Reinforcement learning Final project

Charan Nama Arunkumar
State University of New York at Buffalo
Buffalo, NY 14260
charanna@bufaflo.edu

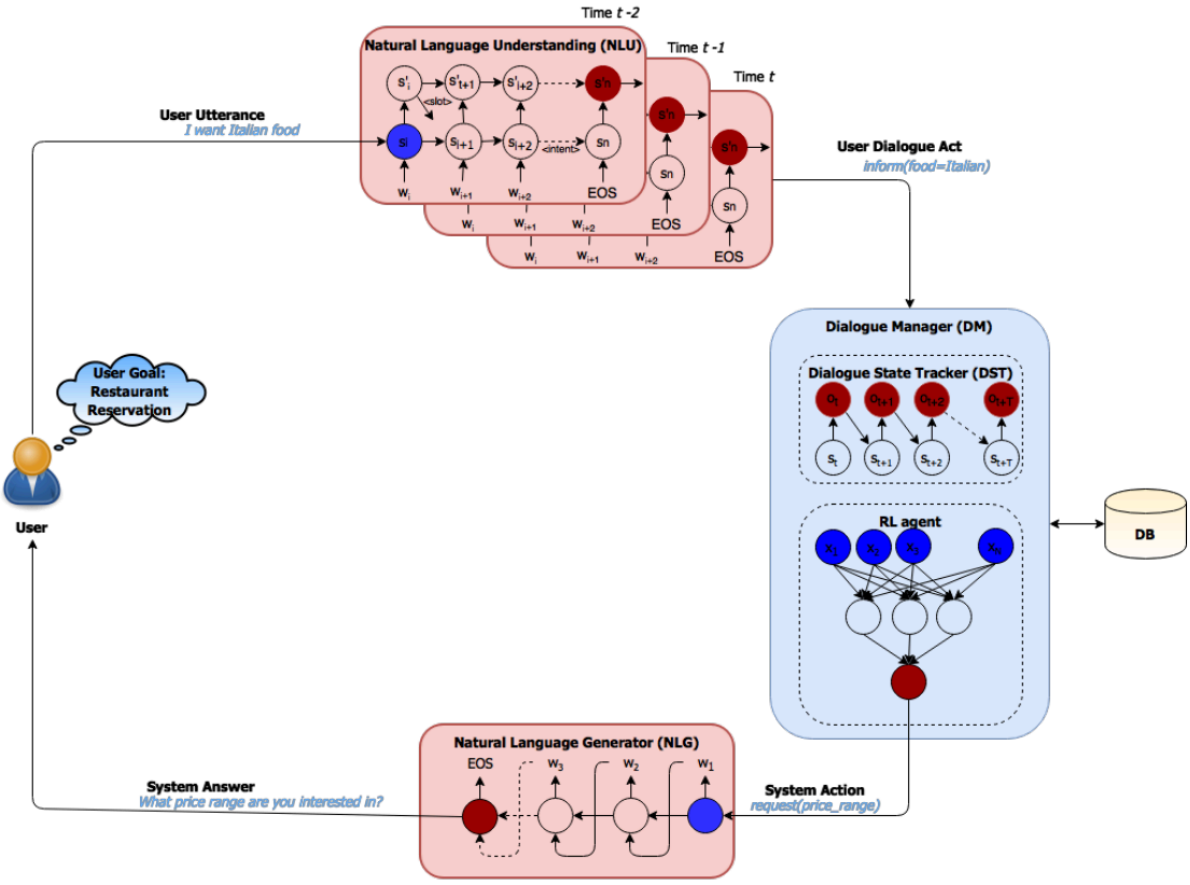
Abstract

Exploring the use of Reinforcement learning to help build goal oriented chatbots. In my project I have tried to build a chatbot which can help book movie tickets given certain user constraints on where he/she is located, time slots available, etc.,

1 Relevance and novelty of project

This domain and application is relevant as chatbots are more common today than ever. We see them on websites to initiate conversations with site visitors, automated speech recognition used in place of call center representatives to help with most common requests like USPS support automation which helps customers track, cancel or schedule pickups without human intervention, we also see this with the smartphones in the form of google assistant or Siri. The project is derived from a paper which was presented in 2017 [1] which uses Deep Reinforcement learning to simulate a goal oriented chatbot. The paper only uses Deep Q networks in their tests. I have tried using Double Deep Q Networks and Actor-Critic in my evaluation and tests.

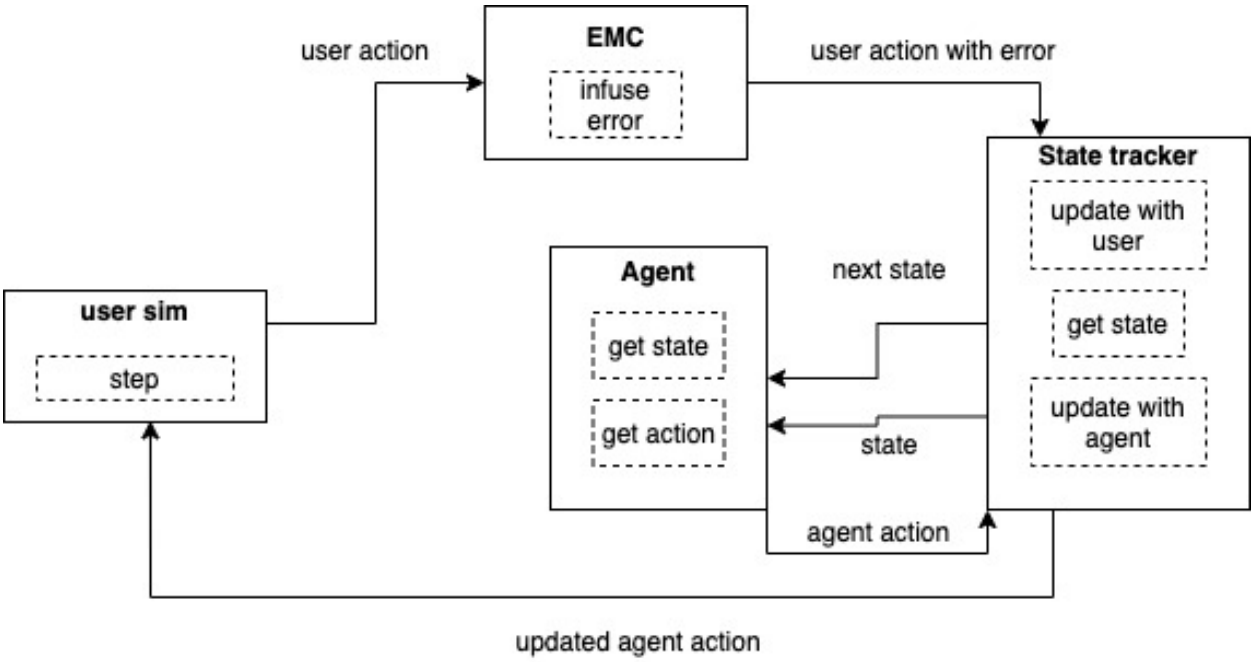
2 Background to the project



Let us go over a brief introduction on what dialogue systems are how we aim to solve the problem. In the given representation of the dialogue system. The user interacts with the agent not directly but through 2 Natural Language systems. The NLU(natural language understanding) unit helps convert the user's utterance in speech or words into semantic frames which are similar to json representation which give the relevant information necessary for this problem. A dialogue manager takes care of maintaining history of utterances and also helps decide what the system utterance will be. In our case there is also a database attached which the dialogue manager checks for possible matches. The NLG(natural language generator) then converts the semantic frame reply from the agent to natural language which the user can understand.

3 Approach to solve the problem

We represent the problem as a Partially Observable Markov Decision Process(POMDP) and go about applying our knowledge on deep learning algorithm to interact and solve the problem. We make use of a rule based chatbot in the initial stages to create the first set of POMDP state, action, reward, new state pairs and then train our algorithms of **DQN, DDQN** and **Actor-Critic** to train them. This is **imitation learning** where we are using the initially learned reward dynamics from the rule based agent to start with. We do this to skip long training and exploration that the vanilla algorithms would need.



4 Special units in the system

4.1 Role of User simulator

In RL methods you need constant feedback from user to understand whether the reply given by the bot is good or not. In the current system the user simulator does this, it decides based on certain rules whether the chatbot asked useful questions or not. The user simulator can give rewards -1, 0 or +1. This way you don't have to bother regular users with feedback for each step.

4.2 Role of Error Model Controller

The error model controller has multifold advantages:
1. It helps simulate real life natural language component errors
2. It helps create more training data from small samples
3. Noise is always a good way to prevent overfitting

The error model controller can do one of 3 things:
1. Replace value of slot with random value
2. Replace whole slot with random key and value
3. Delete the slot

The EMC has equal probabilities for all 3 occurrences.

5 Dataset

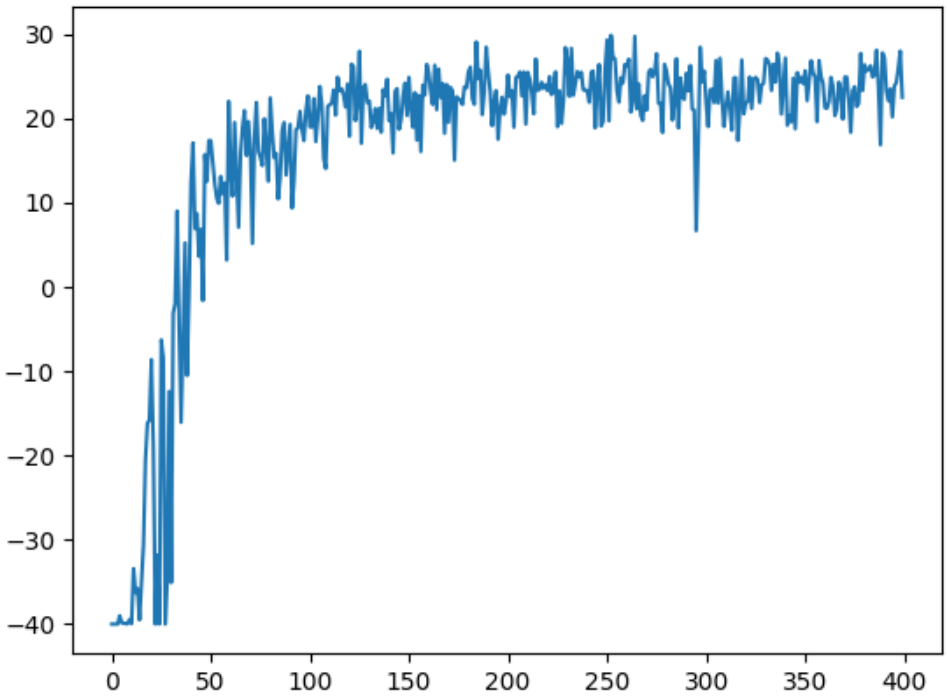
Dataset is collected from 280 actual dialogues with average of 11 turns labelled manually. There are about 29 slots which are mostly inform related and subset is request. There are about 130 user goals in the initial data but this get's augmented in many ways and actual training data differs and can be pretty large number of multiple turns. With Error Model Controller(EMC) and rule based random policy training data is very different and numerous from original user goals.

Three main parts
1. Data dictionary
   {
   'city': ['hamilton', 'manville', 'bridgewater', 'seattle'],
   'theater': ['manville 12 plex', 'amc dine-in theatres bridgewater 7'],
   'genre': ['comedy', 'horror', 'romance']
   }
2. User goal list
   {'intent': 'inform', 'inform_slots': {'moviename': 'the witch'}, 'request_slots': {}}
3. Each action contains an intent, inform and request slots. There are a total of 6 possible intents and 20 request and inform slots

These are all the possible request and inform slots. There can also be 6 different intent slot values inform, request, thanks, match found, reject, done

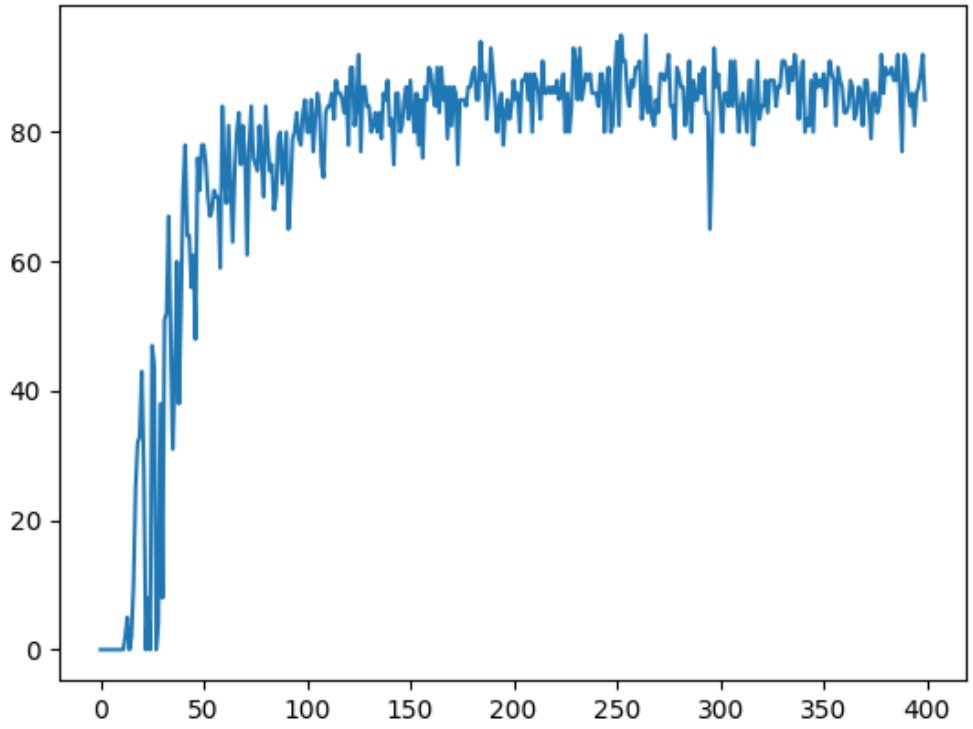
6 Results

6.1 Deep Q networks



Average Reward for every 100 episodes.

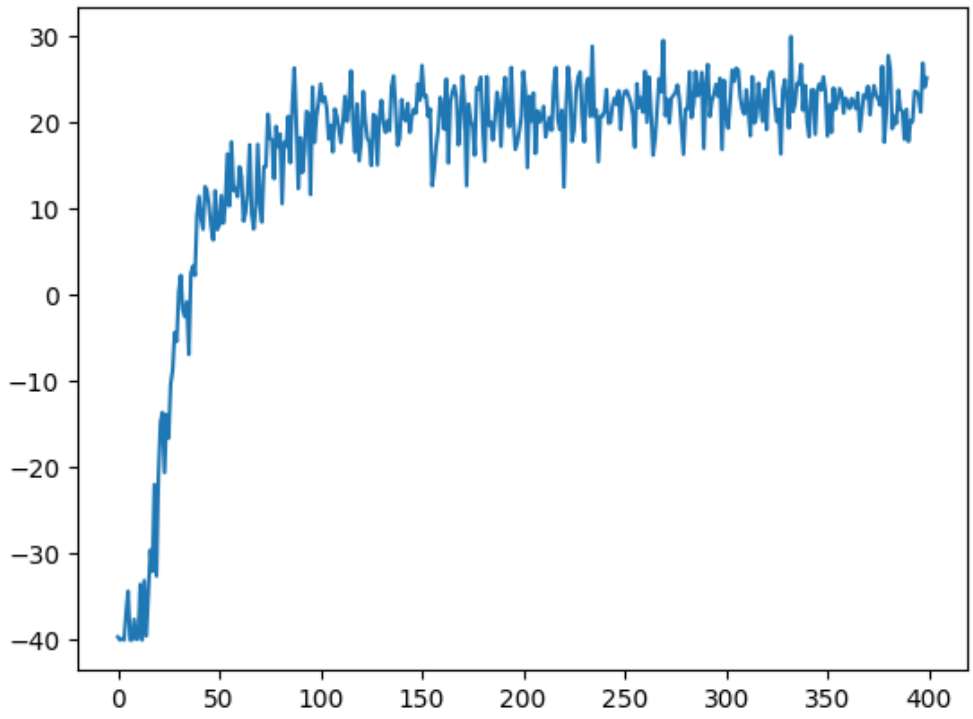
I ran 40,000 epochs and each point on the plot above is an average of every 100 episodes.



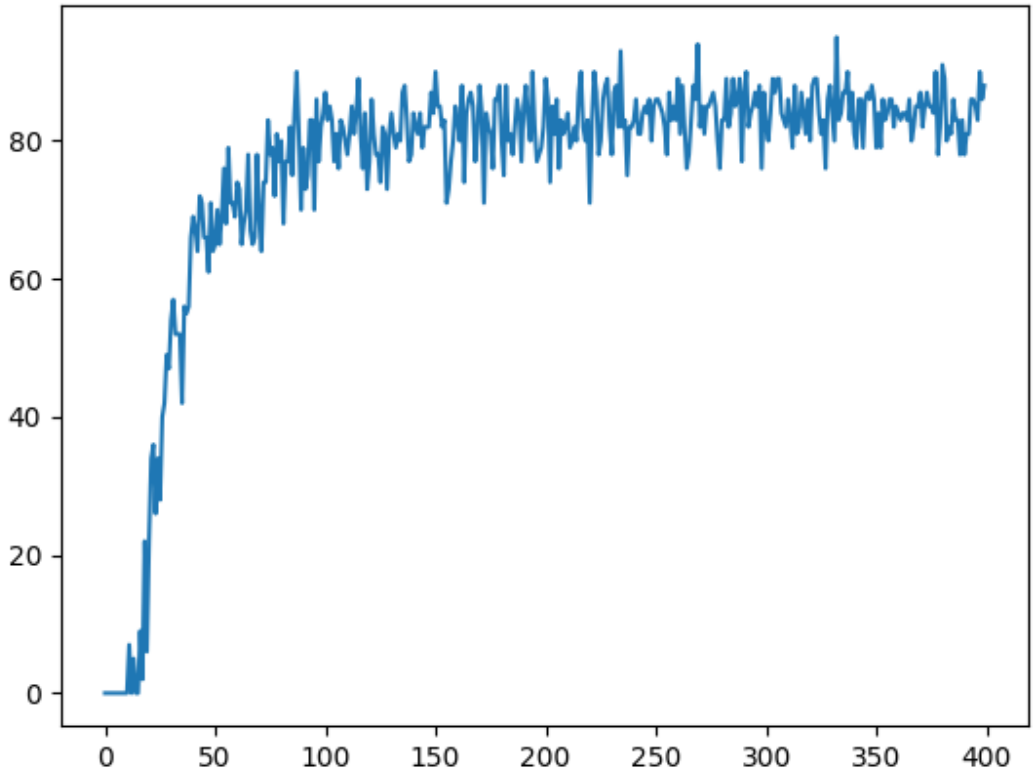
Average Success rate for every 100 episodes.

A success rate is a semantic level check on whether the agent's response was leading towards the goal or not.

6.2 Double Deep Q networks

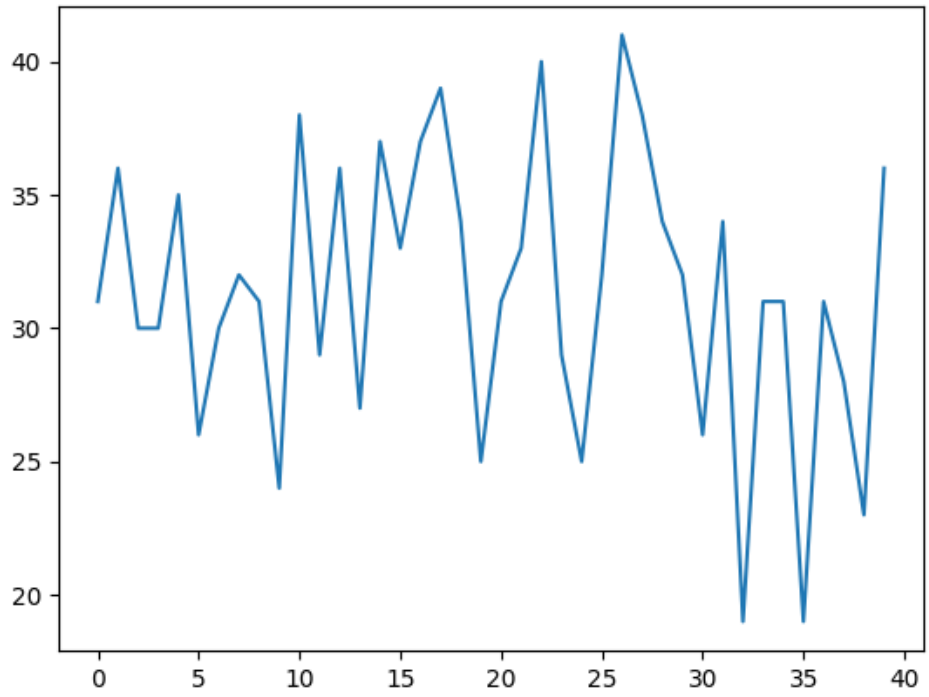


Average Reward for every 100 episodes.

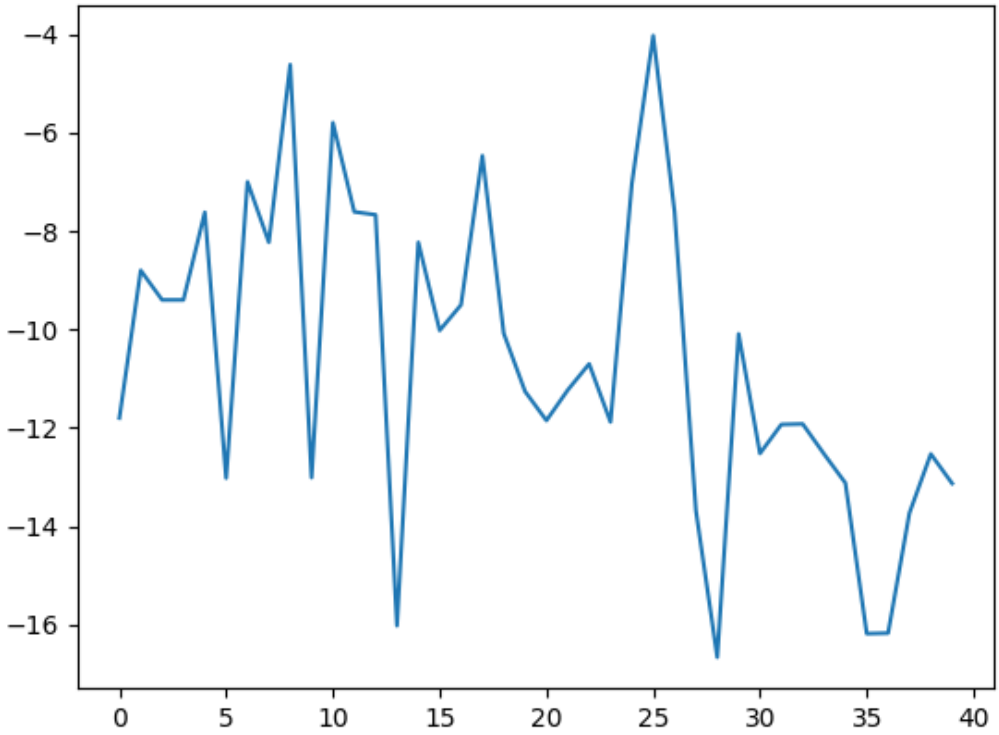


Average Success rate for every 100 episodes.

6.3 Actor-critic



Average Success rate for every 100 episodes.



Average reward for every 100 episodes.

7 Conclusion

I was able to conclude that the double deep q networks did perform better than deep q networks which was the only thing which was implemented in the original paper but actor-critic couldn't stay stable enough or even learn anything to work with.