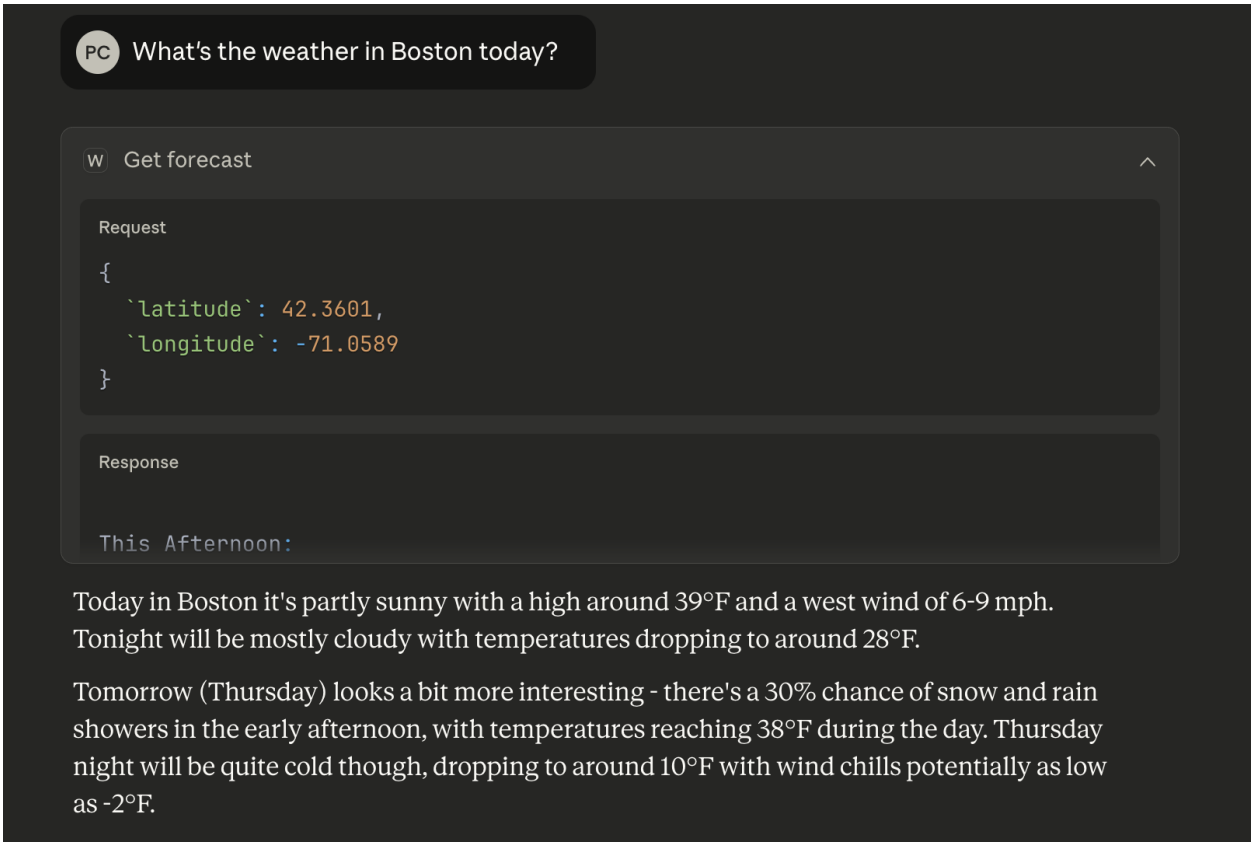
CS 554: Assignment 3 MCP and Model distillation

Author: Phong Cao

Task 1:

1. Working weather sever:



2. Get Company detail tool:
   Tool name: get_company_details
   Input:
   company_id (str): A company name or stock symbol. E.g Apple or APPL.
   Fetching and processing:
   - The tool call external API (e.g. via HTTP, Fetch, Axios, GraphQL) using company_id in the input.
   - It's look up all the information in external database and sever.
   - Return back the information to the calling endpoint.
   - Check the return status such as 404 -> Failed and 200 is good.
   - If 200, proceed and parsing the data to output variables.
     - E.g: stock_price: float, company_overview: str, employee_num: int

   Output format with human readable: The tool returns predefined template that filled in the variable such as:

- "Apple Inc. (AAPL) is currently trading at 192.35 USD.
  The company has about 161,000 employees.
  Overview: Apple designs and sells iPhone, Mac, and Apple Watch."

3.

- The failback is important because without it, the LLM might hallucinate when the external tool call is failed.
- Clear message when checking if the data is correctly retrieved will help LLM notice if the data is usable or not.
- Some of the typical errors might be:
  - Invalid inputs such as they calling weather for.
  - Network, typical HTTP errors.
  - Incomplete, wrong json return from the API.
- For prevent this:
  - Try, except code.
  - Check HTTP return status. For example, if 200 -> OK else e.g. 404, 500 return unable to fetch the relevant information.

4.

- I would return the structural output such as json and the information that match the other function calling.
- For example
  - Function 1: get_top_company: Finding the highest performance company
    - Input: Top-k company stock performance => k: int
    - Output: Get the top-k company-> return the name and current stock price.

    {
    “APPL”: 50,
    “DELL”: 20,
    “AMZ”: 100,
    }
  - Function 2: External information: Finding other information of retrieve company from function 1
    - Input: Machine output such as json from function 1
    - Output: Iterate a for loop and calling external API for each of the retrieved company and get machine output if we the consumer is machine and human readable if human.

- Machine:

{

“APPLE”: Company sell iphone

“DELL”: Company sell laptop

...

}

- If human: “From the top 5 retrieval companies. Apple is selling iphone and their stock price at 50. (Same for Dell and Amazon)”

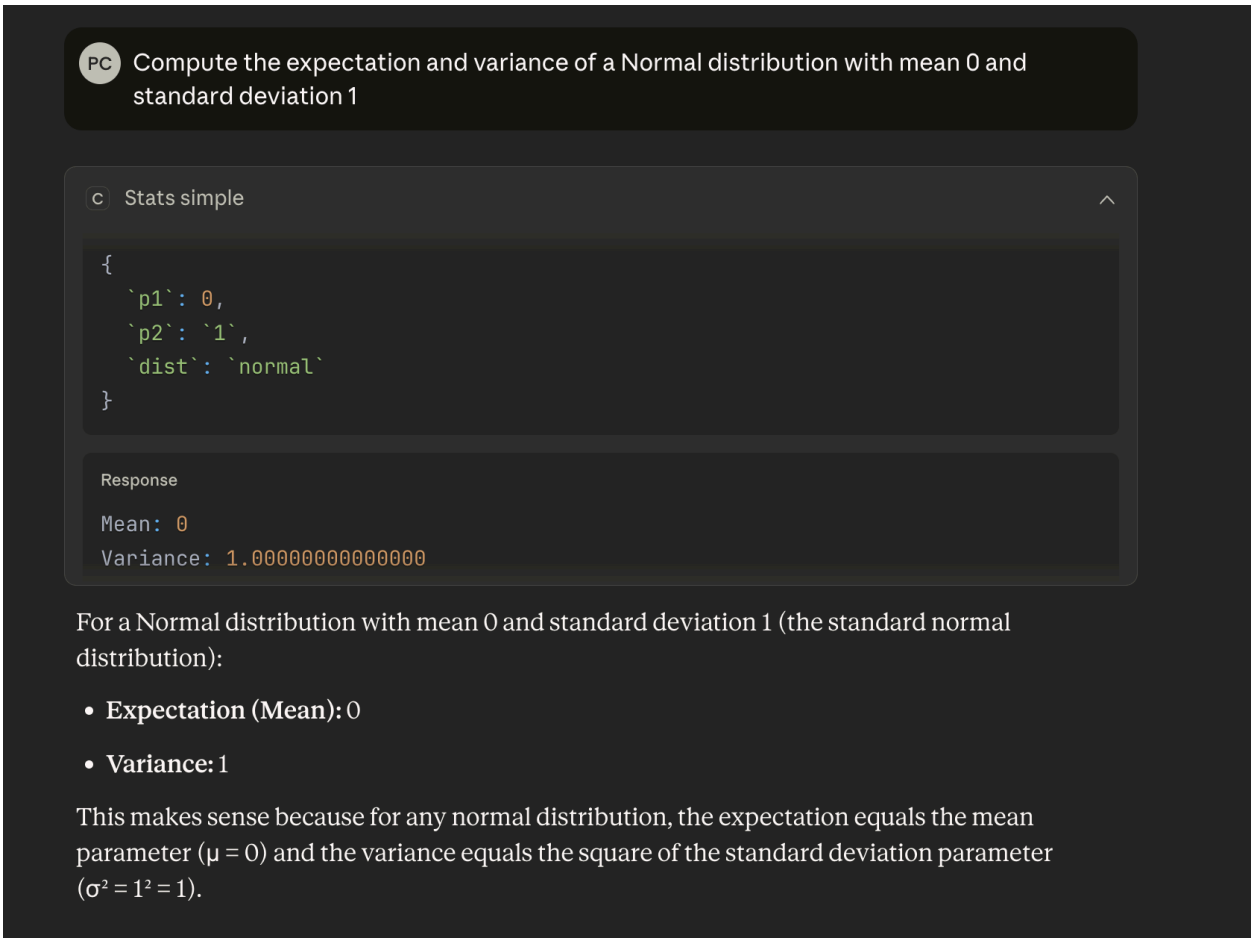
Task 2:

1.

2. The reason we separate symbolic expressions and numeric is because they play different roles in math. A symbolic expression like x+1 can’t be reduced to a single number since x is unknown. However, something like1+1 can be evaluated to 2. This

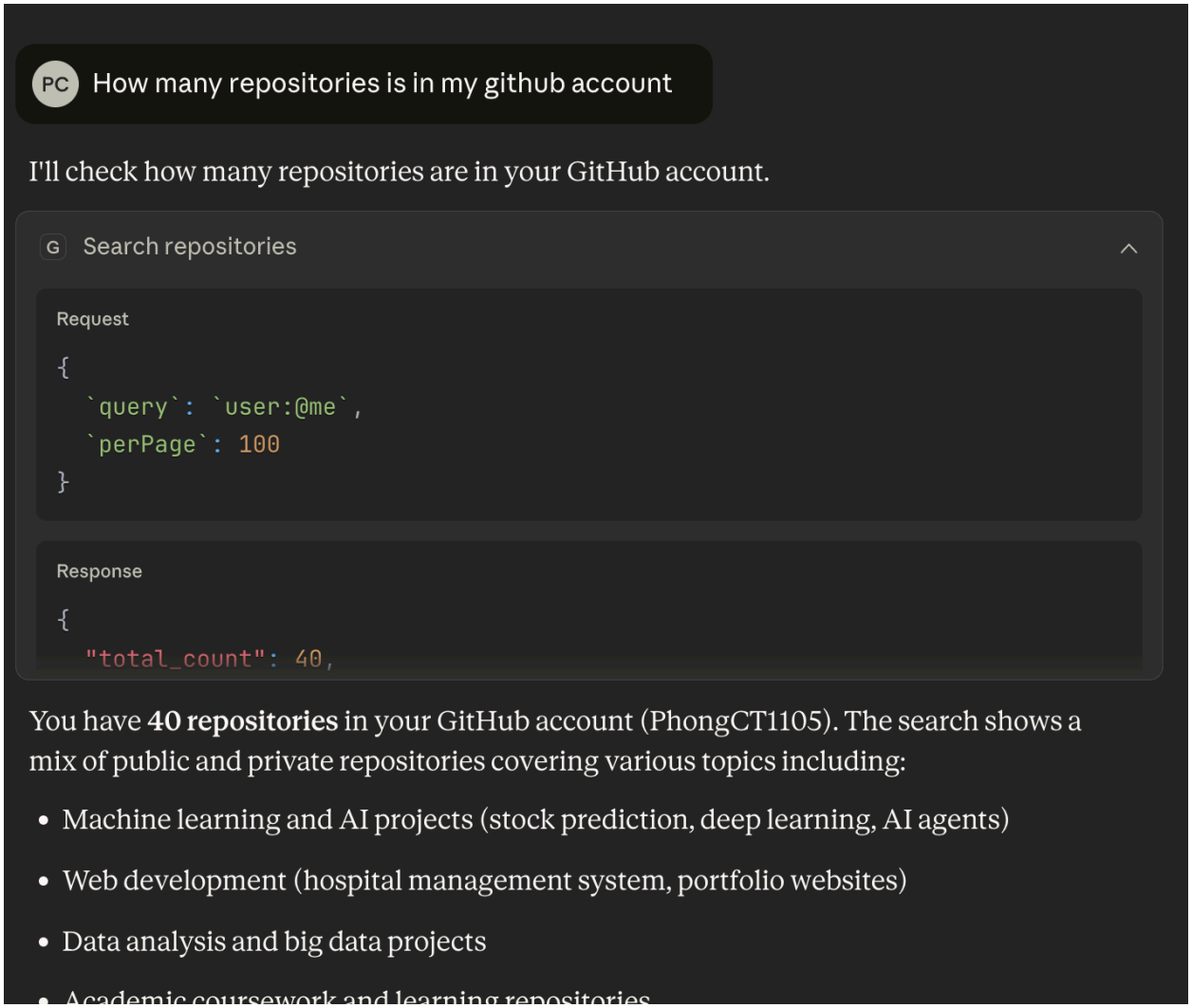
difference of token also matters for operations like derivatives or integrals. For example, taking the derivative of a number gives 0, but the derivative of X depends on the variable. Keeping symbolic and numeric results separate makes sure the tool can still do algebra when needed and only gives a number when the expression is fully known.

3. Edges cases in this math tool:
    - Invalid math equation such as divided by 0 or square root of negative, ...
    - Negative order when derivative
    - Incorrect or missing input such as having open parentheses but missing closes.
    - ⇨ Lots of problem both logically, mistyping, etc from the user that cannot be solve or handle by the library.
4. Distribution calling:



Task 3:

1.



I chose the GitHub MCP server. This server lets the LLM interact directly with GitHub repositories from my account. It utilizes many Github functions as tools, such as reading files, listing repositories, checking commits, searching code, and managing issues or pull requests. This means it gives the model access to real development data and allows it to perform repository-level operations instead of just computing something locally which mean more actionable.

Compared to my calculator and weather servers, the GitHub server is much more powerful and works with real external data rather than simple math or API calls. The GitHub MCP server can read, navigate, and analyze entire codebases. It demonstrates

more advanced MCP features, including authentication, lots of tools interactions, and working with structured developer workflows. This making this MCP server having more actionable insight and function.

2.

I chose the Figma MCP server in Claude. The reason is that it works really well together with my existing GitHub MCP server to create a stronger UI/UX development workflow. With Figma, the LLM can inspect and understand my design files, while the GitHub server gives it access to my React components and project code. By combining both, the model can compare my actual implementation to the intended design, suggest improvements, detect inconsistencies, and help align the codebase with the UI/UX layout I created in Figma.

3.

MCP Sever:

- Benefits:
  - Standardized interface for tools.
  - Easy to scale since you just need to addig function/mcp server without changing others things.
  - Reuseable tool
- Drawbacks:
  - Technical overhead, lots of setup
  - More configuration
  - Slow, not suitable for task such as real time trading data
  - Slower development process, not suitable for small, PoC project

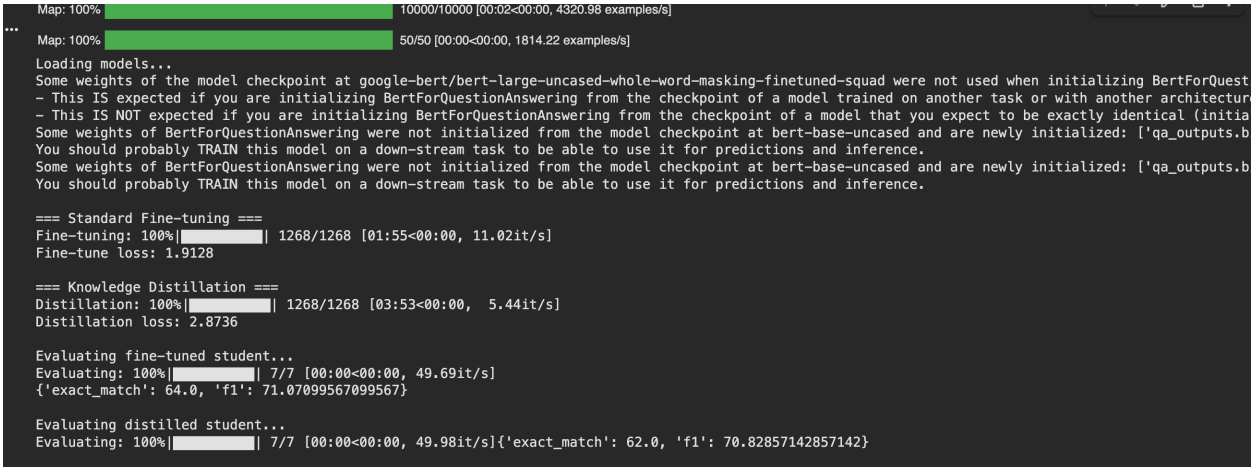
No MCP:

- Benefits:
  - Faster development process
  - Fast, good for real time data
  - Can running more active, background running, scheduler, ...
- Drawbacks:
  - Hard to scale since there is no standardized protocols
  - Less secure protocol for LLM
  - Manual context injection

Task 4:

1.

b.



c.

Hard labels are trained using the cross-entropy loss which just like normal fine-tuning. This makes sure the student still learns the exact start and end positions from the dataset.

Soft labels, on the other hand, come from the teacher's logits (distribution) and are compared using KL-divergence. These probabilities let the student learn not only the final answer but also the teacher's confidence and decision pattern.

The final KD loss is a weighted combination of both parts, so the student learns the correct labels while also mimicking how the teacher reasons which trying to balance the loss.

d.

Model distillation helps a smaller student model get close to the larger teacher's model performance by learning how the teacher makes decisions. This means the student model having the same performance with large model with smaller size, faster inference and easier to deploy. Distillation also removes a lot of noise from raw training data since the teacher's soft probabilities act as a smoother, more informative signal. Overall, we get a compact model that keeps most of the accuracy but with much lower compute cost compared to the original.

2.

1. Briefly explain the workflow of the teacher agent, including how it performs reasoning, selects tools (e.g., retrieval or code execution), and uses their outputs. (3pt)

The teacher agent workflow is:

- First reasoning (Thought) where the teacher planning and reasoning what tasks to do next and break down the problem

- Second is Tool Selection (Action) where based on it's planning, the teacher agent now can select the appropriate tool that match with its own reasoning.

- Thirdly is observation where the teacher agent seeing the output from the tool and consider if using more tool or moving to the next step in its reasoning.

With this workflow, the teacher agent work like a normal agent that fully capable of reasoning, using tool and evaluation by its own (reason-act-observe) pattern.

2. Describe how the distilled student learns to imitate this *reason–act–observe* behavior pattern, even without explicitly calling external tools. (3pts)

The student learns the reason–act–observe loop through supervised fine-tuning from the teacher. During training, the student only predicts the model-generated parts which mean the teacher's thoughts and actions while observations from tools are provided as context. This teaches the student when to trigger a tool, how to format tool calls, and how to continue reasoning after seeing the tool's output.

Because the student is learning the strategy rather than memorizing facts, it can generalize better. For example, instead of memorizing Apple's stock price like a normal CoT model, the student learns the behavior of think, call the stock tool, read the result and plan the next step. Even though the student does not actually execute tools during training, it still learns the correct pattern of when to act and how to react to feedback.

3. The paper introduces two techniques to help the student imitate the teacher agent: **first-thought prefix** and **self-consistent action generation**. Choose **one** and briefly explain what it is, why it matters, and how it helps the student generalize or use tools better. (3 pts)

First-thought prefix is a technique where the teacher agent generate its initial reasoning step before making any tool call. This early reasoning is included in the training data that the student learns it as part of the pattern. It matters because it teaches the student to pause and think before acting, instead of rushing to a final answer. By learning this structured "think first, act later" strategy, the student becomes more consistent in deciding when to use a tool and how to format the tool call correctly.