The City College of New York

# CSC 59866-E: Senior Project I

## *AI Agents for Decision Making in the Real World*

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York



April 27, 2026 CSC 59866

# Advanced Topics: Software Coding Agents, Agentic RAG and Multimodal Embodied Audio-Vision-Language Agents

# Logistics & The State of the Class

**Recall Lecture 22:** We explored physical robotics (RT-2, pi_0.7) and autonomous driving (Alpamayo-R1).

**The Software/Digital Frontier:** Not all agents exist in physical space. Today, we examine agents that navigate the web, write software, and integrate audio-visual reasoning to act in human-centric digital environments.

# Today's Agenda

1. **Overcoming Hallucinations using ReAct**
2. **Looking at Software Engineering Agents (SWE-Bench + Reflexion)**
3. **Digital Embodiment (Mind2Web + Voyager)**
4. **Multimodality in the Physical World (Vision + Language + Audio Agents and PaLM Say-Can)**

# Agentic RAG & Reasoning

# How RAG Can Help with Hallucination

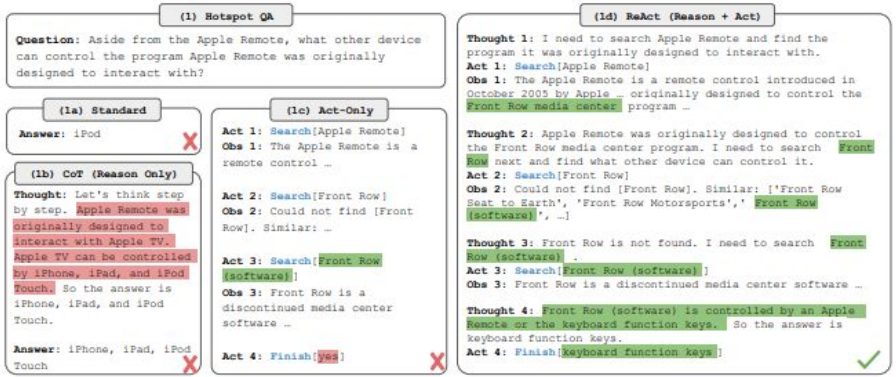
**Standard LLMs:** They generate text autoregressively based on static, pre-trained parametric memory.

**The Problem:** When faced with a multi-step problem or missing knowledge, standard LLMs "guess" (hallucinate).

**Retrieval-Augmented Generation (RAG):** The standard fix is to search a database *once* before prompting the LLM. But what if the task requires sequential, dynamic searching based on intermediate clues?

# ReAct: Synergizing Reasoning and Acting

- **ReAct (Yao et al., 2022):** A paradigm that interleaves internal reasoning traces with external actions.
- **The Loop:** Think -> Act -> Observe.
- **Reasoning traces:** Help the model induce, track, and update action plans, and handle exceptions.
- **Actions:** Allow the model to interface with external sources (Wikipedia, Search APIs, Code Interpreters) to gather missing information.



# ReAct Policies

We can formalize an agent that can act while “thinking” as a policy $\pi$ within a Partially Observable Markov Decision Process (Recall POMDPs from the earlier lecture).

Instead of mapping a context $c_t$ directly to an external action $a_t$, we expand the action space to include internal "thoughts" $t_t \in \mathcal{L}$ (where $\mathcal{L}$ is the language space).

**The Agent Policy takes in actions and “thoughts” and is conditioned on the context:**

$$\pi(a_t, t_t | c_t)$$

Where the context $c_t = (x, a_1, o_1, \ldots, a_{t-1}, o_{t-1})$ includes the original prompt $x$ and the history of actions and observations.

If we allow the action $a_t$ to be a function that executes a search using a query, we can augment the context $c_{t+1}$ with the discovered information to enhance the LLM Agent.

# Implementing Agentic RAG

Standard RAG is a static pipeline: `Retrieve -> Generate`.

**Agentic RAG** is dynamic:

1. **Thought:** "I need to find the CEO of Company X, then find their age."
2. **Action:** `Search("CEO of Company X")`
3. **Observation:** "John Doe"
4. **Thought:** "Now I need John Doe's age."
5. **Action:** `Search("John Doe age")`

The agent decides *when* to stop retrieving and synthesize the final answer.

# Software Coding Agents

# Challenges in Code Generation

Generating a standalone python script is easy. Resolving a bug in a massive, real-world repository is incredibly difficult.

**Dependencies:** Fixing a bug in utils.py might break a function in main.py.

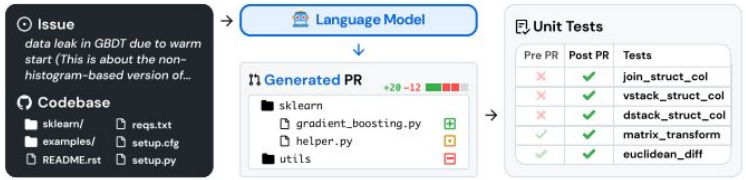
**Context Windows:** A repository like scikit-learn has millions of lines of code. You cannot simply stuff the whole repository into an LLM prompt.

# Coding Agent Evaluation through GitHub Issues

One of the earliest widespread benchmarks for coding LLM Agents was SWE-bench.

The agent is given real GitHub issues and is rewarded only if whatever patch it produces ends up working.

At the beginning, the best Agents solved a measly 2% of issues! Needless to say, coding Agents have come a long way since then.

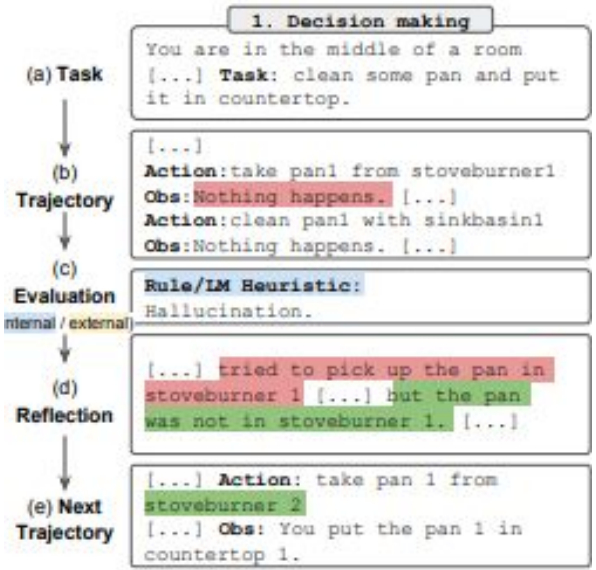


# Reflexion: Verbal RL

If an agent writes bad code, how does it learn from its mistake? Standard RL requires thousands of episodes and gradient updates.

**Reflexion (Shinn et al., 2023):** Replaces weight updates with *linguistic feedback*.

When a software agent's code fails a compiler test, a "Reflection" model reads the error trace and generates a verbal self-critique (e.g., "I forgot to import numpy. In the next trial, I must import it.").



Traditional RL optimizes weights $\theta$ to maximize expected reward $R$ via gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t)$$

**Reflexion** freezes the weights $\theta$ and optimizes the *Context Window* ($c_t$) using a linguistic memory buffer $mem_t$

The policy is conditioned on memory: $a_t \sim \pi_\theta(a_t | c_t, mem_t)$

**The Memory Update Rule:**

$$mem_{t+1} = M(mem_t, a_t, o_t, r_t)$$

Where $M$ is the LLM generating a verbal critique based on the last action $a_t$, observation $o_t$ (e.g., a traceback error), and scalar reward $r_t$.

# Digital Embodiment (Web & Games)

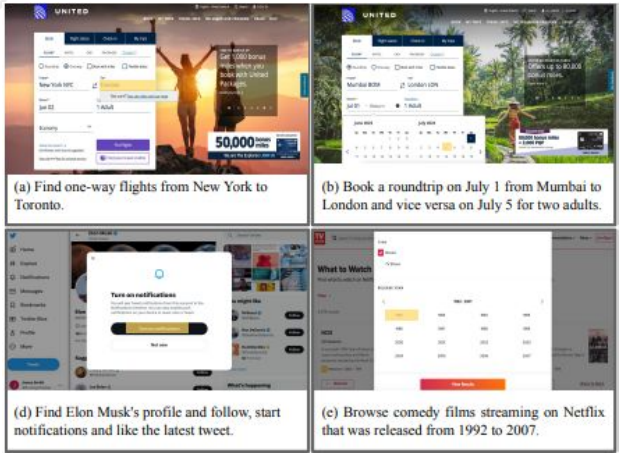
# Mind2Web: Agents on the Internet

The internet is the ultimate digital environment.

**Mind2Web (Deng et al., 2023):** A framework for generalist web agents that follow natural language instructions across diverse real-world websites.

**The Challenge:** Modern websites exceed 100k HTML tokens.

**The Solution:** The agent uses an HTML filtering module to extract only the actionable DOM elements (buttons, inputs) before mapping instructions to the Action Space: Click(element), Type(element, text), or Select(element, option).

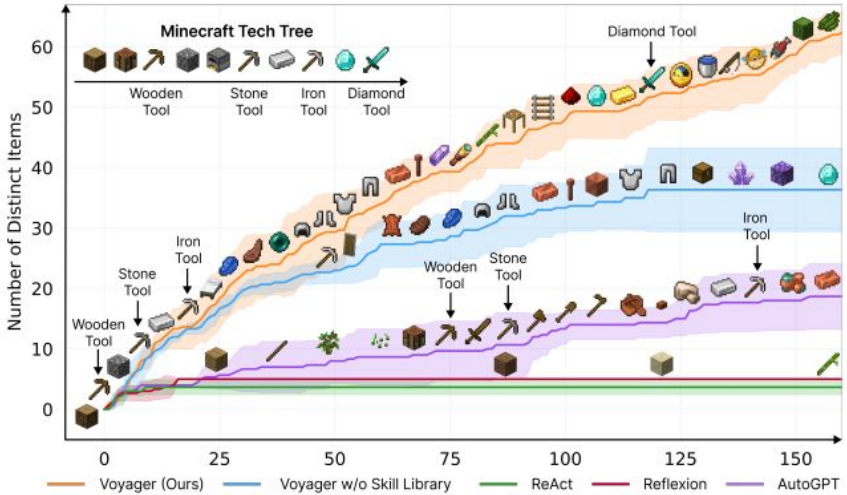


# Voyager: Open-Ended Embodiment

Can an agent survive and invent in an open-world game without human intervention?

**Voyager (Wang et al., 2023):** The first LLM-powered embodied lifelong learning agent in Minecraft.

It uses GPT-4 via API calls. It does not train a neural network to move the mouse; it generates Javascript code (Mineflayer API) to control the avatar.



# Continual Learning and Skill Libraries

Standard agents suffer from *catastrophic forgetting*. Voyager solves this using a **Skill Library**.

**Automatic Curriculum:** The agent proposes its own tasks ("I should gather wood before trying to mine stone").

**Code as Skills:** When it successfully writes code to craft a pickaxe, it saves that function as an embedding in a vector database.

To build a house, it retrieves `craft_pickaxe()` from memory, demonstrating compositional, temporally extended reasoning.

# Physical Multimodality

# The Affordance Problem in the Real World

An LLM might reason: *"To clean a spill, use a sponge."*

But what if there is no sponge in the room? Or what if the sponge is too heavy for the robot's specific motors?

**The Gap:** Language models lack real-world physical grounding. They do not understand **Affordances** (the physical possibilities of an object relative to the robot's body).

# PaLM-SayCan: Do As I Can, Not As I Say

**PaLM-SayCan (Ahn et al., 2022):** Grounds high-level language reasoning in low-level robotic affordances.

**"Say":** The LLM proposes a sequence of useful actions based on semantic knowledge.

**"Can":** Pre-trained reinforcement learning value functions look through the robot's cameras and determine if those actions are physically possible right now.

# Audio-Vision-Language (AVL) Agents

Modern agents are moving beyond just text and images to include **Audio**.

**Why Audio?**

- Acoustic scene classification (e.g., an autonomous car hearing an ambulance siren before it sees it).
- Emotion and prosody detection in human voice commands (detecting urgency).

**Unified Architectures:** Models can natively process Audio, Vision, and Language in the same latent space, allowing agents to correlate the *sound* of glass breaking with the *visual* of a spilled drink, and *speak* a response.

# Summary

**Agentic RAG (ReAct):** Prevents hallucinations by interleaving thought with active retrieval tools.

**Coding Agents (SWE-bench/Reflexion):** Require long-context understanding and utilize verbal feedback loops to fix code errors.

**Digital Embodiment (Mind2Web/Voyager):** Agents can navigate HTML DOM trees or build executable skill libraries to survive in open-ended games.

**Multimodality (SayCan):** Language must be grounded in physical affordances to act safely in the real world.

# Questions?