Week 2 – Domain selection

Wednesday, October 20, 2021 11:23 AM

You will be building a dialogue system in (at least some of) the homeworks for this course. Your first task is to **choose a domain** and imagine how your system will look like and work like. Since you might later find that you don't like the domain, you are now required to **pick two**, so you have more/better ideas later and can choose only one of them for building the system.

Requirements

The required steps for this homework are:

1. Pick **two** domains of your liking that are suitable for building a dialogue system.
2. Write 5 example system-user dialogues for both domains, which are at least 5+5 turns long (5 sentences for both user and system). This will make sure that your domain is interesting enough. You might use Greek or English.
3. Create a flowchart for your two domains, with labels such as "ask about phone number", "reply with phone number", "something else" etc. It should cover all of your example dialogues. Feel free to draw this by hand and take a photo, as long as it's legible.
    - It's OK if your example dialogues don't go all in a straight line (e.g. some of them might loop or go back to the start).
4. In your Github account, create DialogSystemsCourse repository. Inside this repository, create a directory called `hw1/` and save both the example dialogues and the flowchart into this directory. Make sure to invite both Nassos and Antigoni.

Inspiration

You may choose any domain you like, be it tourist information, information about culture events/traffic, news, scheduling/agenda, task completion etc. You can take inspiration from stuff presented in the first lecture, or you may choose your own topic.

Week 3 - Dataset exploration

20 October 2021 15:18

The task in this lab is to explore dialogue datasets and find out more about them. Your job will thus be to write a script that computes some basic statistics about datasets, and then try to interpret the script's results.

1. We provide the dialog dataset in .txt format (in tab "files", named "dialog-babi-task5-full-dialogs-trn.txt"). This file contains data from testing end-to-end dialog systems in the restaurant domain, described in the following paper:

   Antoine Bordes, Y-Lan Boureau, Jason Weston, Learning End-to-End Goal-Oriented Dialog, arxiv:1605.07683.

   The file format for each task is as follows:

   ID user_utterance [tab] bot_utterance
   ...

   The IDs for a given dialog start at 1 and increase. When the IDs in a file reset back to 1 you can consider the following sentences as a new dialog. When the bot speaks two times in a row, we used the special token "<SILENCE>" to fill in for the missing user utterance. See more details in the README included with the dataset. Bot utterances can be sentences or API calls (sentences starting with the special token "api_call"). Here is an example of dialog:

1 hi	hello what can i help you with today
2 can you make a restaurant reservation with italian cuisine for six people in a cheap price range	i'm on it
3 <SILENCE>	where should it be
4 rome please	ok let me look into some options for you
5 <SILENCE>	api_call italian rome six cheap

   Green stands for dialog turns, yellow for user utterances and blue for bot utterances. User and bot utterances are separated by a **tab,** while all other words by space.

2. Write a script that will **read all turns in the data and separate the user and system utterances** in the training set.
   - Make the script **ignore any search results lines** in the data (they **don't** contain a tab character). For example:

     1 good morning hello   what can i help you with today

     2 i'd like to book a table with italian food   i'm on it

     3 <SILENCE>   where should it be

     4 in paris   how many people would be in your party

     5 for six people please   which price range are looking for

     6 in a cheap price range please   ok let me look into some options for you

     7 <SILENCE>   api_call italian paris six cheap

     8 actually i would prefer for two people   sure is there anything else to update

```
9 instead could it be in madrid	sure is there anything else to update
10 instead could it be with spanish food	sure is there anything else to update
11 no	ok let me look into some options for you
12 <SILENCE>	api_call spanish madrid two cheap
13 resto_madrid_cheap_spanish_1stars R_phone resto_madrid_cheap_spanish_1stars_phone
14 resto_madrid_cheap_spanish_1stars R_cuisine spanish
15 resto_madrid_cheap_spanish_1stars R_address resto_madrid_cheap_spanish_1stars_address
16 resto_madrid_cheap_spanish_1stars R_location madrid
17 resto_madrid_cheap_spanish_1stars R_number two
18 resto_madrid_cheap_spanish_1stars R_price cheap
19 resto_madrid_cheap_spanish_1stars R_rating 1
20 resto_madrid_cheap_spanish_6stars R_phone resto_madrid_cheap_spanish_6stars_phone
21 resto_madrid_cheap_spanish_6stars R_cuisine spanish
22 resto_madrid_cheap_spanish_6stars R_address resto_madrid_cheap_spanish_6stars_address
23 resto_madrid_cheap_spanish_6stars R_location madrid
24 resto_madrid_cheap_spanish_6stars R_number two
25 resto_madrid_cheap_spanish_6stars R_price cheap
26 resto_madrid_cheap_spanish_6stars R_rating 6
27 resto_madrid_cheap_spanish_8stars R_phone resto_madrid_cheap_spanish_8stars_phone
28 resto_madrid_cheap_spanish_8stars R_cuisine spanish
29 resto_madrid_cheap_spanish_8stars R_address resto_madrid_cheap_spanish_8stars_address
30 resto_madrid_cheap_spanish_8stars R_location madrid
31 resto_madrid_cheap_spanish_8stars R_number two
32 resto_madrid_cheap_spanish_8stars R_price cheap
33 resto_madrid_cheap_spanish_8stars R_rating 8
34 <SILENCE>	what do you think of this option: resto_madrid_cheap_spanish_8stars
35 no this does not work for me	sure let me find an other option for you
36 <SILENCE>	what do you think of this option: resto_madrid_cheap_spanish_6stars
37 do you have something else	sure let me find an other option for you
38 <SILENCE>	what do you think of this option: resto_madrid_cheap_spanish_1stars
39 it's perfect	great let me do the reservation
40 may i have the phone number of the restaurant	here it is resto_madrid_cheap_spanish_1stars_phone
41 thanks	is there anything i can help you with
42 no thank you	you're welcome
```

You should ignore the grey lines (TIP: search if the line contains a tab. If not, then ignore.)

- If the script finds a turn where the user is silent (the user turn contains only `<SILENCE>`), it should concatenate the system response from this turn to the previous turn. Note that this may happen on multiple consecutive turns, and the script should join all of these together into one system response. For example, for lines 35 and 36, user utterance is: *no this does not work for me* and system response is: *sure let me find an other option for you what do you think of this option: resto_madrid_cheap_spanish_6stars* (concatenated system responses)

- If `<SILENCE>` is the first word in the dialogue, just delete it.

- Don't worry too much about tokenization (word segmentation) -- tokenizing on whitespace is OK.

1. Implement a routine that will **compute the following statistics** for system and user turns (separately):
    - data length (total number of dialogues, turns, words)
    - mean and standard deviations for individual dialogue lengths (number of turns in a dialogue, number of words in a turn)
    - vocabulary size

   Commit this file as `hw2/stats.py`.

Week 4 – Rasa installation

20 October 2021 15:19

Install Rasa X locally in your computer following instructions from here:

https://rasa.com/docs/rasa-x/installation-and-setup/install/local-mode (not the "Share your bot in local mode" section)

(if nothing happens when you hit "rasa x" command, type "rasa init" in your folder and then type again "rasa x")

Open your browser and type: http://localhost:5002

We would like you to create one or more stories and experiment with NLU intents and paraphrases, then train and test your model. Do not forget to update the domain, when you add new stories, responses and intents!

Finally, share your bot description, intents, stories and responses and a few examples of your interaction with your bot, as well as any thoughts, notes you have, improvements you could make, etc., in a report. Upload your report to your github account as `hw3/rasax_report.py`.

Week 5 - NLU in rasa, training and finetuning

20 October 2021 15:19

Now that you have some experience with Rasa X, it is time to play around with NLU intents and training. Choose your domain, create some stories and enrich your NLU intents and paraphrases.
Then train and test your model. Interact with your bot naturally, and evaluate its responses. Then go back to your NLU data, add new paraphrases and train again.

Finally, share your bot description, intents, stories and responses and a few examples of your interaction with your bot, as well as any thoughts, notes you have, improvements you could make, etc., in a report. Upload your report to your github account as `hw4/rasax_nlu.py`.

Week 6 – Installing Rasa - Adding slots and entities

20 October 2021 15:19

This week you will install Rasa locally, on your computer following instructions from:
https://rasa.com/docs/rasa/installation
Additionally, for installation with Dockerfile, you can refer to Rasa github repository:
https://github.com/RasaHQ/rasa

Rasa is a command line tool with 3 basic commands:
- `rasa init` to initialize an empty working rasa directory (creates all necessary files as a basis to work on your project)

- `rasa train` to train your model

- `rasa shell` to talk to your bot and test it

After transferring your project from RasaX to Rasa (nlu.yml, stories.yml, domain.yml need alterations according to your project), the next steps involve creating entities and slots for the parameters given by the user.
Entities are user input extracted from NLU data, e.g. user's name, date, location, etc.
Slots are the bot's memory and can include information needed during the dialog (e.g. the user's name, any search parameter, etc.).
Often slots are filled through entities, but there are also other ways to create and fill slots, that we will discover later on. Entities and slots have to be declared in your domain.yml file. If your slots have the same name as your entities, they get automatically filled by the corresponding entities.

Example of entity inside nlu.yml:

```
- intent: request_restaurant
  examples: |
        - can i get [swedish](cuisine) food in any area
        - a restaurant that serves [caribbean](cuisine) food
        - ..........
```

If your domain contains entities with a finite number of different values you can also add a lookup table in your nlu.yml file. You can read more here: https://rasa.com/docs/rasa/nlu-training-data#lookup-tables

Read more here: https://towardsdatascience.com/building-a-chatbot-with-rasa-3f03ecc5b324
Or check Rasa example Formbot (we are not going to use the form yet, but the example contains information on how to declare entities and slots in your nlu.yml, domain.yml):
https://github.com/RasaHQ/rasa/blob/main/examples/formbot/domain.yml

To sum up, you need to decide and fix the domain that you will be working from now on, taking into account that it needs to involve at least one interactive feature, e.g. the bot will need to parse a web page to provide the user with the needed information (e.g. timetables, recipes, etc.).
After finalizing the domain and creating the initial nlu and stories, you need to add a few entities and slots.
Train and test your bot.

Homework deliverables:
-Share your final bot description, along with the interactive feature you are proposing and commit it as `hw5/bot_description.txt`
-As in previous homeworks, commit your working directory rasa files containing the newly added entities and slots into your github repository under the name of your bot.
-Share a few interactions (containing entities and slots) with your bot as text file `hw5/sample_interactions.txt`

Weeks 7-8 - Adding actions

20 October 2021 15:20

Now that you have added slots, it is time to add some actions to the dialog system.
Actions are custom-made code, that can add interest and real interaction with a bot.
For example, if someone asks a bot for tomorrow's weather, a custom action could mean that the bot, after recognizing the correct intent, initiates an action which connects to an API or webpage, gets real-time data and then presents it to the user in a friendly way. Through actions you can also set slot values that might be needed later in the dialog. Also, the responses that the bot utters to the user, is a very simplified form of action.

Actions in Rasa are written in Python in a file called actions.py and are also declared in the domain file and inserted in the correct positions in stories. In https://github.com/RasaHQ/rasa/blob/main/examples/reminderbot there is a sample actions file to serve as a guideline for your own actions.

According to your scenarios, your task is to think and implement a custom action for your bot, in the form of web scraping. You could either use an API or a webpage content. A nice and useful tutorial on web scraping can be read here: https://realpython.com/beautiful-soup-web-scraper-python/

Read more here: https://rasa.com/docs/action-server/sdk-actions

In order to run your action server, after writing your actions, you need to run the command:
-> rasa run actions

Then, you can try your bot the same way as before: -> rasa shell

As usual, commit your changes to your github account, in your bot's name folder.

Weeks 9-10 – Evaluation

20 October 2021 15:20

For this assignment you will need to have your chatbot evaluated by a group of 5 people. For this purpose, you will use RasaX. The users will need to connect with the system and try to interact with it to complete a specific task. You will need to describe the task to them beforehand. Try to provide a description that would allow them to understand what goal they will need to achieve with the interaction but it would be nice to avoid giving them step-by-step instructions. In this way, you will capture more realistic data on how users may end up interacting with your system.
The goal is to get feedback from real users regarding the usability of the chatbot. The evaluation will be set up as a user study.

You will need to deliver the following:
1. The email you sent to the users to invite them to the study.
2. Any instructions you will give to them beforehand, e.g., what kind of task they will need to fulfill.
3. A questionnaire that you will have provided to the users in order to assess to which extend they found the system useful, if they actually managed to perform the task, etc.
4. An analysis of the results and suggested actions for the future.

Final project and report

20 October 2021 15:20

As your final project you will need to finalize the dialogue system you have been working on since the beginning of the class. It will need to support at least three different stories/scenarios, use slots, and be collecting information from the web as needed. You will need to experiment with different dialogue policies and run a systematic evaluation. Please find below the list of deliverables:

1. An up-to-date github repository with the source code of your dialogue system.
2. A technical report including the following:
   a. System functional specifications, dialogue system description:
      i. What can your dialogue system do? What kind of tasks does it support?
   b. System architecture, description of components:
      i. General architecture
      ii. Description of the various components and what they do
         1. Intent recognition
         2. Feature extraction
         3. Dialogue policy
            a. Try various dialogue policies and comment on your observations
            b. Which one seems to be working better?
         4. DIET architecture (how is it configured in your case)
   c. Evaluation
      i. Describe the evaluation process
         1. Provide link to the questionnaire you have used
      ii. 1st phase evaluation (5 users)
      iii. 2nd phase evaluation (10 users)
      iv. How was the system different for the 2nd phase evaluation?
   d. Conclusions and future work
      i. Challenges that you faced
      ii. Next steps to make the dialogue system more reliable and useful