LT2216/LT2816 V24 Dialogsystem
Lab II – Basic Dialogue Management

Lab report by Anni Nieminen

This assignment was a really nice introduction on how to implement finite state machines in slot-filling dialogue systems. It was also eye-opening in terms of how much planning and how many states go into designing only a simple machine like this.

I think the limitations of this finite state machine stem mainly from the limitations of the grammar, which obviously can make following the machine "happy path" quite difficult. One way of fixing this could have been creating a function that allows adding new keys and values in the grammar.

To take into account the possible variation in the user input, the grammar would need to be quite exhaustive. For instance, just to cover all possible 15min timeslots (let's sayt for 8am-4pm office hours), the grammar would need to hold 32 different keys :

```
"8": { time: "08:00" },

"8,15": { time: "08:15" },

"8,30": { time: "08:30" } . . .
```

Furthermore, we need to be careful with how these other timeslots would be interpreted by the machine. If we want the machine to say "eight fifteen" instead of "eight point fifteen", we need to use a comma! (I did some experiments with this but couldn't get the machine to map "eight fifteen" to "8,15" even though the pronunciations match?). I also tried writing the times with letters (`"eight fifteen": { time: "08:15" })`,but still it was not working.

To ensure a somewhat smooth user experience, the user would thus need to be informed of the precise word form with which they have to comply in order for the machine to work. For instance, what happens when the user says "at 8" or "8 o'clock" instead of stating the only number? What happens when the user says "quarter to nine" instead of "eight forty-five"?

I also noticed that the machine's way of pronouncing the times differs: `{ time: "08:00" }` gets pronounced as "8 o'clock" but `{ time: "16:00" }` is "sixteen hundred hours".

On top of all, the machine also is unable to interpret deictic words, which can also lead to user frustration. For example, the machine is unable to understand this type of user input:

Finite state machine	User input (deictic)	Result
"On which day is your meeting?"	"Tomorrow."	error
"Who would you like to meet?"	"My professor."	error

As stated, problems like this could be avoided by the machine being able to issue more clarification requests (and by being somehow synchronized with external databases?).

I also had some problems with the machine understanding my pronunciation of some of the keys in the grammar. I actually had to check the American English pronunciation of these keys from Google Translate! The machine was especially strict with "Rasmus" not being pronounced the Swedish way!

This finite state machine allows slot-filling only one-by-one, any over-answering by the user will lead to an error.