2021.02.18

Lab II: Improvements Report

Rafal Černiavski

The app developed according to the lab instructions had many limitations. Some of the limitations were rather trivial and were therefore addressed before submitting the app. Some were, however, too challenging for the scale of the project and my limited knowledge of TypeScript. The present report will firstly cover the limitations that were resolved before submitting the assignment. Afterwards, it will provide insight on limitations that were too challenging yet crucial before the deployment (that is, if the app was to be deployed).

Trivial limitations:

1. Adjusted transitions

The finite state machine chart provided in the assignment was mainly improved by adjusting numerous transitions. For instance, whenever the details of a meeting are finalized, the user is now asked if they would like to also add the meeting to their to do list, which is one of the three intents the algorithm is trained to grasp. Even though the 'to_do_item' task is not implemented, this could be a useful addition, for it could boost the user's engagement as well as inform them about other tools available in the app or a software.

Another implemented transition worth being discussed is the 'canceled' state. The 'confirmation' state now provides the user with the choice of either confirming or canceling their appointment. Furthermore, if the appointment is canceled, the user is not taken back to the initial state of the appointment pipeline, but instead to the initial state of the overall app. In other words, if one cancels an appointment, they will be asked '*what would you like to do?'* rather than '*who are you meeting with?*'.

2. 'Be more specific'

Perhaps my favorite improvement, though it is debatable whether it is an improvement, is the addition of 'unsure' option for binary questions. Whenever the user is expected to answer to a question with *yes* or *no*, there is now an option to answer with words such as *maybe* or *I don't know*, resulting in transition to the 'unsure' state where the user is asked to be more specific. For instance, when the user is asked whether their meeting is going to take the whole day, they can answer with *perhaps* and, since such option is not implemented, the algorithm asks the user to be more specific and returns to the 'ask' state.

3. Grammar

Lastly, seemingly the most basic fix involved extending the grammar(s). To ensure that the algorithm is able to recognize the names, days, and times, the three were recorded in the grammar in a variety of formats. The following examples illustrate the extended grammar:

```
1. "John": {person: "John Appleseed"},
2. "John Appleseed": {person: "John Appleseed"},
```

3. "on Monday": {day: "Monday"},
4. "Monday": {day: "Monday"},
5. "10": {time: "10:00"},
6. "at 10": {time: "10:00"},
7. "ten": {time: "10:00"},
8. "at ten": {time: "10:00"},

Such adjustment minimized the number of transitions into the ‘.nomatch’ state, for input provided by the user no longer has to be very specific in its format. In other words, an extended grammar made it easier for the algorithm to grasp the intended person, time, and date. Nevertheless, it is still very limited to the words that are in the grammar. This limitation has not been fully resolved as is further discussed in the *unresolved limitations* section.

Unresolved limitations:

1. Limited vocabulary

The performance of the app is largely influenced by the vocabulary defined during the development. For instance, it is only possible to schedule an appointment with someone whose name has been specified in the vocabulary. Given the variety of names in addition to great variation of ways to provide the name as an input (e.g., *John Appleseed* vs. *John* vs. *Appleseed* vs. *with John* vs. *I’m meeting John*, etc.), the grammar is exponential, and it is therefore impossible to capture all the possible ways the user could attempt to provide the information.

A plausible solution would be to make use of the *recResult* output in the console and, instead of comparing it to what is stored in grammar, utilize the direct output of the console. In other words, the app would not require matching *recResult* against context, but instead use the raw output. This would allow to schedule an appointment with anyone, regardless of whether the name is stored in grammar. This will be attempted in further labs.

2. Language settings

The current version of the app is available only in English. Considering the scale of this project, the implementation of multiple languages appears a bit too time-consuming. Furthermore, such limitation might not be easy to transcend as the employed API, to the best of my knowledge, seems to be developed only for English. Nevertheless, in terms of the architecture, it would perhaps be possible to add an additional step preceding the ‘action menu’, where the user would have the option to choose their preferred language.

3. Switching off the app

The app currently runs in endless loops, as there is no way for the user to turn off the app without closing the browser window. The user should ideally have the option to switch off the app at any point or in every state. Currently it only seems possible with the addition of a separate condition where the user’s input would match a stop word. However, there must be a much more efficient way to do so and it shall be attempted in further labs..