CMSC 35370: AI Agent for Science **Fall 2025**

Assignment 4 Report

Student: Chenxi Peng Due date: November 10th

1 Task 1: Manually complete a benchmark

This part of the report is a Jupyter notebook.

2 Task 2: One shot completion

I build one tool to load the dataset so that the reasoning model can actually read the whole IMU data which contains 4,048,332 rows and 6 columns. In order to store the computed sleep endpoints in a JSON file, I define another tool to create such file in the working path.

When I run the agent with only one user prompt, it successfully output the JSON file but with incorrect prediction results. By looking at the agent's reasoning, I find it is using mathematical expressions to predict the sleep endpoints, not realizing that the `biopsykit` package has a specific function for this: `sleep_processing_pipeline.predict_pipeline_acceleration()`.

To fix this, I added a system prompt that included a role definition and the necessary domain knowledge about the function. From the response, the agent actually use the function from `biopsykit` package, but the results are wrong compared to the reference prediction results got in the manual code.

While working on the task, I encountered a situation where the data index was inconsistent with the function's requirements, so I included this point in the system prompt. At the same time, I told the agent that the sampling rate was 204.8.

In the user prompt, I added a Chain-of-Thought instruction. This time, the agent did not give me the final results but instead provided a Python script, as it cannot (or could not) actually run the code.

I checked the Python script and found that the key part was missing:

```
sleep_endpoints = sleep_results["sleep_endpoints"]
results = {
    "sleep_onset": sleep_endpoints["sleep_onset"],
    "wake_onset": sleep_endpoints["wake_onset"],
    "total_sleep_duration": sleep_endpoints["total_sleep_duration"],
}
```

I think the reason might be the representation of string of data is too large for one agent to track effectively. Therefore, I created a multi-agent system in Task 3.

For this part, code can be found here.

3 Task 3: Create a multi-agent system

I create a multi-agent system including one main agent and two subagents. The main agent receives user response, and decides which subagent to call. Three subagents respectively undertake the tasks of generating the code, reviewing the code and executing the code.

However, the result is wrong. I think the reason may be the tool `PythonREPL` to run the Python code can not be called or implement the code in a less-than-ideal approach. Because I can see the generated code is correct but the result is not very satisfactory.

Compared to one-shot, the multi-agent system at least can generate correct code (after being reviewed by me), although the final result is not correct. So the multi-agent system performs better than the one-shot agent.

For this part, code can be found here.

4 Task 4: Create a multi-agent system

This part of the report is a Jupyter notebook.