

Grokking the Agents

How LLM Coding Assistants Actually Work Under the Hood — A Case Study with a Simple Example

Oliver Dürr, Sep 2025

This is a brown bag seminar!

- Wanted: Talks zu Methoden, Papers, Erfahrungen auf Konferenzen, Ideen, ...
- Perfekt gestylten Power-Points sind nicht nötig, Tafel tut's auch
- Die Vortragenden müssen keine Experten sein, dumme Fragen sind erwünscht
- Keine Marketing-Veranstaltung

**Die Vortragenden müssen keine Experten sein,
dumme Fragen sind erwünscht.**

Disclaimer: I look at Agentic Coding but do not consider myself as an expert in that field.

Introduction

What happens...

The screenshot shows the Cursor IDE interface. The main editor displays a Python script named `fetch_weather_data.py` with the following code:

```
158     output_file="fixtures/espoo_weather.json",
159 )
160
161 # Fetch ensemble data for Konstanz
162 data = fetch_ensemble_data_to_json(
163     latitude=47.6952,
164     longitude=9.1307,
165     model="icon_d2",
166     output_file="fixtures/konstanz_ensemble.
167 )
168
169 for i in range(10):
170
171
172
173
174 print("\n🌤️ All weather data files created successfully")
175 print("📁 Check the fixtures/ folder for the files")
```

The terminal at the bottom shows the command prompt:

```
(base) oli@olivers-MacBook-Pro-2 weather
oli@olivers-MacBook-Pro-2 weather %
```

On the right side, there is an AI chat interface. The chat history shows a previous message: "In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after 'Fischbach'." The current chat input field contains the text: "...when we let the Agent go". A blue arrow points to the "Send to Backend" button in the chat interface.

Coding Agents Selection (Sep, 2025)

Agent / Tool	Open Source	CLI Tool	IDE Plugin	Full IDE
Cursor	✗	✗	✗	✓
Windsurf	✗	✗	✗	✓
Codex (OpenAI, 2025)	✓	✓	✓	✗
Claude Code	✗	✓	✓	✗
Copilot	✗	✗	✓	✗
Roo Code	✓	✓	✓	✗
Kilo Code	✓	✓	✓	✗



- Cursor / Windsurf extra IDE for AI-based coding
- Plugins are inside IDEs (typical VS-Code / Cursor)
- CLI command line tool started w/o IDE



We will use [Roo-Code](#) since it's open source, so we can look into it.



LLMs

The brains we are using

Step 0 Tokenization Text → Numbers (Tokens)

- A simple component takes text and translates that into integer numbers

The weather is really nice today. I'm thinking about going for a

Tokens	Characters
14	66

The weather is really nice today. I'm thinking about going for a

$y_{14:1}=[976, 11122, 382, 2715, 7403, 4044, 13, 5477, 9791, 1078, 2966, 395, 261, 793]$

LLMs are probabilistic models

The weather is
really nice today.
I'm thinking about
going for a

$y_{t:1}$ (Context)



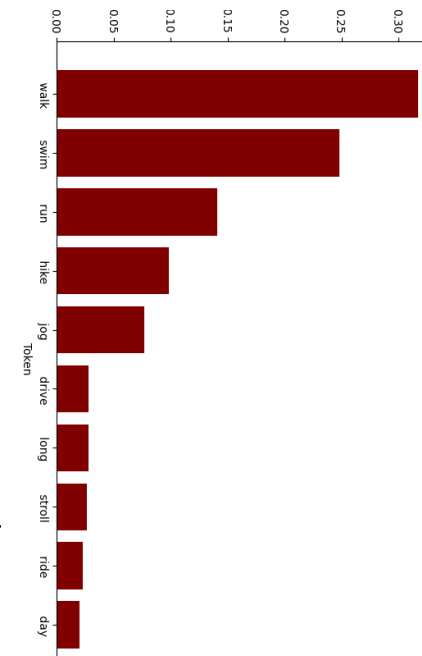
LLM

A model with
parameters θ



y_{t+1}

Probability for y



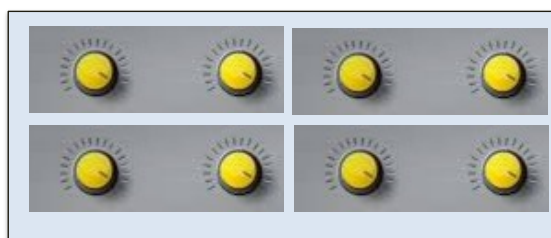
In math an LLM is

$$p_{\theta}(y_{t+1}|y_{t:1})$$

LLMs are probabilistic models

The weather is
really nice today.
I'm thinking
about going for a

$y_{t:1}$ (Context)

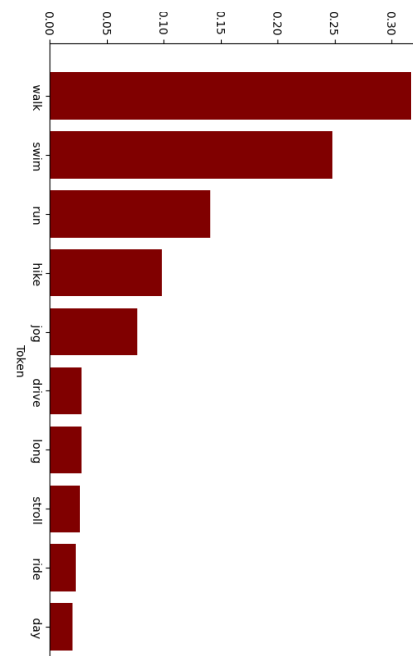


A model with
parameters θ



y_{t+1}

Probability for y



In code an LLM is

```
# Input:  sequence of tokens  $y[0:(t-1)]$  and weights  $\theta$ 
# Output: probability distribution over next token  $y[t]$ 
function next_token_distribution( $y[0:t]$ ,  $\theta$ ):
    ...
    return probs  # =  $p_{\theta}(y_{\{t\}} \mid y_{\{0:(t-1)\}})$ 
```

Creating response in an autoregressive fashion



Describe your technology in one sentence.



I am a Generative Language model based on the transformer architecture that uses self-attention mechanisms and is trained with maximum likelihood principle to predict the next token in a sequence of tokens and generate new text.



At each step, the model samples the next token from $p(y_{t+1}|y_{t:1})$ or takes most likely token (when temperature $T = 0$)

Steps 1: Describe your technology in one sentence. → I

Steps 2: Describe your technology in one sentence. I → am

Steps 3: Describe your technology in one sentence. I am → a

Steps 4: Describe your technology in one sentence. I am a → generative

Step 36 Describe your technology in one sentence. I am a ... new text. → END

Important facts about LLM $p_{\theta}(y_{t+1}|y_{t:0})$

Stateless Models

- Every call is independent $p_{\theta}(y_{t+1}|y_{t:0})$ has no memory
- Native LLM don't remember last conversation.

Limited Context (“window size”)

- Each model can only process a finite number of tokens $t \leq T_{max}^*$

**This requires very tricky engineering
(*context engineering*)**

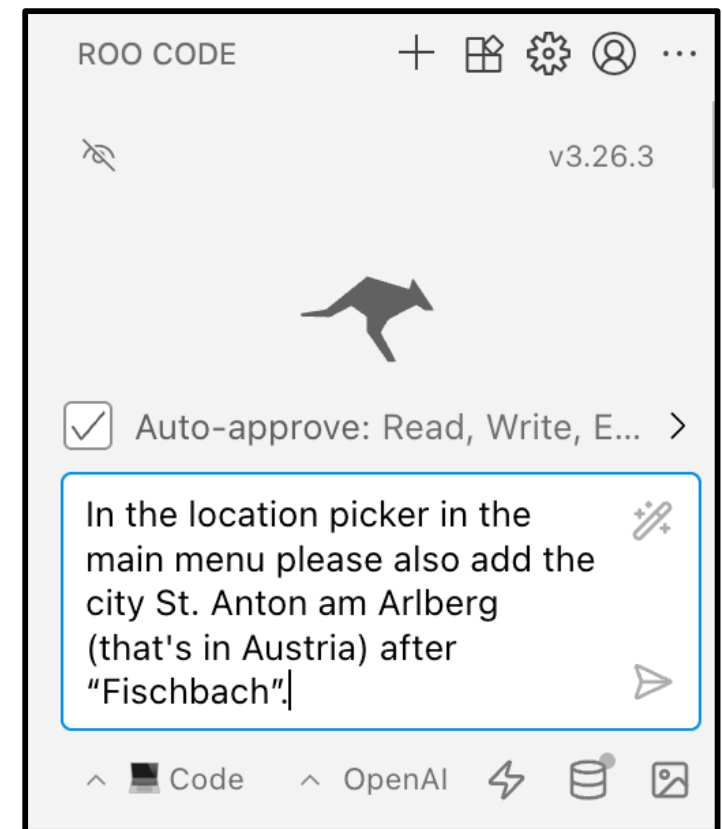
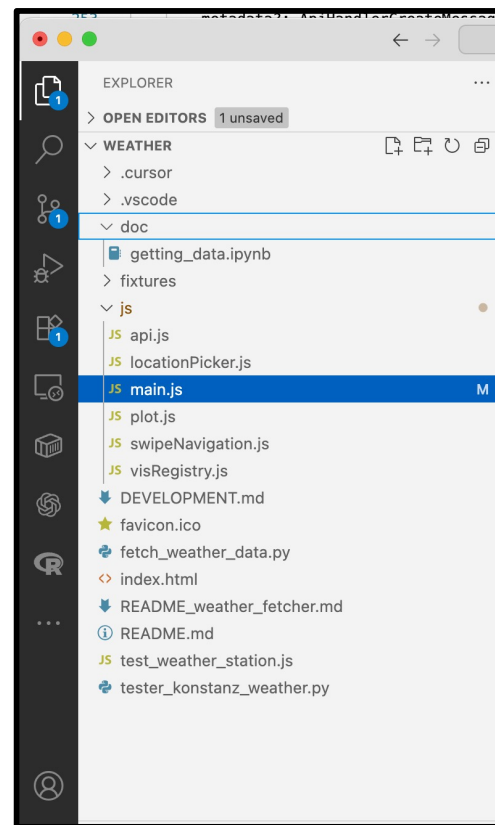
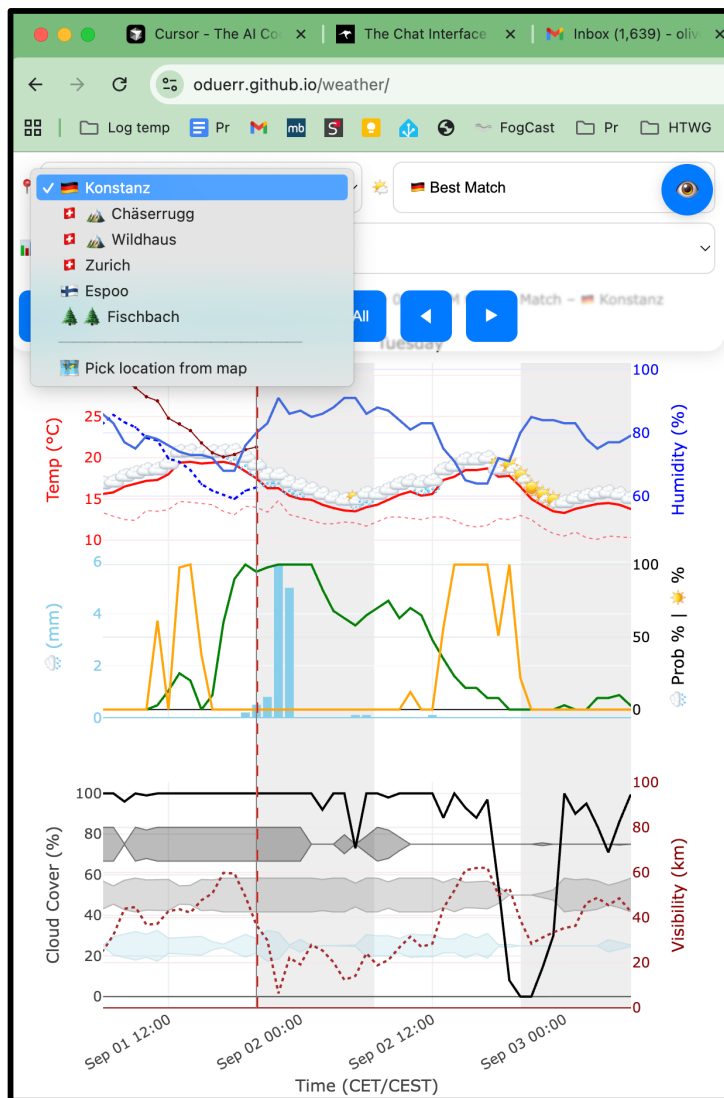
Examples Context Sizes of Current LLMs

- GPT-5 → 400K tokens
- Grok Code Fast → 256K tokens
- Claude Sonnet 4 → 200K (Version with 1M)
- Qwen 3 → 128K tokens (self-hostable)
- DeepSeek-Coder V2 → 128K tokens (self-hostable)

*Technically, in the transformer architecture the input length is fixed, and padding is done

Case Study

Very Simple Task in existing code base



Mini refactoring of a small, partly vibe-coded project. Showing results of weather models for certain locations.

<https://oduerr.github.io/weather/> (gh tag is demo-baseline)

Result, using Roo Code and gpt-4.1

The screenshot displays the Roo Code application interface. On the left, a sidebar contains icons for various tools. The main panel is divided into two sections. The top section, titled 'Task', shows a completed task: 'In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) af'. Below this, it indicates 'Rate limiting for 1 seconds...' and provides coordinates for St. Anton am Arlberg: 'lat: 47.1287, lon: 10.2640'. It also states 'I will update the locations array to include:' followed by a JSON object: `{ name: "St. Anton am Arlberg", lat: 47.1287, lon: 10.2640 }`. The bottom section, titled 'Roo wants to edit this file:', shows the file 'js/main.js' with a 'Start New Task' button and a message input field. The right panel shows the code editor for 'js/main.js'. A blue arrow points to line 49, where a new location entry has been added to the 'locations' array. The text 'New Line Added correctly' is overlaid in green. The code includes a 'PANEL_CONFIG' object and a 'locations' array with several entries. The bottom status bar shows 'Ln 1, Col 1', 'Spaces: 2', 'UTF-8', 'LF', 'JavaScript', 'Go Live', and 'Prettier'.

Task In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) af

18.7k / 1.0m \$0.10

js/main.js

✓ API Request \$0.0390

Rate limiting for 1 seconds...

The location picker options are defined in the `locations` array in `js/main.js`. The dropdown is populated by iterating over this array and appending each location as an option. To add "St. Anton am Arlberg" after "Fischbach", I need to insert a new object for St. Anton am Arlberg (with its coordinates) immediately after the Fischbach entry in the array.

Coordinates for St. Anton am Arlberg, Austria are:

lat: 47.1287, lon: 10.2640

I will update the `locations` array to include:

```
{ name: "St. Anton am Arlberg", lat: 47.1287, lon: 10.2640 }
```

immediately after the Fischbach entry.

Checkpoint Current

Roo wants to edit this file:

js/main.js

✓ API Request \$0.0379

✓ Auto-approve: Read, Write, Execute, Browser, MCP, Mode, Subtasks, Retry, Question, Todo

Start New Task

Type a message...

(@ to add context, / for commands, hold shift to drag in files/images)

Code OpenAI

```
js > JS main.js > ...
14 const PANEL_CONFIG = {
23   uv_wind: {
30   },
31   actuals: {
32     enabled: true,
33     title: 'Actuals',
34     description: 'Observed vs Forecast numeric values',
35     defaultView: '2d',
36     showEnsemble: false,
37     showCurrent: false
38   }
39 };
40
41 // Available locations and models
42 const locations = [
43   { name: "Konstanz", lat: 47.6952, lon: 9.1307 },
44   { name: "Chäserrugg", lat: 47.1549, lon: 9.3128 },
45   { name: "Wildhaus", lat: 47.2033, lon: 9.3505 },
46   { name: "Zurich", lat: 47.3769, lon: 8.5417 },
47   { name: "Espoo", lat: 60.205490, lon: 24.655899},
48   { name: "Fischbach", lat: 48.157652, lon: 8.487578 },
49   { name: "St. Anton am Arlberg", lat: 47.1287, lon: 10.2640 }
50 ];
51 const models = [
52   { id: "bestmatch", label: "Best Match", model: "best_match", type: "deterministi
53   { id: "icon_d2_det", label: "ICON D2 48h", model: "icon_d2", type: "deterministi
54   { id: "icon_seamless_det", label: "Seamless", model: "icon_seamless", type: "det
55   { id: "meteoswiss_icon_ch1", label: "ICON CH1", model: "meteoswiss_icon_ch1", ty
56   { id: "meteoswiss_icon_ch2", label: "ICON CH2", model: "meteoswiss_icon_ch2", ty
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

/Users/oli/.zprofile:7: command not found: pyenv

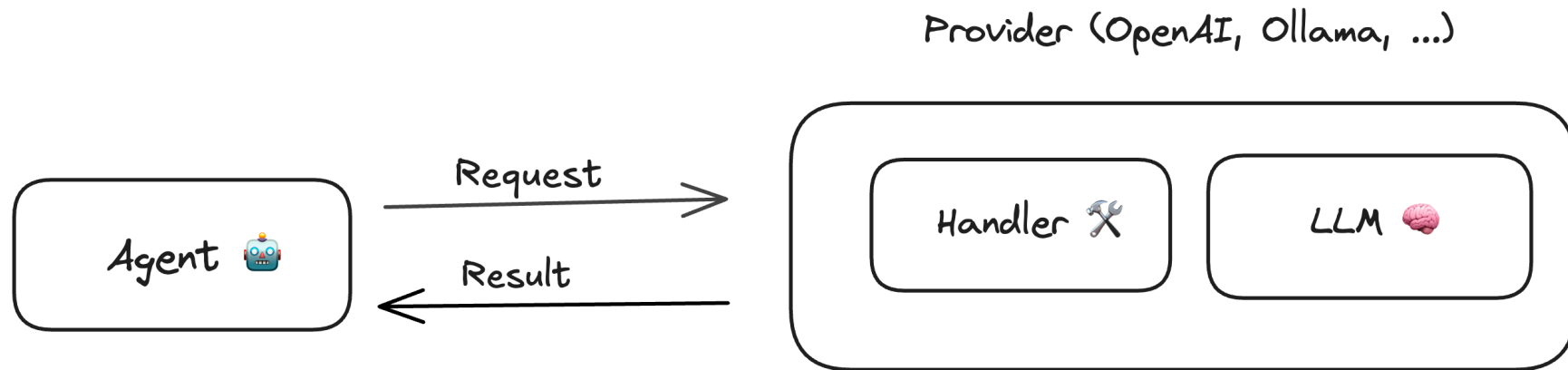
(base) oli@olivers-MacBook-Pro-2 weather %

Ln 1, Col 1 Spaces: 2 UTF-8 LF JavaScript Go Live Prettier

Only one line of code has been changed, perfect!

How does this work (highlevel)

How Agents Communicate with LLM Providers (one shot)



- The Agent 🤖 (Roo/Cursor) sends a request to the Provider (e.g. OpenAI, Ollama, ...).
- The Handler 🔧 makes a single string from the request (e.g. system prompt / user prompt*), tokenizes it to $y_{t:1}$ and sends it to the LLM.
- The LLM 🧠 calculates the probability distribution $p(y_{t+1}|y_{t:1})$ of next token y_{t+1}
- The Handler repeatably samples the tokens and sends them it to the agent

*Also, might add context from previous conversations.

Creating the prompt (Conceptually)

- The system prompt and the user prompt (optionally other prompts) are sent to the provider
- A **single prompt / string** is then built using a template like

```
<|system|>
You are Roo, a helpful...
<|user|>
Please add the location St. Anton...
```
- Important: What enters LLM is just text / tokens $y_{t:1}$
- The handler then **repeatedly samples tokens** until the response is complete.

Under the Hood

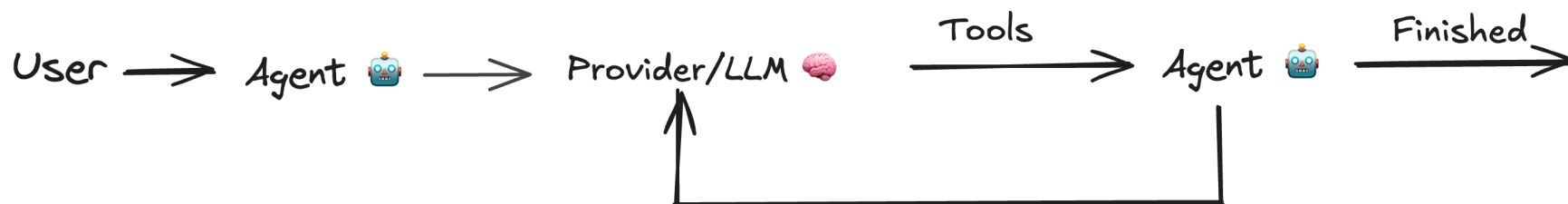
- Providers use **optimizations and tricks** beyond simple prompt assembly
 - **KV-cache reuse**
 - Caching the internals of the transformer which take advantage of the autoregressive nature
 - **Prompt caching**
 - System prompt (and other input at the beginning stays constant)
 - Transformer architecture allows for caching the beginning of network

Likely providers such as OpenAI also have a secret sauce.

The agentic loop

- This usually happens in a loop (aka Reason Act Loop, ReAct)
 1. The Agent sends **the system prompt** (“You are Roo, a helpful...”) and the **user prompt** (“Please add the location St. Anton...”) to the Provider.
 2. The Provider returns the response of the LLM to the Agent (**usually request to use a tool**).
 3. The Agent fulfills the request (e.g. calls the tool) and sends the result back to the LLM.

Step 2 and 3 are iterated until success (kind of while loop)*

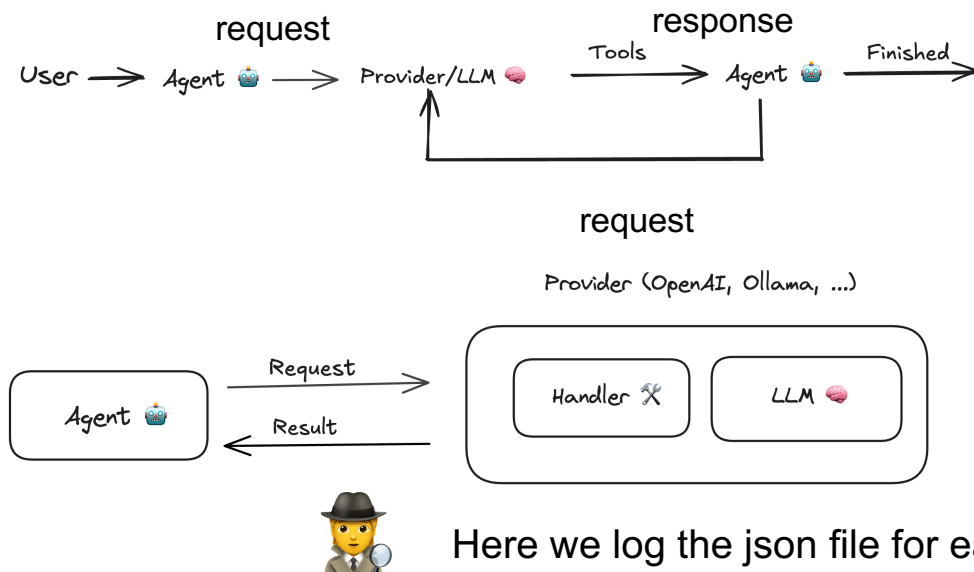


*In principle, also other non-loop-like calls are possible.

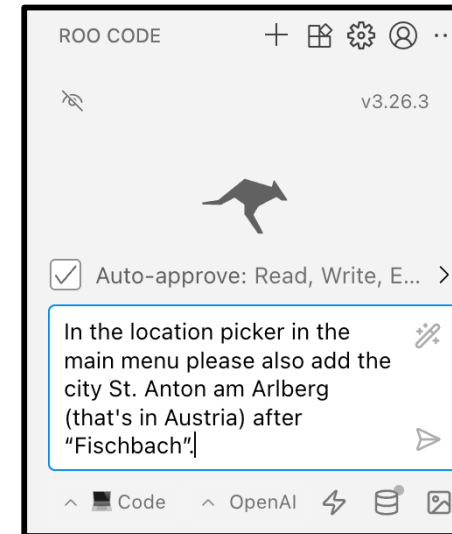
ReAct: <https://arxiv.org/abs/2210.03629>

Eavesdropping the communication

Eavesdropping communication



Here we log the json file for each communication












The screenshot shows a code editor with two files: "request.json" and "result.json". The "request.json" file contains a JSON object with a "request" property, which is an array of messages. The "result.json" file contains a JSON object with a "response" property, which is a string. The code editor also shows a "RUN AND DEBUG" panel on the left with variables and a "JS browser.mjs" file on the right.

Writing out the request and result (shown is Ollama)

[Installing Roo-Code in development mode](#)

Three rounds happend

- Using openAI's GPT-4.1, three rounds in the loop happened (see json files)

 openai-log-viewer.html	Yesterday at 22:41	35 KB	HTML text
▼  llm_input	Yesterday at 21:50	--	Folder
 openai-native-input-2025-09-01T17-56-34-674Z.json	Yesterday at 19:56	44 KB	JSON File
 openai-native-input-2025-09-01T17-55-33-852Z.json	Yesterday at 19:55	76 KB	JSON File
 openai-native-input-2025-09-01T17-54-33-066Z.json	Yesterday at 19:54	44 KB	JSON File
▼  llm_output	Yesterday at 21:50	--	Folder
 openai-native-output-2025-09-01T17-56-43-623Z.json	Yesterday at 19:56	874 bytes	JSON File
 openai-native-output-2025-09-01T17-55-42-902Z.json	Yesterday at 19:55	2 KB	JSON File
 openai-native-output-2025-09-01T17-54-39-349Z.json	Yesterday at 19:54	1 KB	JSON File

- openai-log-viewer.html is a vibe-coded simple viewer for the json files
- Files are available at
- [https://www.dropbox.com/scl/fo/8gtzr2qi3boil0r8fhiqy/AK-
dfu4WPI7M9PXlufd348M?rlkey=z46p92u1ps2tao7rtarg7wd0x&st=s3zjd
nwd&dl=0](https://www.dropbox.com/scl/fo/8gtzr2qi3boil0r8fhiqy/AK-
dfu4WPI7M9PXlufd348M?rlkey=z46p92u1ps2tao7rtarg7wd0x&st=s3zjd
nwd&dl=0)

Step 1 Agent to provider/LLM (Principle) →

Raw Request

```
44167 characters

{
  "requestBody": {
    "model": "gpt-4.1",
    "input": [
      {
        "role": "user",
        "content": [
          {
            "type": "input_text",
            "text": "<task>\nIn the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after",
          },
          {
            "type": "input_text",
            "text": "<environment_details>\n# VSCode Visible Files\njs/main.js\n\n# VSCode Open Tabs\njs/main.js\n\n# Current Time\nCurrent",
          }
        ]
      }
    ],
    "stream": true,
    "store": true,
    "instructions": "You are Roo, a highly skilled software engineer with extensive knowledge in many programming languages, frameworks, des",
    "temperature": 0,
    "max_output_tokens": 32768
  },
  "model": {
    "id": "gpt-4.1",
    "info": {
      "maxTokens": 32768,
      "contextWindow": 1047576,
      "supportsImages": true,
      "supportsPromptCache": true,
      "inputPrice": 2,
      "outputPrice": 8,
      "cacheReadsPrice": 0.5,
      "supportsTemperature": true
    }
  },
  "format": "openai",
  "maxTokens": 32768,
  "temperature": 0
},
"metadata": {
  "mode": "code",
  "taskId": "43f2c6e9-cfd9-4813-b90b-d9ef7329d341"
}
}
```

User msg

System msg

Round 1 System msg 41415 chars, first entries

You are Roo, a highly skilled software engineer with extensive knowledge in many programming languages, frameworks, design patterns, and best practices.

==== MARKDOWN RULES

...

TOOL USE

You have access to a set of tools that are executed upon the user's approval. You can use one tool per message, and will receive the result of that tool use in the user's response. You use tools step-by-step to accomplish a given task, with each tool use informed by the result of the previous tool use.

Round 1 System Msg cont'd (Tools)

...
Tools

Explaining the tool read_file for the LLM



read_file

Description: Request to read the contents of one or more files. The tool outputs line-numbered content (e.g. "1 | const x = 1") for easy reference when creating diffs or discussing code. Supports text extraction from PDF and DOCX files, but may not handle other binary files properly.

****IMPORTANT:** You can read a maximum of 5 files in a single request.** If you need to read more files, use multiple sequential read_file requests.

Parameters:

- args: Contains one or more file elements, where each file contains:
 - path: (required) File path (relative to workspace directory
/Users/oli/Documents/GitHub/weather)

Usage:

<read_file>

<args>

<file>

<path>path/to/file</path>

</file>

</args>

</read_file>

Round 1 System Msg cont'd

Addition Tools (in system prompt, GPT-summarized)

From the JSON you showed (the system prompt's **Tools** section), the LLM is allowed to call these tools:

read_file — read up to 5 files (with line numbers).

fetch_instructions — fetch “how-to” instructions for tasks (e.g., create_mcp_server, create_mode).

search_files — regex search across a directory, returns matches with context.

list_files — list directory contents (optionally recursive).

list_code_definition_names — extract function/class/method names from files/dirs.

apply_diff — make **surgical** search/replace edits in a file via precise diff blocks.

write_to_file — create/overwrite a file with full content (single-shot).

insert_content — insert new lines into a file at a given line (or append).

search_and_replace — find/replace (literal or regex) within a file, with optional line range.

execute_command — run a CLI command (with optional working directory).

ask_followup_question — ask the user for missing details (with suggested answers).

attempt_completion — finalize the task and present the result (only after confirming prior tool success).

switch_mode — switch to another mode (code / ask / architect / debug / orchestrator).

new_task — start a new task instance in a chosen mode.

update_todo_list — replace the whole TODO checklist with updated statuses.

The green one have been picked up be the agent.

Round 1 User Msg

Text that I entered

<task>

In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after "Fischbach".

</task>

<environment_details>

VSCode Visible Files
js/main.js

VSCode Open Tabs
js/main.js

Current Time
Current time in ISO 8601 UTC format: 2025-09-01T17:54:33.033Z
User time zone: Europe/Berlin, UTC+2:00

....
Current Workspace Directory (/Users/oli/Documents/GitHub/weather) Files
DEVELOPMENT.md
favicon.ico
fetch_weather_data.py
index.html
README_weather_fetcher.md
README.md
test_weather_station.js
tester_konstanz_weather.py
.cursor/
.vscode/
doc/
doc/getting_data.ipynb
</environment_details>

Round 1 Response from LLM (GPT-4.1)

<thinking>

The user wants to add "St. Anton am Arlberg" (Austria) to the location picker in the main menu, specifically after "Fischbach". Based on the file structure, the location picker logic is likely in `[`js/locationPicker.js`](js/locationPicker.js:1)`, but it could also be referenced or rendered in `[`js/main.js`](js/main.js:1)`. Since the user has both files available and the picker is likely defined or at least configured in ``locationPicker.js``, I should read both files to find where the list of locations is defined and how to insert the new city after "Fischbach".

I will read both `[`js/locationPicker.js`](js/locationPicker.js:1)` and `[`js/main.js`](js/main.js:1)` to get the full context and ensure I update the correct place.

</thinking>

<read_file>

<args>

<file>

<path>js/locationPicker.js</path>

</file>

<file>

<path>js/main.js</path>

</file>

</args>

</read_file>

Tool call for the two files

Summary of round 1

Usage & Costs			
INPUT TOKENS	OUTPUT TOKENS	TOTAL TOKENS	TOTAL COST
8,950	230	9,180	\$0.0197

Preparation for round 2

The agent now uses the tool / executes the command.
Here it reads the 2 files (js/locationPicker.js) and (js/main.js)
and appends them to the user msg

Round 2 (raw request)

76308 characters

```
{
  "requestBody": {
    "model": "gpt-4.1",
    "input": [
      {
        "role": "user",
        "content": [
          {
            "type": "input_text",
            "text": "[read_file for 'js/locationPicker.js', 'js/main.js'] Result:"
          },
          {
            "type": "input_text",
            "text": "<files>\n<file><path>js/locationPicker.js</path>\n<content lines=\\\"1-323\\\">\n  1 | /**\n  2 |  * Location Picker Module"
          },
          {
            "type": "input_text",
            "text": "<environment_details>\n# VSCode Visible Files\njs/main.js\n\n# VSCode Open Tabs\njs/main.js\n\n# Current Time\nCurrent"
          }
        ]
      }
    ],
    "stream": true,
    "store": true,
    "instructions": "You are Roo, a highly skilled software engineer with extensive knowledge in many programming languages, frameworks, des",
    "temperature": 0,
    "max_output_tokens": 32768,
    "previous_response_id": "resp_68b5ddd9c7d4819193fc38d2d3b94d1f027f2fdceca5f9b3"
  },
  "model": {
    "id": "gpt-4.1",
    "info": {
      "maxTokens": 32768,
      "contextWindow": 1047576,
      "supportsImages": true,
      "supportsPromptCache": true,
      "inputPrice": 2,
      "outputPrice": 8,
      "cacheReadsPrice": 0.5,
      "supportsTemperature": true
    },
    "format": "openai",
    "maxTokens": 32768,
    "temperature": 0
  },
  "metadata": {
    "mode": "code",
    "taskId": "43f2c6e9-cfd9-4813-b90b-d9ef7329d341"
  }
}
```

User msg (new)

System msg, stays the same

Round 2 (User prompt, with additional file information)

[read_file for 'js/locationPicker.js', 'js/main.js'] Result:

<files>

<file><path>js/locationPicker.js</path>

<content lines="1-323">

```
1 | /**
2 |  * Location Picker Module
3 |  * Provides map-based location selection functionality
4 |  */
```

Content of the first file.

...

<file><path>js/main.js</path>

<content lines="1-482">

```
1 | let measured_temp = null;
2 | let measured_water_temp = null;
```

...

```
41 | // Available locations and models
42 | const locations = [
43 |   { name: "🇩🇪 Konstanz", lat: 47.6952, lon: 9.1307 },
44 |   { name: "🇨🇭 🏔 Chäserrugg", lat: 47.1549, lon: 9.3128 },
45 |   { name: "🇨🇭 🏔 Wildhaus", lat: 47.2033, lon: 9.3505 },
46 |   { name: "🇨🇭 Zurich", lat: 47.3769, lon: 8.5417 },
47 |   { name: "🇫🇮 Espoo", lat: 60.205490, lon: 24.655899},
48 |   { name: "🌲🌲 Fischbach", lat: 48.157652, lon: 8.487578 }
49 | ];
```

Content of the second file.

Round 2: Response

<thinking>

The location picker options are defined in the `locations` array in [js/main.js](js/main.js:42-49). The dropdown is populated by iterating over this array and appending each location as an option. To add "St. Anton am Arlberg" after "Fischbach", I need to insert a new object for St. Anton am Arlberg (with its coordinates) immediately after the Fischbach entry in the array.

Coordinates for St. Anton am Arlberg, Austria are:
lat: 47.1287, lon: 10.2640

I will update the `locations` array to include:

```
```js
{ name: "🇨🇭 St. Anton am Arlberg", lat: 47.1287, lon: 10.2640 }
```
```

immediately after the Fischbach entry.

</thinking>

<apply_diff>

<path>js/main.js</path>

<diff>

<<<<<<< SEARCH

:start_line:47

```
{ name: "🇫🇮 Espoo", lat:60.205490, lon: 24.655899},
{ name: "🌲🌲 Fischbach", lat: 48.157652, lon: 8.487578 }
];
=====
{ name: "🇫🇮 Espoo", lat:60.205490, lon: 24.655899},
{ name: "🌲🌲 Fischbach", lat: 48.157652, lon: 8.487578 },
{ name: "🇨🇭 St. Anton am Arlberg", lat: 47.1287, lon: 10.2640 }
];
```

>>>>>>> REPLACE

</diff>

</apply_diff>

It knows the location of Arlberg, out of the box. This needs checking!

That's a tool call

Round 2: Summary

Usage & Costs

INPUT TOKENS

18,009

OUTPUT TOKENS

367

TOTAL TOKENS

18,376

TOTAL COST

\$0.0390

The diff is executed and send to the provider/LLM

Round 3 (Finishing off)

User (System stays like it is)

1113 characters

[apply_diff for 'js/main.js'] Result:

```
<file_write_result>
<path>js/main.js</path>
<operation>modified</operation>
<notice>
<i>You do not need to re-read the file, as you have seen all changes</i>
<i>Proceed with the task using these changes as the new baseline.</i>
</notice>
</file_write_result>
```

274 characters

```
<attempt_completion>
<result>
"St. Anton am Arlberg" (Austria) has been added to the location picker immediately after "Fischbach" in the main menu. The new entry uses the
correct coordinates and will now appear in the dropdown for selection.
</result>
</attempt_completion>
```

Costs

- Costs For the single line change

ROO CODE

Task

In the location picker in the main menu please also add the city St. Anton am Arlberg (that's in Austria) after "Fischbach".

Context Length

18.7k

1

Tokens

↑ 45.6k ↓ 661

API Cost

\$0.10

Size

192 kB

Model

gpt-4.1

Context Window: 1,047,576 tokens

Max output: 32,768 tokens

✓ Supports images

✗ Does not support computer use

✓ Supports prompt caching

Input price: \$2.00 / 1M tokens

Output price: \$8.00 / 1M tokens

Cache reads price: \$0.50 / 1M tokens

GPT5 is (currently)
cheaper! Also
codex-mini-latest \$0.03

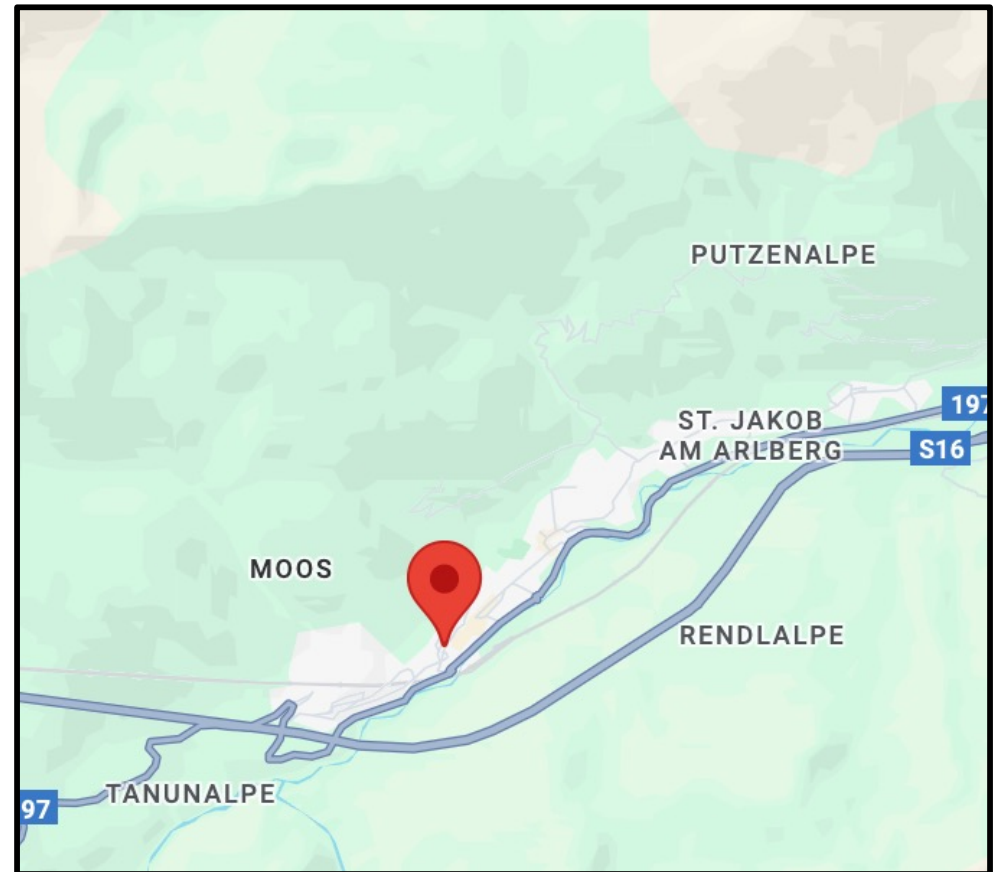
Sidenote Be skeptical and don't trust the LLMs!

- The line added was
{ name: "🇦🇹 St. Anton am Arlberg",
lat: 47.1287, lon: 10.2640 }

Generate google maps link for lat: 47.1287, lon: 10.2640

- This time it worked but I also had runs with hallucinations.

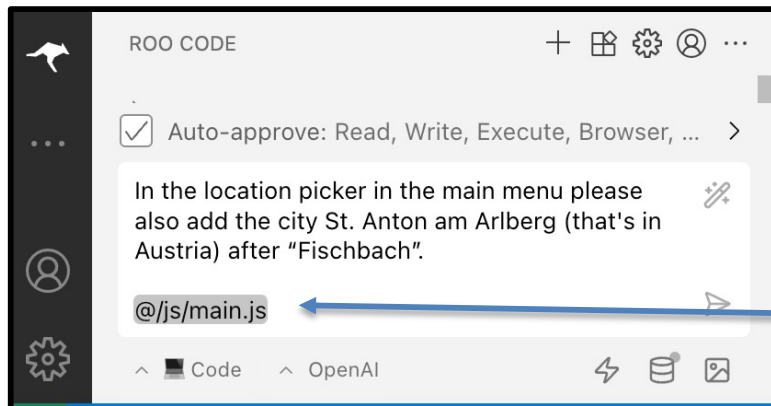
<https://www.google.com/maps/search/?api=1&query=47.1287%2C10.2640>



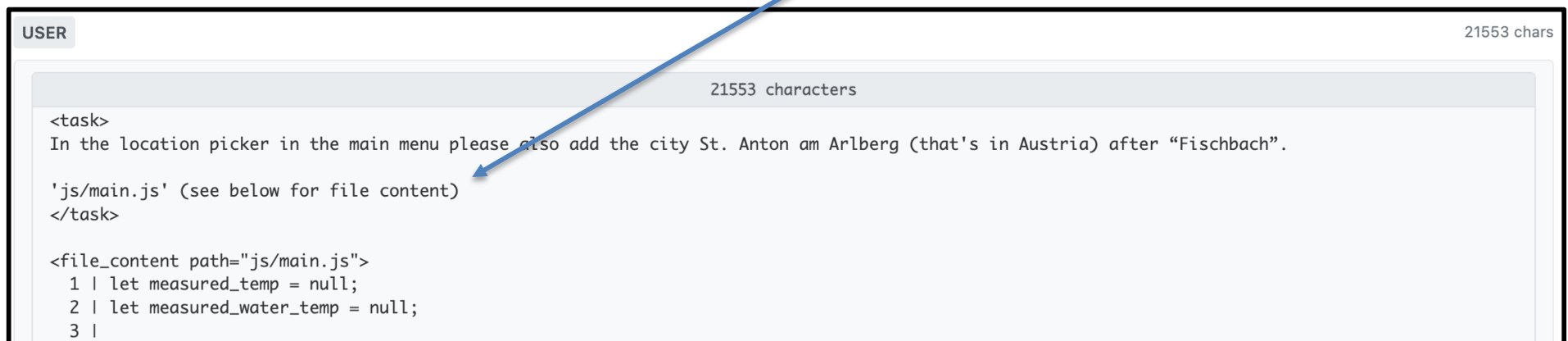
Practical takeaways

Adding Content (in the prompt)

You can help the Agent by providing extra content

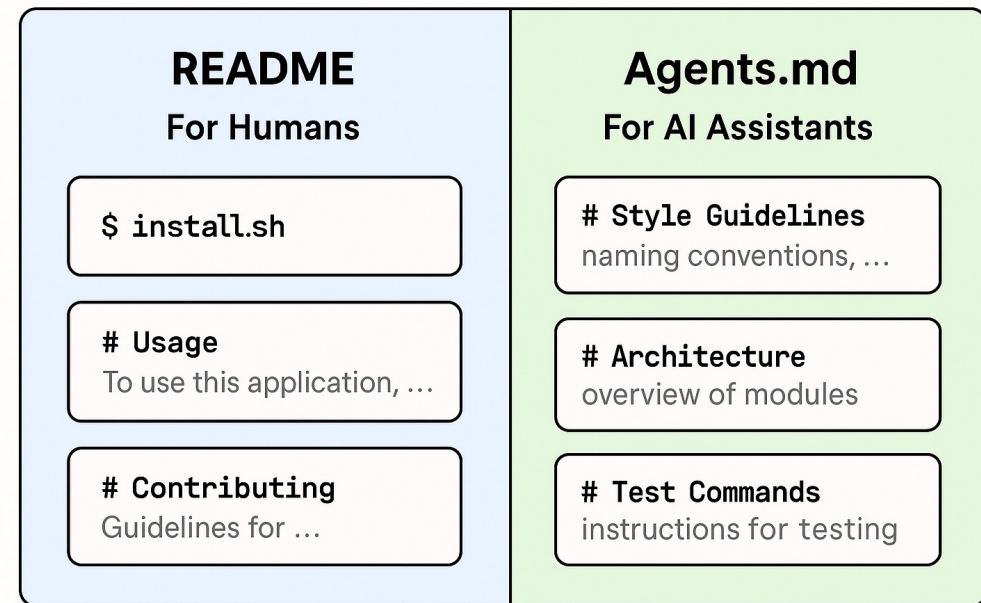
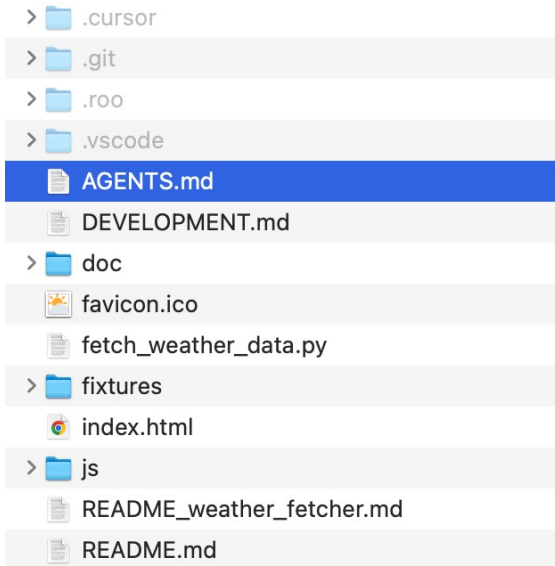


Add the content of
js/main.js to the user prompt



GPT4.1 then does the job in a single round (Ollma models still fail).

Project Context (AGENTS.md)



If present the file AGENTS.md is added always after the system prompt.
All major agents: cursor, codex, Roo supports this now (August 2025).
Replaces custom solutions (like CLAUDE.md)

See <https://agents.md/>

Example for AGENTS.MD for Deep Learning

Tell it to use Keras

```
# AGENTS.md - Keras-first, PyTorch fallback
```

```
## Policy
```

- Implement everything in **Keras 3** by default.
- Fallback to **PyTorch** if Keras cannot express the feature or if the user requests it.
- All tests use the **pytest** framework.

Tell it which tests to run

```
## Protocol
```

1. Apply minimal diffs
2. Validate with **unit + smoke tests**
 - You may skip the **long test suite**.

```
## Device
```

- Auto-select: ``cuda`` if available → else ``mps`` (Apple) → else ``cpu``.

```
## Tests layout
```

```
...
```

```
tests/
```

- ├ unit/ # fast checks
- ├ smoke/ # tiny training runs (≤2 batches, CPU OK)
- └ long/ # integration/benchmark (may skip)

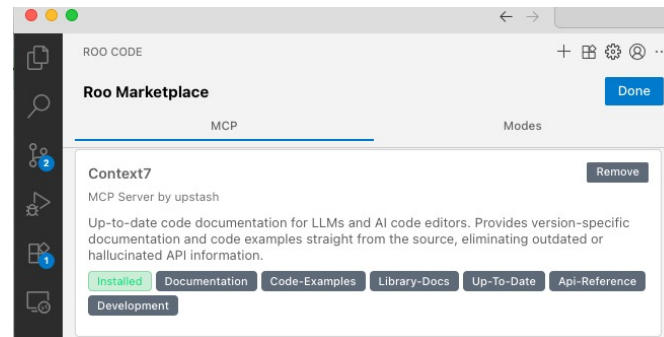
```
...
```

Context Management in Cursor

- Use ~~Cursor Rules~~ AGENTS.md
 - Define project-wide behaviors like “always use TypeScript import”
 - These are always included in the context
- Add Context in Prompt
 - You can webpage with documentation and best practices (use @)
- Context Management by Cursor
 - Automatic context building: Cursor includes files you open or edit into the LLM context.
 - Smart truncation: If files are too large, only the most relevant parts are sent.
 - Session memory: Keeps track of recent edits & conversations, discards irrelevant history uses RAG (cursor’s secret sauce)

Adding Tools via MCP Protocol (Roo-Code)

- You can add your own tools using the MCP protocol (here context7)



- The added MCP Server now appears in the System Instruction.

SYSTEM INSTRUCTIONS

47238 chars

```
47238 characters

## context7 (`npx -y @upstash/context7-mcp`)

### Instructions
Use this server to retrieve up-to-date documentation and code examples for any library.

### Available Tools
- resolve-library-id: Resolves a package/product name to a Context7-compatible library ID and returns a list of matching libraries.

You MUST call this function before 'get-library-docs' to obtain a valid Context7-compatible library ID UNLESS the user explicitly provides a library ID in the format '/org/project' or '/org/project/version' in their query.

Selection Process:
1. Analyze the query to understand what library/package the user is looking for
2. Return the most relevant match based on:
  - Name similarity to the query (exact matches prioritized)
  - Description relevance to the query's intent
  - Documentation coverage (prioritize libraries with higher Code Snippet counts)
  - Trust score (consider libraries with scores of 7-10 more authoritative)

Response Format:
- Return the selected library ID in a clearly marked section
- Provide a brief explanation for why this library was chosen
```

Special MCP Servers for Coding

- Context7 has documentation of 15k libraries (including R's ggplot :-)
 - For Cursor install via <https://docs.cursor.com/en/tools/mcp>
 - Use something like “use context7” in prompt
- Other MCP Servers
 - Github
 - MCP servers for Memory Management
 - Better than the ones provided by the Agents?

Costs / Locally Hosted LLM

- For the adding Arlberg task grok-code-fast-1 needed approx 12 seconds (using cursor), also running linter test.

| | |
|--------------|------------------|
| Cache Read | 1,448,448 |
| Cache Write | 0 |
| Input | 3,477 |
| Output | 2,043 |
| <hr/> | |
| Total | 1,453,968 |

Sep 1, 10:40 PM

grok-code-fast-1

Included

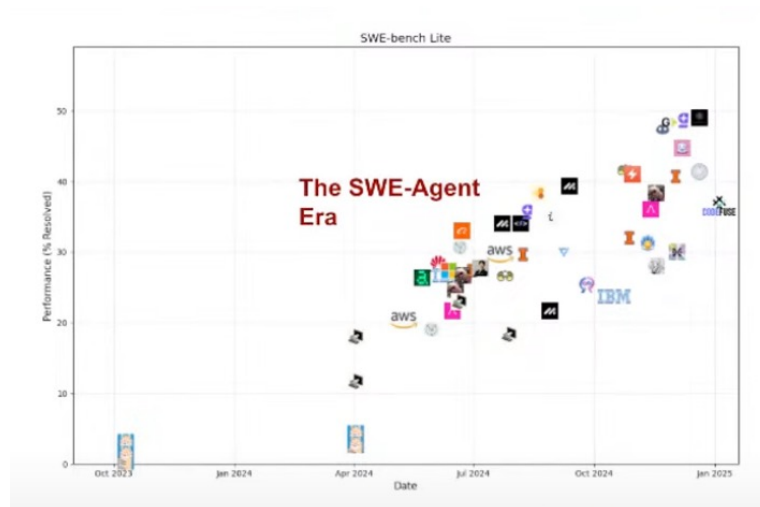
1.5M tokens ⓘ









~~\$0.03~~ Included

- Costs for my extensive coding streak (2 weeks) sum up to ~ 100 USD
- Local LLM could not do the task?
 - only tried 8B models (probably the wrong ones) and did not try hard
 - My belief, just the matter of time when small specialized local LLMs can better use the tools the agents provide them.

How good is it?

- <https://www.swebench.com/>
 - Task: Create fixes in GitHub repositories
 - Different versions of the benchmark also include a standard agent



| Model | <u>% Resolved</u> | Org | Date |
|---|-------------------|--|------------|
|  ExpeRepair-v1.0 + Claude 4 Sonnet | 60.33 |  | 2025-06-25 |
| Refact.ai Agent | 60.00 |  | 2025-04-25 |
|  SWE-agent + Claude 4 Sonnet | 56.67 |  | 2025-05-26 |
| ExpeRepair-v1.0 | 48.33 |  | 2025-06-13 |
|  SWE-agent + Claude 3.7 Sonnet | 48.00 |  | 2025-02-26 |

From 11% to 60% success in 2 years

Literature / Links (fast moving field)

- SWE-Bench (Benchmarks and simple agent in Python*!)
 - <https://www.swebench.com/original.html>
 - <https://www.youtube.com/watch?v=yAQw77juYSg>

A lot of hype noise

- Reddit: <https://reddit.com/r/cursor> r/AI_Agents/ r/ChatGPTCoding
- Youtube
 - Andrej Karpathy: Software Is Changing (Opinion)
 - <https://youtu.be/LCEmiRjPEtQ?si=YDpMXjbRqZDKVZfS>
 - https://www.youtube.com/@matthew_berman (Channel)

Love to get your input!

Also, on applications beyond coding (we plan to use it in the context of Statistics / Causality)