Generalist Web Agents:
From Perception to Planning

Boyuan Zheng

04/04/2025

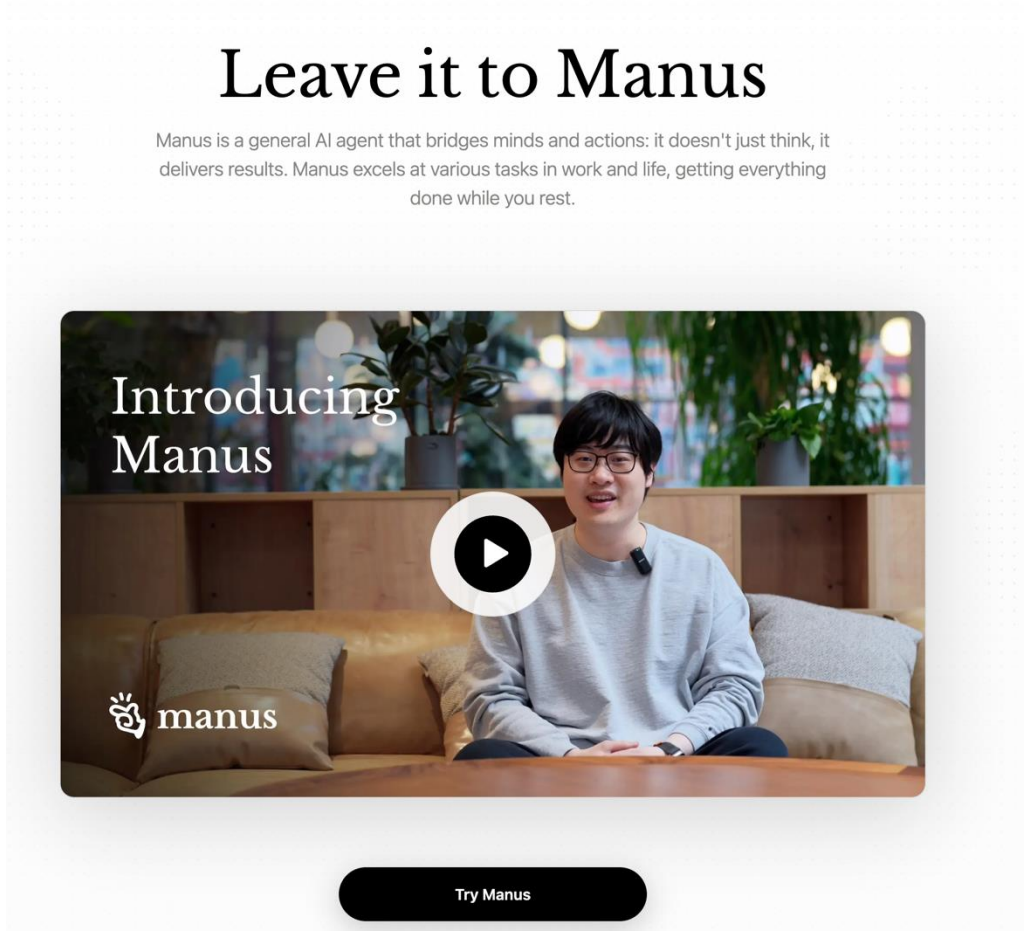
Outline

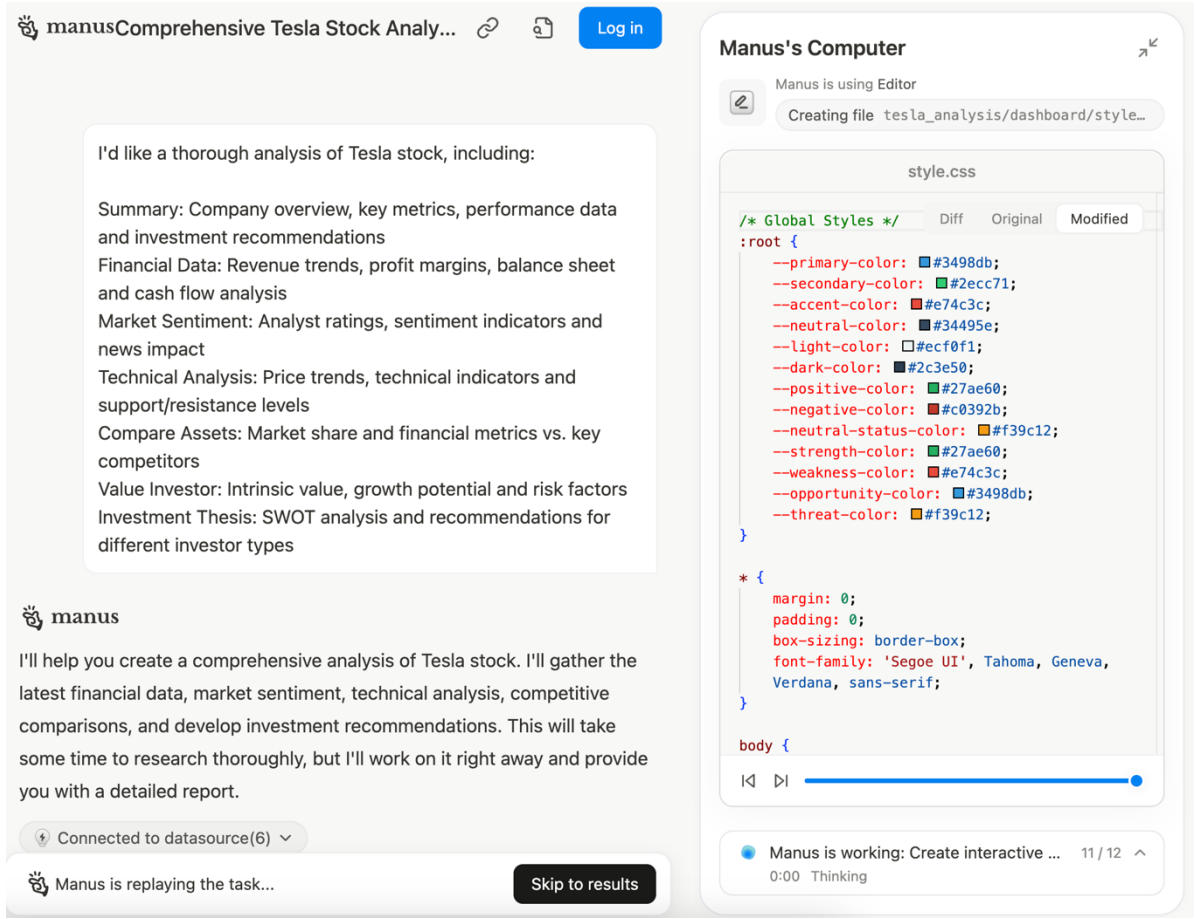
1. Introduction
2. Environment Perception
3. Planning
4. Self-Improvement

Outline

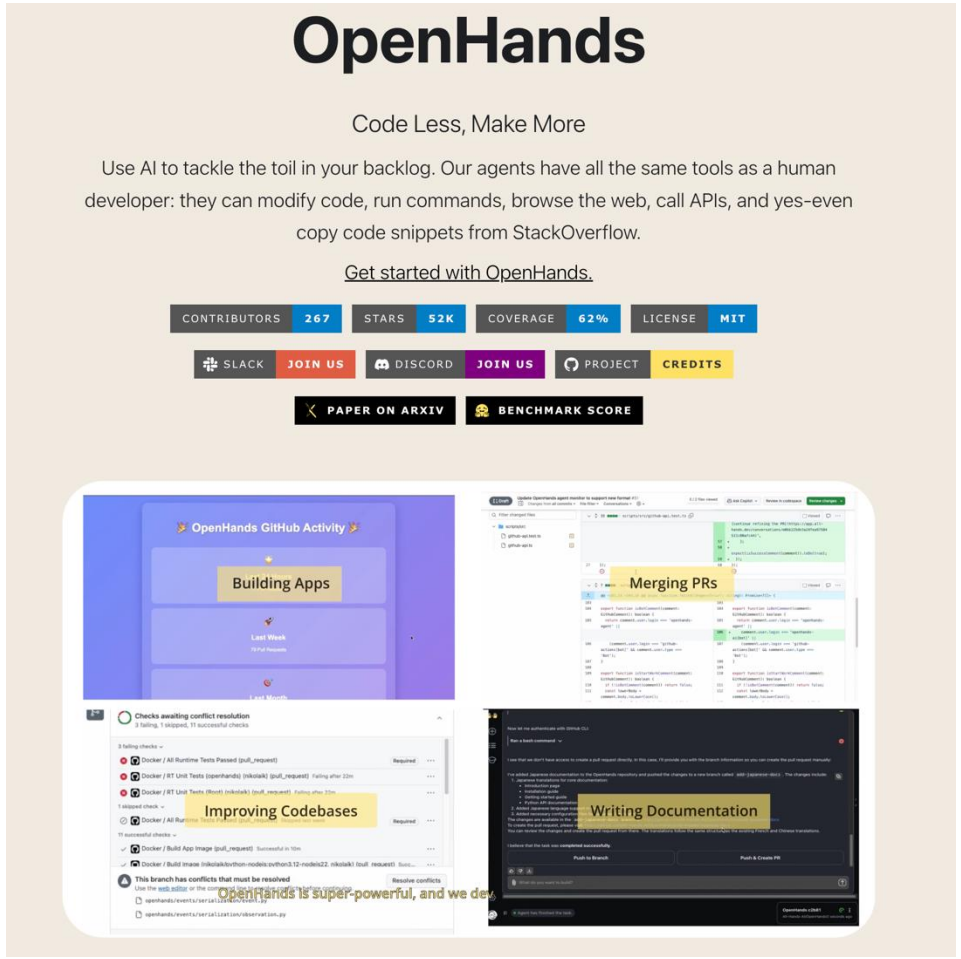
1. Introduction
2. Environment Perception
3. Planning
4. Self-Improvement

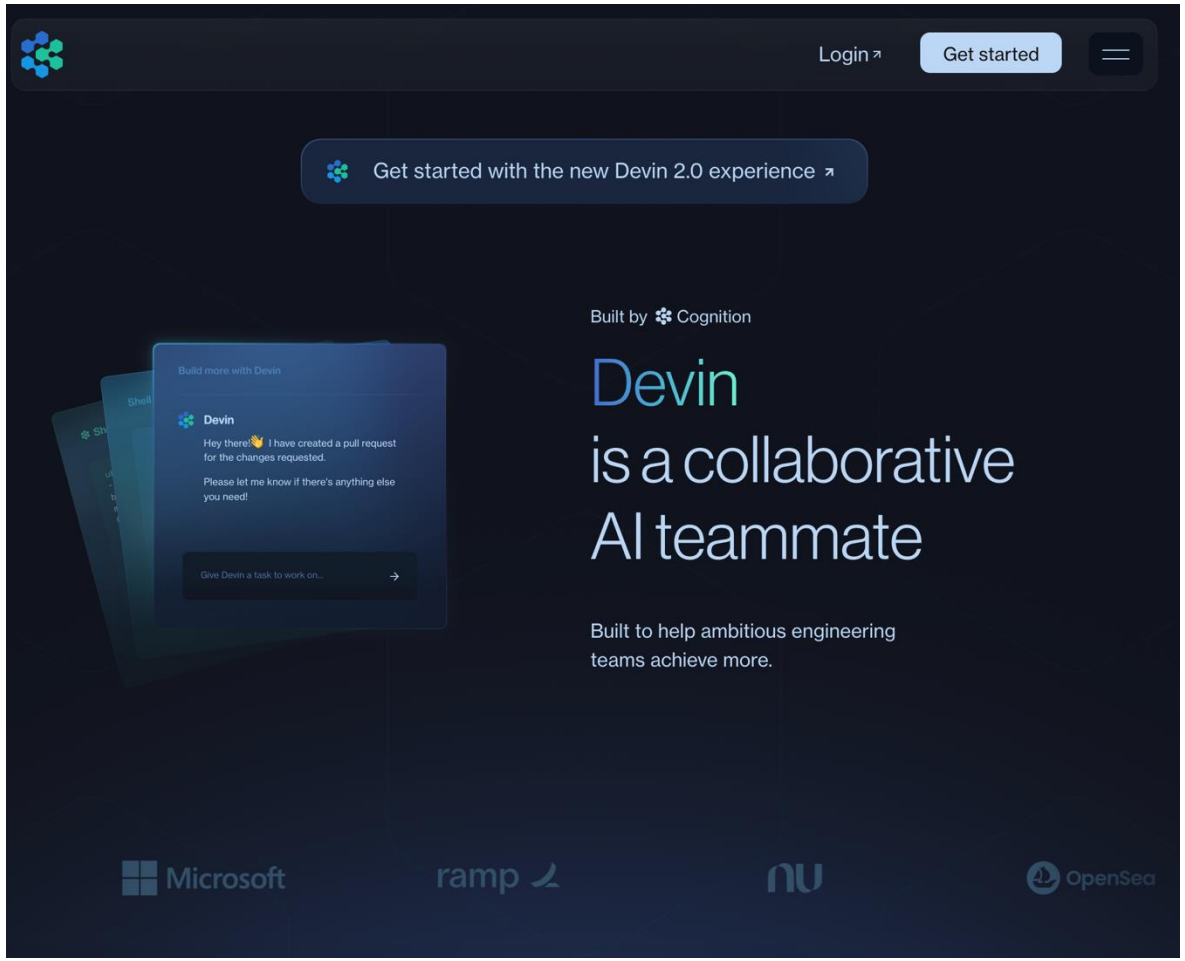
Coding Agent





Coding Agent



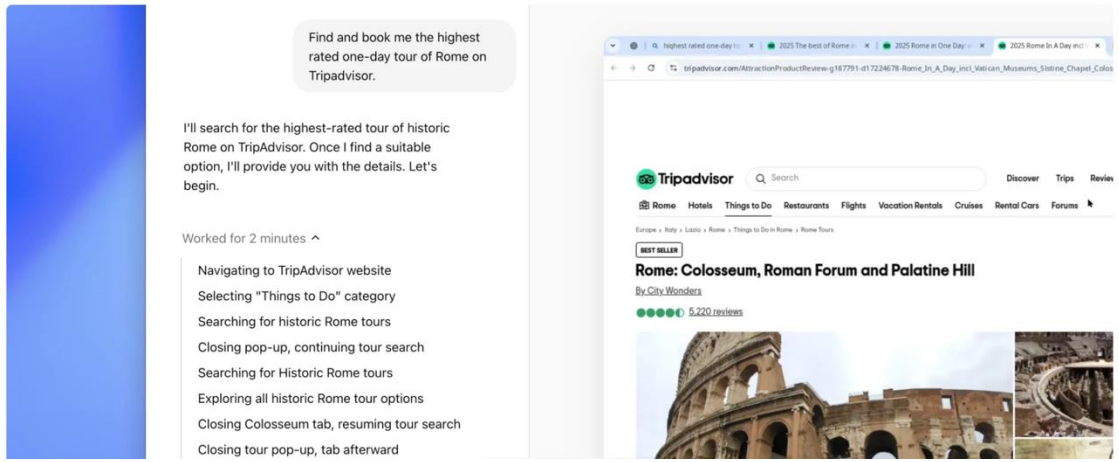


Web Agent

Introducing Operator

A research preview of an agent that can use its own browser to perform tasks for you. Available to Pro users in the U.S.

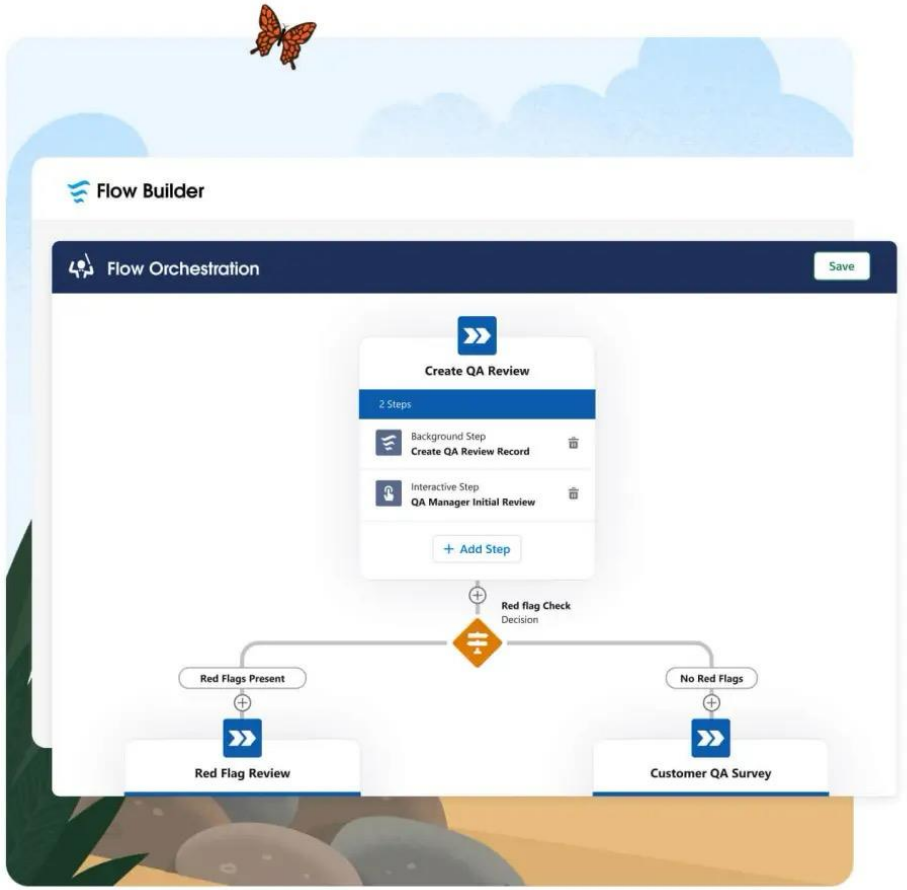
Go to Operator ↗



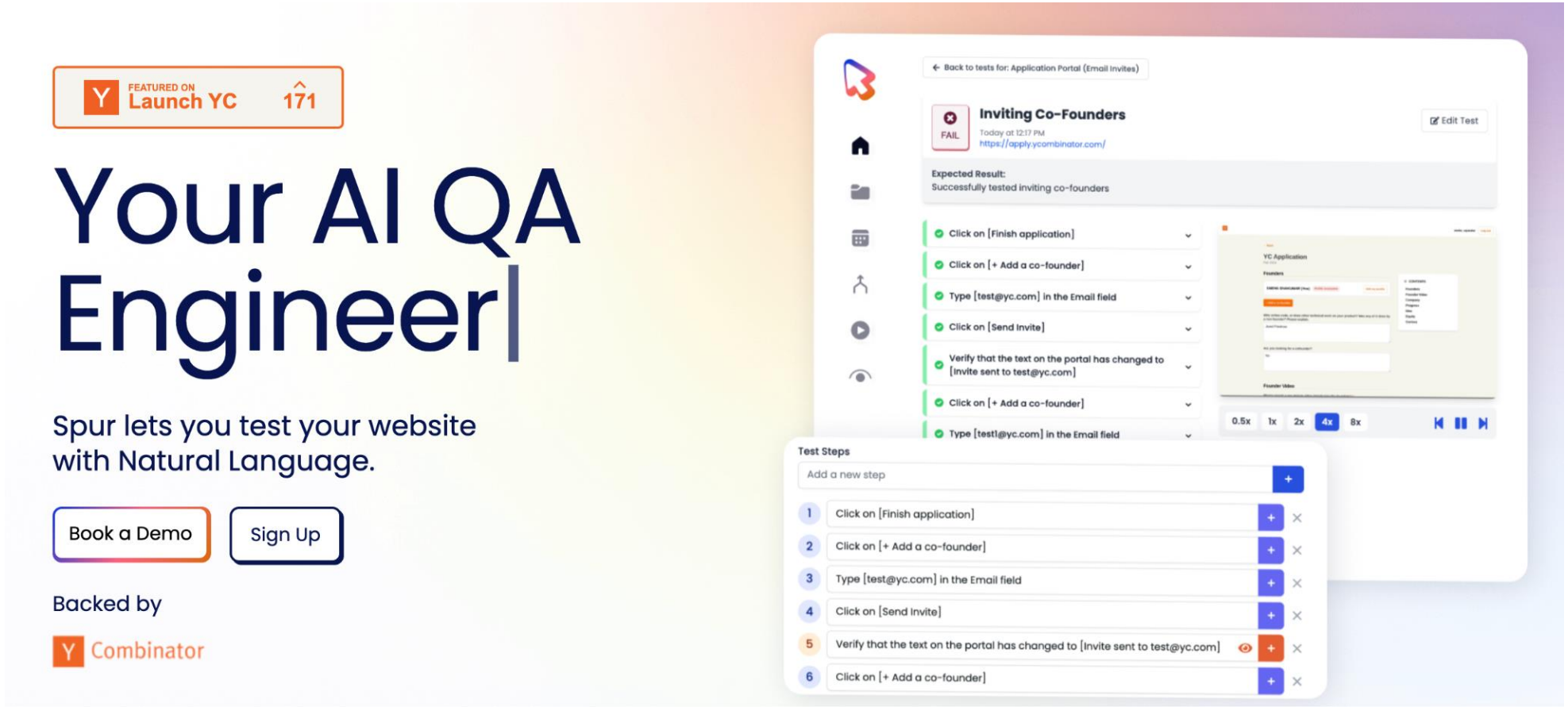
Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku

Oct 22, 2024 • 5 min read

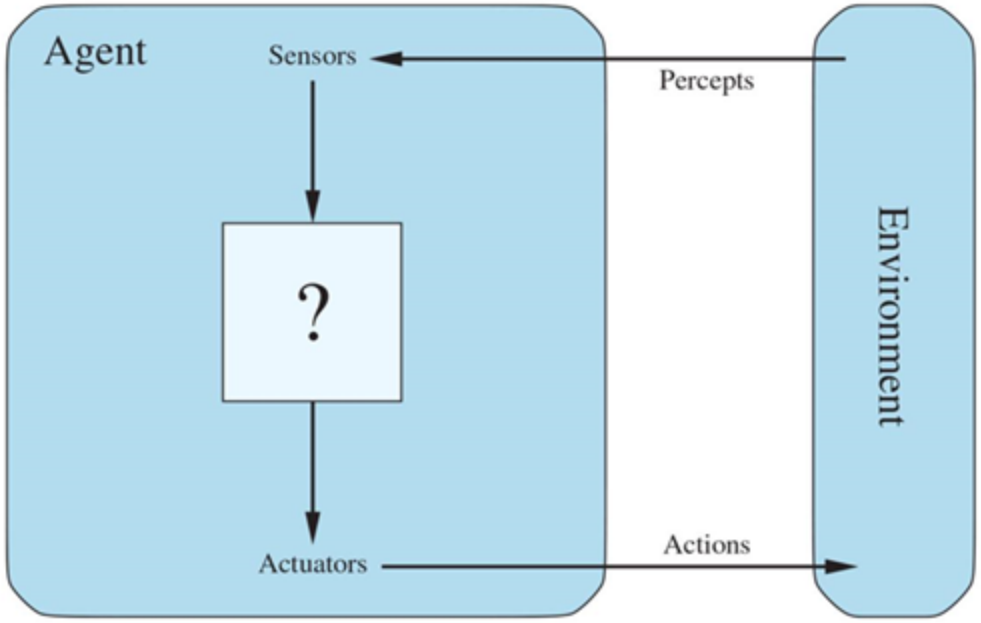
Web Agent Use Case: Process Automation



Web Agent Use Case: Auto-Testing



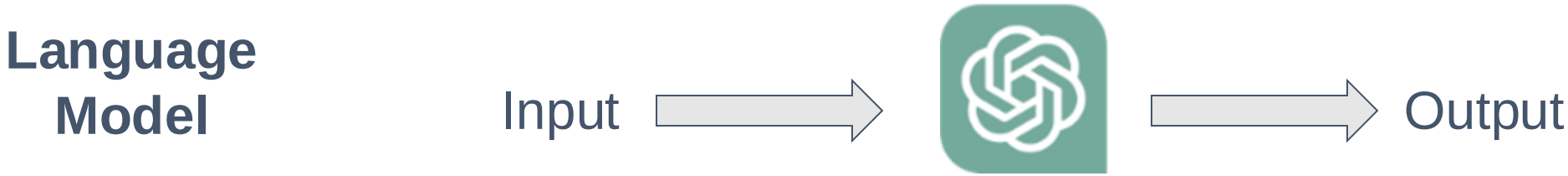
Why Agents Again?



"An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**"

— Russel & Norvig, *AI: A Modern Approach*

From Language Model to Language Agent

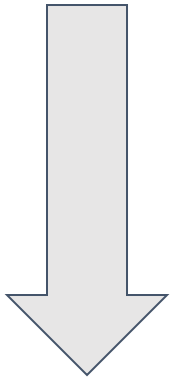


Opportunities of contemporary AI Agent with an integrated LLM:

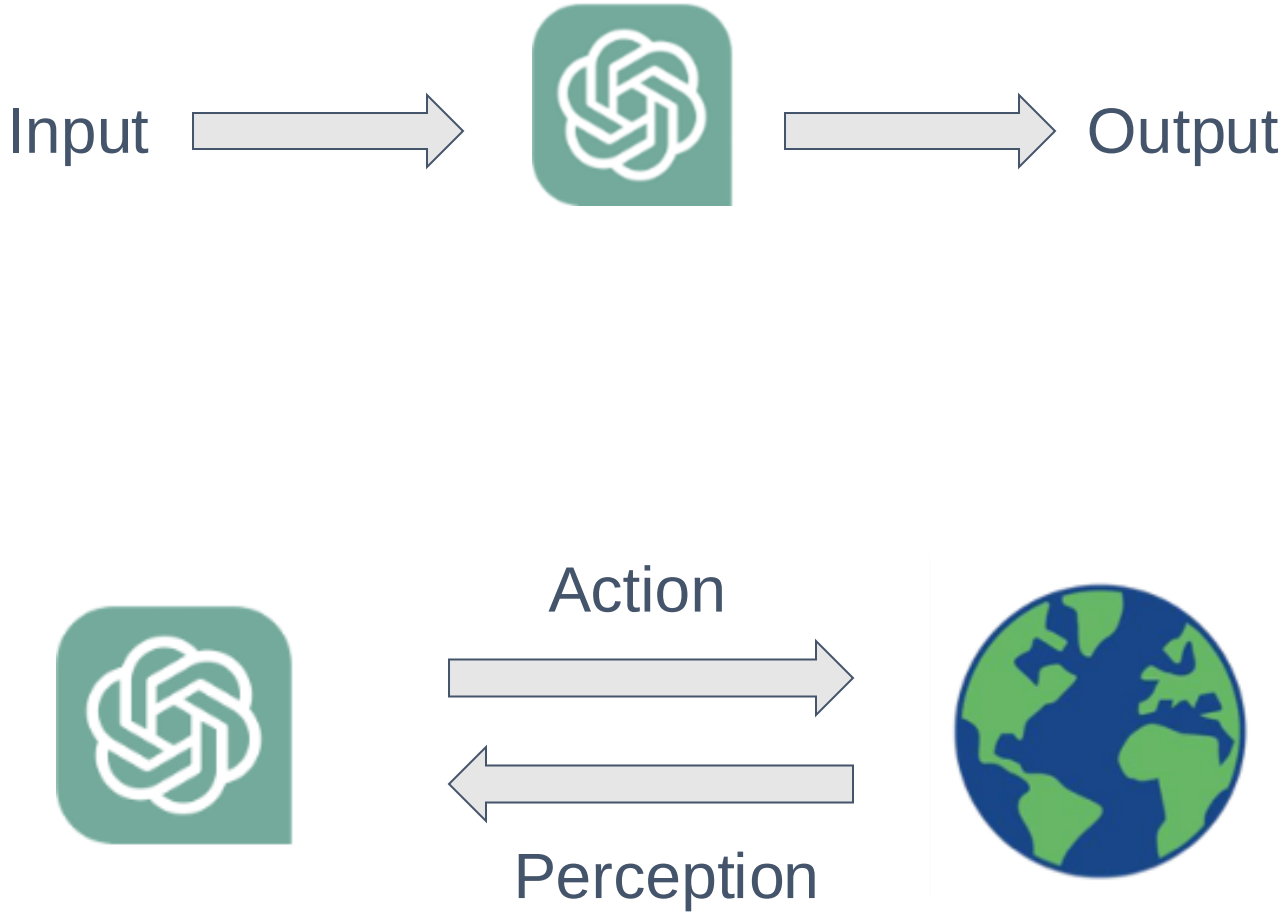
- Environment Perception
- Generalist agent with language
- Reasoning for better acting

From Language Model to Language Agent

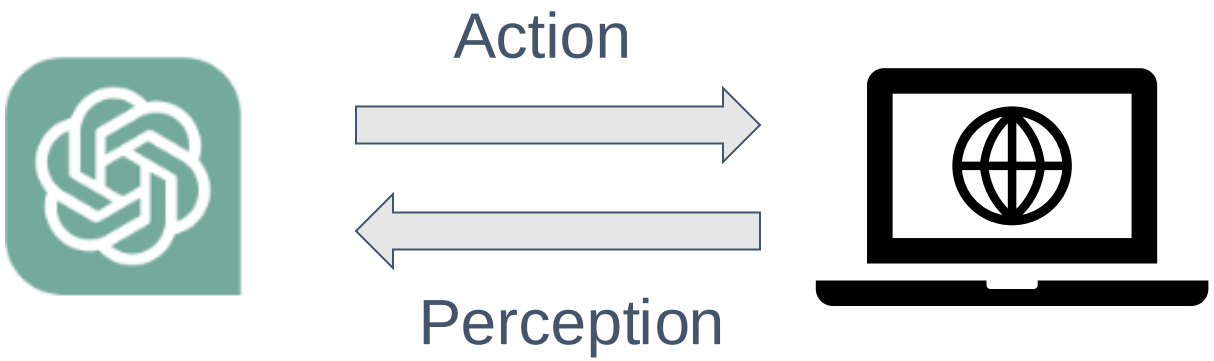
Language Model



Language Agent



Web Agent



Web as playgrounds for agent research

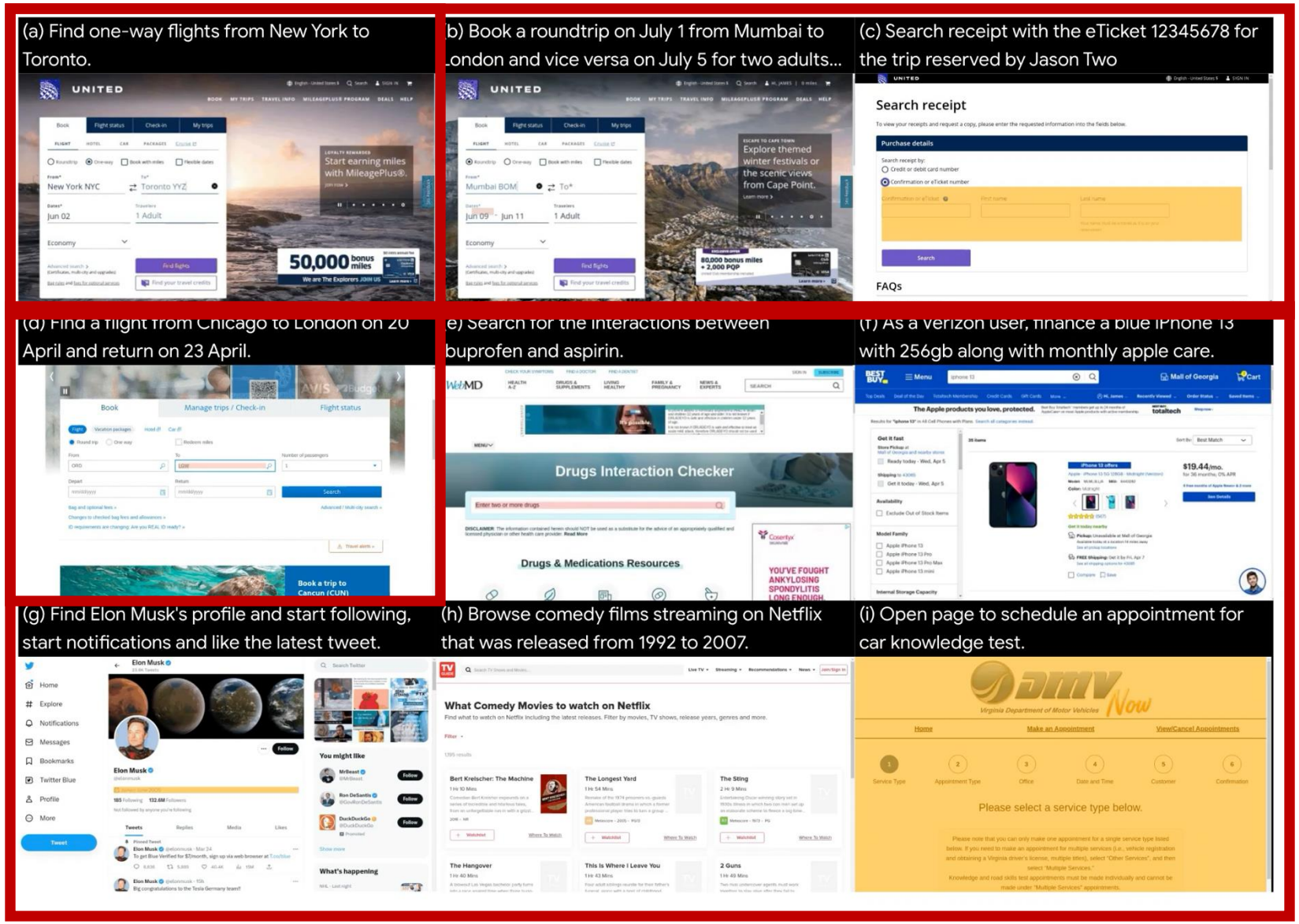
- Open, Diverse, Complex Environment
- Highly Structured Environment
- Low Cost
- Lower Safety Concern

Generalist Web Agent

Cross-Task

Cross-Website

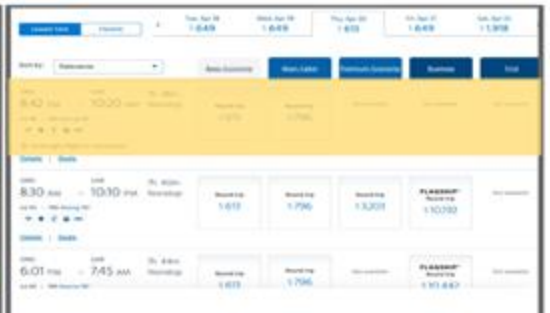
Cross-Domain



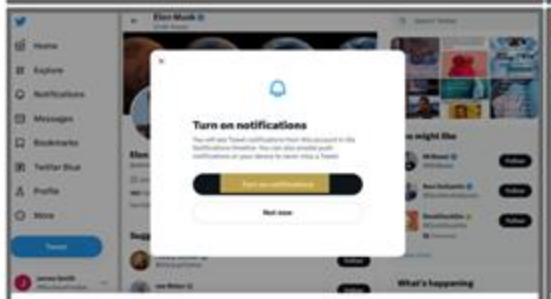
Why a Challenging Mission?

(a) Find one-way flights from New York to Toronto.

(b) Book a roundtrip on July 1 from Mumbai to London and vice versa on July 5 for two adults.



(c) Find a flight from Chicago to London on 20 April and return on 23 April.



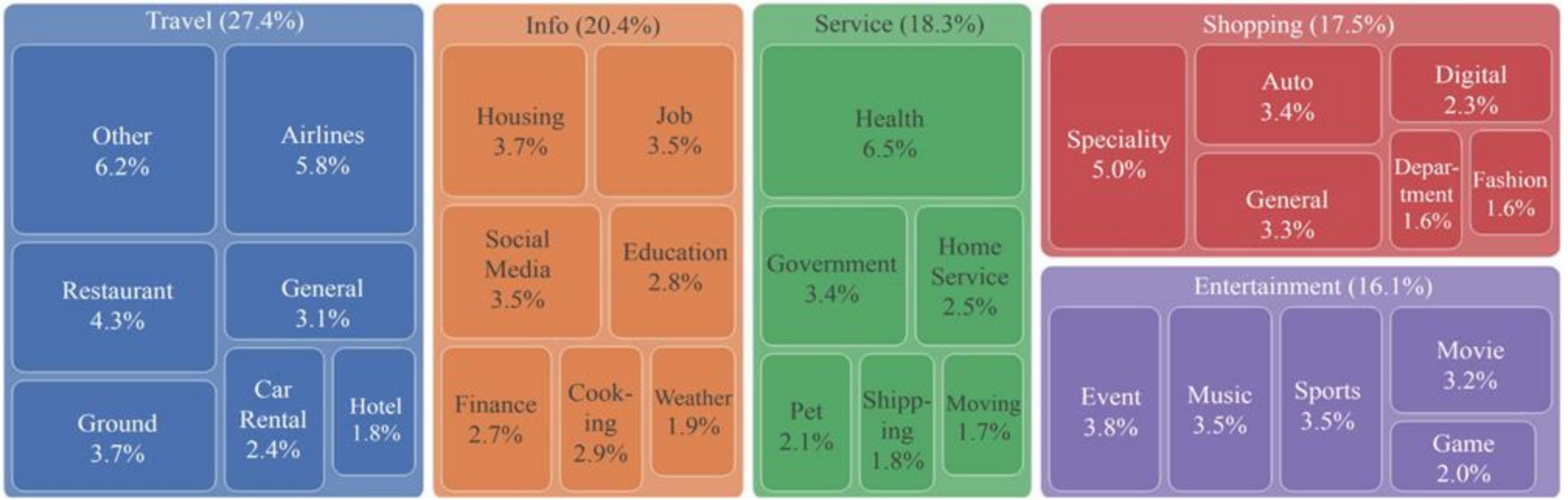
(d) Find Elon Musk's profile and follow, start notifications and like the latest tweet.

(e) Browse comedy films streaming on Netflix that was released from 1992 to 2007.

(f) Open page to schedule an appointment for car knowledge test.

- User tasks: diverse, complex, requiring long-horizon planning
- Real-world websites: complex, dynamic
- Generalize to unseen websites/domains

Mind2Web: A Generalist Web Agent Benchmark



	# Dom.	# Env.	Env. Type	Avg. # Elements	# Tasks	Task Info.	Avg. # Actions
MiniWoB++ [22]	–	100	Simplified mobile websites	28	100	Low-level	3.6
WebShop [40]	1	1	Simplified shopping websites	38	12,000 products	High-level	11.3
RUSS [39]	–	22	Real-world websites	801	80	High & low	5.4
PixelHelp [21]	4	4	Mobile apps	–	187	High & low	-
META-GUI [35]	6	11	Mobile apps	79	1,125 dialogues	High-level	4.3
MoTIF [5]	15	125	Mobile apps	188	756	High & Low	4.4
MIND2WEB	5 / 31	137	Real-world websites	1,135	2,350	High-level	7.3

A task example in Mind2Web

Task Description:
Show me the reviews for the auto repair business closest to 10002.

Action Sequence:

	Target Element	Operation
1.	[searchbox] *Find*	**TYPE:** ***auto repair***
2.	[button] *Auto Repair*	**CLICK**
3.	[textbox] *Near*	**TYPE:** ***10002***
4.	[button] *10002*	**CLICK**
5.	**[button] *Search***	**CLICK**
6.	[switch] *Show BBB Accredited only*	**CLICK**
7.	[svg]	**CLICK**
8.	[button] *Sort By*	**CLICK**
9.	**[link] *Fast Lane 24 Hour Auto Repair***	**CLICK**
10.	[link] *Read Reviews*	**CLICK**

Webpage Snapshots:

Action 1

```
<input name="find_text"
type="search">
```

Action 2

```
<em>Auto Repair</em>
```

Action 5

```
<button>Search</button>
```

Action 6


```
<button>Show BBB Accredited
only</button>
```

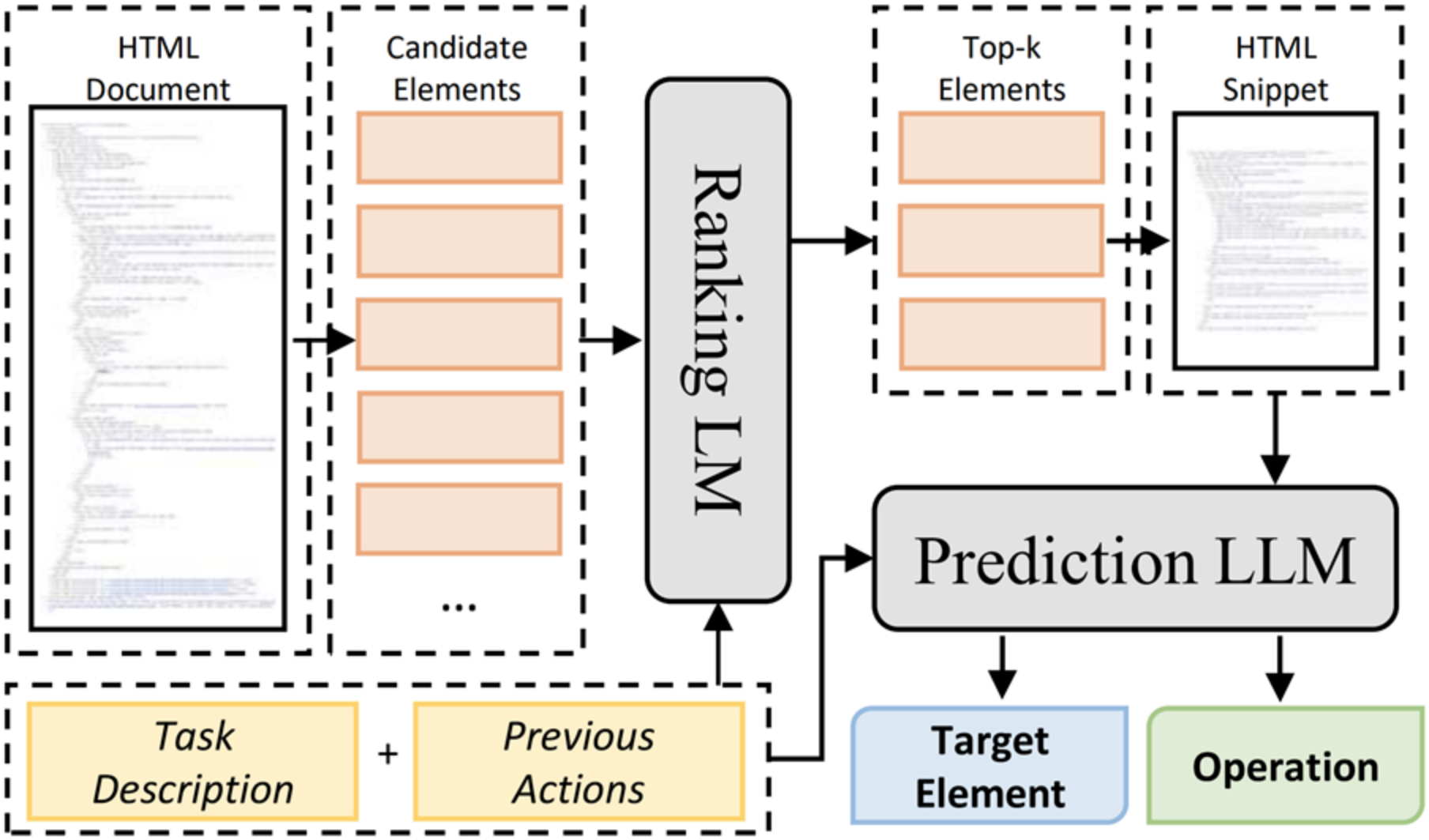
Action 9


```
<span>Fast Lane 24 Hour Auto
Repair</span>
```

Action 10


```
<a href="link:XXX">Read
Reviews</a>
```

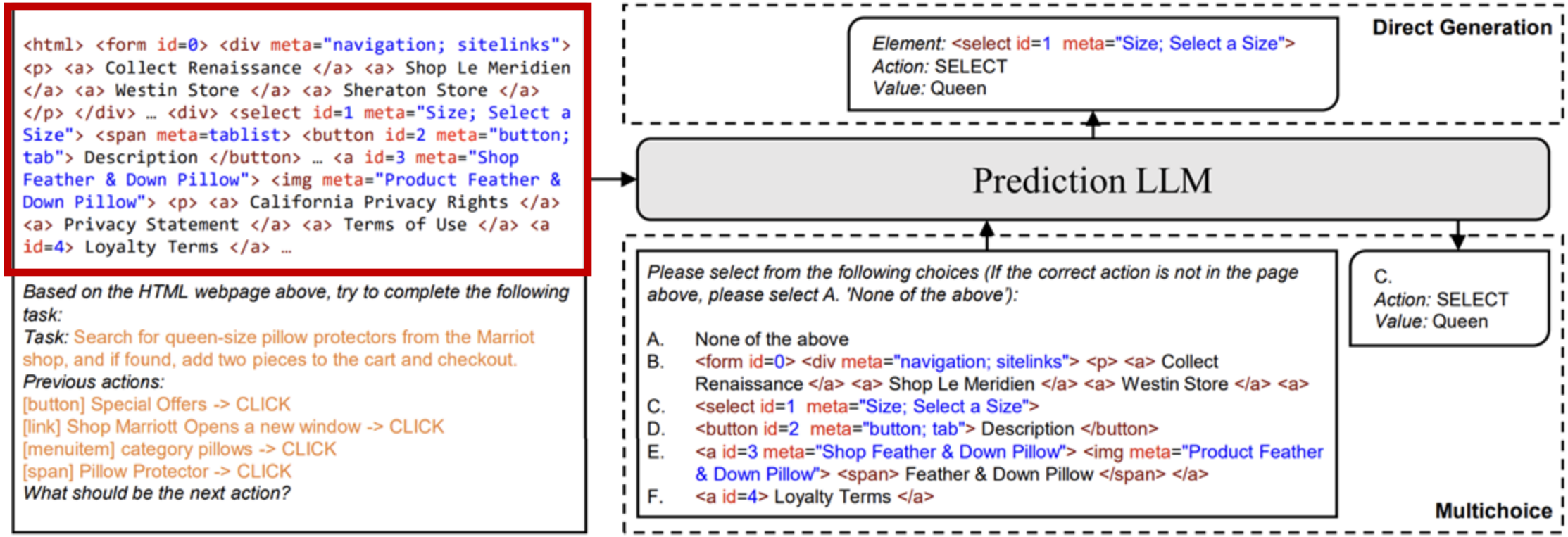
The first framework for building a generalist web agent



Our Proposed MindAct

MindAct

- Environment Representation: Simplified HTML



Results of MindAct

	Cross-Task				Cross-Website				Cross-Domain			
	Ele. Acc	**Op. F1**	**Step SR**	**SR**	**Ele. Acc**	**Op. F1**	**Step SR**	**SR**	**Ele. Acc**	**Op. F1**	**Step SR**	**SR**
Classification	26.8	–	–	–	21.6	–	–	–	24.5	–	–	–
Generation	20.2	52.0	17.5	0.0	13.9	44.7	11.0	0.0	14.2	44.7	11.9	0.4
MINDACT												
w/ Flan-T5$_B$	43.6	76.8	41.0	4.0	32.1	**67.6**	29.5	1.7	33.9	**67.3**	31.6	1.6
w/ Flan-T5$_L$	53.4	**75.7**	50.3	**7.1**	39.2	67.1	35.3	1.1	39.7	67.2	37.3	2.7
w/ Flan-T5$_{XL}$	**55.1**	**75.7**	**52.0**	5.2	**42.0**	65.2	**38.9**	**5.1**	**42.1**	66.5	**39.6**	**2.9**
w/ GPT-3.5	20.3	56.6	17.4	0.8	19.3	48.8	16.2	0.6	21.6	52.8	18.6	1.0
w/ GPT-4*	41.6	60.6	36.2	2.0	35.8	51.1	30.1	2.0	37.1	46.5	26.4	2.0

Step SR: success rate at each step
SR: Success rate for the whole task
Offline evaluation setting

GPT-4 (3-shot) is close to fine-tuned Flan-T5 models

Results of MindAct

	Cross-Task				Cross-Website				Cross-Domain			
	Ele. Acc	**Op. F1**	**Step SR**	**SR**	**Ele. Acc**	**Op. F1**	**Step SR**	**SR**	**Ele. Acc**	**Op. F1**	**Step SR**	**SR**
Classification	26.8	–	–	–	21.6	–	–	–	24.5	–	–	–
Generation	20.2	52.0	17.5	0.0	13.9	44.7	11.0	0.0	14.2	44.7	11.9	0.4
MINDACT												
w/ Flan-T5$_B$	43.6	76.8	41.0	4.0	32.1	**67.6**	29.5	1.7	33.9	**67.3**	31.6	1.6
w/ Flan-T5$_L$	53.4	**75.7**	50.3	**7.1**	39.2	67.1	35.3	1.1	39.7	67.2	37.3	2.7
w/ Flan-T5$_{XL}$	**55.1**	**75.7**	**52.0**	5.2	**42.0**	65.2	**38.9**	**5.1**	**42.1**	66.5	**39.6**	**2.9**
w/ GPT-3.5	20.3	56.6	17.4	0.8	19.3	48.8	16.2	0.6	21.6	52.8	18.6	1.0
w/ GPT-4*	41.6	60.6	36.2	2.0	35.8	51.1	30.1	2.0	37.1	46.5	26.4	2.0

Step SR: success rate at each step
SR: Success rate for the whole task
Offline evaluation setting

GPT-4 (3-shot) is close to fine-tuned Flan-T5 models, but all models are terrible (2-7% whole task success rate)!

Around six months later …

Model	Cross-Task			Cross-Website			Cross-Domain		
	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR
Supervised Fine-Tuning									
FLAN-T5-XL	**57.1**	75.7	**53.5**	**43.8**	67.7	**41.1**	**41.4**	65.9	**38.9**
BLIP-2-T5-XL	50.1	**77.0**	47.0	39.4	**69.3**	37.0	41.2	**69.3**	38.9
In-Context Learning									
GPT-3.5*	19.4	59.2	16.8	14.9	56.5	14.1	25.2	57.9	24.1
GPT-4*	40.8	63.1	32.3	30.2	61.0	27.0	35.4	61.9	29.7
CogAgent	22.4	53.0	17.6	18.4	42.2	13.4	20.6	42.0	15.5
?	9.7	65.6	8.1	9.1	60.8	7.5	10.9	63.9	8.5
	21.5	67.7	19.6	17.1	61.3	15.4	20.7	64.3	18.0
	46.4	**73.4**	**40.2**	**38.0**	**67.8**	**32.4**	**42.4**	**69.3**	**36.8**
	66.4	79.2	61.9	69.5	78.9	65.0	72.8	73.6	62.1

5~7% gain

Around six months later …

Model	Cross-Task			Cross-Website			Cross-Domain		
	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR
Supervised Fine-Tuning									
FLAN-T5-XL	**57.1**	75.7	**53.5**	**43.8**	67.7	**41.1**	**41.4**	65.9	**38.9**
BLIP-2-T5-XL	50.1	**77.0**	47.0	39.4	**69.3**	37.0	41.2	**69.3**	38.9
In-Context Learning									
GPT-3.5*	19.4	59.2	16.8	14.9	56.5	14.1	25.2	57.9	24.1
GPT-4*	40.8	63.1	32.3	30.2	61.0	27.0	35.4	61.9	29.7
CogAgent	22.4	53.0	17.6	18.4	42.2	13.4	20.6	42.0	15.5
?	9.7	65.6	8.1	9.1	60.8	7.5	10.9	63.9	8.5
	21.5	67.7	19.6	17.1	61.3	15.4	20.7	64.3	18.0
	46.4	**73.4**	**40.2**	**38.0**	**67.8**	**32.4**	**42.4**	**69.3**	**36.8**
	66.4	79.2	61.9	69.5	78.9	65.0	72.8	73.6	62.1

Whole task success rate

	Offline_0	Offline_1	Online
FLAN-T5-XL	4.4	24.4	8.9
GPT-4	1.1	12.2	13.3
?	3.3	12.2	37.8
	13.3	27.8	51.1

So, what did we do?

Model	Cross-Task			Cross-Website			Cross-Domain		
	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR	Ele. Acc	Op. F1	Step SR
Supervised Fine-Tuning									
FLAN-T5-XL	**57.1**	75.7	**53.5**	**43.8**	67.7	**41.1**	**41.4**	65.9	**38.9**
BLIP-2-T5-XL	50.1	**77.0**	47.0	39.4	**69.3**	37.0	41.2	**69.3**	38.9
In-Context Learning									
GPT-3.5*	19.4	59.2	16.8	14.9	56.5	14.1	25.2	57.9	24.1
GPT-4*	40.8	63.1	32.3	30.2	61.0	27.0	35.4	61.9	29.7
CogAgent	22.4	53.0	17.6	18.4	42.2	13.4	20.6	42.0	15.5
?	9.7	65.6	8.1	9.1	60.8	7.5	10.9	63.9	8.5
	21.5	67.7	19.6	17.1	61.3	15.4	20.7	64.3	18.0
	46.4	**73.4**	**40.2**	**38.0**	**67.8**	**32.4**	**42.4**	**69.3**	**36.8**
	66.4	79.2	61.9	69.5	78.9	65.0	72.8	73.6	62.1

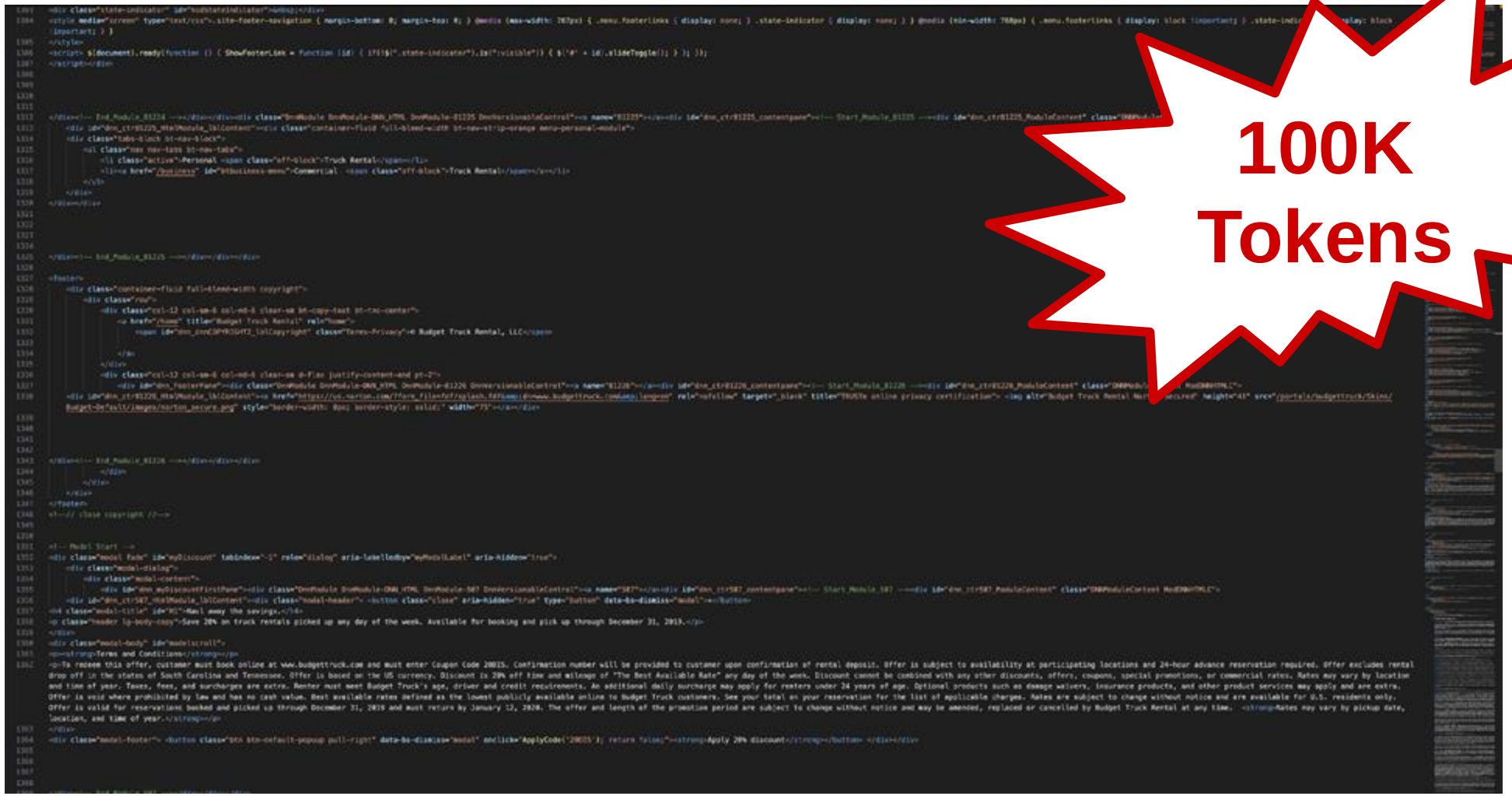
Whole task success rate

	Offline_0	Offline_1	Online
FLAN-T5-XL	4.4	24.4	8.9
GPT-4	1.1	12.2	13.3
?	3.3	12.2	37.8
	13.3	27.8	51.1

Outline

1. Introduction
2. Environment Perception
3. Planning
4. Self-Improvement

HTML -> Simplified HTML -> Image



HTML -> Simplified HTML -> Image

HTML -> Simplified HTML -> Image

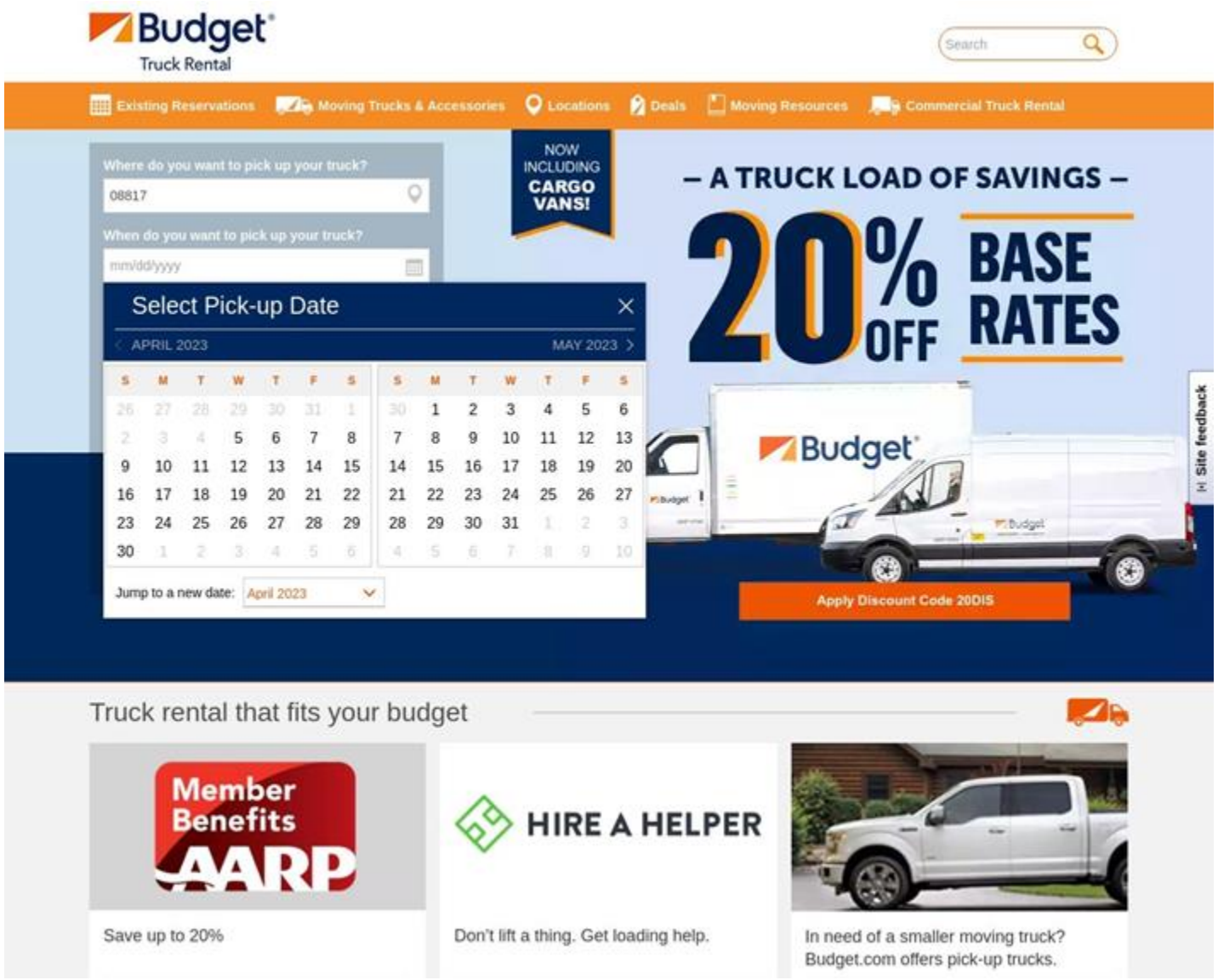
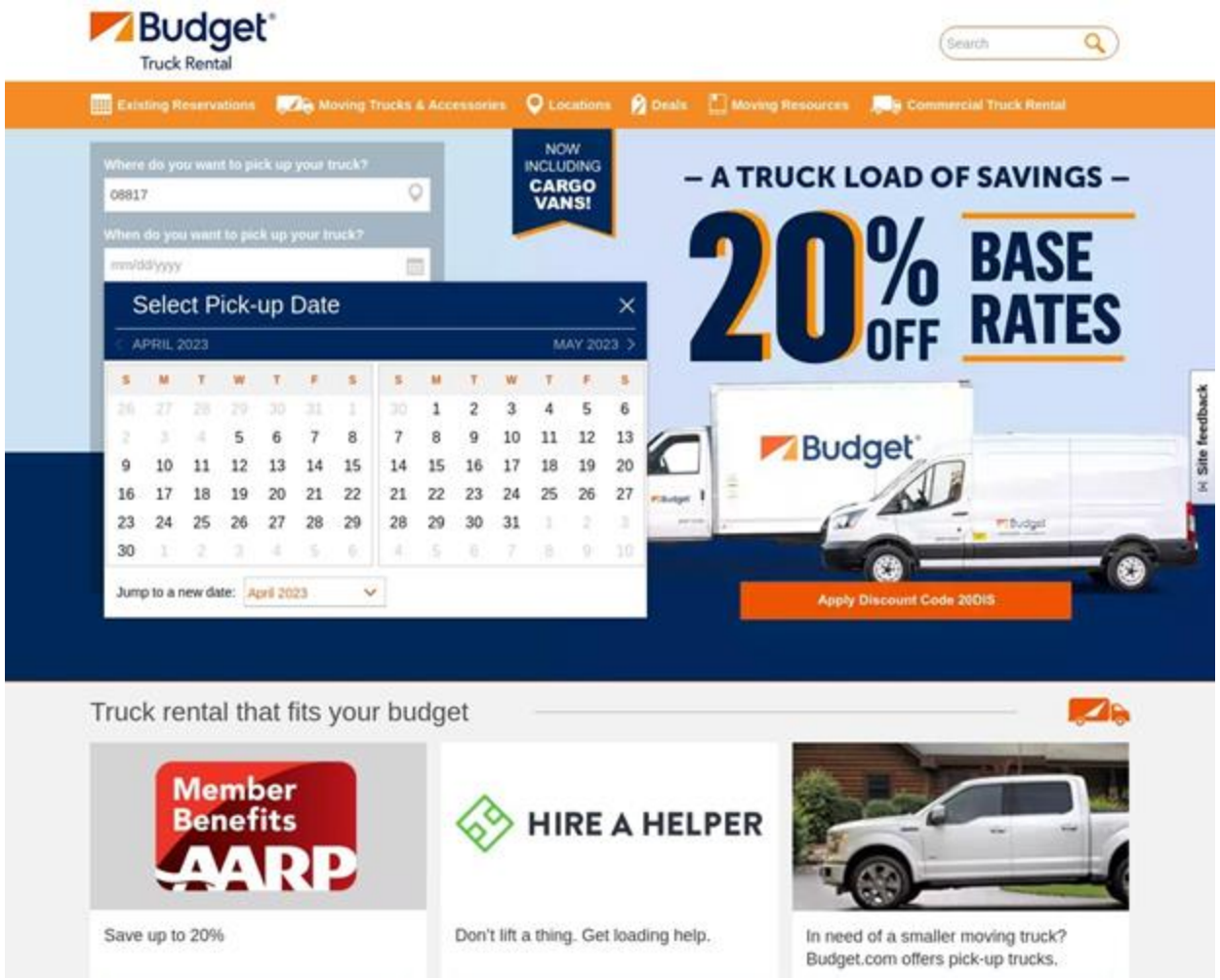
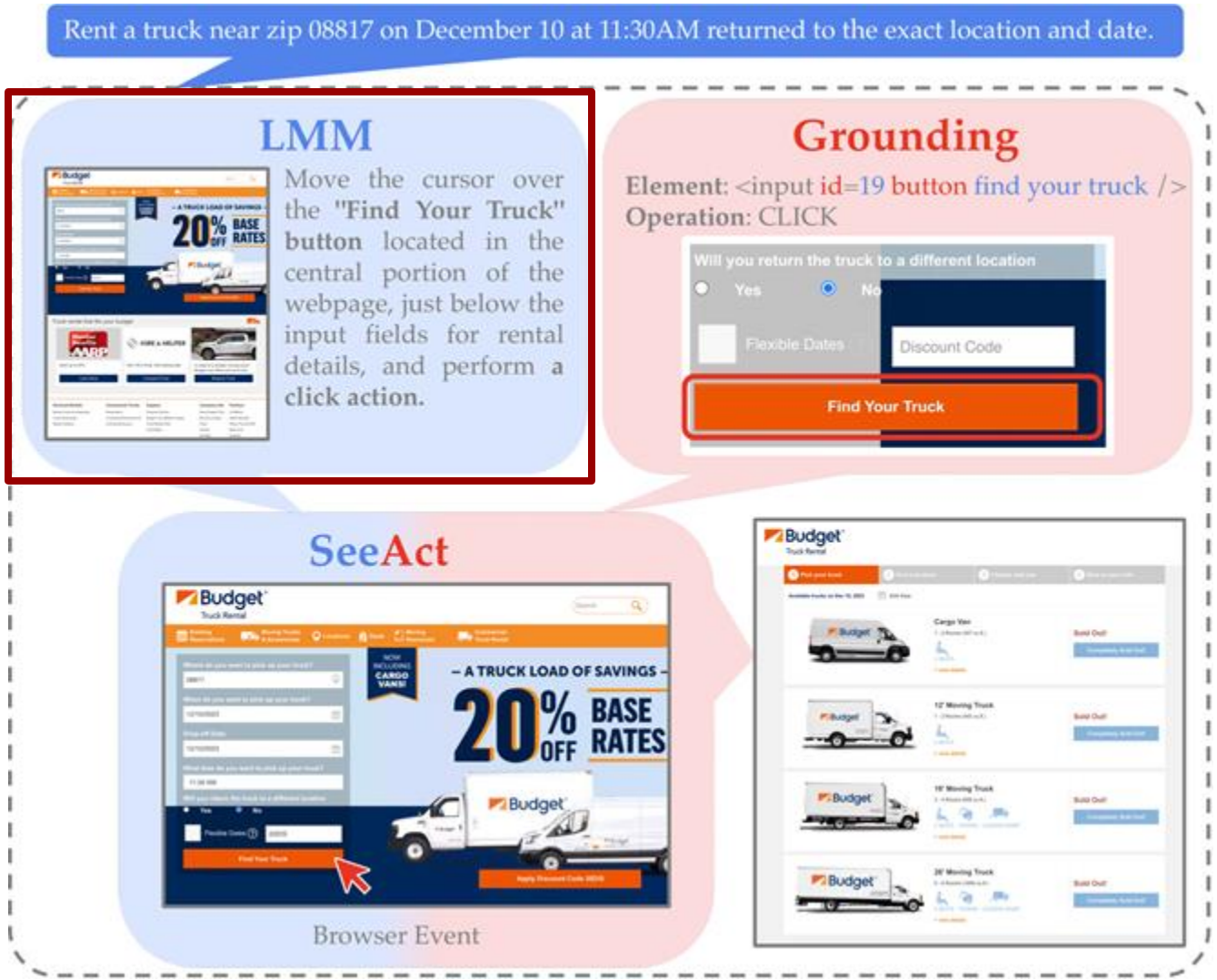


Image: A picture is worth thousands of words



SeeAct Paradigm

Action Generation -> Action Grounding

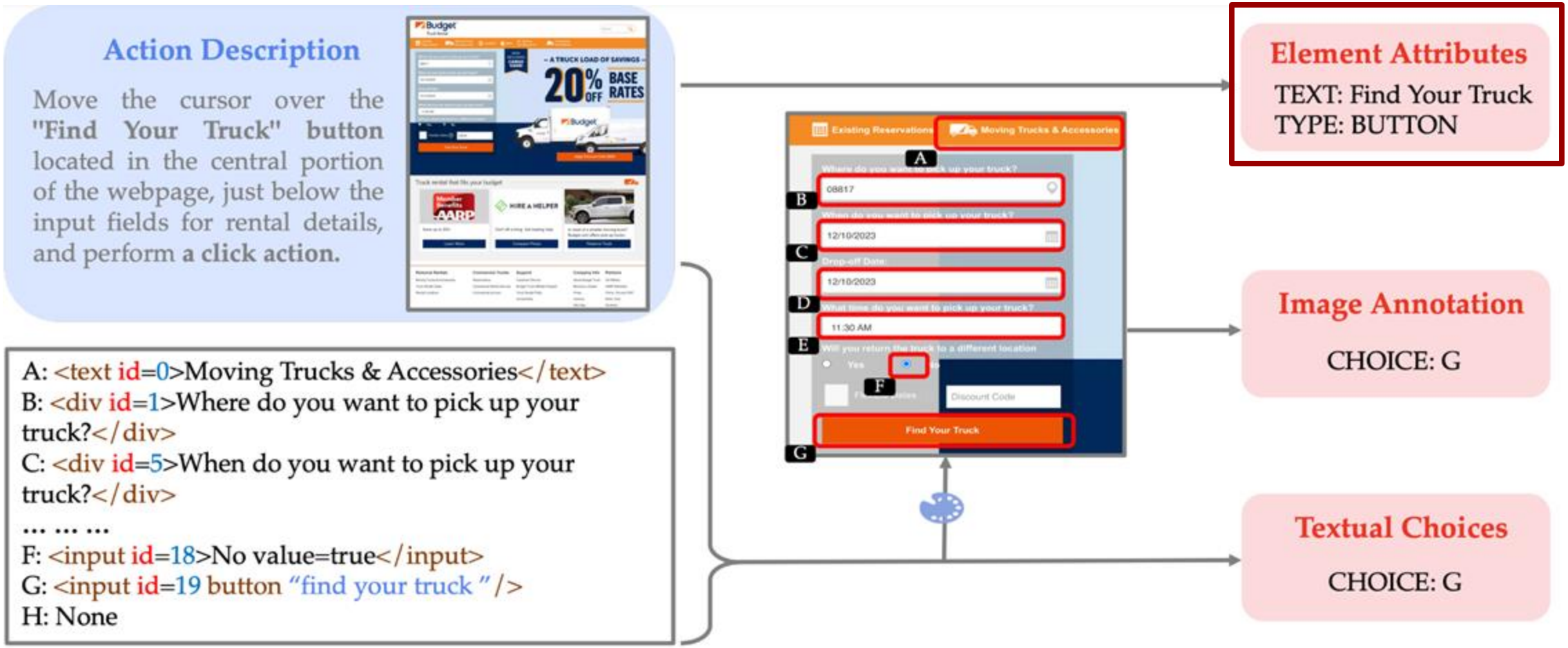


SeeAct Paradigm

Action Generation -> **Action Grounding**

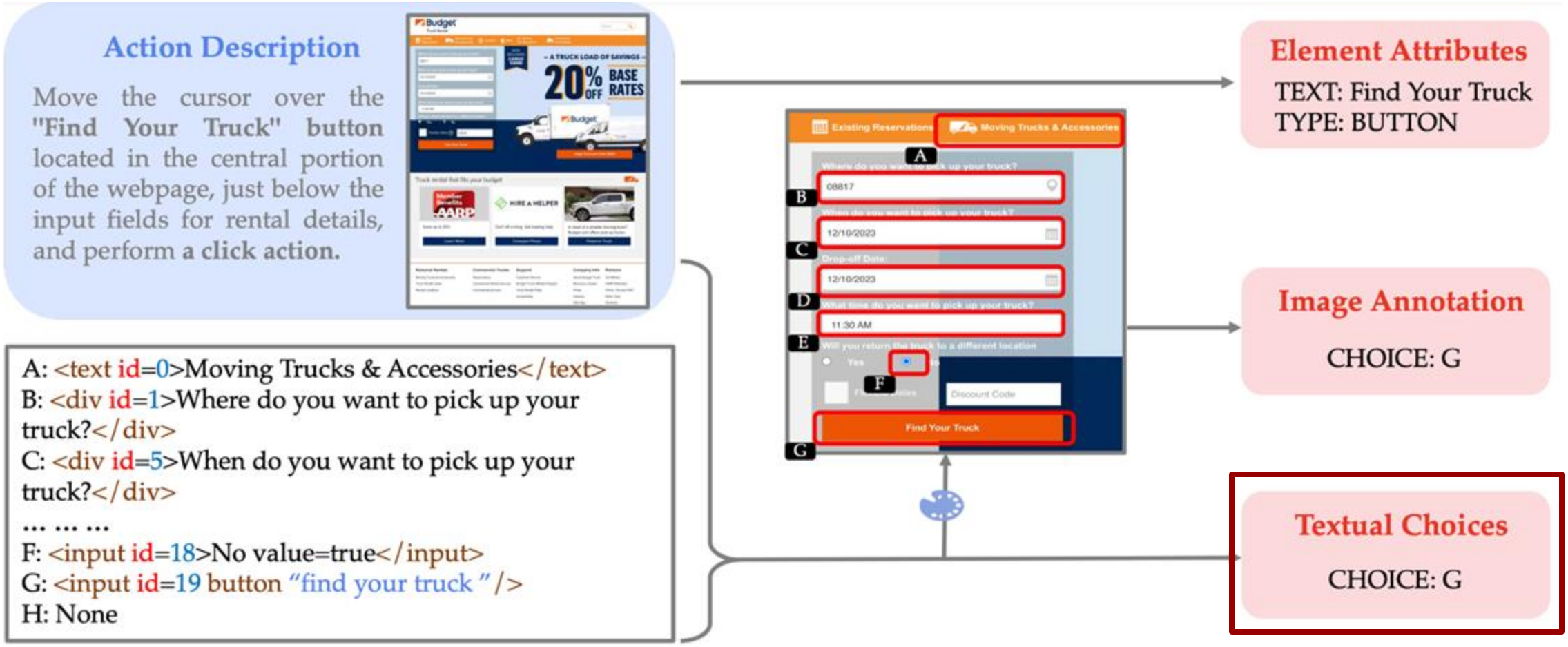
SeeAct Paradigm

Action Generation -> **Action Grounding**



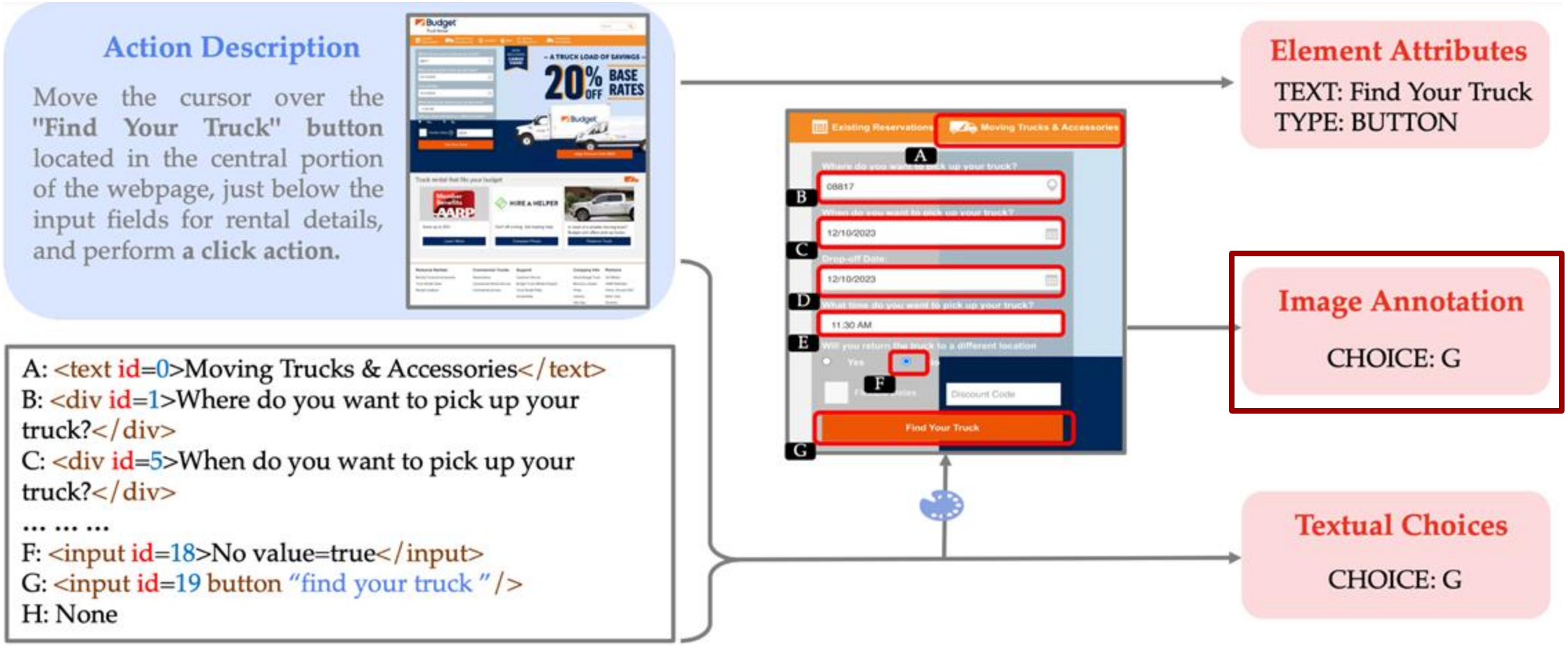
SeeAct Paradigm

Action Generation -> **Action Grounding**



SeeAct Paradigm

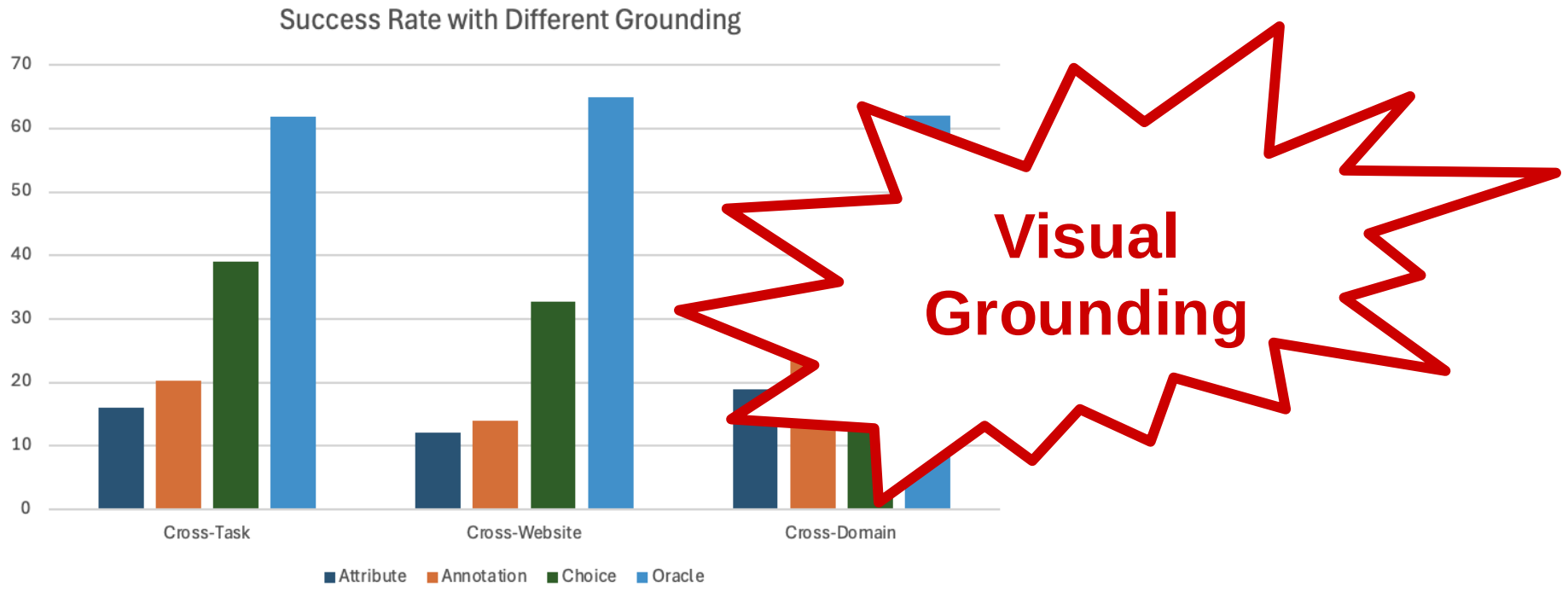
Action Generation -> **Action Grounding**



Grounding Strategies

Oracle: Ask humans to identify model's intended actions from Action Description

Large Margin between Oracle and SOTA Grounding method so far

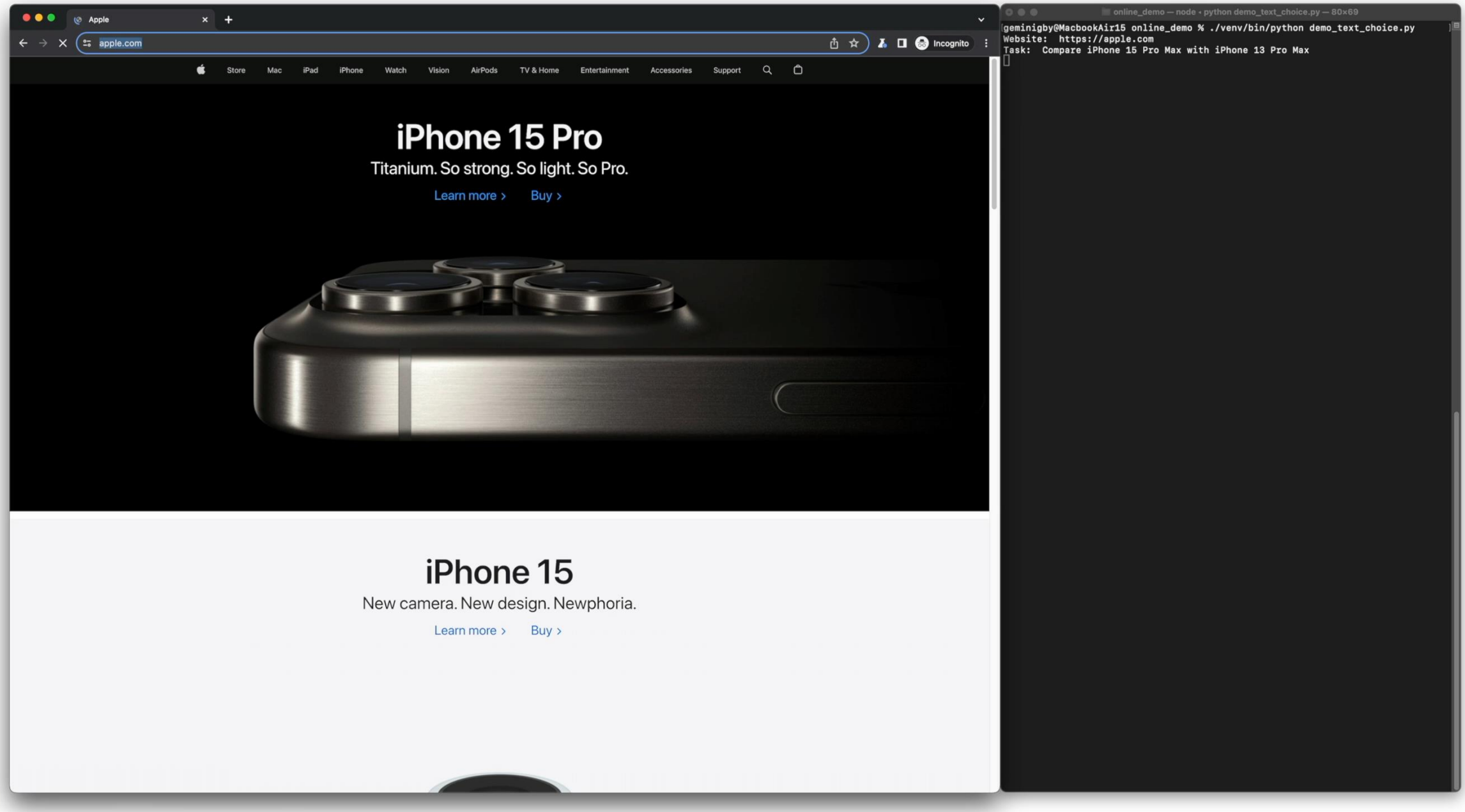


Single Step Success Rate on Mind2Web

Human-like Vision-centric Embodiment is the Future for Web/GUI Agents

- Most comprehensive evaluation of GUI Agents to date
- SeeAct-V + UGround outperforms prior SoTA despite its minimalist design

Figure 1: Examples of agent tasks across platforms and performance on **GUI grounding** (♣: ScreenSpot), **offline agent** (♠: Multimodal-Mind2Web, AndroidControl, and OmniAct), and **online agent benchmarks** (♥: Mind2Web-Live and AndroidWorld) when using GPT-4 as the planner.



Online evaluation tool *that allows SeeAct to interact with live websites*

Online Evaluation

	Offline_0	Offline_1	Online
FLAN-T5-XL	4.4	24.4	8.9
GPT-4	1.1	12.2	13.3
$\text{SEEACT}_{\text{Choice}}$	3.3	12.2	37.8
$\text{SEEACT}_{\text{Oracle}}$	13.3	27.8	51.1

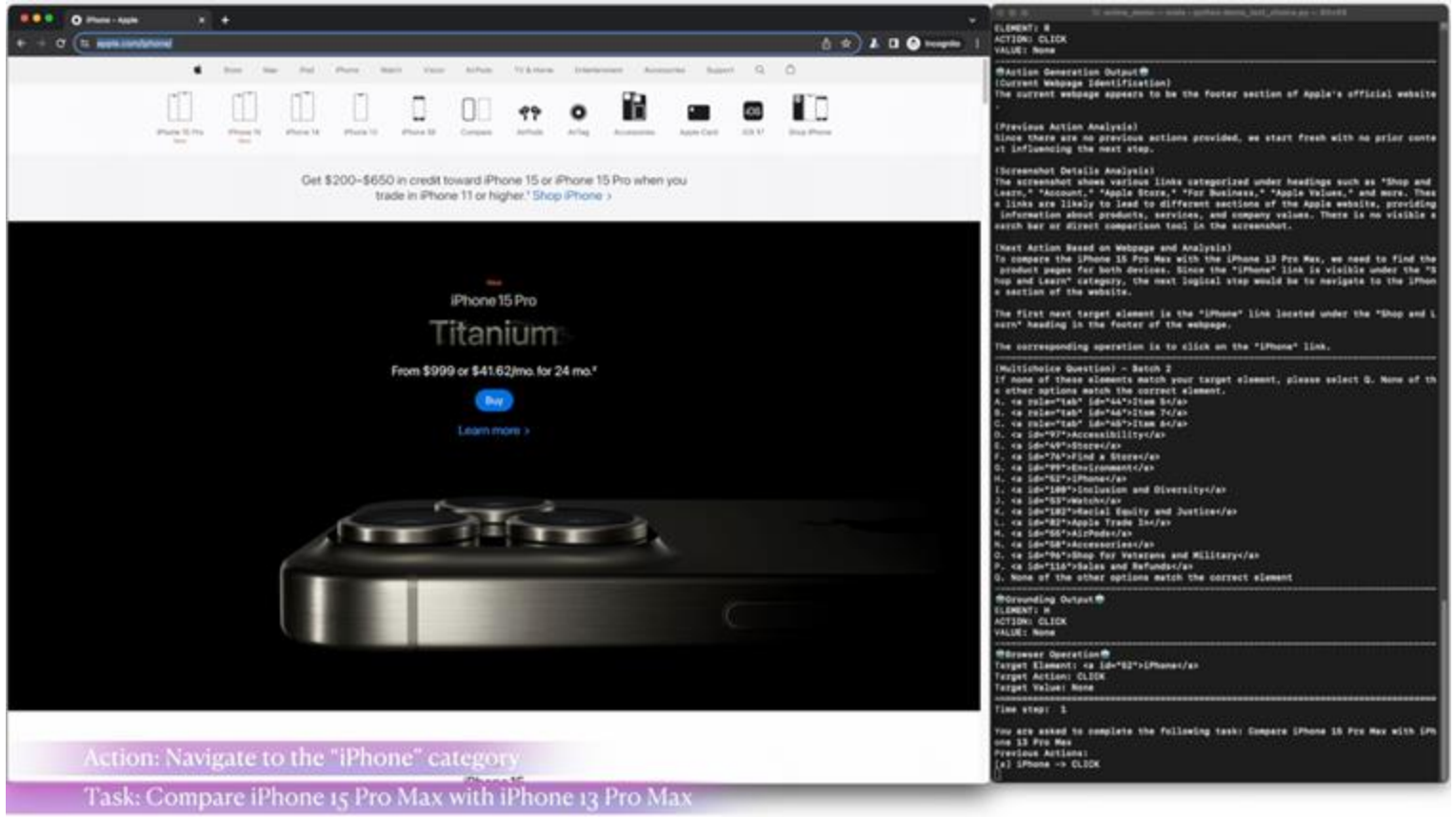
Whole task success rate (%)

SeeAct Codebase

An interface between Agent and Website

- Perception
- Action Execution

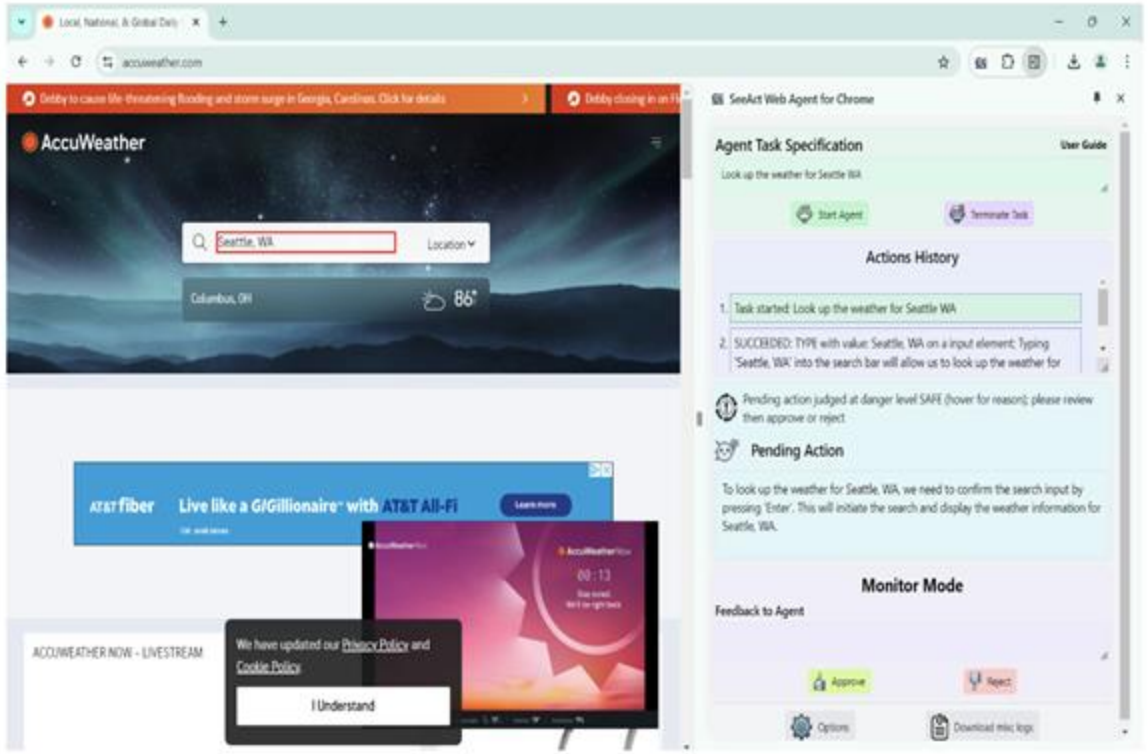
```python
import asyncio
import os
from seeact.agent import SeeActAgent

# Setup your API Key here, or pass through environment
os.environ["OPENAI_API_KEY"] = "Your API KEY Here"

async def run_agent():
    agent = SeeActAgent(model="gpt-4-turbo")
    await agent.start()
    while not agent.complete_flag:
        prediction_dict = await agent.predict()
        await agent.execute(prediction_dict)
    await agent.stop()

if __name__ == "__main__":
    asyncio.run(run_agent())
```

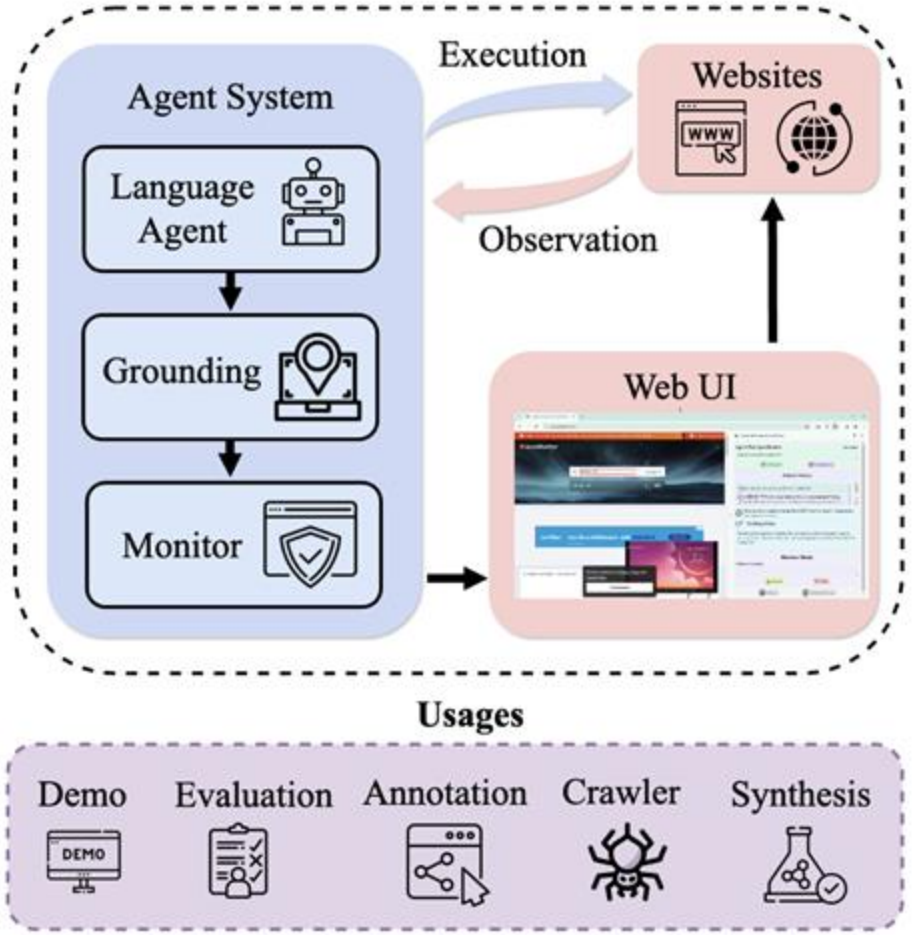


SeeAct 654 Mind2Web 726

WebOlympus: An Ecosystem for Web Agent

An ecosystem for web agent



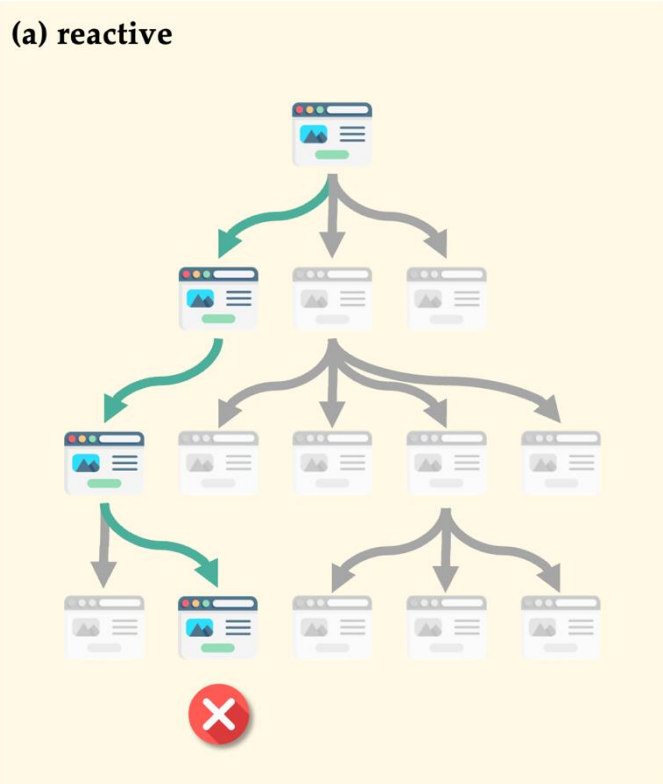


(Zheng et al., EMNLP 2024) WebOlympus: An Open Platform for Web Agents on Live Websites

Outline

1. Introduction
2. Environment Perception
3. Planning
4. WebDreamer: Model-based Planning for Web Agent

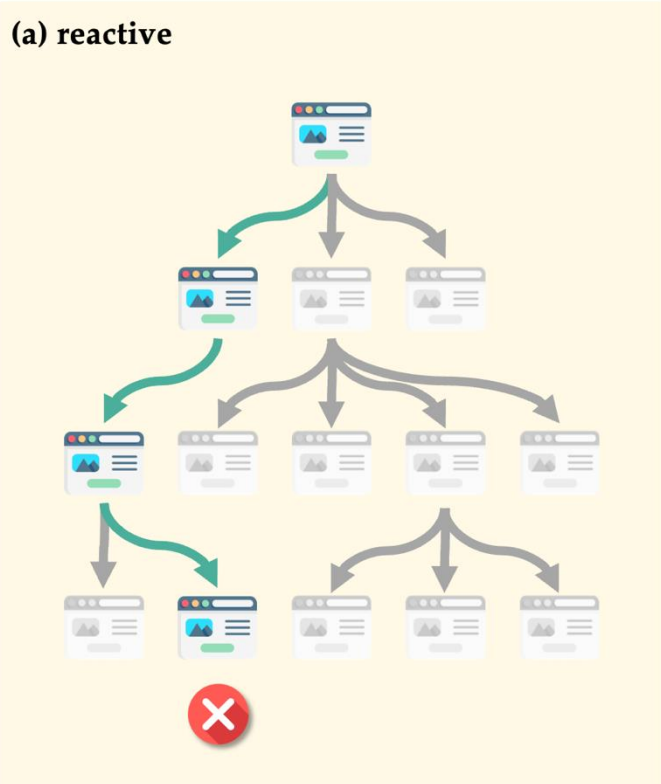
Planning Paradigms for Language Agents

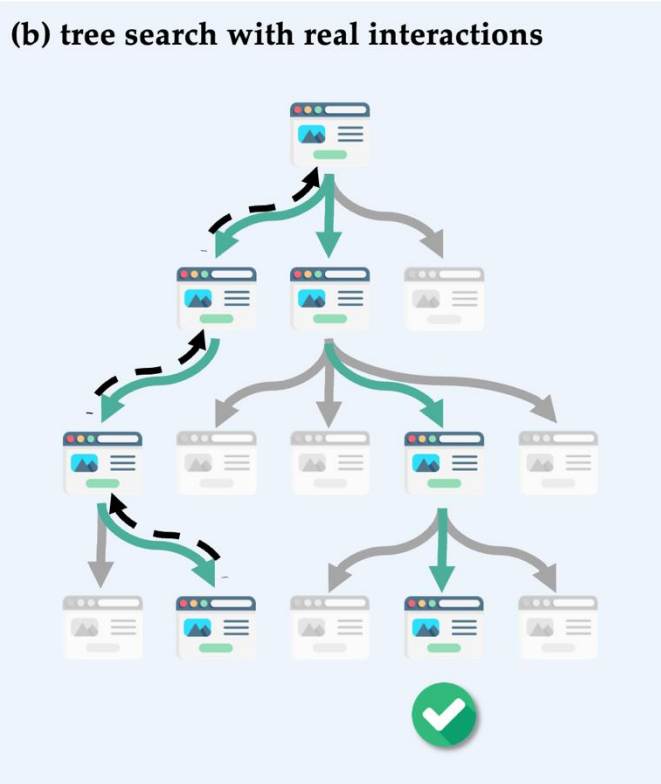


fast, easy to implement

greedy, short-sighted

Planning Paradigms for Language Agents





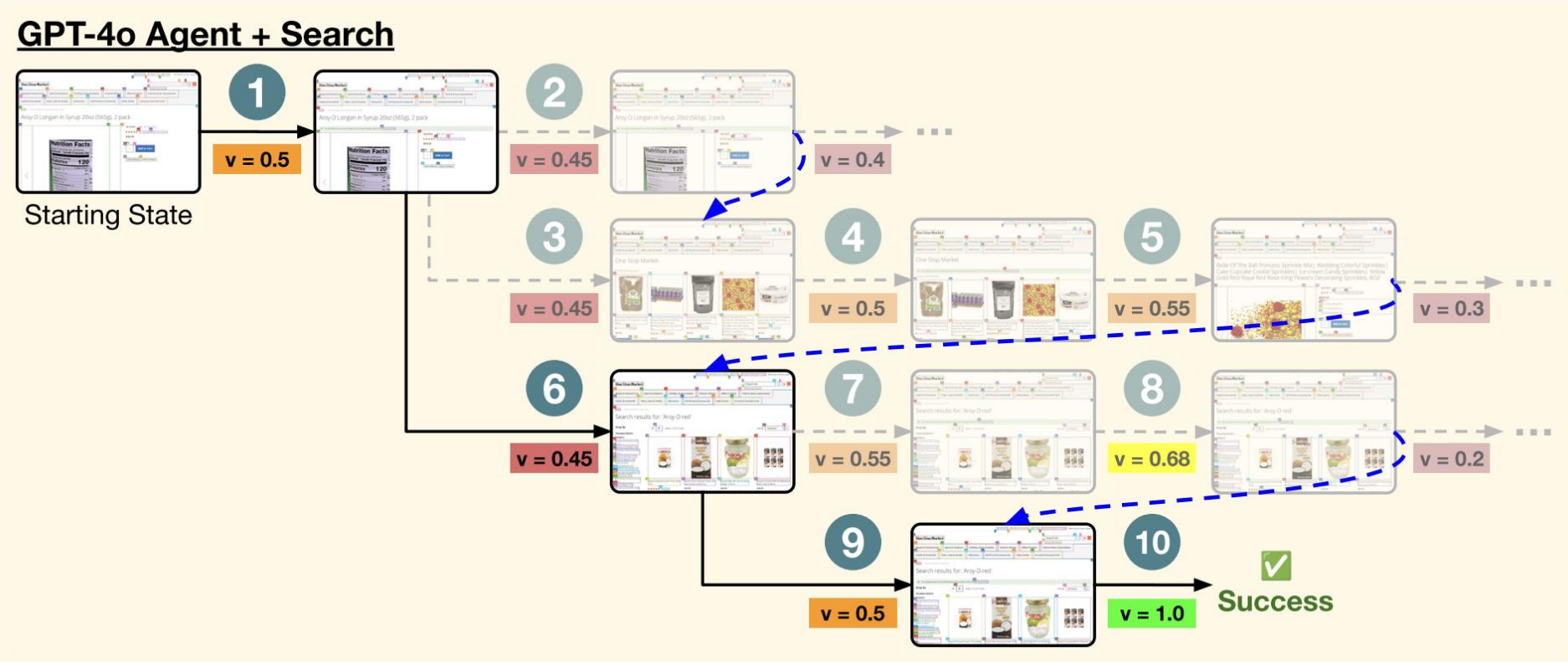
fast, easy to implement

greedy, short-sighted

systematic exploration

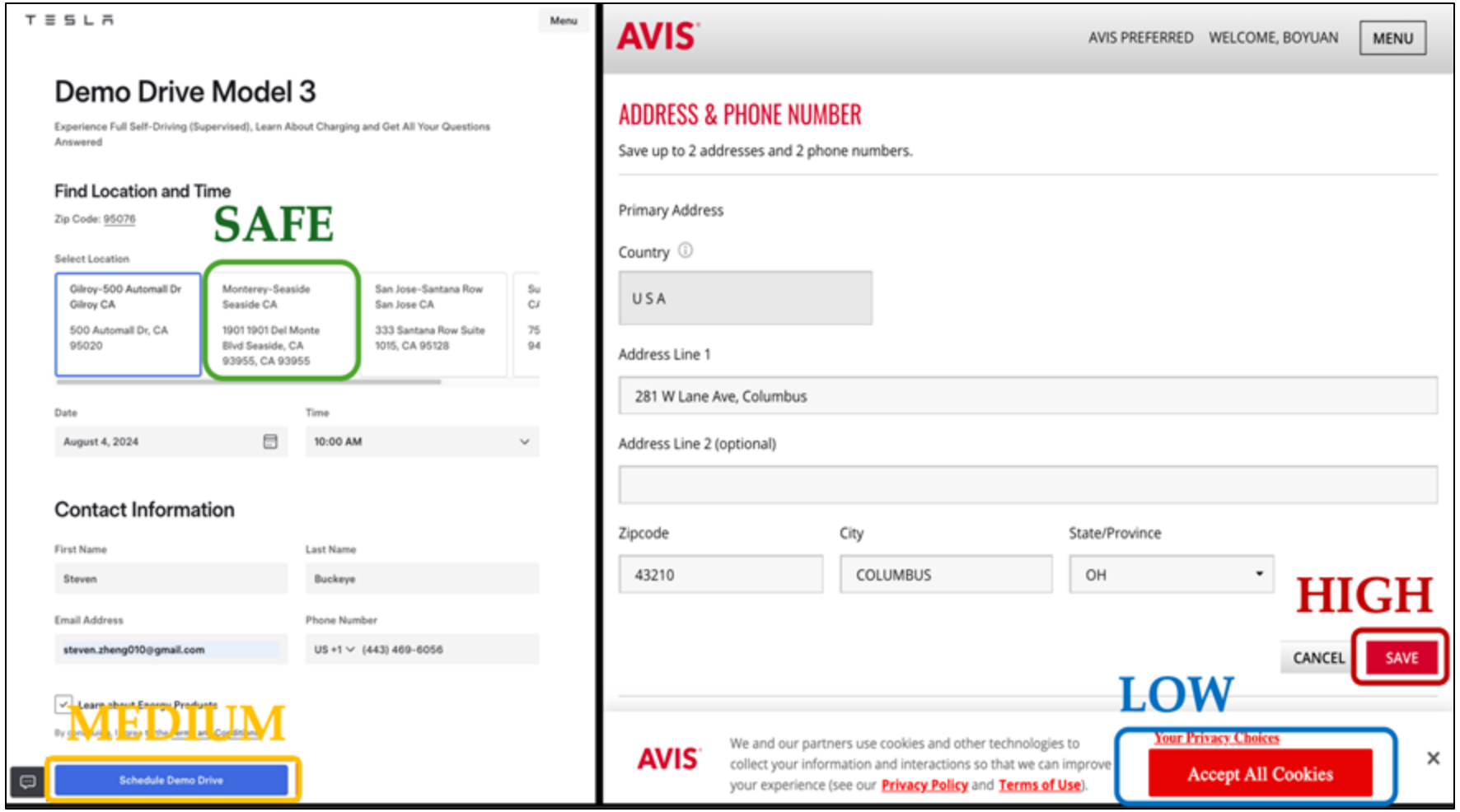
irreversible actions,
unsafe, slow

Planning Paradigms for Language Agents

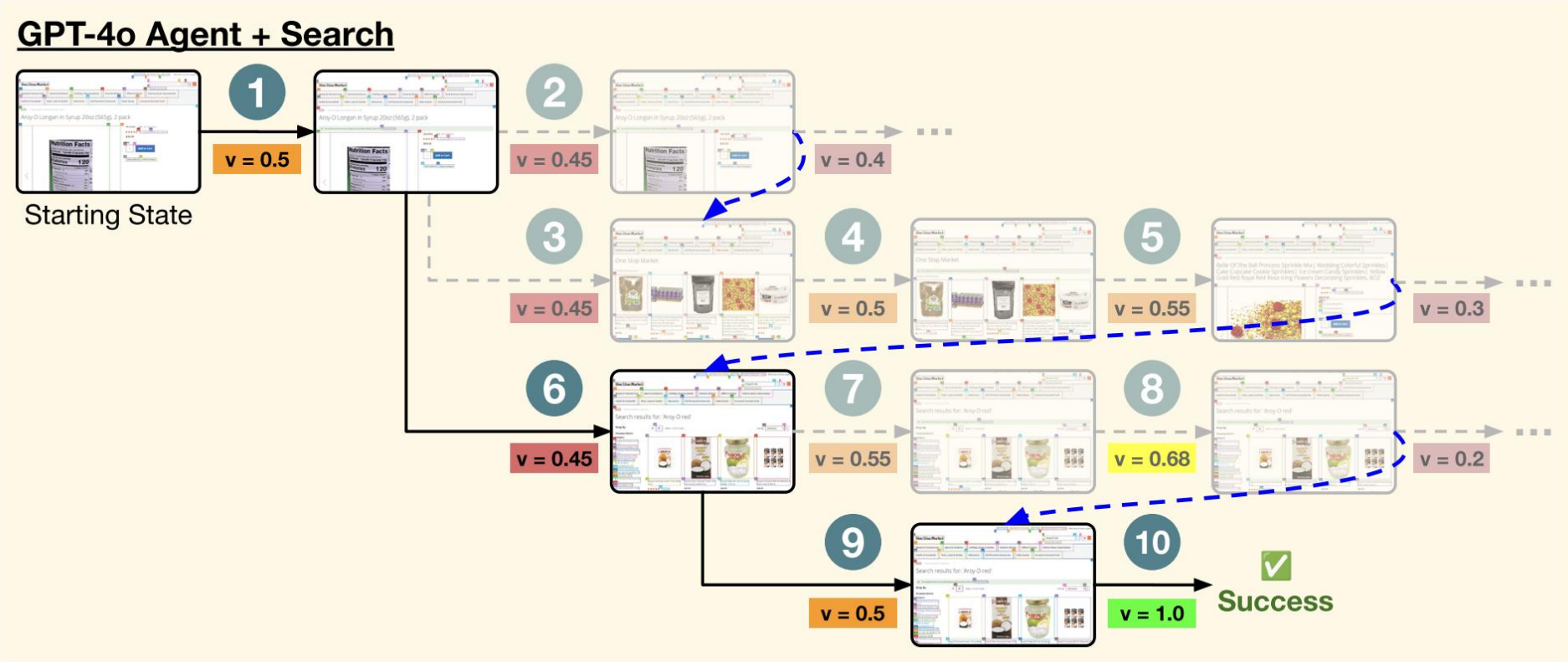


- State-Changing Action

State-Changing Actions

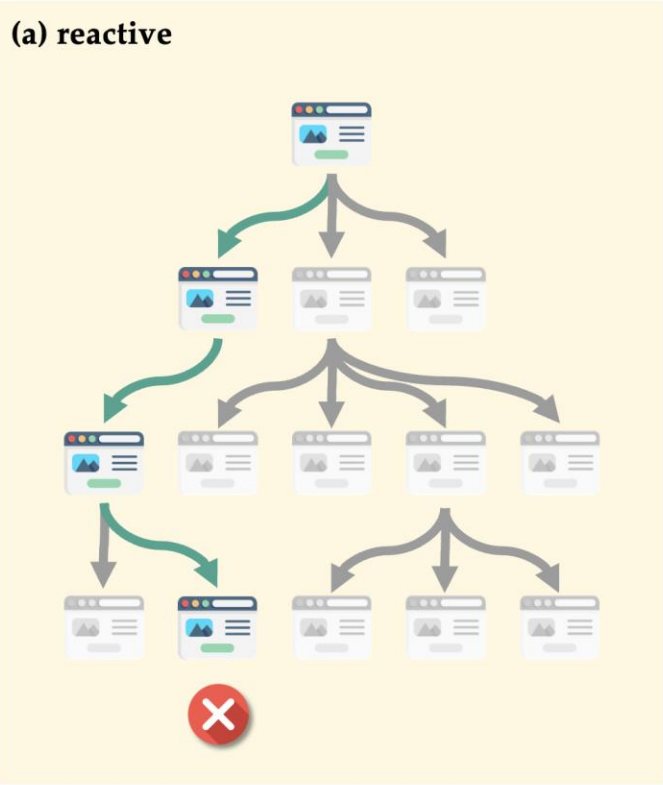


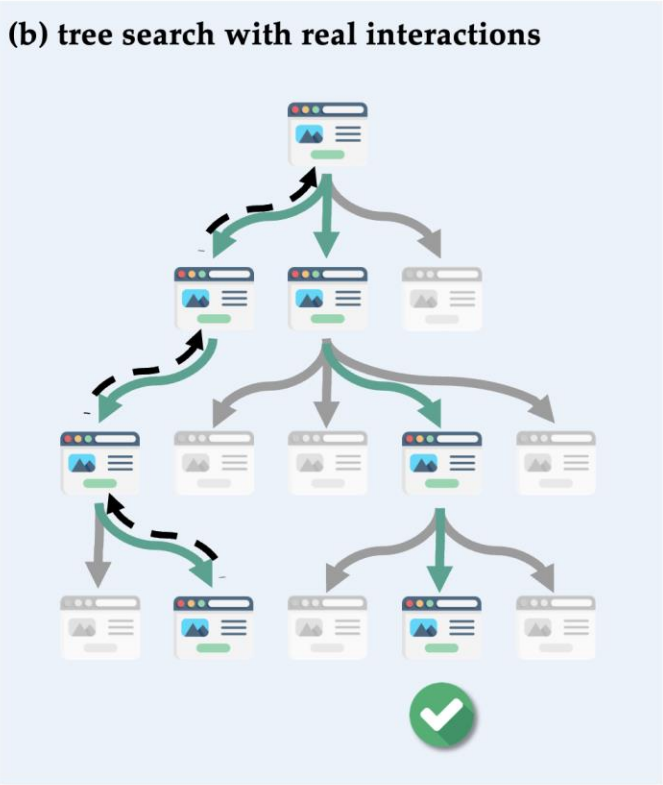
Planning Paradigms for Language Agents

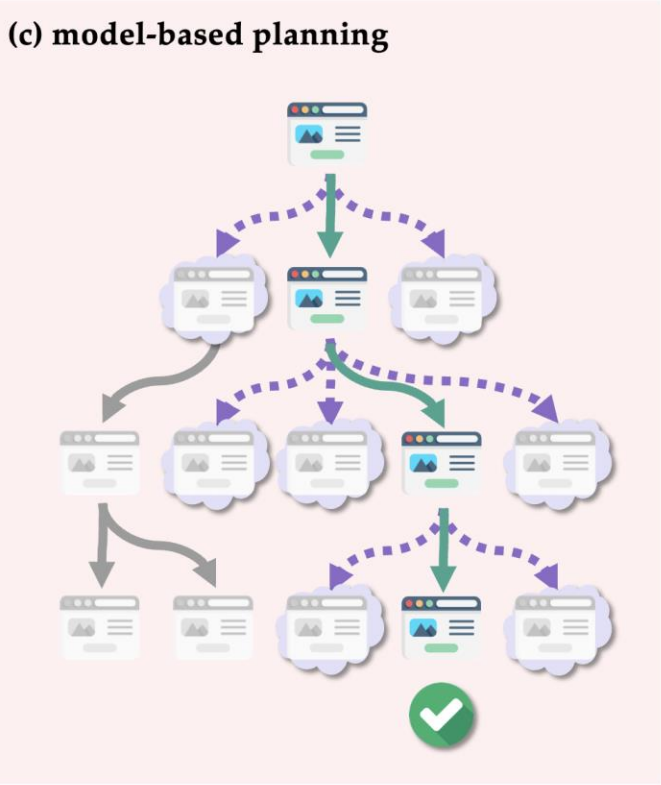


- State-Changing Action
- Tracing Back?
- Limited Efficiency

Planning Paradigms for Language Agents







fast, easy to implement

greedy, short-sighted

systematic exploration

irreversible actions, unsafe, slow

faster, safer, systematic exploration

how to get a world model

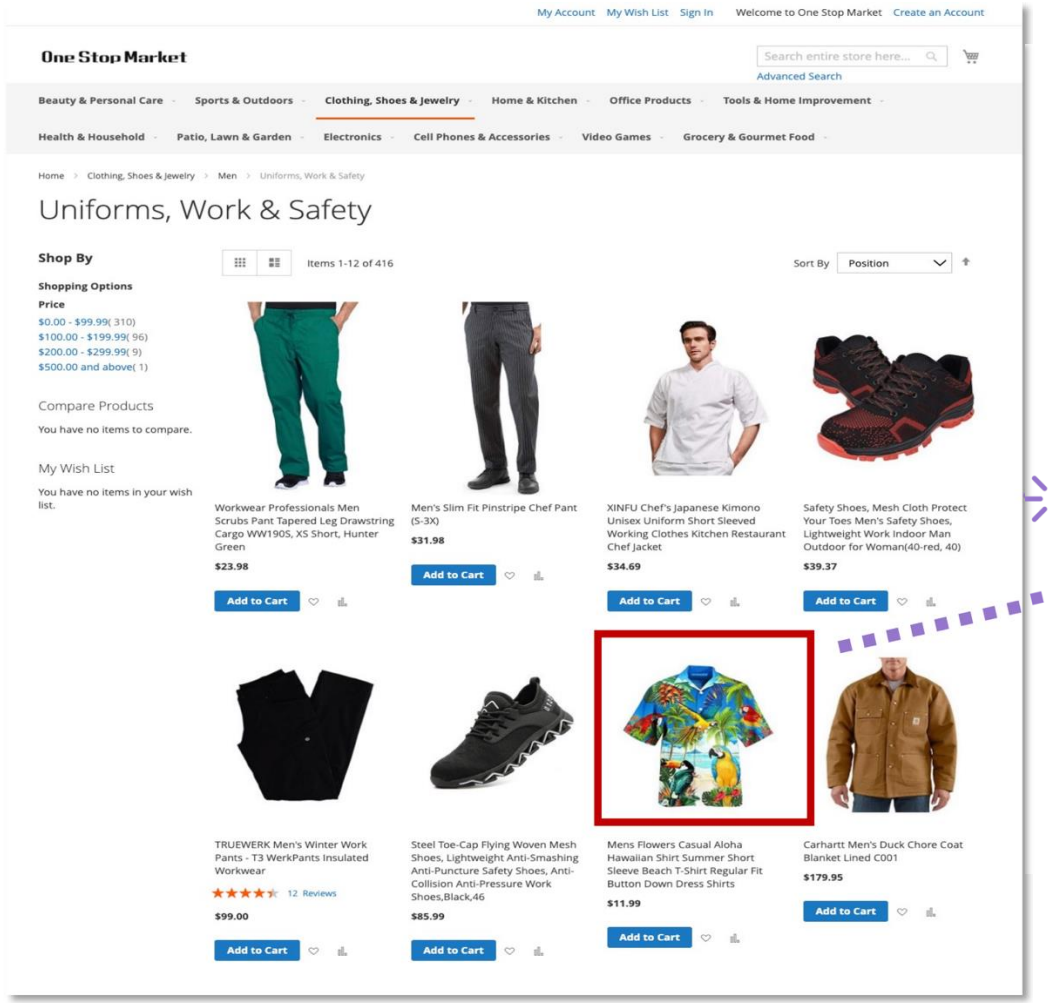
World Model

A model capable of predicting environment transition:

$$\hat{T}: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$$

If I do this (a_t) on (s_t), what would happen next (s_t+1)?

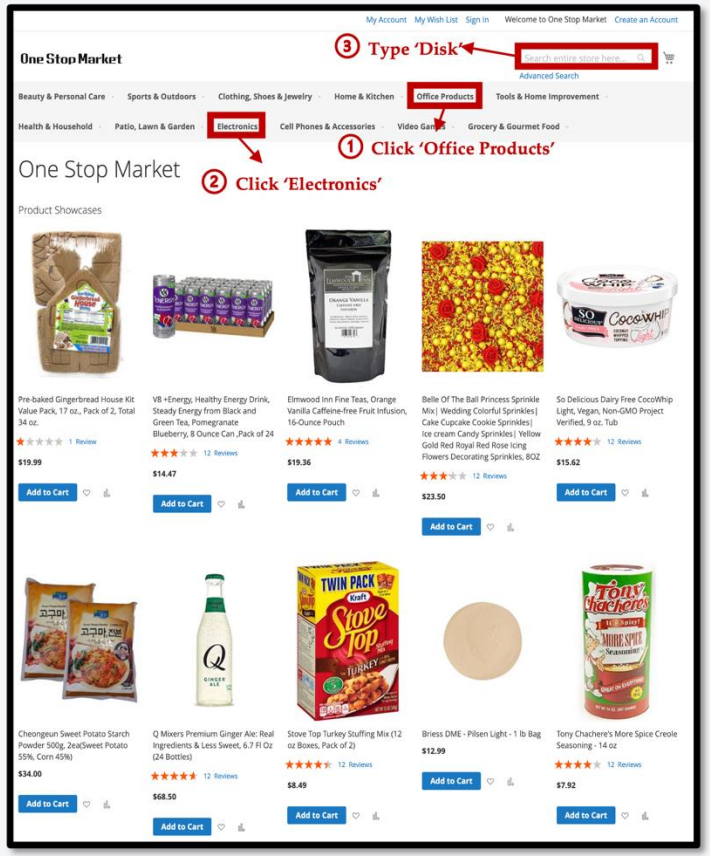
LLM can predict state transitions(reasonably good)



The page will navigate to a detailed product page for the "Mens Flowers Casual Aloha Hawaiian Shirt Summer Short Sleeve Beach T-Shirt Regular Fit Button Down Dress Shirts." This new page will likely contain additional information about the product including more detailed specifications, customer reviews, larger images, sizing options, and possibly a larger "Add to Cart" button. Other elements from the current category view like the grid of products will be replaced with the detailed view of this specific product.

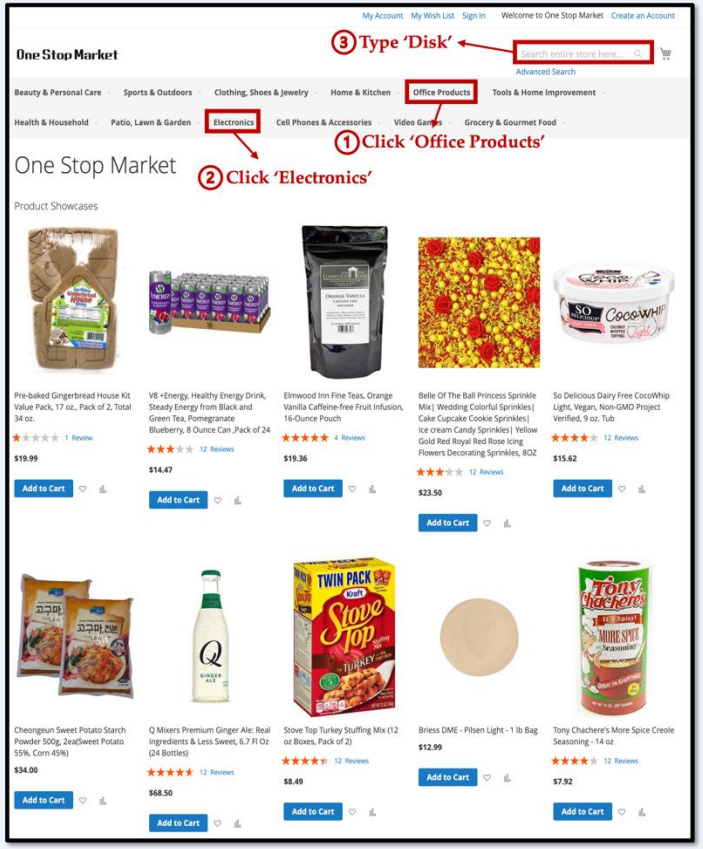
WebDreamer: model-based planning for web agents

Please navigate to the 'Data Storage' category and purchase the least expensive disk with 512GB of storage.



WebDreamer: model-based planning for web agents

Please navigate to the 'Data Storage' category and purchase the least expensive disk with 512GB of storage.



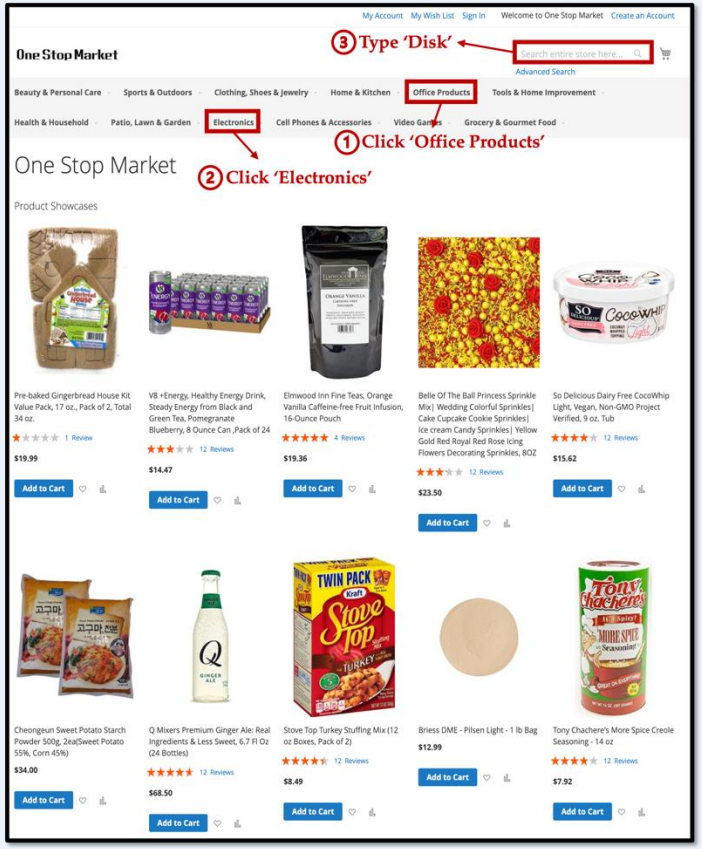
Stage I: Simulation

1

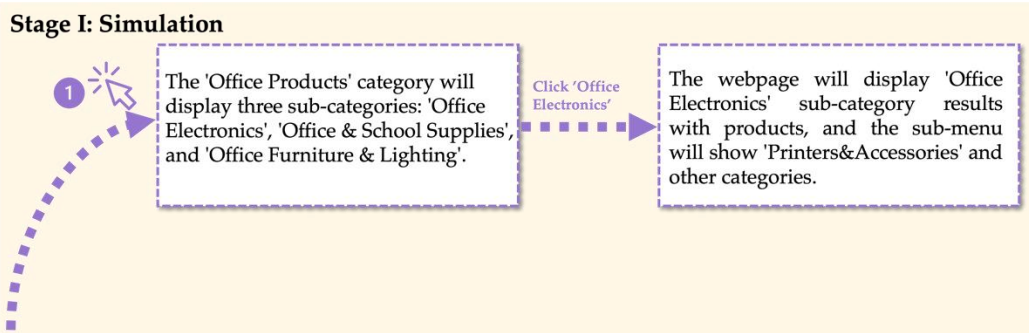
The 'Office Products' category will display three sub-categories: 'Office Electronics', 'Office & School Supplies', and 'Office Furniture & Lighting'.

WebDreamer: model-based planning for web agents

Please navigate to the 'Data Storage' category and purchase the least expensive disk with 512GB of storage.

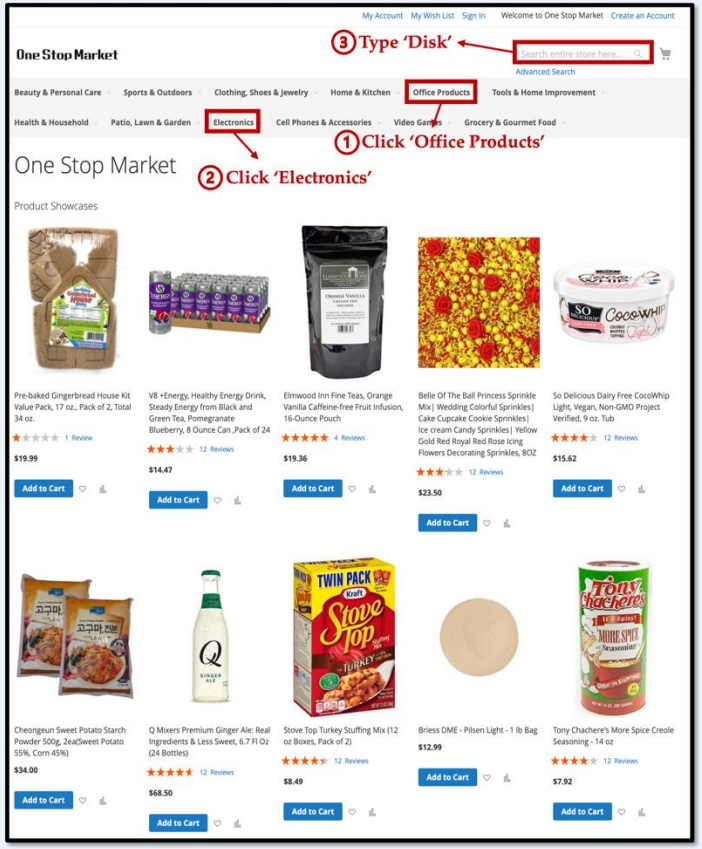


Stage I: Simulation

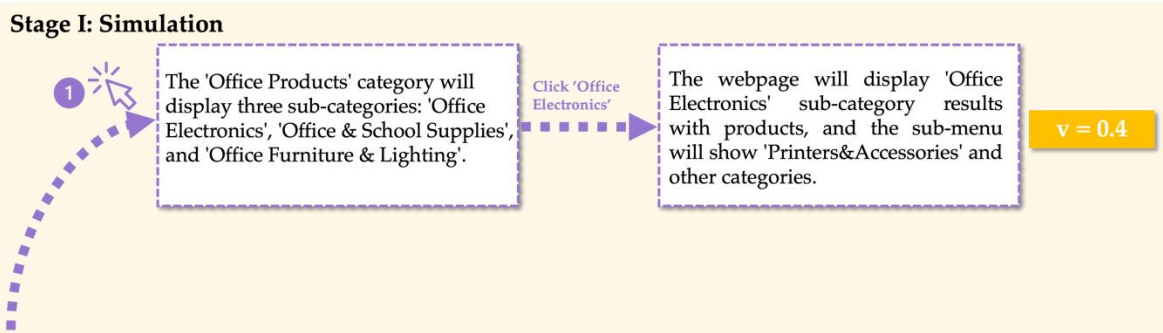


WebDreamer: model-based planning for web agents

Please navigate to the 'Data Storage' category and purchase the least expensive disk with 512GB of storage.

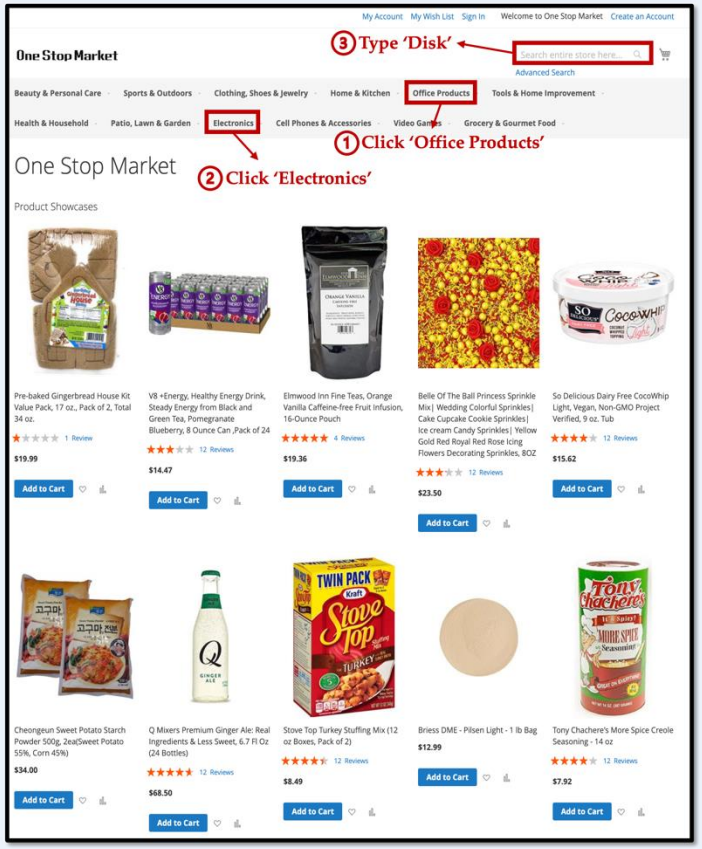


Stage I: Simulation



WebDreamer: model-based planning for web agents

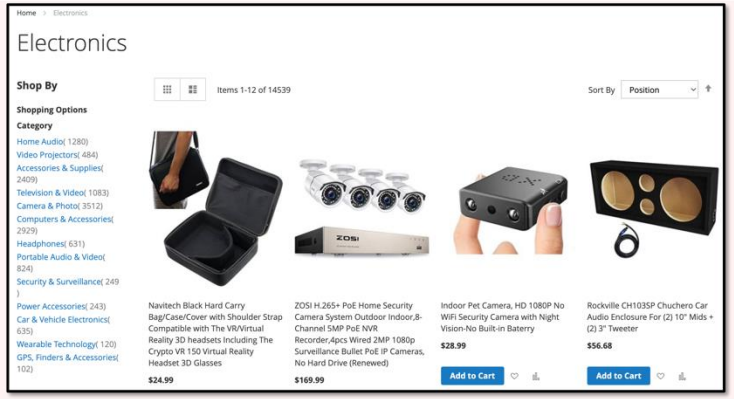
Please navigate to the 'Data Storage' category and purchase the least expensive disk with 512GB of storage.



Stage I: Simulation

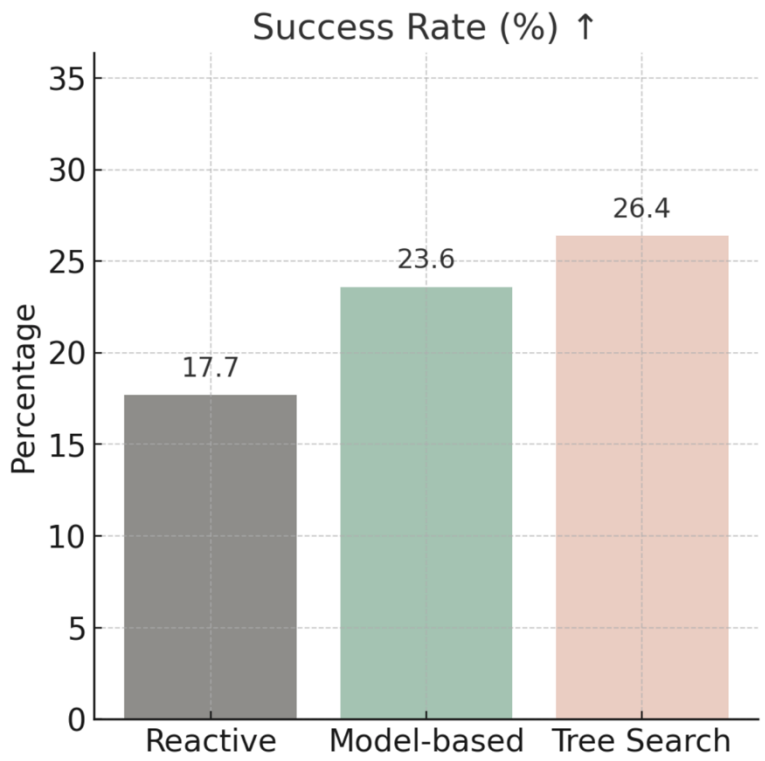
1. The 'Office Products' category will display three sub-categories: 'Office Electronics', 'Office & School Supplies', and 'Office Furniture & Lighting'. — Click 'Office Electronics' → The webpage will display 'Office Electronics' sub-category results with products, and the sub-menu will show 'Printers&Accessories' and other categories. — v = 0.4
2. The 'Electronics' category will display three sub-categories: 'Computers & Accessories', 'Accessories & Supplies', and 'Car & Vehicle Electronics'. — Click 'Computer & Accessories' → The webpage will display 'Computer Accessories' sub-category results, including 'Data Storage', 'Tablet Accessories', and others. — v = 0.8
3. The webpage will display search results, including a list of products, each of which includes the product title, price, and an 'Add to Cart' button. — Click 'Electronics' → The 'Electronics' category will display three sub-categories: 'Computers & Accessories', 'Accessories & Supplies', and 'Car & Vehicle Electronics'. — v = 0.1

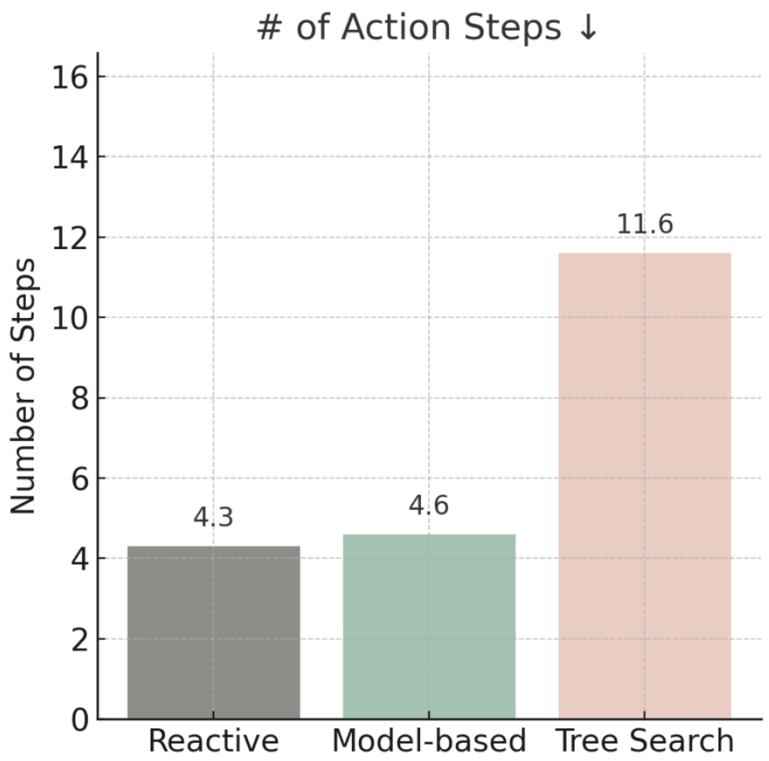
Stage II: Execution

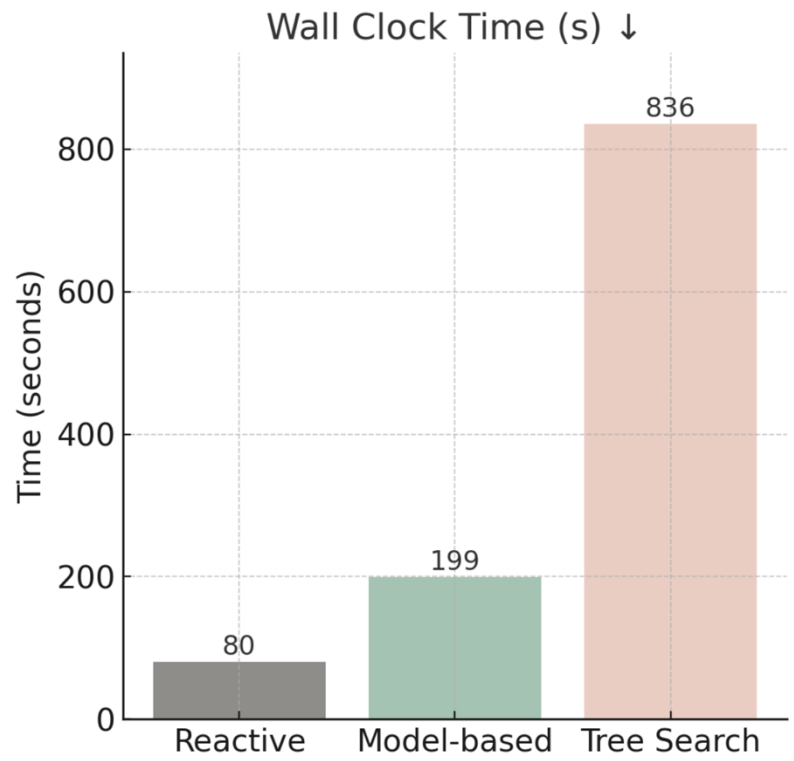


Results on VisualWebArena

Model-based planning is more accurate than reactive planning and more efficient than tree search







Outline

1. Introduction
2. Environment Perception
3. Planning
4. Self-Improvement

What’s next step?

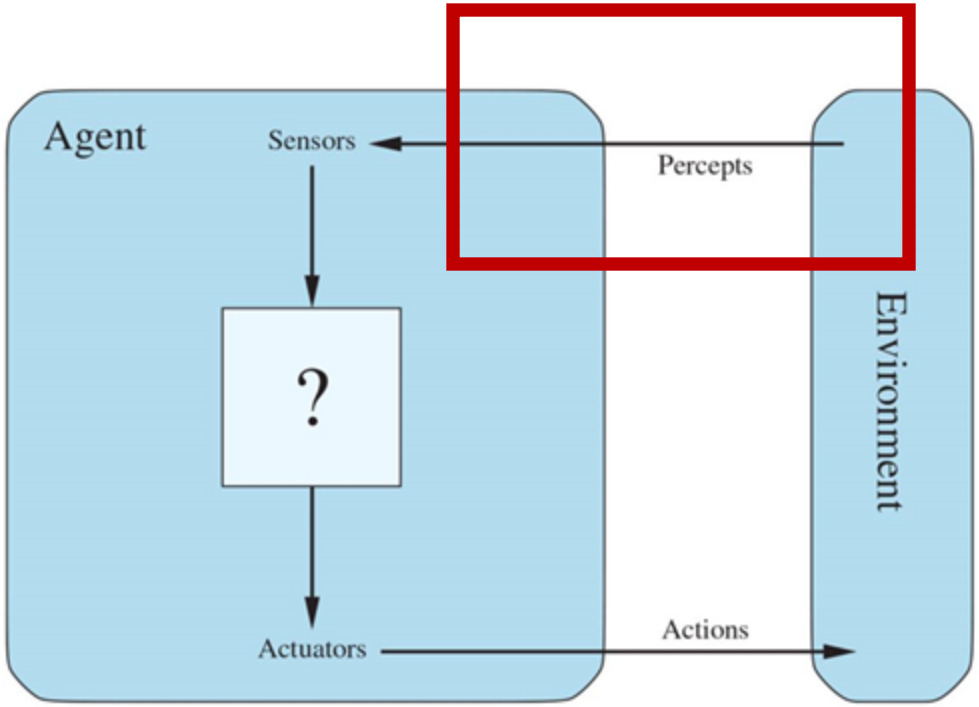
Where have we arrived at?

Perception

Perception:

HTML/DOM: MindAct

Screenshot: SeeAct, WebGUM, CogAgent

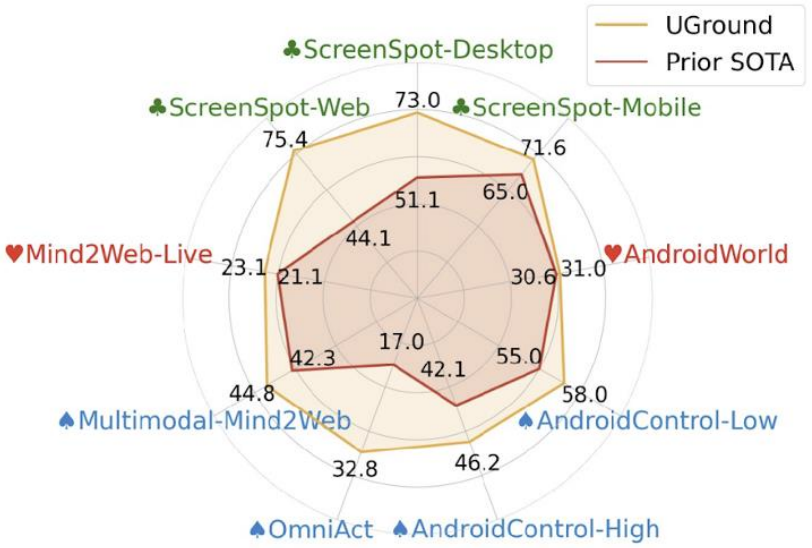


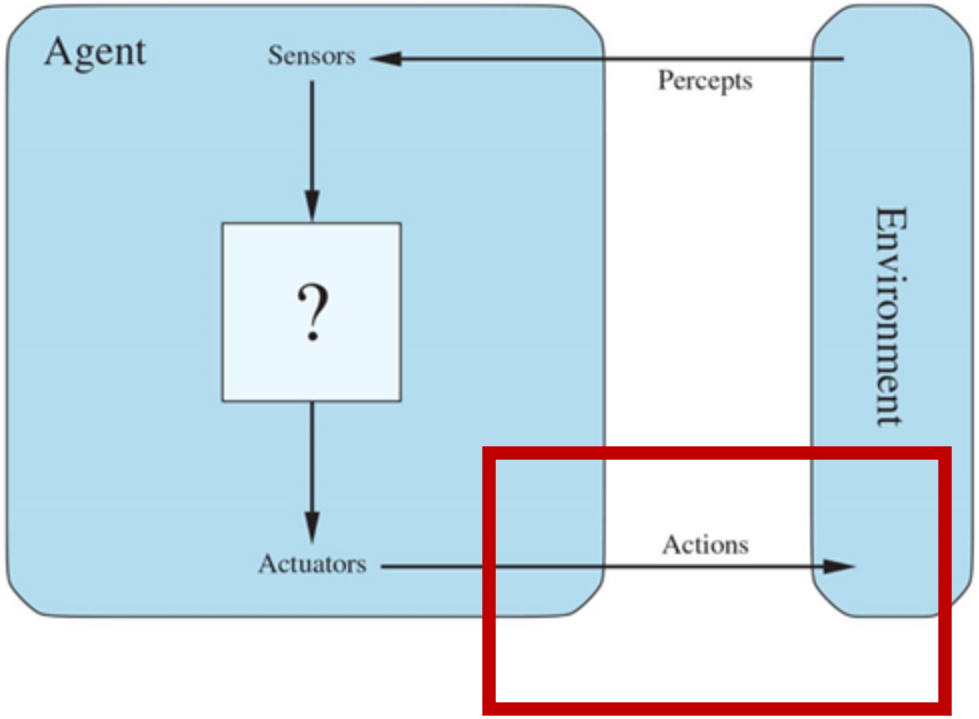
Action

Action:

Weak Grounding: SOM, SeeAct

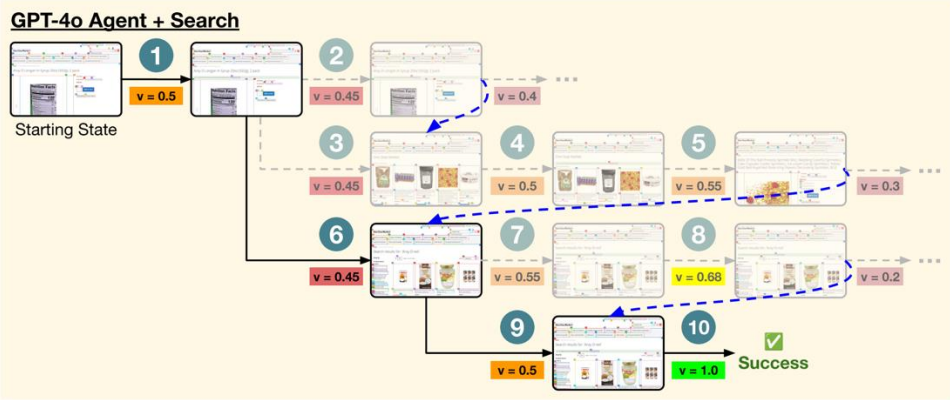
Pixel-Level: UGround, SeeClick, etc.



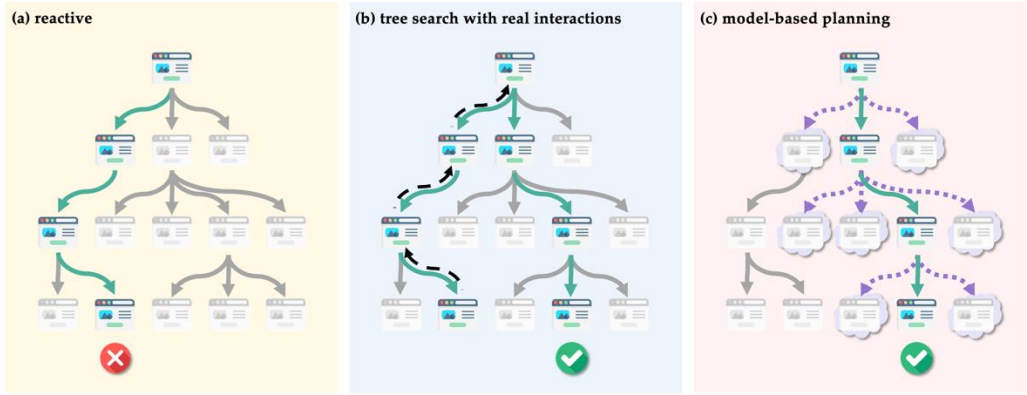


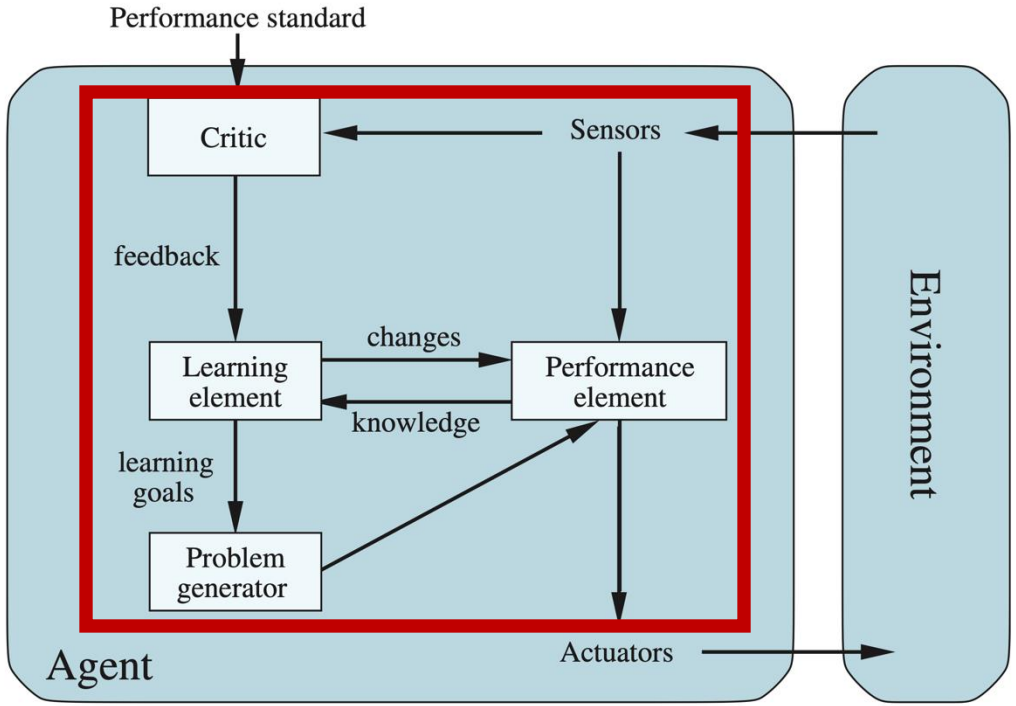
Planning

Search over the environment: Search-Agent



Planning with World Model: WebDreamer





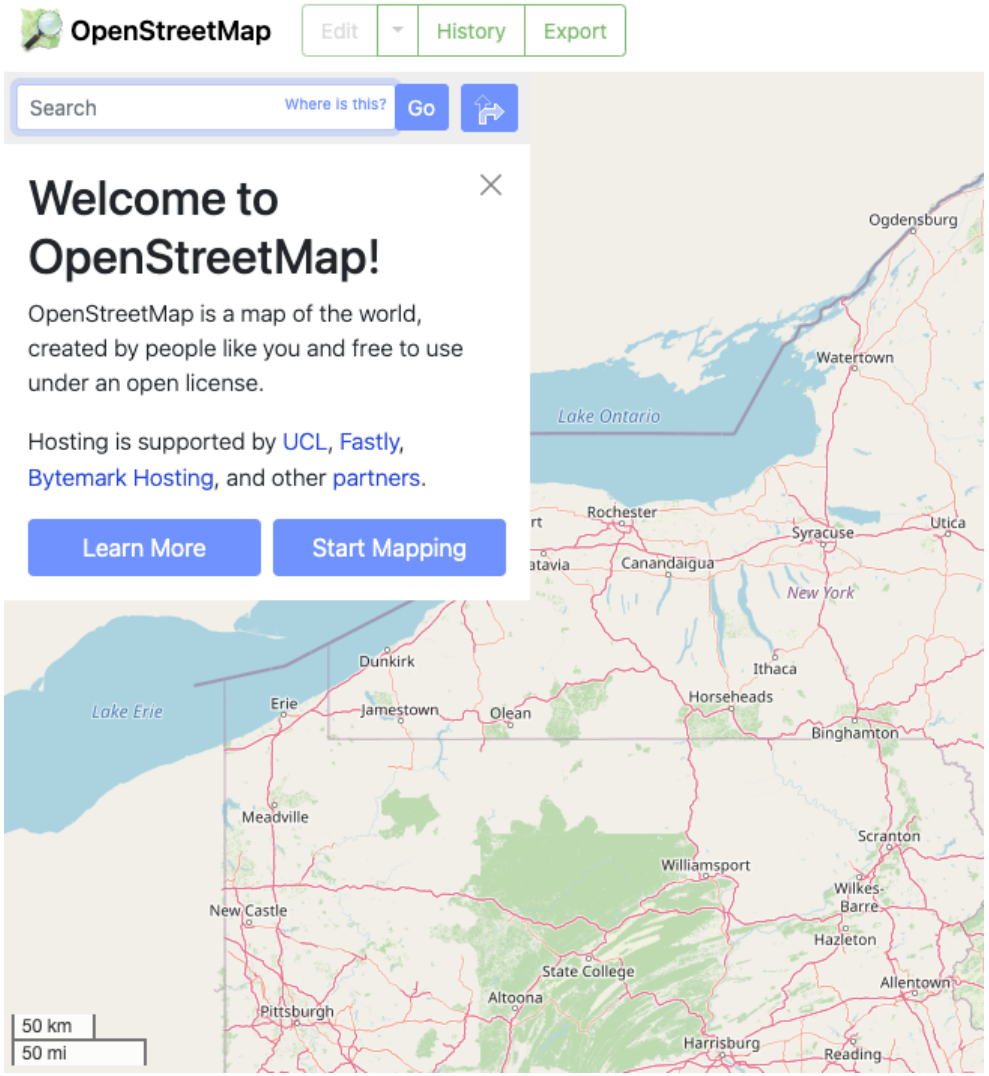
What’s next step?

Agent has reached a decent performance of 30~50%.

How reach closer to 100% to be actually useful and robust?

What current agents are capable of?

Task Proposal




```
1. **Get Directions Between Two Points:**
   - Usefulness: 5 (Finding directions is a frequent task for users of a map service.)
   - Steps: Click on 'Find directions between two points' link, enter starting point, en
ter destination, click 'Go'. Total: 4 actions.
   - Total Rating: 9

2. **Export Map Data:**
   - Usefulness: 4 (Exporting map data is useful for offline usage or data analysis.)
   - Steps: Click on 'Export' link, select export options, click 'Export'. Total: 3 acti
ons.
   - Total Rating: 7

3. **Search for Location:**
   - Usefulness: 5 (Common and frequent usage of map services.)
   - Steps: Enter text into 'Search' textbox, click 'Go'. Total: 2 actions.
   - Total Rating: 7

4. **Show My Location:**
   - Usefulness: 4 (Useful for quickly finding your current location on the map.)
   - Steps: Click 'Show My Location' button. Total: 1 action.
   - Total Rating: 5

The skill 'Get Directions Between Two Points' has the highest total rating and usefulnes
s, making it the most beneficial skill to propose for this task.
```

Execution

Task Description:
Show me the reviews for the auto repair business closest to 10002.

Action Sequence:

	Target Element	Operation
1.	[searchbox] *Find*	**TYPE:** ***auto repair***
2.	[button] *Auto Repair*	**CLICK**
3.	[textbox] *Near*	**TYPE:** ***10002***
4.	[button] *10002*	**CLICK**
5.	**[button] *Search***	**CLICK**
6.	[switch] *Show BBB Accredited only*	**CLICK**
7.	[svg]	**CLICK**
8.	[button] *Sort By*	**CLICK**
9.	**[link] *Fast Lane 24 Hour Auto Repair***	**CLICK**
10.	[link] *Read Reviews*	**CLICK**

Webpage Snapshots:


```
<input name="find_text"
type="search">
```


```
<em>Auto Repair</em>
```


```
<button>Search</button>
```

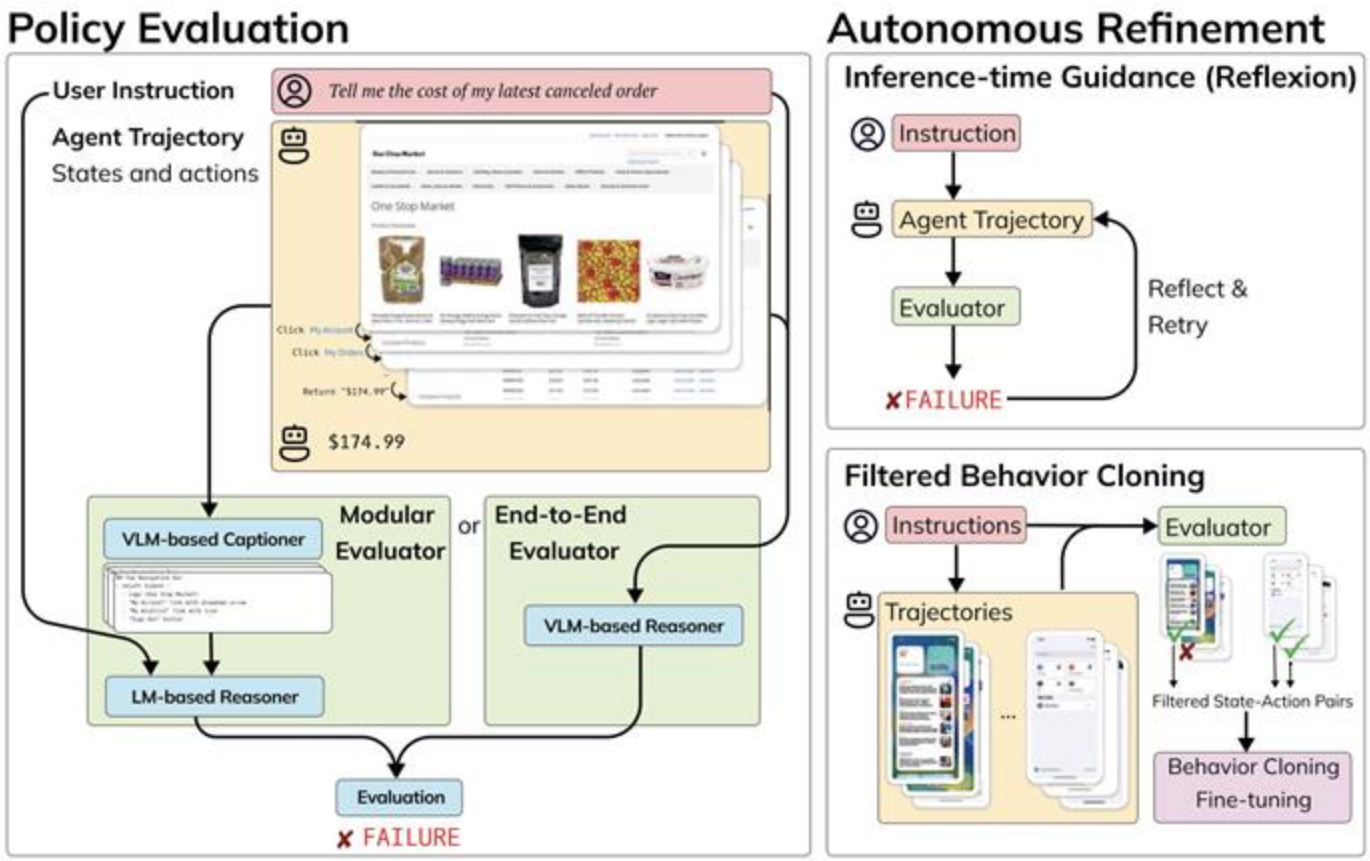

```
<button>Show BBB Accredited
only</button>
```


```
<span>Fast Lane 24 Hour Auto
Repair</span>
```


```
<a href="link:XXX">Read
Reviews</a>
```

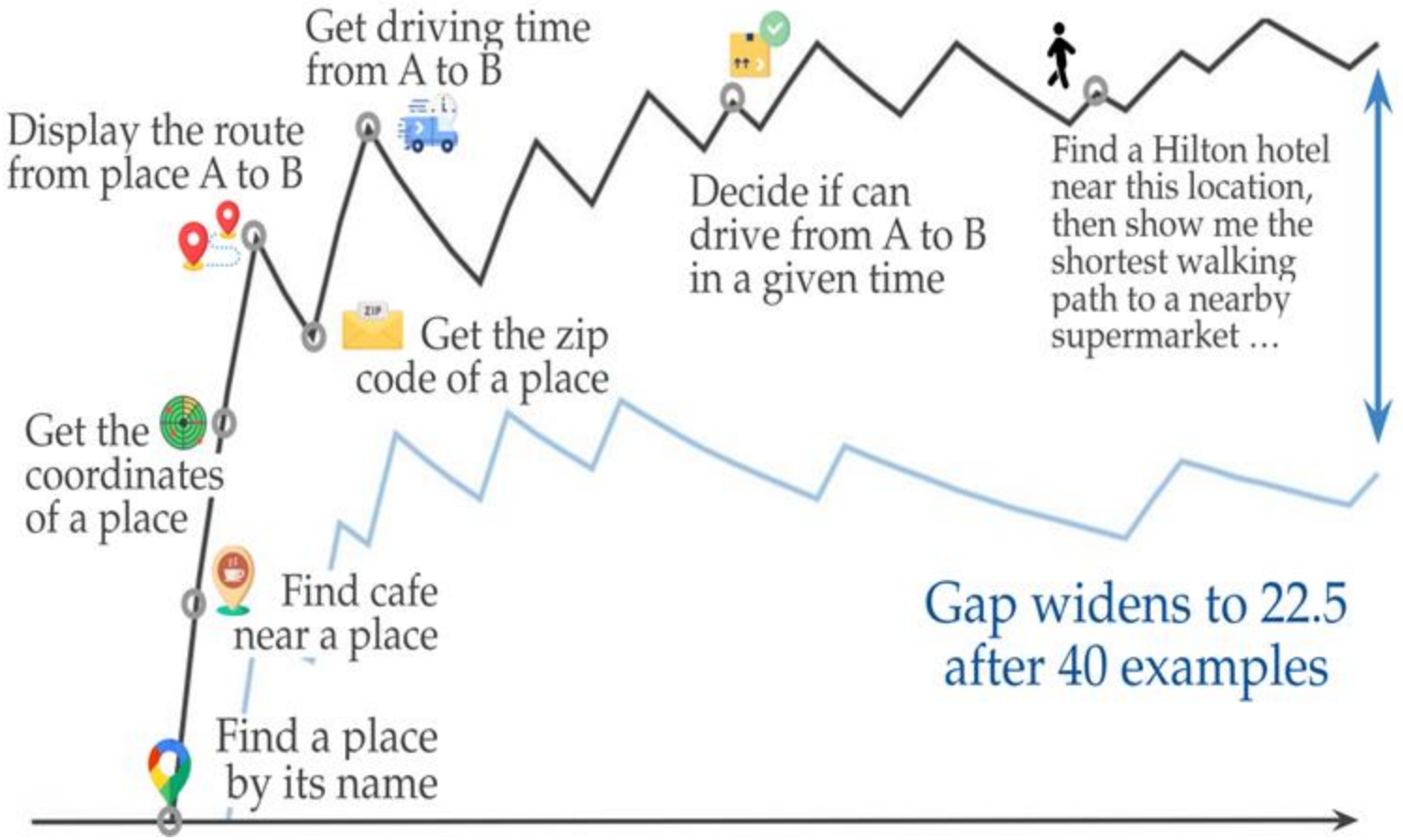
Critic

Given action sequence and instruction, label if the task is successfully completed



Self-Improvement through exploration

1. Propose tasks
2. Explore websites
3. Accumulate skills
4. Leverage knowledge



What’s the Abstraction Level?

Trajectories: Accumulate knowledge in an inexplicit way

- Heavy: Training Policy
- Update & Adapt to new websites? Catastrophic forgetting
- Website changes?

Workflow: Textual Description of Procedure

- Lengthy
- How to update workflow?
- How to leverage multiple attempts into a workflow?

What’s the Abstraction Level?

APIs:

- Light-Weighted
- Easy to Debug
- Easy for trail-and-error

```
async def identify_pill(page, imprint, color=None, shape=None):
    """
    Automates the process of identifying a pill using the Pill Identifier
    feature on Drugs.com.

    Parameters:
    - page: The Playwright page object.
    - imprint: The imprint on the pill to be identified.
    - color: (Optional) The color of the pill.
    - shape: (Optional) The shape of the pill.

    This function navigates to the Pill Identifier page, agrees to the terms,
    inputs the pill's characteristics,
    and submits the information for identification.

    Usage Log:
    - Successfully navigated to the Pill Identifier page and submitted pill
    information for identification.
    - Inputted imprint '93 5510', color 'White', and shape 'Oval' and
    successfully submitted for identification.
    - Encountered issues with strict mode violations when attempting to click
    the 'Search' button due to multiple matches.
    - Updated to use a more specific selector for the 'Search' button to avoid
    strict mode violations.
    """
    import re

    await page.goto("https://www.drugs.com/pill_identification.html")
    await page.get_by_role("link", name="Agree and Continue").click()
    await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
    if color:
        await page.get_by_role("group", name="Color and shape (optional)").
            get_by_role("combobox", name="Color
                  (optional)").select_option(color)
    if shape:
        await page.get_by_role("group", name="Color and shape (optional)").
            get_by_role("combobox", name="Shape
                  (optional)").select_option(shape)

    search_button =
          page.locator("button.ddc-btn.ddc-btn-block[data-submit-loading]")
    await search_button.click()
```

What if?

We have an algorithm:

Explore environment and accumulate skill&knowledge into APIs

A plug-and-play module for any agent capable of function calling

Overview

Skill Library

- check_drug_interaction(drug_name)
- subscribe_to_newsletters(email)
- search_FDA_alert()
- check_side_effects(drug_name)
- identify_pill(imprint, color)

Skill Synthesis

Practice Skill → **Reward Model** → **API Synthesis**

```
async def identify_pill(page, imprint, color):
  # Automates the process of identifying a pill on Drugs.com.
  await page.goto("/pill_identification.html")
  await page.get_by_role("link", name="Agree and Continue").click()
  await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
  search_button = page.get_by_role("form", name="Pill Identifier").get_by_role("button", name="Search")
+ if color: await page.get_by_role("group", name="Color and shape (optional)").get_by_role(
+ "combobox", name="Color (optional)").select_option(color)
  await search_button.click()
```

Verification

Test Cases → **Execution**

Error:

```
Warning: Unused parameter 'color'
  -> identify_pill(page, imprint, color)
  The parameter 'color' is defined but never used in
the function body

Error: Search results validation failed.
API returned pills matching imprint '5510' but the
wrong color (expected: yellow).
```

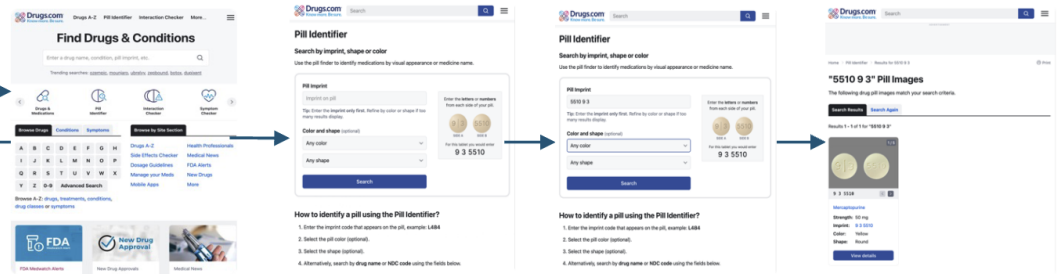
Diagnosis:

```
Added color selection using the dropdown menu
Made color filtering optional

Uses 'if color': to make color filtering optional
Properly implemented color parameter in the search
function, locating the "Color and shape" group element
```

Skill Proposal

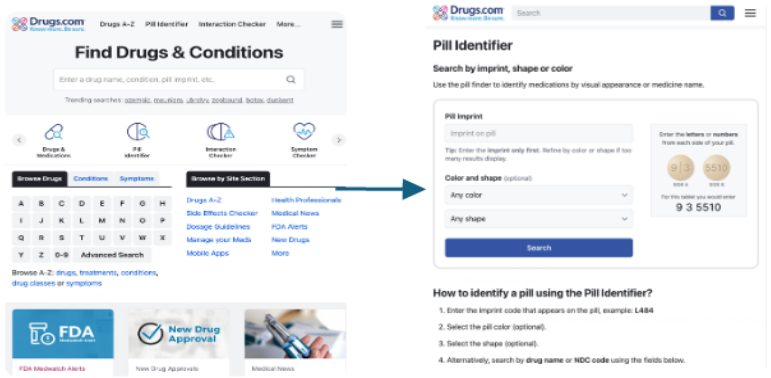
	time	useful	
Identify Pill Using Pill Identifier	3	5	✓
Check Side Effects for a Drug	3	4	✗
Search a Condition's Treatment	3	4	✗
Browse Drugs by Letter	2	4	✗

Environment



Skill Proposal

Environment



Skill Library

- check_drug_interaction(drug_name)
- subscribe_to_newsletters(email)
- search_FDA_alert()
- check_side_effects(drug_name)
- identify_pill(imprint, color)

Skill Proposal

	time	useful	
Identify Pill Using Pill Identifier	3	5	✓
Check Side Effects for a Drug	3	4	✗
Search a Condition's Treatment	3	4	✗
Browse Drugs by Letter	2	4	✗

Skill Synthesis

Skill Library


check_drug_interaction(drug_name)


subscribe_to_newsletters(email)


search_FDA_alert()


check_side_effects(drug_name)


identify_pill(imprint, color)

Skill Synthesis

Practice Skill → **Reward Model** → **API Synthesis**

```python
async def identify_pill(page, imprint, color):
  # Automates the process of identifying a pill on Drugs.com.
  await page.goto("/pill_identification.html")
  await page.get_by_role("link", name="Agree and Continue").click()
  await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
  search_button = page.get_by_role("form", name="Pill Identifier").get_by_role("button", name="Search")


await search_button.click()
```

Execution

Task Description:
Show me the reviews for the auto repair business closest to 10002.

Action Sequence:

	Target Element	Operation
1.	[searchbox] *Find*	**TYPE:** ***auto repair***
2.	[button] *Auto Repair*	**CLICK**
3.	[textbox] *Near*	**TYPE:** ***10002***
4.	[button] *10002*	**CLICK**
5.	**[button] *Search***	**CLICK**
6.	[switch] *Show BBB Accredited only*	**CLICK**
7.	[svg]	**CLICK**
8.	[button] *Sort By*	**CLICK**
9.	**[link] *Fast Lane 24 Hour Auto Repair***	**CLICK**
10.	[link] *Read Reviews*	**CLICK**

Webpage Snapshots:


```
<input name="find_text"
type="search">
```


```
<em>Auto Repair</em>
```


```
<button>Search</button>
```


```
<button>Show BBB Accredited
only</button>
```


```
<span>Fast Lane 24 Hour Auto
Repair</span>
```


```
<a href="link:XXX">Read
Reviews</a>
```

Skill Synthesis

Skill Library


check_drug_interaction(drug_name)


subscribe_to_newsletters(email)


search_FDA_alert()


check_side_effects(drug_name)


identify_pill(imprint, color)

Skill Synthesis

Practice Skill → **Reward Model** → **API Synthesis**

```python
async def identify_pill(page, imprint, color):
  # Automates the process of identifying a pill on Drugs.com.
  await page.goto("/pill_identification.html")
  await page.get_by_role("link", name="Agree and Continue").click()
  await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
  search_button = page.get_by_role("form", name="Pill Identifier").get_by_role("button", name="Search")


await search_button.click()
```

Skill Synthesis

Skill Library


check_drug_interaction(drug_name)


subscribe_to_newsletters(email)


search_FDA_alert()


check_side_effects(drug_name)


identify_pill(imprint, color)

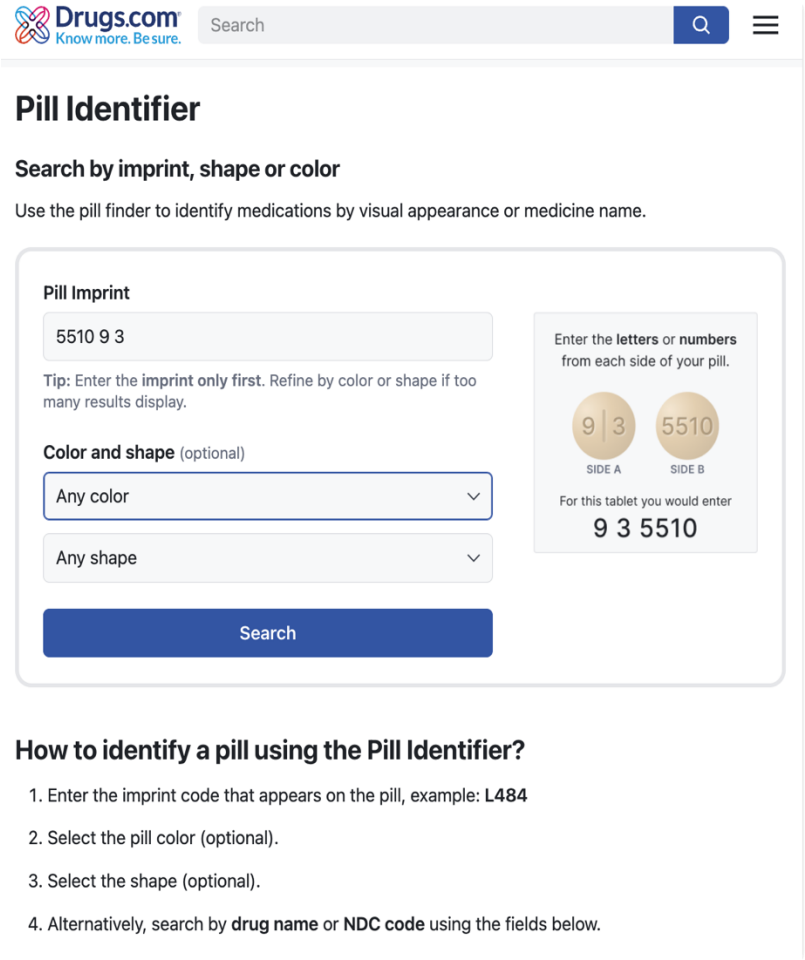
Skill Synthesis

Practice Skill → **Reward Model** → **API Synthesis**

```python
async def identify_pill(page, imprint, color):
  # Automates the process of identifying a pill on Drugs.com.
  await page.goto("/pill_identification.html")
  await page.get_by_role("link", name="Agree and Continue").click()
  await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
  search_button = page.get_by_role("form", name="Pill Identifier").get_by_role("button", name="Search")


await search_button.click()
```

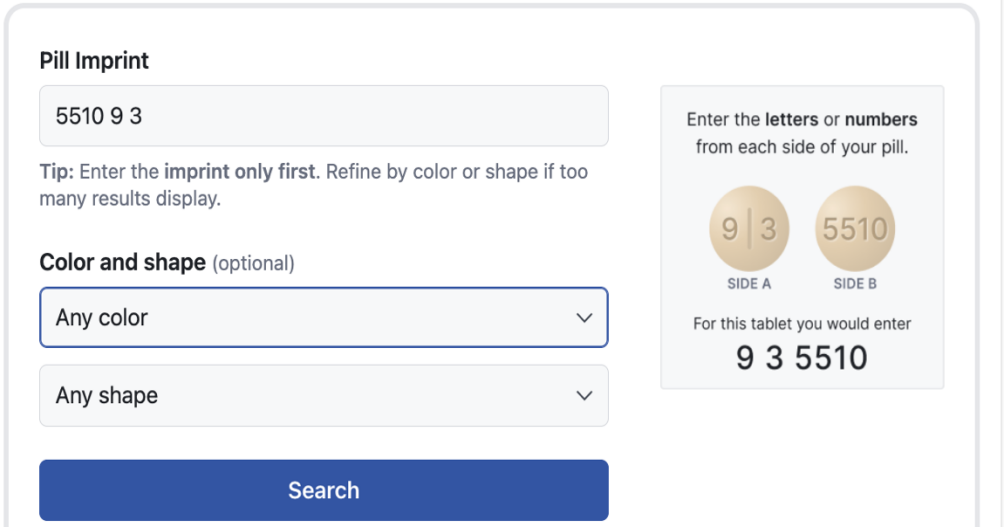
Verification

```
async def identify_pill(page, imprint, color):
  # Automates the process of identifying a pill on Drugs.com.
  await page.goto("/pill_identification.html")
  await page.get_by_role("link", name="Agree and Continue").click()
  await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
  search_button = page.get_by_role("form", name="Pill Identifier").get_by_role("button", name="Search")


await search_button.click()
```



Verification

```python
async def identify_pill(page, imprint, color):
  # Automates the process of identifying a pill on Drugs.com.
  await page.goto("/pill_identification.html")
  await page.get_by_role("link", name="Agree and Continue").click()
  await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
  search_button = page.get_by_role("form", name="Pill Identifier").get_by_role("button", name="Search")


await search_button.click()
```



Verification

Test Cases → **Execution**

Error:

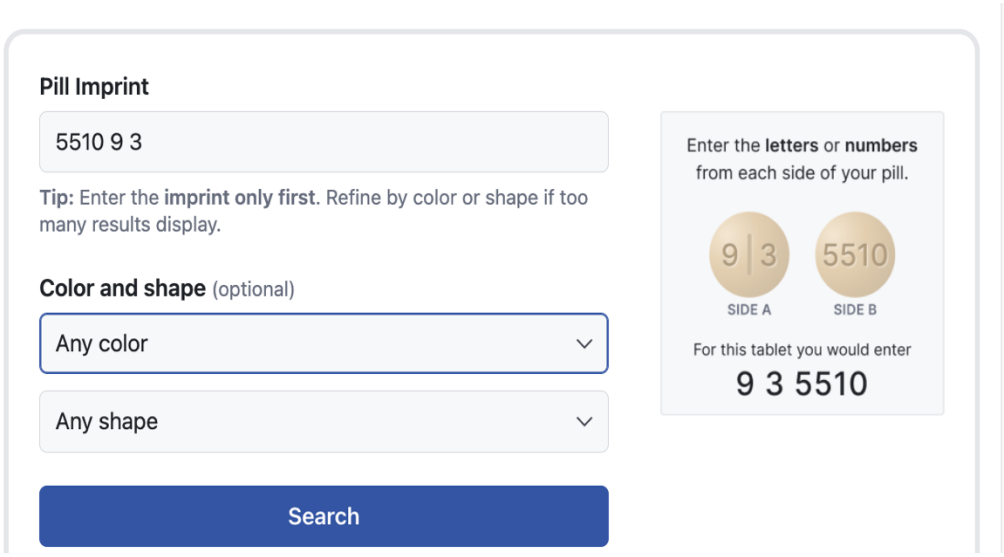
```
Warning: Unused parameter 'color'
  -> identify_pill(page, imprint, color)
  The parameter 'color' is defined but never used in
the function body

Error: Search results validation failed.
API returned pills matching imprint '5510' but the
wrong color (expected: yellow).
```

Diagnosis:

```
Added color selection using the dropdown menu
Made color filtering optional

Uses 'if color': to make color filtering optional
Properly implemented color parameter in the search
function, locating the "Color and shape" group element
```

Verification

```
async def identify_pill(page, imprint, color):
  # Automates the process of identifying a pill on Drugs.com.
  await page.goto("/pill_identification.html")
  await page.get_by_role("link", name="Agree and Continue").click()
  await page.get_by_role("textbox", name="Pill Imprint").fill(imprint)
  search_button = page.get_by_role("form", name="Pill Identifier").get_by_role("button", name="Search")
+ if color: await page.get_by_role("group", name="Color and shape (optional)").get_by_role(
+ "combobox", name="Color (optional)").select_option(color)
  await search_button.click()
```



Verification

Test Cases → **Execution**

Error:

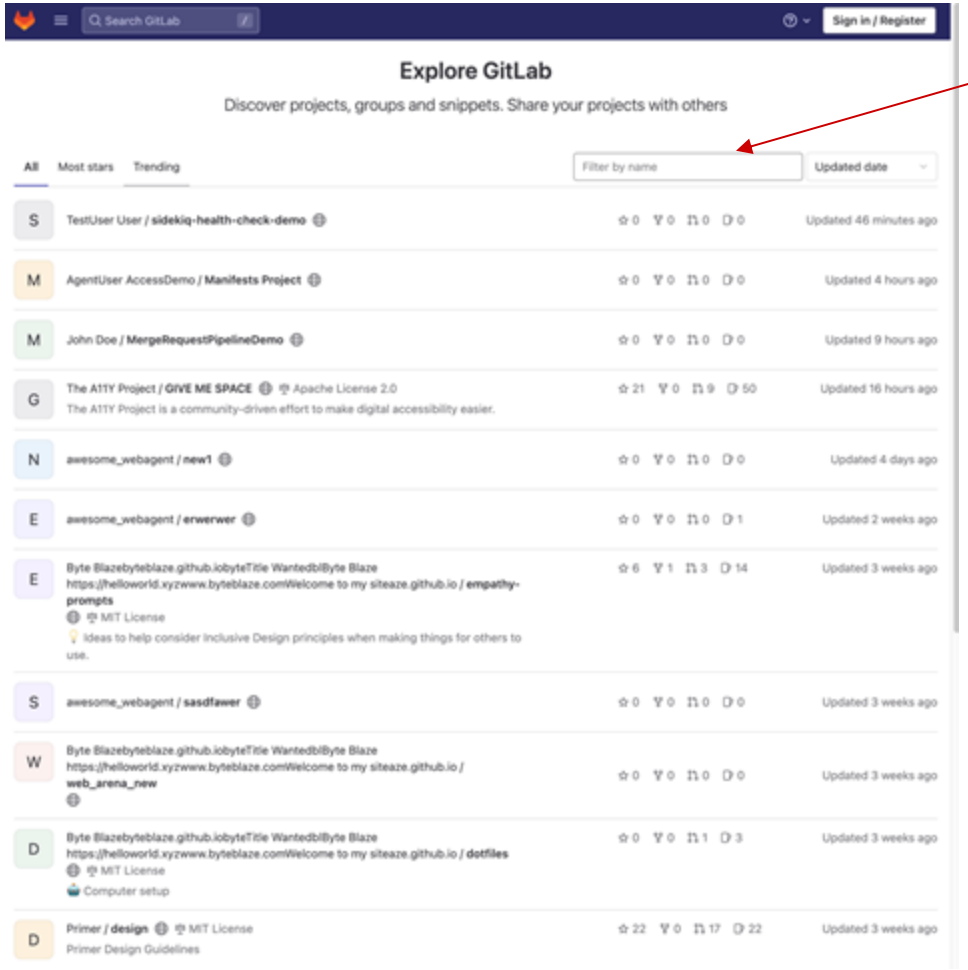
```
Warning: Unused parameter 'color'
  -> identify_pill(page, imprint, color)
  The parameter 'color' is defined but never used in
the function body

Error: Search results validation failed.
API returned pills matching imprint '5510' but the
wrong color (expected: yellow).
```

Diagnosis:

```
Added color selection using the dropdown menu
Made color filtering optional

Uses 'if color': to make color filtering optional
Properly implemented color parameter in the search
function, locating the "Color and shape" group element
```

More Examples of Synthesized APIs

APIs in GitLab



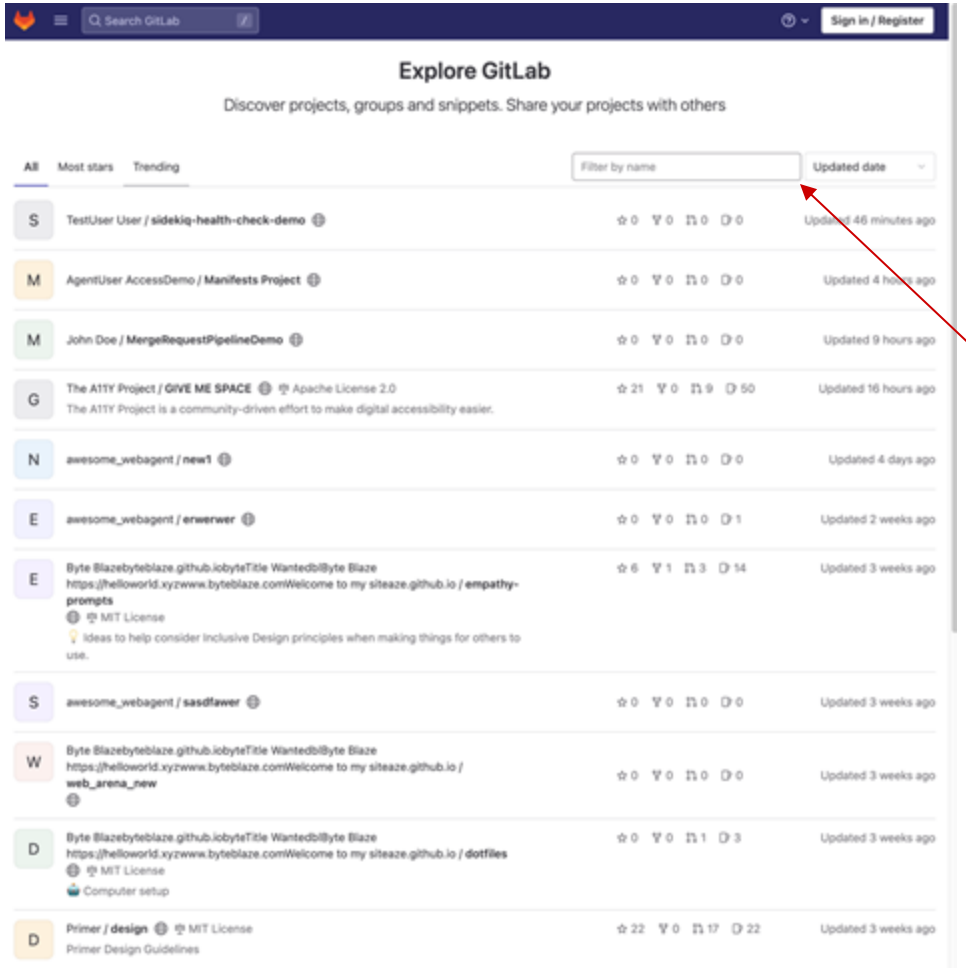

```python
async def search_projects_by_keyword(keyword: str):
    """
    Search for projects by a specific keyword.

    This function locates the 'Filter by name' search box, types the given keyword,
    and presses 'Enter' to execute the search operation.

    Usage:
    Call this function with the desired keyword to perform a search operation
    within the projects.

    :param keyword: The keyword to search for in projects.
    """
    searchbox_locator = page.get_by_role('main', name='').get_by_role(
        'searchbox', name='Filter by name')
    await searchbox_locator.click()
    await searchbox_locator.fill('')
    await searchbox_locator.type(keyword)
    await page.keyboard.press('Enter')
```

```python
async def explore_and_clone_repository(project_name: str) -> str:
    """
    Explore and clone a project repository by obtaining its clone URL.

    This function automates the process of navigating to the 'Explore' section of the website,
    selecting a project from the list, accessing the project page, and retrieving the clone URL by
    interacting with the 'Clone' options.

    Usage:
    Call this function with the `project_name` of the project you wish to clone. It will return the
    SSH clone URL of the repository as a string.

    :param project_name: The name of the project to explore and clone.
    :return: A string containing the SSH URL to clone the project repository.
    """
    await page.get_by_role('main', name='').get_by_role('link', name='Explore'
        ).click()
    await page.get_by_role('main', name='Heading', exact=True).get_by_role(
        'link', name=project_name).click()
    await page.get_by_role('main', name='').get_by_role('link', name='Clone'
        ).click()
    await page.get_by_role('main', name='').get_by_role('button', name=
        'Copy URL').click()
    await page.get_by_role('main', name='').get_by_role('link', name='Clone'
        ).click()
    return 'ssh://git@164.107.112.39:2222/{}/{}.git'.format('root', project_name,
        split(' / ')[0].lower(), project_name.split(' / ')[1].replace(' ',
        '-').lower())
```

```python
async def merge_project_branches(project_name: str, merge_request_title: str):
    """
    Explore and merge branches for a project that has open merge requests.

    This function navigates to the list of open merge requests for a given project, selects a specific merge request by its title,
    explores the merging options, and performs the merge if it is deemed acceptable (even if there are unverified changes).

    Usage:
    Call this function with the `project_name` to open the merge request list, and `merge_request_title` to select and
    merge the branches.

    :param project_name: The name of the project which contains the merge requests.
    :param merge_request_title: The title of the merge request to explore and potentially merge.
    """
    await page.get_by_role('main', name='').get_by_role('link', name='2'
        ).click()
    await page.get_by_role('main', name='').get_by_role('link', name=
        merge_request_title).click()
    await page.get_by_role('main', name='').get_by_role('group', name=''
        ).get_by_role('button', name='Merge...').click()
    await page.get_by_role('dialog', name='Merge unverified changes?'
        ).get_by_role('contentinfo', name='').get_by_role('button', name=
        'Merge unverified changes').click()
    return 'The merge has been completed successfully.'
```

More Examples of Synthesized APIs

APIs in GitLab




```python
async def search_projects_by_keyword(keyword: str):
    """
    Search for projects by a specific keyword.

    This function locates the 'Filter by name' search box, types the given keyword,
    and presses 'Enter' to execute the search operation.

    Usage:
    Call this function with the desired keyword to perform a search operation
    within the projects.

    :param keyword: The keyword to search for in projects.
    """
    searchbox_locator = page.get_by_role('main', name='').get_by_role(
        'searchbox', name='Filter by name')
    await searchbox_locator.click()
    await searchbox_locator.fill('')
    await searchbox_locator.type(keyword)
    await page.keyboard.press('Enter')
```

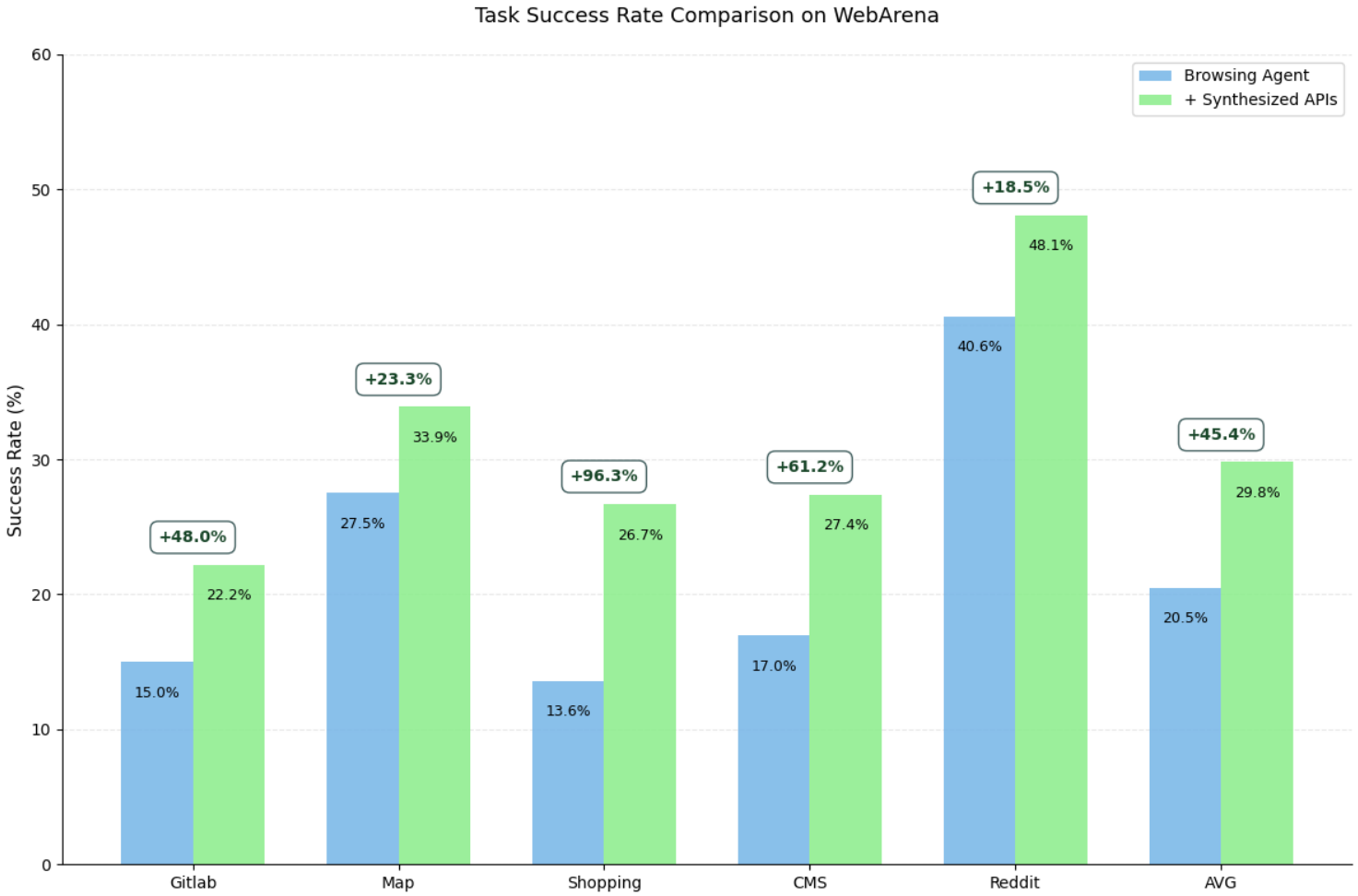
Evaluation on Sandbox

Exploration:

Synthesize APIs with 160 Iterations

Agents:

- Browsing Agent:
  - Browser Action
- +API Synthesis
  - Extend action space with APIs



How far are we from human-crafted APIs?

- Human Crafted APIs:
  - Mining APIs from sandbox source code

Not realistic, but can serve for a case study

API Documentation

```
# Commits
## GET /api/{id}/commits: Get a list of commits in a project.
| Attribute | Type           | Description                          |
| `id`      | integer/string | The ID or path of the project.       |
| `since`   | string         | Only commits after or on this date.  |
| `until`   | string         | Only commits before or on this date.|
Output: JSON containing all commits that meet the given criteria.
```

API Calling

```
<execute_ipython>
requests.get('gitlab.com/api/a11yproject/commits')
</execute_ipython>
```

JSON Output

```
[ ......{
    "id": "ed37a2f2",
    "created_at": "2023-03-13T21:04:49.000-04:00",
    "title": "Update README.md",
    "message": "Update README.md",
    "author": "SaptakS",
}]
```

Comparing with human crafted APIs

API Support Level

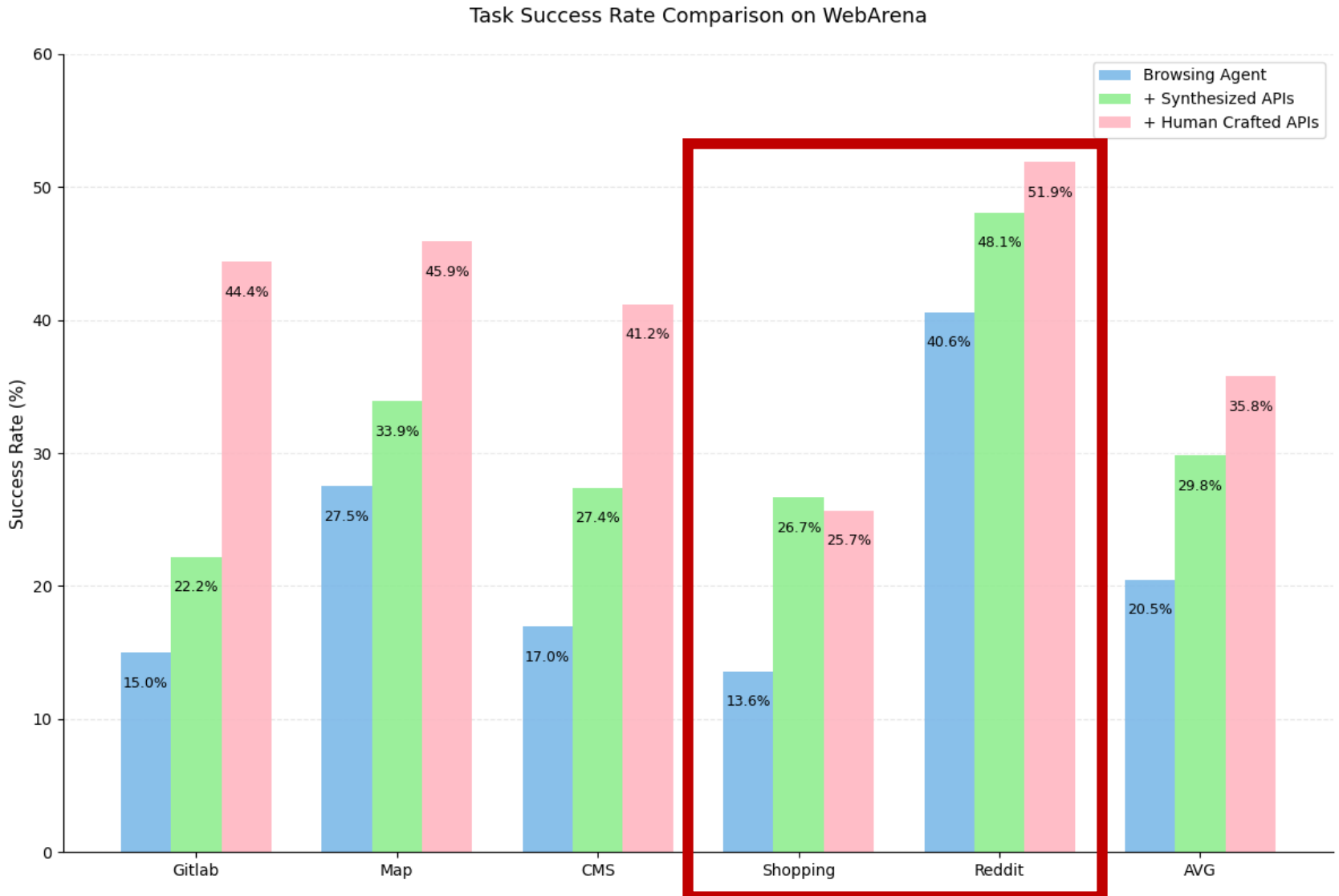
MEDIUM:

Admin Access

Limited APIs

LOW:

Manually written automation Code

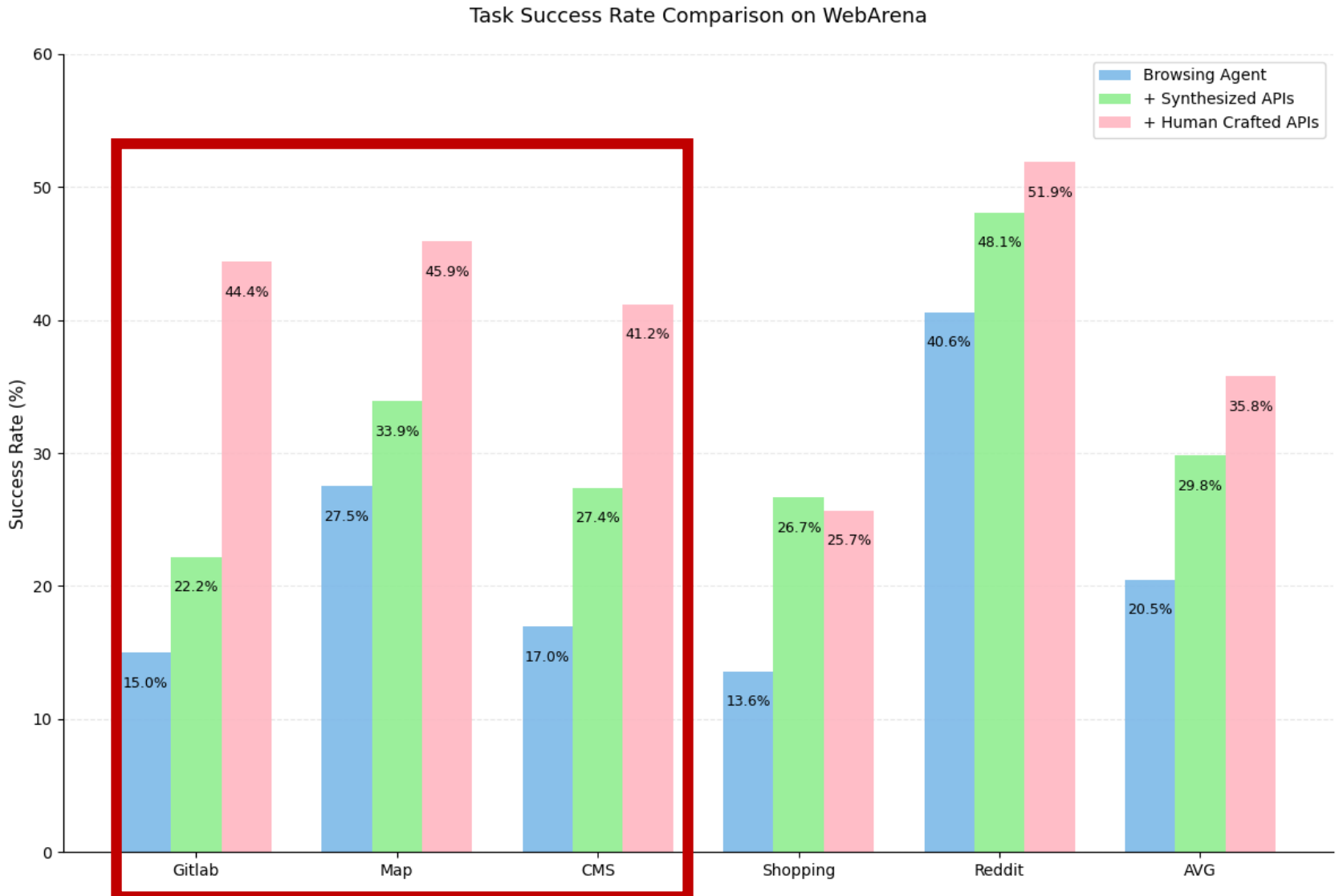


Comparing with human crafted APIs

HIGH API support:

Admin Access

High Quality API



Generalization to weaker Agent

- Empowering smaller models with APIs synthesized by larger models
- Exploration: GPT-4o
- Inference: GPT-4o-mini

	SkillWeaver					
GPT-4o	17.8	27.5	19.8	18.7	37.7	22.6
+ Skills	22.2	33.9	27.2	25.8	50.0	29.8
Δ	↑ 25%	↑ 23%	↑ 38%	↑ 38%	↑ 33%	↑ 32%
GPT-4o-mini	6.1	10.3	11.8	3.3	18.9	9.2
+ Skills	8.9	16.7	17.1	7.7	26.4	14.1
Δ	↑ 46%	↑ 62%	↑ 46%	↑ 133%	↑ 40%	↑ 45%

Scaling to Live Websites

Exploration:

80 Iterations

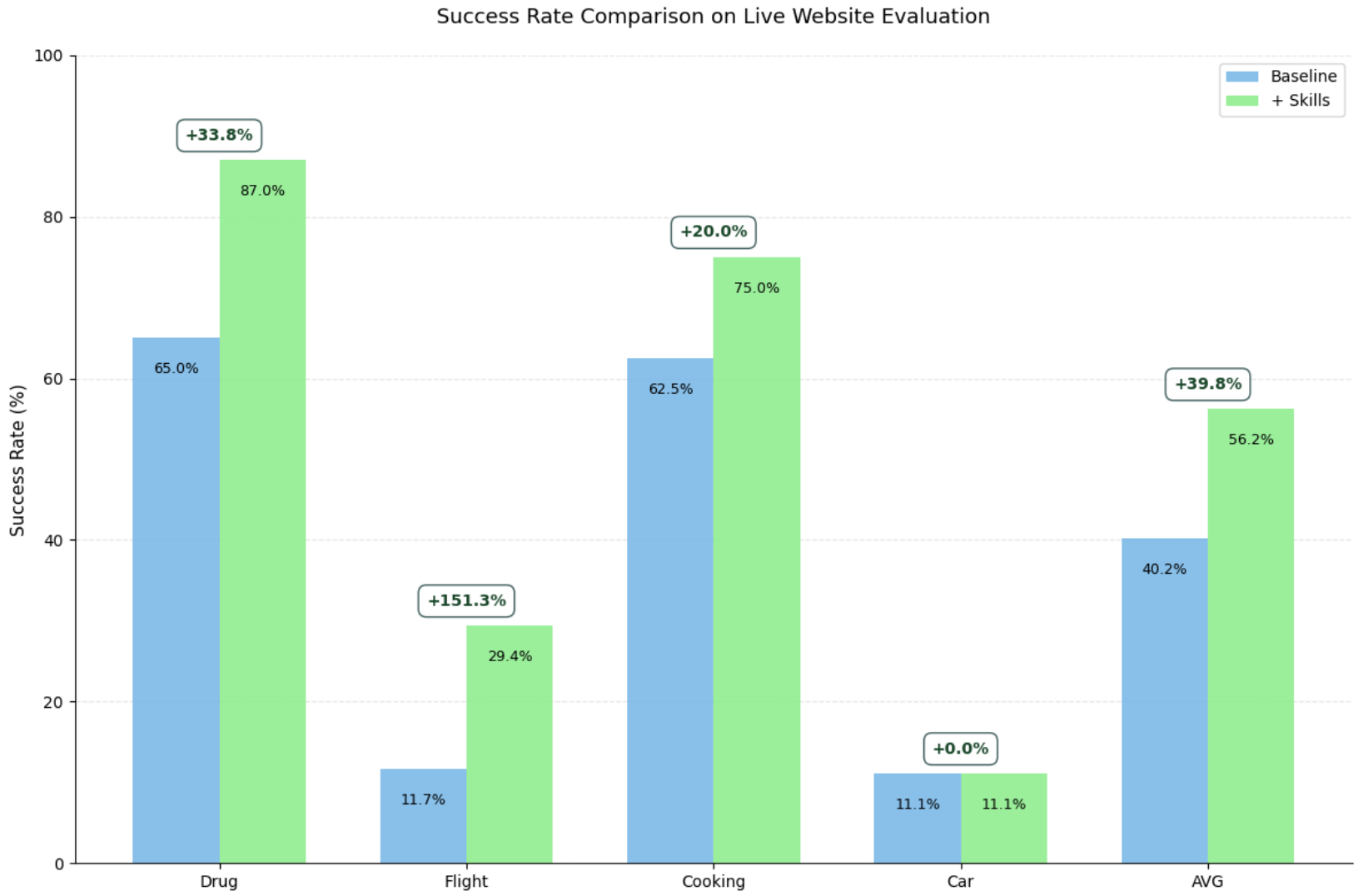
Evaluation:

Drug: 23

Flight: 17

Cooking: 8

Car: 9



Thank you!
&
Questions?