Introduction to AI Agents

January 29, 2026

Scalable Computational Intelligence Group (SCI)

Texas Advanced Computing Center (TACC)

Outline

1. Background on LLMs
2. What even *is* an agent?
3. Examples of agentic systems

LLM is just text in and text out

Gemma 2 - Instruct

- LLMs convert words or partial words to ids called

```
<start_of_turn>user
What is Cramer's Rule?<end_of_turn>
<start_of_turn>model
Cramer's Rule is ...<end_of_turn>
```

mum

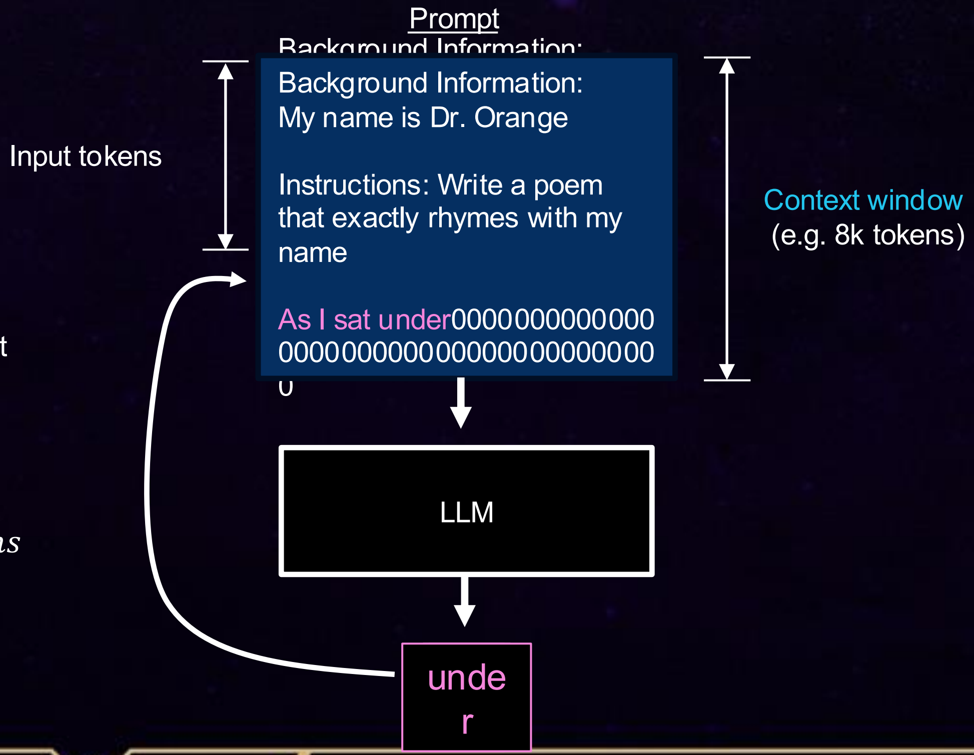
t

- LLM generation is recursive, so your output token amount is limited to:

- Mutli-turn conversations are just recorded as one long prompt with special tokens demarcating user and agent messages

- The “Instruct” versions of models have been fine-tuned to understand this convention

$$\max output\ tokens = context\ window\ - input\ tokens$$



Prompt

Background Information:

Input tokens

Background Information:
My name is Dr. Orange

Instructions: Write a poem that exactly rhymes with my name

As I sat under0000000000000000000000000000000000000
0

Context window
(e.g. 8k tokens)

LLM

unde
r

How fast can I run my model?

A little faster than you can read!

Model Family	# of Parameters	iPhone 16 Pro Max tokens/s	Macbook Pro tokens/s	Vista GPU node tokens/s	Tejas tokens/s
Qwen3	1.7 billion	13	73	313	-
Llama3.1	8 billion	-	25	173	1001
Qwen3	30 billion	-	36	130	-
DeepSeek-R1	70 billion	-	-	36	-
DeepSeek-R1	671 billion	-	-	-	135

Time to generate 100,000 tokens for agentic workloads:

4 hours
40 minutes
10 minutes
2 minutes

Laptop is ~6x faster than phone
1 Vista GPU is ~4x faster than laptop
Tejas is ~5x faster than 1 Vista GPU

Categories of tasks an LLM can accomplish

- Simple task examples
  - Summarize this text: {text}
  - Classify the sentiment of this statement as positive or negative: {statement}
  - Write a python script to implement a binary search of an array

- Complex task example
  - Reproduce the analytical model from the following paper, verify that it matches their results and then apply that model to the following data:

    Paper Text
    {paper_text}

    Data
    {data}

  - requires multiple steps
  - use of other software (web search, image analysis etc.)
  - math

[1] G. Mialon et al. "Gaia: a benchmark for general ai assistants" (2023) arXiv:2311.12983

Outline

1. Background on LLMs
2. What even *is* an agent?
3. Examples of agentic systems

What are AI agents?

AI Agent Definition: “An artificial entity capable of perceiving its surroundings using sensors, making decisions, and then taking actions in response using actuators.”

- The decision-making core of modern AI agents is either a reinforcement learning (RL) algorithm, or a LLM
- **RL-based agents** can struggle with complex tasks and their decision making can be inscrutable
    - e. g. in *long-horizon tasks* where the rewards during policy search are rare, the training times become extreme
- **LLM-based agents** utilize *goal decomposition to* break down goals into smaller subgoals
    - LLM can act as a planner
    - decides what action to take for a given subgoal
    - actions can be simple LLM tasks or external API calls to other software

Z. Xi et al. “The Rise and Potential of Large Language Model Based Agents: A Survey” *arXiv* (2023)

Flow diagram of an agent: Perceive, Plan, and Act

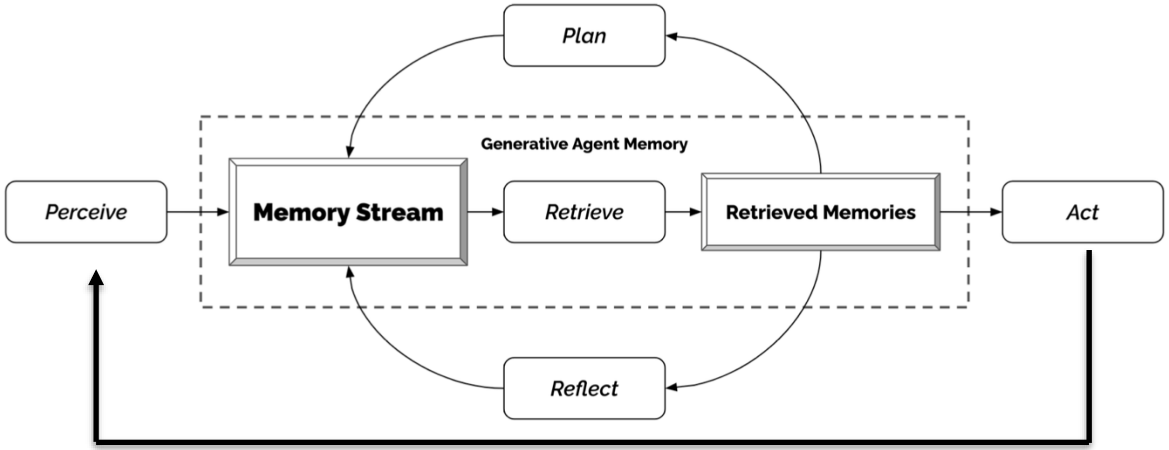


Figure 5: Our generative agent architecture. Agents perceive their environment, and all perceptions are saved in a comprehensive record of the agent's experiences called the memory stream. Based on their perceptions, the architecture retrieves relevant memories and uses those retrieved actions to determine an action. These retrieved memories are also used to form longer-term plans and create higher-level reflections, both of which are entered into the memory stream for future use.

J. S. Park et al. "Generative Agents: Interactive Simulacra of Human Behavior" *UIST* (2023)

Key functionalities an agent needs

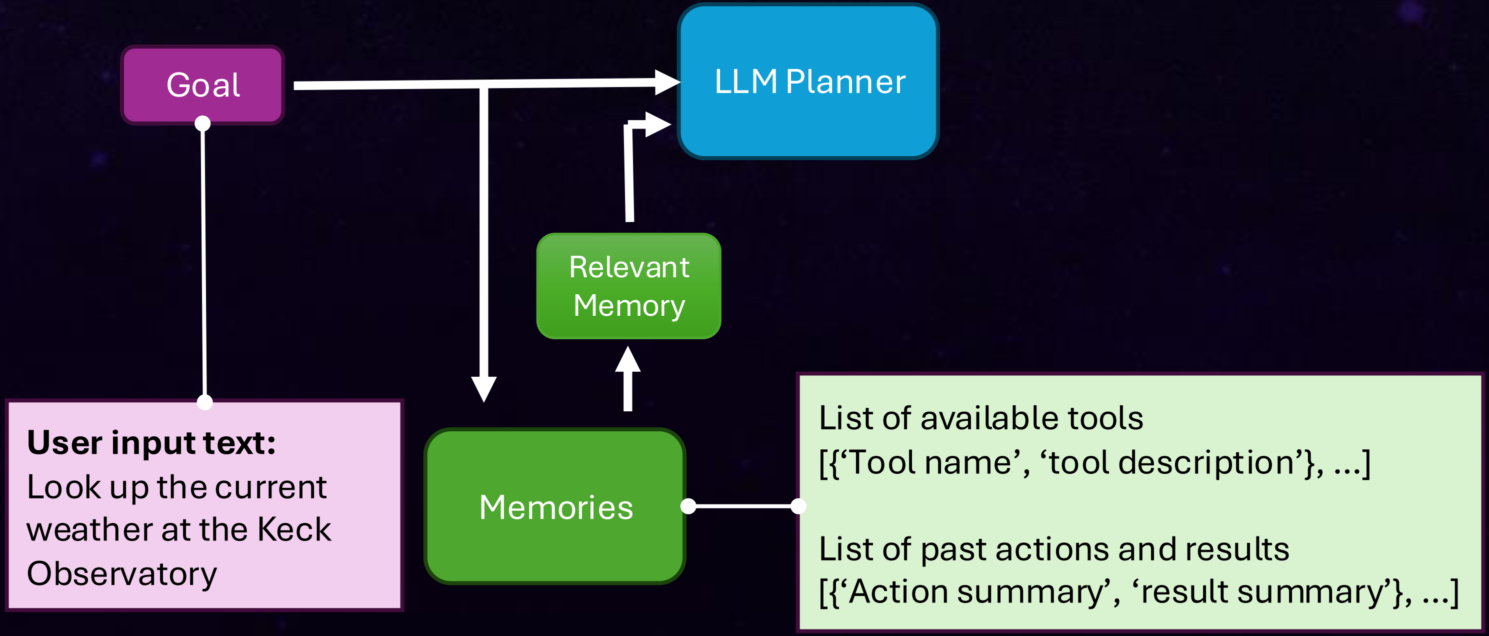
1. Planning and/or Reasoning through goal decomposition
    - Goal decomposition: LLM writes a todo list
    - ReAct: LLM iteratively decided next best step

2. Acting through "tool" use
    - LLM writes a command to trigger another piece of software

3. Sensing current state of surroundings
    - automatic snapshot of agent working area using software

4. Memory retrieval
    - Vector store/search
    - Retrieval augmented generation (RAG)

LLM agent - goal decomposition and tool use example



Goal

LLM Planner

Relevant Memory

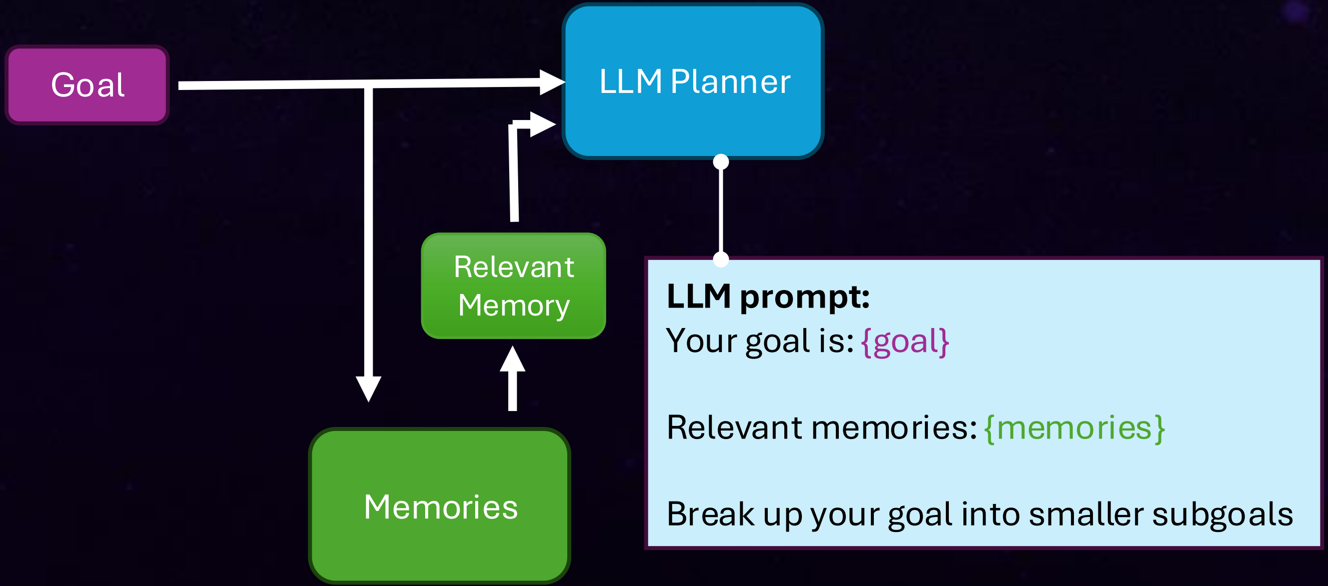
Memories

User input text:
Look up the current weather at the Keck Observatory

List of available tools
[{'Tool name', 'tool description'}, ...]

List of past actions and results
[{'Action summary', 'result summary'}, ...]

LLM agent - goal decomposition and tool use example

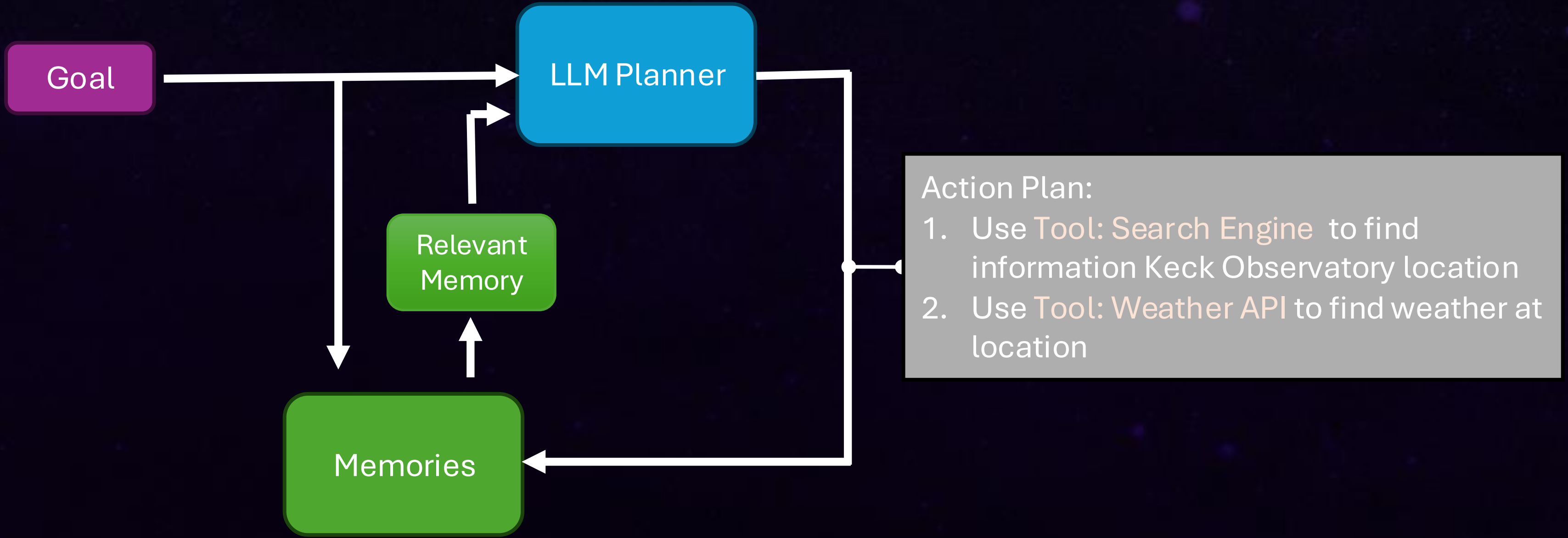


LLM prompt:
Your goal is: {goal}

Relevant memories: {memories}

Break up your goal into smaller subgoals

LLM agent - goal decomposition and tool use example



Goal

LLM Planner

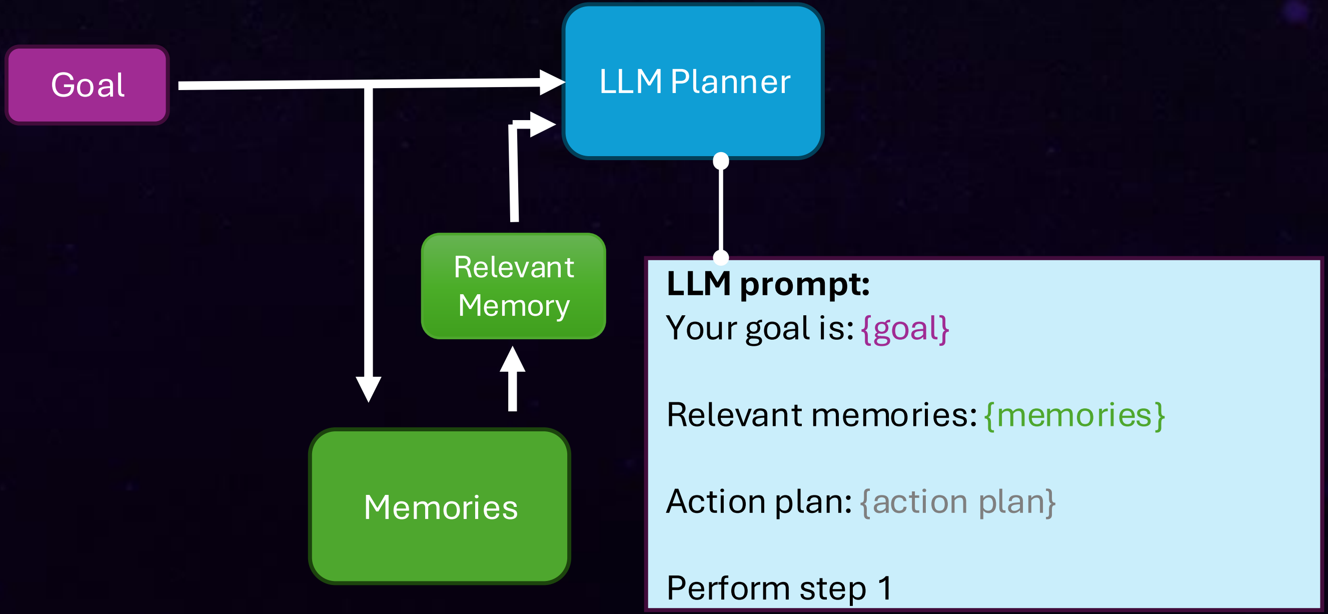
Relevant Memory

Memories

Action Plan:

1. Use Tool: Search Engine to find information Keck Observatory location
2. Use Tool: Weather API to find weather at location

LLM agent - goal decomposition and tool use example



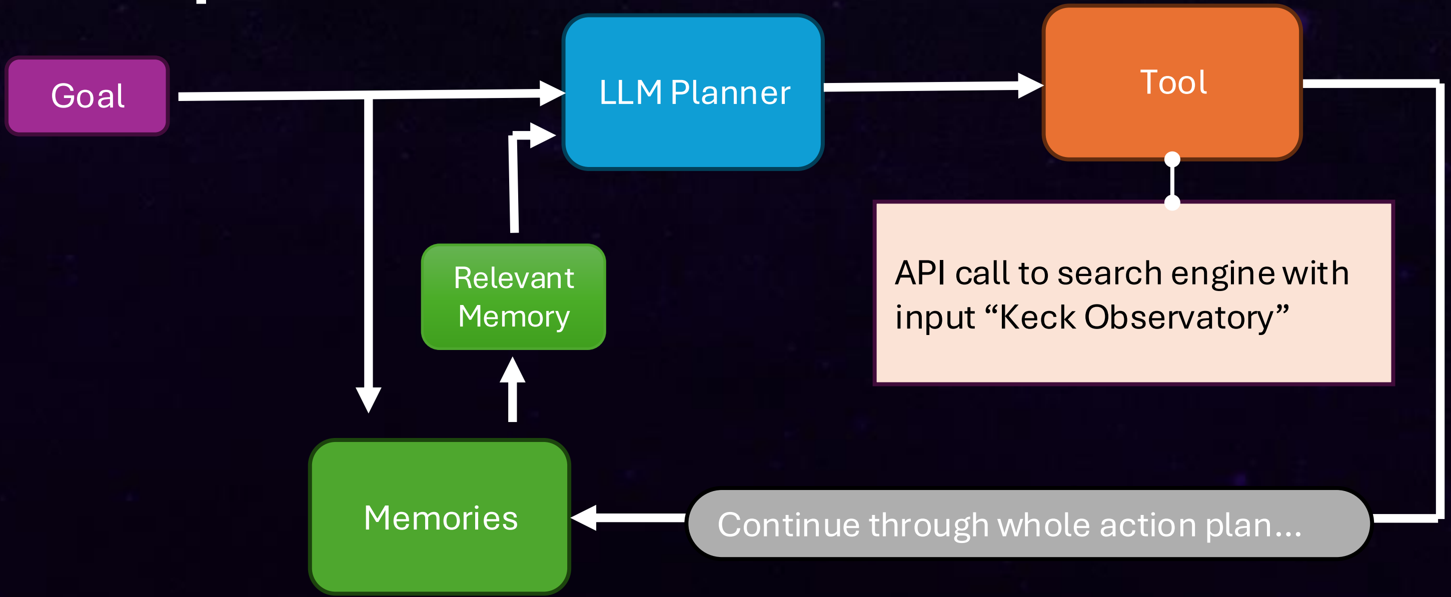
LLM prompt:
Your goal is: {goal}

Relevant memories: {memories}

Action plan: {action plan}

Perform step 1

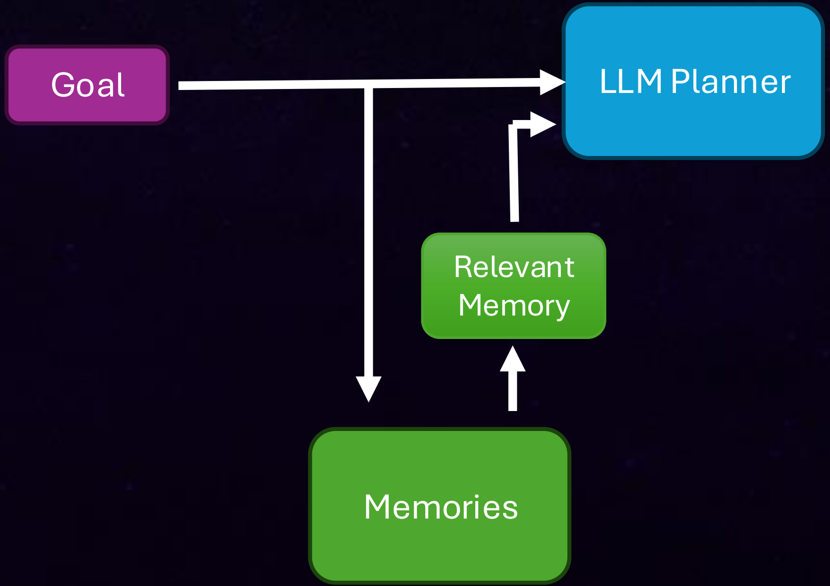
LLM agent - goal decomposition and tool use example



API call to search engine with input “Keck Observatory”

Continue through whole action plan…

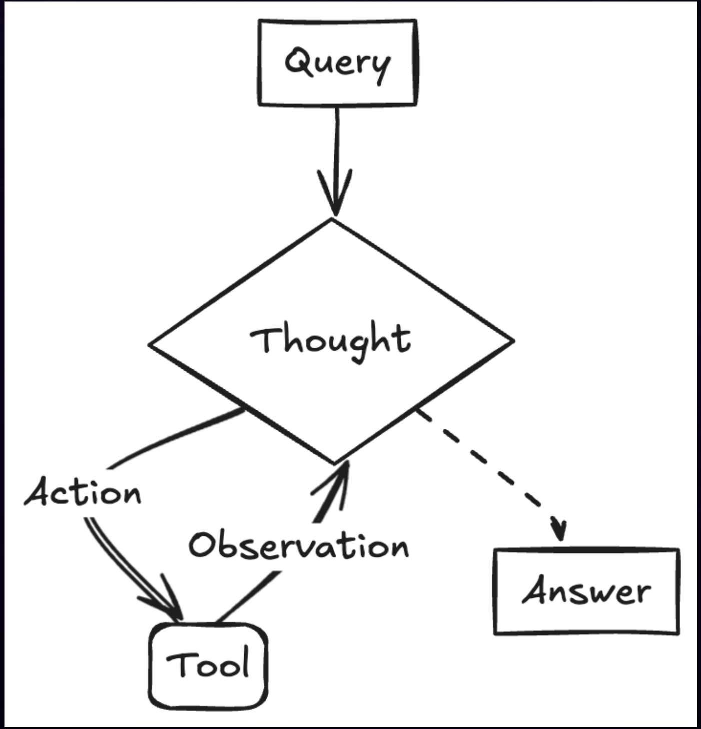
LLM agent - goal decomposition and tool use example



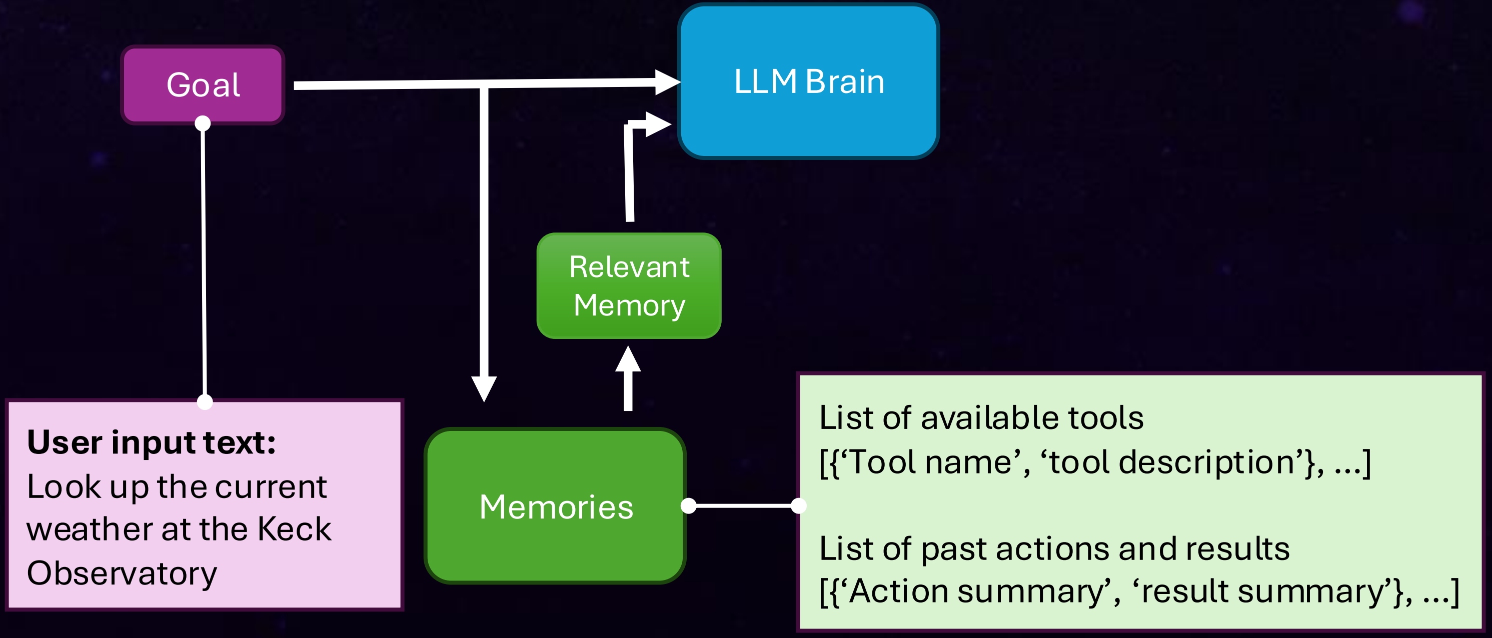
Final output text:
Keck Observatory is located on the summit of Mauna Kea where the current weather is 16°C

ReAct

- ReAct agents are AI systems that combine reasoning and action to solve problems by analyzing tasks and interacting with their environment.
- They operate in a loop of reasoning, acting, and observing, allowing for dynamic decision-making and real-time adjustments based on feedback.
- Contrasting ReAct agents with goal decomposition, ReAct agents only think 1 step ahead when reasoning. If they choose to execute an action they next ingest those results and reason on the next best step.



LLM agent – ReAct and tool use example



Goal

LLM Brain

Relevant Memory

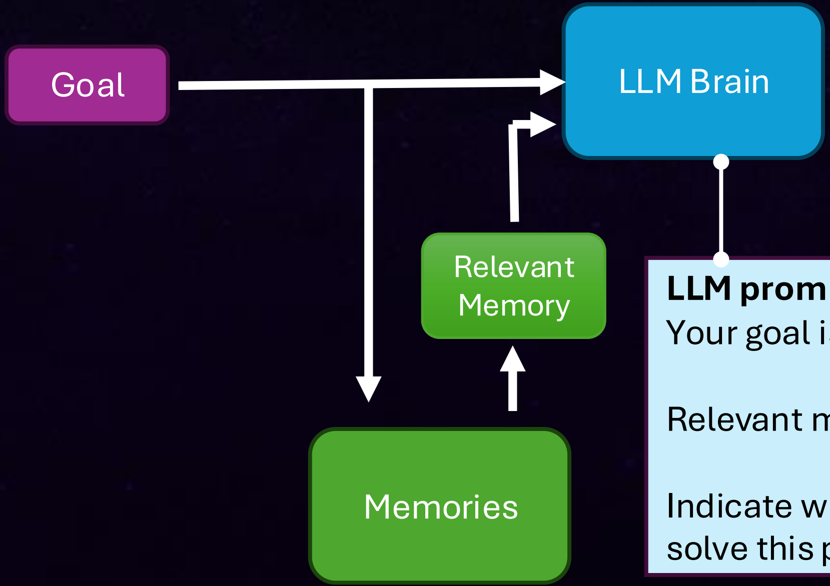
Memories

User input text:
Look up the current weather at the Keck Observatory

List of available tools
[{‘Tool name’, ‘tool description’}, …]

List of past actions and results
[{‘Action summary’, ‘result summary’}, …]

LLM agent – ReAct and tool use example

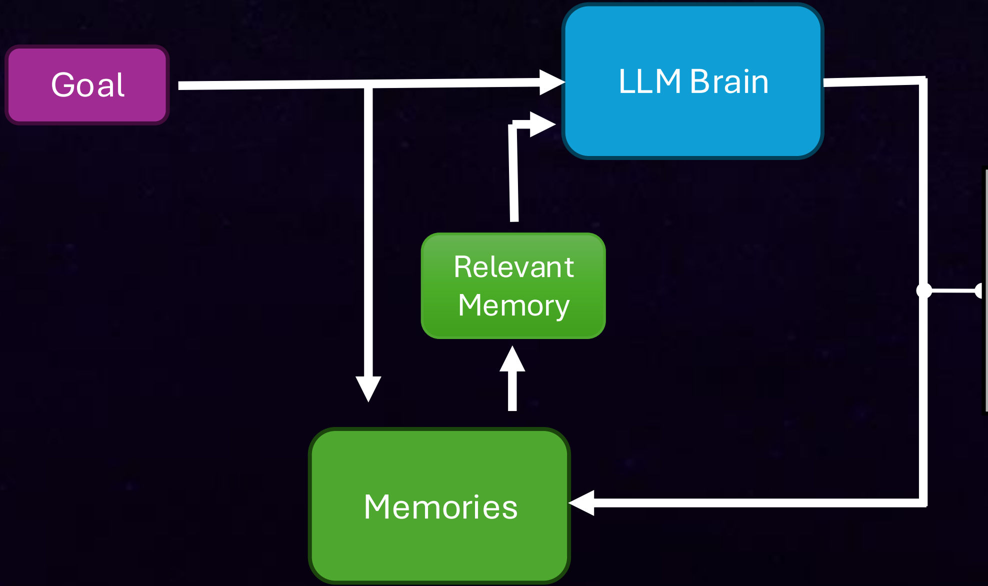


LLM prompt:
Your goal is: {goal}

Relevant memories: {memories}

Indicate what you should do next to solve this problem

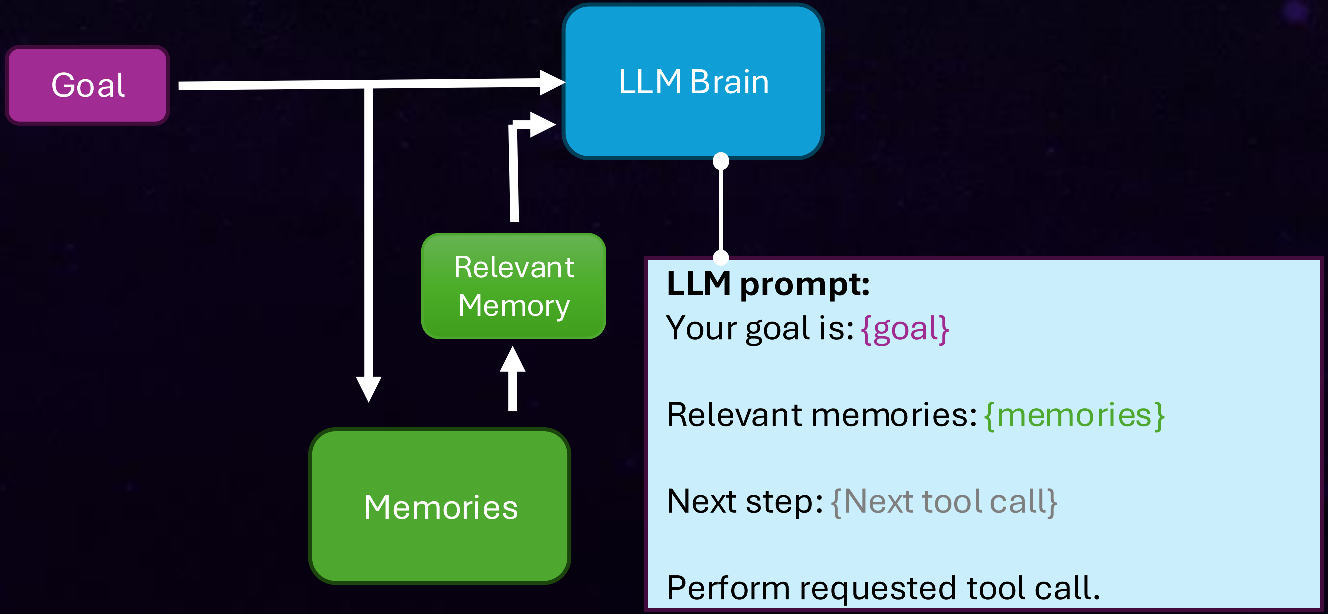
LLM agent – ReAct and tool use example



Next Step:

1. Use Tool: Search Engine to find information Keck Observatory location

LLM agent – ReAct and tool use example



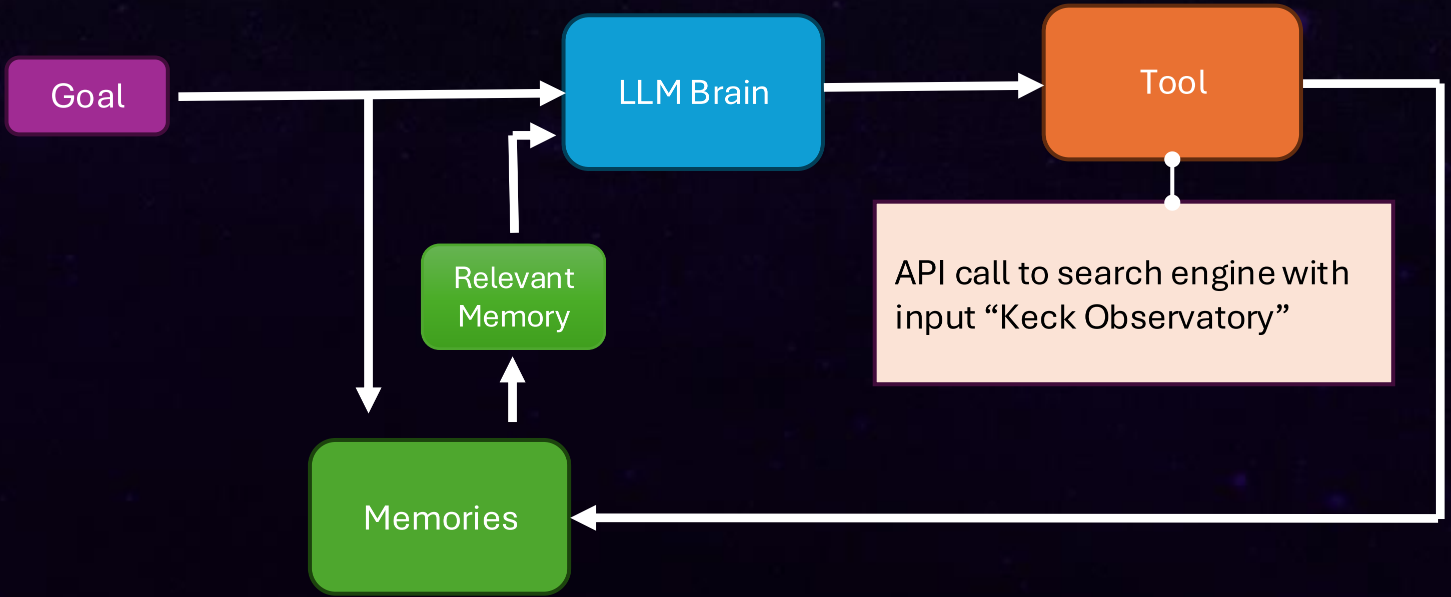
LLM prompt:
Your goal is: {goal}

Relevant memories: {memories}

Next step: {Next tool call}

Perform requested tool call.

LLM agent – ReAct and tool use example



Goal

LLM Brain

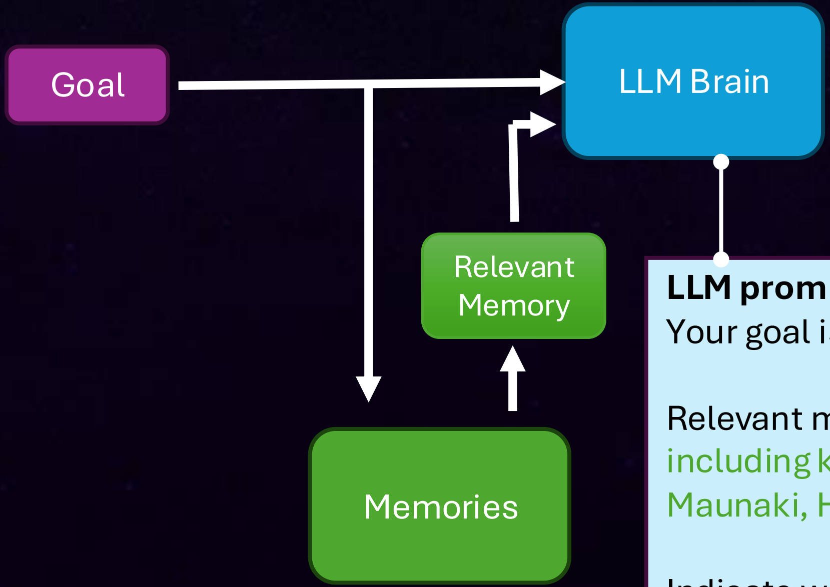
Tool

Relevant Memory

Memories

API call to search engine with input “Keck Observatory”

LLM agent – ReAct and tool use example

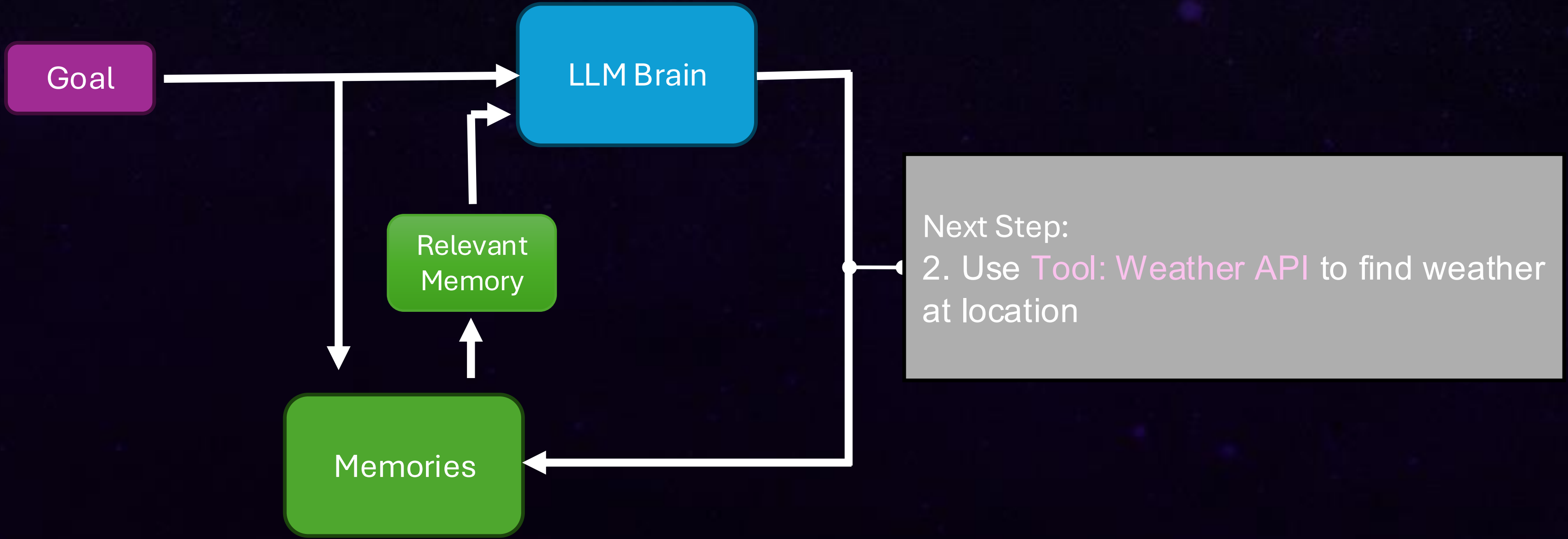


LLM prompt:
Your goal is: {goal}

Relevant memories: {memories… including keck observatory is located in Maunaki, Hawaii}

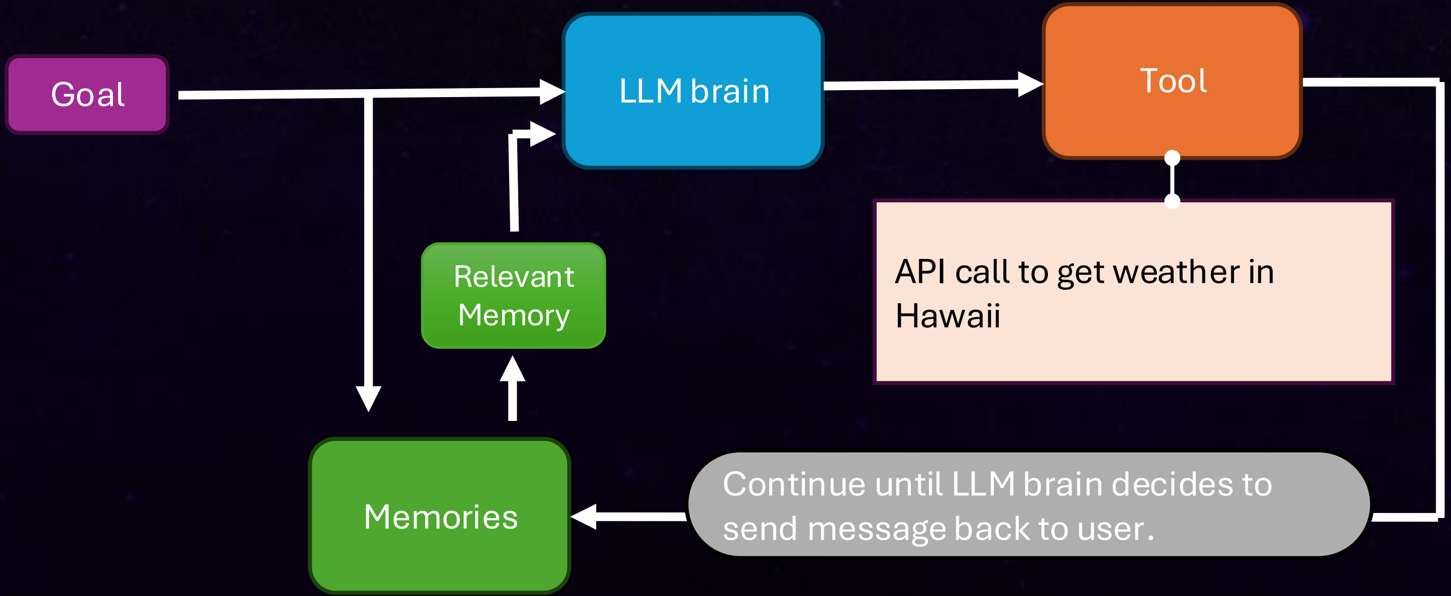
Indicate what you should do next to solve this problem

LLM agent – ReAct and tool use example



Next Step:
2. Use Tool: Weather API to find weather at location

LLM agent – ReAct and tool use example


Goal

LLM brain

Tool

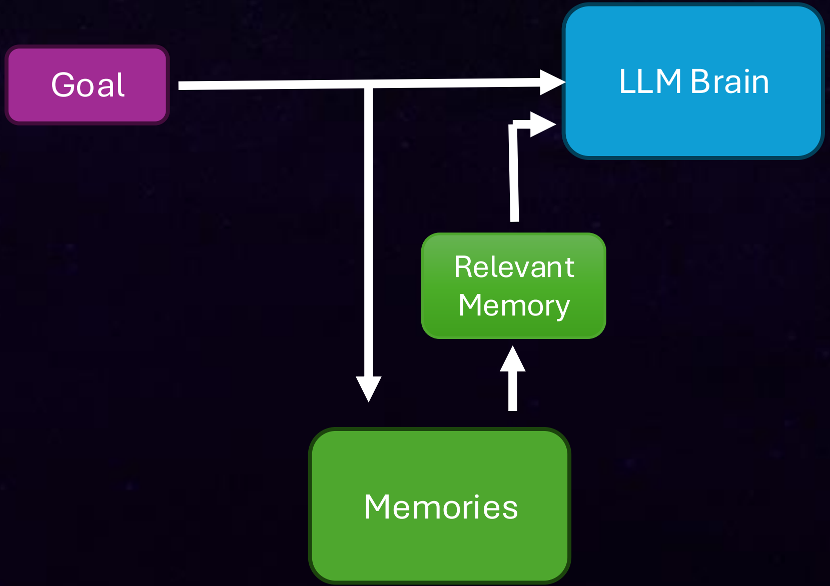
API call to get weather in Hawaii

Relevant Memory

Memories

Continue until LLM brain decides to send message back to user.

LLM agent – ReAct and tool use example



Final output text:
Keck Observatory is located on the summit of Mauna Kea where the current weather is 16°C

Outline

1. Background on LLMs
2. What even *is* an agent?
3. Examples of agentic systems

Comparing RL and LLM agents attempting the collect diamond challenge in Minecraft

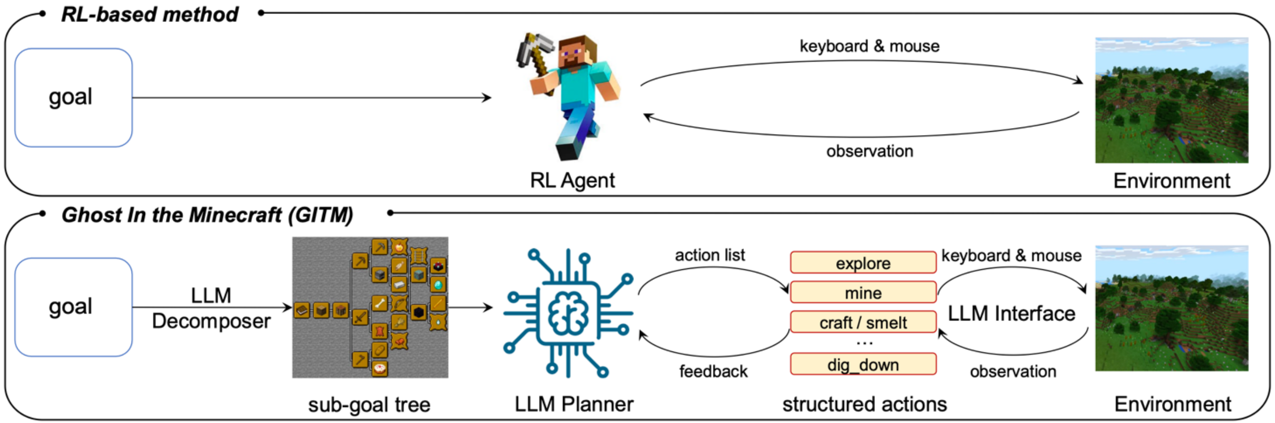
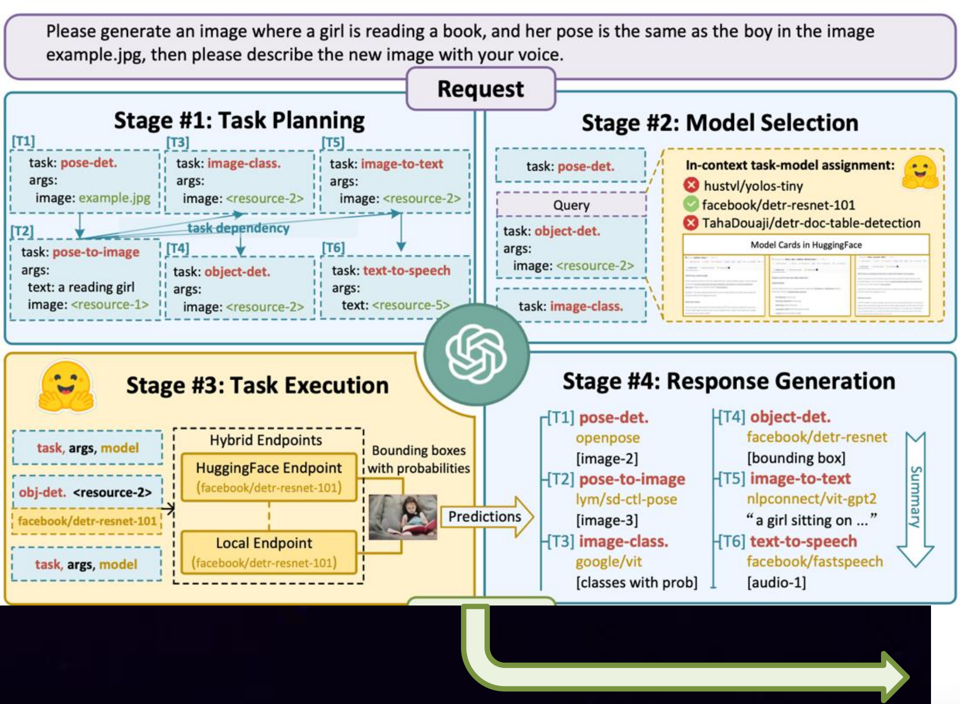


Figure 2: **Comparison between RL-based method and our GITM.** RL agents try to map an complex goal directly to a sequence of low-level control signals, while our GITM leverages LLM to break down the goals and map them to structured actions for final control signals.

X. Zhu et al., "Ghost in the Minecraft" *arXiv* (2023)

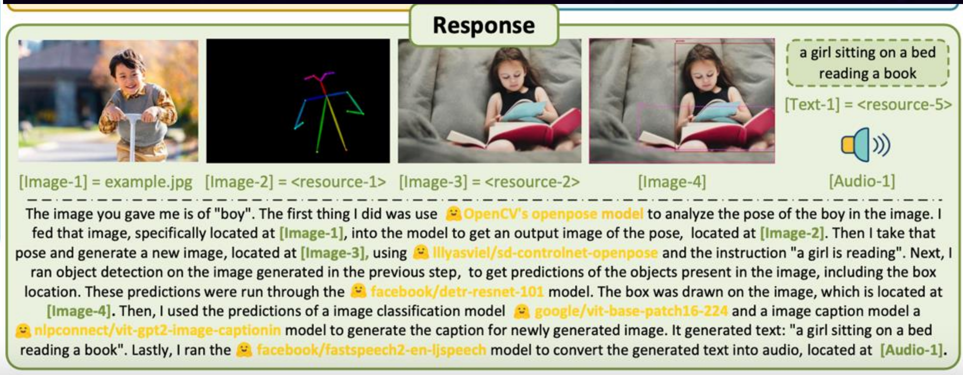
Extreme tool use example: HuggingGPT

- An LLM-agent that looks up models on huggingface repo site
- Attempts to solve sub goals via api calls to these models
- Concatenates final results in the response



Please generate an image where a girl is reading a book, and her pose is the same as the boy in the image example.jpg, then please describe the new image with your voice.

Request
Stage #1: Task Planning
[T1] task: pose-det. args: image: example.jpg
[T3] task: image-class. args: image: <resource-2>
[T5] task: image-to-text args: image: <resource-2>
task dependency
[T2] task: pose-to-image args: text: a reading girl image: <resource-1>
[T4] task: object-det. args: image: <resource-2>
[T6] task: text-to-speech args: text: <resource-5>
Stage #2: Model Selection
task: pose-det.
Query
task: object-det. args: image: <resource-2>
task: image-class.
In-context task-model assignment:
hustvl/yolos-tiny
facebook/detr-resnet-101
TahaDouaji/detr-doc-table-detection
Model Cards in HuggingFace
Stage #3: Task Execution
task, args, model
obj-det. <resource-2>
facebook/detr-resnet-101
task, args, model
Hybrid Endpoints
HuggingFace Endpoint (facebook/detr-resnet-101)
Local Endpoint (facebook/detr-resnet-101)
Bounding boxes with probabilities
Predictions
Stage #4: Response Generation
[T1] pose-det. openpose [image-2]
[T2] pose-to-image lym/sd-ctl-pose [image-3]
[T3] image-class. google/vit [classes with prob]
[T4] object-det. facebook/detr-resnet [bounding box]
[T5] image-to-text nlpconnect/vit-gpt2 "a girl sitting on ..."
[T6] text-to-speech facebook/fastspeech [audio-1]
Summary

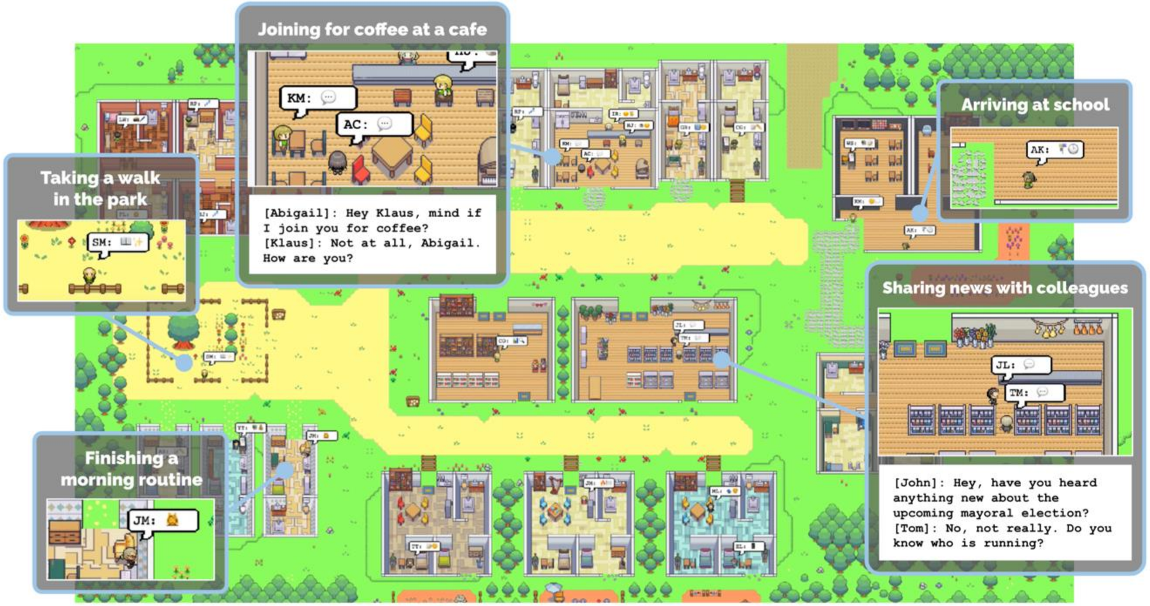


The image you gave me is of "boy". The first thing I did was use OpenCV's openpose model to analyze the pose of the boy in the image. I fed that image, specifically located at [Image-1], into the model to get an output image of the pose, located at [Image-2]. Then I take that pose and generate a new image, located at [Image-3], using lllyasviel/sd-controlnet-openpose and the instruction "a girl is reading". Next, I ran object detection on the image generated in the previous step, to get predictions of the objects present in the image, including the box location. These predictions were run through the facebook/detr-resnet-101 model. The box was drawn on the image, which is located at [Image-4]. Then, I used the predictions of a image classification model google/vit-base-patch16-224 and a image caption model a nlpconnect/vit-gpt2-image-captionin model to generate the caption for newly generated image. It generated text: "a girl sitting on a bed reading a book". Lastly, I ran the facebook/fastspeech2-en-ljspeech model to convert the generated text into audio, located at [Audio-1].

Y. Shen et al. "HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face" *arXiv* (2023)

AI agents controlling multiple non-player characters in a "Sims-like" simulation

- Multi-agent simulation to study social dynamics and information transfer
- Each agent is initialized with a backstory in its memories
- Agents plan their schedules, move around, and interact with the environment and other characters through dialogs


Joining for coffee at a cafe
KM:
AC:
Taking a walk in the park
SM:
Arriving at school
AK:
Sharing news with colleagues
JL:
TM:
Finishing a morning routine
JM:

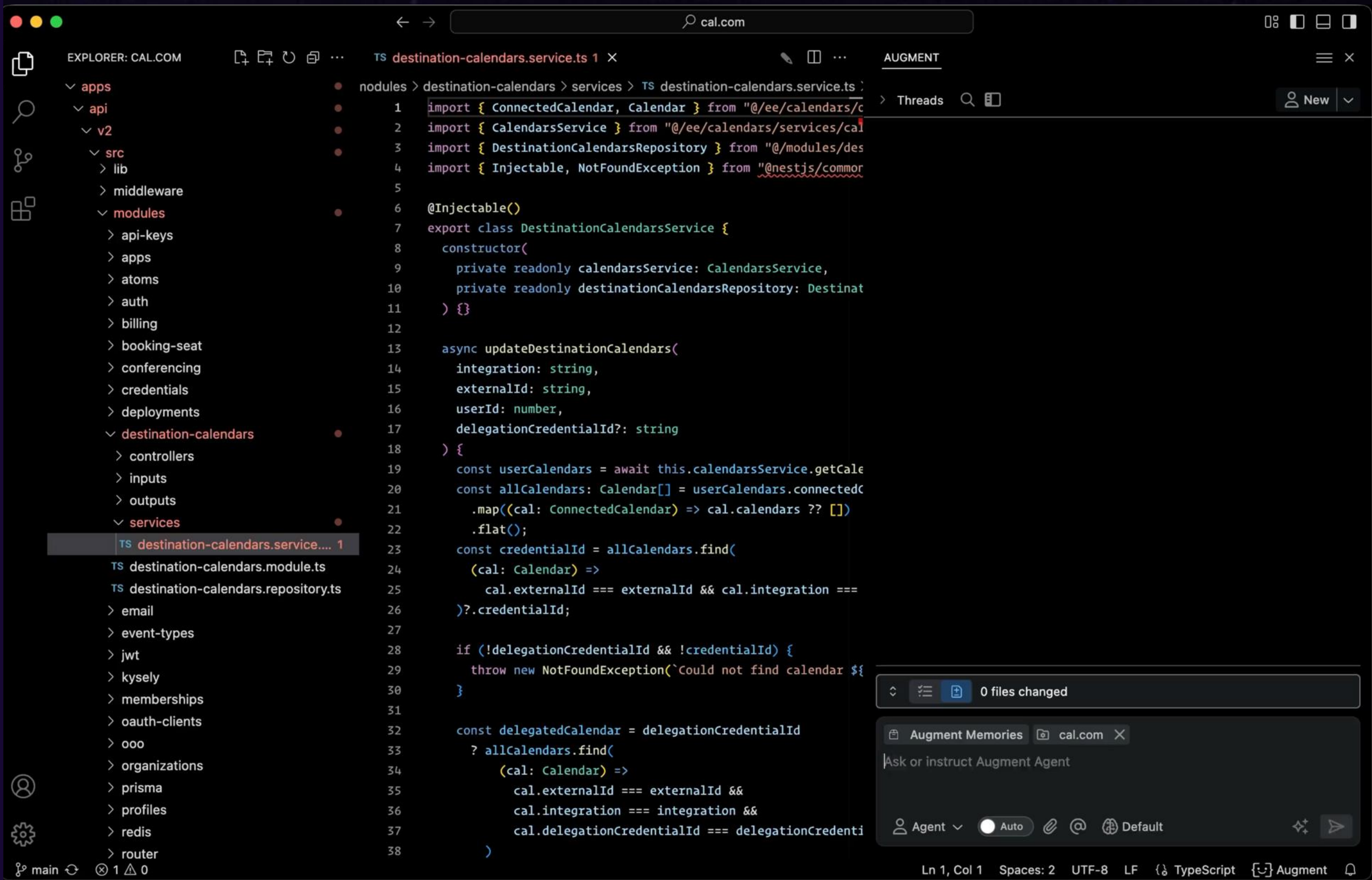
[Abigail]: Hey Klaus, mind if I join you for coffee?
[Klaus]: Not at all, Abigail. How are you?

[John]: Hey, have you heard anything new about the upcoming mayoral election?
[Tom]: No, not really. Do you know who is running?

Figure 1: Generative agents are believable simulacra of human behavior for interactive applications. In this work, we demonstrate generative agents by populating a sandbox environment, reminiscent of The Sims, with twenty-five agents. Users can observe and intervene as agents plan their days, share news, form relationships, and coordinate group activities.

J. S. Park et al. "Generative Agents: Interactive Simulacra of Human Behavior" *UIST* (2023)

Cline



```
import { ConnectedCalendar, Calendar } from "@/ee/calendars/c
import { CalendarsService } from "@/ee/calendars/services/cal
import { DestinationCalendarsRepository } from "@/modules/des
import { Injectable, NotFoundException } from "@nestjs/commo

@Injectable()
export class DestinationCalendarsService {
  constructor(
    private readonly calendarsService: CalendarsService,
    private readonly destinationCalendarsRepository: Destinat
  ) {}

  async updateDestinationCalendars(
    integration: string,
    externalId: string,
    userId: number,
    delegationCredentialId?: string
  ) {
    const userCalendars = await this.calendarsService.getCale
    const allCalendars: Calendar[] = userCalendars.connectedC
      .map((cal: ConnectedCalendar) => cal.calendars ?? [])
      .flat();
    const credentialId = allCalendars.find(
      (cal: Calendar) =>
        cal.externalId === externalId && cal.integration ===
    )?.credentialId;

    if (!delegationCredentialId && !credentialId) {
      throw new NotFoundException(`Could not find calendar ${
    }

    const delegatedCalendar = delegationCredentialId
      ? allCalendars.find(
          (cal: Calendar) =>
            cal.externalId === externalId &&
            cal.integration === integration &&
            cal.delegationCredentialId === delegationCredenti
        )
```

AI Coding Tools

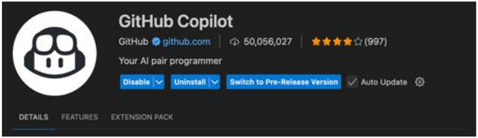
Deployed as:

- Full standalone IDEs
- Plugins for existing IDEs (VS Code etc.)

AI coding suggestions are provided at 3 different scales:

- Single line code completions
  - accepted by pressing Tab
- Highlighted code block rewrites
- Full codebase edits
  - Edits/additions across multiple files
  - Refactoring

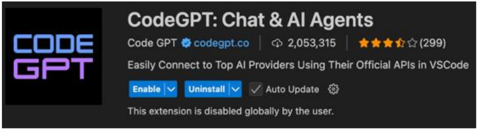
CURSOR



GitHub Copilot

GitHub github.com | 50,056,027 | ★★★★☆ (997)

Your AI pair programmer

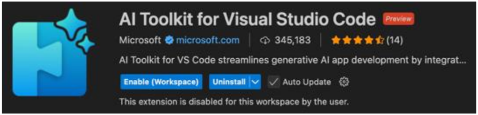


CodeGPT: Chat & AI Agents

Code GPT codegpt.co | 2,053,315 | ★★★½☆ (299)

Easily Connect to Top AI Providers Using Their Official APIs in VSCode

This extension is disabled globally by the user.

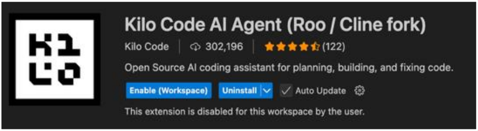


AI Toolkit for Visual Studio Code Preview

Microsoft microsoft.com | 345,183 | ★★★★½ (14)

AI Toolkit for VS Code streamlines generative AI app development by integrat...

This extension is disabled for this workspace by the user.

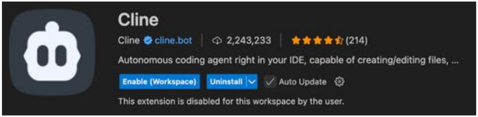


Kilo Code AI Agent (Roo / Cline fork)

Kilo Code | 302,196 | ★★★★½ (122)

Open Source AI coding assistant for planning, building, and fixing code.

This extension is disabled for this workspace by the user.



Cline

Cline cline.bot | 2,243,233 | ★★★★½ (214)

Autonomous coding agent right in your IDE, capable of creating/editing files, ...

This extension is disabled for this workspace by the user.

Today's Tutorial

- In today's tutorial we hope to apply the foundational knowledge just covered on AI Agents to use cases that cover different parts of researcher's scientific workflows.
- The remaining content of this tutorial will include the following labs:
  - Introduction to TACCs computing resources
  - Introduction to building AI agents
  - AI Agents for Simulations and Data Analysis
  - HPC integration of AI Agents