SeaMetrics - A dialogue system for nautical weather report

Caroline Grand-Clement

November 3, 2025

1 Introduction

SeaMetrics is a dialogue system intended to be used for nautical navigation. It includes functions to check different weather parameters (temperature, precipitation, visibility and wind speed) in different locations.

2 Pre-existing services

2.1 Open Meteo's Weather Forecast API

Open Meteo's Weather Forecast API offers weather forecast with a number of parameters. It is completely free to use and does not require sign up, which made it perfect for this project's scope. I chose to focus on temperature, precipitation, wind speed and direction, and visibility, as those seemed the most relevant for nautical navigation. I am retrieving the weather forecast only for the location of the 4 chosen cities (see 4) for the day the system is used.

2.2 AISStream API

The AISStream API offers maritime information about ships, base stations, and safety issues. It uses WebSocket, and requires an API key for authentication that one can obtain by signing in with GitHub. While I would have preferred to use an API that did not require authentication (so that the system could be expanded more easily), this was still suitable for this project. I planned to only retrieve the PositionReport and SafetyBroadcastMessage type of messages, to track 'traffic' and warn users of the safety issues. However, I realized too late that using WebSocket with the browser presented much more of a challenge than I had time for, so I had to pass on using the AISStream API information.

2.3 Ollama

I used Ollama to connect to LLMs for chat generation, as seen in Lab 2.

3 Data collection

3.1 Scenario

The imagined scenario was of a person preparing for a boat trip, and then embarking on their journey. The person wants to check the weather conditions before departing. During the trip, they want to check the traffic conditions (how 'busy' the area is), as well as possible updates on weather conditions, such as wind speed and its direction.

3.2 Collected dialogues

Due to resource constraints, I chose to synthetically create dialogues for my scenario. I first created a base dialogue, and then created variants of it to get a broader idea of how the user would respond. At first, I wanted to have set responses from the system, so did not create variance for that. When testing my system, I took note of different wordings I used and integrated a growing number of them to the NLU.

Appendix 8.1 shows the base dialogue.

3.3 Distillation and analysis results

When coming up with example dialogues, and during development, I realized if the system only gave the same form of answers (”The wind is strong”, ”The temperature is low”, ”The visibility is good”, ...), it would be quite repetitive and dull for the user. I thus thought of integrating an LLM to rephrase my answers, so that it would introduce more variance. I explain this more deeply in 4.

4 Implementation

I decided to implement this system with SISU as there are a limited amount of functions it should accomplish, which meant I could do with less flexibility and more control. I started out by using my assignment for Lab 1 as a baseline for my code. I updated the IS domain with predicates, individuals, and plans I would need. I am starting with four cities (Gothenburg, Halmstad, Varberg and Marstrand) as possible locations for the user to go out on their boat. I implemented different plans, to find out the weather in general (only based on temperature and precipitation), or specific aspects (temperature, visibility, precipitation, wind). The system only needs to ask for the location once, as the location is not question dependent. This means that the user cannot check the conditions for different locations in a same run, but does avoid a lot of repetition.

I then implemented the connection to the Open Meteo Weather Forecast API in a separate file, so that responsibilities within my program would be kept separate. I used their template code provided for typescript, and tried to refactor it into functions, so that the system would make the API call from IS’s consultDB() with a specific location. However, this proved incredibly difficult, as working with asynchronous functions, so Promises, created many issues. The solution I then went for was not to wrap the API call and result manipulation in functions, but to make the API call as the program starts, and then only have a function that extracts the saved results for each city from a predefined dictionary.

Figuring out how to work with this API was a real challenge and took me more time than I had anticipated. For this reason, I decided to tackle something I was already familiar with next: integrating LLM responses. I still wanted to maintain some control over the responses, so rather than having the LLM generate responses from the NextMovesEvent, I wanted to use the NLG mapping to get an utterance and ask the LLM to rephrase the message. This did not go as planned. I ran into four issues:

- the LLM generates empty utterances very regularly, so the user would not hear anything, but the system assumed that it had executed its move correctly;
- when it did generate things, it took too much liberty and did not follow the system prompt to only rephrase the given utterance. I tested out lower top_k and top_p parameters, but did not find it to be very successful in limiting these issues;
- in that spirit, given the freedom to generate however many tokens, it often generated bigger paragraphs, but when using the num_predicted parameter to limit the length of predictions, sentences were often cut off in the middle;
- furthermore, we can see that LLMs are centered around written text, as it would often generate and speak characters like * (’asterisk’), which makes it sound completely unnatural.

Because of this, I decided to revert back to the original NLG utterances. My efforts to implement LLMs efficiently can be found commented out in the code, in the `isu.ts` file.

I then decided to tackle the AIS Stream API. I had spent more time than desired on trying to implement a satisfying LLM chat generation, and so had little time left to figure out WebSocket. I had thought that what I had learnt during my struggle with the weather API would come in handy again, but to my dismay it appeared quickly that it would involved additional libraries and tools that I unfortunately did not have adequate time to research and get accustomed to.

Finally, I decided to implement a ’goodbye’ move triggered by the user thanking the system or saying goodbye.

5 Sample dialogues

For sample dialogues, see appendices 8.2 to 8.4.

6 Discussion

This project encountered quite more obstacles than originally expected, and many compromises had to be made, which I have already mentioned in the implementation section (4)

One such compromises was that I had little flexibility to define different utterances - including the measures. I thus decided to use thresholds and define ranges for the weather metrics: hot/cold for temperature, dry/wet for precipitation, good/low for visibility, strong/light for wind speed, and north/east/south/west for wind direction. This means the weather report loses its precision, which may be important for some.

The integration of APIs was also something that was more challenging than I thought. While state machines can make it easy to use APIs (as seen in Lab 2 with the LLM chat generation), it was more difficult to come up with a working architecture that separated the responsibilities between information state, dialogue management, and database/API access.

Another compromise was having to scope down and exclude the AIS Stream integration from the project was disappointing, as it reduced the functionalities of the system considerably.

However, overall I believe it is the start of a useful project. Once in the middle of the action, it can be a hassle to search the weather on one's phone, so it can be more useful to have a hands-free option.

7 Conclusion

This project showed me a lot about the advantages of SISU (being able to control the flow of the conversation easily), but also highlighted its limitations. The architecture did not lend itself well to a smooth integration of APIs.

However, I also got to saw the limitations of working with LLMs. The creativety with which they can answer prompts was clearly not beneficial to this specific project, where the rigidity of the current NLG model was far more efficient in reaching the system's dialogue goals.

7.1 Future work

Future work would of course take the direction of integrating the AIS Stream API to the system, so as to keep track of traffic and safety issues. One could also explore whether or not having ranges for the weather metrics and not specific numbers affects how people view and trust the system, or whether it is sufficient for Sunday sailors.

8 Appendix

8.1 Base case dialogue

```
System:    Welcome. How can I help you?
User:      What is the weather?
System:    Where do you want to know the weather?
User:      Gothenburg.
System:    The weather is cold and wet. Bring a hat & raincoat,
           or stay inside!
User:      Will the visibility be okay?
System:    The visibility is not great. Be careful out there!
User:      Tell me about the wind.
System:    There is a light wind from the west.
User:      Is there much traffic?
System:    The traffic today is good.
```

8.2 Sample Dialogue A

```
System:    Hello! Welcome to SeaMetrics. How can I help you?
User:      How is the wind?
System:    Where do you want to know that information for?
User:      Varberg.
System:    There is a strong wind from the south. Hold on!
```

```
User:         Will there be rain?
System:       No precipitation today! Coast is clear.
User:         That's all.
System:       Thank you for using SeaMetrics. Have a nice day!
```

8.3 Sample Dialogue B

```
System:       Hello! Welcome to SeaMetrics. How can I help you?
User:         Weather please!
System:       Where do you want to know that information for?
User:         *no_input*
System:       I didn't hear anything from you. Where do you want
              to know that information for?
User:         Halmstad.
System:       The weather is hot and dry. It is suitable for a boat ride.
User:         Goodbye.
System:       Thank you for using SeaMetrics. Have a nice day!
```

8.4 Sample Dialogue C

```
System:       Hello! Welcome to SeaMetrics. How can I help you?
User:         Is there rain?
System:       Where do you want to know that information for?
User:         Here. [hardcoded to Gothenburg]
System:       There is a lot of precipitation. Bring a raincoat!
User:         What is the temperature?
System:       The weather is very cold. Bring a hat and & hot drink!
User:         Thank you, goodbye.
System:       Thank you for using SeaMetrics. Have a nice day!
```