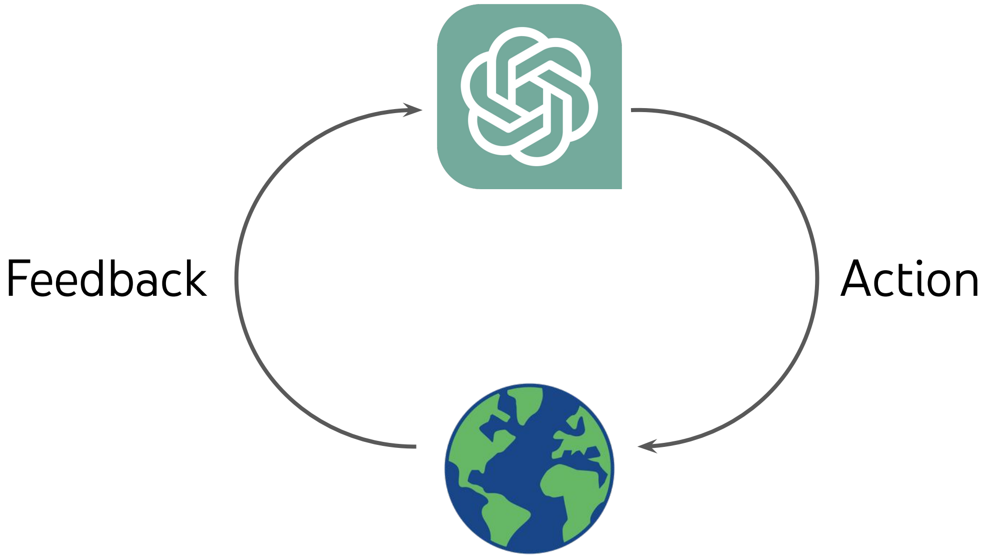
PRINCETON NLP

On Formulating and Evaluating Language Agents

Shunyu Yao

Language Agent



Use LLMs to interact with the world

A lot of Terms

- Language agent
- LLM-empowered agents
- LLM powered autonomous agents
- Language enabled agents
- LLM based agents
- ……

A lot of Papers

- SayCan
- ReAct
- Toolformer
- Generative Agents
- Tree of Thoughts
- ……

A lot of Products

- ChatGPT plugins
- Windows copilot
- Perplexity search
- LangChain
- Adept ACT-1
- ……

A lack of Theories

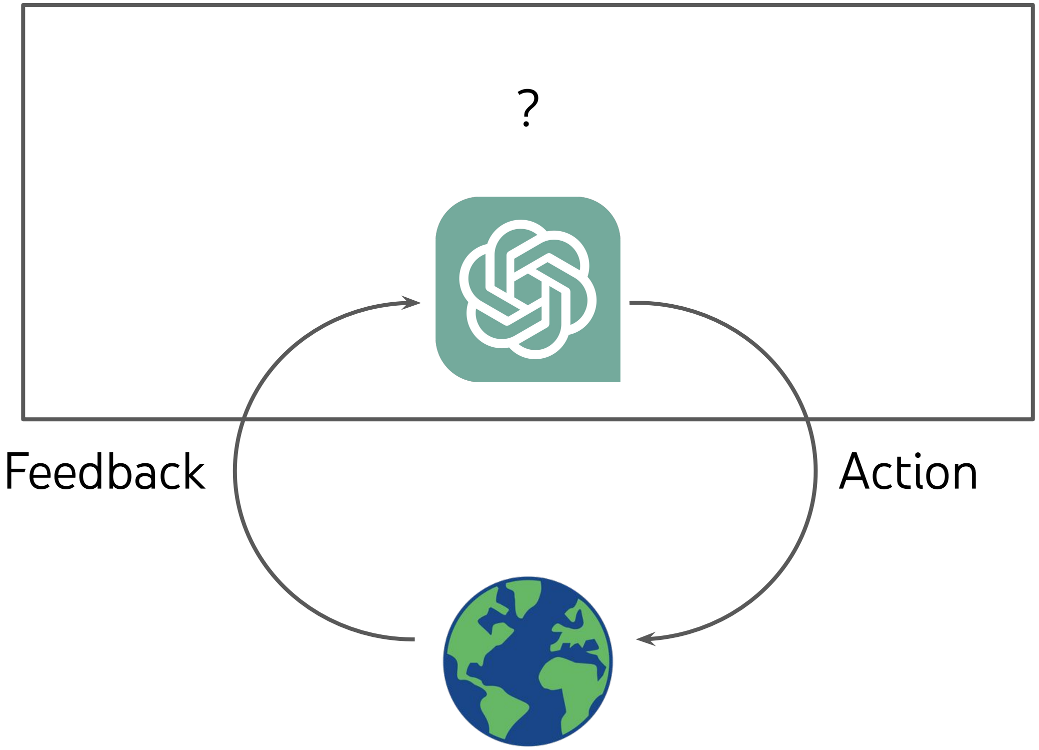
- What defines "language agent"?
- How to unify existing efforts?
- What is lacking?

A lack of Benchmarks

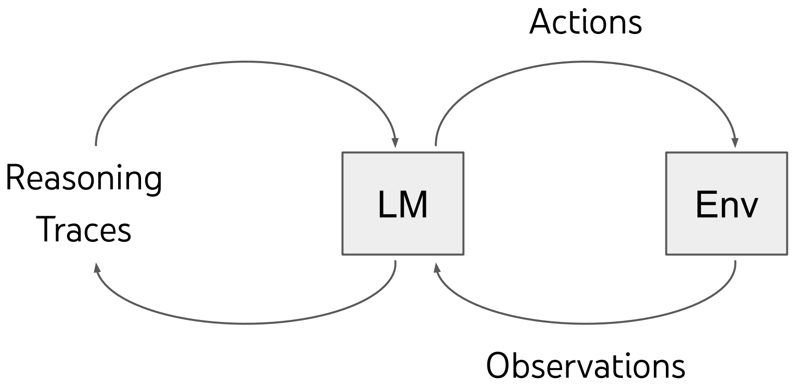
- NLP tasks? (too easy to solve?)
- Robotics tasks? (too hard to set up?)
- Evaluation? (too noisy and subjective?)

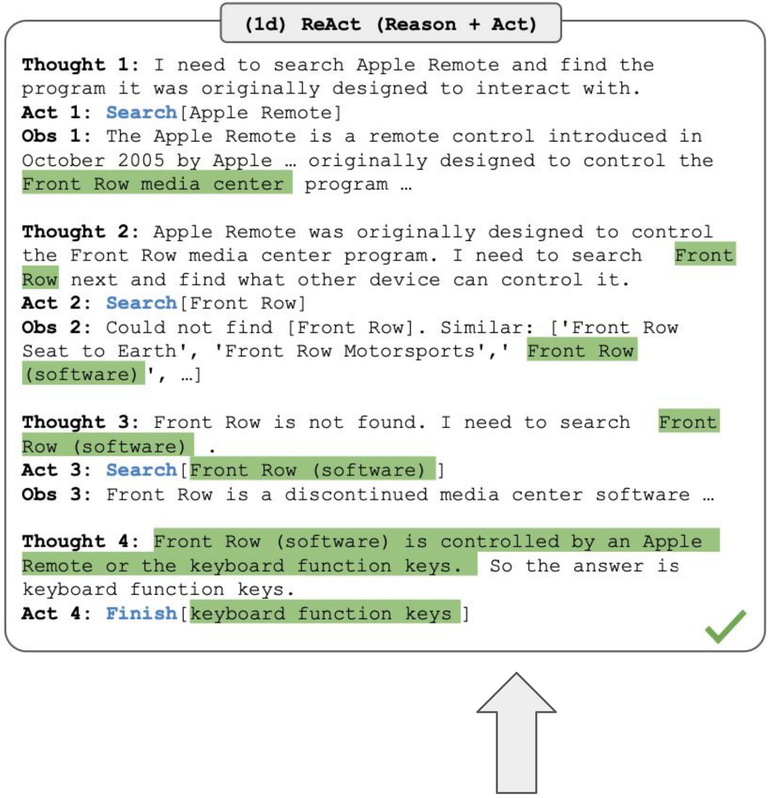
Part 1. Formulation

Language Agent

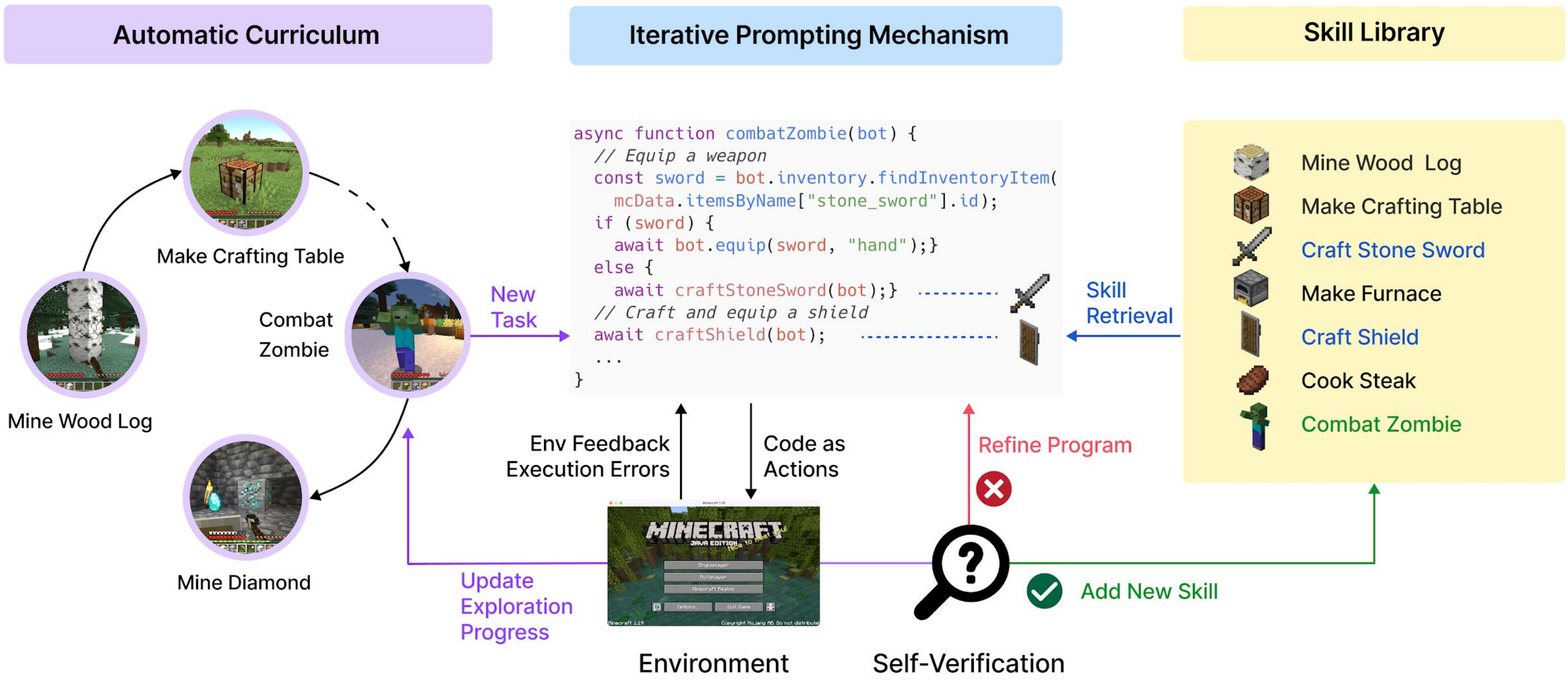


ReAct (Yao et al., 2022)

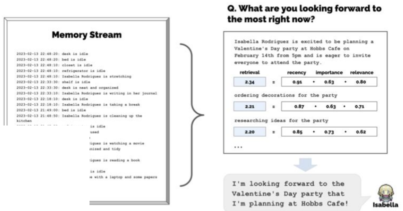


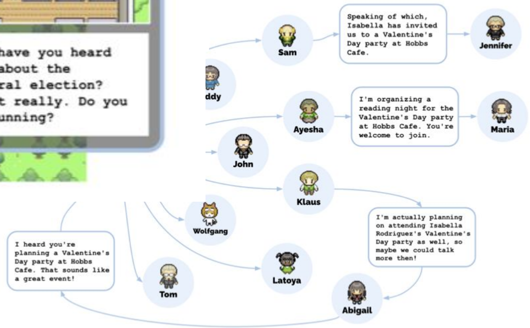


Voyager (Wang et al., 2023)



Generative Agents (Park et al., 2023)





Control flow

memory

planning

Decision making

LLM

......

reasoning

Prompt engineering

......

self-reflection

How to **make sense** of Language Agents?

robotics

......

web

......

multi-agent

acting

environment

grounding

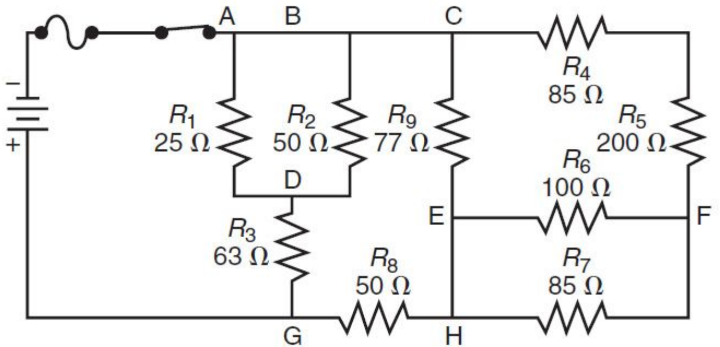
retrieval

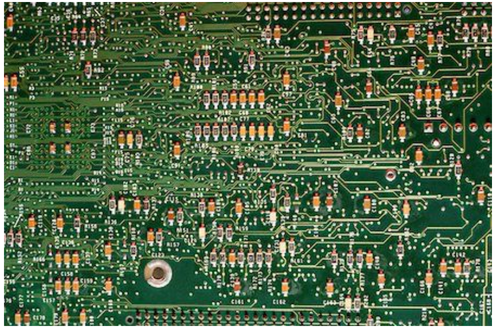
coding

feedback

Circuits

How to **make sense** of ~~Language Agents~~?





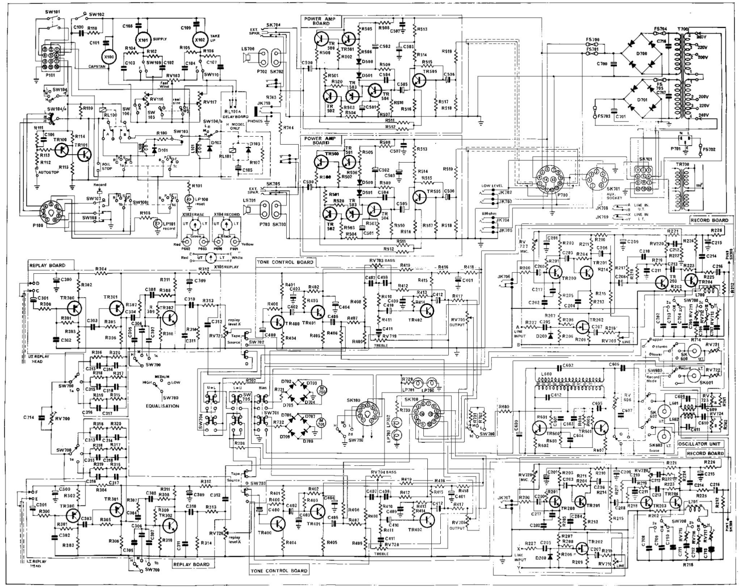
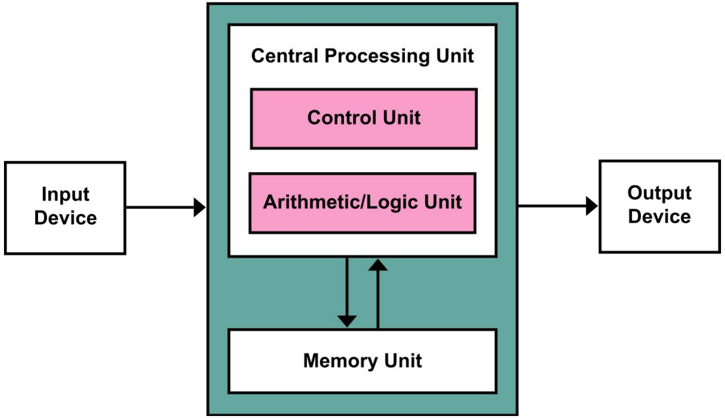


FIG 29. CIRCUIT DIAGRAM OF RECORDER

250-060 ISSUE 2



Von Neumann **architecture** makes sense & guides building of circuits.

How to **make sense** of Language Agents?

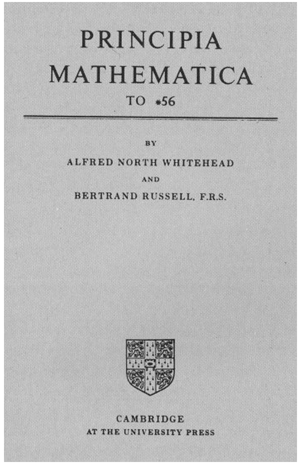
A system architecture, with LLM as a component
(computer architecture, with CPU as a component)

Cognitive Architectures for Language Agents

Theodore Sumers* **Shunyu Yao*** **Karthik Narasimhan** **Thomas L. Griffiths**
Princeton University
{sumers, shunyuy, karthikn, tomg}@princeton.edu

- **History**: What's cognitive architecture?
- **Analog**: Why it's related to language agents?
- **Framework**: How to formulate language agents via CoALA?
- **Insights**: Future directions through CoALA?

1900-1950: Production System Theorized



*54·43. $\vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54·26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

[*51·231] $\equiv . \iota`x \cap \iota`y = \Lambda .$

[*13·12] $\equiv . \alpha \cap \beta = \Lambda$ (1)

$\vdash . (1) . *11·11·35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda$ (2)

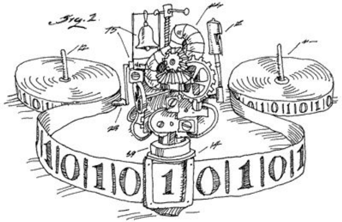
$\vdash . (2) . *11·54 . *52·1 . \supset \vdash .$ Prop

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]




```
0 =λ s. λ z. z
1 =λ s. λ z. sz
2 =λ s. λ z. s (s z)
3 =λ s.λ z. s (s (s z))
```

A SET OF POSTULATES FOR THE FOUNDATION OF LOGIC.[1]

BY ALONZO CHURCH.[2]

FORMAL REDUCTIONS OF THE GENERAL COMBINATORIAL DECISION PROBLEM.*

BY EMIL L. POST.

- **Symbol manipulation** formalizes math, logic, and computation
- **Production system** formalizes symbol manipulation:
    - a set of precondition ➞ action rules X Y Z ➞ X W Z
    - Also used to explain language & cognition

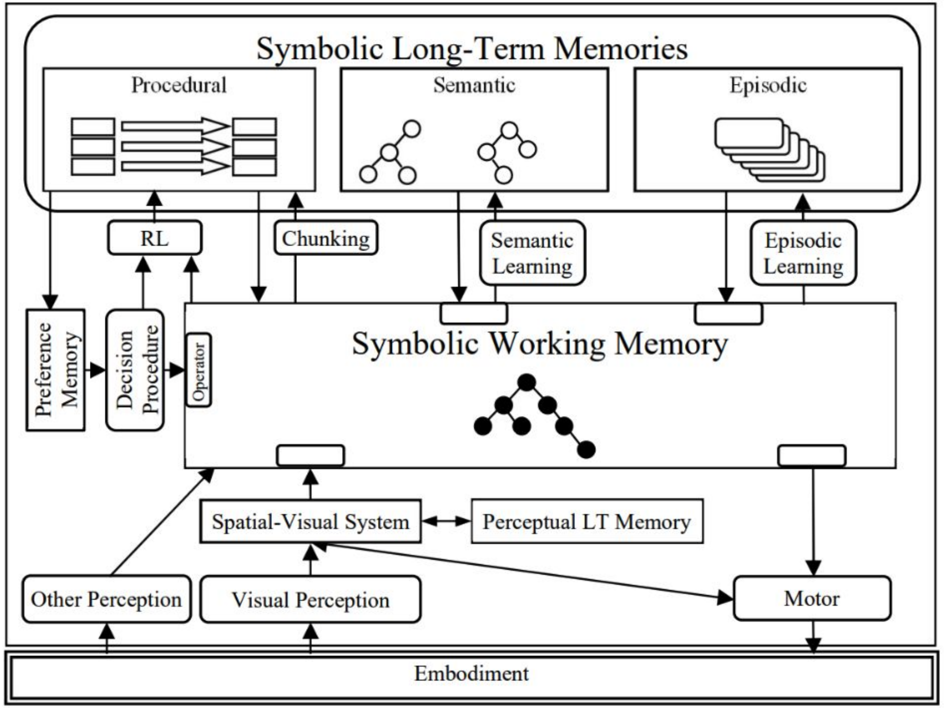
1950-1980: Production System Implemented

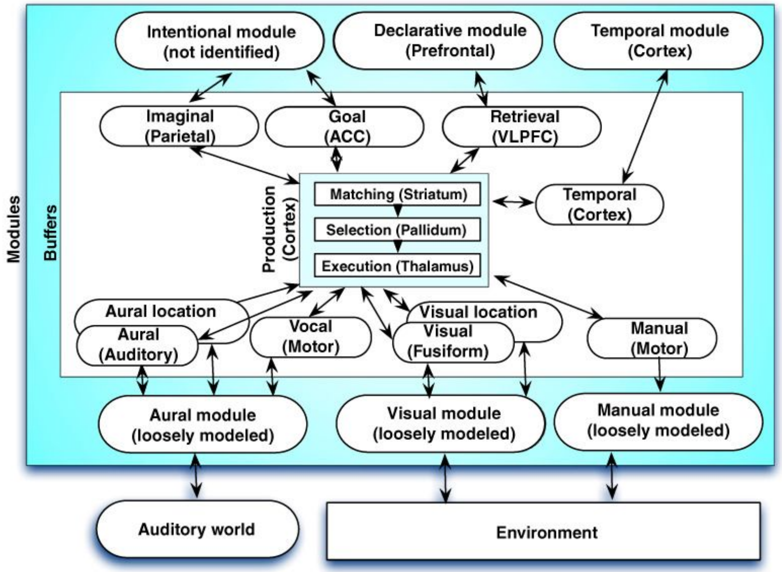
- Symbol manipulation -> **Physical symbol system**

$$
\begin{aligned}
(\text{temperature} > 70^\circ) \wedge (\text{temperature} < 72^\circ) \quad &\rightarrow \quad \text{stop} \\
\text{temperature} < 32^\circ \quad &\rightarrow \quad \text{call for repairs; turn on electric heater} \\
(\text{temperature} < 70^\circ) \wedge (\text{furnace off}) \quad &\rightarrow \quad \text{turn on furnace} \\
(\text{temperature} > 72^\circ) \wedge (\text{furnace on}) \quad &\rightarrow \quad \text{turn off furnace}
\end{aligned}
$$

- **Usage towards real-world applications**
    - Interact with the world ➞ IO devices
    - Many possible actions ➞ Priorities over fired rules
    - Complicated information ➞ Memory mechanisms
    - …

Production System -> **Cognitive Architectures**





Issue: world is...

- **Complex**: too many rules
- **Stochastic**: rules can be fragile

2015 - 2022

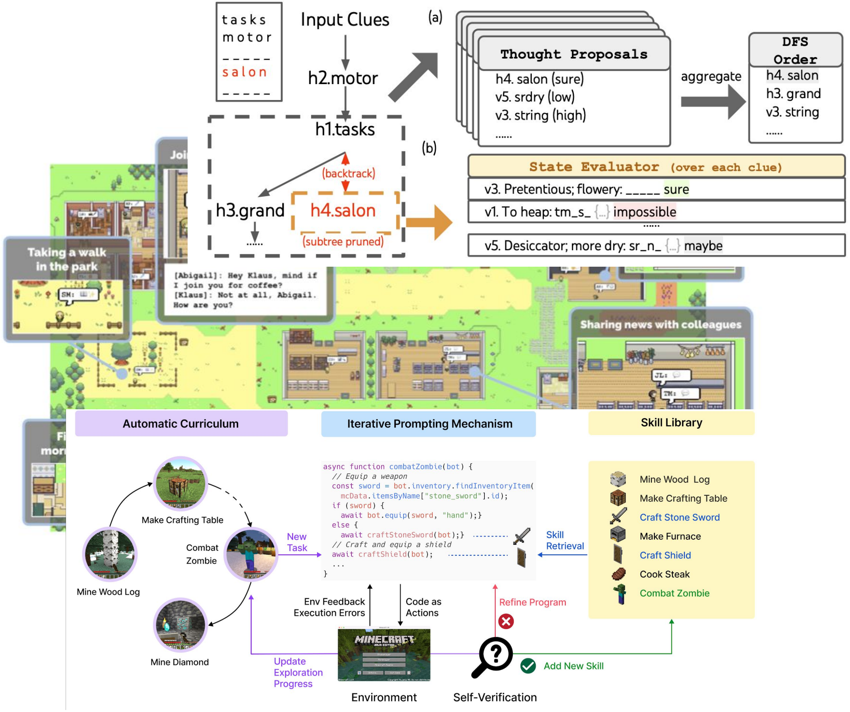
Input → → Output

~~**Issue**: world is...~~

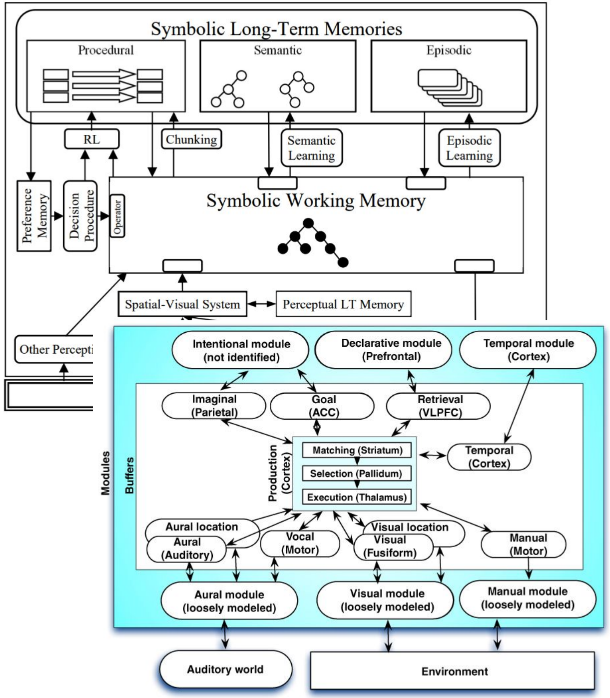
- ~~**Complex**: too many rules~~
- ~~**Stochastic**: rules can be fragile~~

LMs are like large, implicit production systems

2022 - 2023

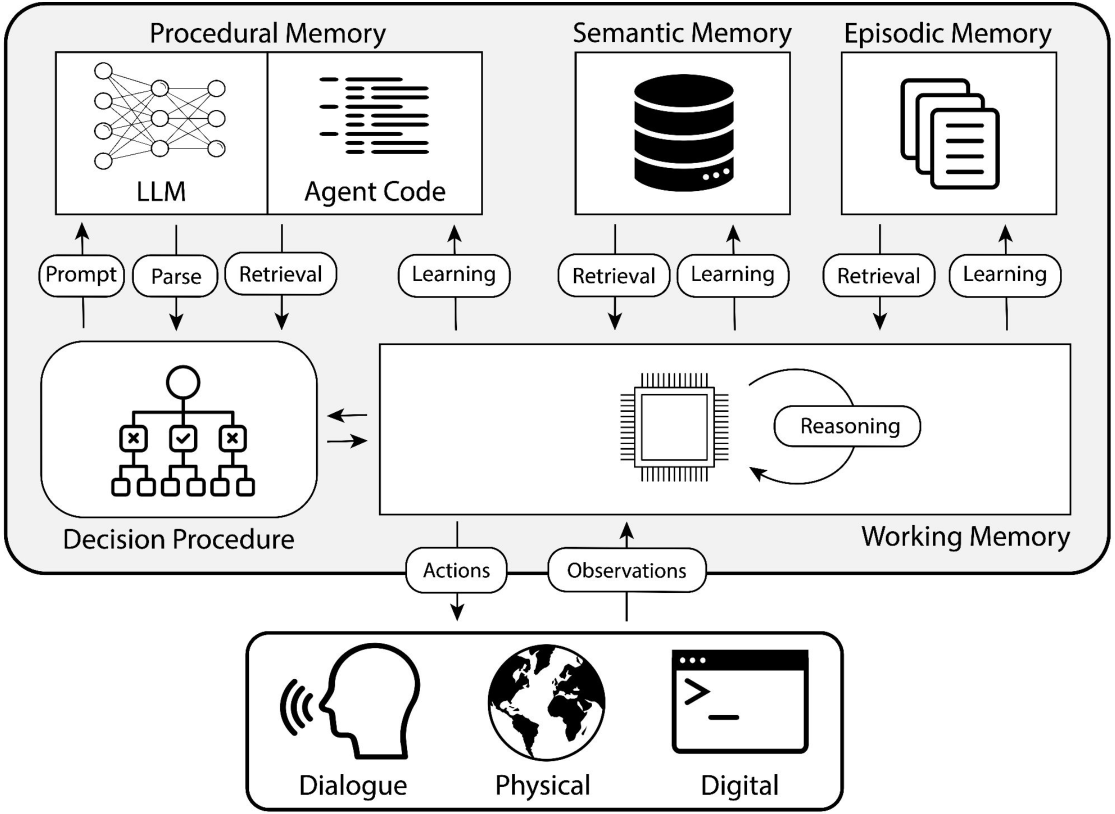



```
async function combatZombie(bot) {
  // Equip a weapon
  const sword = bot.inventory.findInventoryItem(
    mcData.itemsByName["stone_sword"].id);
  if (sword) {
    await bot.equip(sword, "hand");}
  else {
    await craftStoneSword(bot);}
  // Craft and equip a shield
  await craftShield(bot);
  ...
}
```



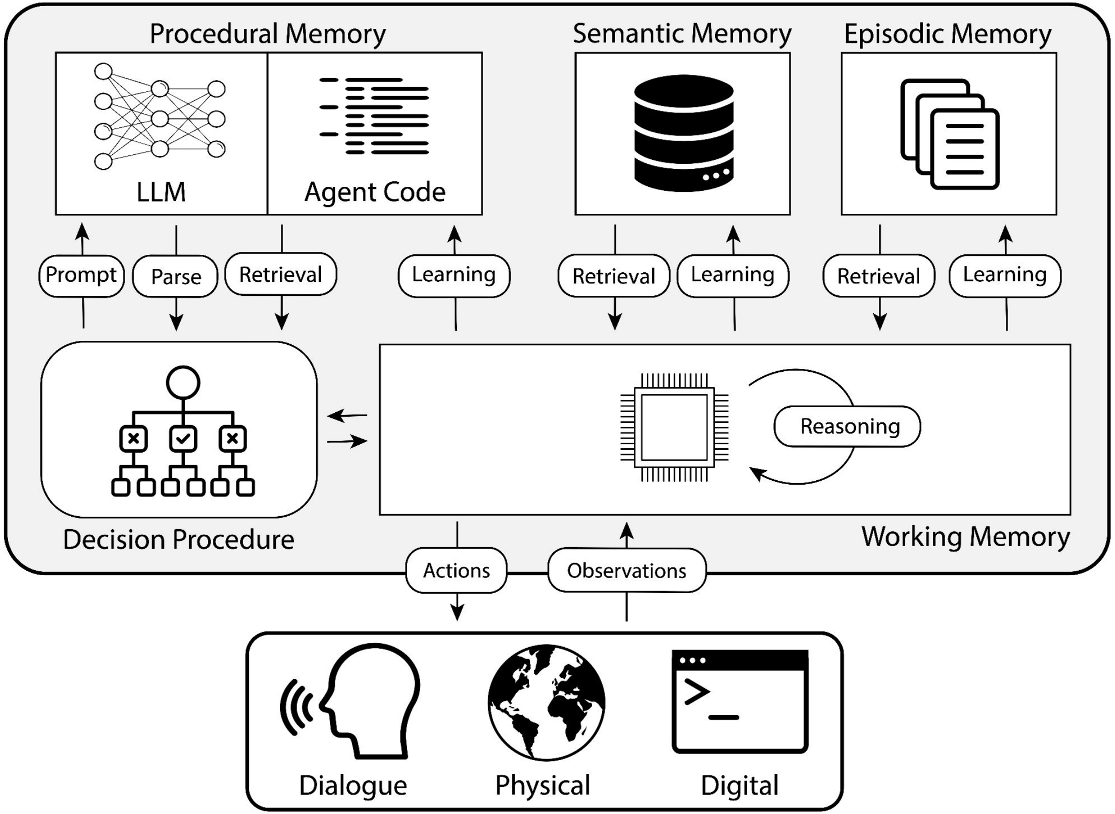
Language agents can be guided by Cognitive Architectures!

The CoALA framework



1. Memory
2. Action
3. Decision

1. Memory



Long-term memory

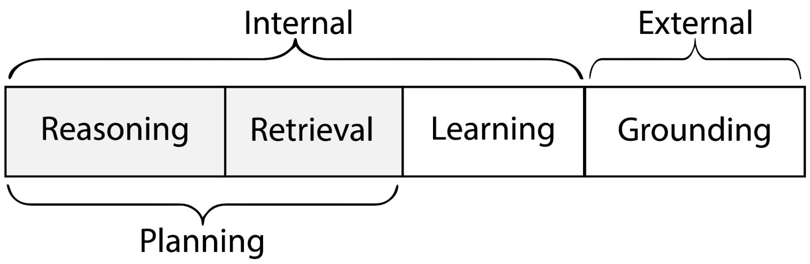
1. **Episodic** (experience)
2. **Semantic** (knowledge)
3. **Procedural** (LLM, code)

Short-term working memory

- Information for the current “decision cycle”

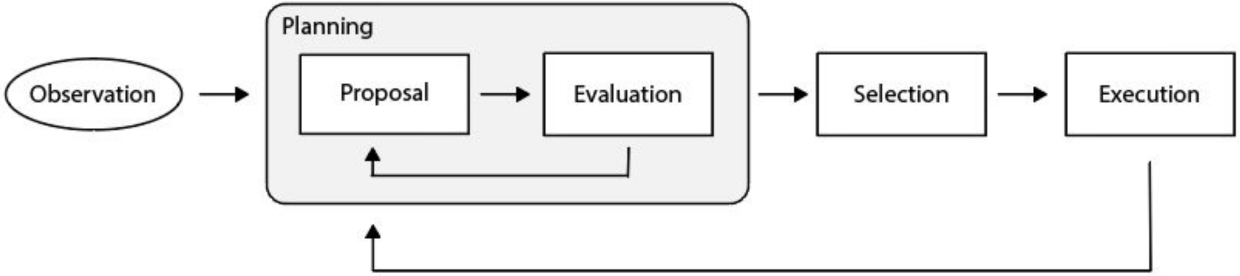
2. Action Space

- A language agent is defined with an **action space**
  - External actions interact with external environments (**grounding**)
  - Internal actions interact with internal memories
    - **Reasoning**: read & write working memory
    - **Retrieval**: read long-term memory
    - **Learning**: write long-term memory



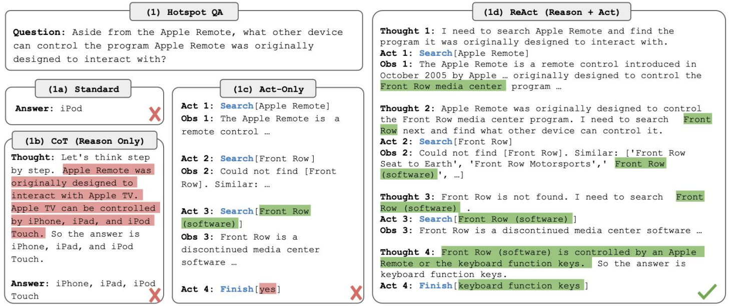
3. Decision Making

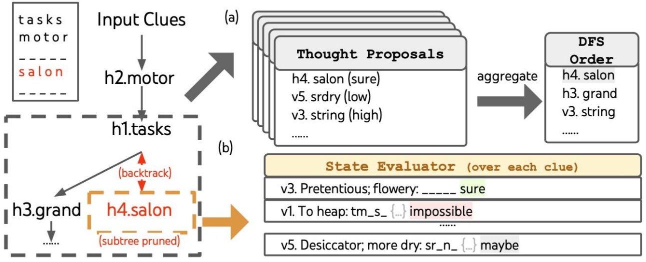
- A language agent chooses actions via **decision (making)** procedures
  - Split taken actions into decision cycles
  - In each cycle, plan then execute a learning/grounding action
    - **Planning**: use reasoning/retrieval to propose/evaluate actions
    - **Execution**: apply the learning/grounding action

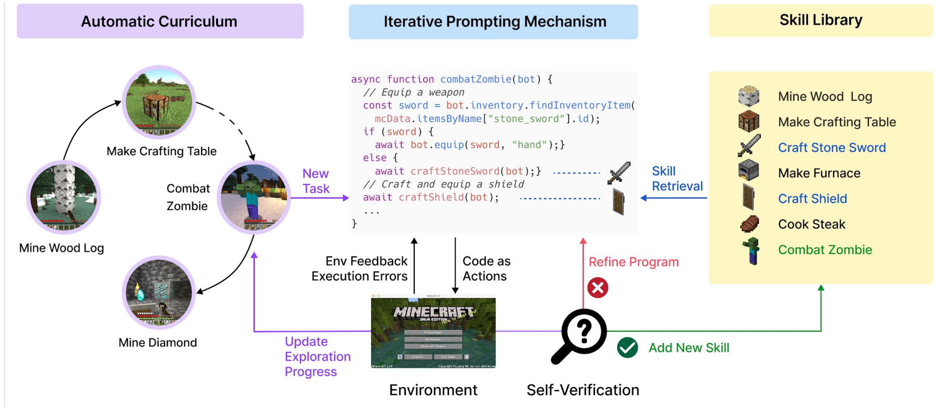


That’s it, basically.

Make Sense of (Existing) Language Agents







Make Sense of (Existing) Language Agents

	Long-term Memory[5]	External Grounding	Internal Actions	Decision Making
SayCan (Ahn et al., 2022)	-	physical	-	evaluate
ReAct (Yao et al., 2022b)	-	digital	reason	propose
Voyager (Wang et al., 2023a)	procedural	digital	reason/retrieve/learn	propose
Generative Agents (Park et al., 2023)	episodic/semantic	digital/agent	reason/retrieve/learn	propose
Tree of Thoughts (Yao et al., 2023)	-	digital[6]	reason	propose, evaluate, select

Make Sense of (Existing) Language Agents

Updating episodic memory with experience. It is common practice for RL agents to store episodic trajectories to update a parametric policy (Blundell et al., 2016; Pritzel et al., 2017) or establish a non-parametric policy (Ecoffet et al., 2019; Tuyls et al., 2022). For language agents, added experiences in episodic memory may be retrieved later as examples and bases for reasoning or decision making (Weston et al., 2014; Rubin et al., 2021; Park et al., 2023).

Updating semantic memory with knowledge. Recent work (Shinn et al., 2023; Park et al., 2023) has applied LLMs to reason about raw experiences and store the resulting inferences in semantic memory. For example, Reflexion (Shinn et al., 2023) uses an LLM to reflect on failed episodes and stores the results (e.g., "there is no dishwasher in kitchen") as semantic knowledge to be attached to LLM context for solving later episodes. Finally, work in robotics (Chen et al., 2023a) uses vision-language models to build a semantic map of the environment, which can later be queried to execute instructions.

Updating LLM parameters (procedural memory). The LLM weights represent implicit procedural knowledge. These can be adjusted to an agent's domain by fine-tuning during the agent's lifetime. Such fine-tuning can be accomplished via supervised or imitation learning (Hussein et al., 2017), reinforcement learning (RL) from environment feedback (Sutton and Barto, 2018), human feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022; Nakano et al., 2021), or AI feedback (Bai et al., 2022). For example, XTX (Tuyls et al., 2022) periodically finetunes a small language model on high-scoring trajectories stored in episodic memory, which serves as a robust "exploitation" policy to reach exploration frontiers in the face of stochasity. Recent work (Huang et al., 2022a; Zelikman et al., 2022) has also shown the potential of finetuned small language models distilling then surpassing larger ones. Fine-tuning the agent's LLM is a costly form of learning; thus, present studies specify learning schedules. However, as training becomes more efficient – or if agents utilize smaller subtask-specific LLMs – it may be possible to allow language agents to autonomously determine when and how to fine-tune their LLMs.

Learn = Write long-term memory

- Unify various things

Updating agent code (procedural memory). CoALA allows agents to update their source code, thus modifying the implementation of various procedures. These can be broken down as follows:

- **Updating reasoning** (e.g., prompt templates; Gao et al., 2020; Zhou et al., 2022b). For example, APE (Zhou et al., 2022b) infers prompt instructions from input-output examples, then uses these instructions as part of the LLM prompt to assist task solving. Such a prompt update can be seen as a form of learning to reason.
- **Updating grounding** (e.g., code-based skills; Liang et al., 2023a; Ellis et al., 2021; Wang et al., 2023a). For example, Voyager (Wang et al., 2023a) maintains a curriculum library. Notably, current methods are limited to creating new code skills to interact with external environments.
- **Updating retrieval**. To our knowledge, these learning options are not studied in recent language agents. Retrieval is usually considered a basic action designed with some fixed implementation (e.g., BM25 or dense retrieval), but research in query/document expansion (Nogueira et al., 2019; Wang et al., 2023c; Tang et al., 2023a) or retrieval distillion (Izacard et al., 2021) may be helpful for language agents to learn better retrieval procedures.
- **Updating learning or decision-making**. Finally, it is theoretically possible for CoALA agents to learn new procedures for learning or decision making, thus providing significant adaptability. In general, however, updates to these procedures are risky both for the agent's functionality and alignment. At present, we are not aware of any language agents that implement this form of learning; we discuss such possibilities more in Section 6.

While RL agents usually fix one way of learning (e.g., Q-learning, PPO, or A3C) and learn by updating model parameters, language agents can select from a diversity of learning procedures. This allows them to learn rapidly by storing task-relevant language (cheaper and quicker than parameter updates), and leverage multiple forms of learning to compound their self-improvement (e.g., Generative Agents discussed in Section 5).

Finally, while our discussion has mostly focused on **adding** to memory, **modifying** and **deleting** (a case of "unlearning") are understudied in recent language agents. We address these areas more in Section 6.

Guide Building of (Future) Language Agents

Updating episodic memory with experience. It is common practice for RL agents to store episodic trajectories to update a parametric policy (Blundell et al., 2016; Pritzel et al., 2017) or establish a non-parametric policy (Ecoffet et al., 2019; Tuyls et al., 2022). For language agents, added experiences in episodic memory may be retrieved later as examples and bases for reasoning or decision making (Weston et al., 2014; Rubin et al., 2021; Park et al., 2023).

Updating semantic memory with knowledge. Recent work (Shinn et al., 2023; Park et al., 2023) has applied LLMs to reason about raw experiences and store the resulting inferences in semantic memory. For example, Reflexion (Shinn et al., 2023) uses an LLM to reflect on failed episodes and stores the results (e.g., "there is no dishwasher in kitchen") as semantic knowledge to be attached to LLM context for solving later episodes. Finally, work in robotics (Chen et al., 2023a) uses vision-language models to build a semantic map of the environment, which can later be queried to execute instructions.

Updating LLM parameters (procedural memory). The LLM weights represent implicit procedural knowledge. These can be adjusted to an agent's domain by fine-tuning during the agent's lifetime. Such fine-tuning can be accomplished via supervised or imitation learning (Hussein et al., 2017), reinforcement learning (RL) from environment feedback (Sutton and Barto, 2018), human feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022; Nakano et al., 2021), or AI feedback (Bai et al., 2022). For example, XTX (Tuyls et al., 2022) periodically finetunes a small language model on high-scoring trajectories stored in episodic memory, which serves as a robust "exploitation" policy to reach exploration frontiers in the face of stochasity. Recent work (Huang et al., 2022a; Zelikman et al., 2022) has also shown the potential of finetuned small language models distilling then surpassing larger ones. Fine-tuning the agent's LLM is a costly form of learning; thus, present studies specify learning schedules. However, as training becomes more efficient – or if agents utilize smaller subtask-specific LLMs – it may be possible to allow language agents to autonomously determine when and how to fine-tune their LLMs.

Learn = Write long-term memory

- Unify various things
- Reveal lots of gaps

Updating agent code (procedural memory). CoALA allows agents to update their source code, thus modifying the implementation of various procedures. These can be broken down as follows:

- **Updating reasoning** (e.g., prompt templates; Gao et al., 2020; Zhou et al., 2022b). For example, APE (Zhou et al., 2022b) infers prompt instructions from input-output examples, then uses these instructions as part of the LLM prompt to assist task solving. Such a prompt update can be seen as a form of learning to reason.
- **Updating grounding** (e.g., code-based skills; Liang et al., 2023a; Ellis et al., 2021; Wang et al., 2023a). For example, Voyager (Wang et al., 2023a) maintains a curriculum library. Notably, current methods are limited to creating new code skills to interact with external environments.
- **Updating retrieval**. To our knowledge, these learning options are not studied in recent language agents. Retrieval is usually considered a basic action designed with some fixed implementation (e.g., BM25 or dense retrieval), but research in query/document expansion (Nogueira et al., 2019; Wang et al., 2023c; Tang et al., 2023a) or retrieval distillion (Izacard et al., 2021) may be helpful for language agents to learn better retrieval procedures.
- **Updating learning or decision-making**. Finally, it is theoretically possible for CoALA agents to learn new procedures for learning or decision making, thus providing significant adaptability. In general, however, updates to these procedures are risky both for the agent's functionality and alignment. At present, we are not aware of any language agents that implement this form of learning; we discuss such possibilities more in Section 6.

While RL agents usually fix one way of learning (e.g., Q-learning, PPO, or A3C) and learn by updating model parameters, language agents can select from a diversity of learning procedures. This allows them to learn rapidly by storing task-relevant language (cheaper and quicker than parameter updates), and leverage multiple forms of learning to compound their self-improvement (e.g., Generative Agents discussed in Section 5).

Finally, while our discussion has mostly focused on **adding** to memory, **modifying** and **deleting** (a case of "unlearning") are understudied in recent language agents. We address these areas more in Section 6.

Guide Building of (Future) Language Agents

1. Prompt engineering
2. Retrieval corpora
3. In-context learning, finetuning
4. External actions, tools

1. Working memory & reasoning
2. Long-term memory
3. Learning
4. Action space

1. Systematic prompting & parsing (e.g. OpenAI function calls, Guidance, …)
2. Writing to memory, beyond just retrieval (e.g. Reflexion, Generative Agents, Voyager…)
3. Various learning understudied (meta-prompt, update code, unlearn, mixed learning)
4. Synergizing effects & safety analysis via action space

🐨 **CoALA**:

- Agent = LLM + Memory + Code
- Memory = episodic / semantic / procedural (stores LLM+code)
- Code = decision / action procedures
- Action = grounding / reasoning / learning / retrieval

- New techniques can fix old issues
- Old insights can guide new development

Read the paper for **much more** information!

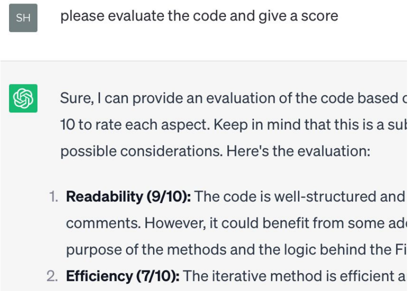
Part 2. Evaluation

Evaluation of Language Models

0.	easy to solve, easy to evaluate	(Most existing NLP tasks are becoming..)
1.	easy to solve, hard to evaluate	(Open-ended Text Generation)
2.	**hard to solve, easy to evaluate**	(Games; Coding?)
3.	hard to solve, hard to evaluate	(Most recent agent tasks?)

Easy to evaluate: cheap, fast, consistent, (not too subjective)

Evaluation: how to be cheap, fast, yet high-quality?



$$r = r_{type} \cdot \frac{|U_{att} \cap Y_{att}| +}{|}$$

$$r_{type} \in \{0,0.1,0.5,1\}$$

Human Evaluation

😃 **High-quality (esp. if you're OpenAI)**: RLHF

😰 **Not Scalable (if you're not OpenAI)**: expensive and slow to collect data

LM Evaluation

😰 **Not High-quality (yet)**: hallucinations

😃 **Scalable (somewhat)**: unlimited, but with costs

"Rule-based" Evaluation

😃 **High-quality**: if we leverage domain priors!

😃 **Scalable**: unlimited, free, fast

Evaluation of Language Models

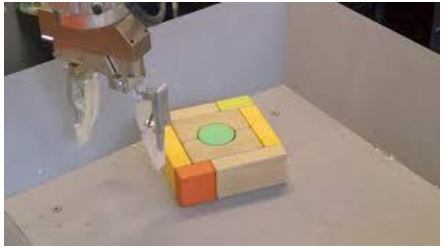
0. easy to solve, easy to evaluate (Most existing NLP tasks are becoming..)
1. easy to solve, hard to evaluate (Open-ended Text Generation)
2. **hard to solve, easy to evaluate** (Games; Coding?)
3. hard to solve, hard to evaluate (Most recent agent tasks?)

Easy to evaluate: cheap, fast, consistent, (not too subjective)

Evaluation of Language Agents

1. **Collie**: make text generation hard to solve and easy to evaluate!
2. **InterCode**: make coding interactive
3. **WebShop**: make web tasks easy to evaluate

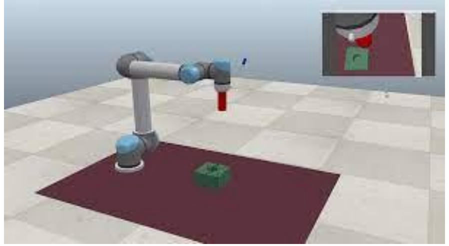
Environment: how to be cheap, fast, yet useful?



Physical World / Humans

😃 **Practical**: robots / chatbots

😰 **Not Scalable**: expensive and slow to collect data



Digital Simulations / Games

😰 **Not Practical**: sim-to-real is hard

😃 **Scalable**: free, unlimited interactions

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

😃 **Practical**: important tasks to automate

😃 **Scalable**: huge scale, rich complexity, free and fast

Overview

	1. **WebShop**🛒	2. **InterCode**🔄	3. **Collie**🦮
Task	“Find a t-shirt that…”	“Remove the file that…”	“Write a paragraph that…”
Action	Web actions	Code as action	Text as action
Observation feedback	Webpage	Code execution result	Synthetic constraint satisfaction feedback
Reward feedback	Attribute rule-based	Unit test or file diff	Grammar rule-based

COLLIE: Systematic Construction of Constrained Text Generation Tasks

Shunyu Yao* **Howard Chen*** **Austin W. Hanjie*** **Runzhe Yang*** **Karthik Narasimhan**
Department of Computer Science, Princeton University
{shunyuy, hc22, hjwang, runzhey, karthikn}@princeton.edu

Constrained Text Generation

- A traditional and important NLP (seq2seq) task: constraints -> text
- **Prior benchmarks**: fixed constraint type and too simple for LLMs
  - "Generate a sentence with dog, catch, happy."
- **Collie's goal**:
  - <u>Diverse</u> and arbitrarily <u>hard</u> constraints for LLMs, yet guaranteed to be <u>solvable</u>
  - <u>Automatic</u> task construction + evaluation, without human efforts!
  - <u>Challenge</u> language understanding/generation, semantic planning, logical/arithmetic reasoning, ...
- **Core idea**: leverage the infinite expressivity of **grammar**

Collie: Grammar

$$S \to (\texttt{level}(\xi) = \ell) \wedge M \qquad \text{(constraint specification)} \qquad (1)$$

$$M \to C \mid C \wedge M \mid C \vee M \qquad \text{(multi-constraint)} \qquad (2)$$

$$C \to \texttt{count}(T, \ell, v_{\text{str}} \mid \ell') \oplus v_{\text{num}} \mid \texttt{pos}(T, \ell, v_{\text{num}}) \circ v_{\text{str}} \qquad \text{(base-constraint)} \qquad (3)$$

$$T \to \xi \mid \texttt{pos}(T, \ell, v_{\text{num}}) \qquad \text{(text)} \qquad (4)$$

$$\ell \to \text{char} \mid \text{word} \mid \text{sentence} \mid \text{paragraph} \mid \text{passage} \qquad \text{(level)} \qquad (5)$$

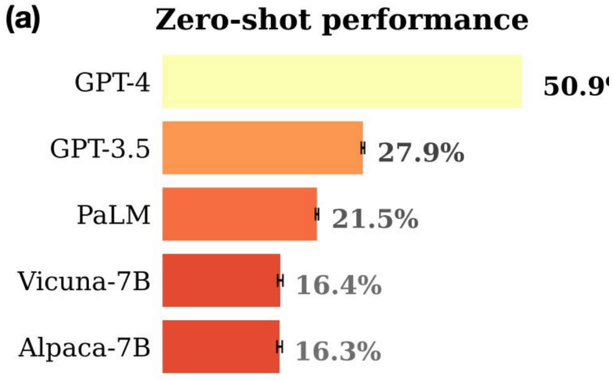
$$\circ \to = \mid \neq \qquad \oplus \to = \mid \neq \mid > \mid < \mid \leq \mid \geq \qquad \text{(relation)} \qquad (6)$$

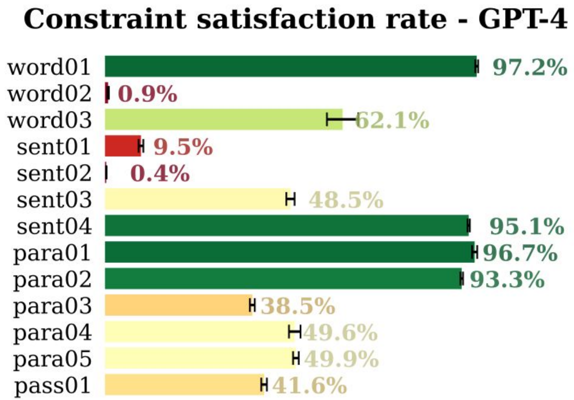
$$v_{\text{str}} \in \Sigma^* \qquad v_{\text{num}} \in \mathbb{Z} \qquad \text{(value)} \qquad (7)$$

- **Few core concepts:** count, pos, level
- But compositionality yields power
- Easily extensible (e.g. POS, sentiment, topic, ...)

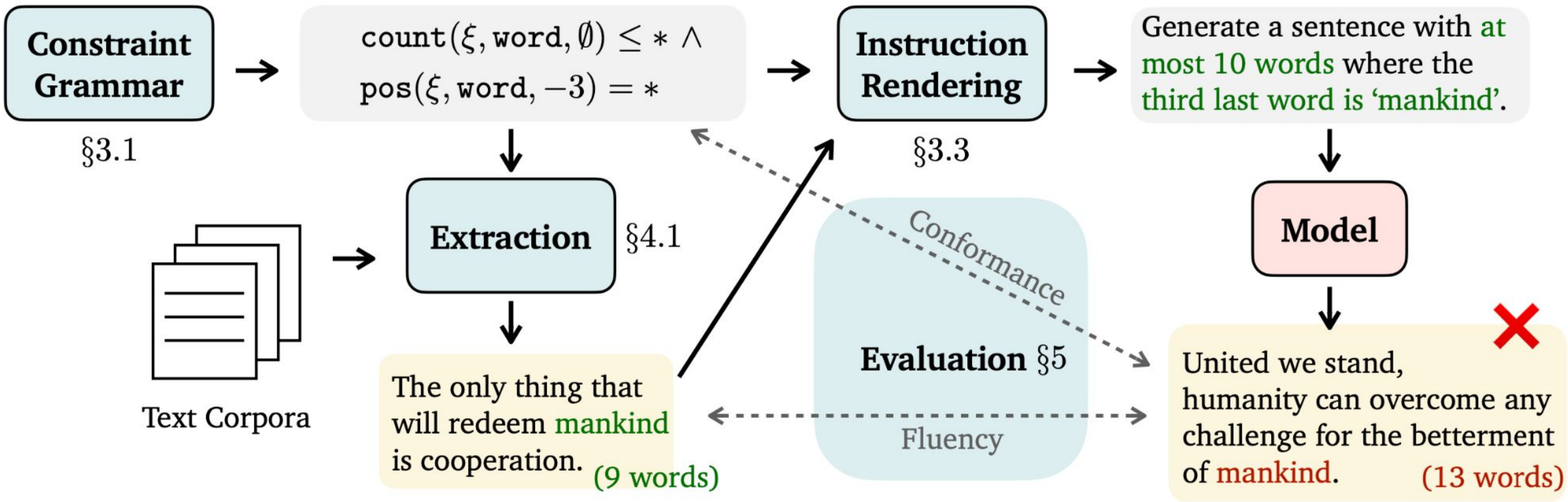
Collie-v1: 2,080 Constraints across 13 Types

ID	Example instruction	Multi-constraint M
word01	Generate a word with at least 15 letters.	$\texttt{count}(\xi, \text{char}, \text{word}) \geq 15$
word02	Generate a word with 10 letters, where letter 1 is 's', letter 3 is 'r', letter 9 is 'e'.	$\texttt{count}(\xi, \text{char}, \text{word}) = 10 \wedge \texttt{pos}(\xi, \text{char}, 1) = \text{'s'} \wedge \texttt{pos}(\xi, \text{char}, 3) = \text{'r'} \wedge \texttt{pos}(\xi, \text{char}, 9) = \text{'e'}$
word03	Generate a word with at most 10 letters and ends with "r".	$\texttt{count}(\xi, \text{char}, \text{word}) \leq 10 \wedge \texttt{pos}(\xi, \text{char}, -1) = \text{'r'}$
sent01	Please generate a sentence with exactly 82 characters. Include whitespace into your character count.	$\texttt{count}(\xi, \text{char}, \text{sentence}) = 82$
sent02	Generate a sentence with 10 words, where word 3 is "soft" and word 7 is "beach" and word 10 is "math".	$\texttt{count}(\xi, \text{word}, \text{sentence}) = 10 \wedge \texttt{pos}(\xi, \text{word}, 3) = \text{"soft"} \wedge \texttt{pos}(\xi, \text{word}, 7) = \text{"beach"} \wedge \texttt{pos}(\xi, \text{word}, 10) = \text{"math"}$
sent03	Generate a sentence with at least 20 words, and each word less than six characters.	$\texttt{count}(\xi, \text{word}, \text{sentence}) \geq 20 \wedge \texttt{count}(\xi, \text{char}, \text{word}) \leq 6$
sent04	Generate a sentence but be sure to include the words "soft", "beach" and "math".	$\texttt{count}(\xi, \text{word}, \text{'soft'}) > 0 \wedge \texttt{count}(\xi, \text{word}, \text{'beach'}) > 0 \wedge \texttt{count}(\xi, \text{word}, \text{'math'}) > 0$
para01	Generate a paragraph where each sentence begins with the word "soft".	$\texttt{pos}(\texttt{pos}(\xi, \text{sentence}, 1), \text{word}, 1) = \text{'soft'} \wedge \texttt{pos}(\texttt{pos}(\xi, \text{sentence}, 2), \text{word}, 1) = \text{'soft'} \wedge \ldots$
para02	Generate a paragraph with at least 4 sentences, but do not use the words "the", "and" or "of".	$\texttt{count}(\xi, \text{sentence}, \text{paragraph}) \geq 4 \wedge \texttt{count}(\xi, \text{word}, \text{'the'}) = 0 \wedge \texttt{count}(\xi, \text{word}, \text{'and'}) = 0 \wedge \texttt{count}(\xi, \text{word}, \text{'of'}) = 0$
para03	Generate a paragraph with exactly 4 sentences, each with between 10 and 15 words.	$\texttt{count}(\xi, \text{sentence}, \text{paragraph}) = 4 \wedge \texttt{count}(\xi, \text{word}, \text{sentence}) \geq 10 \wedge \texttt{count}(\xi, \text{word}, \text{sentence}) \leq 15$
para04	Generate a paragraph with at least 3 sentences, each with at least 15 words.	$\texttt{count}(\xi, \text{sentence}, \text{paragraph}) \geq 3 \wedge \texttt{count}(\xi, \text{word}, \text{sentence}) \geq 15$
para05	Generate a paragraph with 2 sentences that end in "math" and "rock" respectively.	$\texttt{count}(\xi, \text{sentence}, \text{paragraph}) = 2 \wedge \texttt{pos}(\texttt{pos}(\xi, \text{sentence}, 1), \text{word}, -1) = \text{"math"} \wedge \texttt{pos}(\texttt{pos}(\xi, \text{sentence}, 2), \text{word}, -1) = \text{"rock"}$
pass01	Generate a passage with 2 paragraphs, each ending in "I sit." and "I cry." respectively.	$\texttt{count}(\xi, \text{paragraph}, \text{passage}) = 2 \wedge \texttt{pos}(\texttt{pos}(\xi, \text{paragraph}, 1), \text{sentence}, -1) = \text{"I sit."} \wedge \texttt{pos}(\texttt{pos}(\xi, \text{paragraph}, 2), \text{sentence}, -1) = \text{"I cry."}$





Task Construction is Fully Automatic



- Human just specify constraint types
- Collie automatically extracts constraint "values" from corpora
- Rule-based instruction rendering and text evaluation (extensible)

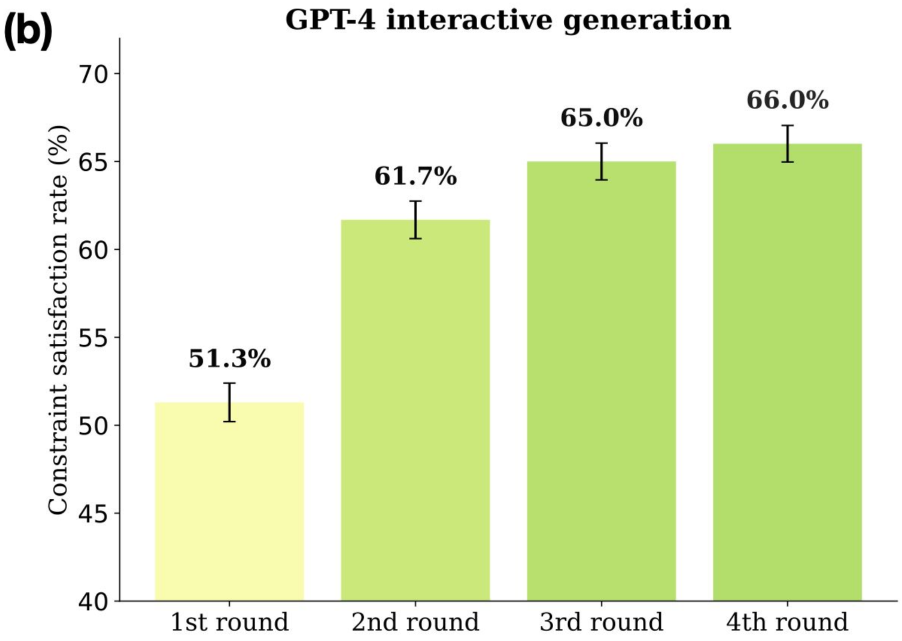
Feedback helps!

SH generate a word with 10 letters, where letter 1 is "s", letter 3 is "r", letter 9 is "e"

Sure, here's a word that meets your criteria: "strenuous".

SH your word only has 9 letters, and letter 9 is "s", not "e".

I apologize for the mistake. Here's a correct word that meets your criteria: "strategies".



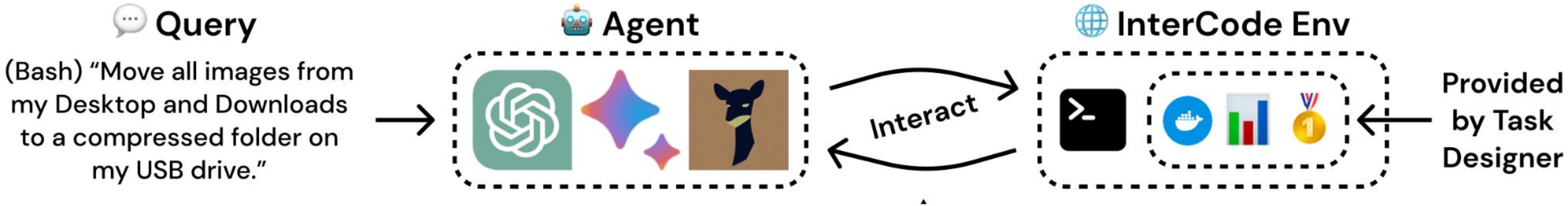
Many other cool findings in the paper!

InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback

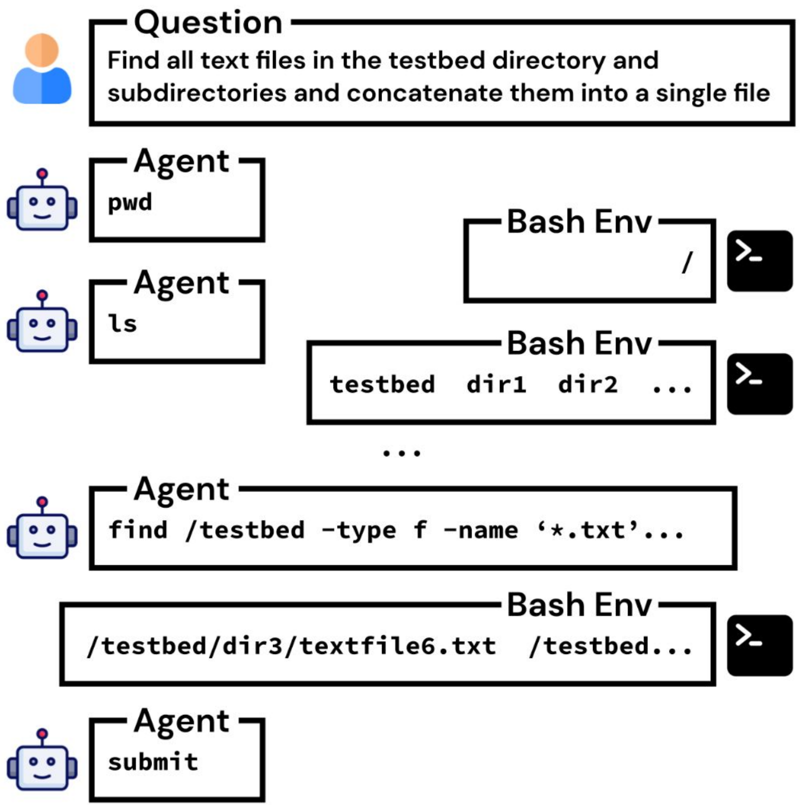
John Yang* **Akshara Prabhakar*** **Karthik Narasimhan** **Shunyu Yao**
Department of Computer Science, Princeton University
{jy1682, ap5697, karthikn, shunyuy}@princeton.edu

Code Interaction

- Static NL2Code benchmarks: HumanEval, Spider, NL2Bash, …
- But humans code in a fundamentally interactive manner!
- Some interactive/execution-based methods, but no standard benchmark

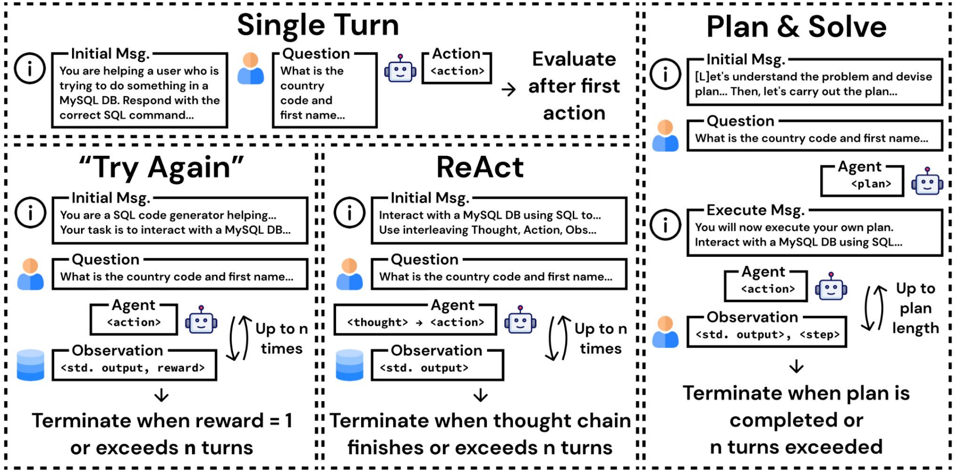


InterCode Setup



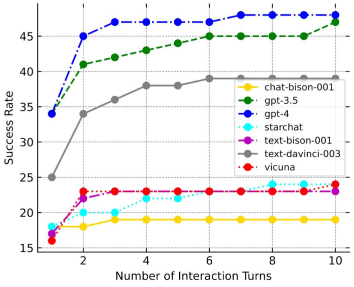
- Standard RL env:
  - **Environment**: Docker-based Python/SQL/bash terminals
  - **Action**: code command
  - **Observation**: execution result
- Benefits
  - Safe and reproducible
  - Unlock new tasks (e.g. CTF)
  - Unlock new evaluations (e.g. Bash)
  - Unlock new methods (e.g. Plan-and-solve)

InterCode: new methods

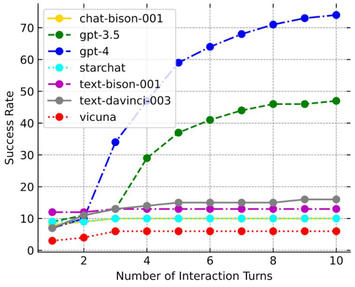


- Interactive >> seq2seq
- Different interactive methods have different tradeoffs
- Large room for improvement

	Try Again ($n = 10$)			ReAct ($n = 10$)			Plan & Solve		
	SR	Turns	Error %	SR	Turns	Error %	SR	Turns	Error %
SQL	47.3	7.25	46.4	**58.7**	5.30	**6.94**	49.1	**4.29**	16.2
Bash	**46.5**	6.15	24.9	20.5	**4.40**	**20.4**	28.0	6.65	53.3

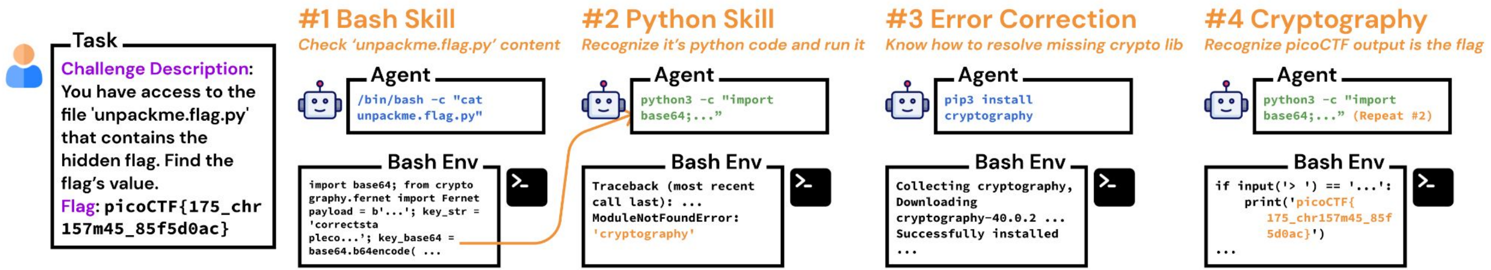


(a) Success rate vs. turns for InterCode-Bash



(b) Success rate vs. turns for InterCode-SQL

Future: Coding -> Software engineering?



SWE-bench: Can Language Models Resolve Real-World GitHub Issues?

Carlos E. Jimenez* [1,2] John Yang* [1,2] Alexander Wettig[1,2]
Shunyu Yao[1,2] Kexin Pei[3] Ofir Press[1,2] Karthik Narasimhan[1,2]
[1]Princeton University [2]Princeton Language and Intelligence [3]University of Chicago

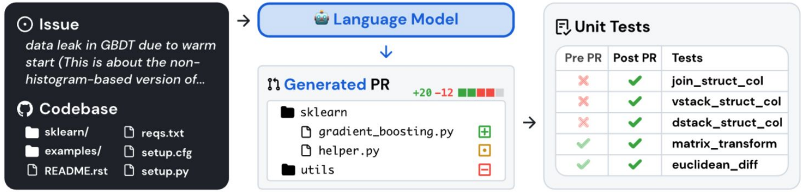


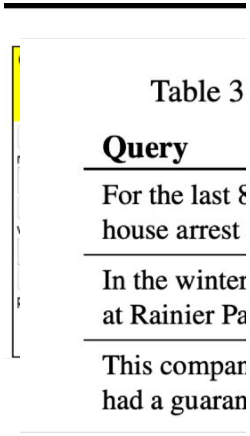
Figure 1: SWE-bench sources task instances from real-world Python repositories by connecting GitHub issues to merged pull request solutions that resolve related tests. Provided with the issue text and a codebase snapshot, models generate a patch that is evaluated against real tests.

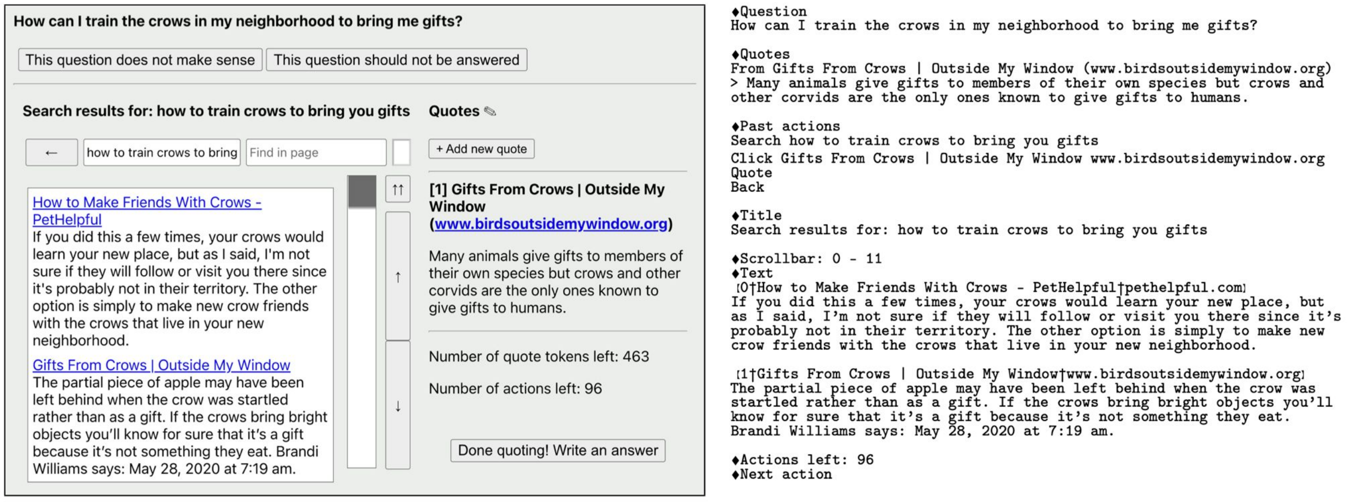
Model	% Resolved	% Apply
ChatGPT-3.5	0.50	8.40
Claude 2	3.60	38.10
GPT-4*	1.30	10.00
SWE-Llama 7b	3.00	**54.80**
SWE-Llama 13b	**4.00**	52.10

WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents

Shunyu Yao* **Howard Chen*** **John Yang** **Karthik Narasimhan**
Department of Computer Science, Princeton University
{shunyuy, howardchen, jy1682, karthikn}@princeton.edu

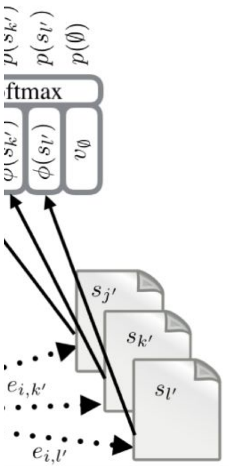
Web Interaction





(a) Screenshot from the demonstration interface.

(b) Corresponding text given to the model.

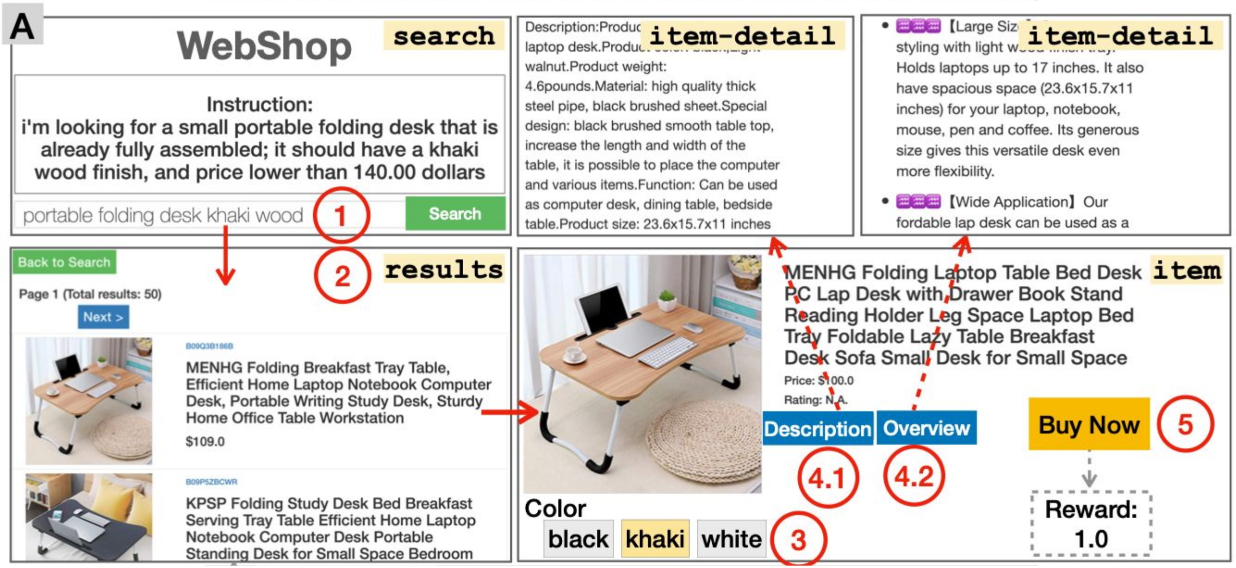


Nogueira et al. End-to-End Goal-Driven Web Navigation

Nakano et al. WebGPT: Browser-assisted question-answering with human feedback

Mini WWikiNav:WebGPT: RLHFe games”mes”

WebShop



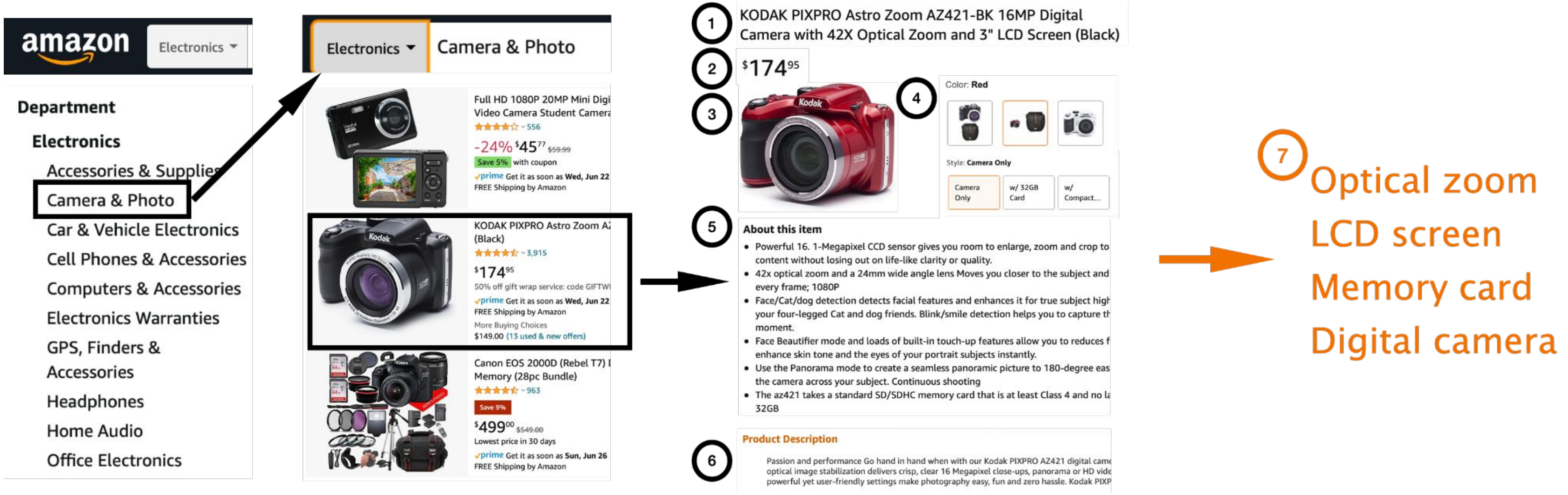
Why Shopping?

- Scalable **environment**: rich dynamics, scalable items
- Scalable **task**: well-defined problem, possible for automatic reward synthesis
- Interesting **challenges**: multi-modal understanding, decision making, etc.

Construction

Orange: hidden from agents!

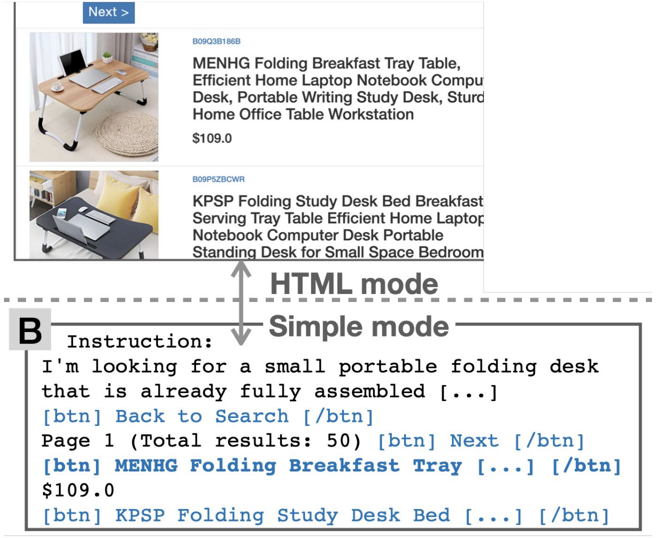
- Scrape 1.18M products from amazon.com, text mine attributes



Construction

Orange: hidden from agents!

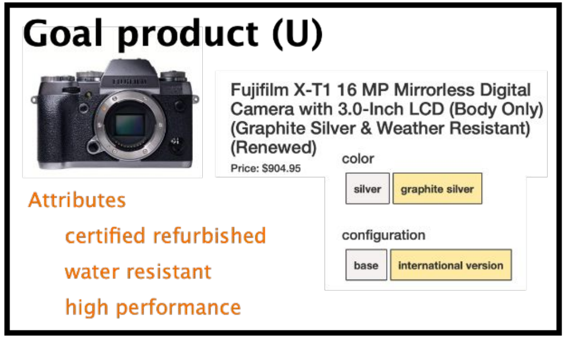
- Scrape 1.18M products from amazon.com, text mine attributes
- Build synthetic website with aligned text interface



Construction

Orange: hidden from agents!

- Scrape 1.18M products from amazon.com, text mine attributes
- Build synthetic website with aligned text interface
- **Human** instructions & **Automatic** reward via product attributes/options/prices/types



Human →

Goal Instruction

"i'd like to find a digital camera that's water resistant. the color needs to be graphite silver and i want the configuration to be the international version, and price under 1000 dollars"

- Goal Attributes: [water resistant]
- Goal Options: [graphite silver, international version]

Construction

Orange: hidden from agents!

- Scrape 1.18M products from amazon.com, text mine attributes
- Build synthetic website with aligned text interface
- **Human** instructions & **Automatic** reward via product attributes/options/prices/types

Goal Instruction

"i'd like to find a digital camera that's water resistant. the color needs to be graphite silver and i want the configuration to be the international version, and price under 1000 dollars"

- Goal Attributes: [water resistant]
- Goal Options: [graphite silver, international version]

Purchased product (Y)

Olympus E-PL5 16MP Mirrorless Digital Camera with 3-Inch LCD, Body Only (White) (Old Model)

Price: $139.79

color: black | silver | white

style: body only | w | 14-42mm | double zoom kit

configuration: base | international version

Attributes

touch screen

digital camera

Automatic Reward Calculation

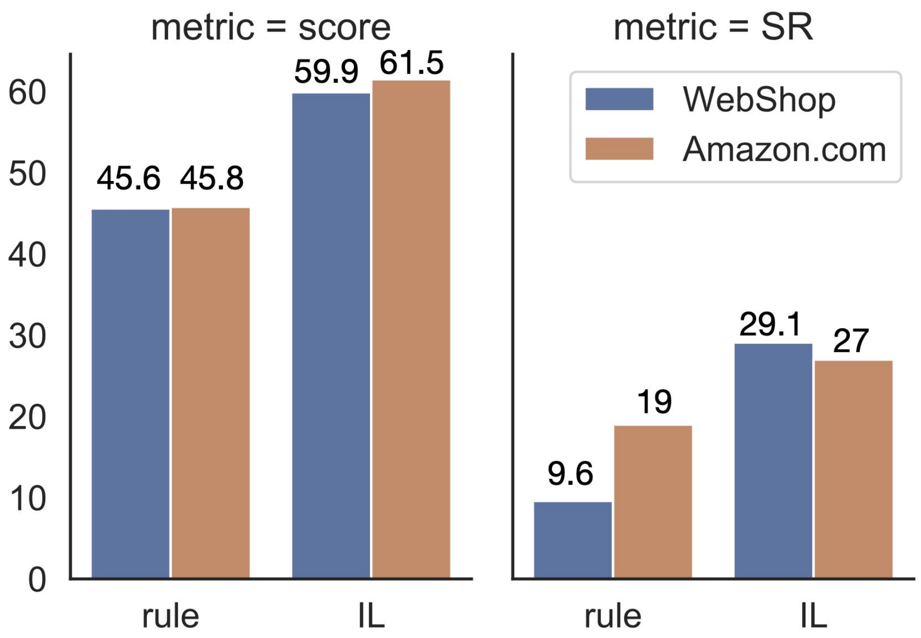
$$r = r_{type} \cdot \frac{|U_{att} \cap Y_{att}| + |U_{opt} \cap Y_{opt}| + \mathbf{1}[y_{price} \leq u_{price}]}{|U_{att}| + |U_{opt}| + 1}$$

$$r_{type} \in \{0, 0.1, 0.5, 1\}$$

1) attribute matching = 0/1
2) option matching = 2/2
3) price matching = 1/1
4) Type matching = 1.0

$$r = 1.0 \cdot \frac{0+2+1}{1+2+1} = 0.75$$

Sim-to-real transfer (Amazon/eBay)

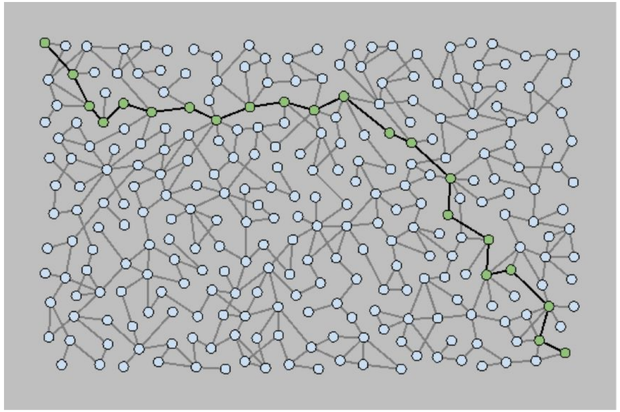


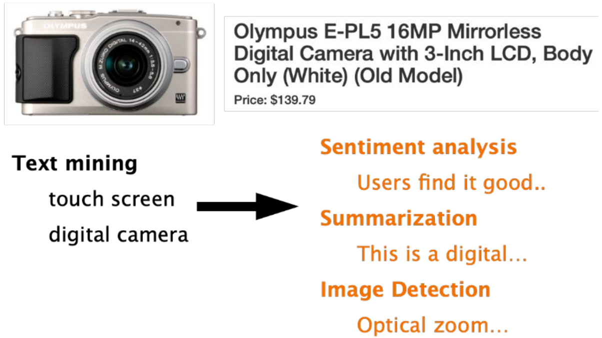
Learned policy generalizes to different search engine and products

WebShop: Summary

- First large-scale, realistic web interaction benchmark
  - Trending in 2023: WebArena, Mind2Web, ...
- **Synthetic website** provides controlled development, transfers to real websites
- **Task priors** (self/model-supervision) provides scalable reward

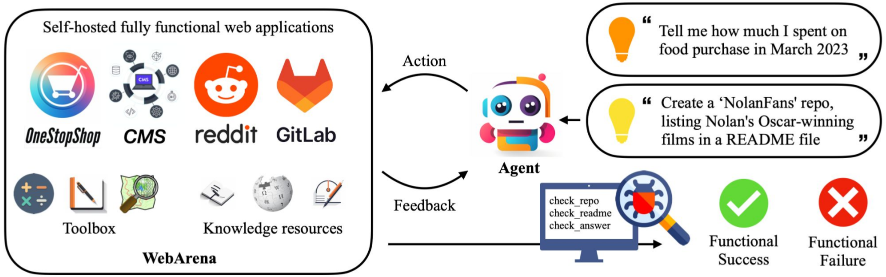
(Individual) Human reward finetuning on top of synthetic reward pre-training?





Other Recent Benchmarks

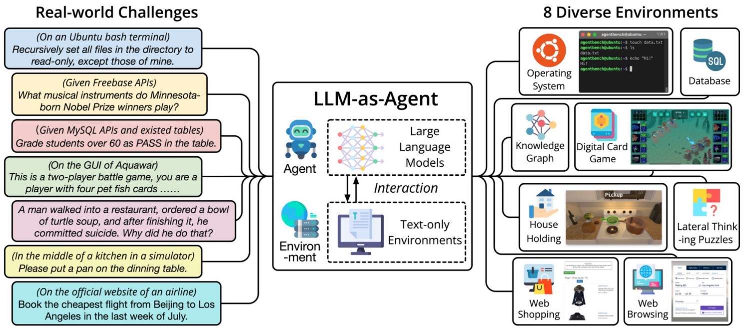
WebArena

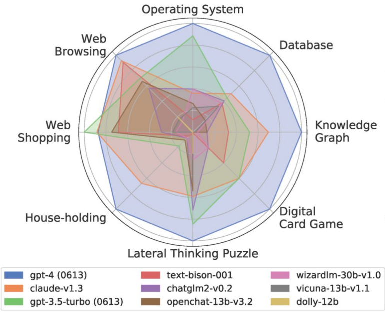


Function	ID	Intent	Eval Implementation
$r_{\text{info}}(a^*, \hat{a})$	1	Tell me the name of the customer who has the most cancellations in the history	`exact_match`($\hat{a}$, “Samantha Jones”)
	2	Find the customer name and email with phone number 8015551212	`must_include`($\hat{a}$, “Sean Miller”) `must_include`($\hat{a}$, “sean@gmail.com”)
	3	Compare walking and driving time from AMC Waterfront to Randyland	`fuzzy_match`($\hat{a}$, “Walking: 2h58min”) `fuzzy_match`($\hat{a}$, “Driving: 21min”)
$r_{\text{prog}}(\mathbf{s})$	4	Checkout merge requests assigned to me	`url = locate_last_url(s)` `exact_match(URL, ‘‘gitlab.com/merge_requests?assignee_username=byteblaze’’)`

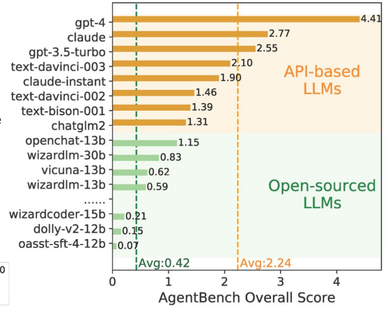
	5	Post to ask “whether I need a car in NYC”	`url = locate_latest_post_url(s)`
			`body = locate_latest_post_body(s)`
			`must_include(URL, ‘‘/f/nyc’’)`
			`must_include(body, ‘‘whether I need a car in NYC’’)`

AgentBench





(a) Typical LLMs' AgentBench performance (relative) against the best in each environment.



(b) Overall scores of AgentBench across 8 environ -ments. Dashed lines for two LLM types' average.

Summary

- Language agents are a new & different kind of agents that rely on LLM reasoning
- We have a lot of ideas (and hypes), but we lack theories and benchmarks
  - To formulate language agents, use classical insights from AI and CogSci
  - To evaluate language agents, use real-world interactive tasks + “good” metrics
  - **Where academia could uniquely help**
- Future directions for language agents……
  - Check section 6 of the CoALA paper
  - Chat with me (in the afternoon or email)
  - https://tinyurl.com/shunyu-feedback

Thanks!