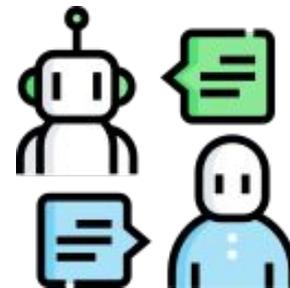


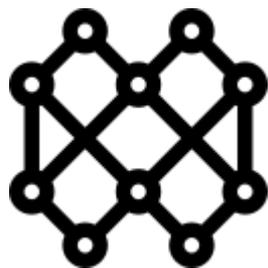
# Language Model-based Agents

Advanced NLP

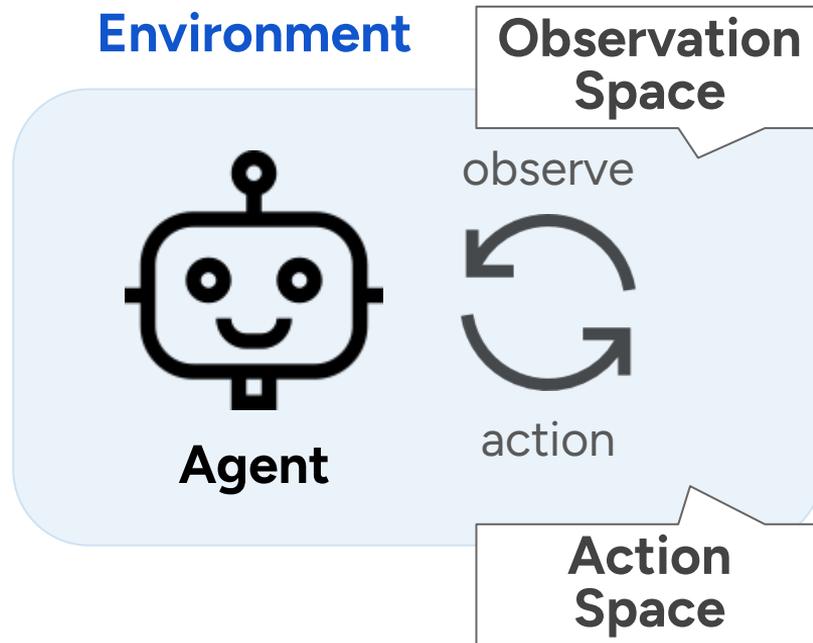
Zora Wang



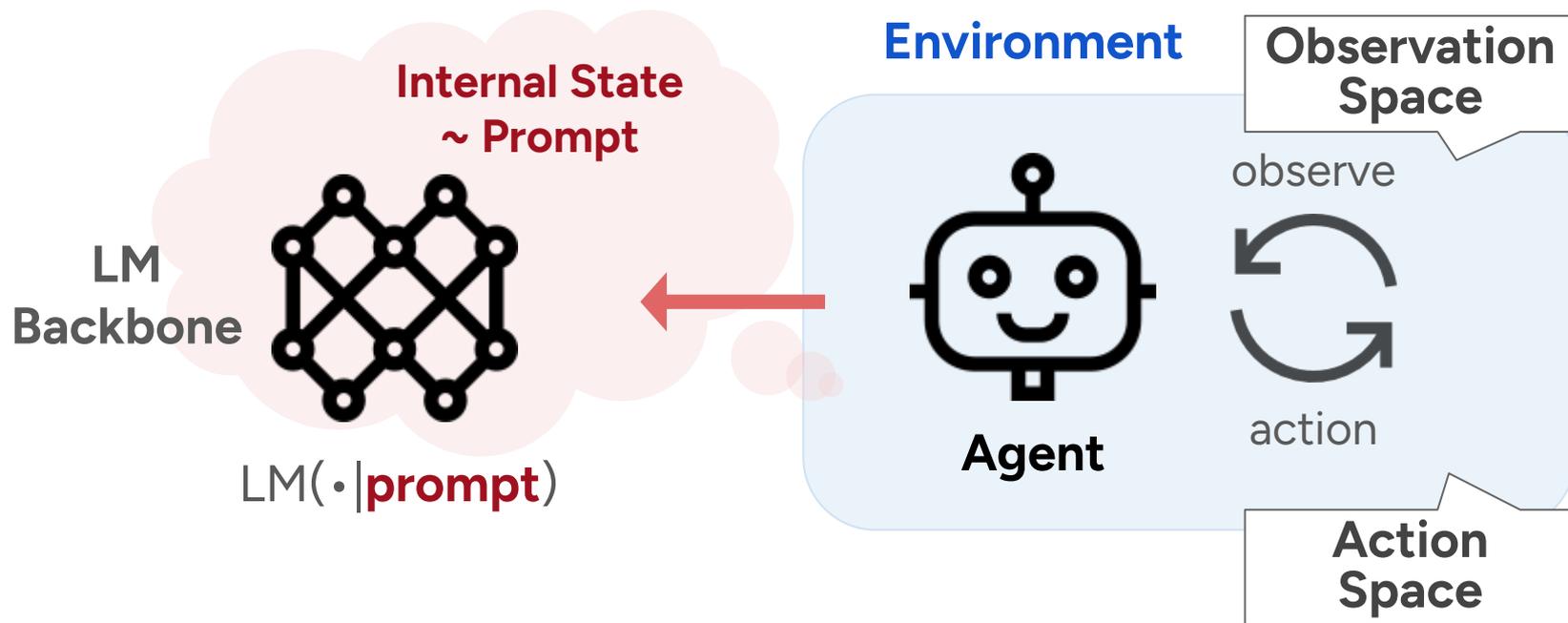
# From Language Modeling to Agents



LM



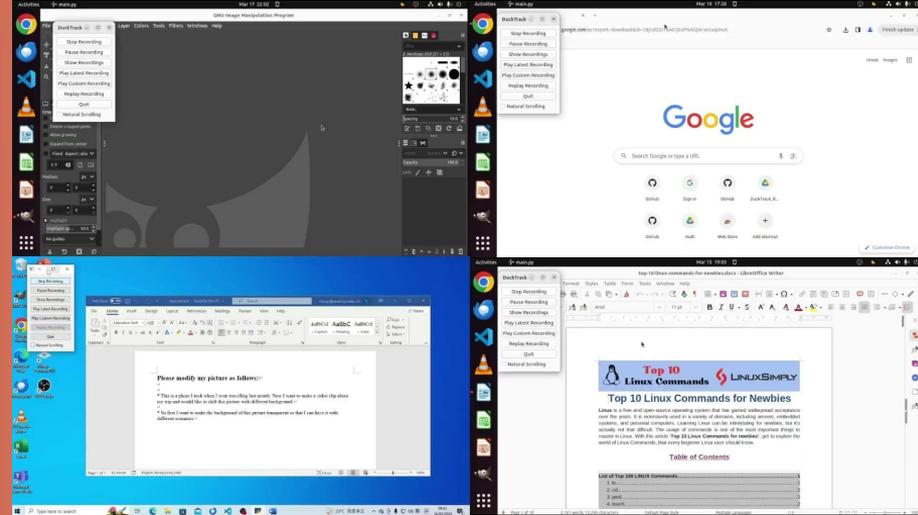
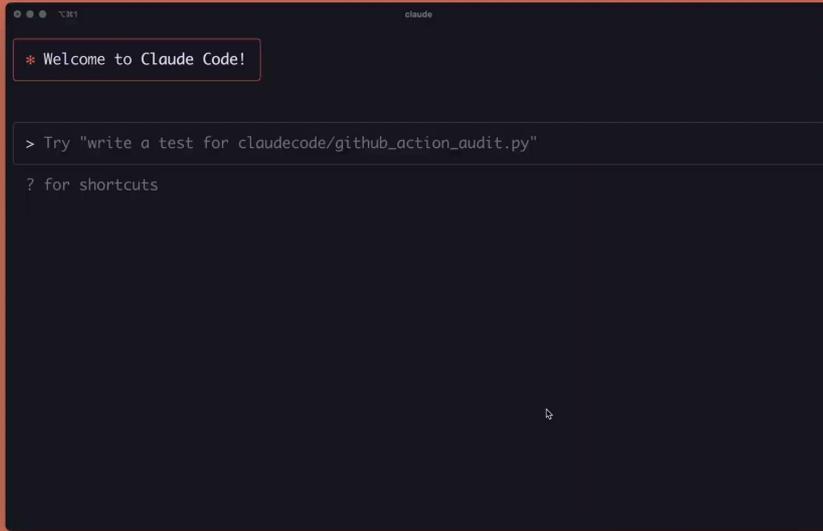
# What Are LM-based Agents?



# Examples of Agents: Digital Agents

Software Engineering

Web Navigation/Computer Use

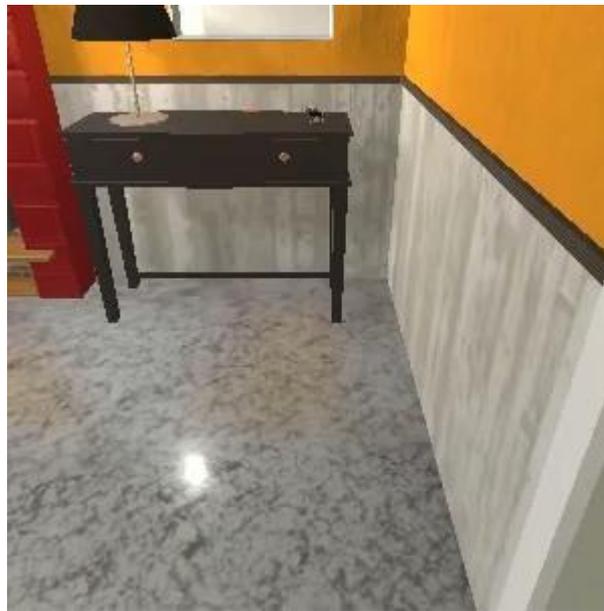


# Embodied Agents

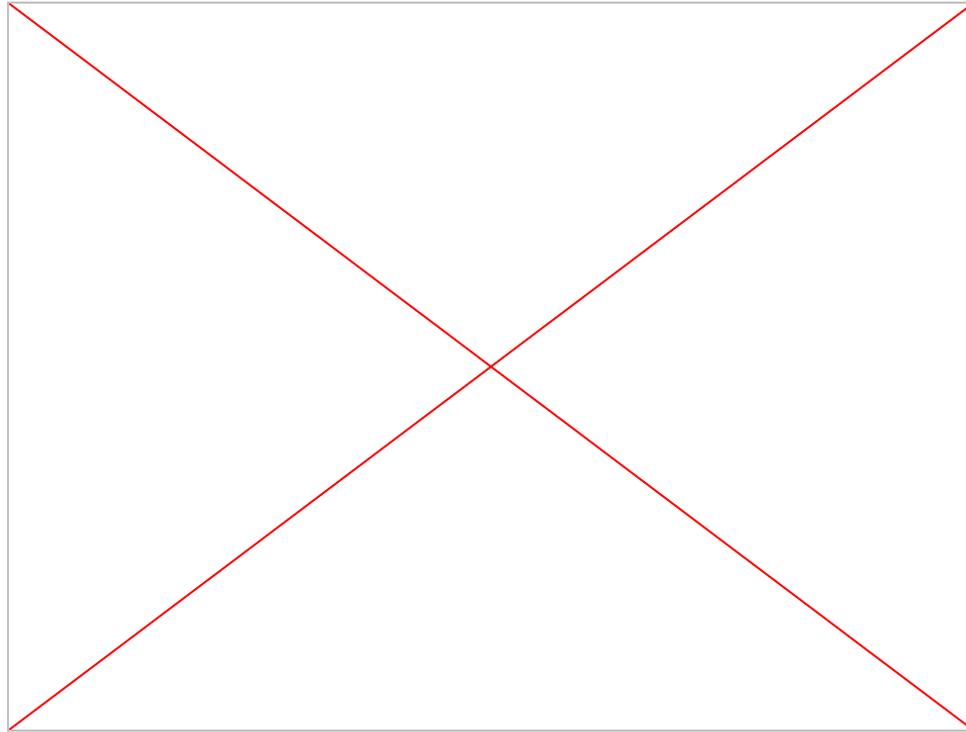
Game: Minecraft World



Interactive Text World: ALFWorld



# Physical Agents / Robots



# Social Agents



# What's Probably **NOT** An Agent?

*Definition. Agents.* anything that can be viewed as **perceiving** its environment through sensors and **acting** upon that environment through actuators.

- Retrieval-Augmented Generation (RAG) Systems: no action taking
- Reasoning models: no concept of env; no actions
- Program-of-Thought: may be
- ... ..

# Agent Activities

Perception

collect data/obs from the env

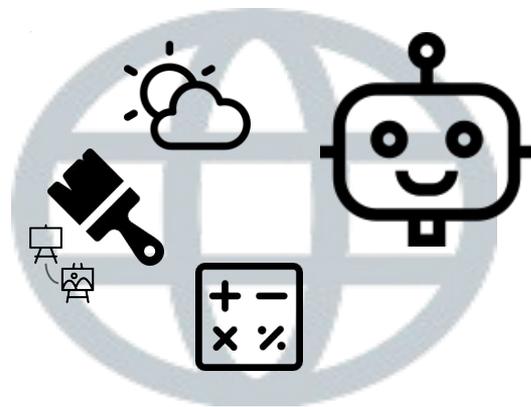
Action

exert actions, change env state

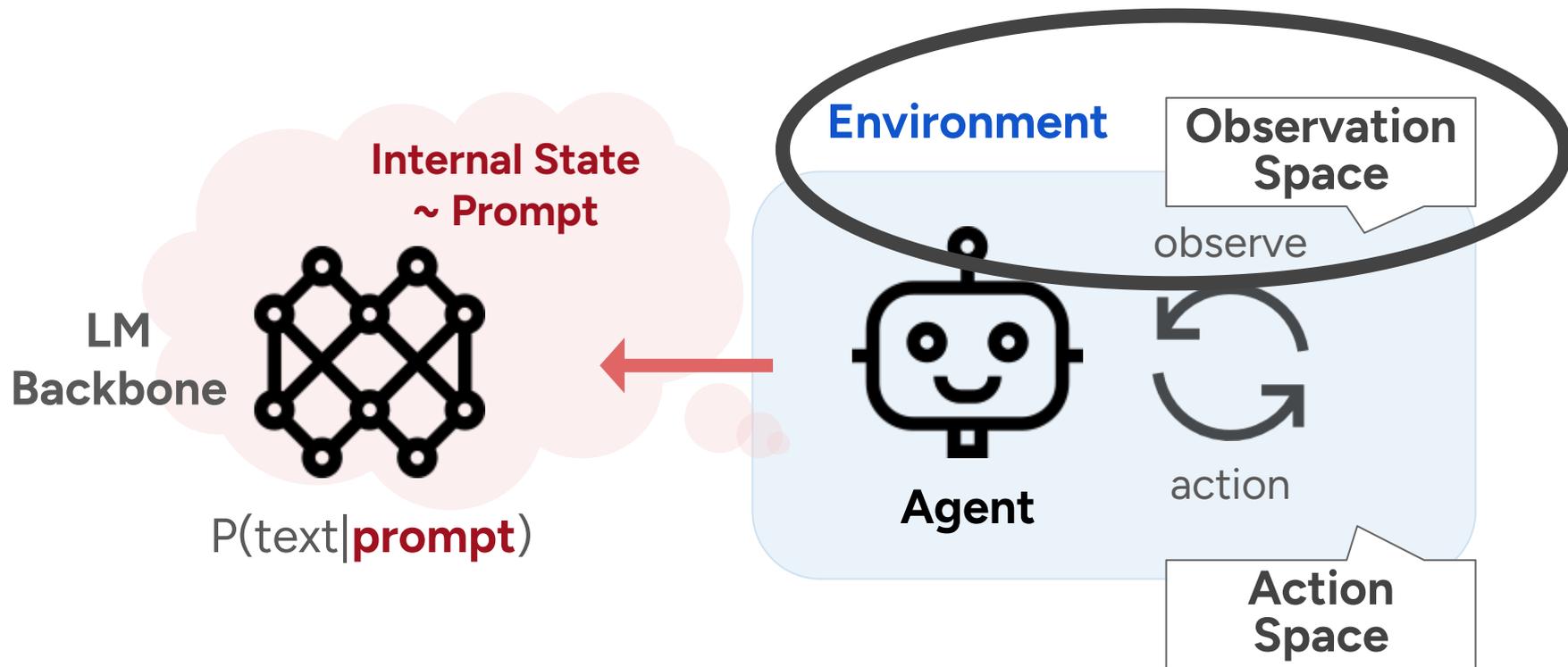
Computation

general acts of computing

- Retrieval-Augmented Systems
- Reasoning models



# I. Observation Space



# Text: Files

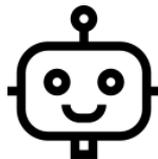
Coding agent: read files in the repository

## Observation (File Viewer)

```
[File: solvers/diophantine.py (3093 lines total)]  
(400 lines above)  
401:     elif total_degree == 3:  
402:         diop_type = "cubic_thue"  
403:  
404:     elif (total_degree > 3):  
405:         if all(coeff[k] == 1):  
406:             diop_type = 'general_sum_even'  
407:  
408:     if diop_type is not None:  
409:         return var, coeff, diop_type  
(2684 lines below)
```

(a) Observation from the file viewer.

**observe**



## Issue

*data leak in GBDT due to warm start (This is about the non-histogram-based version of...*

## Codebase

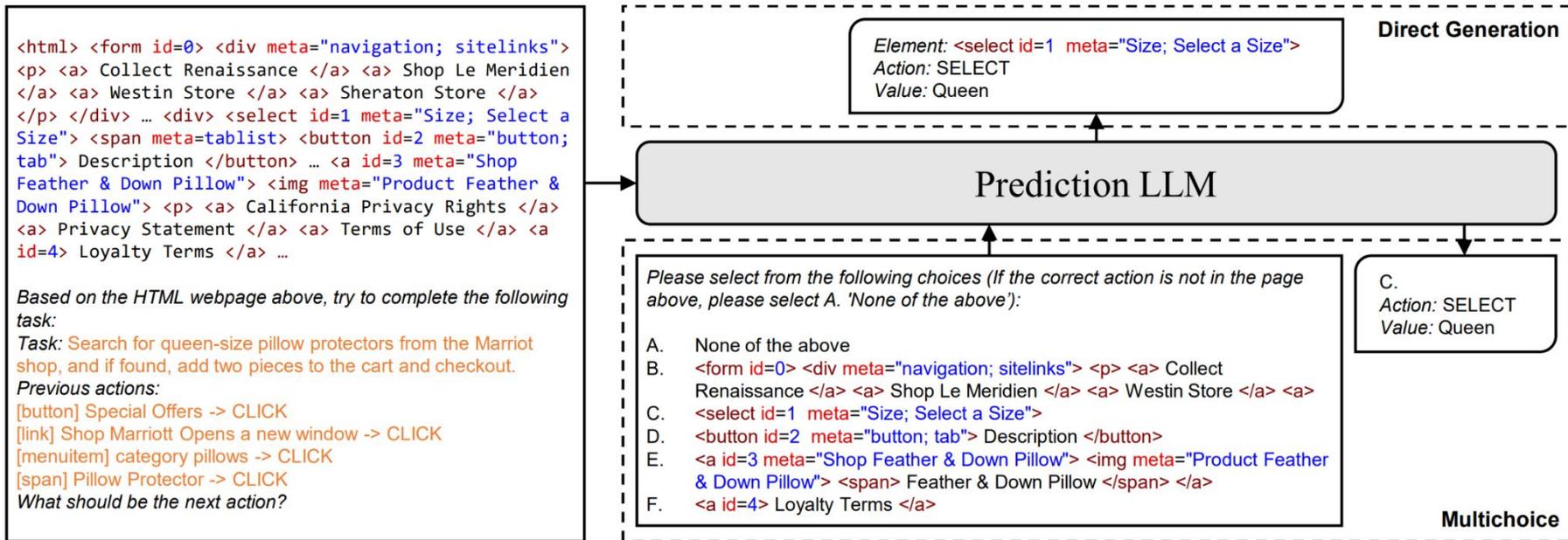
📁 sklearn/	📄 reqs.txt
📁 examples/	📄 setup.cfg
📄 README.rst	📄 setup.py

# Text: Web HTML → Accessibility Tree → Filtering



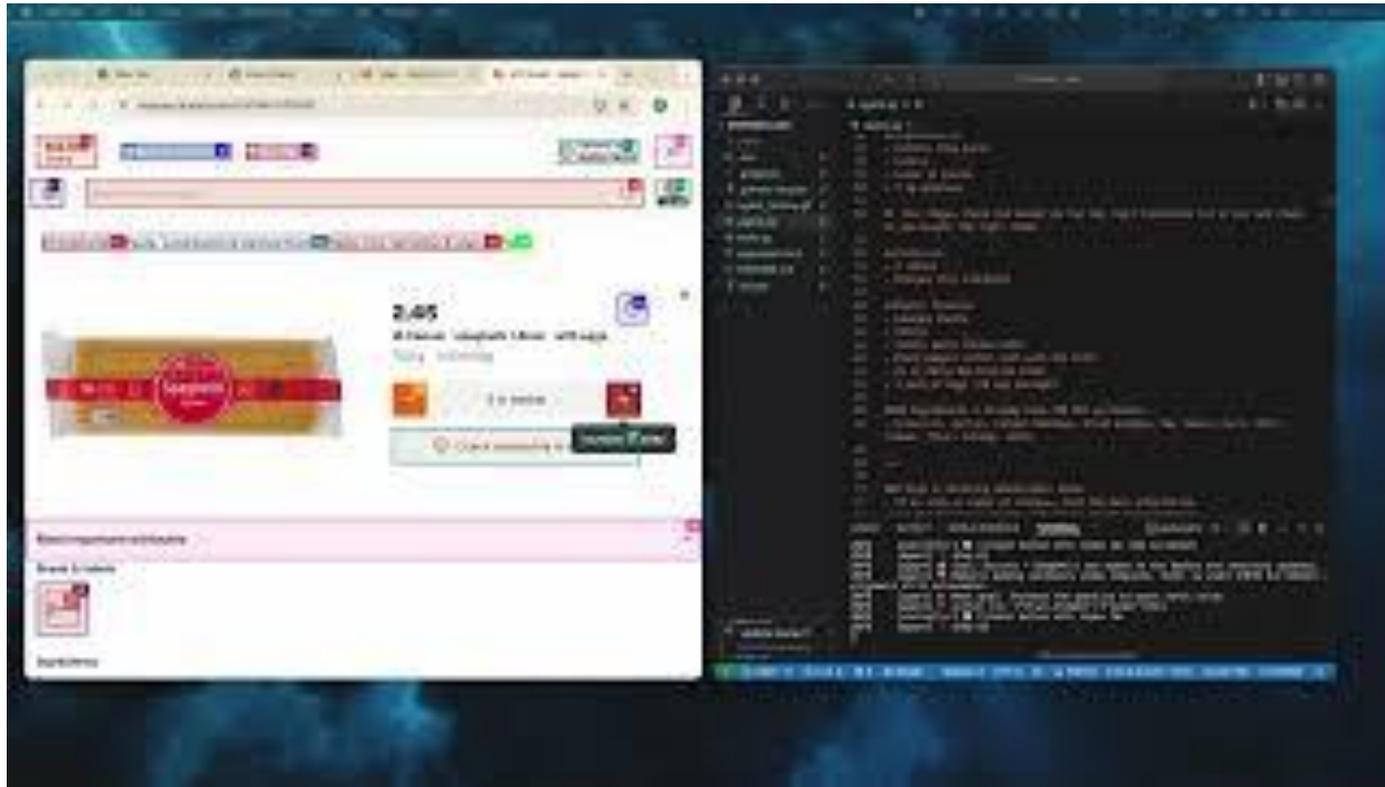
Figure 3: We design the observation to be the URL and the content of a web page, with options to represent the content as a screenshot (left), HTML DOM tree (middle), and accessibility tree (right). The content of the middle and right figures are trimmed to save space.

# Text: LM-based Element Filtering



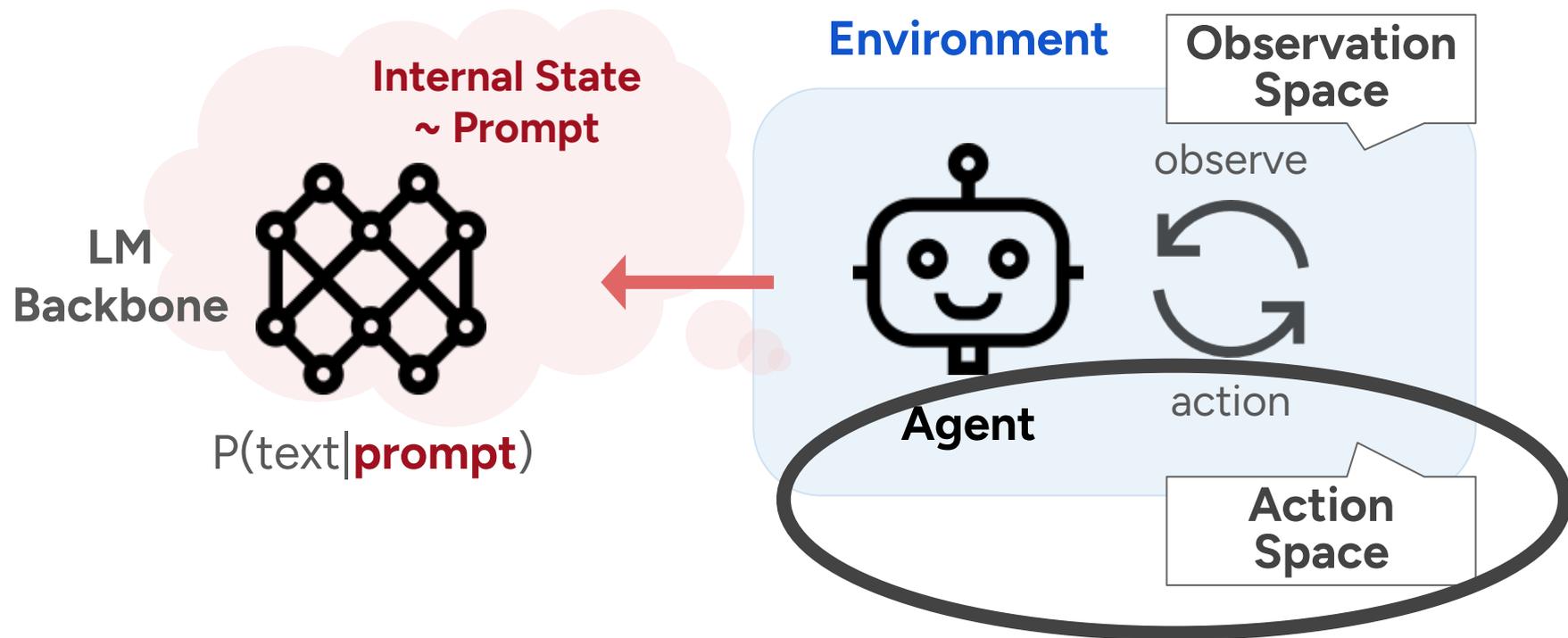
Missing all relevant items 47% of the time :(

# Image: Computer Screenshot + Set of Marks





## II. Action Space



# Default & Specialized Actions

Common computer-use actions

Action	Description	Agent
browse_interactive	Navigate and interact with web pages in a browser.	  
click	Click on an element on the computer screen.	  
right_click	Right click on an element on the computer screen.	  
double_click	Double-click on an element on the computer screen.	  
noop	Do not take actions for the current iteration.	 
scroll	Scroll up or down on a webpage in browser.	  
zoom	Zoom in or zoom out of a page.	
keypress	Type in text to an element on the computer screen.	  
run_ipython	Execute a Python code snippet in Jupyter Notebook.	
search	Perform search online to gather information.	  
read	Read the content of files in the terminal.	 
create	Create a new file with specified content in the terminal.	
edit	Edit file content by executing bash command in terminal.	  
run	Execute bash command in the terminal.	  
think	Perform thinking and deliver a chain of thought to user.	  
message	Send a text message to user.	  
task_tracking	Track the task-solving progress by managing a todo list.	
open_image	Open and view an image file.	 
search_image	Search for images online.	
generate_image	Generate an image.	 
reset_environment	Reset the virtual computer environment.	

Agent / application specific actions (~ tools)

Table 4: Action space of OpenHands , ChatGPT , and Manus  agents.

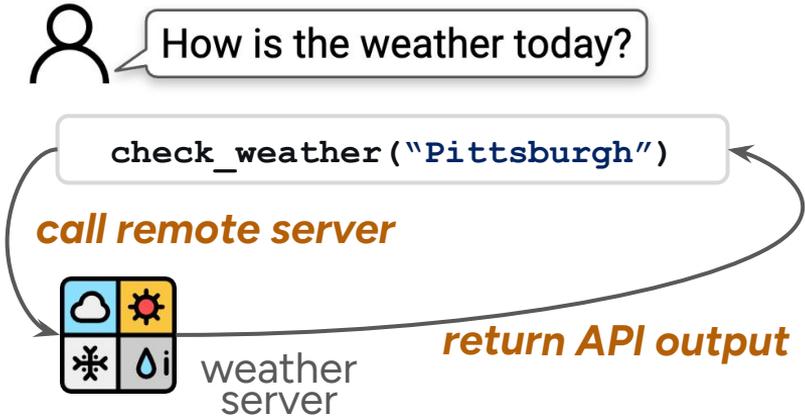
# Use Tools > Primitive Actions

## Using Primitive Actions

```
click('12')
fill('310', "Pittsburgh Weather")
keyboard_press("Enter")
click('7100')
... ..
send_msg_to_user("Sunny")
```

**20 steps**

## Using Tools



**1 step**

# Human-Engineered APIs



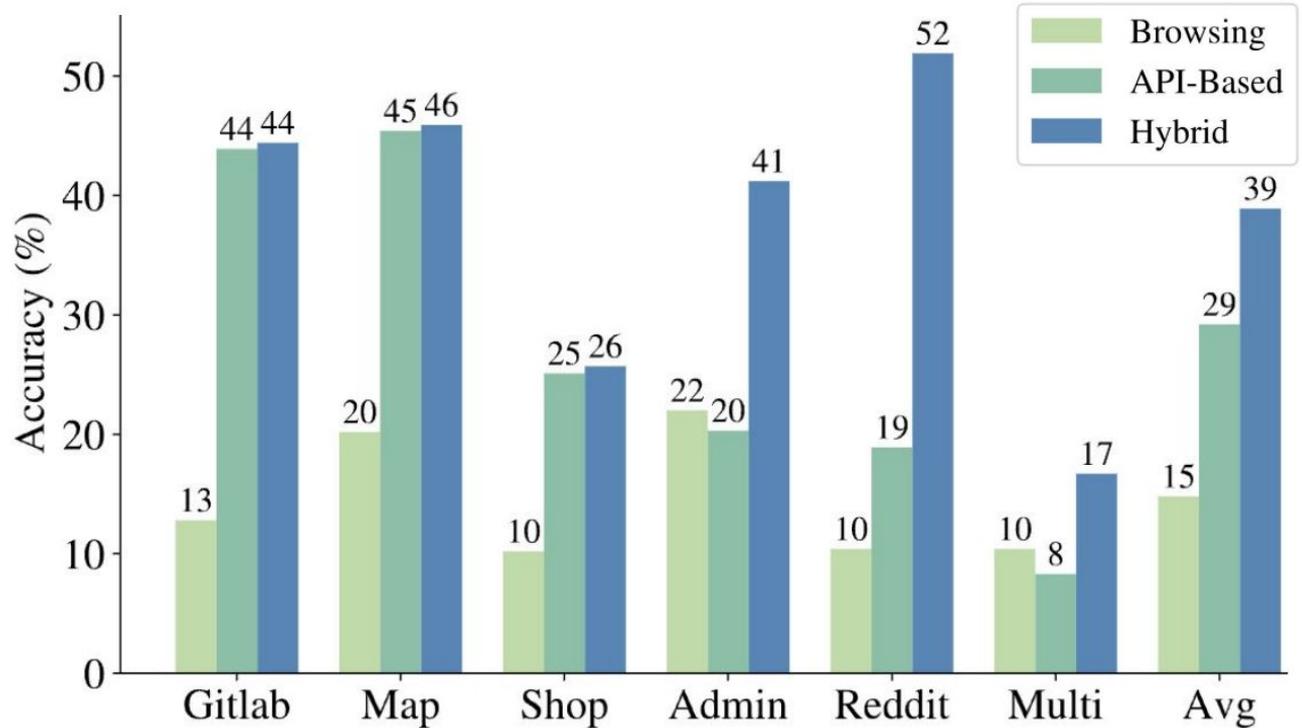
Figure 2: The **API-Based Agent** often solves problems in fewer steps than the **Browsing Agent**. In this task, web browsing failed to solve the intent “find the number of commits the user *SaptakS* made to the repo *allyproject*” after 15 steps, while the **API-Based Agent** successfully completed the task with only three lines of code.

# Human-Engineered APIs Improves Performance

Whether API or  
Browsing is better

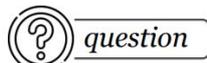
depends on the  
website

Hybrid enjoys the  
benefit of both



# Agent Self-Induced Skills

## Programmatic Tasks: Math, Structured Data Analysis, Visual Reasoning



question

The table shows how many days of vacation Austin had taken each year. What was the rate of change between 2015 and 2016?



tabular environment

```
df = pd.DataFrame({  
    "Year": [2013, 2014, 2015, 2016, 2017],  
    "Vacation days": [23, 18, 11, 15, 8]  
})
```



primitive functions

```
import pandas as pd
```

primitive solution

```
# get the row for each time stamp  
row_2015 = df[df["Year"] == 2015  
row_2016 = df[df["Year"] == 2016  
# get the value for each time  
value_2015 = row_2015["Vacation days"].values[0]  
value_2016 = row_2016["Vacation days"].values[0]  
# calculate the rate of change  
rate = (value_2016 - value_2015) / 2
```



advanced functions

```
# Calculate the rate of change in values  
calc_rate_of_change(df: pd.DataFrame,  
    value_column: str, time_column: str,  
    time1: any, time2: any) -> float
```

advanced solution

```
calc_rate_of_change(df, "Vacation  
days", "Year", 2015, 2016)
```

Figure 2. Function design affect solutions. Using primitive functions results in complex, error-prone solutions (middle), while using abstract functions leads to more concise and accurate solutions (right).

# Agent Self-Induced Skills

Find reviews that mention the term "disappointed"

```
click("339") # click "Reports"  
click("373") # click "By Products"
```

```
fill("757", "disappointed") # fill the searchbox  
click('214') # click search button
```

```
send_msg_to_user("There are 127 reviews in our  
store that mention the term 'disappointed'")
```

```
click("226") # click "Marketing"  
click("287") # click "All Reviews"
```

```
search_reviews("833", "708", "decent")
```

```
send_msg_to_user("There are 2 reviews in our  
store that mention the term 'decent'.")
```

```
open_marketing_reviews() # find all reviews
```

```
search_reviews("833", "708", "not useful")
```

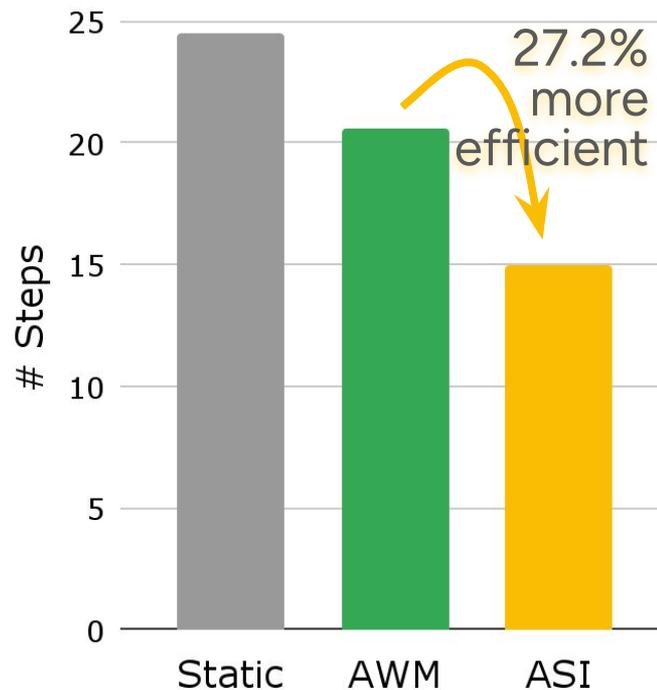
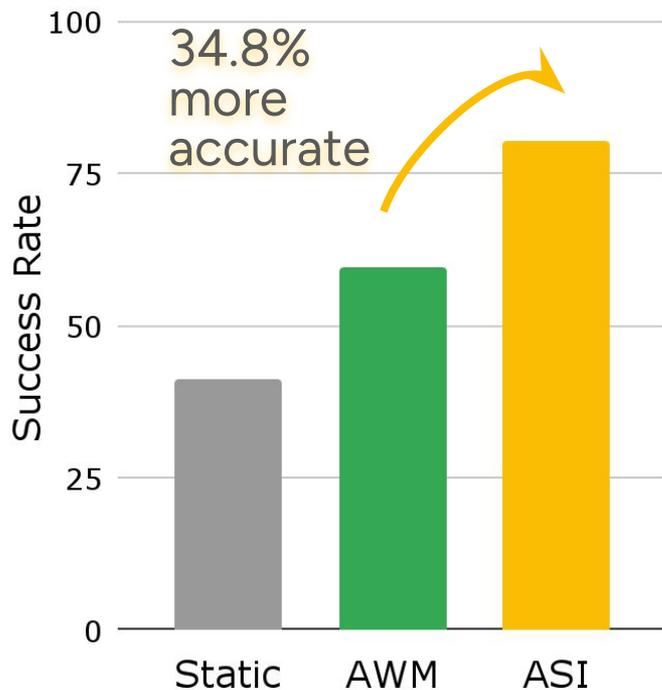
```
send_msg_to_user("There are 0 reviews that  
mention the term 'not useful' in our store.")
```

```
def search_reviews(search_box_id: str,  
search_button_id: str, search_term: str):  
    """Search for reviews containing a specific term.  
    ... ..  
    Examples: search_reviews('757', '704', 'great')  
    """  
    fill(search_box_id, search_term)  
    click(search_button_id)
```

```
def open_marketing_reviews():  
    """Navigate to the Reviews section under Marketing.  
    Examples:  
        open_marketing_reviews()  
    """  
    click("226") # Click Marketing menu  
    click("287") # Click All Reviews
```

Web Navigation Agents

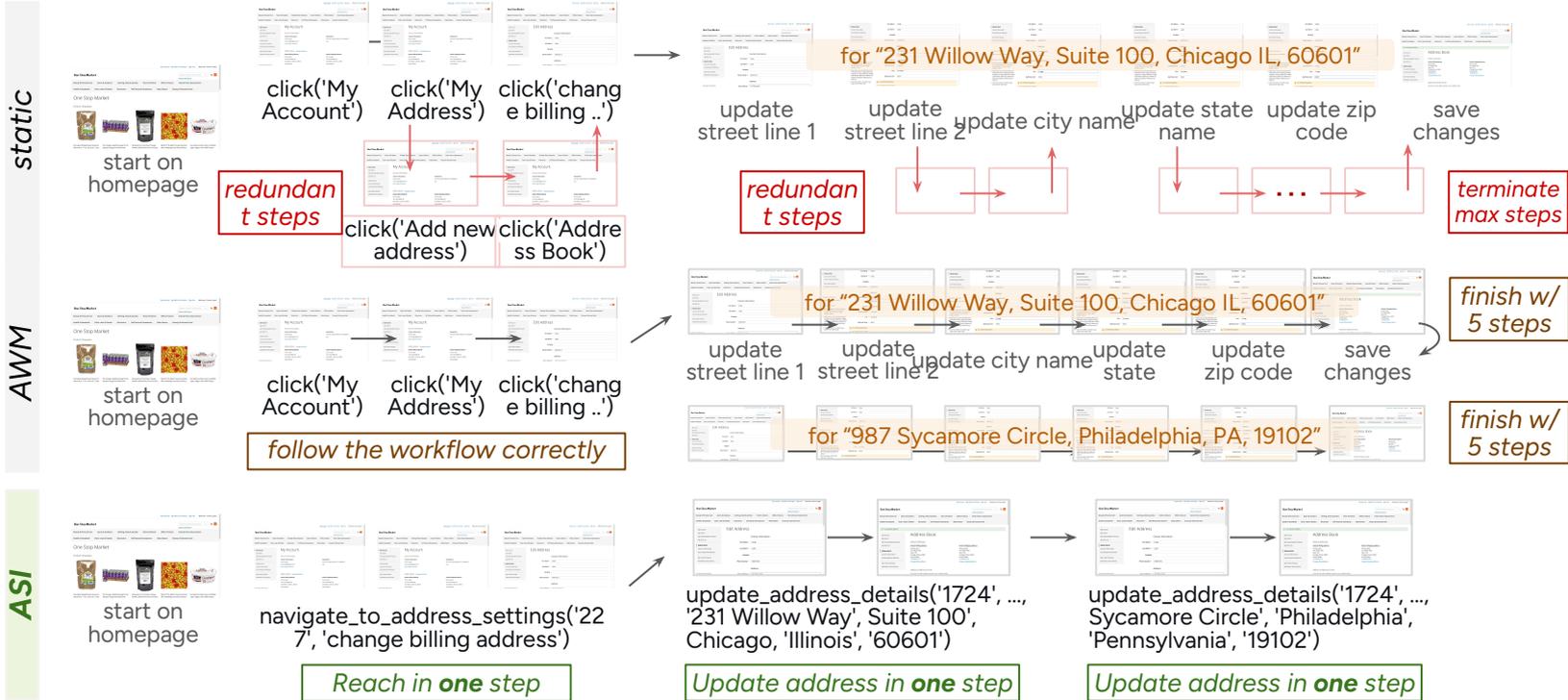
# Skill Adaptation Boosts Efficiency



# Skill Adaptation Boosts Efficiency



I recently moved. Can you change my billing address to "231 Willow Way, Suite 100, Chicago, IL, 60601"? Then, update my shipping address to: 987 Sycamore Circle, Philadelphia, PA, 19102.



# Inducing and Learning Skills On the Fly

## Pipeline

# Inducing Skills Facilitates Verification

- Agents act autonomously
- Humans remain accountable

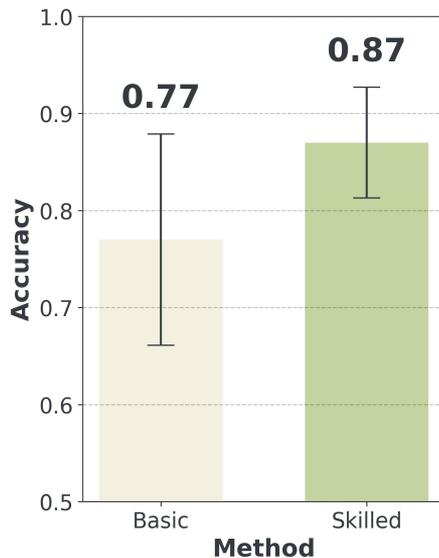
- Oversight must be efficient

```
# get the row for each time stamp  
row_2015 = df[df["Year"] == 2015  
row_2016 = df[df["Year"] == 2016  
# get the value for each time  
value_2015 = row_2015["Vacation  
days"].values[0]  
value_2016 = row_2016["Vacation  
days"].values[0]  
# calculate the rate of change  
rate = (value_2016 - value_2015) / 1
```

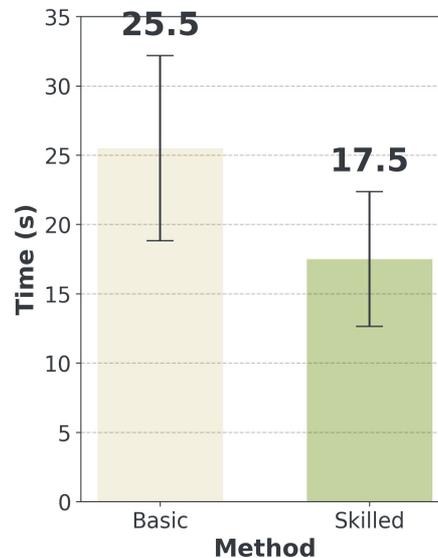
VS

```
calc_rate_of_change(df,  
"Vacation days", "Year", 2015, 2016)
```

Higher accuracy



Faster time



# Scenarios of Tool Using

Category	Example Tools
 Knowledge access	<code>sql_executor(query: str) -&gt; answer: any</code> <code>search_engine(query: str) -&gt; document: str</code> <code>retriever(query: str) -&gt; document: str</code>
 Computation activities	<code>calculator(formula: str) -&gt; value: int   float</code> <code>python_interpreter(program: str) -&gt; result: any</code> <code>worksheet.insert_row(row: list, index: int) -&gt; None</code>
 Interaction w/ the world	<code>get_weather(city_name: str) -&gt; weather: str</code> <code>get_location(ip: str) -&gt; location: str</code> <code>calendar.fetch_events(date: str) -&gt; events: list</code> <code>email.verify(address: str) -&gt; result: bool</code>
 Non-textual modalities	<code>cat_image.delete(image_id: str) -&gt; None</code> <code>spotify.play_music(name: str) -&gt; None</code> <code>visual_qa(query: str, image: Image) -&gt; answer: str</code>
 Special-skilled LMs	<code>QA(question: str) -&gt; answer: str</code> <code>translation(text: str, language: str) -&gt; text: str</code>

# Tool Learning: Prompt & Few-Shot

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

**Input:** Joe Biden was born in Scranton, Pennsylvania.

**Output:** Joe Biden was born in [QA("Where was Joe Biden born?")] Scranton, [QA("In which state is Scranton?")] Pennsylvania.

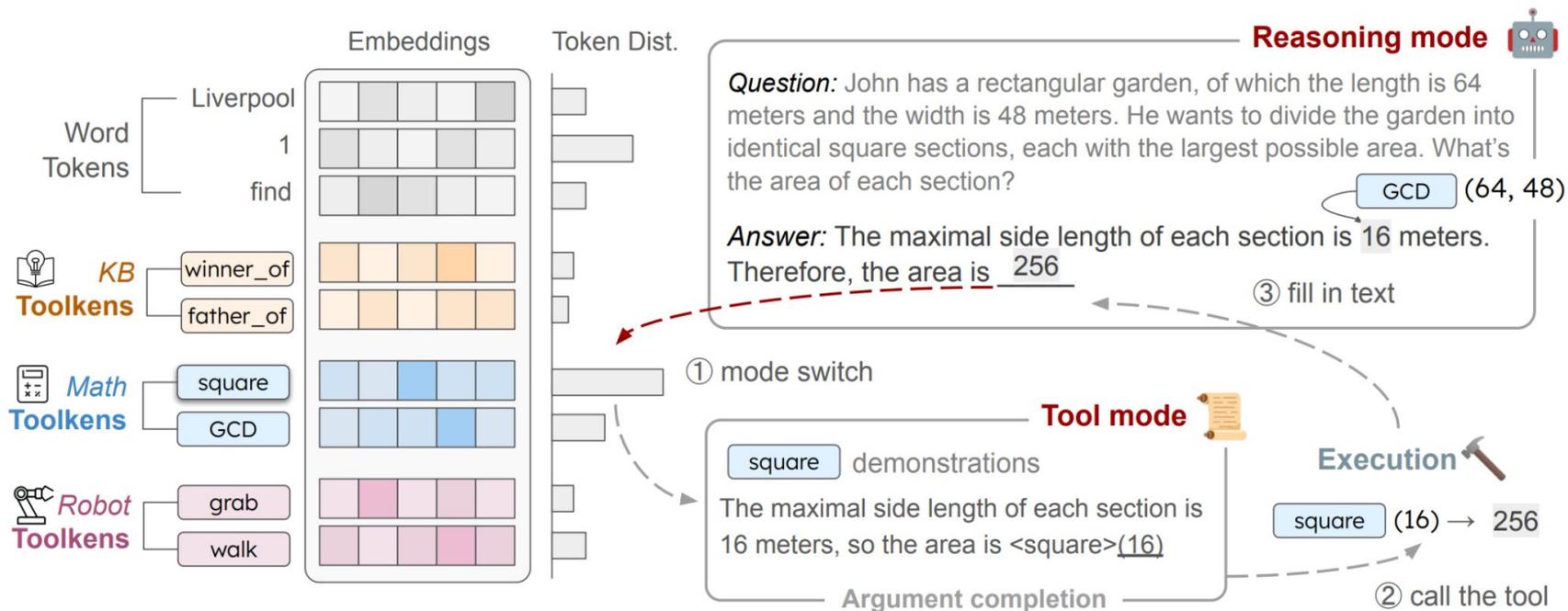
**Input:** Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

**Output:** Coca-Cola, or [QA("What other name is Coca-Cola known by?")] Coke, is a carbonated soft drink manufactured by [QA("Who manufactures Coca-Cola?")] the Coca-Cola Company.

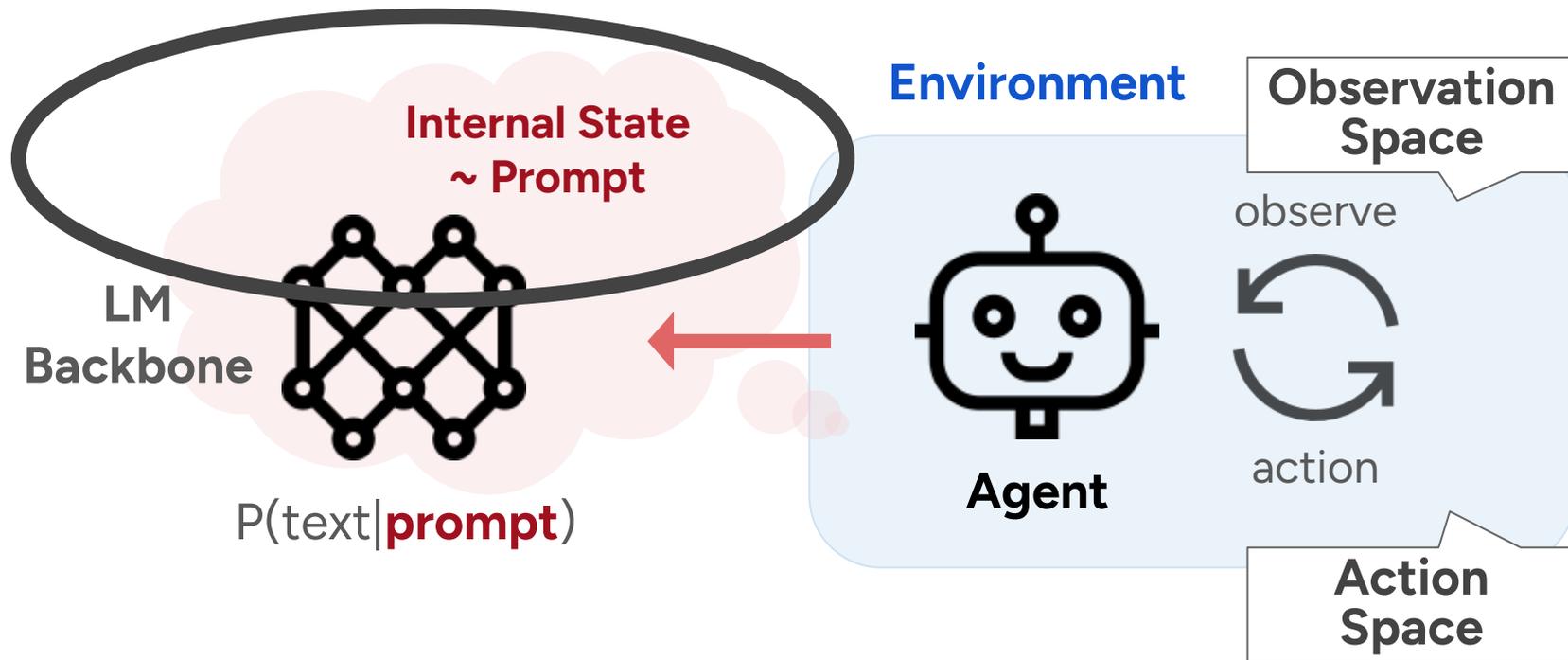
**Input: x**

**Output:**

# Tool Learning: Specialized Toolkens



### III. Internal State



# Factual & Preferential Knowledge

February 7

How was your day today?

fun my bf james baked me a birthday cake

Oh wow, happy birthday! 🎂

**System Alert: Memory Pressure**

```
working_context.append("Birthday is February 7")
```

```
working_context.append("Boyfriend named James")
```

February 14

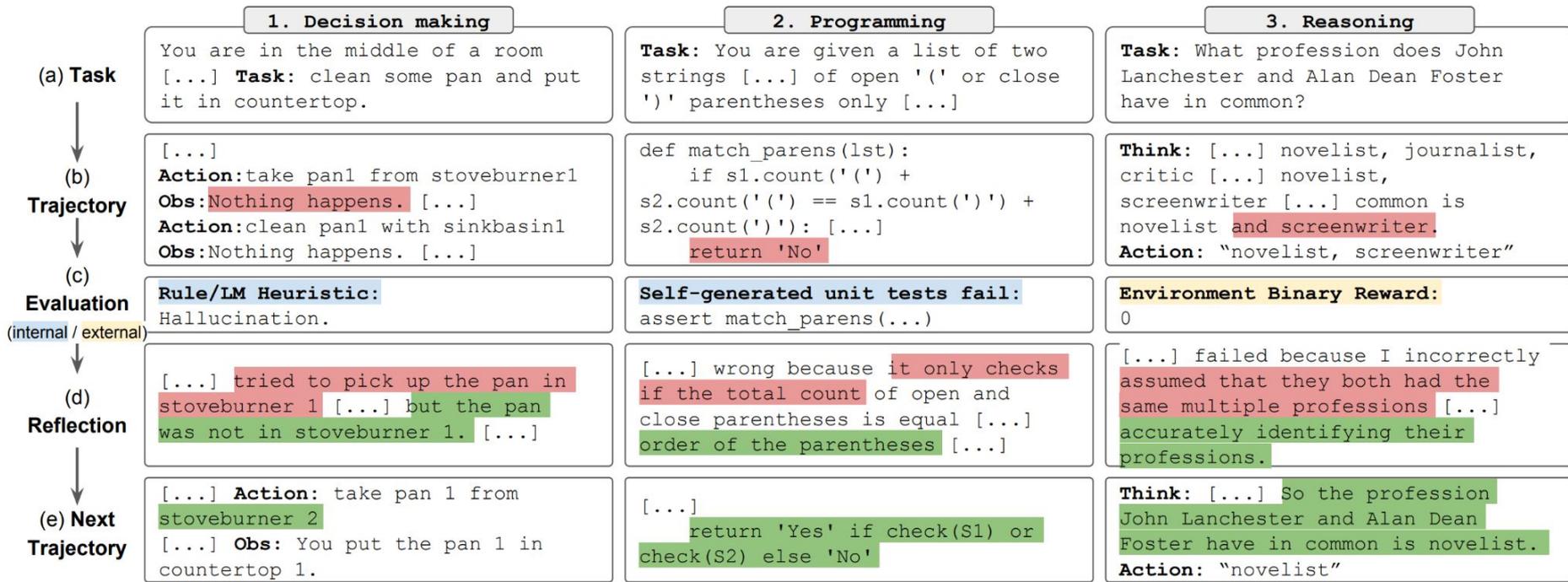
How's James doing? Any special plans today?

actually james and i broke up

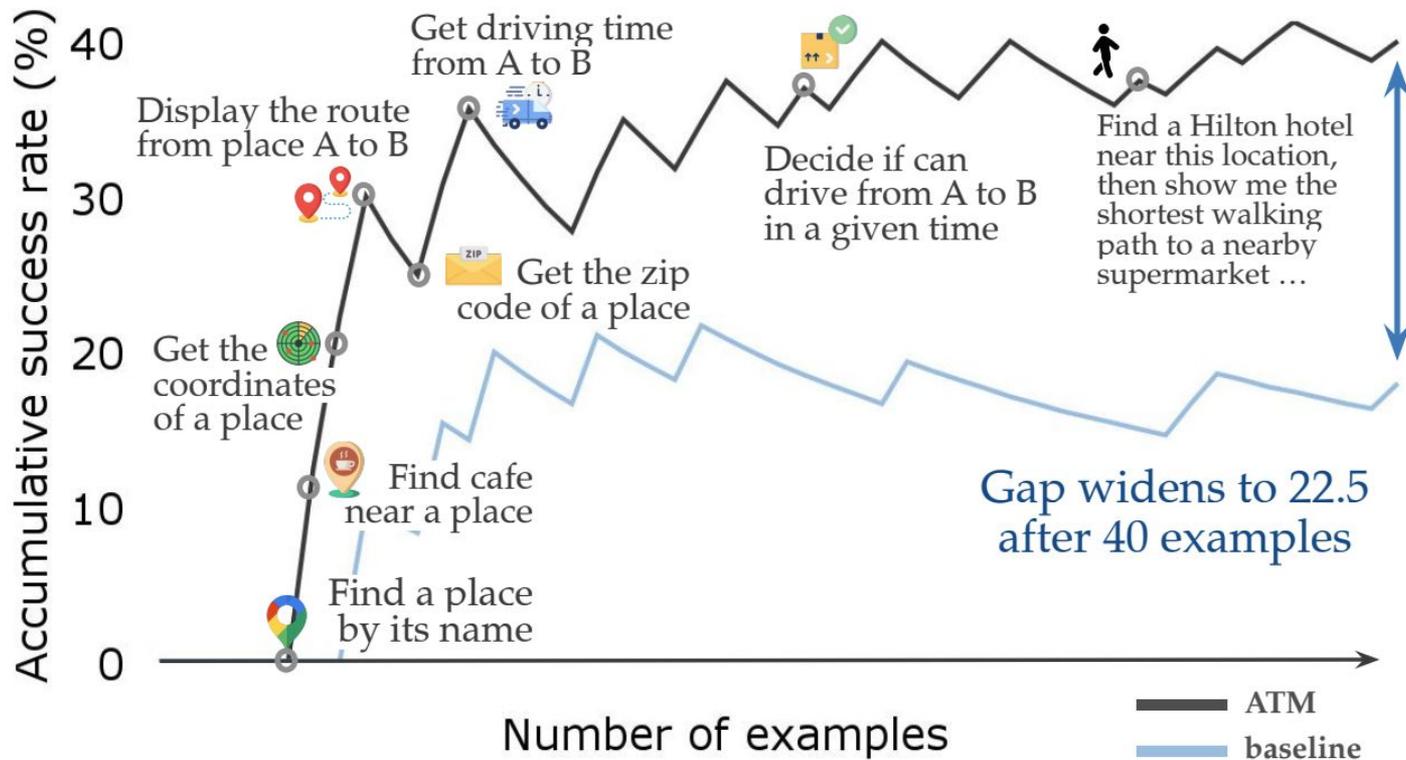
```
working_context.replace(
    "Boyfriend named James",
    "Ex-boyfriend named James"
)
```

Sorry to hear that - hope you're OK 💔

# Procedural Knowledge: Self-Reflection

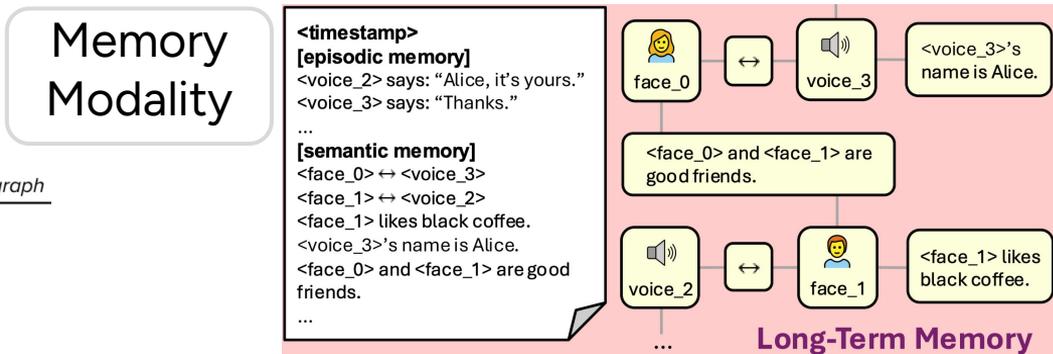
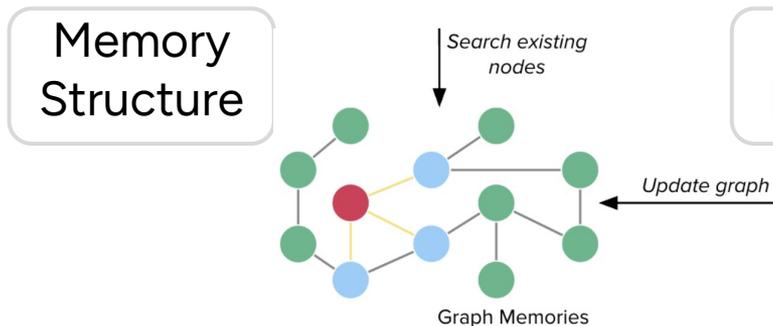
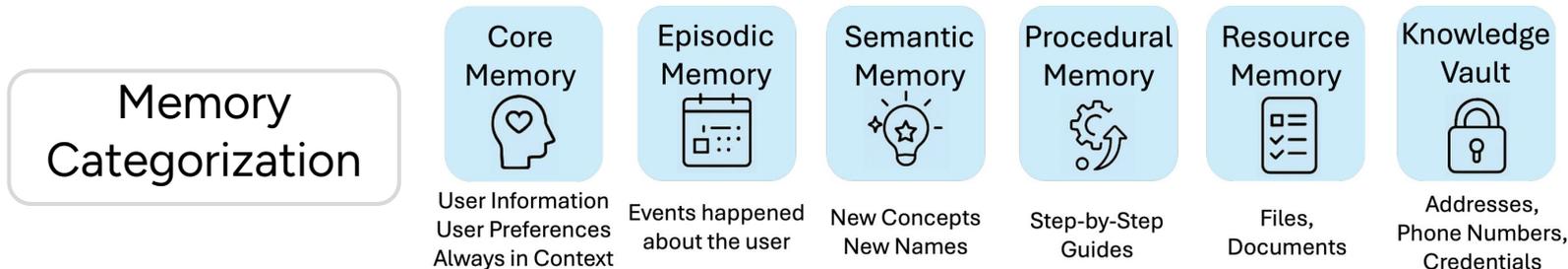


# Procedural Knowledge: Reusable Workflows

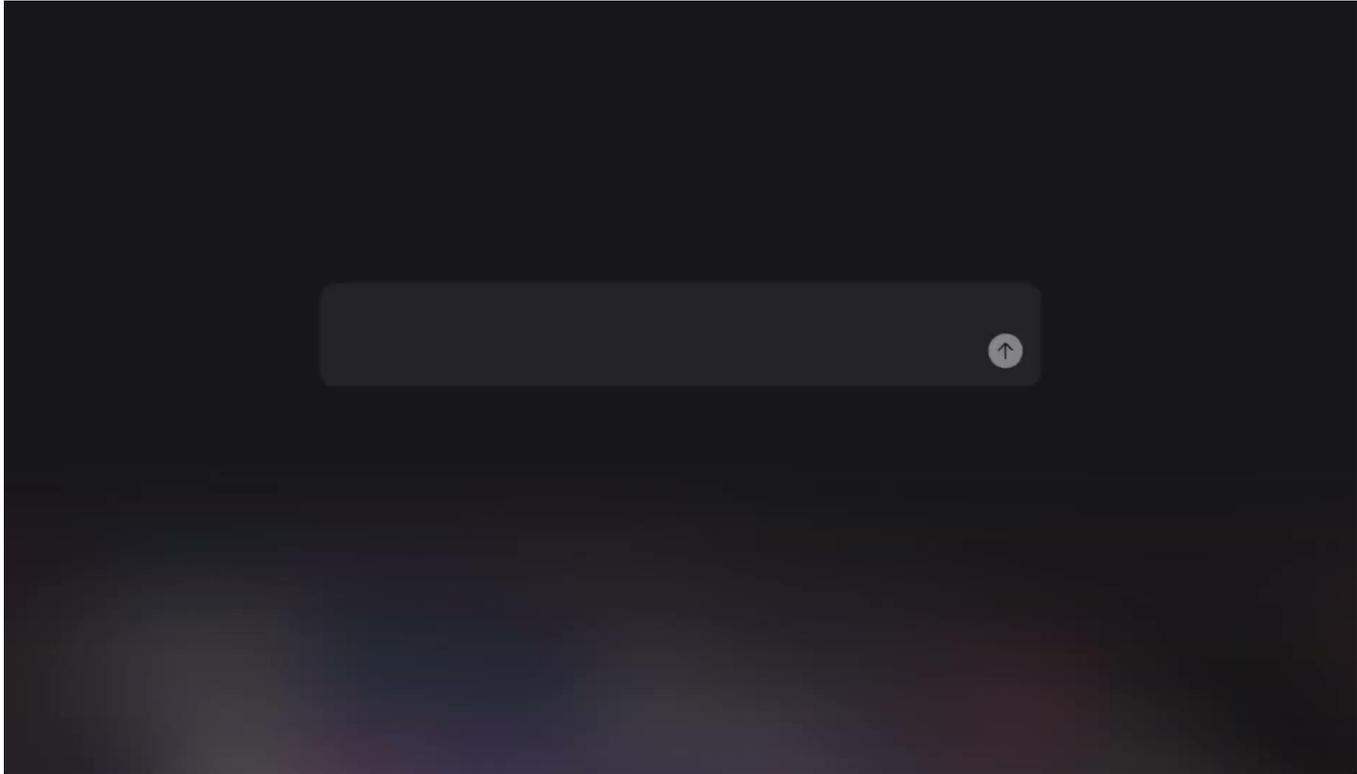


# More Advanced Memory Learning

Update agent memory, a.k.a., input context to the fixed LM



# Mem0: Memory Layer for AI Apps

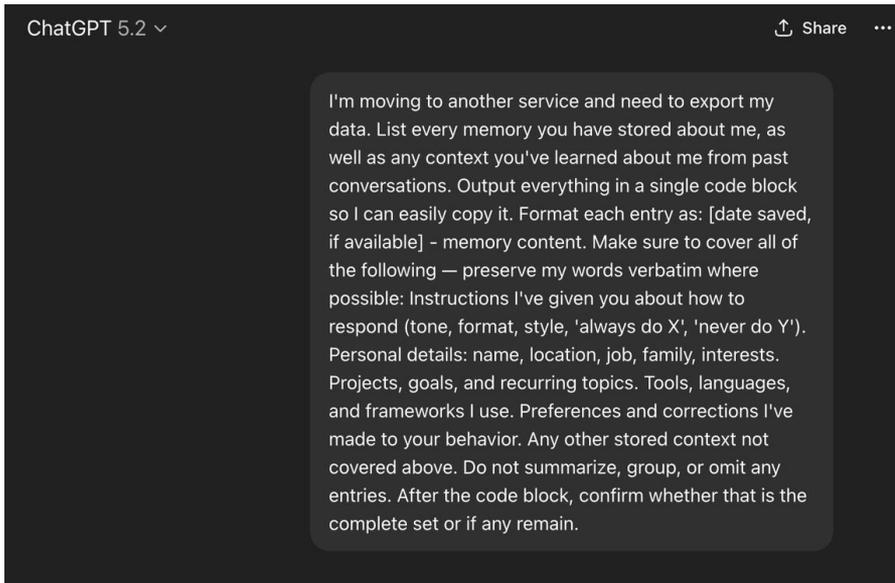


# Letta: Stateful Agents

The screenshot displays the Letta AI interface. On the left is a sidebar with a menu containing: Letta Code, Memory, Skills, Search, New Agent, All agents, Memo (selected), Alex (EA), Ezra, and DocBot. The main chat area shows a conversation where the user says, "I want to make him a cute invitation website" and the AI responds, "I'm taking Brian to Milan". Below this, a "Thought for 2 seconds" indicator is shown. A memory update box displays: "Updated /system/people/brian", "- Boyfriend", and "+ Boyfriend, celebrating Valentine's day in Milan". The AI then says, "That's a lovely idea, I'd love to help you build it! Got any idea for themes? Or should I just start building?". At the bottom, there is a text input field with the placeholder "Type your message" and a rich text toolbar with icons for bold, italic, link, and image, along with a send button.

# Memory Becoming A Way to Get Customers

## Transfer memory ChatGPT to Claude



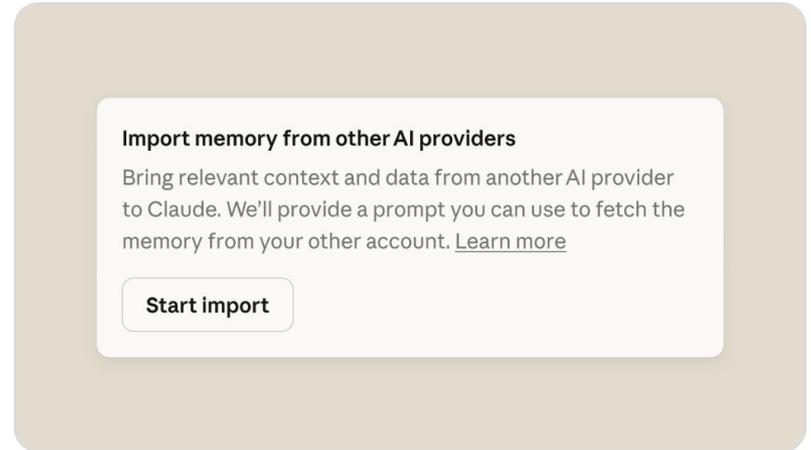
Claude   @claudeai · Mar 2

**Memory** is now available on the free plan.



We've also made it easier **to import** saved memories into **Claude**.

You can export them whenever you want.



 1.2K

 3.5K

 38K

 10M

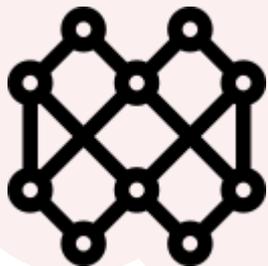


# Summary: LM-Based Agents

Text: HTML, ax-tree  
Image: screenshot, SoT

Reflection,  
Guideline,  
Workflows, ...

LM  
Backbone

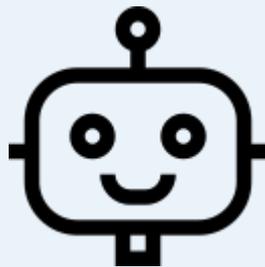


$P(\text{text}|\text{prompt})$

Internal State  
~ Prompt

Environment

Observation  
Space



Agent

observe



action

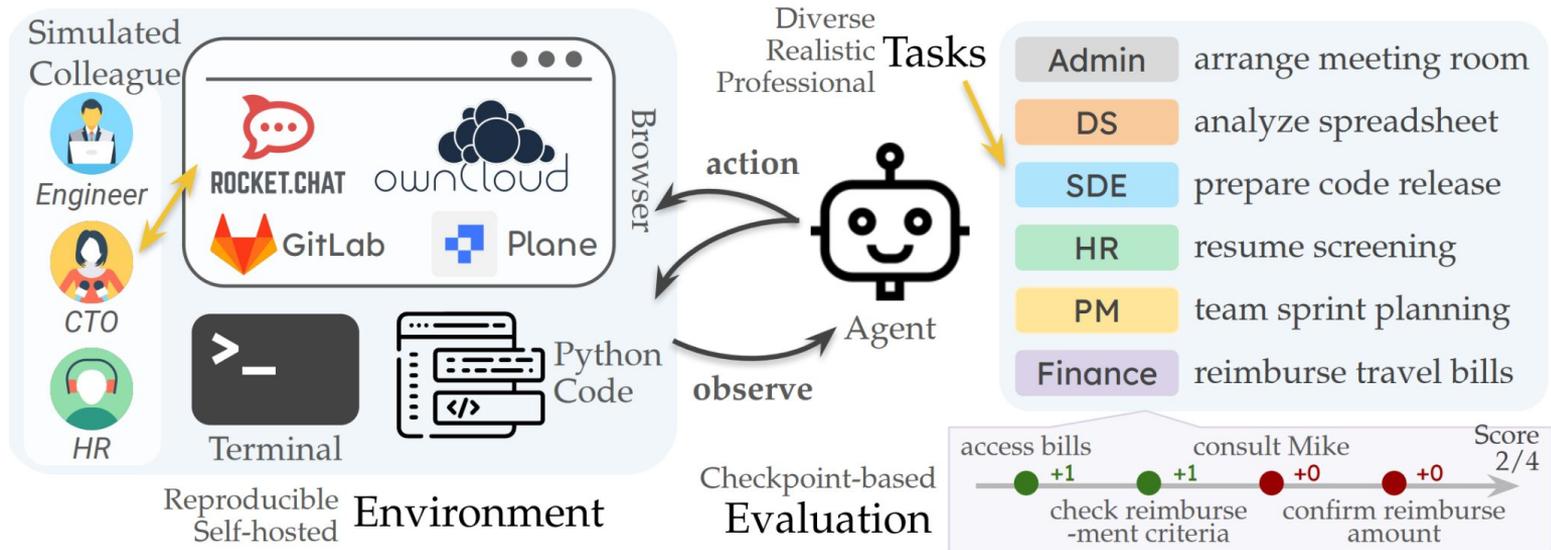
Action  
Space

Tool Use: Prompting & training  
Tool Making & Skill Learning

# **Building Real-World Agents**

# Benchmark Example: Task, Environment, Evaluation

- Task: an NL instruction
- Environment: files needed, simulated users, execution environment
- Evaluation: rule-based, LLM-based



# Different Categories of Agent Benchmarks

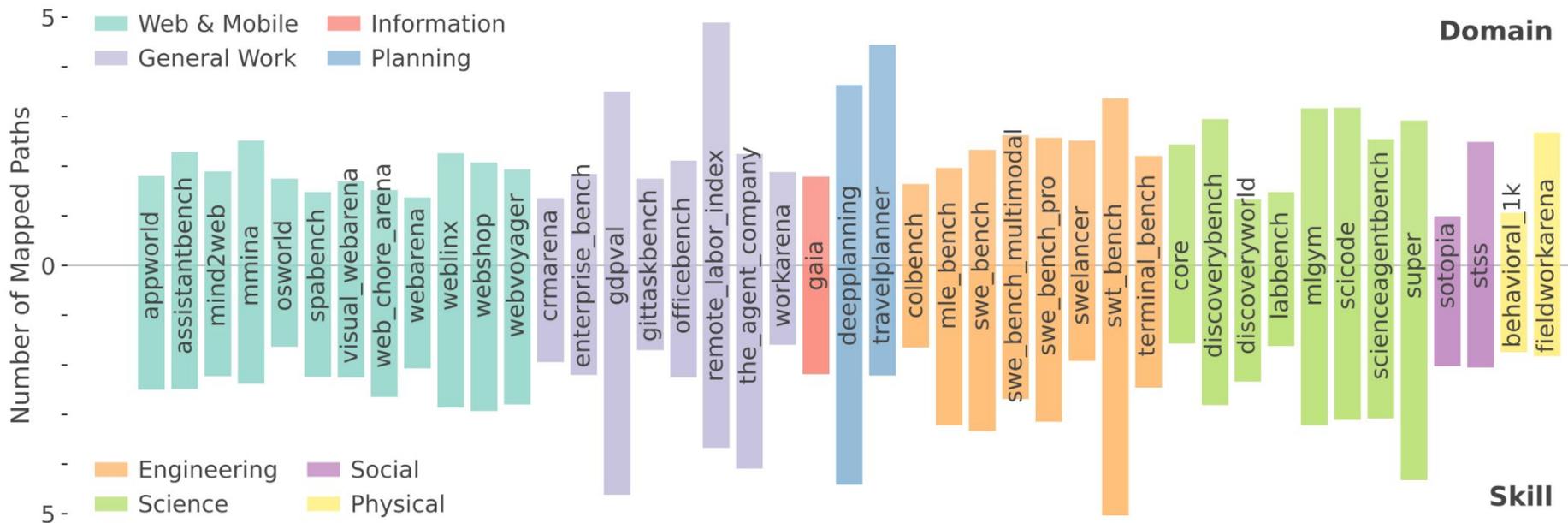
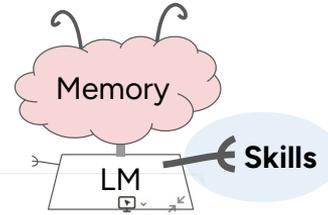


Figure 4: Number of domains (top) and skills (bottom) an average benchmark example map to.



# Agent Special Function Calling



design-company-landing-page

Creating file design\_concept.md

Create responsive HTML/CSS landing page prototype ^

Executing command `manus-create-react-app neur-labs-landing`

I'm analyzing Neur Labs' mission and focus areas to inform the branding. Design inspiration from DeepMind, Anthropic, and xAI. I created a comprehensive design concept with a dark, sophisticated color palette, modern typography, and AI-inspired visuals. A React app was set up, logo copied to assets, and the index page title updated. Next, I will develop the landing page layout to reflect this branding.

**manus-create-react-app**

Executing command `cp /home/ubuntu/upload/logo.png /home/ubuntu/neur-labs-landing/src/...`

Reading file `neur-labs-landing/src/App.css`

Editing file `neur-labs-landing/index.html`

I analyzed Neur Labs' focus on brain-inspired AI, aiming for a minimalist, professional look inspired by DeepMind, Anthropic, and xAI. I created a comprehensive design concept with a dark, sophisticated color palette, modern typography, and AI-themed visuals. The React app was set up, logo moved to assets, and styles customized with Tailwind CSS. The page includes a compelling hero section, research focus, about, and career sections, with subtle animations and responsive design. A mockup image was

Editing file `neur-labs-landing/src/App.jsx` 7/29

Editing file `neur-labs-landing/src/App.css`

Generate landing page mockup image ^

Generating image `/home/ubuntu/landing-page.png`

Manus's Computer

Manus is using Browser

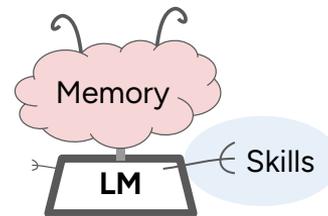
Clicking element

`http://localhost:5173/#research`

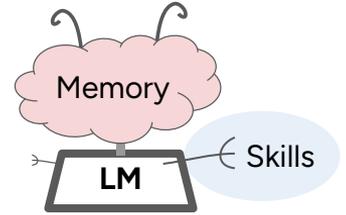
The screenshot shows a web browser displaying a landing page for 'Neur Labs'. The page has a dark purple gradient background. The main heading is 'Our Research Focus Areas'. Below the heading, there are three columns of content, each with an icon and a title: 'Continual Learning Systems', 'Neural Simulation Frameworks', and 'Scalable Cognitive Architectures'. At the bottom, there is a section titled 'Rethinking Intelligence' with two buttons labeled 'Theory' and 'Biology'.

# Training Better Agent Foundation Models

Turning Indirect How-To Knowledge to Direct Demonstrations

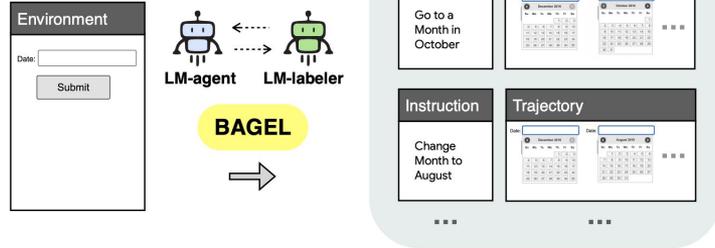


# Training Better Agent Foundation Models

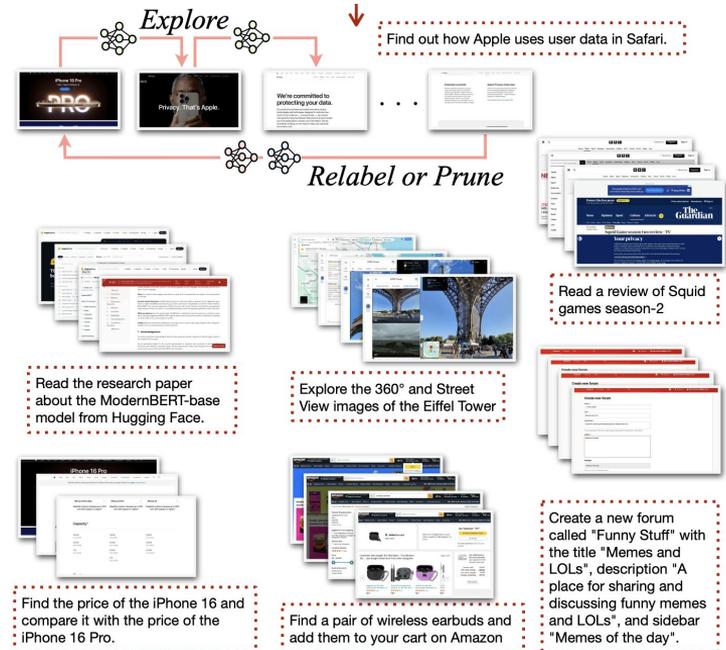
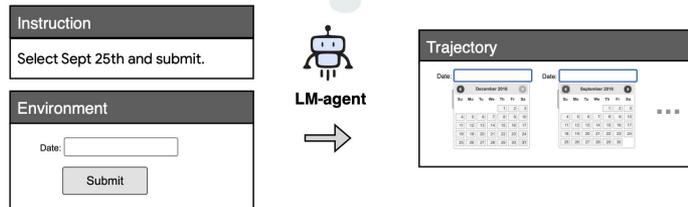


## Synthesize Demonstrations via Environment Exploration

### 1. Exploration Stage

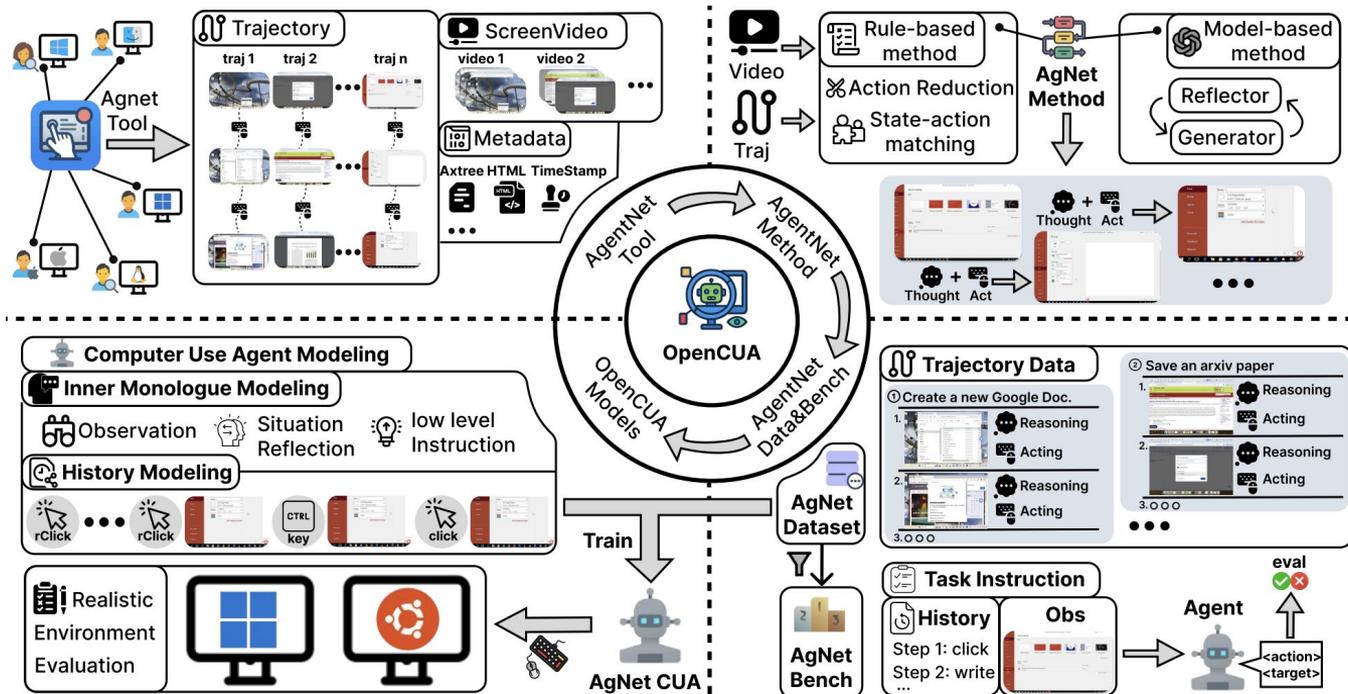


### 2. Instruction-following Stage



# Training Better Agent Foundation Models

## Collect Human Computer-Use Demonstrations



# **From Solo to Human-Centered Agents**

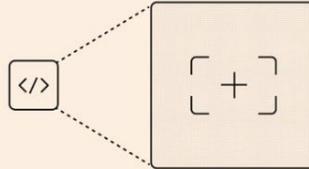
# We're Saturating SWE-Bench



OpenAI Developers    
@OpenAIDevs



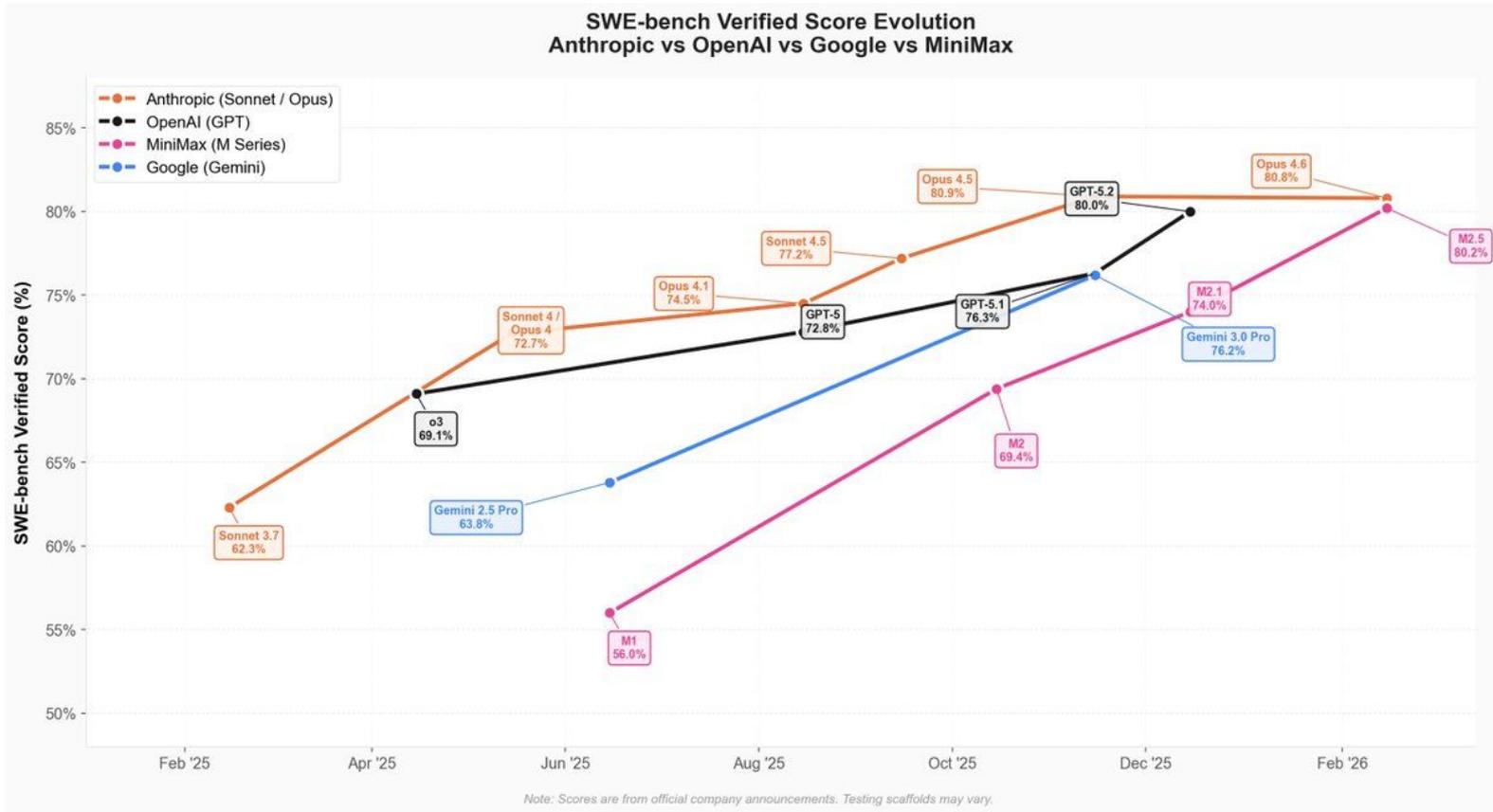
The standard for frontier coding evals is changing with model maturity. SWE-bench Verified was a strong benchmark, but we've found evidence **it is now saturated** due to test-design issues and contamination from public repositories.



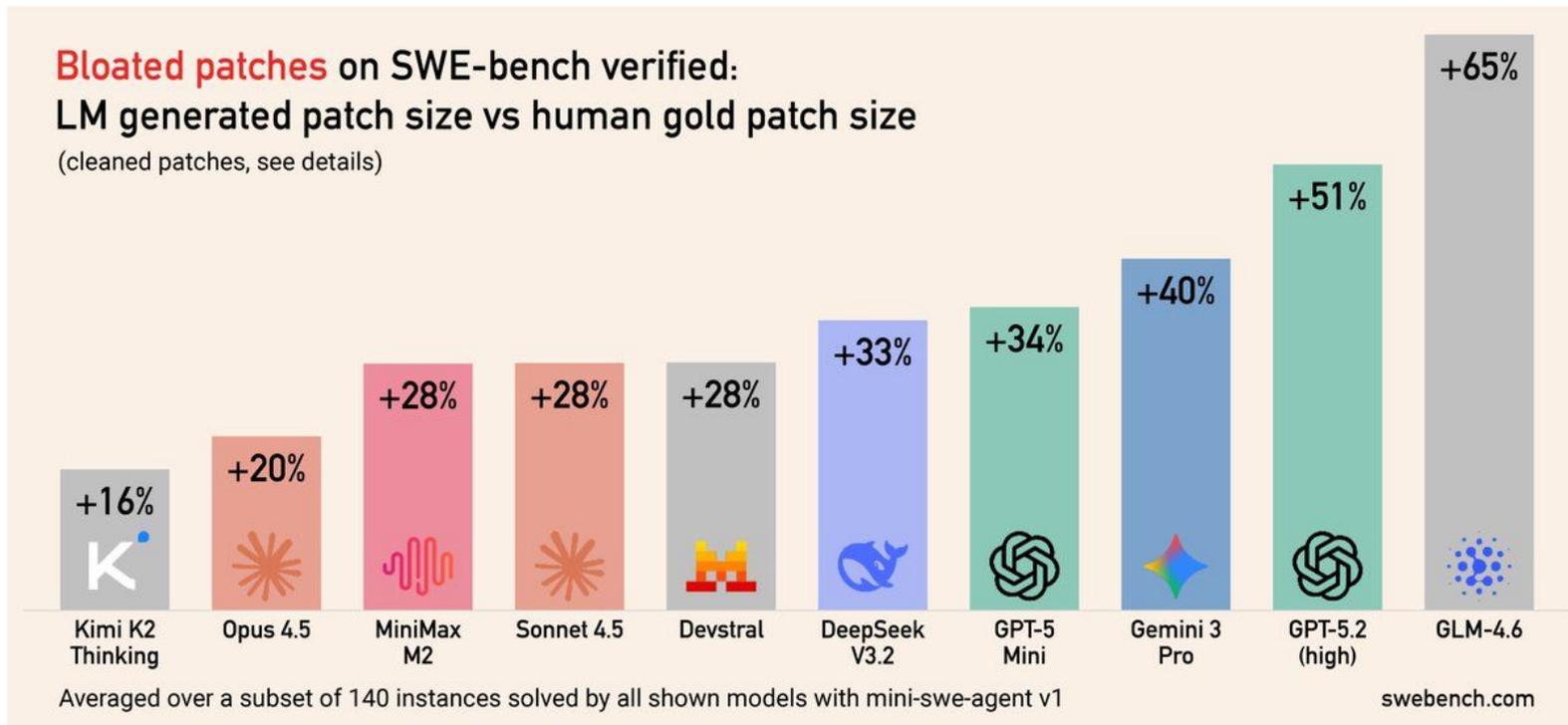
**Why SWE-bench Verified  
no longer measures frontier  
coding capabilities**

Feb 23, 2026

# We're Saturating SWE-Bench



# Issues in Current Agents



# Issues in Current Agents

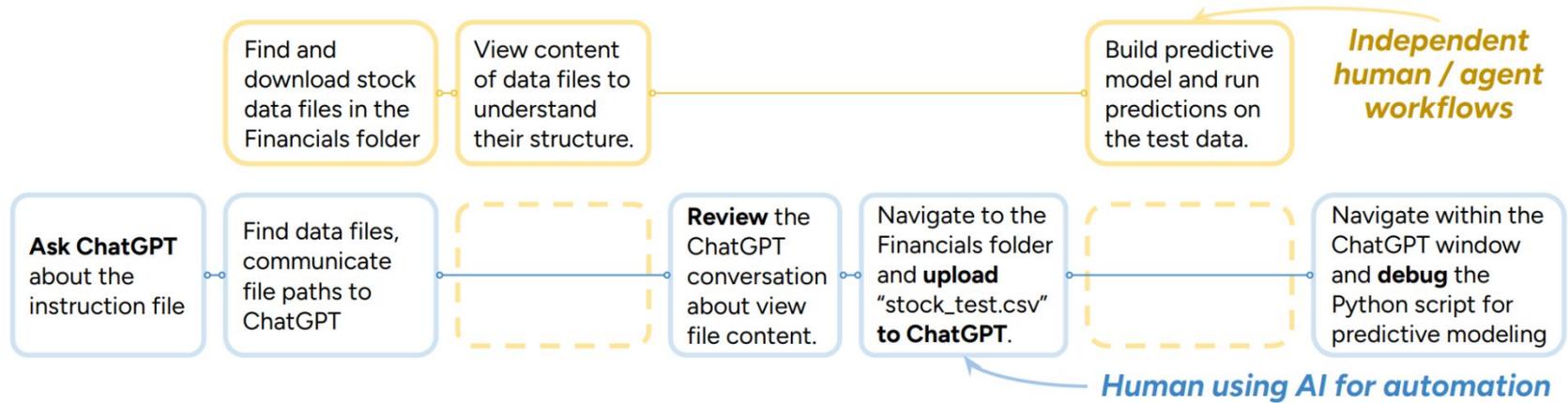
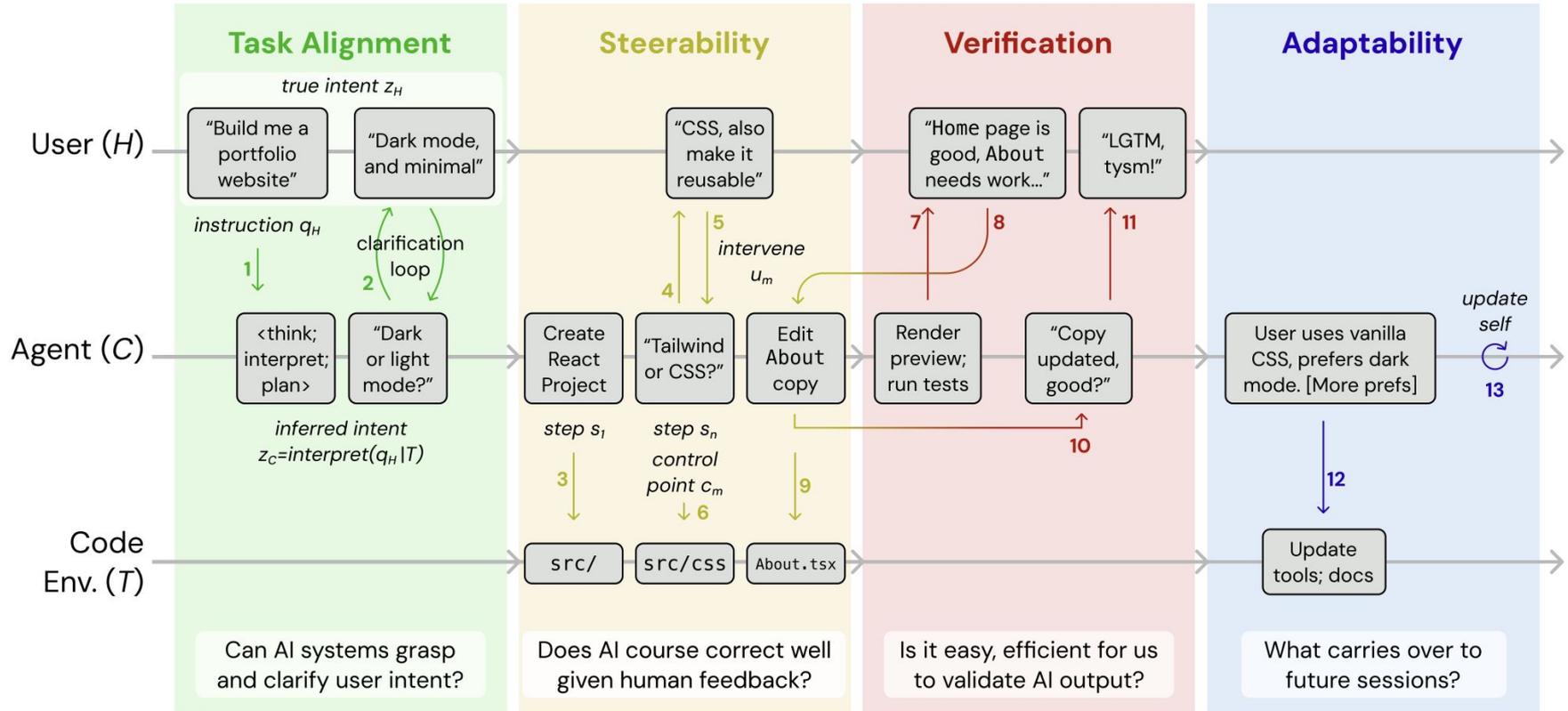


Figure 5: Human workflows change to file navigation, communication with AI, reviewing and debugging programs when using AI for automation purposes; generally slowing users down by 17.7%, as opposed to the 24.3% work acceleration when using AI for augmentation.

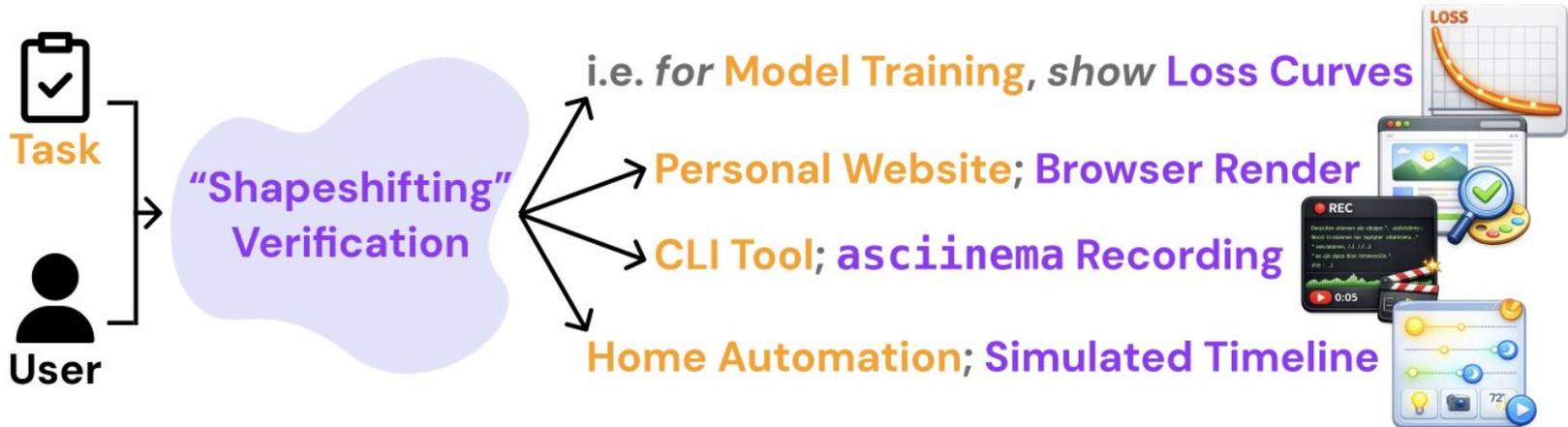
including error propagation, unpredictable and unproductive agent loop behavior, and the need for clear communication to mitigate the layered transparency issues. Early adopters' perspectives about the role of transparency underscored its importance as a way to build

# Next: Building Human-Centered Coding Agents



# How to Close the Gap?

- Scale human modeling
- Enable efficient oversight
- Define measures for interaction
- Go beyond software engineering

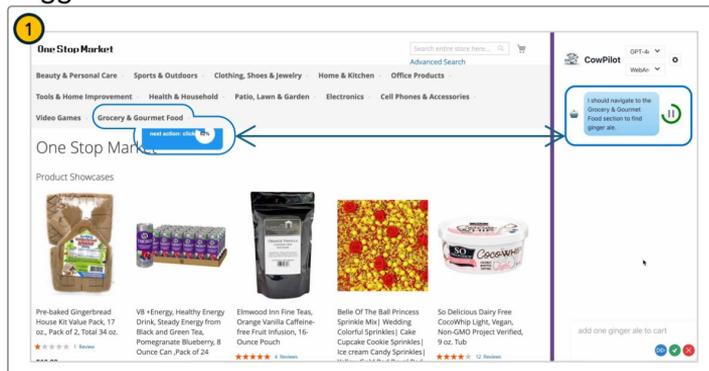


# Autonomous → Collaborative Agents

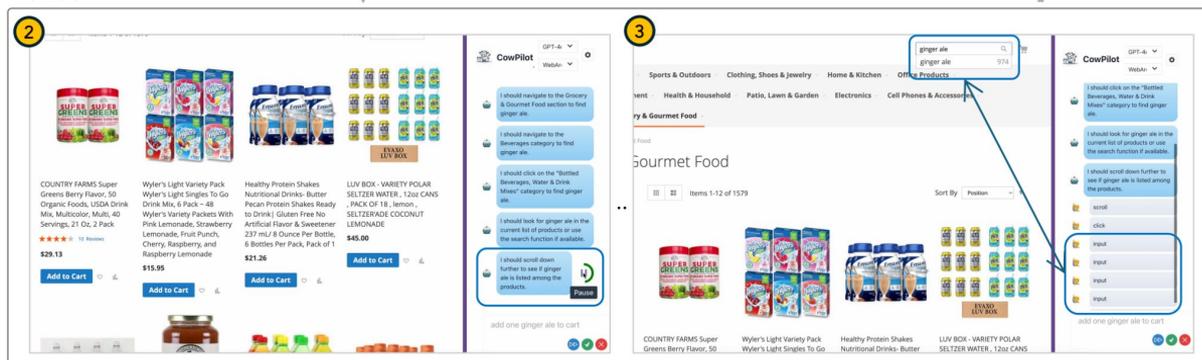
## CowPilot:

- Agent-Drive Task Sessions
- Human Verify and Intervene
- Improve overall success rate than human/agent-only

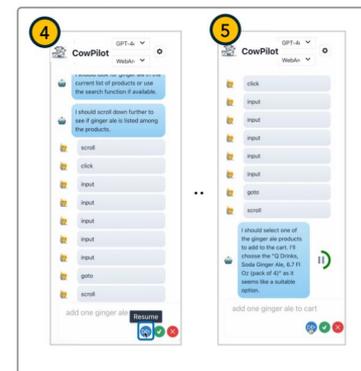
### Suggest



### Pause



### Resume



# Asynchronous, Collaborative Agents

**EditorUpdate**

**A detailed task description (126 words in total):**  
Help me plan a trip to Korea. For context, this trip is for ... We will land in Seoul 12/18 4:15 PM. We will leave Seoul for Taiwan on ...

Great! I'll have the updated itinery ready for you shortly. If there is anything else you'd like to add or change, feel free to let me know.

Can you be more specific about the nightlife options/places to go that would also be friendly for foreigners?

Also, can you recommend some good shopping places near that area?

**Editor Search Distance Matrix**

### Seoul Itinerary (12/18 - 12/22)

**\*\*Day 1: Arrival in Seoul (12/18)\*\***

- 4:15 PM: Arrive at Incheon International Airport.
- 5:30 PM: Take the Airport Railroad Express (AREX) to Hongdae (approx. 1 hour).
- 6:30 PM: Check into Airbnb in Hongdae.
- 7:30 PM: Dinner at a local Korean BBQ restaurant in Hongdae.
- 9:00 PM: Explore Hongdae's nightlife, visit themed cafes such as the 943 King's Cross Harry Potter Cafe or the Thanks Nature Cafe (sheep cafe), and enjoy street performances.

**\*\*Day 2: Sightseeing and Shopping (12/19)\*\***

The human adds more requests after reviewing the initial outcome. The LM agent refines the outcome accordingly.

**WaitTeammateContinue**

... Please review the report and let me know if there are any additional details you would like to include!

We don't need the intro part.

I still feel the conclusion is a bit unintuitive. Is there more analys we can do?

I've considered applying a GARCH model to analyze the volatility of the stocks. ... However, I encountered an error. Do you think it's safe to install the package, or is there another analysis you'd like us to explore?

**Editor CSV Viewer Jupyter Notebook**

```
IN [11]:
from arch import arch_model

# Function to fit GARCH model and summarize results
def fit_garch(data, name):
    returns =data['close'].pct_change().dropna() * 100
    model = arch_model(returns, vol='Garch', p=1, q=1)
    model_fit = model.fit(disp='off')
    print(f'GARCH Model Summary for {name}:')
    print(model_fit.summary())
    return model_fit

# Fit GARCH model for each stock
garch_300760 = fit_garch(stock_300760, '300760.XSHE')
garch_000796 = fit_garch(stock_000796, '000796.XSHE')
```

OUT [11]:  
ERROR: ModuleNotFoundError: No module named 'arch'

```
IN [12]:
1 %pip install arch
```

The LM agent asks for help. The human makes the critical decision.

# Summary

## LM Agent Basics

- Definition
- Observation
- Action space
- Memory

## Agent Building

- Benchmarks & Frameworks
- Agent Foundation Model Training
- Autonomous vs. Collaborative Agents

# LM-based Agents

Zora Wang

