CSCI 644: Natural Language Dialogue Systems
Spring 2020

HW 1
Due on February 14th, 2020 at 10:00 AM PST

Description:
For this assignment you are tasked with building a finite state dialogue manager system that provides restaurant recommendations (text input, text output). We provide some details on each of the modules of this system below.

Dialogue Manager:
You should have a simple frame-based dialogue manager (DM). The system should have three slots which are associated with the following information: type of food, price information, and location.

The possible values for each slot are:

<FOOD_TYPE>: empty | any | Italian | Japanese | Chinese | Mexican | Greek
<PRICE>: empty | any | cheap | medium-priced | expensive
<LOCATION>: empty | any | Marina Del Rey | Venice | Santa Monica | Korea Town | Playa Vista | Hollywood

Initially all the slots have the "empty" value.

The DM should first query for food type (e.g., "what type of food do you want?"), then for price (e.g., "how expensive a restaurant do you want?"), and last for location (e.g., "where do you want the restaurant to be located?").

Your system should have an explicit confirmation policy that confirms every value for each slot value provided by the user. So for each slot there should also be a confirmation slot that must be filled in positively before moving on in the dialogue. For example, for the slot FOOD_TYPE there should also be a slot FOOD_TYPE_CONF. If FOOD_TYPE has a value and the user has positively confirmed this value then "FOOD_TYPE_CONF=yes". If FOOD_TYPE has a value and the user has not confirmed this value then "FOOD_TYPE_CONF=no". If FOOD_TYPE has a value and the user has negatively confirmed this value then "FOOD_TYPE_CONF=no" and the slot FOOD_TYPE should get the "empty" value. Your system should not move on until it receives positive confirmation and in the case of negative confirmation it should return to the previous query/slot.

The DM should be able to handle cases where the user provides multiple pieces of the expected restaurant information at any point in the dialogue. The DM should skip the query corresponding to the slot in these cases for which the user already provided the slot's associated value. If a user provides more than one slot value in response to a query, each value should be confirmed individually.

For example, your system should be able to have the following dialogue.
System: What type of food do you want?
User: I want a cheap Italian restaurant.
System: Ok you said you wanted an Italian restaurant, right?
User: Yes.
System: Ok you said you wanted a cheap restaurant, right?
User: Yes.
System: Ok where do you want the restaurant to be located?

After all slots in all frames have been filled the DM should then query a database that is provided in a tab separated value file (TSV) consisting of five columns: <RESTAURANT_NAME> \t <RESTAURANT_PHONE_NUMBER> \t <FOOD_TYPE> \t <PRICE> \t <LOCATION>

The DM should then select all restaurants that match the user's stated preferences and output the total number of restaurants that match, the names of the restaurants, and their phone numbers. Note that the value *any* indicates that all values for that category are acceptable to the user and thus no restaurants should be filtered from the results for that value.

Natural Language Understanding:
Natural language understanding should be performed using keyword matching. The expected key words have been listed for each slot above in the DM section. For confirmation queries the keywords are "yes" and "no" respectively.

Natural Language Generation:
Natural Language Generation should be template-based and your confirmation utterances should contain the hypothesized user-provided values. Exact wording for each query isn't specified but each query must be understandable enough to an English speaker that it can be easily associated to the expected dialogue act.

Here are two complete example dialogues that your system should be able to handle:

Example 1
System: What type of food do you want?
User: Hello.
System: What type of food do you want?
User: I want Japanese food.
System: Ok you said you wanted a Japanese restaurant, right?
User: Yes.
System: How expensive a restaurant do you want?
User: I want something medium-priced.
System: Ok you said you wanted a medium-priced restaurant, right?
User: No, I want something cheap.

System: Ok you said you wanted a cheap restaurant, right?
User: Yes.
System: Ok where do you want the restaurant to be located?
User: Marina Del Rey.
System: Ok you said you wanted a restaurant in Marina Del Rey, right?
User: Yes.
System: I found 1 restaurant matching your query. Tsujitan Artisan Noodles is a cheap Japanese restaurant in Marina Del Rey. The telephone number is 310-235-4352.

Example 2
System: What type of food do you want?
User: Maybe something.
System: What type of food do you want?
User: Any.
System: Ok you said you wanted any type of food restaurant, right?
User: Yes.
System: How expensive a restaurant do you want?
User: Something expensive in Hollywood.
System: Ok you said you wanted an expensive restaurant, right?
User: What?
System: Ok you said you wanted an expensive restaurant, right?
User: Yes.
System: Ok you said you wanted a restaurant in Hollywood, right?
User: Yes.
System: I found 3 restaurants matching your query. Hiroshima is an expensive Japanese restaurant in Hollywood. The phone number is 310-356-7264. Mr. Changs is an expensive Chinese restaurant in Hollywood. The phone number is 310-535-7545. Canyon Road is an expensive Mexican restaurant in Hollywood. The phone number is 310-235-4636.

Grading Notice:
Please follow the instructions carefully. Any deviations from the instructions will result in your grade being reduced. If you have any doubts, please use the discussion board on Piazza. Please do ***not*** post code snippets on Piazza. Do not assume anything that is not explicitly stated. **You need to email your solution to Eli (pincus@ict.usc.edu). You need to send a zip file even if it contains just 1 Python file because otherwise it could be rejected by the email server.**

•You must use PYTHON (Python 3.8.1, the latest version) to implement your code. You must not use any other Python libraries besides the default libraries provided by the Python environment (Python Standard Library). You must implement any other functions or modules by yourself.

•You need to create a file named “hw1cs644s20.py”. The command to run your program should be as follows: “python hw1cs644s20.py”.

•All files used in this assignment that need to be read by your code will use UNIX line endings (“\n”).

•If we are unable to execute your code successfully, you will not receive any credits.

•You will get partial credit based on the percentage of test cases that your program gets right.

•There will be a penalty for late submission.