For the project, we are also required to let the users add their own custom skills to the digital assistant. These custom skills should be saved in a simple text file and users should be able to load them in from a given text file. A GUI could also later be added such that anyone should be able to add skills to a text file without knowing the inner workings of the way the skills work.

Fortunately, with the way our pattern language is set up, it should be relatively easy to create a simple dialogue with which this text file can be set up by converting the user's input to the pattern language that's been set up. It's also a tool which makes adding this to the existing code doesn't require any modifications to the existing code. In other words, it's a totally separate module of the application that can be easily connected or disconnected.

Customization of skills

- **Pattern:** Obviously, the custom skill should have a query which it responds to. This can be done by creating a pattern for this skill, according to the conventions described in the document about pattern languages. It should be possible to describe multiple patterns for a single skill, though for simplicity, it might be nice to start with just one pattern per skill.
- **Simple response:** For simple skills, where the given pattern only has one possible response (so only has String content with possibly blank content), this is as simple as supplying our assistant with a single string.
- **Parametric response:** In the case where the custom pattern has a parameter value, there may be different responses for different parameter values. In this case, the response should have special placeholders in the string and a set of tuples with the parametric value and the String/value that should be entered in the placeholder's place. Referring to a given parameter can be easily done with the index from the parameter in the list of slots.

Some examples of custom skills

Simple response
<What is the name of the bot?, What is this bot called?>;MultiModal-y;

The pattern followed by the simple response.

Single parametric response
<How many hours of lectures><...><on, at><@1, param:day>;There are @1 hours of lectures on this day.;[(Monday, 4),(Tuesday, 6),(Friday, 4),(default, 0)]

The pattern, followed by the response with placeholders, followed by a set of tuples containing the value for the parameter and the string that should be entered in the placeholder. The placeholder is the tag that's being used for the slot that's related to the parameter. To simplify this, an idea could be to use simple tags with a number, as to be able to search for more parameters simply by going through numbers.

Future Development

- **Skill Editor GUI:** Create a GUI for creating custom skills, where you can enter content into input fields and our code automatically converts those input fields to a text file, according to the language we've described above
- **Add domain:** The custom skill may fall under an existing domain, or a user may add several custom skills that belong to a single domain. In this case, it might be nice to group these skills together.
- **Multi-parametric responses:** Adding more parameters to a query
- **Custom parameters (optional, possibly only in later development):** A skill may need certain parameters to function properly. In case a user wants to add new parameters than the ones we've defined so far, the user may need to give the valid values for the parameter
- **Actions:** This should take the place of responses, but there may be custom actions from a set of predefined actions that a bot could perform given the custom skill. Not sure if this is worth adding, as I'd imagine we'd make domains ourselves for special actions.

Edited on 24th of February