NLU Tutorial

José Lopes

jd.lopes@hw.ac.uk

Adapted from Emanuele Bastianelli's slides

Before we start

- Clone

```
git clone
https://github.com/HWUConvAgentsProject/CA2020
_instructions.git
```

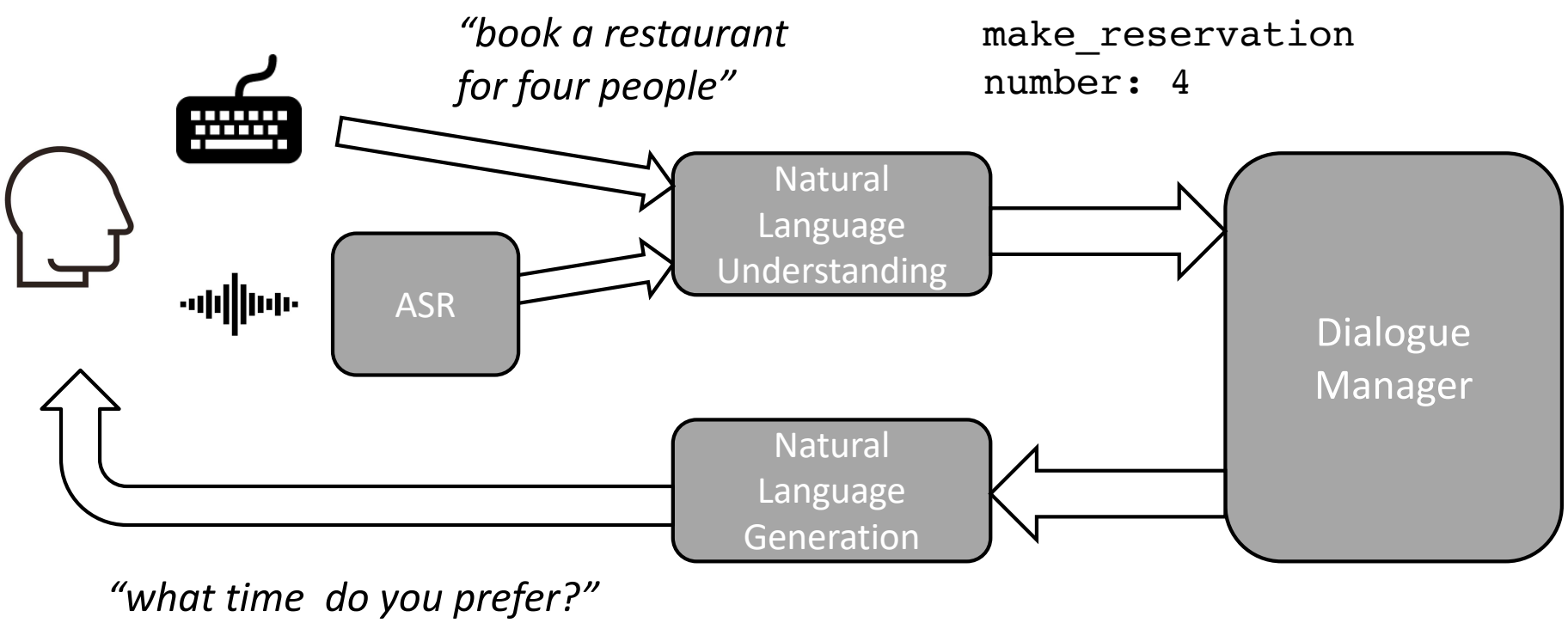
- Update

```
git pull
```

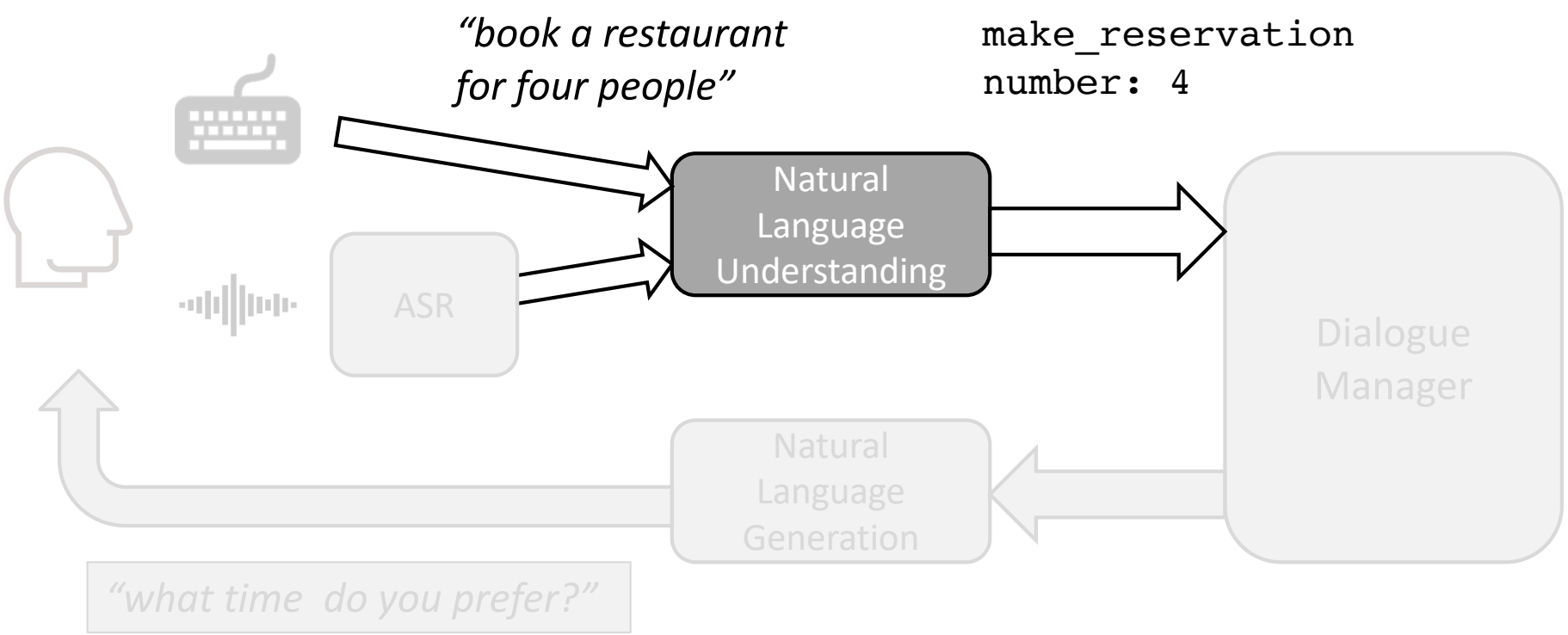
What we are going to do today

- Create RASA Project
- Understanding RASA input and output formats
- Building and training RASA NLU pipelines
- Testing RASA NLU pipelines

Dialogue Systems

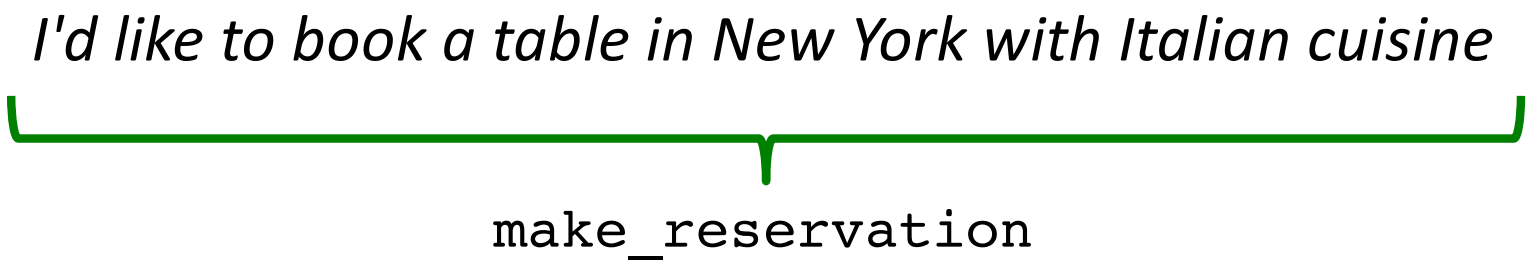


Dialogue Systems

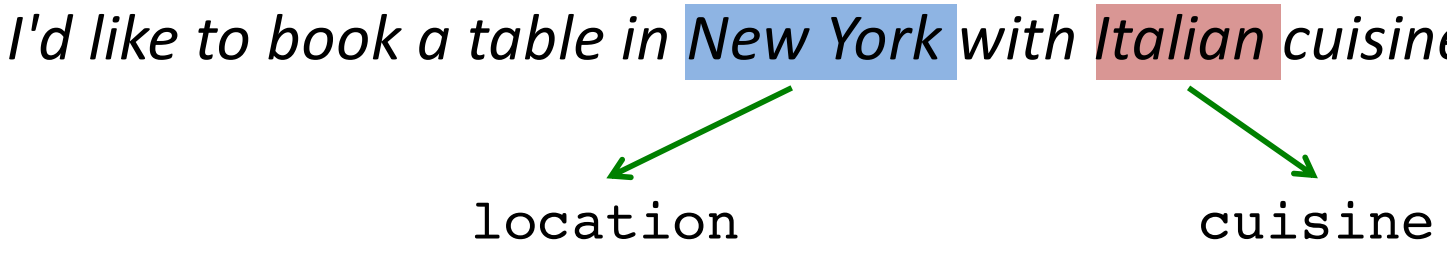


Recap: intent and slots

- What's NLU
  - What's an **intent**



  - What's an **entity** (or slots, or arguments…)



Before we start

- Let’s test if everything is ok
  - `rasa`
- if not ...

Before we start

- Let’s install RASA
  1. `conda activate Alana`
  2. `pip install --no-cache-dir rasa`
  3. `rasa`

Create a RASA project

- Create a RASA project
  1. `mkdir rasa-nlu-tutorial`
  2. `cd rasa-nlu-tutorial`
  3. `rasa init --no-prompt`
  4. `rasa shell`

Create a RASA project

- Directory structure:
  - `actions/`
  - `data/`
  - `models/`
  - `tests/`
  - `config.yml`
  - `credentials.yml`
  - `domain.yml`
  - `endpoints.yml`

Create a RASA project

- Directory structure:
  - `actions/`
  - `data/`
  - `models/`
  - `tests/`
  - `config.yml`
  - `credentials.yml`
  - `domain.yml`
  - `endpoints.yml`

Create a RASA project

- Directory structure:
  - `actions/`
  - `data/`
    - `nlu.yml`
    - `stories.yml`
    - `rules.yml`
  - `models/`
  - `tests/`
  - `config.yml`
  - `credentials.yml`
  - `domain.yml`
  - `endpoints.yml`

Create a RASA project

- Directory structure:
  - actions/
  - data/
    - nlu.yml
    - stories.yml
    - rules.yaml
  - models/
  - tests/
  - config.yml
  - credentials.yml
  - domain.yml
  - endpoints.yml

The bAbI dataset

- https://research.fb.com/downloads/babi/
- Using the (6) dialog bAbI tasks
  - Dialogues about restaurant booking
- We're using the dialog-babi-task5-full-dialogs-trn.txt file for this tutorial, but
  - `CA2020_instructions/rasa_tutorial/nlu/babi_nlu.yml`

The bAbI dataset

- Example dialogue:

U: good morning
A: hello what can i help you with today
U: i'd like to book a table with french food in paris
A: i'm on it. how many people would be in your party?
U: six people please
A: which price range are looking for?
U: a cheap price range please
A: ok let me look into some options for you
A: what do you think of this option: Chez Gladine?
U: it's perfect
A: great let me do the reservation
U: thanks

The bAbI dataset

- Example dialogue:

U: good morning
A: hello what can i help you with today
U: i'd like to book a table with french food in paris
A: i'm on it. how many people would be in your party?
U: six people please
A: which price range are looking for?
U: a cheap price range please
A: ok let me look into some options for you
A: what do you think of this option: Chez Gladine?
U: it's perfect
A: great let me do the reservation
U: thanks

The bAbI dataset

- Example dialogue:

U: good morning → `greet`
A: hello what can i help you with today
U: i'd like to book a table with french food in paris → `make_reservation`
A: i'm on it. how many people would be in your party?
U: six people please → `inform`
A: which price range are looking for?
U: a cheap price range please → `inform`
A: ok let me look into some options for you
A: what do you think of this option: Chez Gladine?
U: it's perfect → `affirm`
A: great let me do the reservation
U: thanks → `thanking`

The bAbI dataset

- Example dialogue:

U: good morning → `greet`
A: hello what can i help you with today
U: i'd like to book a table with french food in paris → `make_reservation`
A: i'm on it. how many people would be in your party?
U: six people please → `inform`
A: which price range are looking for?
U: a cheap price range please → `inform`
A: ok let me look into some options for you
A: what do you think of this option: Chez Gladine?
U: it's perfect → `affirm`
A: great let me do the reservation
U: thanks → `thanking`

bAbI intents

- Recap on the dataset: the bAbI
  - Intent defined for the dataset
    - `greet`: *hello, hi, good morning, …*
    - `affirm`: *yes, of course, right, …*
    - `deny`: *no, I don't like it, …*
    - `make_reservation`: *can I book a table for six people…*
    - `inform`: *a cheap one, my number is 555, I like indian cuisine, …*
    - `repair_inform`: *actually I prefer spanish cuisine, …*
    - `get_info`: *can I have the address of the restaurant, …*
    - `thanking`: *thanks, many thanks, …*

bAbI entities

- Recap on the dataset: the bAbI
  - Entities defined for the dataset
    - `location`: *in **paris**, in **new york**, …*
    - `cuisine`: *an **indian** restaurant, with **spanish** cuisine, …*
    - `number`: *for **six** people, a table for **two**, …*
    - `price_range`: *a **cheap** restaurant, an **expensive** one, …*
    - `info`: *can I have the **address**, what's restaurant **number** …*
    - `phone_number`: *my phone number is **555-1234**, …*

RASA NLU Input format

- `CA2020_instructions/rasa_tutorial/nlu/nlu.yml`
  - to be placed in a `nlu.yml` under `data/`
- Four sections:
  - Common examples
  - Synonyms
  - Regex features
  - Lookup tables

YAML: examples

- Common examples syntax

```
- intent: intent1
  examples: |
    - word1 [word3](entity1) word4 word5 [word6 word7](entity2)
…

- intent: intent2
…
```

- Example

```
- intent: greet
  examples: |
    - hi

- intent: make_reservation
    examples: |
    - i want [spanish](cuisine) cuisine in [New York](location)
```

YAML: examples

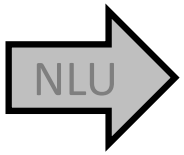
- ASSIGNMENT 1: train rasa nlu
  - `rasa train nlu`

- ASSIGNMENT 2: launch rasa nlu shell
  - `rasa shell nlu`
  - parse "*can you book a restaurant in new york*"

RASA NLU output format

- Json output format

can you book a restaurant in new york




```
{
    “text”: “can you book a restaurant in new york”,
    “intent”: {
        "id": -5795578177912592 31,
        "name": "make_reservation",
        "confidence": 0.991264641284942 6
    },
    “entities”: [
      {
        “entity”: “location”,
        “start”: 29,
        “end”: 37,
        “confidence”: 0.9759525656700134,
        “value”: “new york”,
        “extractor": "DIETClassifier“
      }
    ],
    “intent_ranking”: [...]
}
```

YAML: synonyms (1/2)

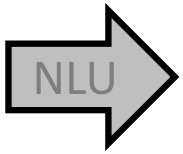
- Synonyms syntax

```
- synonym: referred_entity_filer
  examples: |
   - word1
   - word3 word4
```

- Example

```
- synonym: new york
- examples: |
   - NYC
   - New York city
```

*can you book a restaurant in **NYC***




```
{
   “text”: “can you book a restaurant in NYC”,
   “intent”: “make_reservation”,
   “entities”:[
       {
          “entity”: “location”,
          “value”: “new york”,
          ...,
       }
}
```

YAML: synonyms (2/2)

- Synonyms syntax

```
- synonym: referred_entity_filer
  examples: |
    - word1
    - word3 word4
```

- Example

```
- synonym: new york
- examples: |
    - NYC
    - New York city
```

DISCLAIMER: defining synonyms this way does not automatically add examples to your dataset. You still need to add examples with the synonyms to have them correctly identifies.

Ex: `i'd like to book a restaurant in [NYC](location)`

YAML: synonyms

- ASSIGNMENT 3: try using synonyms
  1. parse "*book a restaurant in NYC*"
  2. add synonyms to the `nlu.yml` file

```
- intent:make_reservation
   examples: |
    - i'd like to book a restaurant in [NYC](location)
    - can you book a restaurant in [NYC](location)
    - i'd like to book a table in [new york city](location)
- synonym: new york
- examples: |
  - new york city
```

  1. re-train rasa nlu: `rasa train nlu`
  2. parse again "*book a restaurant in NYC*"
  3. parse "*book a restaurant in new york city*"

YAML: lookup tables

- Lookup table syntax

```
- lookup:entity_type
  examples: |
   - word1 word2
   - …
```

- Example

```
- lookup:infos
  examples: |
   - phone number
   - address
```

DISCLAIMER: this does not automatically add examples to your dataset. It only defines a regex for each line, which matches exactly the related string.

YAML: lookup tables

- ASSIGNMENT 5: try using lookup tables
  1. parse "*can i have the directions*"
  2. add lookup to your `nlu.yaml` file

```
- lookup: info
  examples: |
    - phone number
    - number
    - phone
    - address
    - location
    - street
    - directions
```

  3. re-train rasa nlu: `rasa train nlu`
  4. parse again "*can i have the directions*"

Training RASA - Pipelines

- https://rasa.com/docs/rasa/nlu/choosing-a-pipeline/

- Configuration of a nlu pipeline
  - `config.yml`

Training RASA - Pipeines

- default pipeline
  - https://rasa.com/docs/rasa/nlu/components/

```
language: en

pipeline:
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  epochs: 100
- name: EntitySynonymMapper
- name: ResponseSelector
  epochs: 100
- name: FallbackClassifier
  threshold: 0.3
  ambiguity_threshold: 0.1
```

Training RASA

- Training rasa via command line
  - `rasa train nlu`

- Training rasa via Python API
  - script in `CA2020_instructions/rasa_tutorial/nlu/train_nlu.py`

Testing RASA

- Testing via command line
  - `rasa shell nlu`
- Using RASA http API
  1. `rasa run --enable-api -m models/[model_name]`
  2. `curl localhost:5005/model/parse -d '{"text":"can i book a table in madrid"}'`
- Testing via Python API
  - script in

    `CA2020_instructions/rasa_tutorial/nlu/test_nlu.py`

Useful links

- Some useful links
  - https://rasa.com/docs/
  - https://rasa.com/docs/rasa/user-guide/rasa-tutorial/
  - https://rasa.com/docs/rasa/nlu/about/