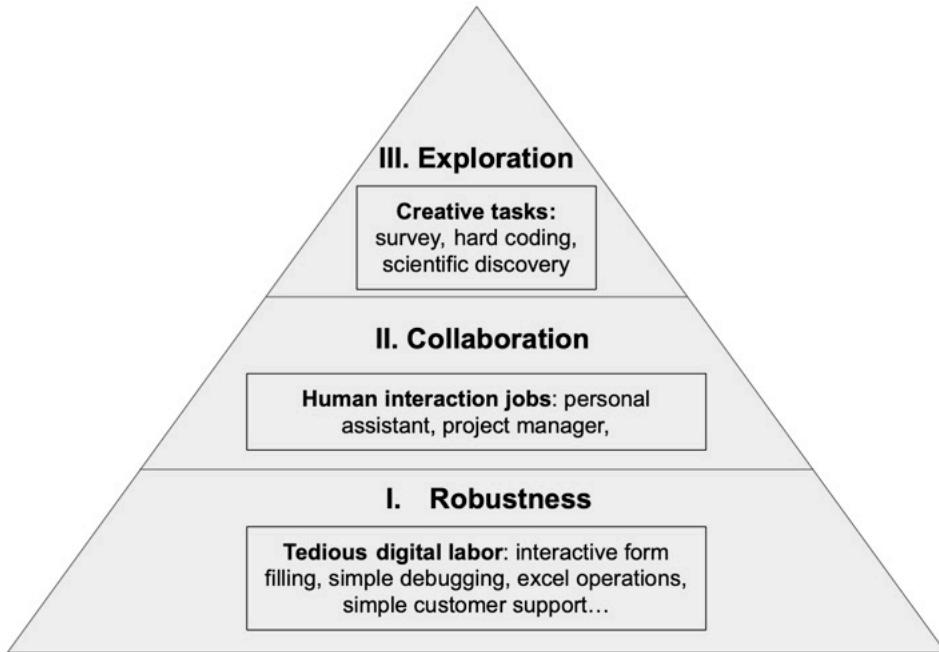


Figure 4: $\text{pass}^k (-)$ and $\text{pass}@k (..)$ in τ -retail.

Agent cannot see private database or human info - need to interact with them via tools or chat !

τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains

What's next?



Training

FireAct: Toward Language Agent Fine-tuning.

Interface

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering

Robustness

Human

τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains

Benchmark

<https://princeton-nlp.github.io/language-agent-impact/>

To-do: Sudoku Solving with LLM Agents

Objective: Build an LLM agent to solve Sudoku puzzles and compare it with a constraint propagation baseline.

Task:

- Implement a baseline solver using backtracking + constraint propagation
- Implement an LLM-based agent using one of these approaches:
 - Direct prompting (single call for complete solution)
 - Chain-of-thought (step-by-step reasoning) [not in this task]
- Try ReAct agent (reasoning + tools: check_constraint, get_candidates, fill_cell)
- Test on 10 Sudoku puzzles (varying difficulty)

You should deliver:

- Python code for baseline and LLM agent
- Results comparison: accuracy, solve time, number of LLM calls
- Discuss: What works? Where does the LLM struggle compared to search?