CSCI 644: Natural Language Dialogue Systems
Spring 2020

HW 2
Due on April 13th, 2020 at 10:00 AM PDT

Description:
For this assignment you are tasked with using reinforcement learning to train a dialogue policy to be incorporated into the dialogue system that you built for Assignment 1. Compared to Assignment 1, the capabilities of the resulting system will be limited (see below). This is in order to simplify the learning problem and facilitate faster training.

So the system is not required to handle over-answering. We assume that the user will respond to the system's prompt and provide only the information requested by the system, nothing more, nothing less. The system will use explicit confirmation to confirm the user's input, and the user will not be able to change their mind. When the system tries to confirm their input, they have to say "yes".

As in Assignment 1, the system should handle three slots which are associated with the following information: type of food, price, 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. 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".

So far everything is similar to Assignment 1, with the exception that the user has to provide the information requested (empty responses, over-answering, or irrelevant responses are not allowed), and always confirm their previous input.

Reinforcement Learning
The reinforcement learning policy should learn to request information about each slot and then confirm this slot (querying the database will be done during testing as in Assignment 1). So for the reinforcement learning problem the state space can be simplified. You actually need the following state variables:

<FOOD_TYPE_FILLED>: no | yes

<PRICE_FILLED>: no | yes
<LOCATION_FILLED>: no | yes
<FOOD_TYPE_CONF>: no | yes
<PRICE_CONF>: no | yes
<LOCATION_CONF>: no | yes

There are 6 possible system actions: REQUEST_FOOD_TYPE, REQUEST_PRICE, REQUEST_LOCATION, EXPLICIT_CONFIRM_FOOD_TYPE, EXPLICIT_CONFIRM_PRICE, EXPLICIT_CONFIRM_LOCATION.

To train the policy you need a simulated user. The simulated user should always provide the information requested by the system. For example, if the system action is "REQUEST_FOOD_TYPE" then the simulated user response should be "PROVIDE_FOOD_TYPE" with probability 1.0. The simulated user should always respond "yes" to the system's confirmation request. For example, if the system action is "EXPLICIT_CONFIRM_FOOD_TYPE" then the simulated user response should always be "YES" with probability 1.0. Likewise for the rest of the slots.

The reward function is as follows: 500 points when all slots are both filled and confirmed and -5 points for every system action. These are the only rewards/penalties you should use.

For reinforcement learning you have to use the Q-learning algorithm with the following configuration: exploration rate starts at 20% (ε-greedy exploration) and is decreased by 2% every 10 episodes. The value of γ is 0.99. The learning rate α is equal to 1/(1+current_episode_number). Each episode should time out after 20 system actions. With this setup the policy should converge in about 100 episodes.

Note that the order of handling the slots (e.g., whether price is handled before location or vice versa) does not matter. But the policy should learn first to fill a slot and then try to confirm it.

After training, the learned dialogue policy should be saved in a file using the following format (comma-delimited) where each line corresponds to a state and the best action for that state:
<FOOD_TYPE_FILLED>,<PRICE_FILLED>,<LOCATION_FILLED>,<FOOD_TYPE_CONF>,<PRICE_CONF>,<LOCATION_CONF>,<BEST_ACTION_FOR_THIS_STATE>
Your program should also save a comma-delimited file with information about the total reward per episode where each line corresponds to a training episode and has the current format:
<CURRENT_EPISODE_NUMBER>,<TOTAL_REWARD_AT_END_OF_THIS_EPISODE>
The information in this file will help you track whether the policy converges or not.

During testing your program should read the dialogue policy from the saved file and use this policy to drive the system behavior. You may use the natural language understanding and natural language components from Assignment 1. The goal of the

reinforcement learning policy is to fill and confirm the slots. After all the slots are both filled and confirmed the system should query the database in the same way as in Assignment 1. The database will also remain the same as in Assignment 1.

More explanations can be found in the slides of Lecture 7.

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: I want Japanese food. – the user has to provide information about the type of food only
System: Ok you said you wanted a Japanese restaurant, right?
User: Yes. – the user has to say "yes" only
System: How expensive a restaurant do you want?
User: I want something cheap. – the user has to provide information about the price only
System: Ok you said you wanted a cheap restaurant, right?
User: Yes. – the user has to say "yes" only
System: Ok where do you want the restaurant to be located?
User: Marina Del Rey. – the user has to provide information about the location only
System: Ok you said you wanted a restaurant in Marina Del Rey, right?
User: Yes. – the user has to say "yes" only
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: Any. – the user has to provide information about the type of food only and "any" is a possible value
System: Ok you said you wanted any type of food restaurant, right?
User: Yes. – the user has to say "yes" only
System: How expensive a restaurant do you want?
User: Something expensive. – the user has to provide information about the price only
System: Ok you said you wanted an expensive restaurant, right?
User: Yes. – the user has to say "yes" only
System: Ok where do you want the restaurant to be located?
User: In Hollywood. – the user has to provide information about the location only
System: Ok you said you wanted a restaurant in Hollywood, right?
User: Yes. – the user has to say "yes" only
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 containing all the required files.**

•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). Also, the NumPy library is allowed. You must implement any other functions or modules by yourself.

•You need to create a file named "hw2cs644s20_train.py" and a file named "hw2cs644s20_test.py". The commands to run your programs should be "python hw2cs644s20_train.py" and "python hw2cs644s20_test.py" respectively.

•Running "hw2cs644s20_train.py" should generate the following files: "policy.txt" and "rewards.csv".

•When you run "hw2cs644s20_test.py" the program should read "policy.txt" and use this policy to drive the behavior of the system.

•All files used in this assignment that need to be read or saved by your code should 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.

•If you made errors in Assignment 1, you need to carefully read the report accompanying your grade to make sure that these errors are corrected. Some errors will not be applicable due to simplifications but some will. For example, if the user enters "I want a restaurant in Marina Del Rey" or "I want something medium-priced" the system should be able to handle these inputs without problems because the format of the input keywords is exactly the same as the format of the database entries.

•There will be a penalty for late submission.