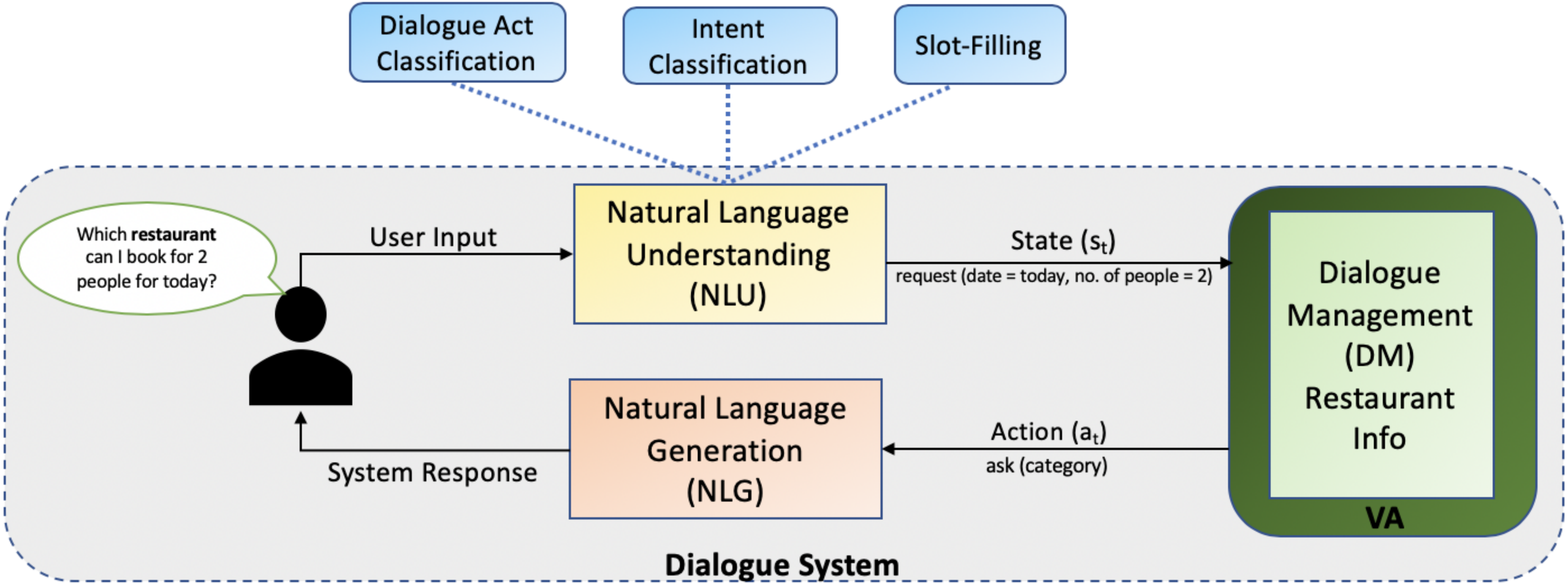
End-to-End Goal Oriented Dialogue Systems

Tulika Saha
PhD Scholar, CSE, AI-NLP-ML Lab
Indian Institute of Technology Patna, India
Under the Supervision of
Dr. Sriparna Saha & Prof. Pushpak Bhattacharyya
tulika.pcs16@iitp.ac.in / sahatulika15@gmail.com


Hello!! How can I help you ??
My Computer isn't working.
....
....
Anytime
Great!!!! that solved my problem

DIALOGUE SYSTEM



REINFORCEMENT LEARNING

REINFORCEMENT LEARNING

- RL allows the machine or software agent to learn its ideal behavior based on feedback from the environment i.e. REWARDS. Belo

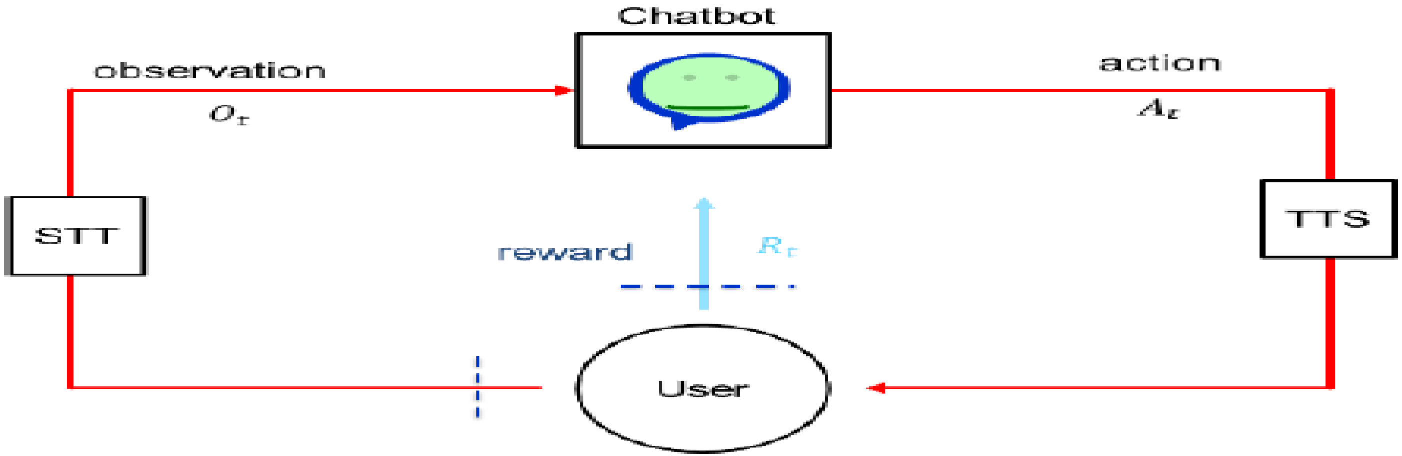
The underlying principle behind RL :-

- Law of Effect :-
  - You do something, something good happens, so you try to do it more often if a similar situation arises.
  - You do something, something bad happens, so you try to do it less often if a similar situation arises.

A RL CHATBOT

- The RL Chatbot takes an action (a_t) at time t which is converted into speech by text-to-speech (TTS) unit
- The user's response, in the form of speech is converted to text by speech-to-text (STT) unit
- The output of the STT unit i.e. the observation (o_t) is received by the chatbot
- The user may either generate a reward signal (r_t) explicitly or it is derived from the user's response
- The task of the RL Chatbot is to respond in a way so as to maximize it's reward

A RL CHATBOT



WHY SEQUENTIAL DECISION MAKING ?

- Limited Supervision : you know **what** you want but not **how** to get it
- Actions have consequences
- Unlike classification, regression

REINFORCEMENT LEARNING

- Thus, any RL setup is mathematically described in terms of an abstraction called the Markov Decision Process (MDP)
- A MDP model is described by M = < S; A; R; P; Rho >, where
  - S is the finite state space
  - A is the finite action space (corresponding to VA’s action)
  - R is the reward model
  - P is the transition model
  - Rho is the initial state distribution
- The idea is to implement RL for learning dialogue strategies using the Q-learning algorithm
  - The main idea in Q-learning is that we can iteratively approximate the Q-function using the Bellman Equation (via Dynamic programming but still slow)
  - Hence, the Q-function will be approximated using a neural network (incorporation of Deep RL)

Q LEARNING

- Q-learning defines a function *Q(s,a)* which represents the **discounted future reward** when we perform action *a* in state *s*, and continue optimally from that point on.
- The way to think about *Q(s,a)* is that it is “the best possible score at the end of an episode after performing action *a* in state *s*”
- It is called Q-function, because it represents the “quality” of certain action in given state.
- If you want to select the action that results in the highest score at the end of an episode
    - Pick the action with the highest Q-value : $\pi(s)=argmax_a Q(s,a)$
- The main idea in Q-learning is that we can iteratively approximate the Q-function using the Bellman equation
    - $Q(s,a) = r + \gamma \; max_{a'} Q(s',a')$
    - Here γ is the discount factor between 0 and 1 – the more into the future the reward is, the less we take it into consideration

Deep Q Network (DQN)

- In DRL, the Q-function is implemented and approximated using a neural network,
  - i.e., by acting as a function approximator, we can take any number of possible states that can be represented as a vector and learn to map them to Q-values.
- The Q-function of a DQN agent is parameterized as $Q(s,a;\theta_i)$, where θ_i s are the parameters or the weights of the neural network at iteration ***i***.
  - A Q-network can be trained by minimizing a sequence of loss functions, $L(\theta_i)$, that get updated at each iteration ***i*** as follows :

$$L(\theta_i) = E_{MB}[(r + \gamma * max_{\ a'} Q(s', a'; \bar{\theta}_i) - Q(s, a; \theta_i))]$$

where θ_i s are the parameters of the neural network at iteration ***i***, $\bar{\theta}_i$ s are the target parameters of the neural network at iteration ***i*** and **MB**s are the mini-batches of experience.

DOUBLE DEEP Q NETWORK (DDQN)

- One issue with the DQN algorithm is that it is more likely to overestimate the Q function values due to the ***max*** in Equation used to set targets.
- Thus, the action with the maximal positive error is selected and this value is consequently propagated further to the next states.
  - This leads to positive bias value overestimation which severely impacts the stability of the learning algorithm.
- The DDQN algorithm is said to overcome this particular problem. In this algorithm, two Q functions Q_1 and Q_2 are independently learned.
  - One function is then used to determine the maximizing action and second to estimate its value. Either Q_1 or Q_2 is updated randomly as :

$$Q_1(s,a) = r + \gamma * Q_2(s', argmax_{a'} Q_1(s',a'))$$

$$Q_2(s,a) = r + \gamma * Q_1(s', argmax_{a'} Q_2(s',a'))$$

- Hence, it was proven that by decoupling the maximizing action from

DDQN with Priority Experience Replay (DDQN-PER)

- Another concept that is acknowledged in the literature is how the experiences are used for training.
- When all the samples are treated alike, one of the aspects which gets ignored is that learning happens more from some transitions than from others.
- PER is one such strategy that tries to leverage this fact by changing the sampling distribution.
- The main idea is that those transitions are preferred that do not fit well to the current estimate of the Q function, because these are some transitions from which maximum learning can happen.

DIALOGUE MANAGEMENT STRATEGY

DIALOGUE MANAGEMENT STRATEGY

- The main function of the dialogue management component is to control the flow of the dialogue. This involves the following task:
  - Determining whether sufficient information has been elicited from the user in order to take a suitable action
  - Execute an action
    - Communicate with an application
    - Communicate with the user

PROBLEM STATEMENT

- Developing a Dialogue Management Strategy for Task-Oriented Virtual Agent
- Using Reinforcement Learning (Deep RL)

DOMAIN - ‘FLIGHT’ ENQUIRY

Dataset used : Airline Travel Information System (ATIS)

Task : The Virtual Agent (VA) need to interact with a user so as to gather information from him/her regarding the details of flight that he/she wishes to book.

Slots to be filled for the information system :

‘deptCity’ : The city of departure eg. Washington
‘arrCity’ : The arrival city eg. Baltimore
‘deptDay’ : The date of departure eg. fourth july
‘deptTime’ : The time of departure eg. afternoon, 4 pm
‘class’ : The class of flight for eg. economy, business

PROPOSED MDP

State : a tuple of 5 variables

[deptCity, arrCity, deptDay, deptTime, class]

- The variables correspond to confidence scores for each slot outputted from NLU module
- Permissible range is from 0 to 1

Action : 14 possible actions are there with the agent as follows :

- **Salutation :** closing_conversation
- **Ask :** askdeptCity, askarrCity, askdeptTime, askdepDay, askclass
- **Reask/Confirm :** reaskdeptCity, reaskarrCity, reaskdeptTime, reaskdepDay, reaskclass
- **Hybrid Actions :** askDeptandArr, askDateTime

REWARD MODEL

Different reward function at different time-step/ for different actions in conversation

- For any action except terminating action

$$R(s,a,s') = (w_1 * (||\overrightarrow{NS}||_1 - ||\overrightarrow{CS}||_1) - w_2)$$

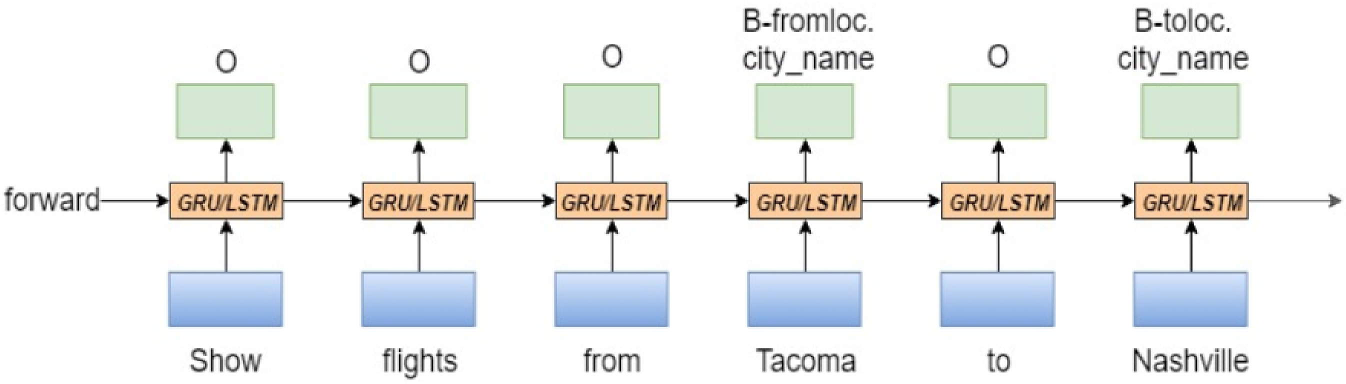
  - $||\overrightarrow{NS}||_1$ state is the summation of the confidence scores of all state variables in the state vector ‘s`’ obtained after taking an action ‘a’ in state ‘s’.
  - $||\overrightarrow{CS}||_1$ state is the summation of confidence scores for all state variables in state vector ‘s’
  - w_1 is used to encourage the agent to act in a way as to increase confidence on the acquired slots
  - w_2 is used to discourage unnecessary questions and iterations with the user.
  - w_2 = 1 and w_1 = 8

REWARD MODEL

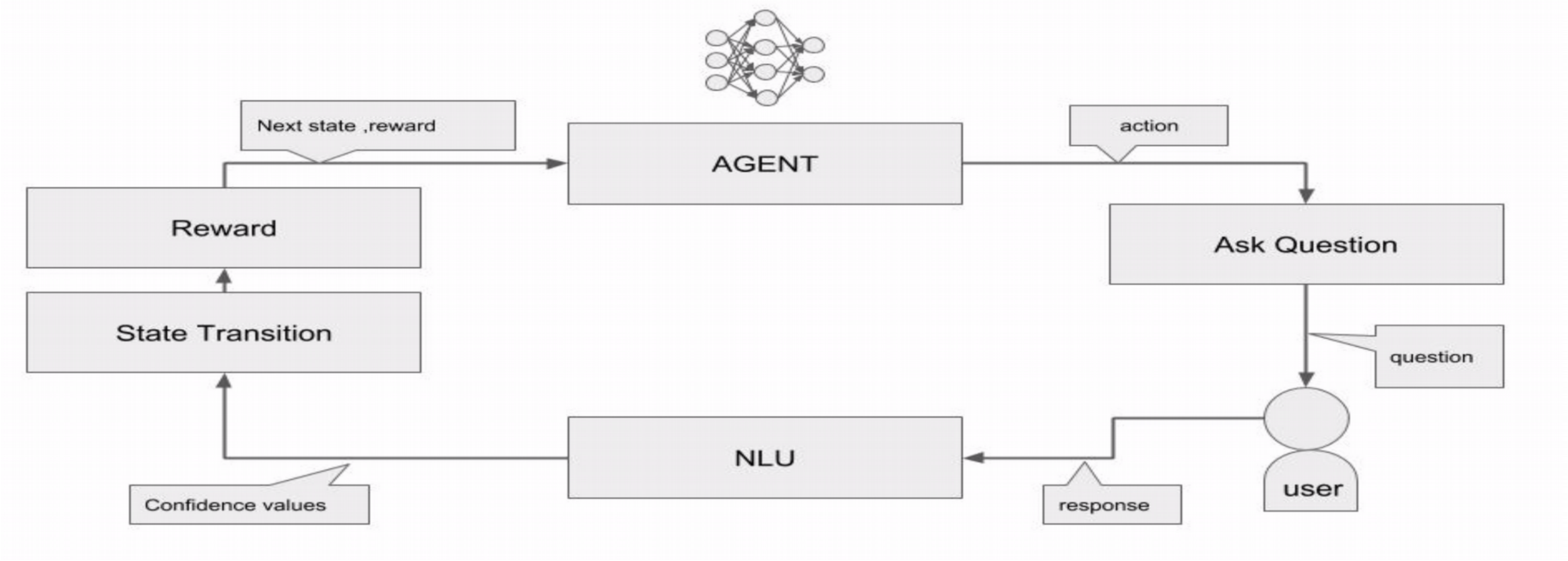
- For terminating action,
  - The **condition** is that how many slots has a confidence score above a threshold say 0.7, the condition is satisfied when all slots match the criteria.
  - If the condition is not satisfied , the reward function is :
  - $$R(s, a, s') = -w_1 * (||\overrightarrow{EV}||_1 - ||\overrightarrow{CS}||_1)$$
  - $||\overrightarrow{EV}||_1$ is the summation of maximum expected confidence scores of different slots that adds up to 5.
  - If the condition is satisfied :
  - $$R(s, a, s') = w_1 * ||\overrightarrow{CS}||_1$$

NLU MODULE

- LSTM module is used for the NLU part.
- The task of the NLU is to take in a **sentence** and give out the **slots identified** from the sentence.
- We extract the slot identified and the probability of that slot from the model



FLOW DIAGRAM OF THE SYSTEM



METRICS TO EVALUATE THE PERFORMANCE OF VA

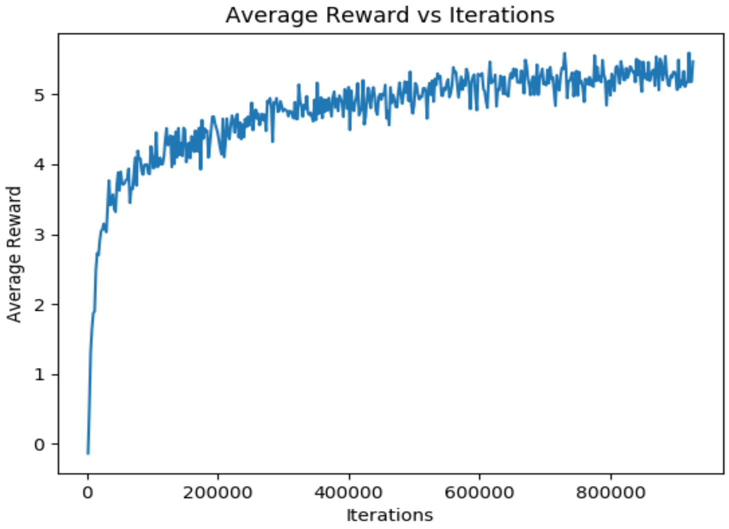
- **Learning Curve during training** : This gives a visual representation of the learning pattern and growth of the VA during training.
- **Training Time** : It gives an estimate of the computational requirement of different VAs.
- **Average Episodic Reward** : It is the average cumulative reward through all the time-steps at the end of a dialogue.
    - Higher the episodic reward, better is the chance of task-completion.
- **Average Dialogue Length** : It is basically the average system actions per dialogue.
    - The VA should be able to complete its task in less number of time-steps.

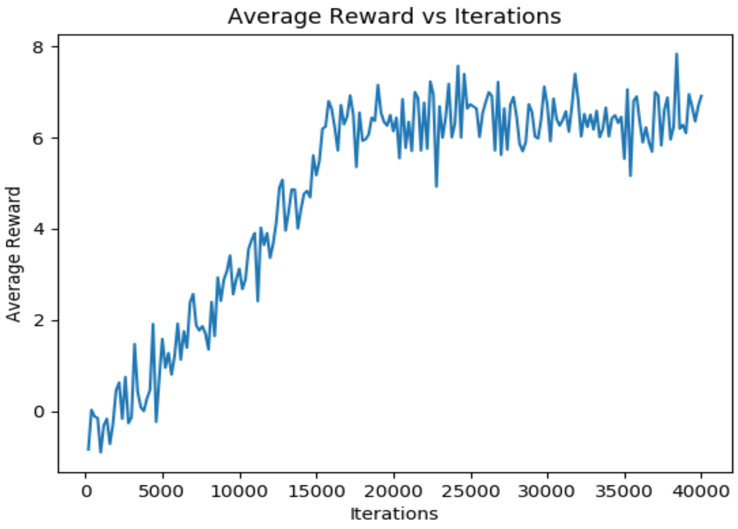
RESULTS

- Many variations of the Q learning approach were used the best results were obtained from Double Deep Q- Network using Prioritized Experience Replay

	Algorithm	Average Episodic Reward	Average Dialogue Length	Training Time (in hrs)
Simple Reward Model	DQN with SVM	-6.89 $\pm$ 5.62	673.45 $\pm$ 564.02	71.97
	DDQN with SVM	-8.51 $\pm$ 5.40	791.65 $\pm$ 529.12	93.65
	DDQN-PER with SVM	-13.51 $\pm$ 9.15	1342.3 $\pm$ 915.20	52.56
	DDQN-PER	-11.26 $\pm$ 9.17	1039.07 $\pm$ 933.62	112.12
Proposed Reward Model	DQN with SVM	-313.25 $\pm$ 308.63	367.52 $\pm$ 315.03	40.71
	DDQN with SVM	-273.52 $\pm$ 271.97	330.52 $\pm$ 278.67	54.2
	DQN-PER with SVM	-131.80 $\pm$ 181.13	183.2 $\pm$ 182.07	18.47
	DQN-PER	-569.85 $\pm$ 469.48	589.03 $\pm$ 479.09	16.83
	DDQN-PER with SVM	**57.20 $\pm$ 7.99**	**7.67 $\pm$ 0.53**	**20.82**
	DDQN-PER	**50.07 $\pm$ 8.11**	**8.09 $\pm$ 1.06**	**17.74**

Learning Curves of the Virtual Agent





Left : shows the learning curve for the Q-table implementation
Right : shows the learning curve for DDQN-PER algorithm

SAMPLE CONVERSATION

```
State : [0. 0. 0. 0. 0.]
Agent > Hello How may I help you?
User  > flights from pittsburgh to denver on fifteenth june at afternoon
State : [1. 1. 0.558 1. 0.]
Agent[askclass] > Please specify the class of flight?
User  > business class
State : [1. 1. 0.558  1. 1.]
Agent[reaskdepDay] > Are you travelling on fifteenth june?
User  > yes
State : [1. 1. 1. 1. 1.]
Agent[closing_conversation] > The flights from pittsburgh to denver
on fifteenth june at afternoon via business class are JW345, US298
Thanks for using the flight attendant.
```

Dialogue 1 (Reasks the dept Date when unsure)

Dialogue 2

```
State : [0. 0. 0. 0. 0.]
Agent > Hello How may I help you?
User  > I want to travel from pittsburgh to denver
State : [1. 1. 0. 0. 0.]
Agent[askDateTime] > Specify the time and date?
User  > Need to travel on twenty eight december at evening
State : [1.    1.    0.97  0.994 0.   ]
Agent[askclass] > Please specify the class of flight?
User  > I want to travel in economy class
State : [1.    1.    0.97  0.994 0.968]
Agent[closing_conversation] > The flights from pittsburgh to denver
on twenty eight december at evening via economy class are TG564, GF298.
Thanks for using the flight attendant.
```

For further reading,

Saha, T., Gupta, D., Saha, S. & Bhattacharyya, P. (2018). Reinforcement learning based dialogue management strategy. **In Proceedings of International Conference on Neural Information Processing (pp. 359–372). Springer**

Code : https://github.com/sahatulika15/RLDS_DEMO

THANK YOU !