**Part 1c: reasoning and configurability**

In the third part you will implement a simple reasoning component, which uses inference rules to determine additional restaurant properties based on initial properties. The list of inference rules is provided below. Note that the first rule is an AND rule, which means that both conditions need to be present for the rule to be applicable.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | antedecent | consequent | t/f | description |
| 1 | cheap AND good food | touristic | True | a cheap restaurant with good food attracts tourists |
| 2 | romanian | touristic | False | Romanian cuisine is unknown for most tourists and they prefer familiar food |
| 3 | busy | assigned seats | True | in a busy restaurant the waiter decides where you sit |
| 4 | long stay | children | False | spending a long time is not advised when taking children |
| 5 | busy | romantic | False | a busy restaurant is not romantic |
| 6 | long stay | romantic | True | spending a long time in a restaurant is romantic |

First, add the new properties (food quality, crowdedness, length of stay) into the csv file for each restaurant, using random values. Then build a question into your dialog manager after preferences are collected that asks for additional requirements, i.e., values for the consequents of the rules. These requirements do not need to be used during the dialog, i.e., the dialog state transition system from Part 1b can be used to find the list of suitable restaurants. After this list is determined, for each of the candidate restaurants, apply all rules for which the properties of the restaurant satisfy the antecedent of the rule. Present the reasoning for the final recommendation to the user.

Example:

user: I'm looking for an Italian restaurant

system: What part of town do you have in mind?

user: The south part of town please

system: Which price range would be suitable for you?

user: Any price is ok

-----------------------------------------------------------------------

all preferences collected, suitable candidates:  
1. pizza hut cherry hinton, 2. frankie and bennys, 3. la margherita, 4. zizzi cambridge

-----------------------------------------------------------------------

system: Do you have additional requirements?

user: I would like the restaurant to be romantic

-----------------------------------------------------------------------

assume candidate 1 and 2 have the property 'busy', and 3 and 4 have the property 'not busy' and 'long time' in the csv  
inference: apply rule 5 on restaurant 1 and 2, infer 'romantic=false'  
inference: apply rule 6 on restaurant 3 and 4, infer 'romantic=true'  
choose randomly from the two remaining options

-----------------------------------------------------------------------

system: I recommend 'zizzi cambridge', it is an expensive Italian restaurant in the south of town.

The restaurant is romantic because it allows you to stay for a long time.

**Configurability**

As preparation for Part 2 of the project you need to implement some degree of configurability in your dialog manager, i.e., implement the dialog system such that certain features can be switched on or off:

* Levenshtein edit distance for preference extraction
* Ask user about correctness of match for Levenshtein results
* Allow preferences to be stated in random order or not
* Allow preferences to be stated in a single utterance only, or in multiple utterances with one preference per utterance only, or without restrictions (any number of utterances and any number of preferences per utterance)
* Use one of the baselines for dialog act recognition instead of the machine learning classifier
* Insert artificial errors in preference extraction
* Start offering suggestions after the first preference type is recognized vs. wait until all preference types are recognized
* Use formal or informal phrases in system utterances
* Allow dialog restarts or not
* Introduce a delay before showing system responses
* Always offer a small or large number of recommendations
* Ask confirmation for each preference or not
* Allow users to change their preferences or not
* OUTPUT IN ALL CAPS OR NOT
* Use text-to-speech for system utterances
* **Any other modification** of the state transition diagram you consider interesting for user testing

**Deliverables:**

* Implementation of implication rules
* Implementation of configurability for selected features
* Integration of the two deliverables above as part of the dialog management system from Part 1b