PyDial - Tutorial

Multi-domain Statistical Dialogue System Toolkit

Developed by the Dialogue Systems Group at the University of Cambridge

Thang Vu, Chia-Yu Li, Glorianna Jagfeld, Daniel Ortega

Today's Goals

- Download and play with Pydial
- Add a new domain to PyDial
- Create a rule-based NLU module
- Create a template-based NLG module
- Run a multi-domain dialog system

Documentation

- Paper: http://www.aclweb.org/anthology/P17-4013
- Webpage: http://www.camdial.org/pydial/

Installation

Downloading the code:

- Go to your working directory using the Terminal

```
> cd path_to_your_working_directory
```

- Download the code

```
> git clone https://bitbucket.org/dialoguesystems/pydial.git
```

- Enter the pydial folder

```
> cd pydial
```

- Install requirements

```
> pip install -r requirements.txt --user
```

Testing PyDial

Testing the installation:

- Test pydial

```
> sh testPyDial
```

- Expected Output:

```
Running PyDial Tests
  1 tests/test_DialogueServer.py   time 0m1,529s
  2 tests/test_Simulate.py         time 0m0,363s
  3 tests/test_Tasks.py            time 0m1,755s
3 tests:  16 warnings,   0 errors
See test logs in _testlogs for details
```

- Run a chat conversation:

```
> python pydial.py chat config/Tut-hdc-CamInfo.cfg
```

PyDial Log Files

Folder: path_to_your_pydial_directory/_log

```
 FlatOntologyManager.py <_set_ontology>126 :  Loading ontology: ontology/ontologies/CamRestaurants-rules.json
       FlatOntologyManager.py <_set_db>141 :  Loading database: ontology/ontologies/CamRestaurants-dbase.db
 FlatOntologyManager.py <_set_ontology>126 :  Loading ontology: ontology/ontologies/CamHotels-rules.json
       FlatOntologyManager.py <_set_db>141 :  Loading database: ontology/ontologies/CamHotels-dbase.db
                  pydial <chat_command>729 :  *** Chatting with policies ['CamRestaurants', 'CamHotels']: ***
                Agent.py <_hand_control>430 :  Launching Dialogue Manager for domain: topicmanager
                   Agent.py <start_call>178 :  >> NEW DIALOGUE SESSION. Number: 1
                Agent.py <_hand_control>447 :  Domain topicmanager is both already running and has control
                  Agent.py <_print_turn>554 :  ** Turn 0 **
               Agent.py <_print_sys_act>569 :  | Sys > hello()
                 Agent.py <_agents_semo>653 :  Domain with CONTROL: topicmanager
                  Agent.py <_print_turn>554 :  ** Turn 1 **
    RuleTopicTrackers.py <infer_domain>151 :  CamHotels keyword found in: I want a cheap hotel in the centre of tow
n.
         TopicTracking.py <track_topic>136 :  TopicTracker switched domains. From topicmanager to CamHotels
         TopicTracking.py <track_topic>142 :  After user_act - domain is now: CamHotels
                Agent.py <_hand_control>430 :  Launching Dialogue Manager for domain: CamHotels
                        SemI.py <decode>195 :  [(inform(area=centre)|inform(kind=hotel)|inform(pricerange=cheap)|
                                               inform(type=placetostay)', 1.0)]
     SemI.py <_add_context_to_user_act>254 :  Possibly adding context to user semi hyps: [(inform(area=centre)|info
rm(
                                               kind=hotel)|inform(pricerange=cheap)|inform(type=placetostay)', 1.0)]
ModSemBeliefTrack.p<update_belief_state>68 :  SemI   > [(inform(area=centre)|inform(kind=hotel)|
                                               inform(pricerange=cheap)|inform(type=placetostay)', 1.0)]
               Agent.py <_print_sys_act>569 :  | Sys > inform(name=none, kind='hotel',pricerange='cheap',area='centr
e')
                 Agent.py <_agents_semo>653 :  Domain with CONTROL: CamHotels
                  Agent.py <_print_turn>554 :  ** Turn 2 **
....
```

Exercise: Chat with PyDial

Run

```
> python pydial.py chat config/Tut-hdc-CamInfo.cfg
```

See logfile in:

```
> less _tutoriallogs/Tut-hdc-CamInfo-seed<ID>.chat.log
```

The chat ID is shown when the chat starts

PyDial Architecture

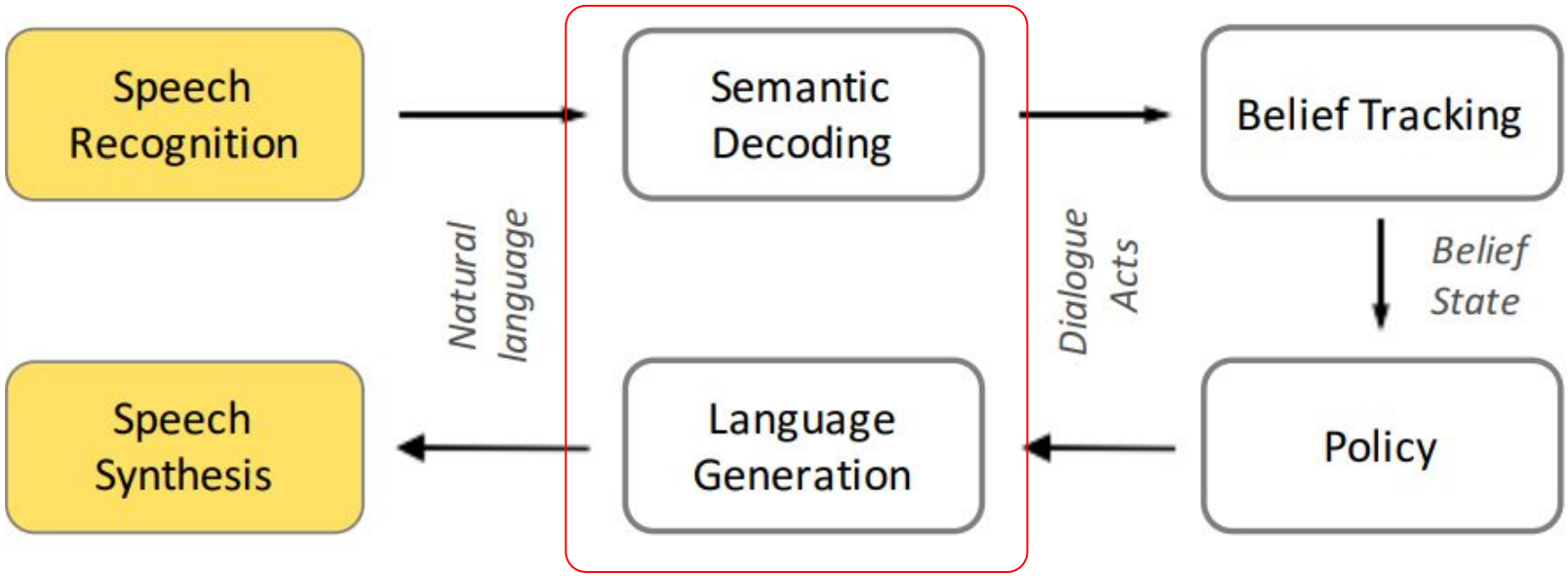

Image source: Stefan Ultes et al., *PyDial: A Multi-domain Statistical Dialogue Systen Toolkit*, 2017

PyDial Architecture

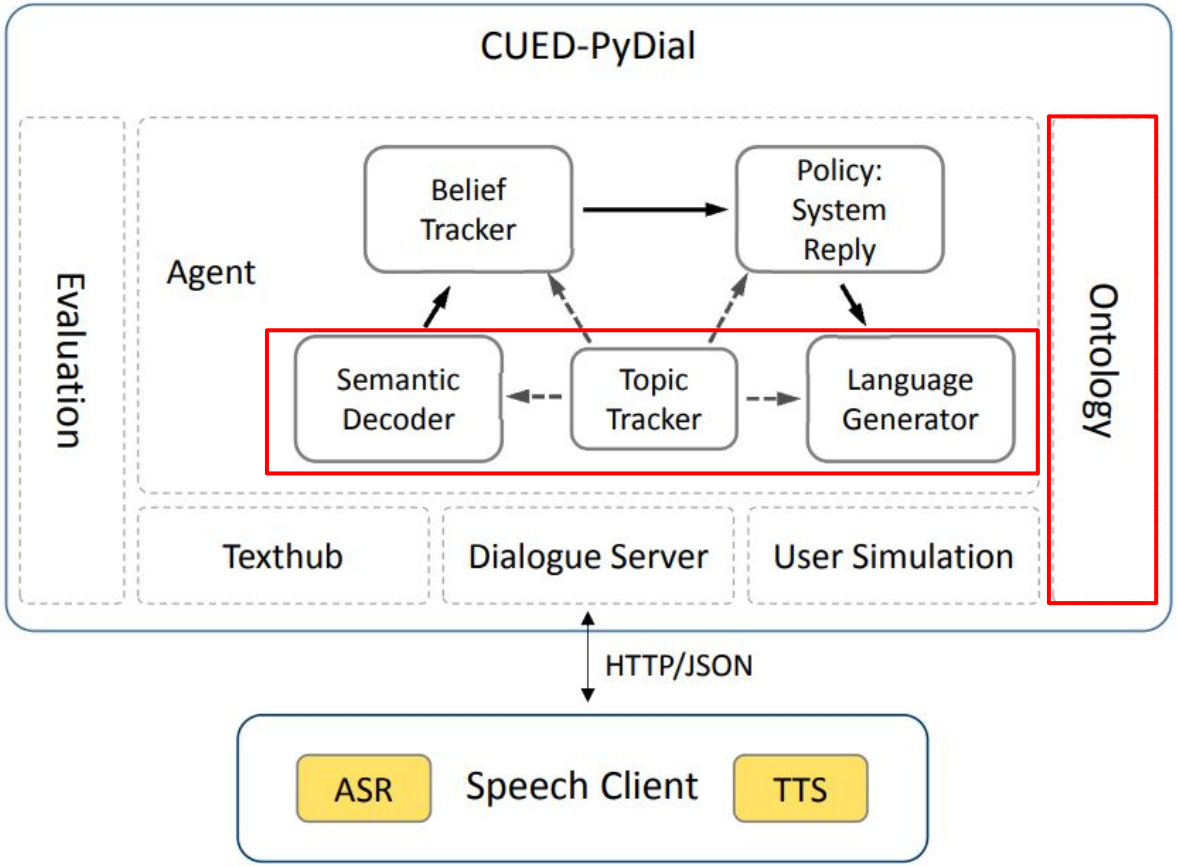

Image source: Stefan Ultes et al., *PyDial: A Multi-domain Statistical Dialogue System Toolkit*, 2017

Configuration File


```
python pydial.py chat config/Tut-hdc-CamInfo.cfg
```

PyDial relies on a configuration file, regardless of the mode, in order to set up the options throughout the system.

The configuration file is global that follows the python config file format

- Sections headed by a name in brackets []
- option=value per line

```
[GENERAL]
 singledomain = True                          # turn off multi-domain handling ...
 domains = CamRestaurants,CamHotels, ....     # or list of possible domains
 tracedialog = 0                              # set trace level to 0,1 or 2
 seed = 12345                                 # set to make simulation reproducible
```

Understanding a Configuration File

config/Tut-hdc-CamInfo.cfg

Adding a New Domain

PyDial provides **ontologyTool.py** in order to add a new domain, if

- The data is stored in an SQLite database (DB)
- The DB has only one table, whose name is the domain identifier (CamRestaurants)
- Attribute *name* is the primary key!
- Each table attribute is a slot
- Binary slots are represented as 1: True and 0:False
- Tip: Use only char to avoid type inconsistencies

```
python scripts/ontologyTool.py -n -d NEW_DOMAIN_NAME -db PATH_TO_DB_FILE --type ENTITY_TYPE
```

Types of slots

- *Informable*: information the user can inform the system about
- *System requestable*: information the system can actively ask the user
- *Requestable*: information the user can request from the system
- *Binary*: information which is in the form of a yes/no or true/false

ATIS: Automatic Terminal Information Service

- Recordings of spontaneous human utterances when interacting with a flight booking system
- ATIS database is relational with flight information

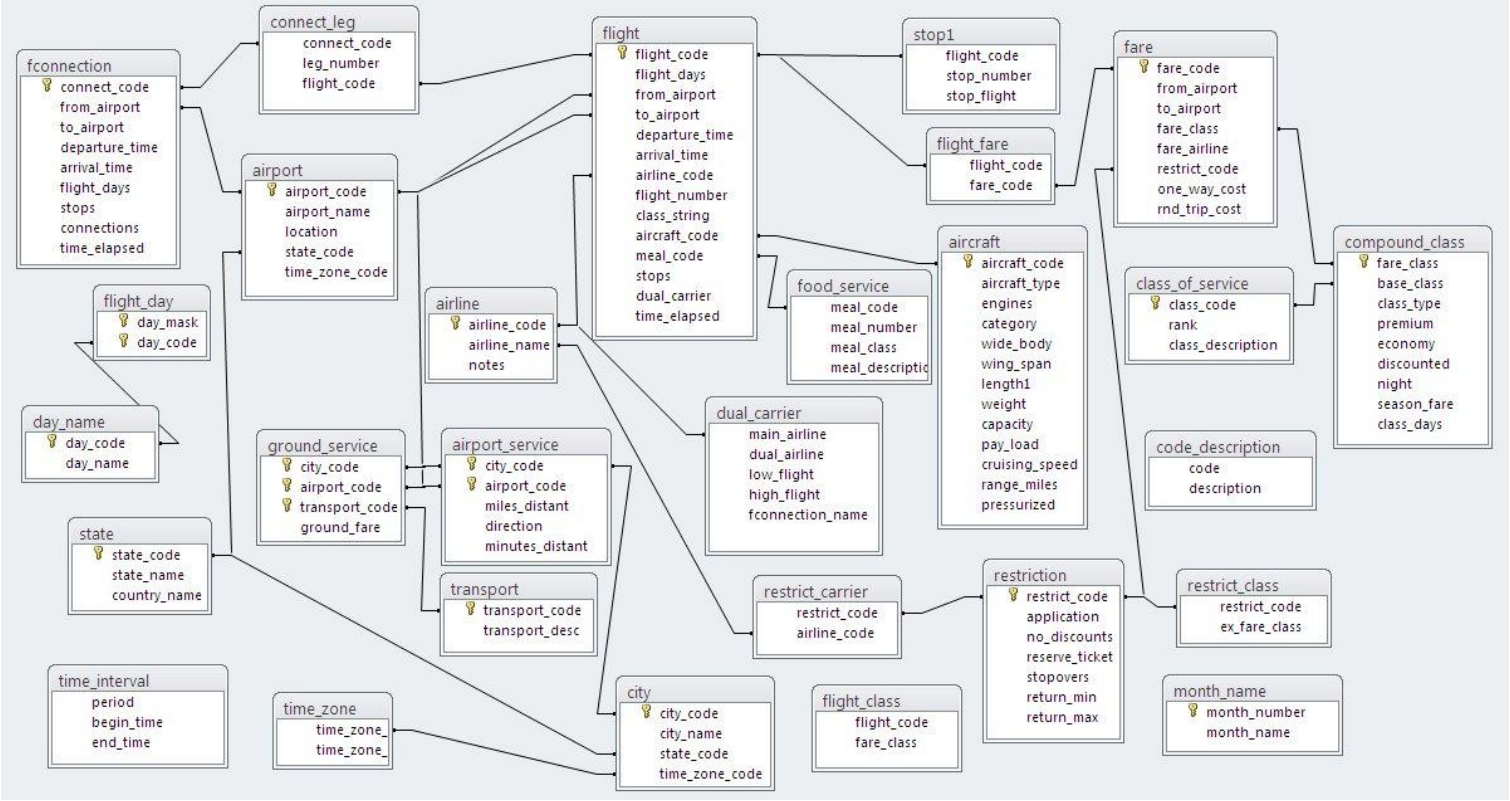

Image source: http://nlp.itcm.edu.mx:8080/nli/description.html

AtisFlights Table

AtisFlights	
Attribute	Meaning (slot type)
🔑 **name**	Numeric id
fromcity	Departing city
fromairpot	Departing airport
fromtime	Departing time
tocity	Arrival city
toairport	Arrival airport
totime	Arrival time
totaltime	Flight duration
days	Days
nonstop	1:direct, 0:non-direct
price	Price

- An oversimplified flight representation
- 3683 flights
- SQlite

['429', 'chicago', 'midway', '805', 'new york', 'westchester county', '1240', '215', 'daily', '1', '662']

['1360', 'las vegas', 'mccarran international', '700', 'san diego', 'burbank', '802', '62', 'not su', '0', '348']

['1592', 'memphis', 'nashville international', '644', 'ontario', 'burbank', '1021', '337', 'daily', '1', '921']

['2849', 'san francisco', 'burbank', '635', 'new york', 'la guardia', '1629', '414', 'daily', '1', '1095']

['2900', 'san jose', 'burbank', '1136', 'columbus', 'hopkins international', '2043', '367', 'daily', '1', '1068']

AtisFlights Domain

User requests:

- I want to fly from Denver to Miami
- I am looking for a direct flight from Denver to Miami
- I would like to travel to Miami from Denver

Information about flights:

- Departing and arrival airports
- Departing and arrival times
- Flight duration
- Price

Exercise: Adding a New Domain

Copy the database to your Pydial directory:

```
> cp /mount/studenten/team-lab-phonetics/2018/pydial-tutorial/exercises/AtisFlights.db
<your_working_dir>/
```

Command to add the domain

```
> python scripts/ontologyTool.py -n -d NEW_DOMAIN_NAME -db PATH_TO_DB_FILE --type ENTITY_TYPE
```

Our case

```
> python scripts/ontologyTool.py -n -d AtisFlights -db <your_working_dir>/AtisFlights.db --type
flight
```

Exercise: Adding a New Domain

Slots vs slot types. Mark the slots according to this table, confirm using ENTER

	Slot	Informable	System requestable	Requestable	Binary
1	days			✔	
2	fromairpot			✔	
3	fromcity	✔	✔	✔	
4	fromtime			✔	
5	name	✔		✔	
6	nonstop	✔	✔	✔	✔
7	price			✔	
8	toairport			✔	
9	tocity	✔	✔	✔	
10	totaltime			✔	
11	totime			✔	

Exercise: Adding a New Domain

The new ontology is created in:

`ontology/ontologies/AtisFlights-rules.json`

Terminal

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
{
    "binary": [
        "nonstop"
    ],
    "discourseAct": [
        "ack",
        "hello",
        "none",
        "repeat",
        "silence",
        "thankyou"
    ],
    "informable": {
        "fromcity": [
            "atlanta",
            "baltimore",
            "boston",
            "burbank",
            "charlotte",
            "chicago",
            "cincinnati",
            "cleveland",
            "columbus",
            "dallas",
            "denver",
            "detroit",
            "fort worth",
            "houston",
            "indianapolis",
            "kansas city",
```

Exercise: Adding a New Domain

Copy files from *exercise* folder:

- Config file

```
cp
/mount/studenten/team-lab-phonetics/2018/pydial-tutorial/exercises/tutorial-AtisFlights.cfg
<your_pydial_dir>/config/
```

- System-response templates file

```
cp
/mount/studenten/team-lab-phonetics/2018/pydial-tutorial/exercises/AtisFlightsMessages.txt
<your_pydial_dir>/semo/templates/
```

Exercise: Adding a New Domain

Run a new dialog session:

```
python pydial.py chat tutorial-AtisFlights.cfg
```

Log by running pydial after this change

```
INFO  :: 11:30:42: root    FlatOntologyManager.py <_set_ontology>93 :  Loading ontology: ontology/ontologies/AtisFlights-rules.json
INFO  :: 11:30:42: root    FlatOntologyManager.py <_set_db>108 :  Loading database: ontology/ontologies/AtisFlights-dbase.db
INFO  :: 11:30:42: root    pydial.py <initialise>373 :  Seed = 679608811
INFO  :: 11:30:42: root    pydial.py <initialise>374 :  Root = /mount/arbeitsdaten34/projekte/slu/Daniel/test_pydial_2/pydial
DIAL  :: 11:30:42: root    pydial.py <chat_command>1125 :  *** Chatting with policies ['AtisFlights']: ***
INFO  :: 11:30:42: root    Agent.py <_hand_control>454 :  Launching Dialogue Manager for domain: topicmanager
```

Exercise: Adding a New Domain

Keywords to detect the flight domain?

User requests:

- I want to fly from Denver to Miami
- I am looking for a direct flight from Denver to Miami
- I would like to travel to Miami from Denver

Keywords = {fly, flight, travel, ...}

Exercise: Adding a New Domain

Setting the domain reference and key words. Add these lines to the following files

semo/RuleSemOMethods.py

```
elif dom=="AtisFlights":
    text="Flight"
```

topictracking/RuleTopicTrackers.py

```
elif dom=="AtisFlights":
    kwds=["flights","flight", "plane", "ticket", "fly", "travel", "trip"]
```

Spoken Language Understanding (SLU)

The module semi (semantic input) is dedicated for semantic decoding of the user's utterances

Dialog turn: ***I am looking for a flight from Denver to Miami***
Dialog act type: inform
Semantic slots: fromcity, tocity
Semantic values: denver, miami
Dialog act: inform(fromcity=denver, tocity=miami)

During the semantic decoding process, the information about dialogue act type, which encodes the system or the user intention in a dialogue turn, and semantic slots and values

The interface SemI.py must be implemented (RegexSemI.py, SVMSemI.py)

Adding SLU Rules for AtisFlights

We have to implement the interface RegexSemI for AtisFlights. Then, we create the class RegexSemI_AtisFlights

For simplicity, we make a copy of *semi/RegexSemI_CamRestaruants.py* in the same directory, naming it *RegexSemI_AtisFlights.py*

Change the class name

```
Line 48 : class RegexSemI_AtisFlights(RegexSemI.RegexSemI):
```

Replace (/mount/studenten/team-lab-phonetics/2018/pydial-tutorial/exercises/methodsFor_RegexSemI_AtisFlights.txt)

```
def __init__(self, repoIn=None):
    RegexSemI.RegexSemI.__init__(self)
    self.domainTag = "AtisFlights"
    self.create_domain_dependent_regex()
```

Adding SLU Rules for AtisFlights

Replace

```
def create_domain_dependent_regex(self):
    self._domain_init(dstring=self.domainTag)

    # DOMAIN DEPENDENT SEMANTICS:
    self.slot_vocab= dict.fromkeys(self.USER_REQUESTABLE,'')

    # Generate regular expressions for requests:
    self._set_request_regex()

    # FIXME:  many value have synonyms -- deal with this here:
    self._set_value_synonyms()
    self._set_inform_regex()
```

Exercise: Adding a New Domain

Keywords to detect the informable slots (fromcity, tocity, direct)?

User requests:

- I want to fly from Denver to Miami
- I would like to travel to Miami from Denver

Keywords → slot: *from* → fromcity, *to* → tocity

inform(fromcity=denver, tocity=miami)

- I am looking for a direct flight from Denver to Miami

Keywords → slot : *from* → fromcity, *to* → tocity, *direct* → nonstop

inform(fromcity=denver, tocity=miami, nonstop='1')

Adding SLU Rules for AtisFlights

Replace

```
    def _set_inform_regex(self):
        self.inform_regex = dict.fromkeys(self.USER_INFORMABLE)
        for slot in self.inform_regex.keys():
            self.inform_regex[slot] = {}
            for value in self.slot_values[slot].keys():
                self.inform_regex[slot][value] = self.rINFORM+"\ " +
self.slot_values[slot][value]
                if slot == "fromcity":
                    self.inform_regex[slot][value] += "|(from)\ " + \
                                                      self.slot_values[slot][value]
                if slot == "tocity":
                    self.inform_regex[slot][value] += "|(to)\ " + \
                                                      self.slot_values[slot][value]
                if slot == "nonstop":
                    if value == '1':
                        self.inform_regex[slot][value] += "|(direct)"
                    else:
                        self.inform_regex[slot][value] += "|(with\ stops)"
```

Exercise: Adding a New Domain

Keywords to detect the requestable slots (price, totaltime, totime, fromtime, fromairport, toairport, days)?

User requests:

- I want to know the flight price / how much is it ?

  Keywords or phrases → slot: *price*, *how much is it* → price

- On which days does this flight happen?

  Keywords or phrases → slot: *days*, *day, (when?)* → days

Adding SLU Rules for AtisFlights

Replace

```
    def _set_request_regex(self):
        self.request_regex = dict.fromkeys(self.USER_REQUESTABLE)
        for slot in self.request_regex.keys():
            self.request_regex[slot] = self.rREQUEST+"\ "+self.slot_vocab[slot]
            self.request_regex[slot] += "|(?<!"+self.DONTCAREWHAT+")(?<!want\ )"+self.IT+"\
"+self.slot_vocab[slot]
            self.request_regex[slot] += "|(?<!"+self.DONTCARE+")"+self.WHAT+"\
"+self.slot_vocab[slot]

        self.request_regex["price"] += "|(price)|(how\ much\ is\ it)"
        self.request_regex["days"] += "|(day)|(days)"
```

Adding SLU Rules for AtisFlights

Replace

```
def _set_value_synonyms(self):
    self.inform_type_regex = r"(fly|flight|travel|(want|looking for) a\ flight)"
```

Natural Language Generation (NLG)

The module semo (semantic output) is dedicated for transforming a dialog act to a human-readable sentence

System dialog act → System output

General outputs
hello() → *Welcome to the Atis Flights dialogue system. How may I help you?*
bye() → *Thank you, good bye.*

Request(slot)
request(fromcity) → *From which city will you depart?*
request(tocity) → *To which city would you like to arrive?*
request(nonstop) → *Do you want a direct flight or with stops?*

Creating NLG Templates for AtisFlights

The file *semo/templates/AtisFlightsMessages.txt* specified in the config file must be created to contain NLG templates (rules) for each dialog act with its slot-value pairs

```
# General Rules
hello() : "Welcome to the Atis Flights dialogue system. How may I help you?";
hello(more) : "Can I help you with anything else?";
null() : "Sorry I am a bit confused; please tell me again what you are looking for.";
repeat() : "Could you please repeat that?";
bye() : "Thank you, goodbye.";

# Requests / Rules for System-Requestable slots
request(fromcity) : "From which city would you like to depart?";
request(tocity) : "To which city would you like to arrive?";
request(nonstop) : "Do you want a direct flight or with stops?";
```

Natural Language Generation (NLG)

The module semo (semantic output) is dedicated for transforming a dialog act to a human-readable sentence

System dialog act → System output

Inform(slot-value pairs)

inform(tocity="denver",name="17",nonstop="*1*",fromcity="atlanta") →
We found a direct flight from Atlanta *to Denver with code* 17 *. What do you want to know about it?*

```
inform(tocity=$W, name=$X, nonstop=$Y, fromcity=$Z) :
'We found a direct flight from $Z to $W with code $X . What do you want to know
about it '
```

inform(name="17",price="866") → *The price of flight with code* 17 *is* 866 *euros*

```
inform(name=$X, price=$Y) :
"The price of the requested flight with ID $X is $Y euros";
```

Creating NLG Templates for AtisFlights


```
#Recomendation inform() in AtisFlightsMessages.txt

inform(name=$W,nonstop=$X,fromcity=$Z, tocity=$Y) : "We found a flight from $Z to $Y,
the flight ID is $W . What do you want to know about it?";

inform(name='none',nonstop=$X, fromcity=$Z, tocity=$Y) : "no flight available";

inform(name=$W,price=$A) : "The price of the requested flight with ID $W is $A euros";
```

Exercise: Adding a New Domain

Run a new dialog session:

```
python pydial.py chat tutorial-AtisFlights.cfg
```

Exercise: Adding SLU Rules for AtisFlights

Create new regular expressions to detect user requests like:

- *What is the flight duration ?*
- *What is the arrival airport ?*
- *Tell me the departing airport ?*
- *What is the departure time ?*

Think of more ways to ask the same information and detect the patterns (keywords and phrases)

Exercise: Adding SLU Rules for AtisFlights

Create new regular expressions to detect user requests for fromairport, toairport, *totaltime, fromtime, totime in semi/RegexSemI_AtisFlights.py*

```
def _set_request_regex(self):
     ….
     self.request_regex["totaltime"] += "|<your_regular_expression>"
     self.request_regex["fromtime"] += "|<your_regular_expression>"
     self.request_regex["totime"] += "|<your_regular_expression>"
     self.request_regex["toairport"] += "|<your_regular_expression>"
     self.request_regex["fromairport"] += "|<your_regular_expression>"
```

Exercise: Add NLG Templates for AtisFlights

Create new NLG templates. Examples:

- *The flight with code 114 lasts 125 minutes*
- *The flight with code 114 arrives at John F. Kennedy International Airport*
- *The flight with code 114 departs at 7:45 h*
- *The flight arrives at with code 114 at 19:45 h*

Exercise: Add NLG Templates for AtisFlights

Add the templates for these cases in semo/templates/AtisFlightsMessages.txt

```
#Recomendation inform()


inform(name='none',fromcity=$Z, tocity=$Y) : "<your_message>";

inform(name=$W,fromairport=$A) : "<your_message>";
inform(name=$W,toairport=$A) : "<your_message>";

inform(name=$W,fromtime=$A) : "<your_message>";
inform(name=$W,totime=$A) : "<your_message>";
inform(name=$W,totaltime=$A) : "<your_message>";
```

Multi-domain Dialog System

Let's use the config file below that supports Restaurants and Hotels to add Flights:

`config/texthub.cfg`

Add the domain

```
Line 2: domains = CamRestaurants,CamHotels,AtisFlights
```

Append sections and options

```
[policy_AtisFlights]
policydir = _policydir

[semi_AtisFlights]
semitype = RegexSemI

[semo_AtisFlights]
semotype = BasicSemO
templatefile = semo/templates/AtisFlightsMessages.txt
```

Multi-domain Dialog System

Run a new dialog session:

```
python pydial.py chat config/texthub.cfg
```

References

- Stefan Ultes, Lina M. Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Pawel Budzianowski, Nikola Mrksić, Tsung-Hsien Wen, Milica Gasic, and Steve Young. PyDial: A Multidomain Statistical Dialogue System Toolkit. In Proceedings of ACL 2017, System Demonstrations, pages 73–78, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL http: //aclweb.org/anthology/P17-4013.

Files

In the directory: /mount/studenten/team-lab-phonetics/2018/pydial-tutorial/solutions

There is a copy of the files mentioned in this document (with the corresponding changes) that must be located according to the list below in your own pydial

- config/texthub.cfg
- config/tutorial-AtisFlights.cfg
- topictracking/RuleTopicTrackers.py
- semi/RegexSemI_AtisFlights.py
- semo/RuleSemOMethods.py
- semo/templates/AtisFlightsMessages.txt