Language Agents

Hanane Nour Moussa, Ph.D. Student, The Ohio State University

04/17/2026

Outline

- Why Agents?
- What are Agents?
- Language Agents: A Conceptual Framework
  - Agent
  - Memory
  - Environment
  - Reasoning and Planning
  - Tool Augmentation
  - Embodiment
- Language Agents: An Engineering Perspective
  - Workflow vs Agent, common design patterns
- Research on agents
  - Coding Agents
  - Broader research overview

Why Agents?

- Current language models (LMs) recipe:
  - Pretrained with large corpus by next token prediction objective
  - Postrained via:
    - Instruction following
    - RLHF (reinforcement learning with human feedback)
    - RLVR (reinforcement learning with verifiable rewards)
- This recipe has led to useful **models with strong natural language understanding capabilities**
- ***But …***

Why Agents?

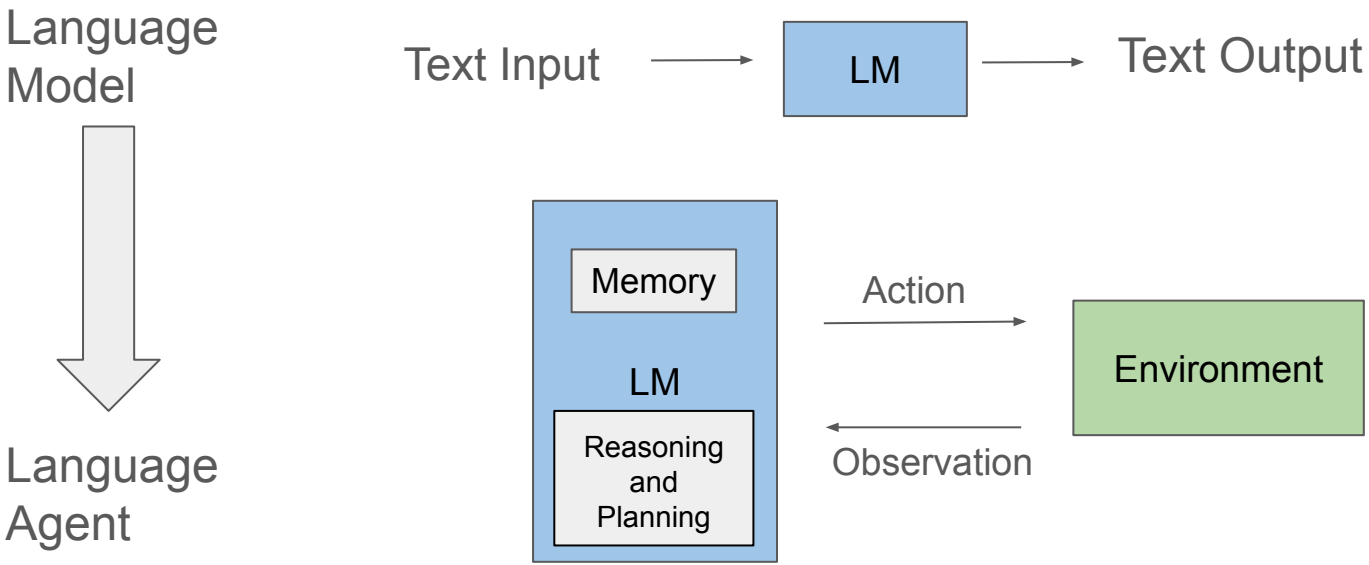
- Current language models **still have many limitations**
- **Hallucinations:** Generation of incorrect information with high confidence
- **Knowledge cutoff:** limited to training data timeframe
- **Lack of attribution:** No direct source citations
- **Data privacy:** Limited to public training data, no access to proprietary information
- **Limited context length:** Constraints due to attention mechanism architecture

Why Agents?

- **How to overcome these limitations?**
- **RAG** (Retrieval augmented generation)
  - Augments LMs with knowledge from external sources
  - Addresses hallucination, lack of attribution, data privacy
- **Tool usage**
  - Expands capabilities
  - Addresses real-time information, (domain-specific) computations
- **Reasoning and Self-Reflection**
  - Overcome hallucinations

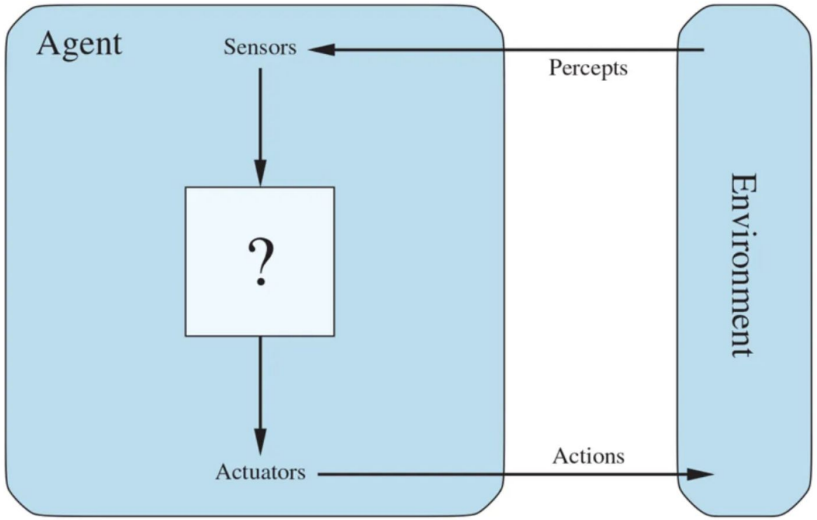
Why Agents?

All these move LMs **away from static language generators** towards **agents that interact with the environment** (via retrieving new knowledge, using tools, etc.)



What are Agents?

- The concept of agent has been introduced in AI since its dawn
- “**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

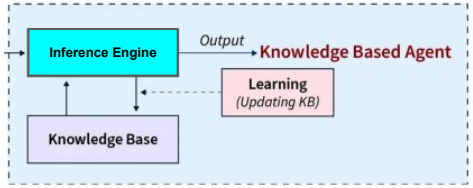


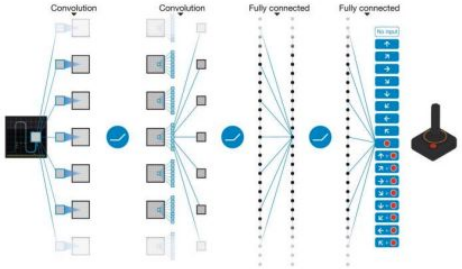
Schematic illustration of a classic agent (Russell & Norvig, 2020, Ch. 2).

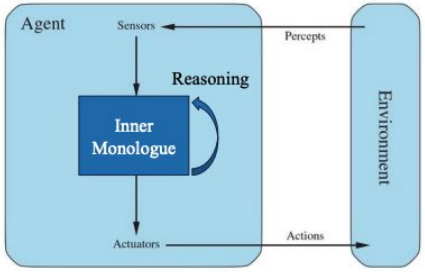
What are Agents in the LM era?

- LMs are integrated into AI agents so they can ***use language for reasoning and communication*** ⇒ **Language Agents**
- Enhanced language capabilities **increase expressiveness, reasoning ability , and adaptivity**
- This makes it much easier to incorporate heterogeneous external percepts and do multi-step (speculative) planning and reasoning, all in a non-programmed and explicit way

Evolution of AI Agents







	Logical Agent	**Neural Agent**	**Language Agent**
Expressiveness	Low bounded by the logical language	Medium anything a (small-ish) NN can encode	High almost anything, esp. verbalizable parts of the world
Reasoning	Logical inferences sound, explicit, rigid	Parametric inferences stochastic, implicit, rigid	Language-based inferences fuzzy, semi-explicit, flexible
Adaptivity	Low bounded by knowledge curation	Medium data-driven but sample inefficient	High strong prior from LLMs + language use

Increasing Expressiveness, Reasoning Ability & Adaptivity

Language Agents: a Conceptual Framework

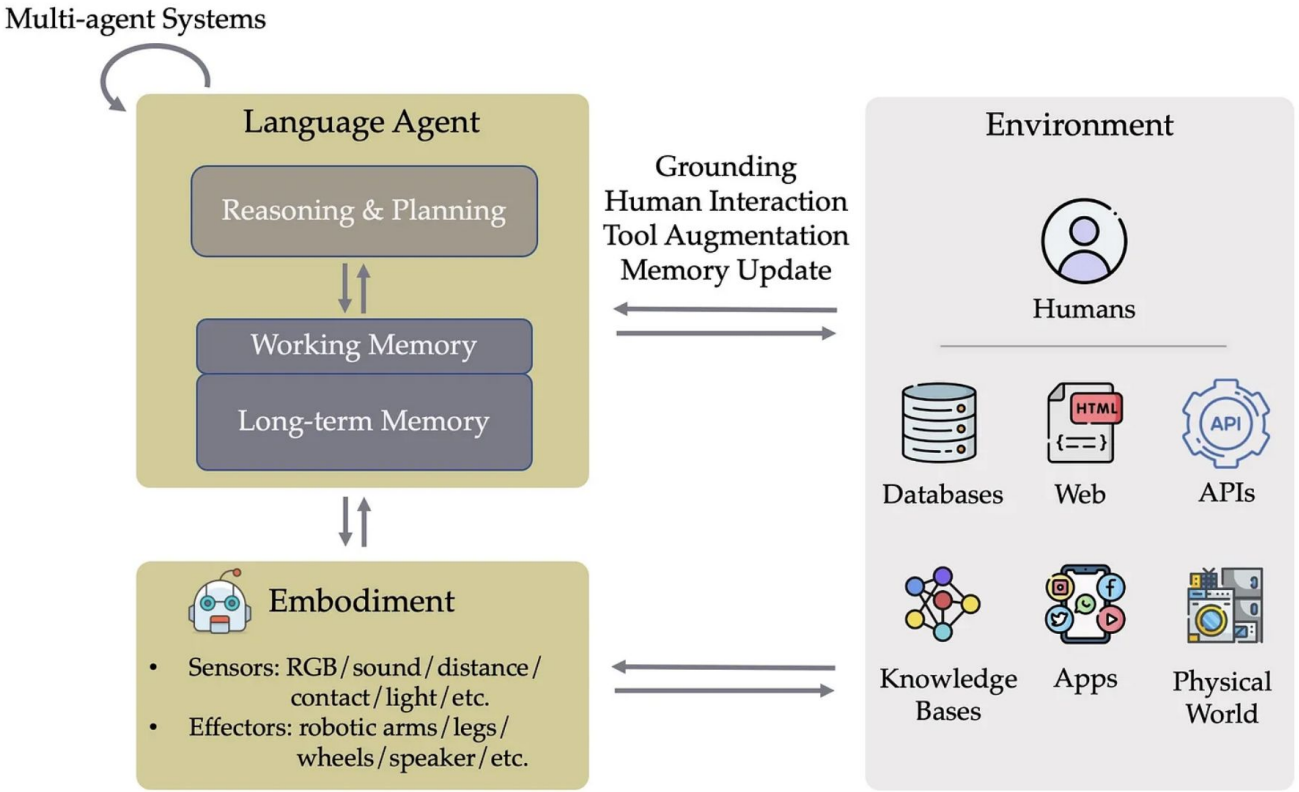


Figure credit: [Yu's Substask, Language agents: a critical evolutionary step of artificial intelligence]

Language Agents: a Conceptual Framework

Agent

- An agent is a system that **receives observations or inputs from an environment, maintains an internal state or policy**, and **produces actions that can change the environment**, in order to **achieve specified goals or maximize a given objective**
- Language agents are LM-based agents where **natural language is the main interface for state, reasoning, and action selection**

Language Agents: a Conceptual Framework

Agent

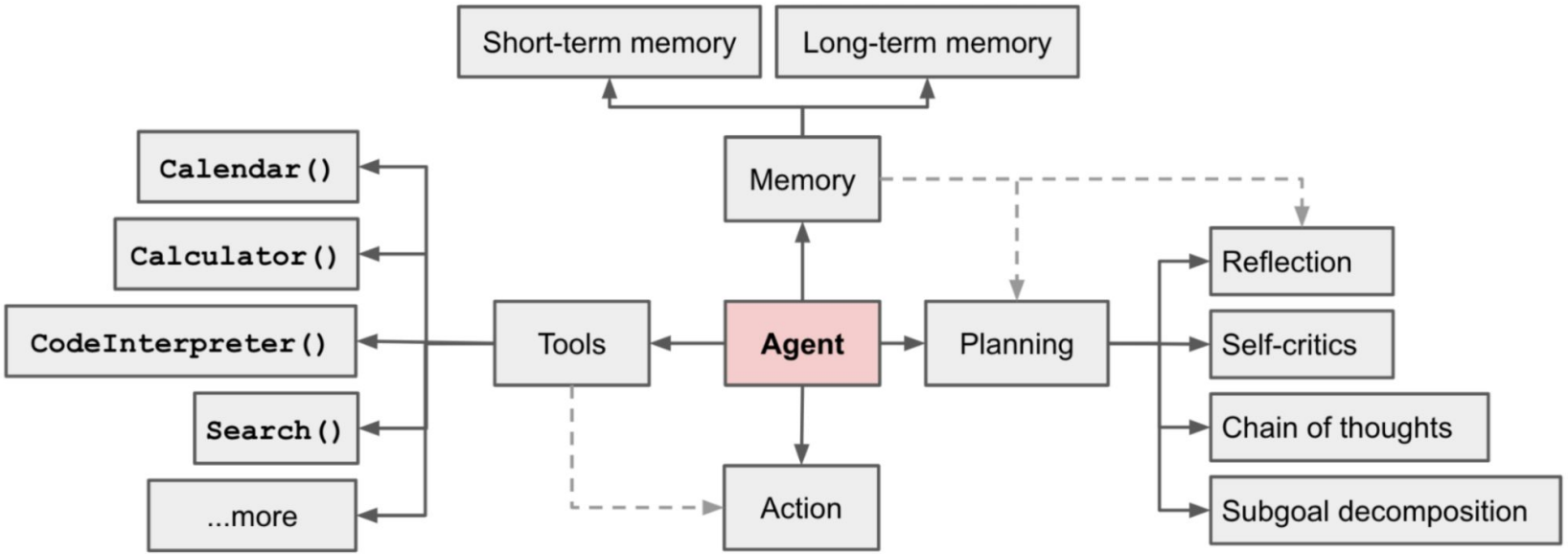


Figure Credit: "LLM-powered Autonomous Agents". Lil'Log.

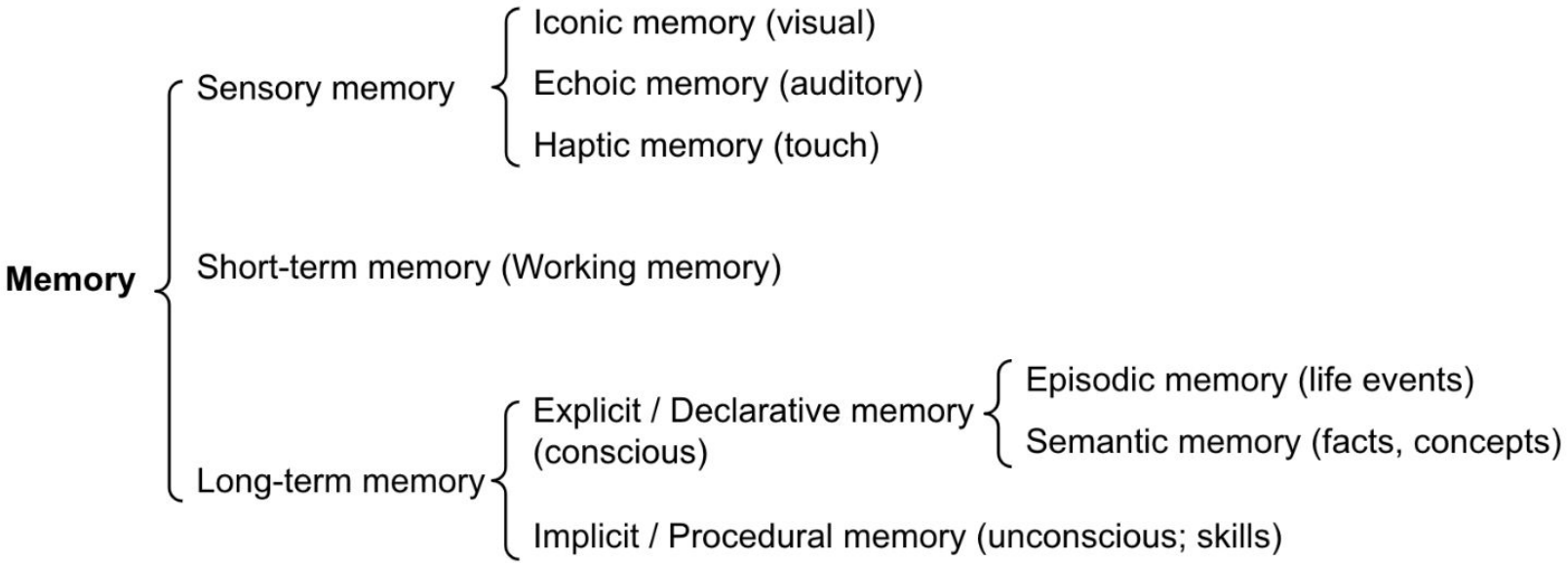
Language Agents: a Conceptual Framework

Memory

- Memory in AI agents is the ability to recall important information across multiple user interactions
- At a high level, we can differentiate between two types:
  - **Long-term memory** =
    - **Parametric:** pretrained parameters of the LM
    - **External:** information stored in external storage (e.g. vector or graph database)
  - **Working-memory** = LM context and in-context learning capabilities
    - Scratchpad to temporarily hold task-related information

Language Agents: a Conceptual Framework

Memory



Categorization of Human Memory

Figure Credit: "LLM-powered Autonomous Agents". Lil'Log.

Language Agents: a Conceptual Framework

Memory

- Hot topic currently, some open problems:
  - **Context management:** How to effectively manage working memory over long interactions?
  - **Forgetting:** How do you automate a mechanism that decides when and what information to permanently delete/unlearn?
  - **Continual Learning:** How to dynamically add more information and knowledge to long-term memory?

Language Agents: a Conceptual Framework

Planning and Reasoning

How to break a goal into steps and how to decide what to do next.

- **Reasoning**: processing inputs (goals, context, tool results) into conclusions or intermediate thoughts
- **Planning**: Special case of reasoning where the agent explicitly organizes actions over time

Language Agents: a Conceptual Framework

Planning and Reasoning

Techniques of reasoning and planning / task decomposition

- **Chain of thought** (Wei et al, 2022)
  - “Think Step by Step”
- **Tree of thoughts** (Yao et al, 2023)
  - Extend CoT by exploring multiple reasoning possibilities at each step

These are not agent frameworks but rather building blocks of agent frameworks such as ReAct, Plan-and-Execute, etc. (Next slide)

Language Agents: a Conceptual Framework

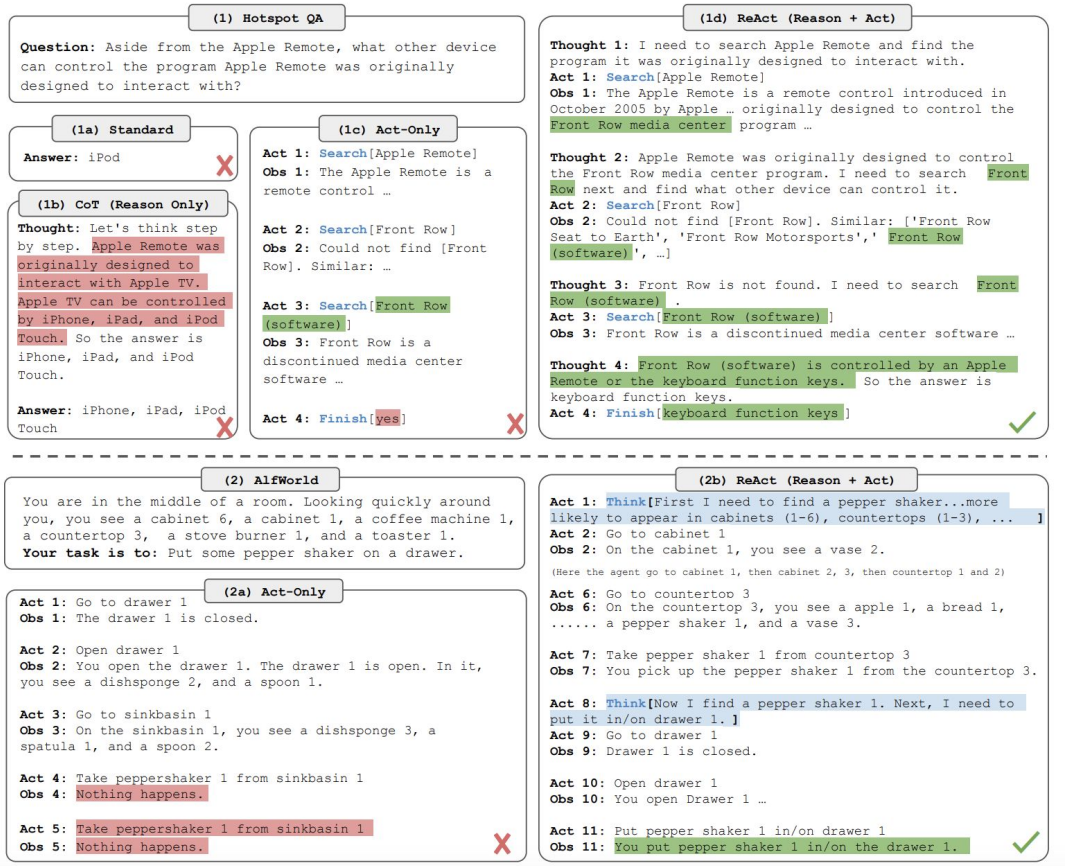
Planning and Reasoning

ReAct = Reason + Act (Yao et al, 2023)

- Agent alternates between 3 phases:
  - Observe: See the current state or tool results
  - Think (Reason): Write out short natural-language thoughts (e.g. CoT)
  - Act: Choose and execute an action
- **Observe → Think → Act → (repeat)**

Language Agents: a Conceptual Framework

ReAct



Language Agents: a Conceptual Framework

Planning and Reasoning

Plan and Execute [1]

- Instead of interleaving reasoning and actions, we do it in two stages:
  - **Planning** stage: output full high level plan
  - **Execution** stage: Go through the plan, choose tools, act, update state
- Gives more coherent long-horizon plans compared to ReAct, but plan can become stale if environment changes

[1] https://langchain-ai.github.io/langgraph/tutorials/plan-and-execute/plan-and-execute/

Language Agents: a Conceptual Framework

Planning and Reasoning

Plan and Execute

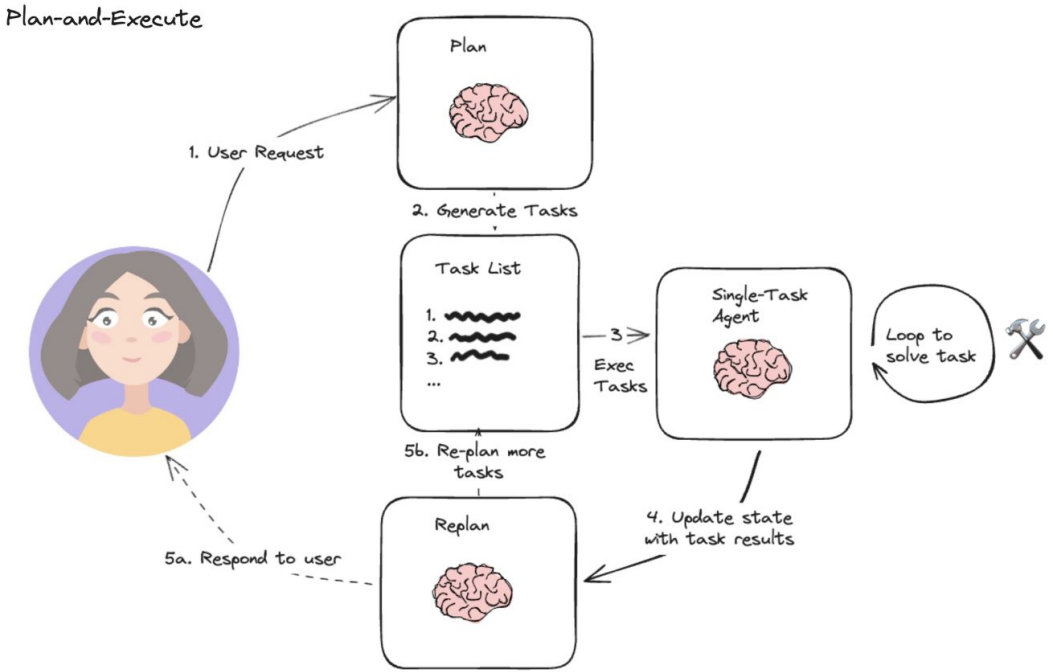


Figure Credit: [Langchain, Plan-and-Execute]

Language Agents: a Conceptual Framework

Tool Augmentation

Tool augmentation mainly serves three purposes:

- Provide **up-to-date and/or domain-specific information**.
- Provide **specialized capabilities** (e.g., high-precision calculation, route planning on a map) that a language agent may not have or be best at.
- Enable a language agent to take **actions in real-world environments** (compile code, click on button, etc.)

Language Agents: a Conceptual Framework

Tool Augmentation

Generally speaking, there are two types of tools

- **Read-only** (e.g. a calculator, search engines, knowledge databases, etc.)
- And ones that produce side-effects in the world (e.g., APIs for sending emails or scheduling meetings) ⇒ **state-changing** tools
- Most existing tool augmentation efforts limit themselves to read-only tools, or only use tools with side effects in a sandbox or with a human-in-the-loop / explicit permission

Language Agents: a Conceptual Framework

Tool Augmentation

Two measures are key for successful tool augmentation: robustness and flexibility

- **Robustness** = language agent uses the right tool at the right time in the right way
- **Flexibility** = how easy it is to integrate new tools

Language Agents: a Conceptual Framework

Tool Augmentation

Active areas of research related to tooling:

- How to create more effective (open-source) models for tool use? Especially in real world use cases representing various perturbations?
- Benchmarking model tool use over diverse, realistic, and long-horizon tasks [1]

[1] Li et al. The Tool Decathlon: Benchmarking Language Agents for Diverse, Realistic, and Long-Horizon Task Execution. https://arxiv.org/abs/2510.25726.

Language Agents: a Conceptual Framework

Environment

- **The external world in which the agent operates:**
  - Has a **state** (possibly hidden from the agent)
  - Produces **observations or feedback** to the agent (and optionally a **reward**)
  - Is **affected by the agent's actions** according to some dynamic (e.g. actions produced by tools)
- E.g. Coding Agent
  - Environment:
    - State: entire file system, git repo content, etc.
    - Observations: file contents, diffs, test outputs, compilations errors, etc.
    - Actions: editing files, compiling code, opening PRs, etc.

Language Agents: a Conceptual Framework

Embodiment

- The **form or interface (physical or digital) through which an agent is instantiated in an environment, giving it concrete ways to sense and act**
- A robot: embodiment = sensors + motors + grippers.

- A coding agent: embodiment = file system access, editor APIs, test runner.

- A chat assistant: embodiment = text I/O channel, plus any tools (search, DB, email APIs).

Language Agents: An Engineering Perspective

- The term “Agent” is sometimes too loosely used
- Some define agents as fully autonomous systems that operate independently over extended periods using various tools to accomplish complex tasks
- Others use the term to describe more prescriptive implementations that follow predefined workflows. ⇒ Both are often loosely grouped under agentic systems
- But there are distinct differences

Language Agents: An Engineering Perspective

- Workflows vs Agents
  - **Workflows** are systems where LLMs and tools are orchestrated through predefined code paths.
  - **Agents** are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.

Language Agents: An Engineering Perspective

- **When to use each?** Always find the **simplest solution possible**, and **only increase complexity when needed**
- **Workflows** offer **predictability** and **consistency** for well-defined tasks
- **Agents** are the better option when **flexibility** and **model-driven decision-making** are needed at scale.

Language Agents: An Engineering Perspective

Workflow Design Pattern: Prompt chaining

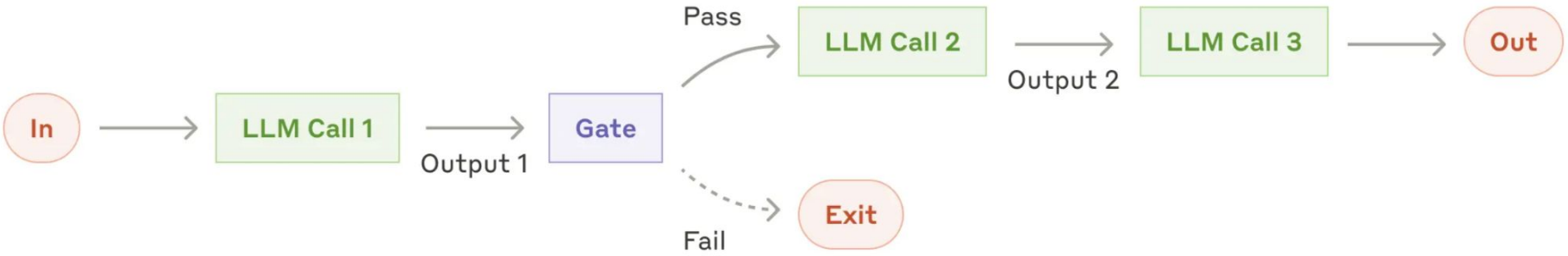


Figure Credit: Anthropic, Building effective agents

Language Agents: An Engineering Perspective

Workflow Design Pattern: Routing

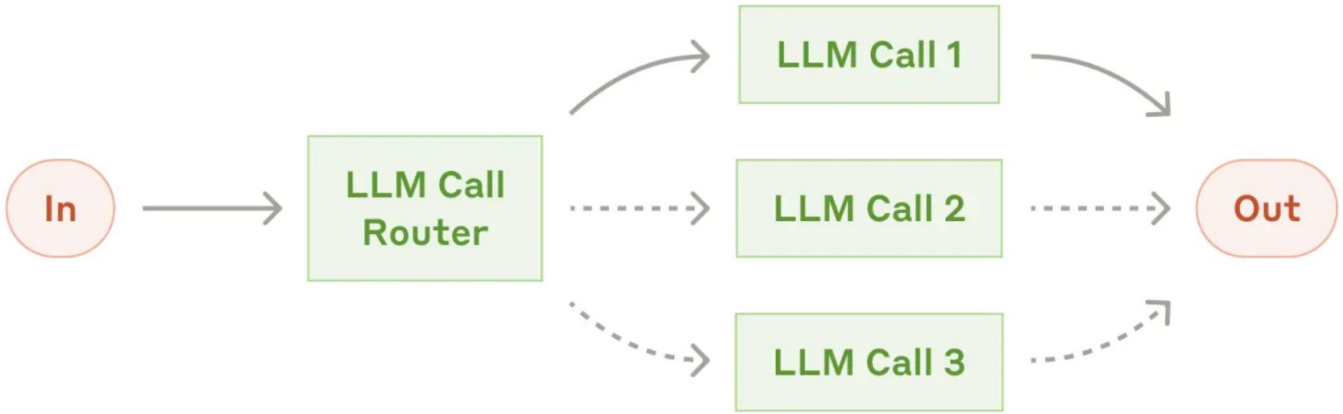


Figure Credit: Anthropic, Building effective agents

Language Agents: An Engineering Perspective

Workflow Design Pattern: Parallelization

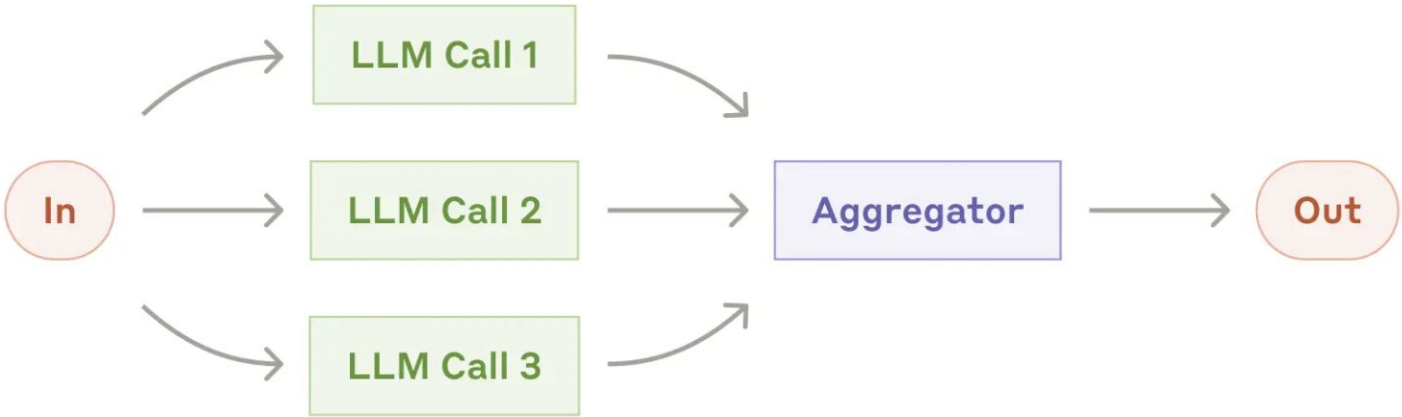


Figure Credit: Anthropic, Building effective agents

Language Agents: An Engineering Perspective

Workflow Design Pattern: Orchestrator-Workers

Figure Credit: Anthropic, Building effective agents

Language Agents: An Engineering Perspective

Workflow Design Pattern: Evaluator-Optimizers

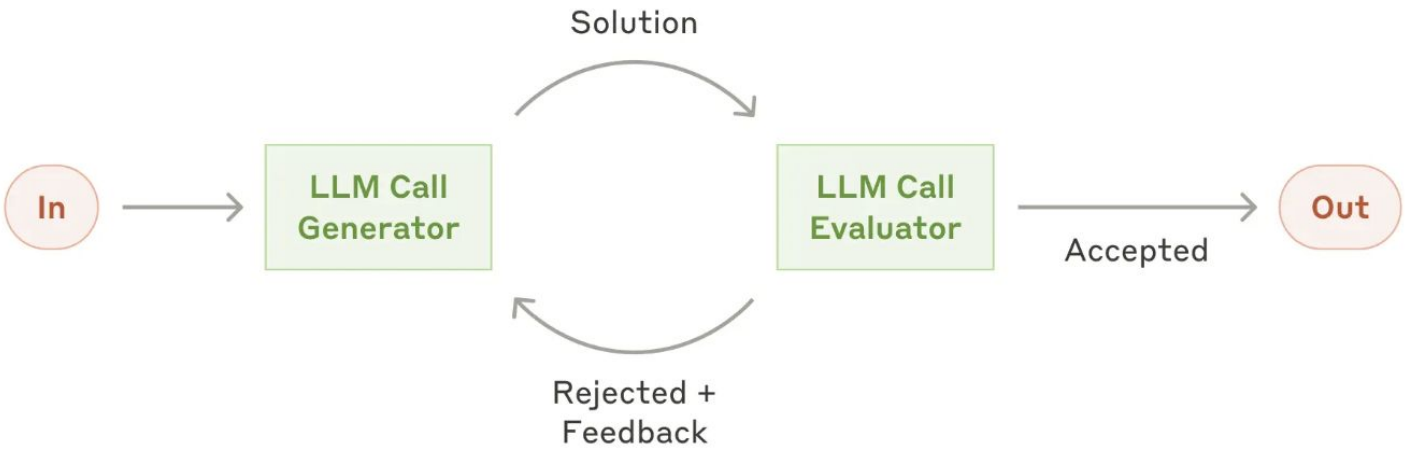
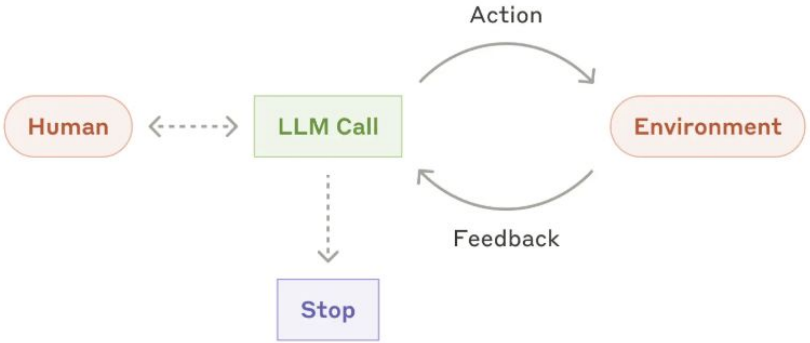


Figure Credit: Anthropic, Building effective agents

Language Agents: An Engineering Perspective

- **When to use an agent?**
- Agents can be used for open-ended problems where **it's difficult or impossible to predict the required number of steps, and where you can't hardcode a fixed path.**
- The autonomous nature of **agents** means **higher costs**, and the potential for **compounding errors**.



Language Agents: An Engineering Perspective

Some commonly used frameworks to build agentic systems:

- AutoGen https://microsoft.github.io/autogen/stable//index.html
- LangChain https://www.langchain.com/
- Or for the most flexibility, build your own!

AI agents applications and ongoing research

Coding Agents

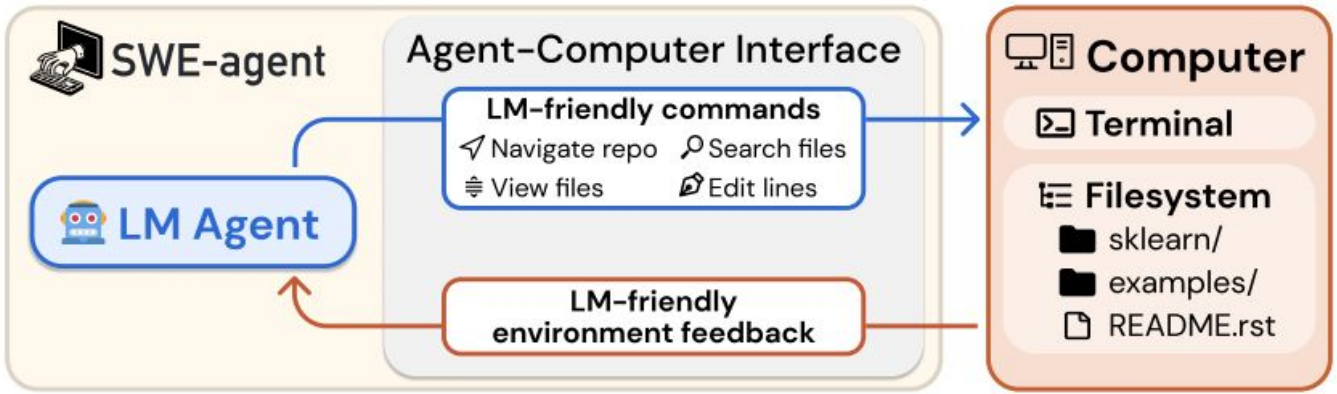
An (LLM-based) agent that has access to a set of tools for reading, writing, and executing code, and runs in a loop to autonomously complete coding tasks.

Lots of work focuses on **software engineering tasks** (SWE-Bench style, i.e. producing code patches that solve real-world GitHub issues, judged by whether the repository's own unit tests pass after the patch is applied.)

Following slides give a brief view about the progression of research on coding agents.

SWE-agent (Yang et al. 2024)

- First introduction of Agent-Computer Interfaces (now referred to as scaffolds/harnesses)
- Idea: **Frozen language model + Agent-Computer Interface + ReAct loop** can significantly improve performance on software engineering tasks (SWE-Bench)



SWE-RL (Wei et al., 2025)

- Instead of using a frozen model, this work (and similar ones) focuses on **training the underlying LM for code reasoning.**

1. Data: Curate ~200K GitHub pull requests (issue description, code context, oracle patch)

2. Task: Prompt the model with the issue and ask it to produce reasoning + final solution

3. Reward: Compare model solution to oracle patch using string similarity to give a score [0, 1]. -1 is output does not respect format

4. Optimization: train with RL (GRPO)

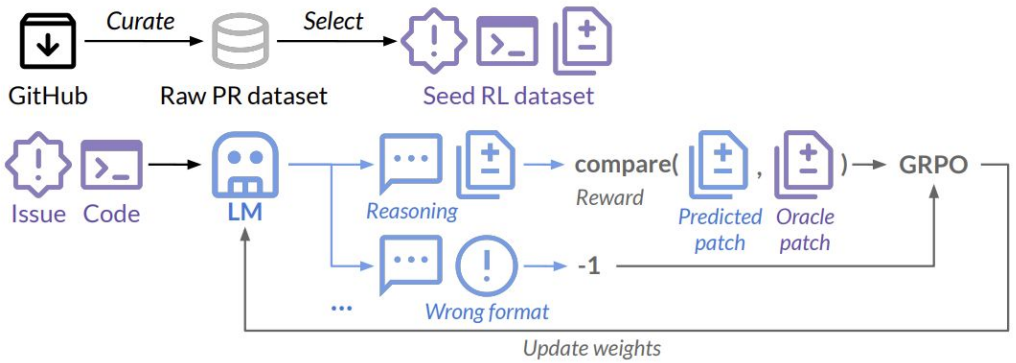


Figure 1: **Overview of SWE-RL.** We create a seed RL dataset from GitHub PRs, including issue descriptions, code context, and oracle patches. A policy LLM generates code edits via reasoning. Rewards are based on the similarity to oracle patches, with penalties for formatting errors. We optimize the policy using GRPO.

SWE-Gym (Pan et al., 2025)

- Creating a **‘Gym-environment’** composed of real-world task instances comprising a python codebase with executable runtime environment, unit tests, and a task specified in natural language.
- Record the full **trajectories** (sequences of reasoning, tool calls, and observations ending with the final patch) using **teacher models** running inside an agent scaffold on the SWE-Gym tasks
- Keep the trajectories that pass the unit tests
- Train the underlying LM on these successful trajectories → **rejection sampling fine-tuning**

A Self-Improving Coding Agent (Robeyns et al., 2025)

- A coding agent ***is code***. If it has tools to edit files and run tests, it can edit its own files and run its own benchmarks.

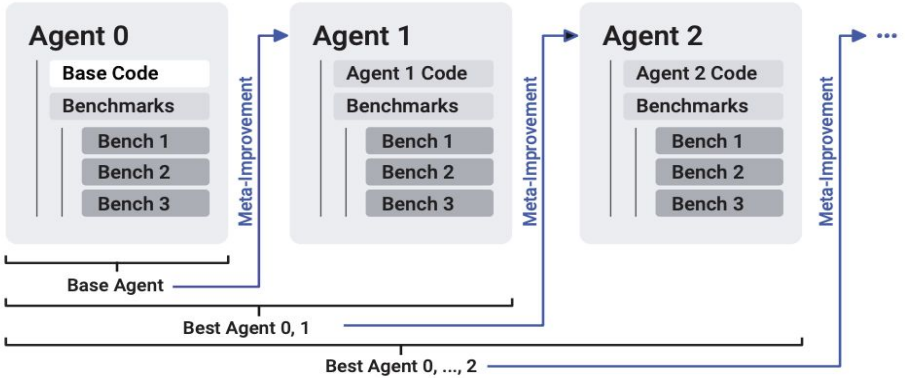
1. **Evaluate** every agent in the archive on a benchmark set (SWE-Bench, LiveCodeBench)

2. **Score** each using a utility function: U= 0.5 . score + 0.25 . (1 - cost) + 0.25 . (1 - time)

3. **Select** the highest-utility agent so far as the meta-agent for the next round

4. **Meta-improvement step:** the meta-agent is pointed at the archive, inspects past agents' successes and failures, and writes code changes to produce new agent versions

5. **Append** the new agent to the archive. **Repeat**.



Other Agents

(Generalist) Web Agents (or Computer Use Agents (CUA))

An (LLM-based) agent that can **take natural language goals** and **operate a browser-like interface to solve different tasks across websites**

Why web agents?

- The **web offers an open, diverse, and complex environment for agents**
- **User tasks are diverse, complex, and require long-horizon planning**
- **Generalization can be measured on unseen websites/domains**

SeeAct: A Generalist Web Agent Based on Action Grounding

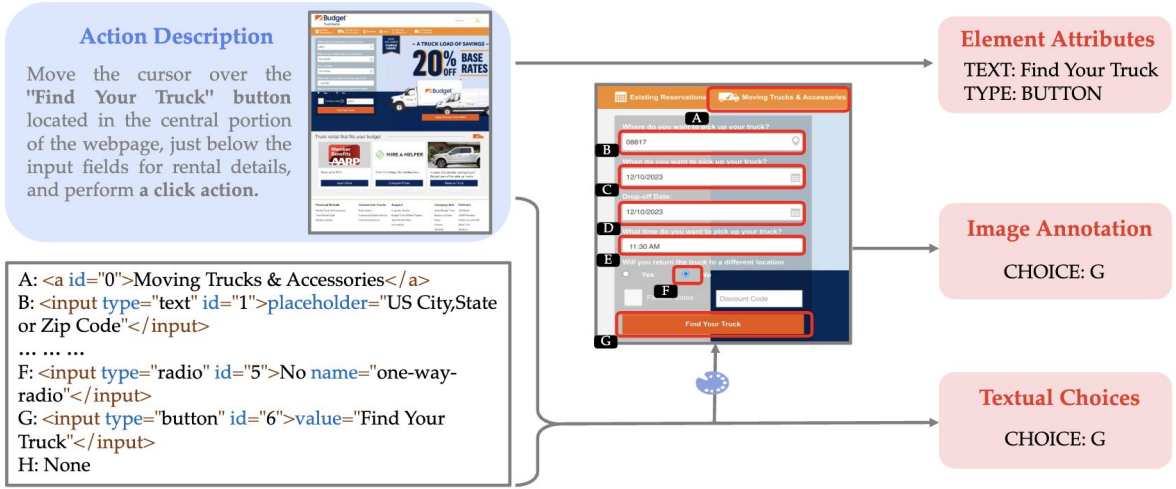
GPT-4V(ision) is a Generalist Web Agent, if Grounded

Boyuan Zheng [1] **Boyu Gou** [1] **Jihyung Kil** [1] **Huan Sun** [1] **Yu Su** [1]

https://osu-nlp-group.github.io/SeeAct

SeeAct: A Generalist Web Agent Based on Action Grounding

- Generalist web agent based on large multimodal models (LMMs) like GPT-4V
- Given a task on any website, the agent first performs **Action Generation** to produce a textual description of the action at each step and then performs **Action Grounding** to identify the corresponding HTML element.

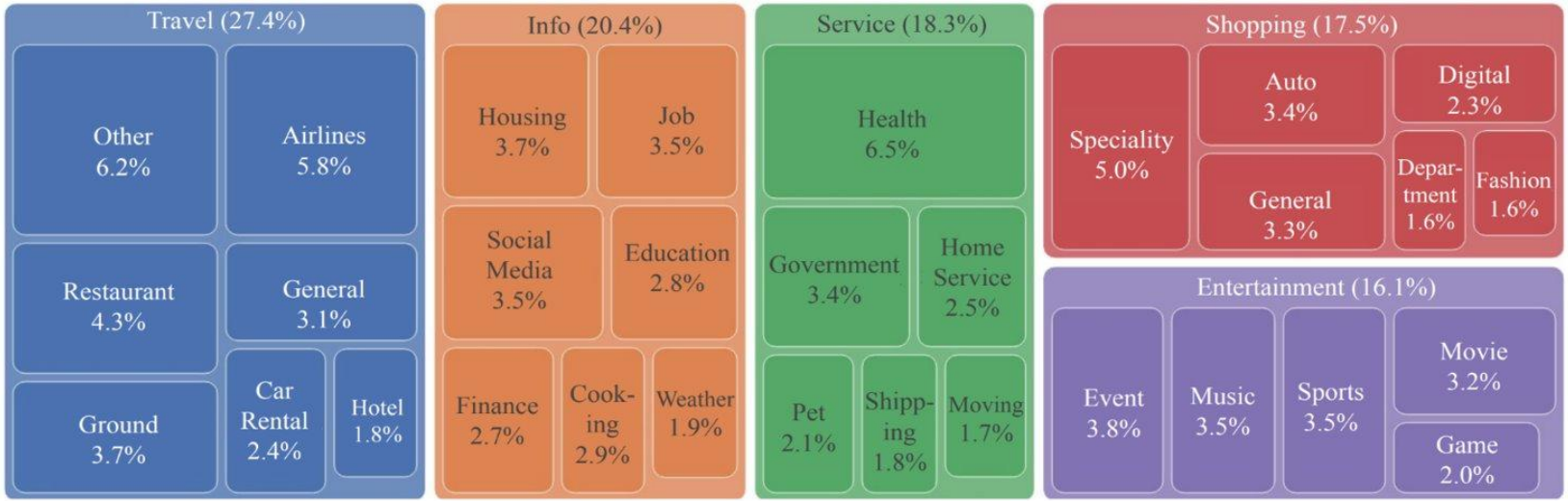


Mind2Web, a Generalist Web Agent Benchmark

MIND2WEB: Towards a Generalist Agent for the Web

Xiang Deng* **Yu Gu** **Boyuan Zheng** **Shijie Chen**
Samuel Stevens **Boshi Wang** **Huan Sun*** **Yu Su***
The Ohio State University
https://osu-nlp-group.github.io/Mind2Web

Mind2Web, a Generalist Web Agent Benchmark



Example tasks:

- Find one way flights from New York to Toronto
- Open page to schedule appointment for car knowledge test

Mind2Web, a Generalist Web Agent Benchmark

Agent	Model	Organization	Source	Easy	Medium	Hard	Average SR	Date
Navigator	n1-preview-11-2025	Yutori	Halluminate	90.1	76.2	71.1	78.7	2025-11-18
Google Cor	Gemini 2.5 Computer Use	Google DeepMind	Google DeepM	77.1	71.3	55.4	69.0	2025-9-29
Operator	OpenAI Computer-Using Agent	OpenAI	OSU NLP	83.1	58.0	43.2	61.3	2025-3-22
ACT-1-202	o3-2025-04-16 and Claude-sonnet-4-20250514	Enhans	Enhans	81.9	54.5	35.1	57.3	2025-8-23
Claude Cor	Claude-3-7-sonnet-20250219	Anthropic	OSU NLP	90.4	49.0	32.4	56.3	2025-4-20
ACT-1-202	o3-2025-04-16 and Claude-sonnet-4-20250514	Enhans	Enhans	65.1	46.2	23.0	45.7	2025-7-16
SeeAct	gpt-4o-2024-08-06	OSU	OSU NLP	60.2	25.2	8.1	30.7	2025-3-22
Browser U	gpt-4o-2024-08-06	Browser Use	OSU NLP	55.4	26.6	8.1	30.0	2025-3-22
Claude Cor	claude-3-5-sonnet-20241022	Anthropic	OSU NLP	56.6	20.3	14.9	29.0	2025-3-22
Agent-E	gpt-4o-2024-08-06	Emergence AI	OSU NLP	49.4	26.6	6.8	28.0	2025-3-22

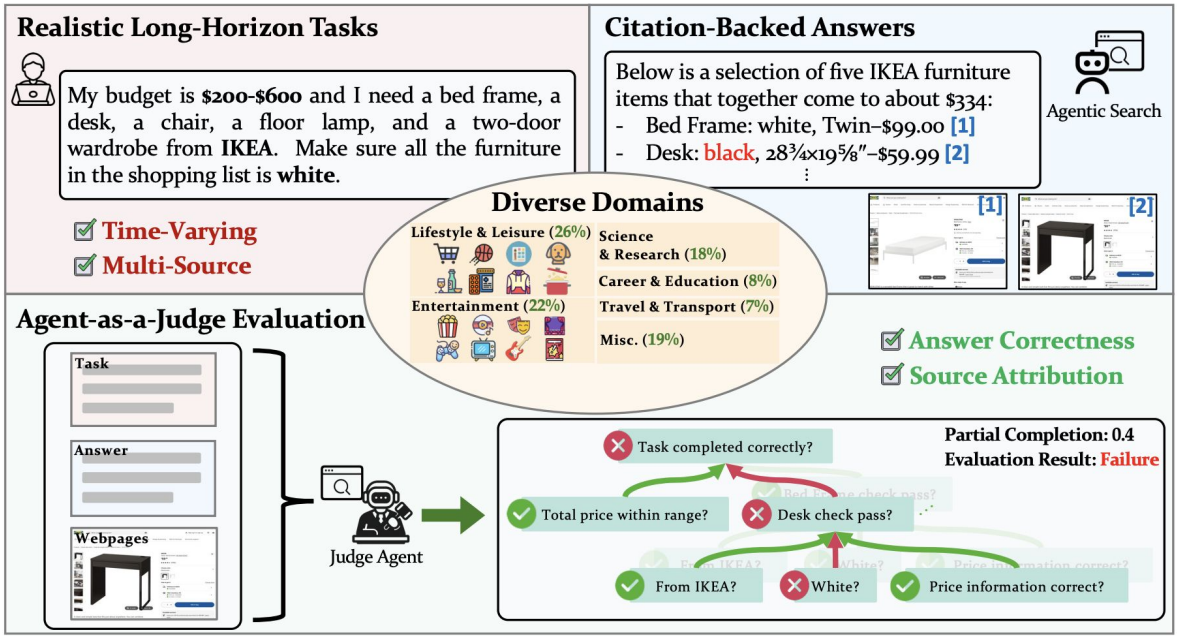
Mind2Web 2, Evaluating long horizon web agents tasks

Mind2Web 2:
Evaluating Agentic Search with Agent-as-a-Judge

Boyu Gou[1*] **Zanming Huang**[1*] **Yuting Ning**[1*] **Yu Gu**[1] **Michael Lin**[1]
Weijian Qi[1] **Andrei Kopanev**[1] **Botao Yu**[1] **Bernal Jiménez Gutiérrez**[1]
Yiheng Shu[1] **Chan Hee Song**[1] **Jiaman Wu**[1] **Shijie Chen**[1] **Hanane Nour Moussa**[1]
Tianshu Zhang[1] **Jian Xie**[1] **Yifei Li**[1] **Tianci Xue**[1] **Zeyi Liao**[1] **Kai Zhang**[1]
Boyuan Zheng[1] **Zhaowei Cai**[2] **Viktor Rozgic**[2] **Morteza Ziyadi**[2]
Huan Sun[1] **Yu Su**[1]
[1]The Ohio State University [2]Amazon AGI

https://osu-nlp-group.github.io/Mind2Web-2/

Mind2Web 2, Evaluating long horizon web agents tasks



Mind2Web 2, Evaluating long horizon web agents tasks

Agent	Date	Partial Completion ↓	Success Rate	Pass@3	Time (min)	Answer Length
Human	*2025-06-26*	*0.79*	*0.54*	*0.83*	*18.40*	*186*
Tencent Youtu Deep Research	2025-10-23	**0.60**	0.27	-	-	805
OpenAI Deep Research	2025-06-26	0.54	**0.28**	**0.40**	8.40	559
Grok DeeperSearch	2025-06-26	0.52	0.27	**0.40**	5.72	1362
Gemini Deep Research	2025-06-26	0.45	0.18	0.30	7.38	3357
Perplexity Deep Research	2025-06-26	0.42	0.15	0.26	5.67	585
Grok DeepSearch	2025-06-26	0.40	0.18	0.36	2.58	1428
Claude Research	2025-07-03	0.32	0.10	0.19	7.39	742
Perplexity Pro Search	2025-06-26	0.28	0.08	0.12	<1	408
HF Open Deep Research (w/ o3)	2025-06-26	0.26	0.11	0.18	13.65	209
ChatGPT Search	2025-06-26	0.26	0.06	0.11	<1	314
OpenAI Operator	2025-06-26	0.26	0.10	0.17	9.74	160

SkillWeaver: Enabling Self-Improving Web Agents

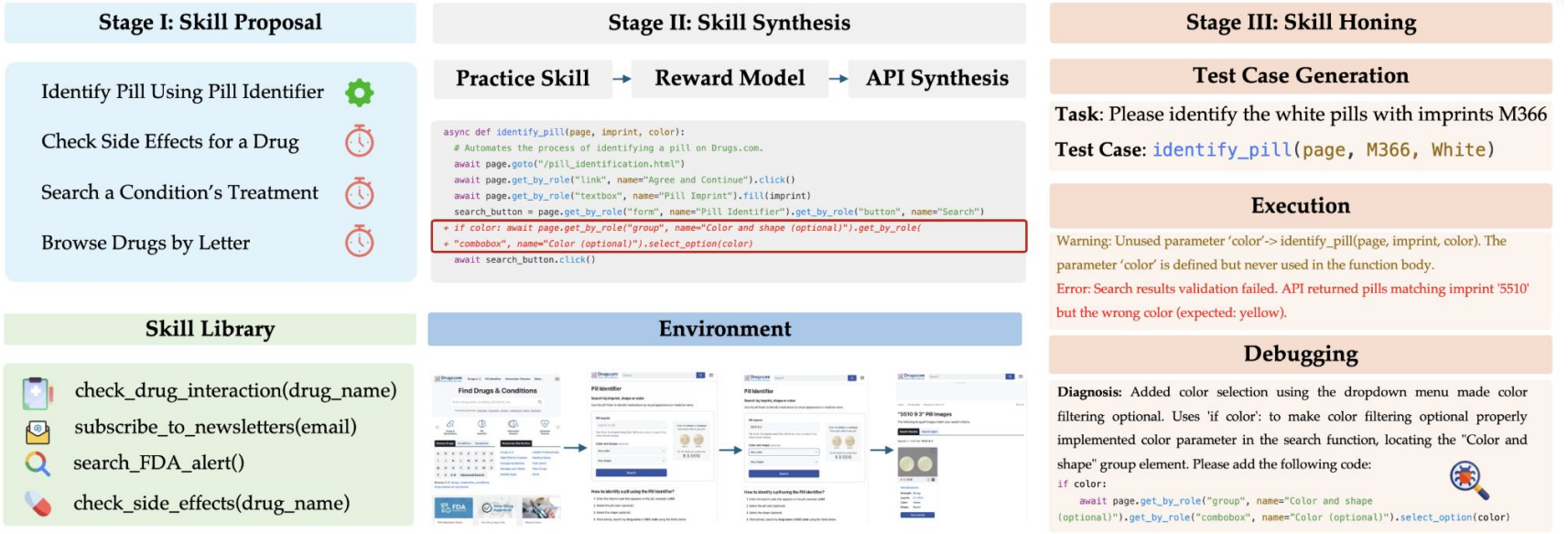
SkillWeaver: Web Agents can Self-Improve by Discovering and Honing Skills

Boyuan Zheng[1*], Michael Y. Fatemi[2*], Xiaolong Jin[3*],
Zora Zhiruo Wang[4], Apurva Gandhi[4], Yueqi Song,[4], Yu Gu[1], Jayanth Srinivasa[5], Gaowen Liu[5],
Graham Neubig[4], Yu Su[1]

[1] The Ohio State University [2] University of Virginia [3] Purdue University
[4] Carnegie Mellon University [5] Cisco Research
{zheng.2372, su.809}@osu.edu

SkillWeaver: Enabling Self-Improving Web Agents

SkillWeaver collects skills as Python functions as it learns to use websites, and then uses those collected skills to improve its performance on future tasks through a purely self-driven curriculum



AI agents applications and ongoing research

Science Agents

AI agent designed to **assist with or automate steps of the scientific process (e.g. literature review, hypothesis generation, etc.)**

Active areas of research:

- Improving open-source models for agentic scientific tasks
- Creating verifiable environments to train scientific agents via reinforcement learning
- Safety of agentic co-scientist systems
- Benchmarks and rigorous evaluation
- And so much more … **Feel free to talk to me if interested!**

Agent for Chemistry

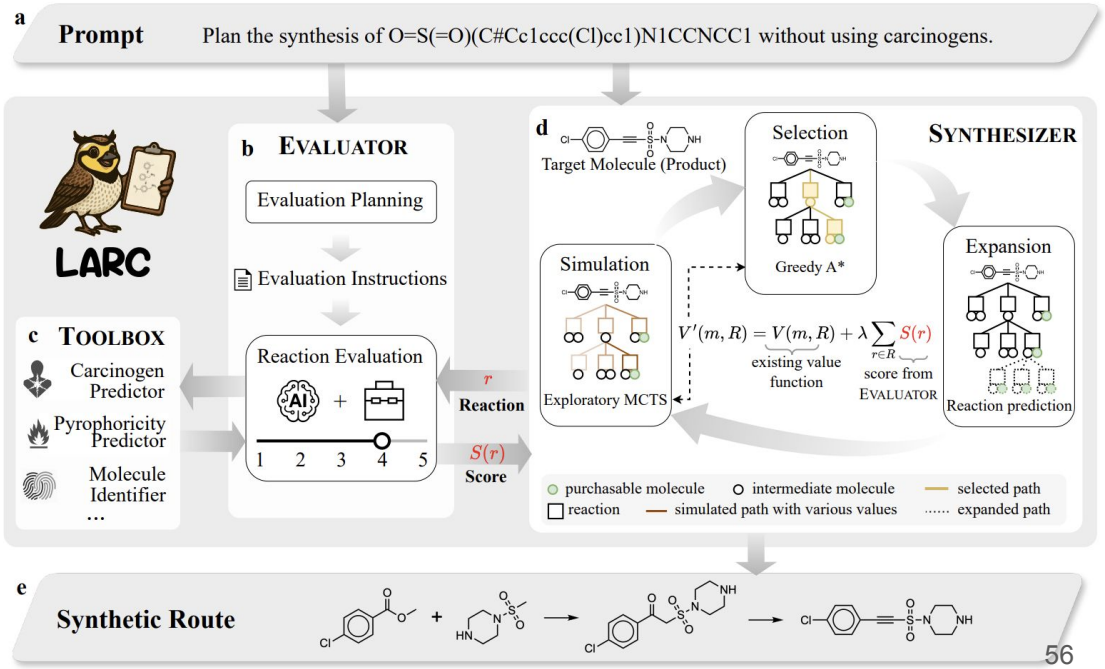
LARC: Towards Human-level Constrained Retrosynthesis Planning through an Agentic Framework

Frazier N. Baker[1], **Daniel Adu-Ampratwum**[2], **Reza Averly**[1], **Botao Yu**[1], **Huan Sun**[1] **& Xia Ning**[1,2,3,4]
[1]Department of Computer Science and Engineering,
[2]Division of Medicinal Chemistry & Pharmacognosy,
[3]Department of Biomedical Informatics,
[4]Translational Data Analytics Institute,
The Ohio State University, Columbus, OH, USA
{baker.3239,adu-ampratwum.1,averly.1,yu.3737,sun.397,ning.104}@osu.edu

Agent for Chemistry

LARC: An Agent for Constrained Retrosynthesis Planning (identifying synthetic routes from commercially available starting materials to desired target molecules, subject to practical constraints.)

Given a target molecule (i.e. product), LARC plans its synthetic routes that satisfy constraints specified by user prompts based on feedback grounded in tool-based reasoning



References

- The lecture material is based on the following resources:
  - Prof. Yu Su's substack: https://yusu.substack.com/p/language-agents
  - Prof. Yu Su's talks: https://llmagents-learning.org/slides/language_agents_YuSu_Berkeley.pdf

    & https://ysu1989.github.io/resources/language_agents_YuSu_2023.pdf
  - Stanford Agentic AI Webinar: https://www.youtube.com/watch?v=kJLiOGle3Lw
  - Anthropic Blogpost on Agentic Systems: https://www.anthropic.com/engineering/building-effective-agents
  - Lilian Weng blogpost on Autonomous AI Agents: https://lilianweng.github.io/posts/2023-06-23-agent/
  - Leonie Monigatti blogpost on AI Agent Memory: https://www.leoniemonigatti.com/blog/memory-in-ai-agents.html
  - And multiple papers references throughout the slides