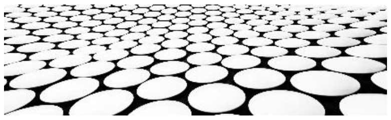
SEG3125 User Interface Design and Analysis



MODULE 10 – TUTORIAL/LAB

Exploring *Rasa* for conversational design

GOALS

The goal of Lab 10 is to get a bit of hands-on experience in building a chatbot that could be used as an alternative to a traditional (WIMP style) UI.

Although there are a few alternative platforms for building chatbots, many require a registration step including credit card information. The chosen one, Rasa is not super intuitive, but it does have an Open Source version for us to try without any registration process and it has good documentation. With Rasa, we will be able to build a very simple chatbot.

SUBMISSION DEADLINE

- Submission opens: Thursday July 15th, 12pm
- Submission closes: Sunday, July 25th, 11:30 pm
- There will not be any peer review for this lab.

SUBMISSION METHOD

In Peergrade, submit a report containing various screenshots showing the content of your conversation design. You can also submit a short video showing the unfolding of the conversation if you prefer. You will also need to submit the various files that you modified from the online example to meet the requirements (see coding section).

INSTRUCTIONS / TUTORIALS

Step 1 – Familiarize yourself with Rasa

A good way to get familiar with Rasa is to go to the Rasa Playground. In this playground, you can explore:

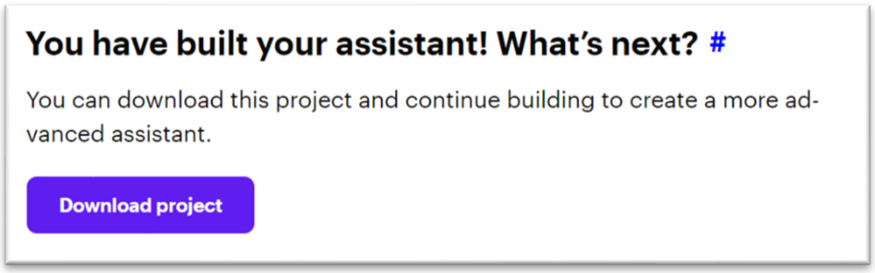
1. NLU data (intents + paraphrases provided for each intent)
2. Responses (the responses that the chatbot will give)
3. Stories (the conversation flow expected)

Defining stories is where it can become quite complex, and we will focus on linear conversation (no branching).

The playground also presents forms and rules, but I will **not** focus on those in this lab.

Make sure you train and test the small example built for subscribing to a newsletter, you can do that in the playground.

After training, the **Download project** button will be activated, and you can download this example (it will be a project.zip file). It will become our starting point for our lab (see Step 3) once you've installed Rasa locally (see Step 2).


You have built your assistant! What's next? #

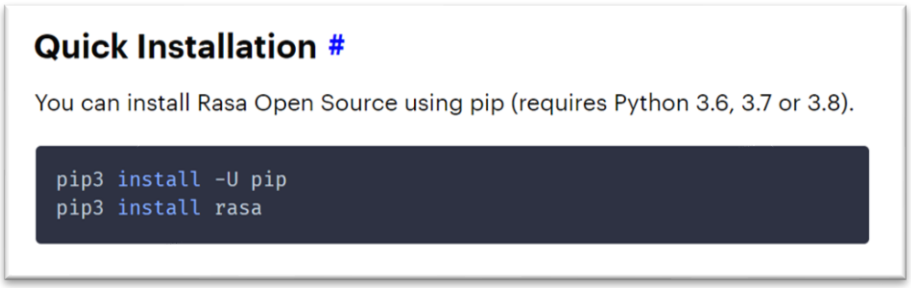
You can download this project and continue building to create a more advanced assistant.

Step 2 – Install the open-source version of Rasa

Although the playground would allow you to modify the example and expand on it, it is not practical to do so, as it takes a very long time to train online, and whatever you do will be lost as soon as you refresh the page.

A better alternative is to install the open source version or RASA and work locally with the playground example.

If you already have Python 3.6 (or latest version) installed, installing RASA is easy, as shown below:



Quick Installation #

You can install Rasa Open Source using pip (requires Python 3.6, 3.7 or 3.8).

```
pip3 install -U pip
pip3 install rasa
```

If not, follow the Step-by-Step installation guide, shown on the Rasa Installation page.

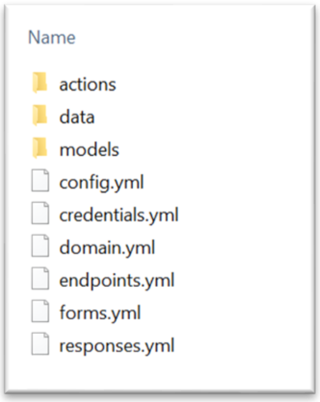
In this open-source version, you will use the Command Line Interface to perform different Rasa commands. Two commands are very important:

- rasa train – to train the machine learning model based on the intents, responses and stories.
- rasa shell – to test your chatbot.

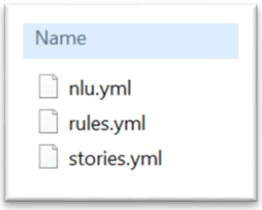
For those commands to work, the directory set up of a Rasa project needs to conform to Rasa's expectations. Let see what that is in Step 3.

Step 3 – Test the example provided.

First unzip your project.zip file (from Step 1). The structure you obtain is shown to the right. And if you open the data folder, you will further see the 3 files below.



```
Name
actions
data
models
config.yml
credentials.yml
domain.yml
endpoints.yml
forms.yml
responses.yml
```



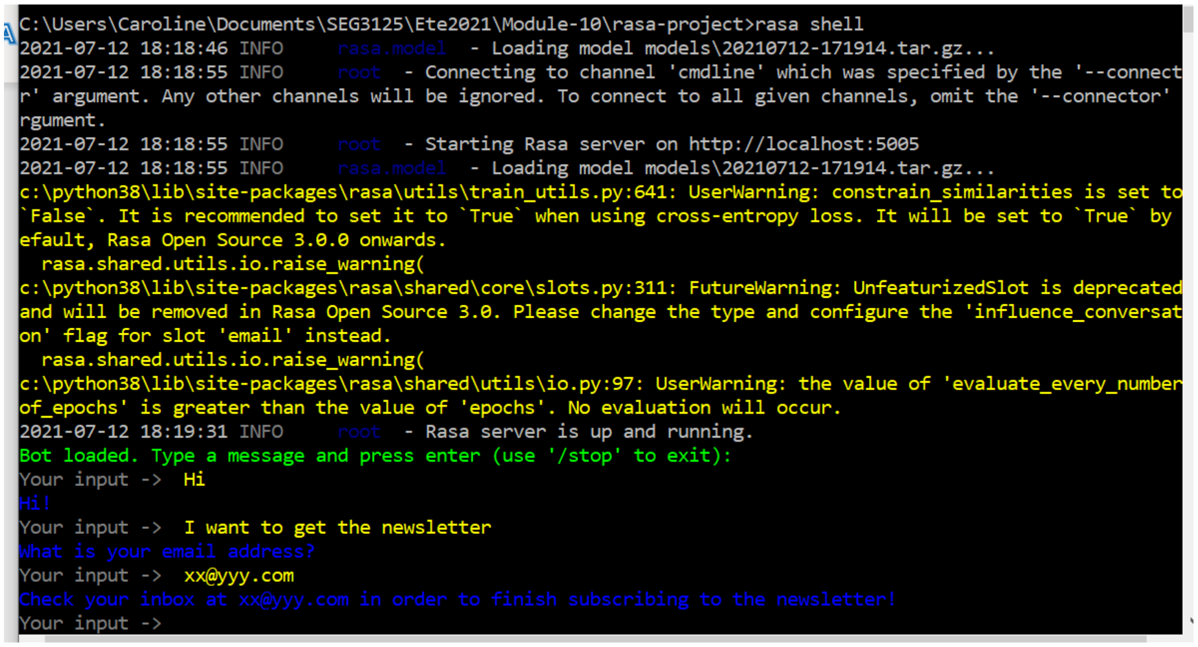
```
Name
nlu.yml
rules.yml
stories.yml
```

You can open the .yml file in Notepad++ or any editor you want.

Before we do any changes to these files, you can train and test this simple chatbot. At the command line, you can type rasa train (to train, which takes a bit of time) and then rasa shell to test. If you are not in the same directory where the project files are, you will have to specify the path toward that directory. You can see some options if you type rasa -h (for help).

Just to show you below, I unzipped project.zip in my repertory …/Module-10/rasa-project. I did *rasa train* first. Below I'm showing *rasa shell*. I will ignore the warnings (some version incompatibility we will not deal with) and try the small conversation. My input is easy to see (in yellow) but the bot response is impossible to read (blue on black!).

This is our starting point. You will now be asked to modify some of the files (nlu.yml, stories.yml, responses.yml, domain.yml).



```
C:\Users\Caroline\Documents\SEG3125\Ete2021\Module-10\rasa-project>rasa shell
2021-07-12 18:18:46 INFO     rasa.model  - Loading model models\20210712-171914.tar.gz...
2021-07-12 18:18:55 INFO     root  - Connecting to channel 'cmdline' which was specified by the '--connect
r' argument. Any other channels will be ignored. To connect to all given channels, omit the '--connector'
rgument.
2021-07-12 18:18:55 INFO     root  - Starting Rasa server on http://localhost:5005
2021-07-12 18:18:55 INFO     rasa.model  - Loading model models\20210712-171914.tar.gz...
c:\python38\lib\site-packages\rasa\utils\train_utils.py:641: UserWarning: constrain_similarities is set to
`False`. It is recommended to set it to `True` when using cross-entropy loss. It will be set to `True` by
efault, Rasa Open Source 3.0.0 onwards.
  rasa.shared.utils.io.raise_warning(
c:\python38\lib\site-packages\rasa\shared\core\slots.py:311: FutureWarning: UnfeaturizedSlot is deprecated
and will be removed in Rasa Open Source 3.0. Please change the type and configure the 'influence_conversat
on' flag for slot 'email' instead.
  rasa.shared.utils.io.raise_warning(
c:\python38\lib\site-packages\rasa\shared\utils\io.py:97: UserWarning: the value of 'evaluate_every_number
of_epochs' is greater than the value of 'epochs'. No evaluation will occur.
2021-07-12 18:19:31 INFO     root  - Rasa server is up and running.
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  Hi
Hi!
Your input ->  I want to get the newsletter
What is your email address?
Your input ->  xx@yyy.com
Check your inbox at xx@yyy.com in order to finish subscribing to the newsletter!
Your input ->
```

DESIGN

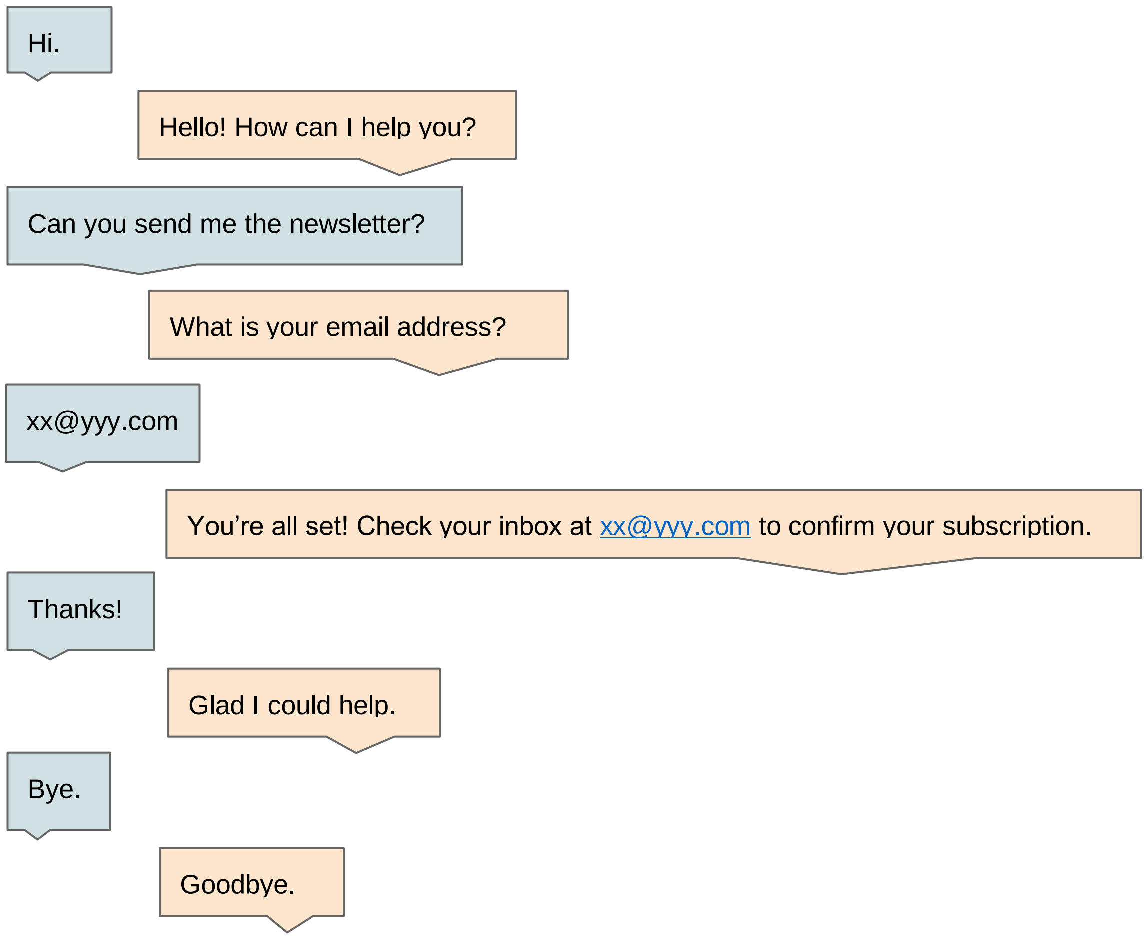
In general, when we build a chatbot, much effort goes toward "Conversational Design", meaning how we want the conversation to unfold.

But in this laboratory 10, we will not go very far in that design. You will develop 2 stories in relation to the newsletter.

- One story to subscribe to the newsletter.
- One story to unsubscribe to the newsletter.

Both stories will follow a linear design (no branching).

For example, the subscribing stories would unfold as follow:



Hi.

Hello! How can I help you?

Can you send me the newsletter?

What is your email address?

xx@yyy.com

You're all set! Check your inbox at xx@yyy.com to confirm your subscription.

Thanks!

Glad I could help.

Bye.

Goodbye.

STARTING POINT

Your starting point is the example from the Rasa Playground (see instructions).

CODING

There is no coding in this lab, but rather the "programming" of your chatbot in terms of:

- intents to be recognized and for which you provide training sentences (paraphrases for the same intent),
- responses to be provided by the chatbot,
- stories that can unfold.

<u>Here are the requirements:</u>

1. <u>Expand the "greet and subscribe" story</u>. First, the variations to say "hello" are limited. Also, it is not possible in the current story to end the conversation with thank you and good bye (as shown in the examples in the Design section).
    a. Add at least 4 training sentences for the greet intent already defined.
    b. Define a new Thank You intent for which you will define at least 4 training examples, as well as 4 possible responses from the chatbot (e.g. "Thanks", "Glad I could help").
    c. Define a new Goodbye intent for which you will define at least 4 training examples, as well as 4 possible responses from the chatbot (e.g. "Bye", "Goodbye").
    d. Modify the story to include the Thank you and Goodbye intents at the end (see example in Design section).
2. <u>Add a story, called "greet and unsubscribe"</u> in which a user will ask the chatbot to unsubscribe and then the chatbot will ask to explain why, the user will provide some reason (not interested anymore, not time, etc), and the chatbot will answer with empathy (ok, we understand / no problem, etc). Provide at least 4 training examples for each intent, and 4 alternatives for each response of the chatbot.
3. (optional) You can play and add anything else you want in this chatbot.

Attention: To achieve the requirements, you will need to modify the files nly.yml, stories.yml, responses.yml as well as domain.yml (which is pretty much a copy of parts of the other files). I do not fully understand why we need the domain.yml, as it seems very redundant to me, but your system won't train without it.

☆☆☆

EVALUATION

- Lab 10 is worth 3.5%.
- Since it's the last lab of the semester and you are entering in your exam period, a TA will do the evaluation instead of a peer review.
- The TA will read your report and fill out the rubrics shown in peergrade.
- Your report should contain:
  - Screen shots of a test conversation with your chatbot (or a short video).
  - Files nly.yml, stories.yml, responses.yml which you modified.
- Any delay beyond the deadline will have a penalty of 10% per day.

QUESTIONS

- You can ask your questions in the Module 10 discussion forum on Brightspace.
- You can also send your questions directly to the TA you are assigned to. If your last name starts with a letter between A and J - please send mail to Abdorrahim (abahr010@uottawa.ca), otherwise send an email to Xinyi (xhe068@uottawa.ca).