UFAL Dialogue act scheme

The purpose of this document is to describe the structure and function of dialogue acts used in spoken dialogue systems developed at UFAL, MFF, UK, Czech Republic.

Filip Jurcicek
06/05/12

Table of Contents

Definition of dialogue acts

In a spoken dialogue system, the observations and the system actions are represented by dialogue acts. Dialogue acts represent basic intents (such as inform, request, etc.) and the semantic content in the input utterance (e.g. type=hotel, area=east). In some cases, the value can be omitted, for example, where the intention is to query the value of a slot e.g. ``request(food)''.

In the UFAL Dialogue Act Scheme (UDAS), a dialogue act (DA) is composed of one or more dialogue act items (DAI). A dialogue act item is defined as a tuple composed of a dialogue act type, a slot name, and the slot value. Slot names and slot values are domain dependent, therefore they can be many. In the examples which follows, the names of the slots and their values are drown form a information seeking application about restaurants, bars and hotels. For example in a tourist information domain, the slots can include “food” or “pricerange” and the values can be such as “Italian”, “Indian” or ”cheap”, “midpriced”, or “expensive”.

This can be described in more formal way as follows:

```
DA    = (DAI)+
DAI   = (DAT, SN, SV)
DAT   = (ack, affirm, apology, bye, canthearyou, confirm, iconfirm, deny,
hangup, hello, help, inform, negate, notunderstood, null, repeat, reqalts,
reqmore, request, restart, select, thankyou)
```

where SN denotes a slot name and SV denotes a slot value.

The idea of dialogue comes from the information state update (ISU) approach of defining a dialogue state. In ISU, a dialogue act is understood as a set of deterministic operations on a dialogue state which which result in a new updated state. In the UFAL dialogue act scheme, the update is performed on the

slot level.

The following explains each dialogue act type:

```
ack         - "Ok" - back channel
affirm      - simple "Yes"
apology     - apology for misunderstanding
bye         - end of a dialogue – simple "Goodbye"
confirm     - user tries to confirm some information
canthearyou – system or user does not hear the other party
deny        - user denies some information
hangup      - the user hangs up
hello       - start of a dialogue – simple "Hi"
help        - request for help
inform      - user provide some information or constraint
negate      - simple "No"
null        - silence, empty sentence, something that is not possible to
              interpret, does nothing

              It can be also used when converting a dialogue act item
              confusion network into a N-best list to hold all the
              probability mass connected with all dialogue acts  which were
              not added to the N-best list. In other words probability mass
              of pruned DA hypotheses.

notunderstood – informs that the last input was not understood
repeat      - request to repeat the last utterance
irepeat     - repeats the last utterance
reqalts     - ask for alternatives
reqmore     - ask for more details
request     - user requests some information
restart     - request to restart
select      - user or the system wants the other party to select between
              two values for one slot
thankyou    - simply thank you
```

NOTE: Having this set of acts we cannot confirm that something is not equal to something, e.g. confirm(x!=y) → confirm(pricerange != 'cheap') → “Is'n it cheap?” If we used confirm(pricerange = 'cheap') then it means “Is it cheap?” In both cases, it is appropriate to react in the same way e.g. inform(pricerange='cheap') or deny(pricerange = 'cheap').

NOTE: Please note that all slot values are always placed in quotes ' '.

Dialogue act examples

This section presents examples of dialogue acts:

```
ack()                       'ok give me that one'
                            'ok great'

affirm()                    'correct',
```

```
                              'erm yeah'

appology()                    'sorry'
                              'sorry I did not get that'

bye()                         'allright bye'
                              'allright then bye'

canthearyou()                 'halo'
                              'are you still there'

confirm(addr='main square') 'erm is that near the central the main
                            square'
                              'is it on main square'

iconfirm(addr='main square') 'Ack, on main square,'

iconfirm(near='cinema')     'You want something near cinema,'

deny(name='youth hostel')   'not the youth hostel'
deny(near='cinema')         'ok it doesn't have to be near the cinema'

hello()                       'hello',
                              'hi',
                              'hiya please'

help()                        'can you help me'

inform(='main square')      'main square'

inform(addr='dontcare')     'i don't mind the address'

inform(food='chinese')      'chinese'
                              'chinese food'
                              'do you have a chinese food'

negate()                      "erm erm no i didn't say anything"
                              'neither'
                              'no'

null()                        '' - empty sentence
                              'abraka dabra' - somthing not interpretable

repeat()                      'can you repeat'
                              'could you repeat that'
                              'could you repeat that please'

reqalts()                     'and anything else'
                              'are there any other options'
                              'are there any others'

reqmore()                     'can you give me more dtails'
```

```
request(food)                'do you know what food it serves'
                             'what food does it serve'

request(music)               'and what sort of music would it play'
                             'and what type of music do they play in these
                             bars'

restart()                    'can we start again please'
                             'could we start again'

select(food=”Chinese”,       'do you want Chines or Italian food'
food=”Italian”)

thankyou()                   "allright thank you then i'll have to look
                             somewhere else"
                             'erm great thank you'
```

If the system wants to inform that no venue is matching provided constraints, e.g. “There is no Chinese restaurant in a cheap price range in the city centre” the system uses the inform(name='none') dialogue acts as in

Utterance: There is no Chinese restaurant in a cheap price range in the city centre”
Dialogue act: `inform(name='none')&inform(venue_type='restaurant')&`
`inform(food_type='Chinese')&inform(price_range='cheap')`

There are examples of dialogue acts composed of several DAIs:

```
reqalts()&thankyou()         'no thank you somewhere else please']

request(price)&thankyou()    'thank you and how much does it cost',
                             'thank you could you tell me the cost',

affirm()&inform(area='south')&inform(music='jazz')&inform(type='bar')
&request(name)
"yes i'd like to know the name of the bar in the south part of town that
plays jazz music",
'yes please can you give me the name of the bar in the south part of town
that plays jazz music'

confirm(area='central')&inform(name='cinema')
'is the cinema near the centre of town'

deny(music='pop')&inform(music='folk')
"erm i don't want pop music i want folk folk music"]

hello()&inform(area='east')&inform(drinks='cocktails')&
inform(near='park')&inform(pricerange='dontcare')&inform(type='hotel')
"hi i'd like a hotel in the east of town by the park the price doesn't
matter but i'd like to be able to order cocktails"
```

An example dialogue form tourist information domain is in the following table:

Turn	Transcription	Dialogue act
System	Hello. How may I help you?	hello()
User	Hi, I am looking for a restaurant.	inform(venue=”restaurant”)
System	What type of food would you like?	request(food)
User	I want Italian.	inform(food=”Italian”)
System	Did you say Italian	confirm(food=”Italian”)
User	Yes	affirm()

Semantic Decoding and Ambiguity

Very often there are many ways as to map (to interpret) a natural utterance into a dialogue act, , some times because of natural ambiguity of a sentence – sometimes because of the speech recognition errors. Therefore, a semantic parser will generate multiple hypotheses. In this case, each hypothesis will be assigned a probability meaning the likelihood of being correct and the dialogue manager will resolve this ambiguity in the context of the dialogue (e.g. other sentences).

For example, the utterance “I wan an Italian restaurant erm no Indian” can be interpreted as:

```
inform(venue=”restaurant”)&inform(food=”Italian”)&deny(food=Indian)
```

or

```
inform(venue=”restaurant”)&inform(food=”Indian”)
```

In the first case, the utterance is interpreted that the user wants Italian restaurant and does not want Indian. However, in the second case, the user corrected what he just mistakenly said (that he wants Indian restaurant).

Please remember that semantic parsers should interpret an utterance only on the information present in the sentence. It is up to the dialogue manager to interpret it in the context of the whole dialogue

```
inform(type=restaurant)&inform(food='Chinese')
'I want an Chinese restaurant'

inform(food='Chinese')
'I would like some Chinese food'
```

In the first case, the user explicitly says that he/she is looking for a restaurant. However, in the second case, the user said that he/she is looking for some venue serving Indian food which can be both a restaurant or only a take-away.

Building a statistical SLU parser for a new domain

From experience, it appears that the easiest approach to build a statistical parser for a new domain is to start with build a handcrafted (rule based) parser. There are several practical reasons for that:

1. a handcrafted parser can serve as a prototype module for a dialogue system when no data is available,
2. a handcrafted parser can serve as a baseline for testing data driven parsers,
3. a handcrafted parser in information seeking applications, if well implemented, achieves about 95% accuracy on transcribed speech, which is close to accuracy of what the human annotators achieve,
4. a handcrafted parser can be used to obtain automatic SLU annotation which can be later hand corrected by humans.

To build a data driven SLU, the following approach is recommended:

1. after some data is collected, e.g. a prototype of dialogue system using a handcrafted parser, the audio from the collected calls is transcribed (using humans) and then parsed using the handcrafted parser,
2. the advantage of using automatic SLU annotations is that they are easy to obtain and reasonably accurate only several percent lower to what one can get from human annotators.
3. if better accuracy is needed then it is better to fix the automatic semantic annotation by humans,
4. then a data driven parser is trained using this annotation

Note that the main benefit of data driven SLU methods comes from the ability to robustly handle erroneous input. Therefore, the data driven SLU should be trained to map **the recognised speech** to the dialogue acts (e.g. obtained by the handcrafted parser on the transcribed speech and then corrected by human annotator).

Comments

The previous sections described the general set of dialogue acts in UFAL dialogue systems. However, exact set of dialogue acts depends on a specific application domain and is defined by the domain specific semantic parser.

The only requirement is that all the output of a parser must be accepted by the dialogue manager developed for the particular domain.

Apendix A: UFAL Dialogue acts

Act	Description
ack()	back channel – simple OK
affirm()	acknowledgement - simple "Yes"
apology()	apology for misunderstanding
bye()	end of dialogue
canthearyou()	signalling problem with communication channel or that there is an unexpected silence
confirm(x=y)	confirm that x equals to y
iconfirm(x=y)	Implicitly confirm that x equals to y
deny(x=y)	denies some information, equivalent to inform(x != y)
hangup()	end of call because someone hungup
hello()	start of dialogue
help()	provide context sensitive help
inform(x=y)	inform x equals to y
inform(name=none)	inform that “there is no such entity that ... “
negate()	negation - simple “No”
notuderstood()	informs that the last input was not understood
null()	silence, empty sentence, something that is not possible to interpret, does nothing
repeat()	asks to repeat the last utterance
irepeat()	repeats the last uttered sentence by the system
reqalts()	request for alternative options
reqmore()	request for more details bout the current option
request(x)	request for information about x
restart()	restart the dialogue, forget all provided info
select(x=y, x=z)	select between two values of the same slot
thankyou()	simply thank you