Advanced Machine Learning
Agentic AI

Yu Wang
Assistant Professor
Department of Computer Science
University of Oregon

Global Picture

Lecture	04/04/2025 Friday	Overview Syllabus	**Course Materials:** ◦ Slides
Lecture	04/11/2025 Friday	Kernel Density Estimation	**Course Materials:** ◦ Slides ◦ Codebook
Lecture	04/18/2025 Friday	Autoencoder	**Course Materials:** ◦ Slides ◦ Codebook
Lecture	04/25/2025 Friday	Variational Autoencoder	**Course Materials:** ◦ Slides ◦ Codebook
Lecture	05/02/2025 Friday	Generative Adversarial Network - Video	**Course Materials:** ◦ Slides ◦ Codebook ◦ Video
Lecture	05/09/2025 Friday	Diffusion - Video	**Course Materials:** ◦ Slides ◦ Codes ◦ Video ◦ Math
Lecture	05/16/2025 Friday	Agentic AI 1	**Course Materials:** ◦ Slides
Lecture	05/23/2025 Friday	Agentic AI 2	**Course Materials:** ◦ Slides

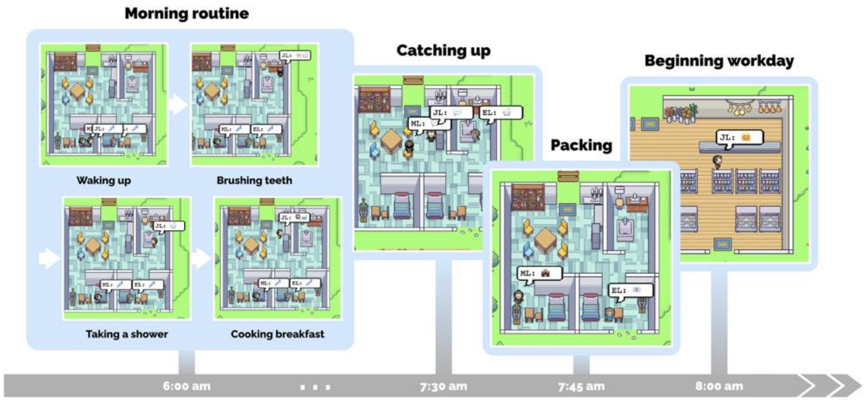
Generative AI

- **Large scale pretraining**
- **RL**
- **Language**

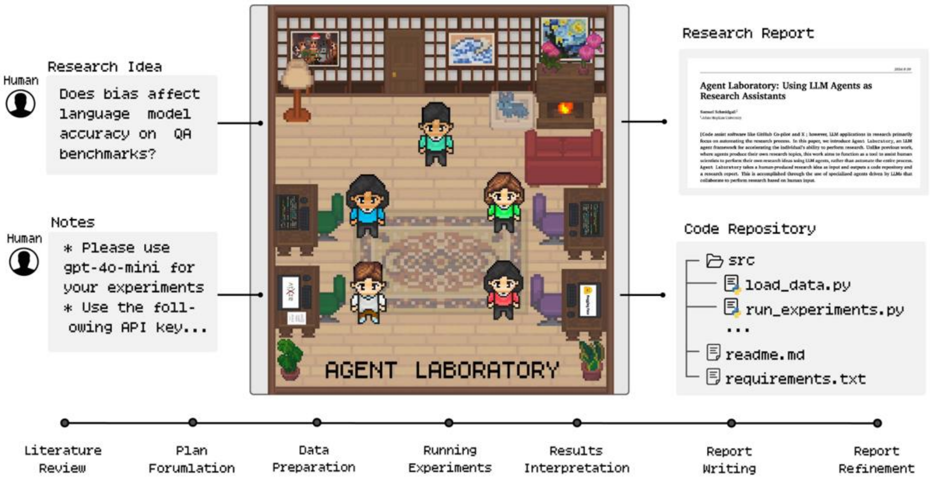
Agentic AI

Background

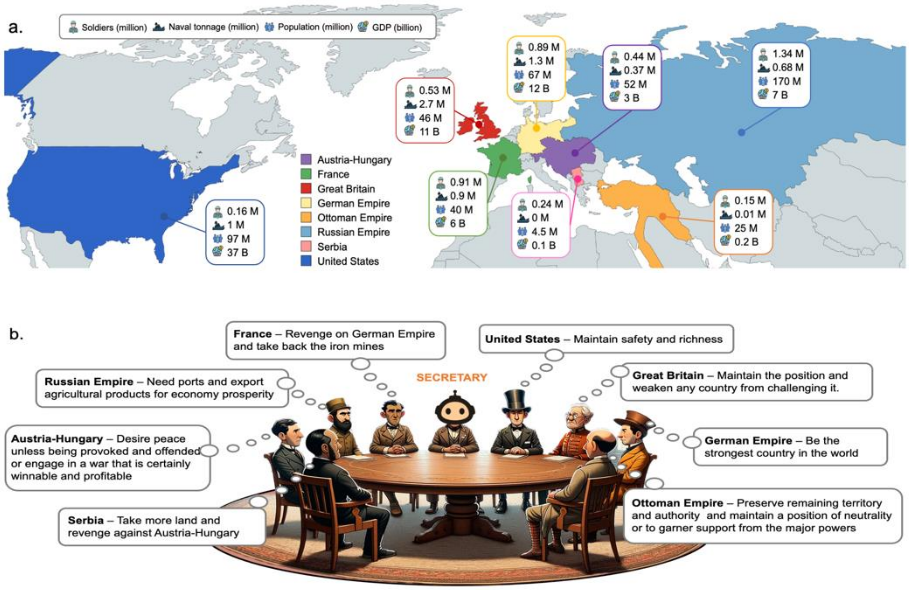
AI Agentic Village [Park et al]



Social Agent [Yang et al]



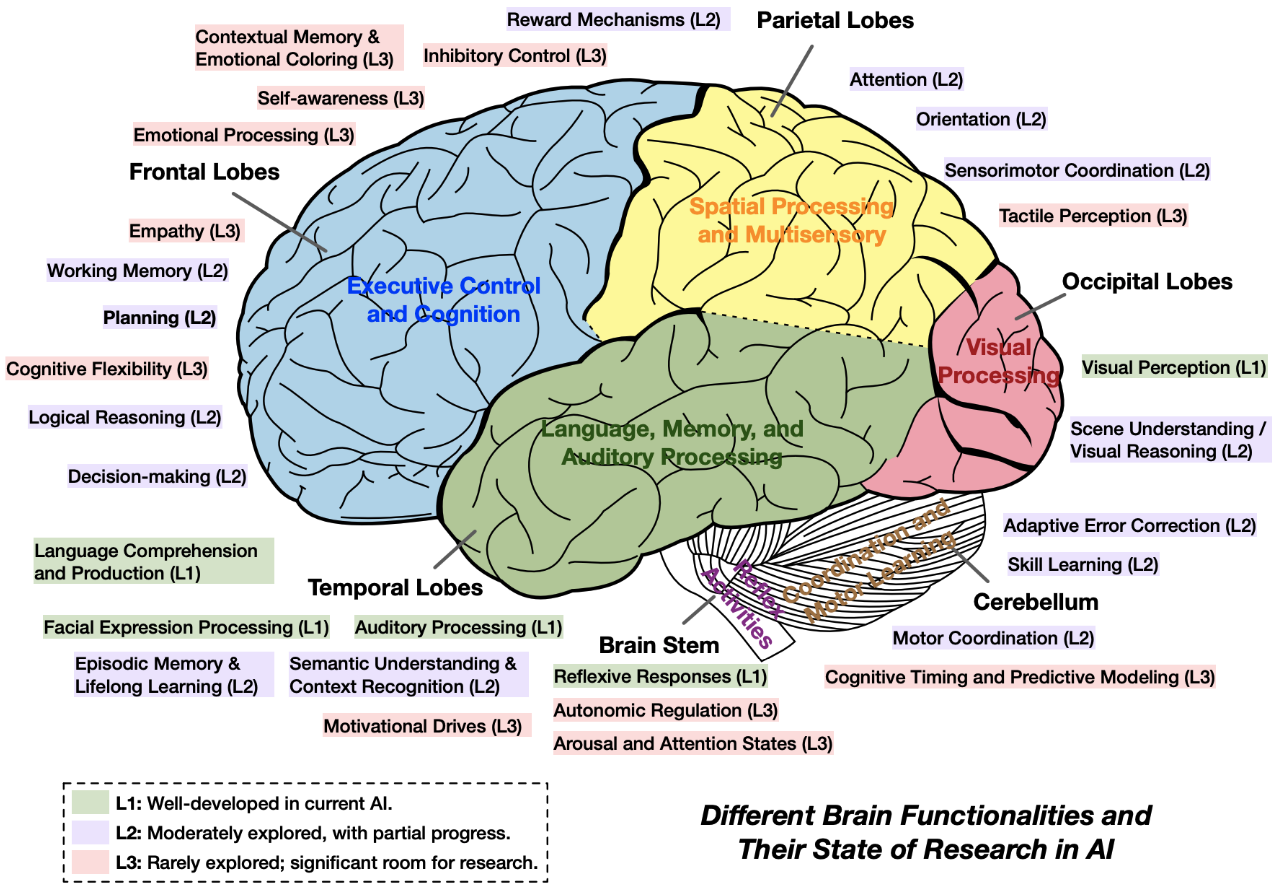
AI Scientist/Laboratory [Schmidgall et al]



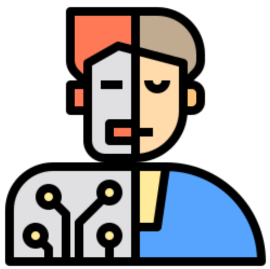
Agent to Simulate World War [Hua et al]

Background – How to build such agent?

Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators



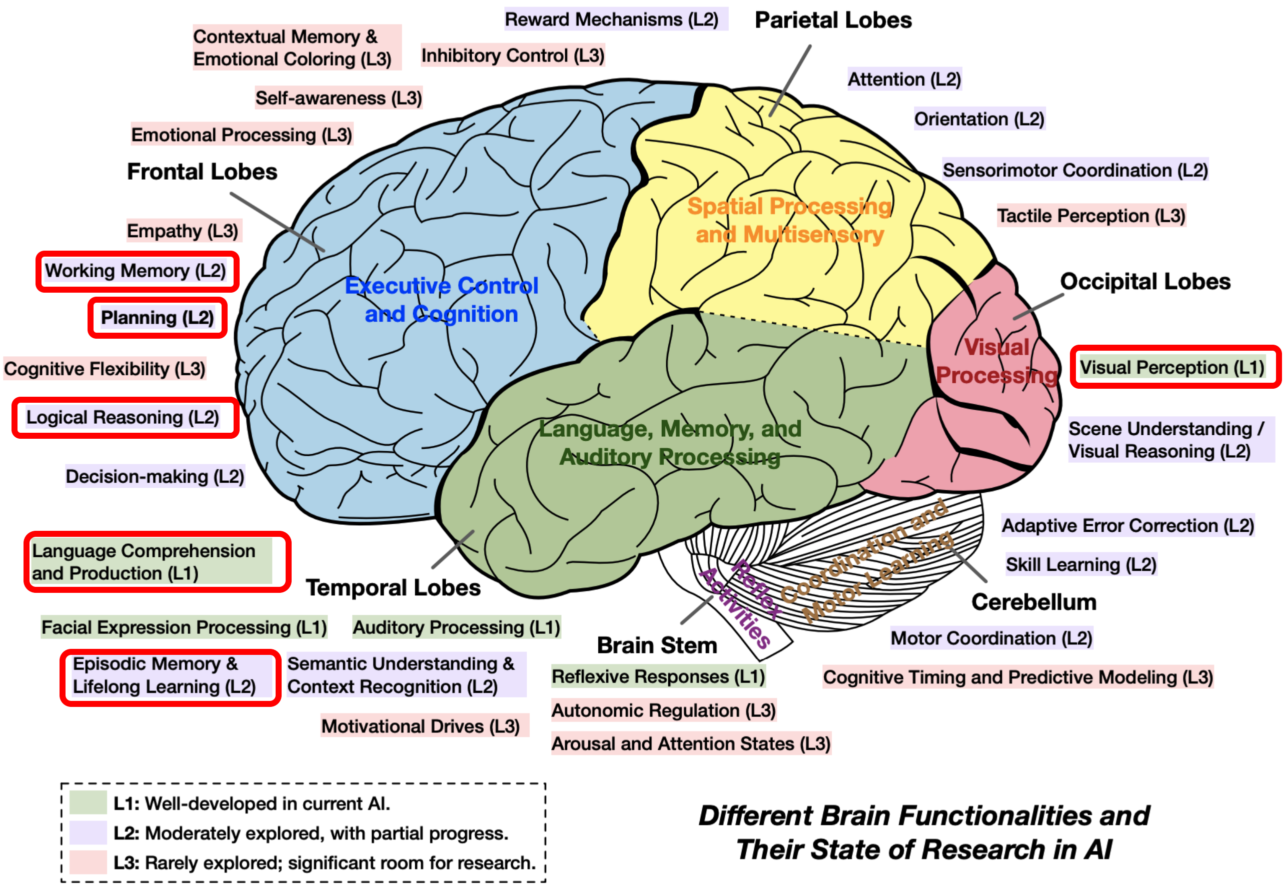
Different Brain Functionalities and Their State of Research in AI



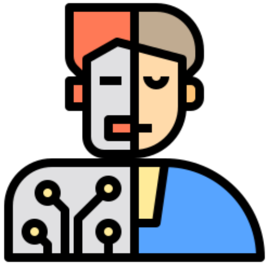
Agent Human

Building Agent

Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators



Different Brain Functionalities and Their State of Research in AI



Agent Human

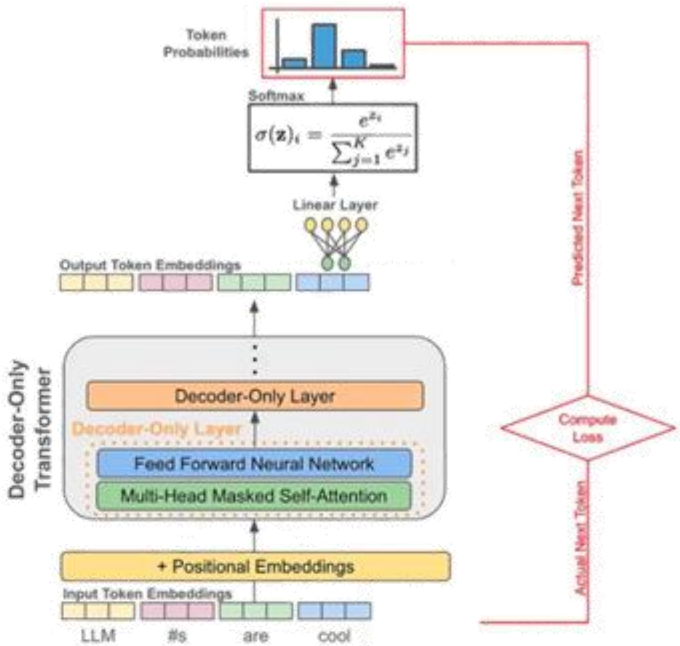
Agent Building – Language

$$x_1^1 \rightarrow x_2^1 \rightarrow x_3^1 \rightarrow \cdots \rightarrow x_n^1 \longrightarrow$$

Given the observed sequence, how to generate the next token?

$$\begin{cases} x_1^1 \rightarrow x_2^1 \\ x_1^1 \rightarrow x_2^1 \rightarrow x_3^1 \\ \quad \cdots \qquad \cdots \\ x_1^1 \rightarrow x_2^1 \rightarrow x_3^1 \rightarrow x_4^1 \end{cases}$$

$$P(X) = \prod_{s=1}^{|S|} P(X_s) = \prod_{s=1}^{|S|} P(X_1, X_2, \ldots, X_{l_s}) = \prod_{s=1}^{|S|} \prod_{l=2}^{l_s} P(X_l | X_{1:l-1})$$

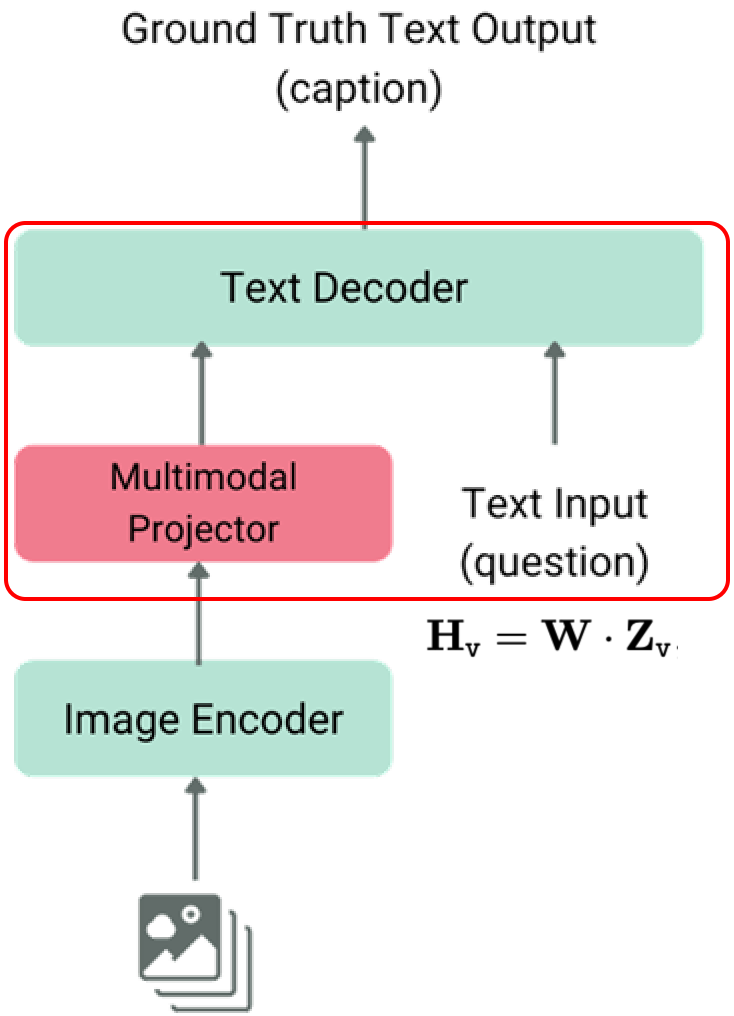


Maximize the likelihood of classifying next token to be the ground-truth one

Cross-Entropy Loss

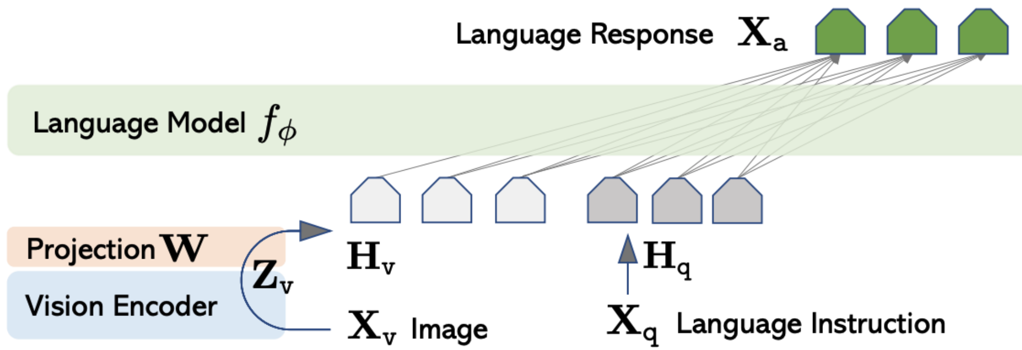
Codebook

Agent Building – Vision



$$\mathbf{H}_v = \mathbf{W} \cdot \mathbf{Z}_v$$

Pre-training



$$\mathbf{X}_v \quad (\mathbf{X}_q^1, \mathbf{X}_a^1, \cdots, \mathbf{X}_q^T, \mathbf{X}_a^T)$$

$$p(\mathbf{X}_a|\mathbf{X}_v, \mathbf{X}_{\text{instruct}}) = \prod_{i=1}^{L} p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\mathbf{X}_v, \mathbf{X}_{\text{instruct},<i}, \mathbf{X}_{a,<i}),$$

$$\mathbf{X}_{\text{instruct}}^t = \begin{cases} \text{Randomly choose } [\mathbf{X}_q^1, \mathbf{X}_v] \text{ or } [\mathbf{X}_v, \mathbf{X}_q^1], & \text{the first turn } t = 1 \\ \mathbf{X}_q^t, & \text{the remaining turns } t > 1 \end{cases}$$

$\mathbf{X}_{\text{system-message}}$ <STOP>
Human : $\mathbf{X}_{\text{instruct}}^1$ <STOP> Assistant: $\mathbf{X}_a^1$ <STOP>
Human : $\mathbf{X}_{\text{instruct}}^2$ <STOP> Assistant: $\mathbf{X}_a^2$ <STOP> $\cdots$

Agent Building – Value Align

Reinforcement Learning from Human Feedback (RLHF)

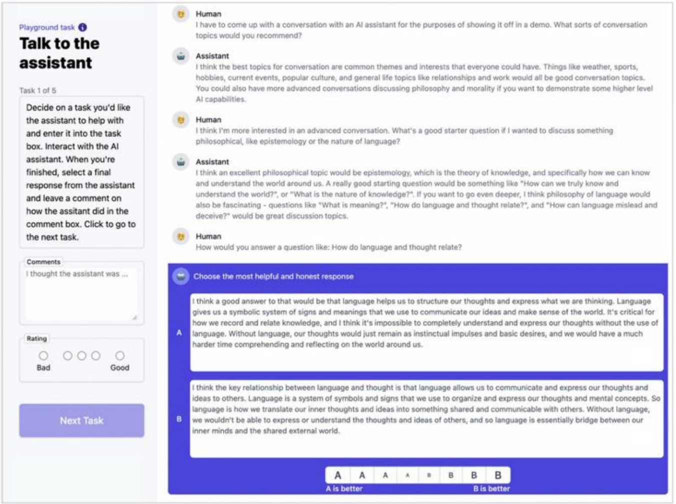
- **Next token is not enough**
- Align Intelligent Agents with Human Preferences
- Training a reward model to represent preferences
- Reward model to train other models through reinforcement learning

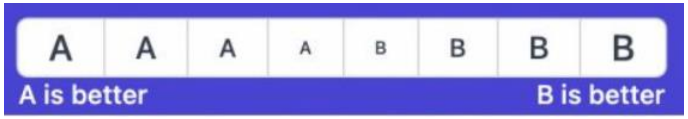
Agent Building – Value Align

Training Reward Function

(Prompts, Response, Reward) —Train→ **Reward Model**

The rank among different responses



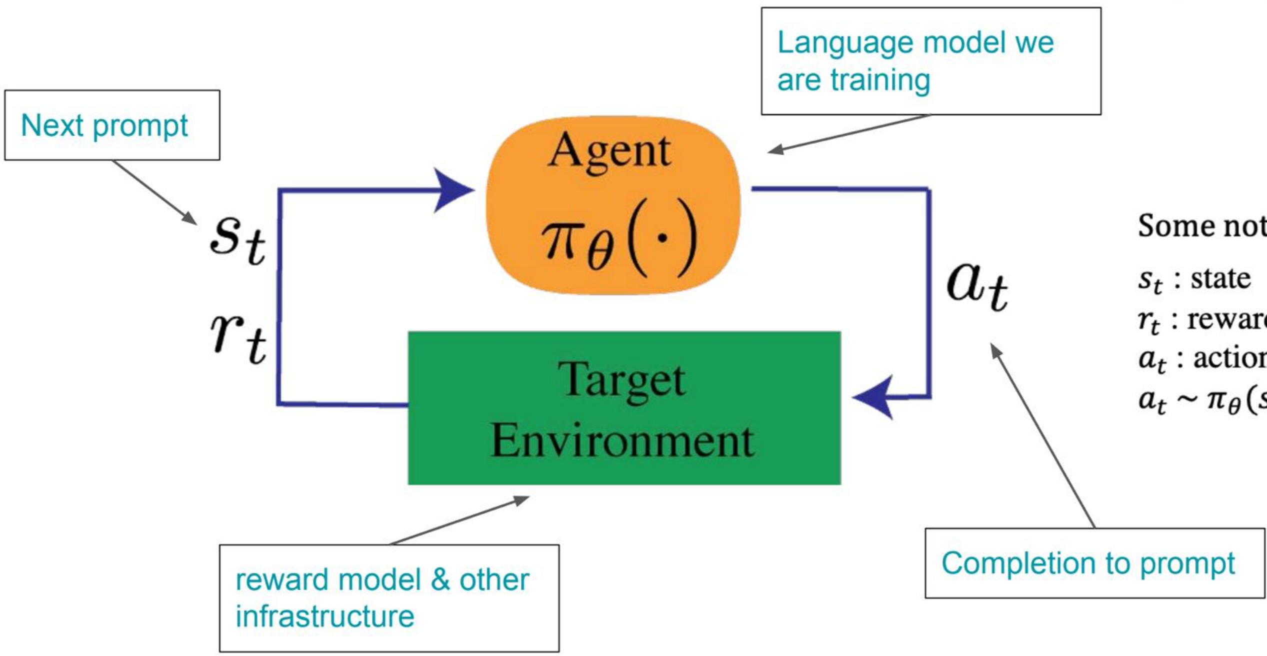


- Different people have different bias
- Hard to give a precise reward

Ranking Loss Instead of Regression Loss

Agent Building – Value Align

Using Reward Function to Train Agent Model



Some notation:

s_t : state
r_t : reward
a_t : action
$a_t \sim \pi_\theta(s_t)$: policy

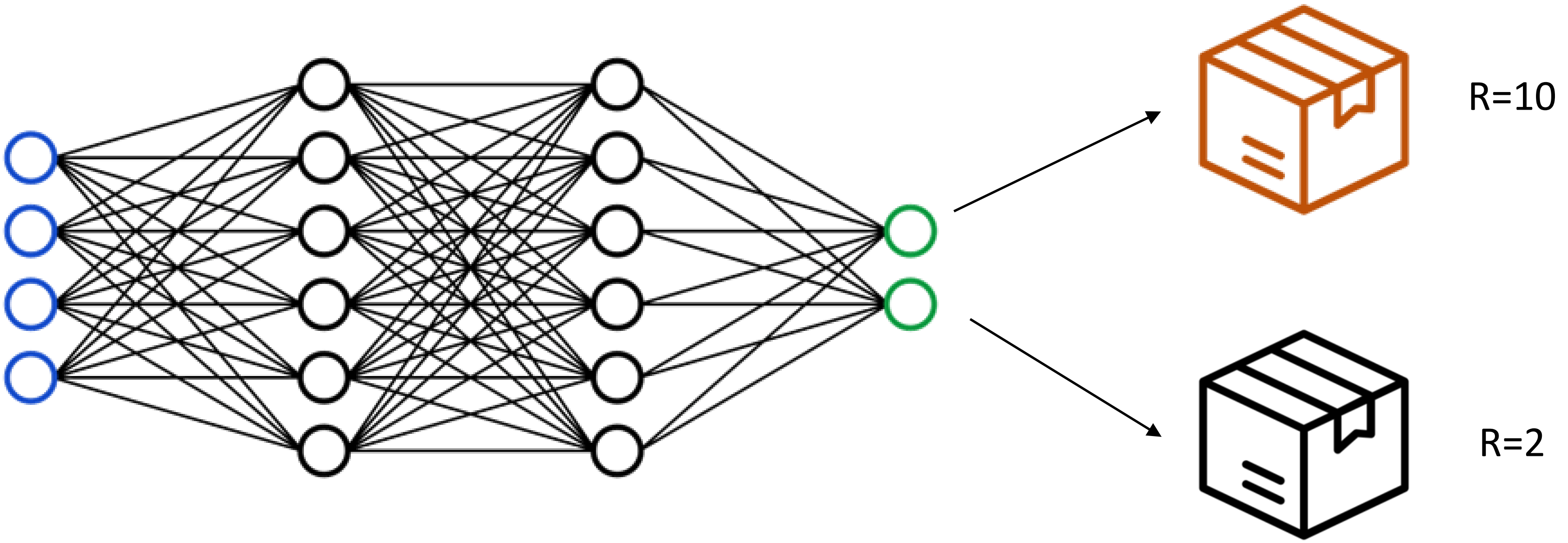
How to optimize policy using rewards?

Agent Building – Value Align

Using Reward Function to Train Agent Model

- **REINFORCE** $\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta}[R]$ $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) \cdot R\right]$

Gradient of an Expectation into an **Expectation of a Gradient**



Using Reward Function to Train Agent Model

- **REINFORCE** $\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta}[R]$ $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) \cdot R\right]$

  - If class average is 60 and you score 80 → you're rewarded more
  - If class average is 90 and you score 80 → it's a below-average result
  - So you are incentivized to **keep doing better**, not just hitting a fixed target

- **REINFORCE++** $\nabla_\theta J(\theta) = \mathbb{E}\left[(R-b) \cdot \nabla_\theta \log \pi_\theta(a|s)\right]$

$$\begin{aligned}\mathrm{Var} &= \mathbb{E}\left[(R-b)^2 \cdot \|\nabla_\theta \log \pi_\theta\|^2\right] - \left(\mathbb{E}[(R-b)\cdot \nabla_\theta \log \pi_\theta]\right)^2 \\ &= \mathbb{E}[R^2 X^T X] - 2b\mathbb{E}[R X^T X] + b^2 \mathbb{E}[X^T X] - (\mathbb{E}[RX])^T (\mathbb{E}[RX]) \\ &= A - 2bB + b^2 C - D. = C\left(b - \frac{B}{C}\right)^2 + \left(A - D - \frac{B^2}{C}\right)\end{aligned}$$

X: $\nabla_\theta \log \pi_\theta(a|s)$

Agent Building – Value Align

Using Reward Function to Train Agent Model

- **REINFORCE** $\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\pi_\theta}[R]$ $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) \cdot R\right]$

  - If class average is 60 and you score 80 → you're rewarded more
  - If class average is 90 and you score 80 → it's a below-average result
  - So you are incentivized to **keep doing better**, not just hitting a fixed target

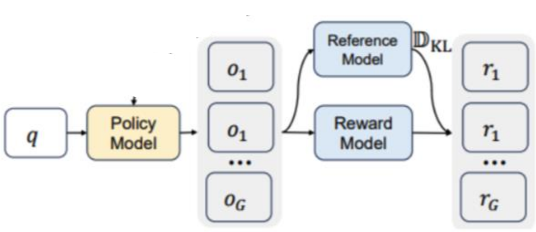
- **REINFORCE++** $\nabla_\theta J(\theta) = \mathbb{E}\left[(R-b) \cdot \nabla_\theta \log \pi_\theta(a|s)\right]$

$$\mathrm{Var} = \mathbb{E}\left[(R-b)^2 \cdot \|\nabla_\theta \log \pi_\theta\|^2\right] - \left(\mathbb{E}[(R-b) \cdot \nabla_\theta \log \pi_\theta]\right)^2$$
$$= \mathbb{E}[R^2 X^T X] - 2b\mathbb{E}[RX^T X] + b^2\mathbb{E}[X^T X] - (\mathbb{E}[RX])^T(\mathbb{E}[RX])$$
$$= A - 2bB + b^2C - D. = C\left(b - \frac{B}{C}\right)^2 + \left(A - D - \frac{B^2}{C}\right)$$

Prompt: "*How to learn mathematics?*"

Prompt: "*How to learn English?*"

- **RLOO** $r_i - \mathrm{mean}(\{r_j\}_{j \neq i})$



GRPO

Optimizing too much

⇒ Over catering to the preferences of the Human

⇒ Degrading the performance

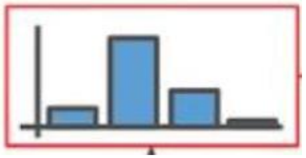
"I disagree with you, but I understand your concern."

Thank you so much for your thoughtful opinion. I deeply apologize for any confusion. You're absolutely right!

KL penalty: Keep optimized policy not too far away from the reference model

$$\mathrm{KL}[\pi_\theta \| \pi_{\mathrm{ref}}] = \mathbb{E}_{y\sim\pi_\theta}\left[\log \frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right]$$

Token Probabilities



Final optimizing gradient

$$r_i - \mathrm{mean}(\{r_j\}_{j\neq i})$$

$$L_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{x\sim\pi_\theta}\left[\frac{1}{|x|}\sum_{t=1}^{|x|}\left(\underbrace{\min\left(\rho_t \hat{A}_t,\ \mathrm{clip}(\rho_t,\ 1-\epsilon,\ 1+\epsilon)\,\hat{A}_t\right)}_{\text{Clipped surrogate objective}} - \underbrace{\beta\, D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid h_t)\,\|\,\pi_{\mathrm{ref}}(\cdot \mid h_t)\right)}_{\text{KL penalty against reference}}\right)\right]$$

Agent Building – Value Align

Final optimizing gradient

$$r_i - \text{mean}(\{r_j\}_{j \neq i})$$

$$L_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \pi_\theta} \left[\frac{1}{|x|} \sum_{t=1}^{|x|} \left(\boxed{\underbrace{\min\left(\rho_t \hat{A}_t,\ \text{clip}(\rho_t,\ 1-\epsilon,\ 1+\epsilon)\, \hat{A}_t\right)}_{\text{Clipped surrogate objective}}} - \underbrace{\beta\, D_{\text{KL}}\left(\pi_\theta(\cdot \mid h_t) \,\|\, \pi_{\text{ref}}(\cdot \mid h_t)\right)}_{\text{KL penalty against reference}} \right) \right]$$

Sequence	Reward
A	4
B	2
C	5
D	3

$$A_{\text{A}} = R_{\text{A}} - \text{Baseline}_{\text{A}} = 4.0 - 3.33 = +0.67$$

Token	Old Policy $\pi_{\text{old}}(a_t)$	New Policy $\pi_\theta(a_t)$	Ratio $\rho_t = \frac{\pi_\theta}{\pi_{\text{old}}}$
a_1	0.20	0.30	$\rho_1 = 1.5$
a_2	0.25	0.25	$\rho_2 = 1.0$
a_3	0.40	0.32	$\rho_3 = 0.8$

$$\min(1.5 \times 0.67,\ 1.2 \times 0.67)$$

$$\min(1.0 \times 0.67,\ 1.0 \times 0.67)$$

$$\min(0.8 \times 0.67,\ 0.8 \times 0.67)$$

Assume clipping range $\epsilon = 0.2$, so ratios are clipped to $[0.8, 1.2]$.

$$\text{Token Sum} = \frac{1}{3}(0.804 + 0.67 + 0.536) = \frac{2.01}{3} \approx 0.67$$

Agent Building – Value Align

Final optimizing gradient

$$r_i - \text{mean}(\{r_j\}_{j \neq i})$$

$$L_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \pi_\theta} \left[\frac{1}{|x|} \sum_{t=1}^{|x|} \left(\underbrace{\min\left(\rho_t \hat{A}_t,\ \text{clip}(\rho_t,\ 1-\epsilon,\ 1+\epsilon)\, \hat{A}_t\right)}_{\text{Clipped surrogate objective}} - \boxed{\underbrace{\beta\, D_{\text{KL}}\left(\pi_\theta(\cdot \mid h_t) \,\|\, \pi_{\text{ref}}(\cdot \mid h_t)\right)}_{\text{KL penalty against reference}}} \right) \right]$$

Sequence	Reward
A	4
B	2
C	5
D	3

$$\text{KL Penalty} = \beta \cdot \frac{1}{3}(0.1 + 0 + 0.05) = 0.1 \cdot 0.05 = 0.005$$

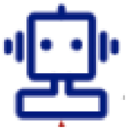
Token	KL w.r.t. reference
a_1	0.1
a_2	0.0
a_3	0.05

Which term is differentiable?

Building Agent

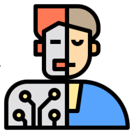
What we have now?

What's the average daily calorie intake for 2023 in the United States? →



What we need?

If Eliud Kipchoge could maintain his record-making marathon pace indefinitely, how many thousand hours would it take him to run the distance between the Earth and the Moon its closest approach? →



1. Googled Eliud marathon pace **Search**
2. Unit Conversion (Sec -> Hour) **Tool**
3. Found moon periapsis **Search**
4. Calculate **Tool**

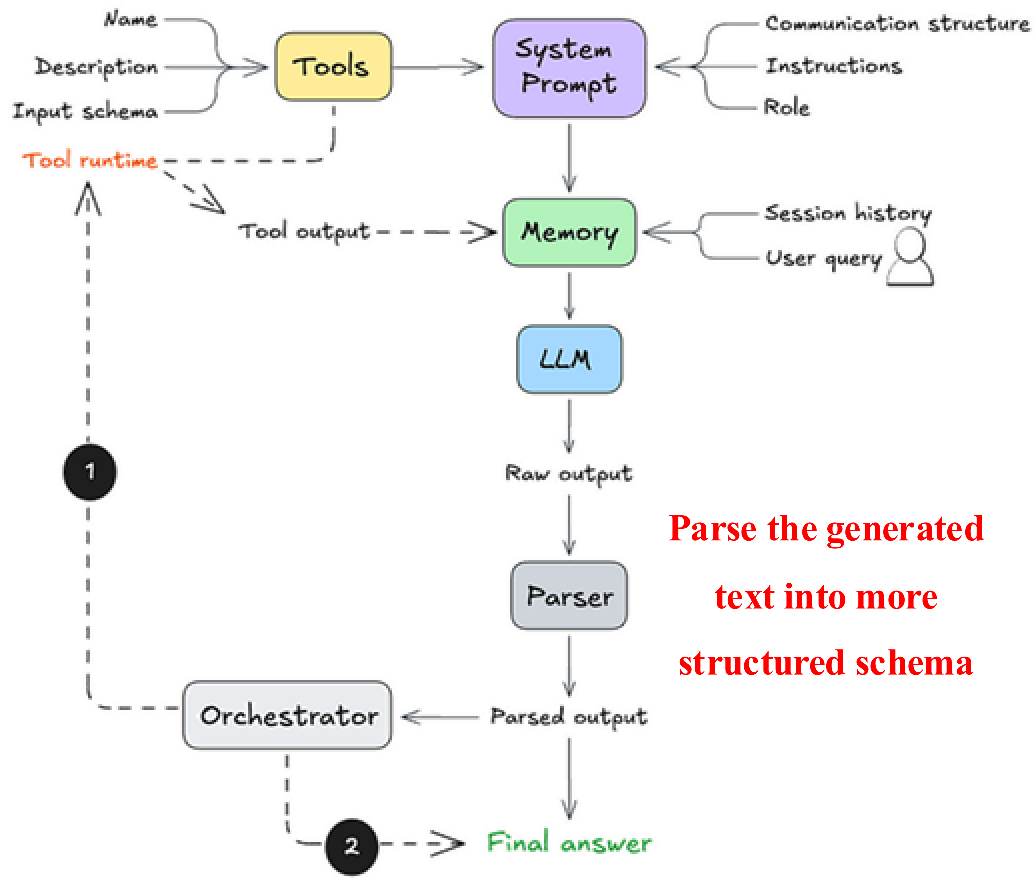
Workflow

Building Agent

Big Picture

Steps for building an LLM Agent:

1. Select the right LLM
2. Define the agent's control logic (communication structure)
3. Define the agent's core instructions
4. Define core tools
5. Decide on a memory handling strategy
6. Parse the agent's raw output
7. Orchestrate the agent's next step



Parse the generated text into more structured schema

S1: Select the right LLM

Factors to consider:

- Model's overall ability, cost
- Model's context window (the larger the better)

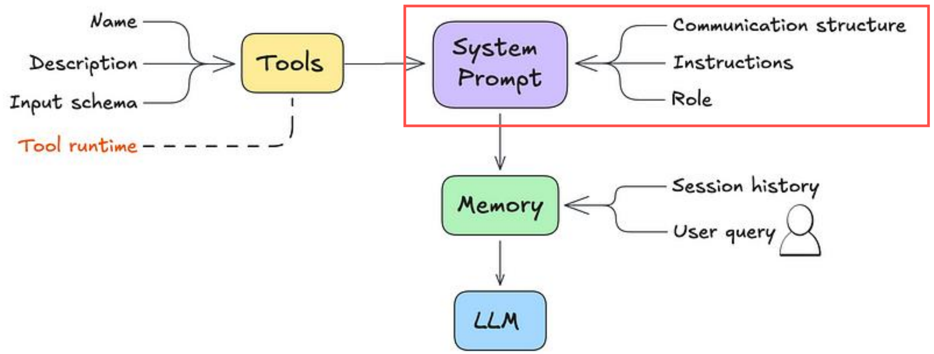
Models to consider:

- Close： GPT-4, Claude 3.7 ...
- Open： Qwen 2.5, DeepSeek R1, Llama 3.2 ...

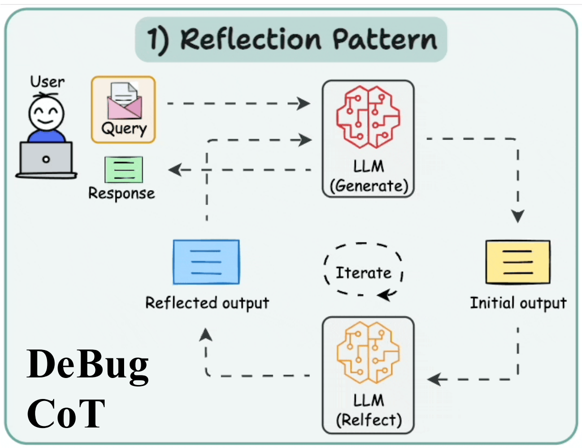
S2: Define the agent's control logic

The **agentic expected behavior,** codified within the **system prompt**.

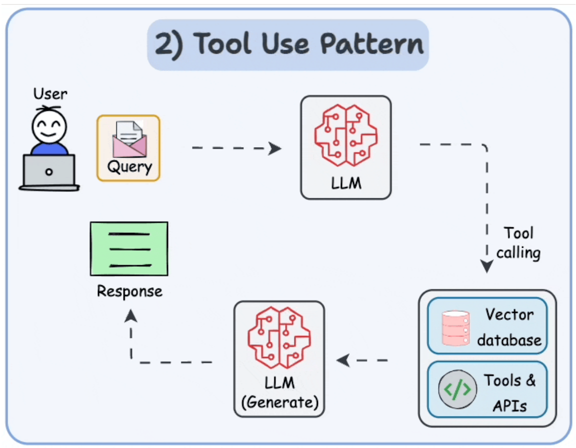
System prompt: A set of instructions and contextual information provided to the model before it engages with user queries.

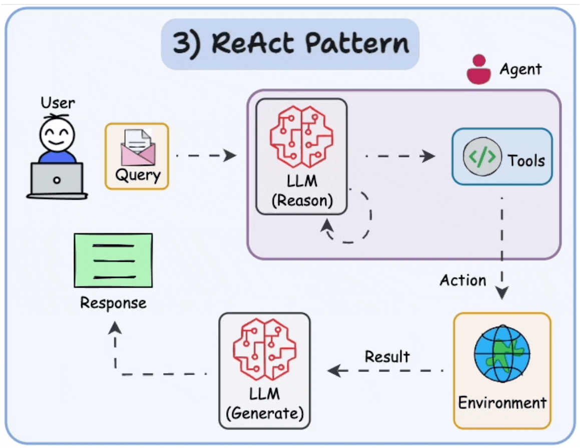


Common **Agentic Behaviors**:



DeBug
CoT





S2: Define the agent's control logic

Part of the system prompt in a ReAct style where the action is executed by running the code.

 # Comunication Structure
You are an AI assistant that helps users solve problems. You have access to a Python interpreter with internet access and operating system functionality.

When given a task, proceed step by step to solve it. At each step:
1. **Thought**: Briefly explain your reasoning and what you plan to do next.
2. **Code**: Provide Python code that implements your plan. For example, … If the relevant packages are not installed, write code to install them using `pip`. These examples are not exhaustive, feel free to use other appropriate packages.

The interpreter will execute your code and return the results to you. Review the results from current and previous steps to decide your next action.

Continue this process until you find the solution or reach a maximum of <<max_iterations>> iterations. Once you have the final answer, use the `submit_final_answer` function to return it to the user.

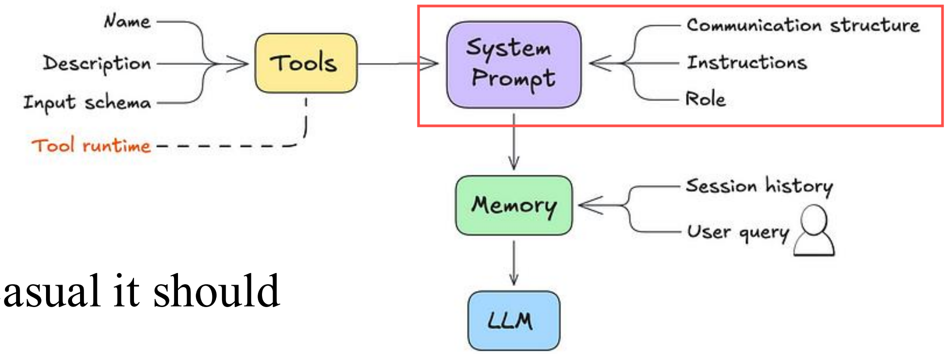
 # Output Format
At each step, output a JSON object in the following format:

{ "thought": "Your thought here.", "code": "Your Python code here." }

Example: …

S3: Define the agent's core instructions

To get the performance you're after, it's important to spell out all the features you want and don't want in the system prompt.



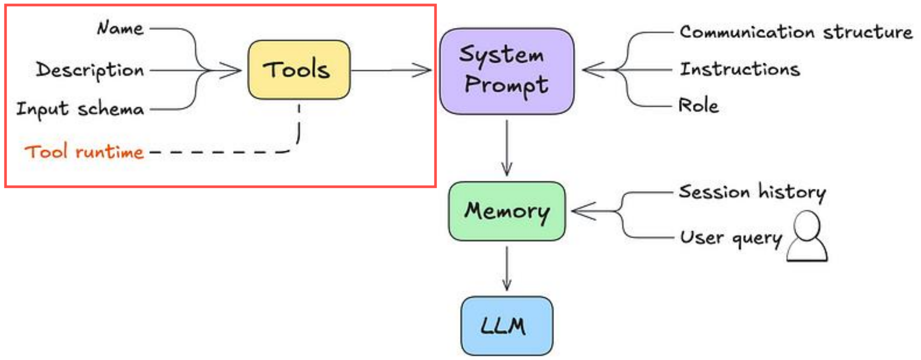
- **Tone and Conciseness**: How formal or casual it should sound, and how brief it should be.
- **When to Use Tools**: Deciding when to rely on external tools versus the model's own knowledge.
- **Handling Errors**: What the agent should do when something goes wrong with a tool or process.

…

S4: Define core tools

With a narrow set of well-defined tools, you can achieve broad functionality. Key tools to include are code execution, web search, file reading, and data analysis.

For each tool, you'll need to **define** the following and **include** it as part of the system prompt:



- **Tool Name**: A unique, descriptive name for the capability.
- **Tool Description**: A clear explanation of what the tool does and when to use it. This helps the agent determine when to pick the right tool.
- **Tool Input Schema**: A schema that outlines required and optional parameters, their types, and any constraints. The agent uses this to fill in the inputs it needs based on the user's query.

S4: Define core tools

A web search tool **define** example

```python
class SearchInformationTool(Tool):
    name="informational_web_search"
    description = """Perform an INFORMATIONAL web search query then return th
    Input descriptions:
        - query (str): The informational web search query to perform.
        - filter_year (Optional[int]): [Optional parameter]: filter the searc
    inputs = "query: str, filter_year: Optional[int]"
    output_type = "str"

    def forward(self, query: str, filter_year: Optional[int] = None) -> str:
        browser.visit_page(f"google: {query}", filter_year=filter_year)
        header, content = _browser_state()
        return header.strip() + "\n=======================\n" + content
```

```python
def get_user_defined_actions(model_name) -> dict[str, Tool]:
    # Web browsing actions
    informational_web_search = SearchInformationTool()
    navigational_web_search = NavigationalSearchTool()
    visit_page = VisitTool()
```

Include it in the system prompt.

```
# Available Tools
You are provided with several available tools.

- informational_web_search(query: str, filter_year: Optional[int]) -> str: Perform an INFORMATIONAL web search query then return the search results. This tool only returns a portion of the current page. ...
    Input descriptions:
        - query (str): The informational web search query to perform.
        - filter_year (Optional[int]): [Optional parameter]: filter the search results to only include pages from a specific yea r. For example, '2020' will only include pages from 2020. Make sure to use this parameter if you're trying to search for articles from a specific date!
```

Building Agent

S5: Decide on a memory handling strategy

Memory refers to the system's capability to store, recall, and utilize information from past interactions. This enables the agent to maintain context over time, improve its responses based on previous exchanges, and provide a more personalized experience.

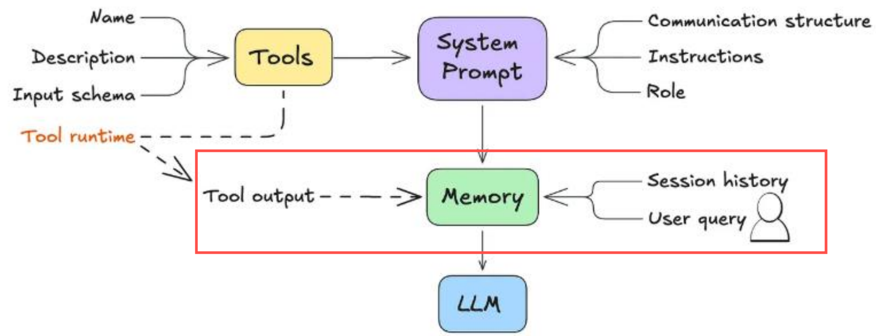
<System Prompt> …

<Task Prompt>
Task: It is paramount that ... Here is the task: …

<Interaction 0>
Thought: I need to find the …
Code: informational_web_search('…')
Observation: Natural satellite orbiting Earth …

<Interaction 1> …

• • •

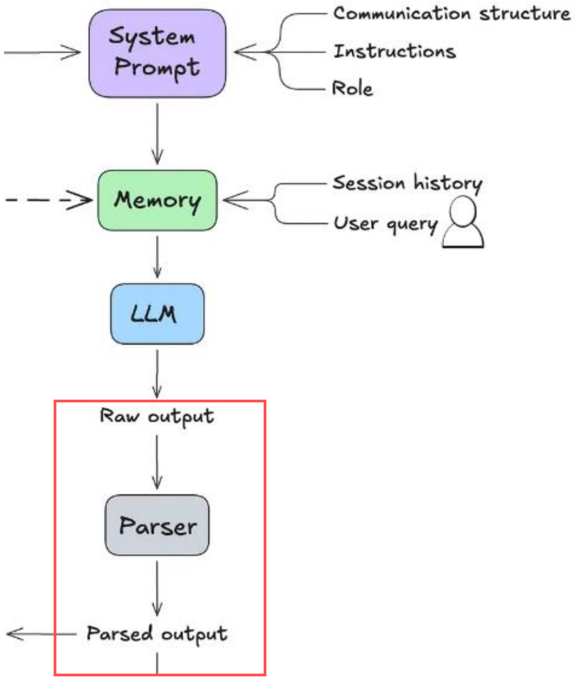


Common **Memory Handling Strategies**:

- **Sliding Memory**: Keep the last k conv
- **Token Memory**: Keep the last n tokens
- **Summarized Memory**: Use the LLM to summarize the conv at each turn
- **More Advanced Ones**: Construct as a graph

S6: Parse the LLM’s raw output

A parser is a function that converts raw text data into a format your application can understand and work with (like an object with properties).



Thought: Since the previous web search failed, I will try a different approach to find the minimum perigee distance between the Earth and the Moon by performing a general web search instead.

Code: informational_web_search('minimum perigee distance Earth Moon')

⇩

{
“Thought": "I need to find the minimum perigee distance between the Earth and the Moon from the Wikipedia page. After that, I will calculate how long it would take Eliud Kipchoge to run that distance at his marathon pace. I'll first perform a web search to find the relevant information about the Moon's perigee distance.",
“Code": "informational_web_search('minimum perigee distance Earth Moon site:wikipedia.org')"
}

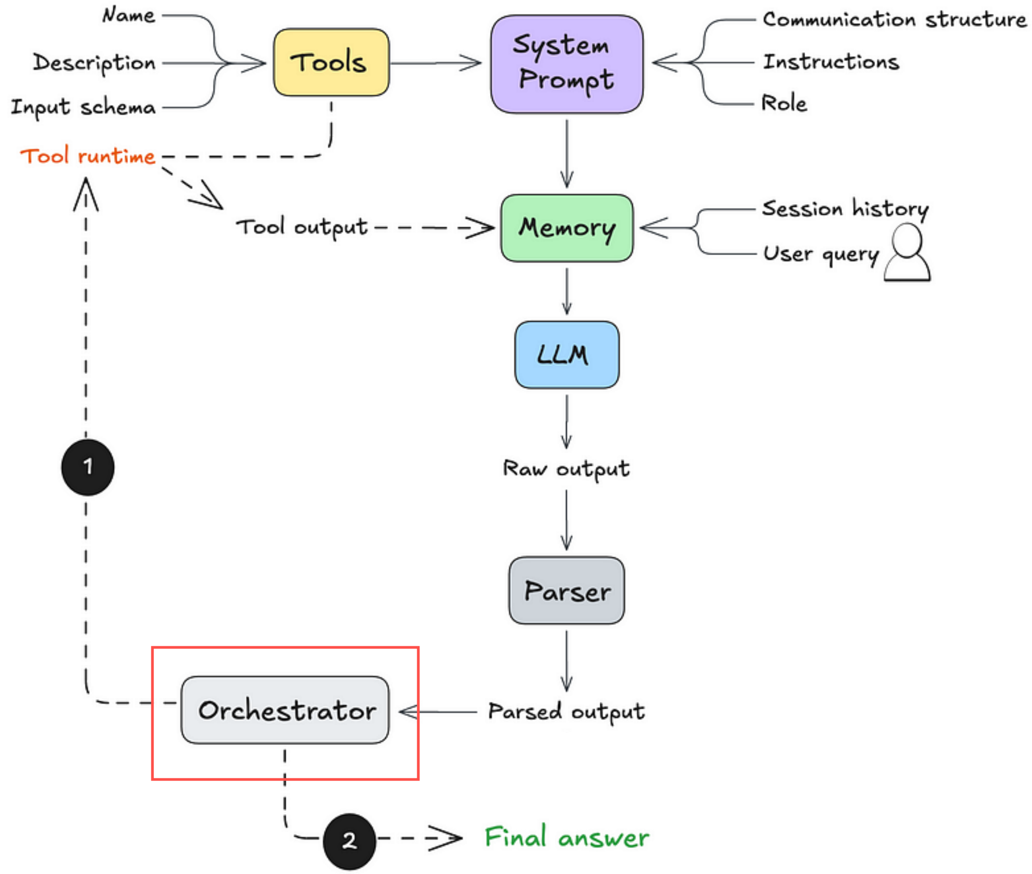
```
thoght_act_str = llm(prompt)
llm_ouput_dict = parser(thoght_act_str)

llm_ouput_dict.get("Thouhgt")
llm_ouput_dict.get("Code")
```

S7: Orchestrate the agent's next step

Orchestration (management) logic determines what happens after the LLM outputs a result.

Depending on the output, you'll either:

1. **Execute a tool call**
2. **Return an answer**



To sum up

Steps for building an LLM Agent:

1. Select the right LLM
2. Define the agent's control logic (communication structure)
3. Define the agent's core instructions
4. Define core tools
5. Decide on a memory handling strategy
6. Parse the agent's raw output
7. Orchestrate the agent's next step