Intelligent Conversational Bot

Policy Optimization (2)

Apr 25th, 2017

YUN-NUNG (VIVIAN) CHEN

WWW.CSIE.NTU.EDU.TW/~YVCHEN/S105-ICB

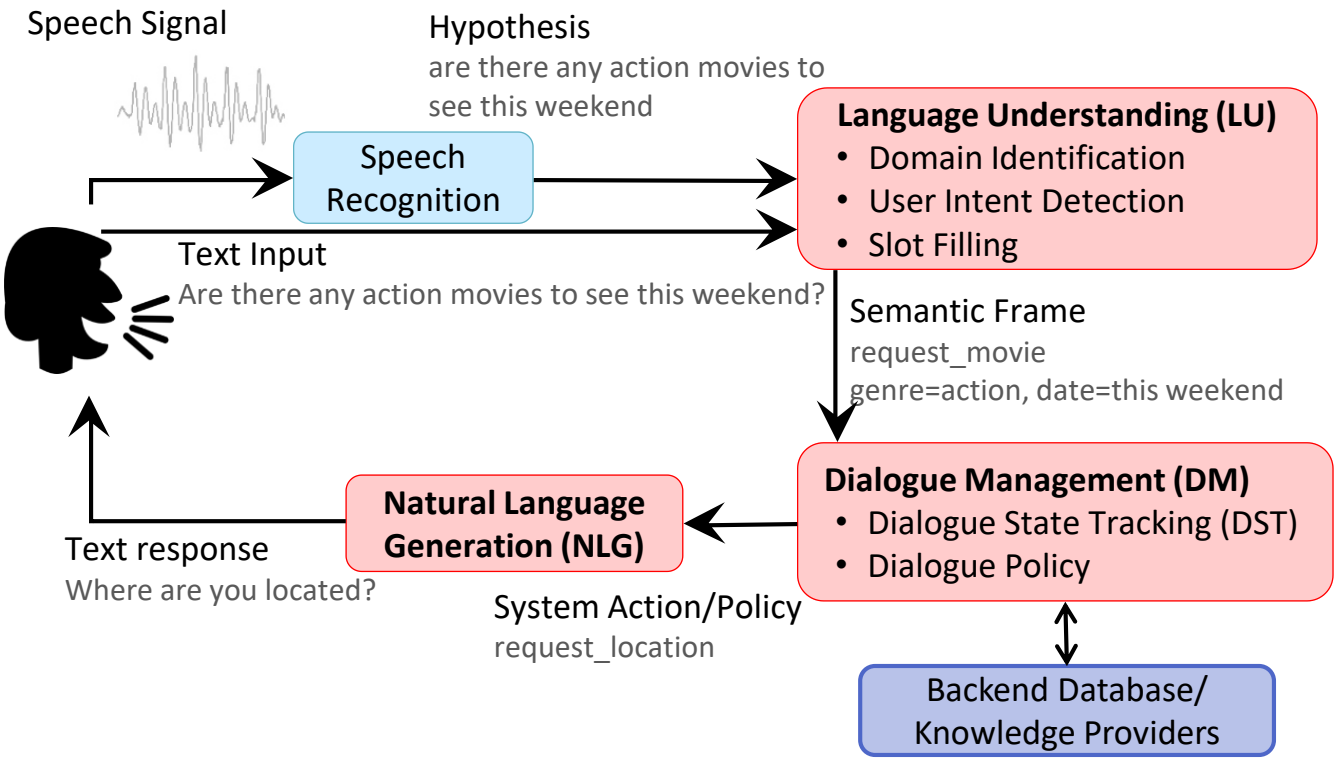
國立臺灣大學
National Taiwan University

Slides credit from Gašić

Review

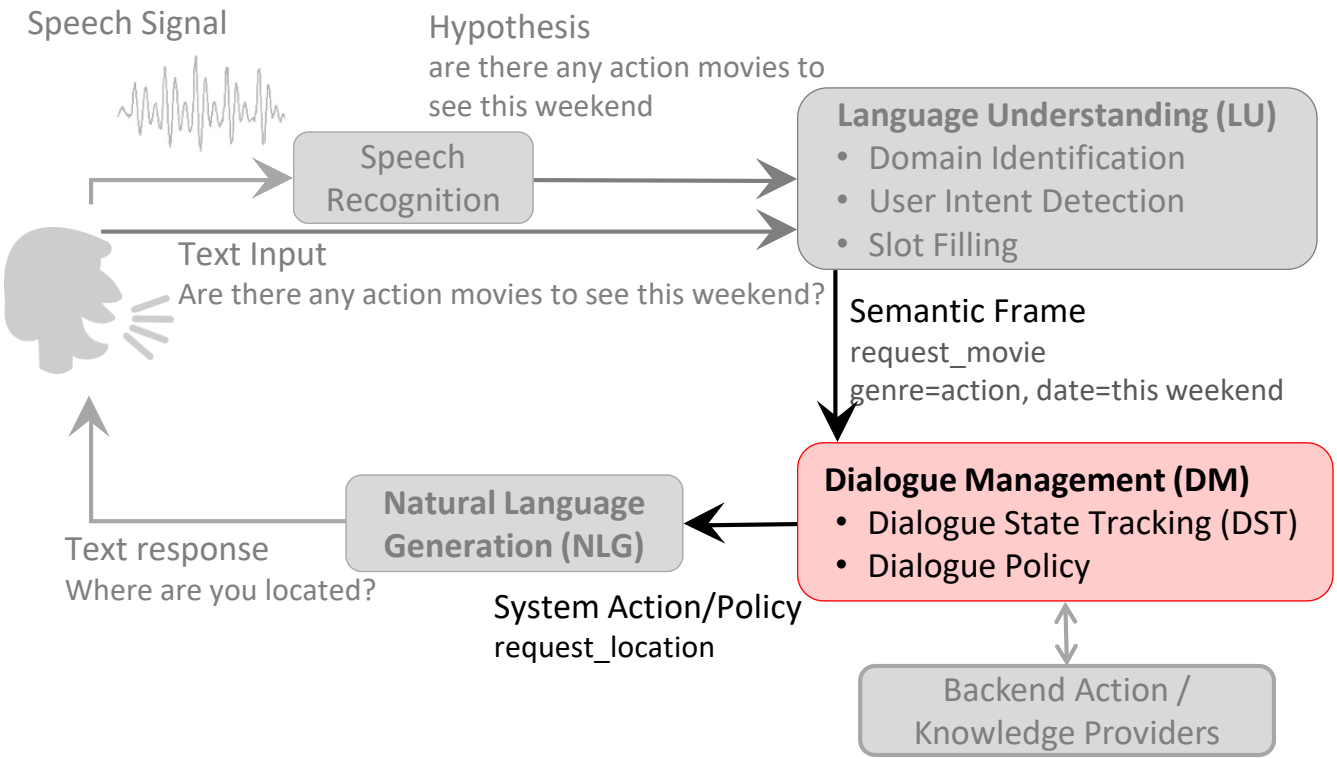
Task-Oriented Dialogue System (Young, 2000)

http://rsta.royalsocietypublishing.org/content/358/1769/1389.short



Task-Oriented Dialogue System (Young, 2000)

http://rsta.royalsocietypublishing.org/content/358/1769/1389.short



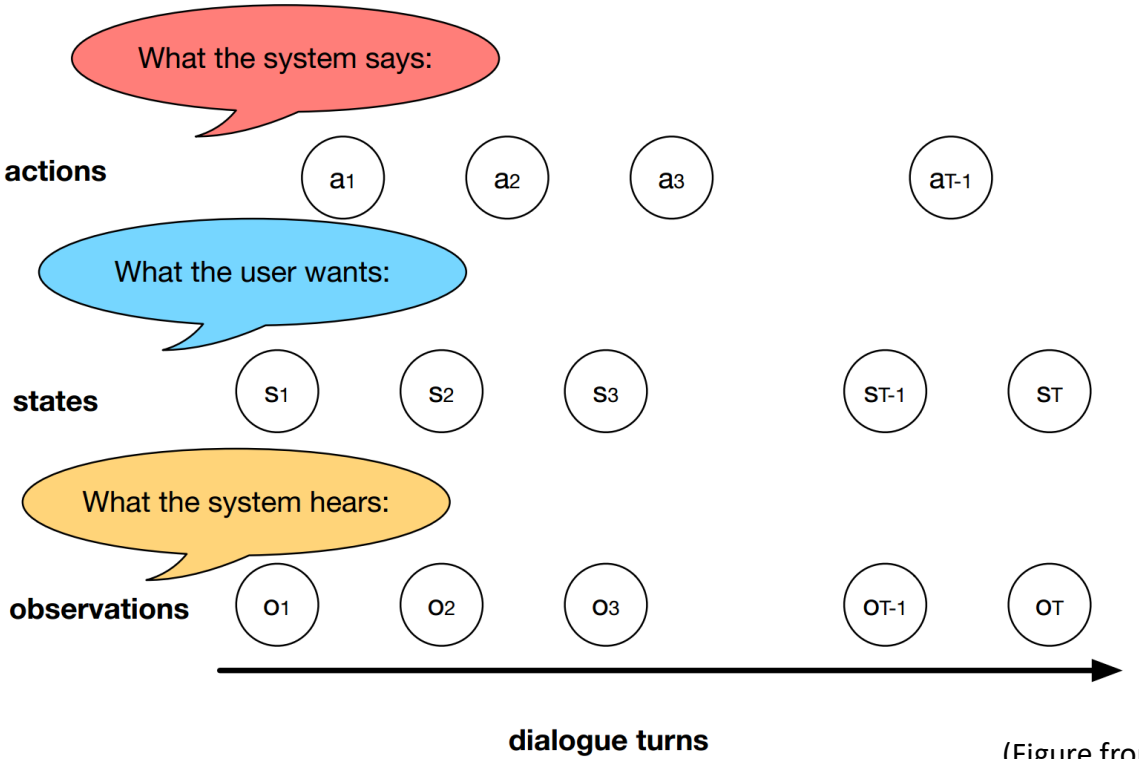
Dialogue Management

Example Dialogue

Hello, how may I help you?	greeting ()
I'm looking for a Thai restaurant.	request (restaurant; foodtype=Thai)
What part of town do you have in mind?	request (area)
Something in the centre.	inform (area=centre)
Bangkok city is a nice place, it is in the centre of town and it serves Thai food.	inform (restaurant=Bangkok city, area=centre of town, foodtype=Thai)
What's the address?	request (address)
Bangkok city is a nice place, their address is 24 Green street.	inform (address=24 Green street)
Thank you, bye.	bye ()

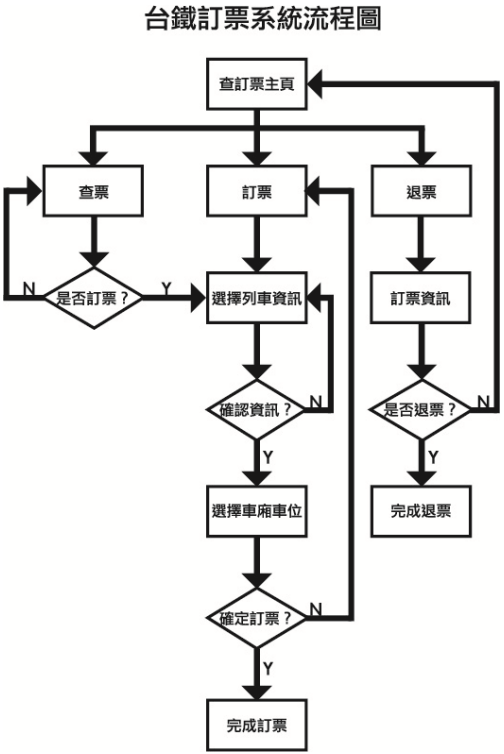
Example Dialogue

Elements of Dialogue Management

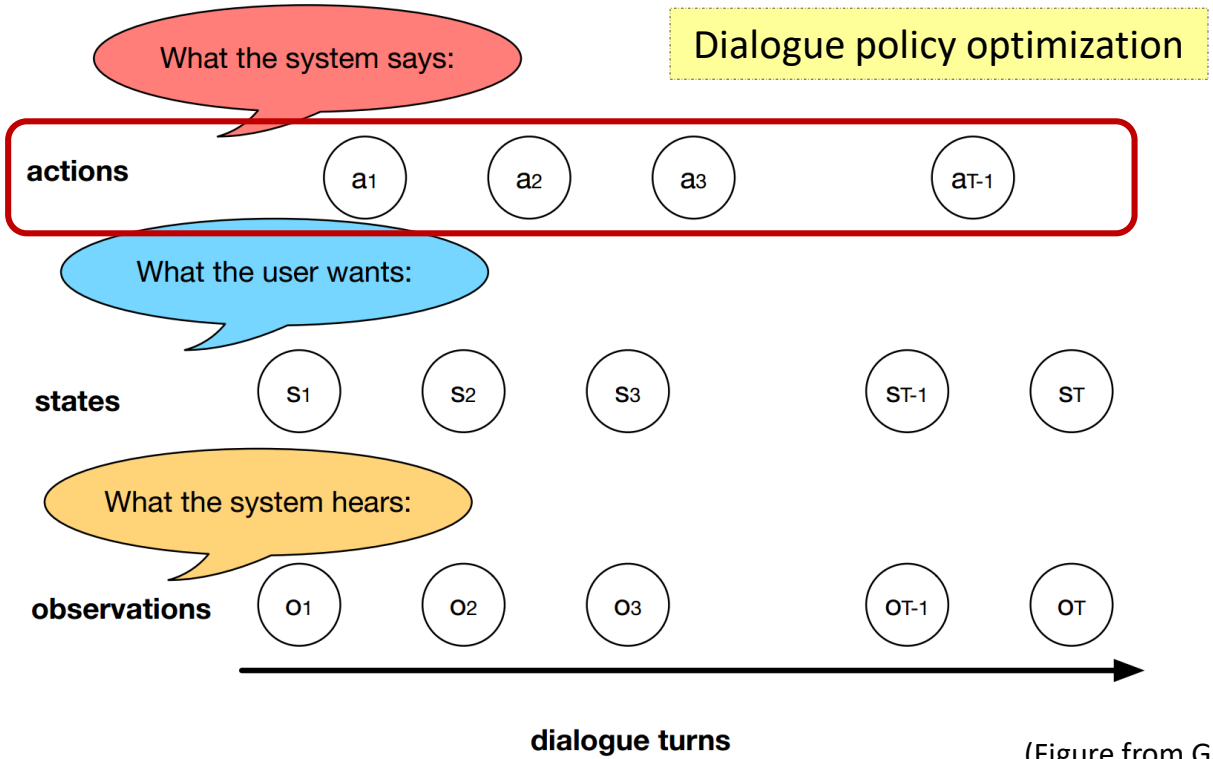


(Figure from Gašić)

Rule-Based Management



Elements of Dialogue Management



(Figure from Gašić)

Dialogue Policy Optimization

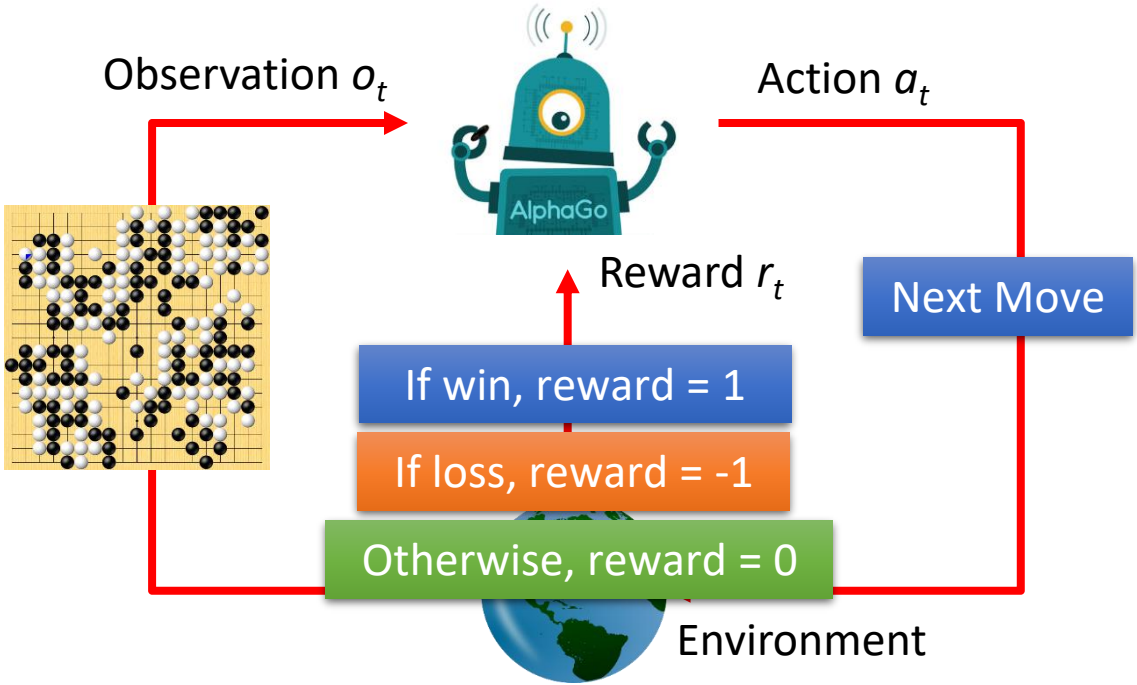
Reinforcement Learning

Reinforcement Learning

- RL is a general purpose framework for **decision making**
  - RL is for an *agent* with the capacity to *act*
  - Each *action* influences the agent's future *state*
  - Success is measured by a scalar *reward* signal
  - Goal: *select actions to maximize future reward*

Big three: action, state, reward

Scenario of Reinforcement Learning



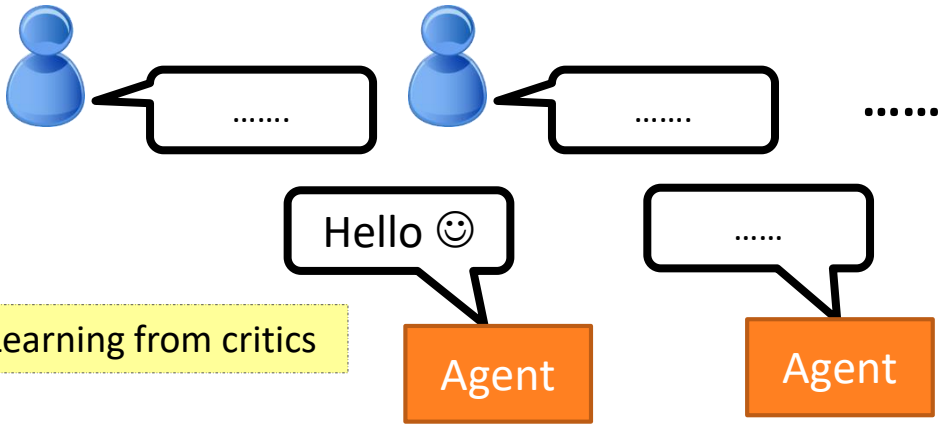
Agent learns to take actions to maximize expected reward.

Supervised v.s. Reinforcement

- Supervised

Learning from teacher

- Reinforcement



Bad

Learning from critics

Dialogue as Reinforcement Learning

- Problems in solving dialogue as an RL task
  1) Optimization problem size
     - Belief dialogue state space is large and continuous
     - System action space is large

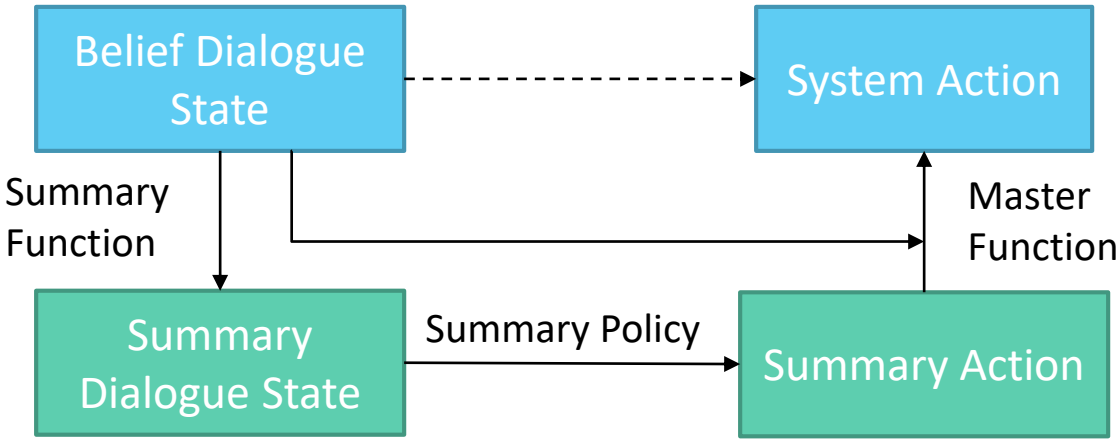
  Solution: learn in reduced summary space

  2) Knowledge environment (user)
     - Transition probability is unknown (user status)
     - How to get rewards
  3) RL takes long time to converge

  Solution: learn in interaction with a simulated user

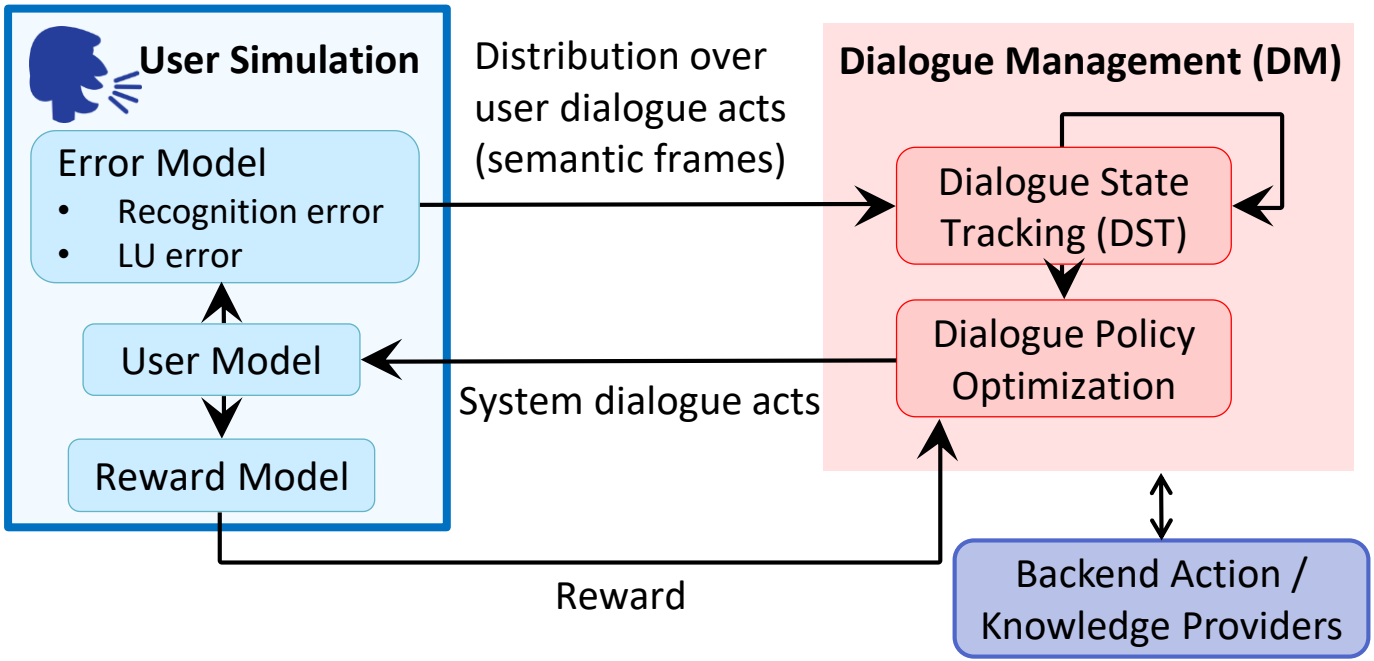
Large Belief Space and Action Space

- Solution: perform optimization in a reduced summary space built according to the heuristics

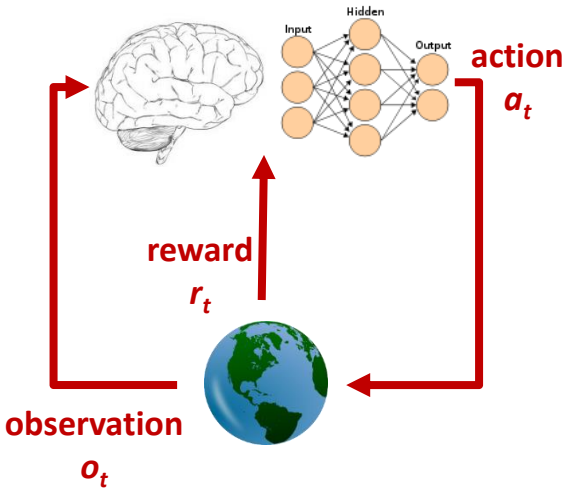


Transition Probability and Rewards

- Solution: learn from a simulated user



Agent and Environment



- At time step t
  - The agent
    - Executes action a_t
    - Receives observation o_t
    - Receives scalar reward r_t
  - The environment
    - Receives action a_t
    - Emits observation o_{t+1}
    - Emits scalar reward r_{t+1}
  - t increments at env. step

State

- Experience is the sequence of observations, actions, rewards

$$o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t$$

- State is the information used to determine what happens next
  - what happens depends on the history experience
    - The agent selects actions
    - The environment selects observations/rewards
- The state is the function of the history experience

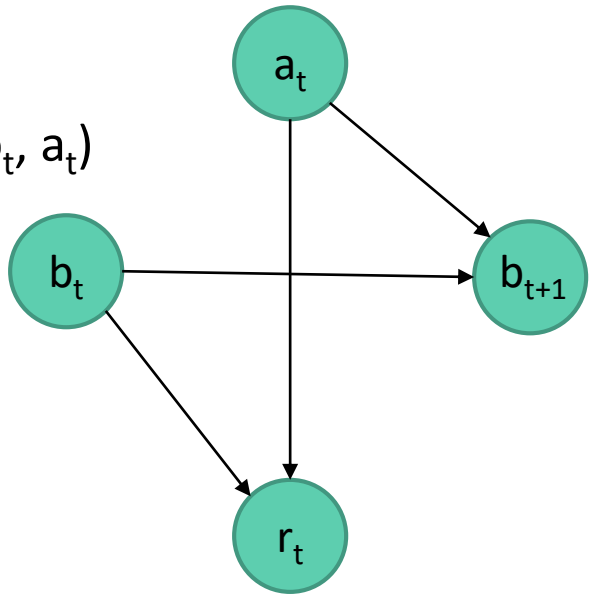
$$s_t = f(o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t)$$

POMDP Policy Optimization

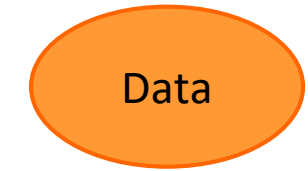
- Finding value function associated with optimal policy, i.e. the one that generates maximal return
  - Problem: tractable only for very simple cases (Kaelbling et al., 1998)
  - Alternative solution: discrete space POMDPs can be viewed as a continuous space MDP with states as belief states $b_t = b(s_t)$

Markov Decision Process (MDP)

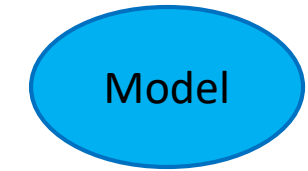
- Belief state from tracking: $b_t = s_t$
- System actions: a_t
- Rewards: r_t
- Transition probability: $p(b_{t+1} | b_t, a_t)$



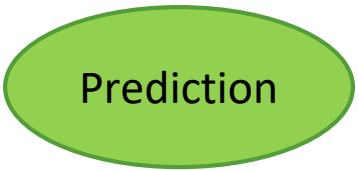
DM as Markov Decision Process (MDP)



- Belief dialogue states (continuous)
- Reward – a measure of dialogue quality



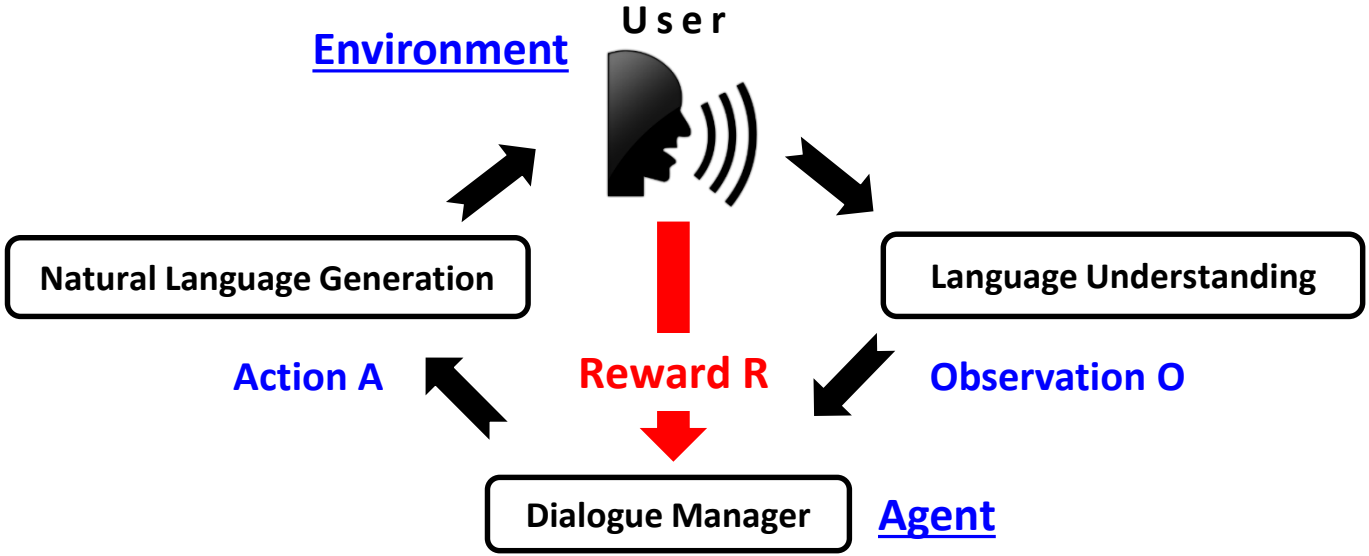
- Markov decision process (MDP) & reinforcement learning



- System actions – Dialogue Policy Optimization

Dialogue Policy Optimization

- Dialogue management in a RL framework



The optimized dialogue policy selects the best action that maximizes the future reward. Correct rewards are a crucial factor in dialogue policy training

Reward

- Reinforcement learning is based on reward hypothesis
- A reward r_t is a scalar feedback signal
  - Indicates how well agent is doing at step t

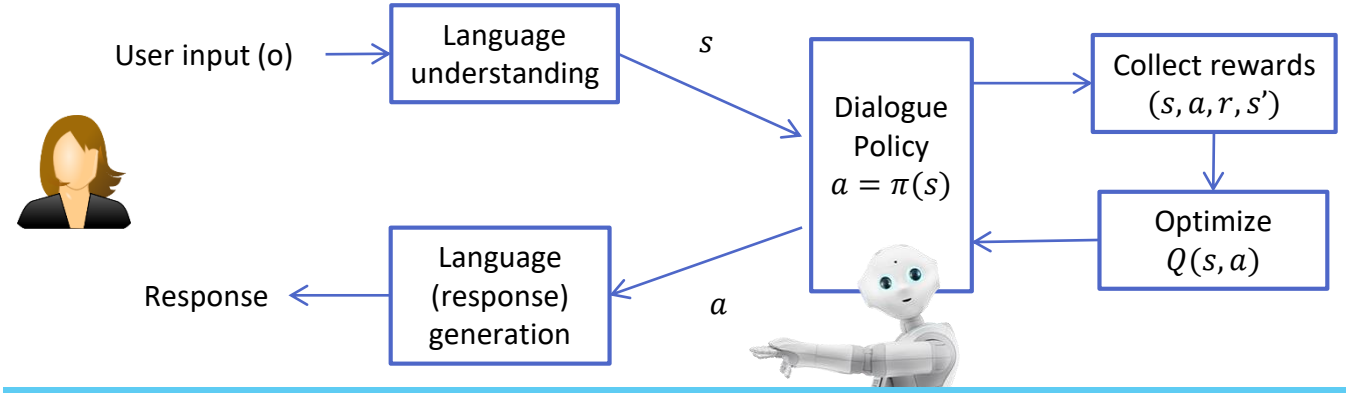
Reward hypothesis: all agent goals can be desired by maximizing expected cumulative reward

Reward for RL ≅ Evaluation for System

- Dialogue is a special RL task
  - Human involves in **interaction** and **rating (evaluation)** of a dialogue
  - Fully human-in-the-loop framework
- Rating: correctness, appropriateness, and adequacy

- Expert rating	high quality, high cost
- User rating	unreliable quality, medium cost
- Objective rating	Check desired aspects, low cost

Reinforcement Learning for Dialogue Policy Optimization



Type of Bots	State	Action	Reward
Social ChatBots	Chat history	System Response	# of turns maximized; Intrinsically motivated reward
InfoBots (interactive Q/A)	User current question + Context	Answers to current question	Relevance of answer; # of turns minimized
Task-Completion Bots	User current input + Context	System dialogue act w/ slot value (or API calls)	Task success rate; # of turns minimized

Goal: develop a generic deep RL algorithm to learn dialogue policy for all bot categories

Dialogue Reinforcement Learning Signal

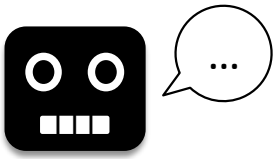
Typical reward function

- -1 for per turn penalty
- Large reward at completion if successful

Typically requires domain knowledge

- ✔ Simulated user
- ✔ Paid users (Amazon Mechanical Turk)
- ✖ Real users

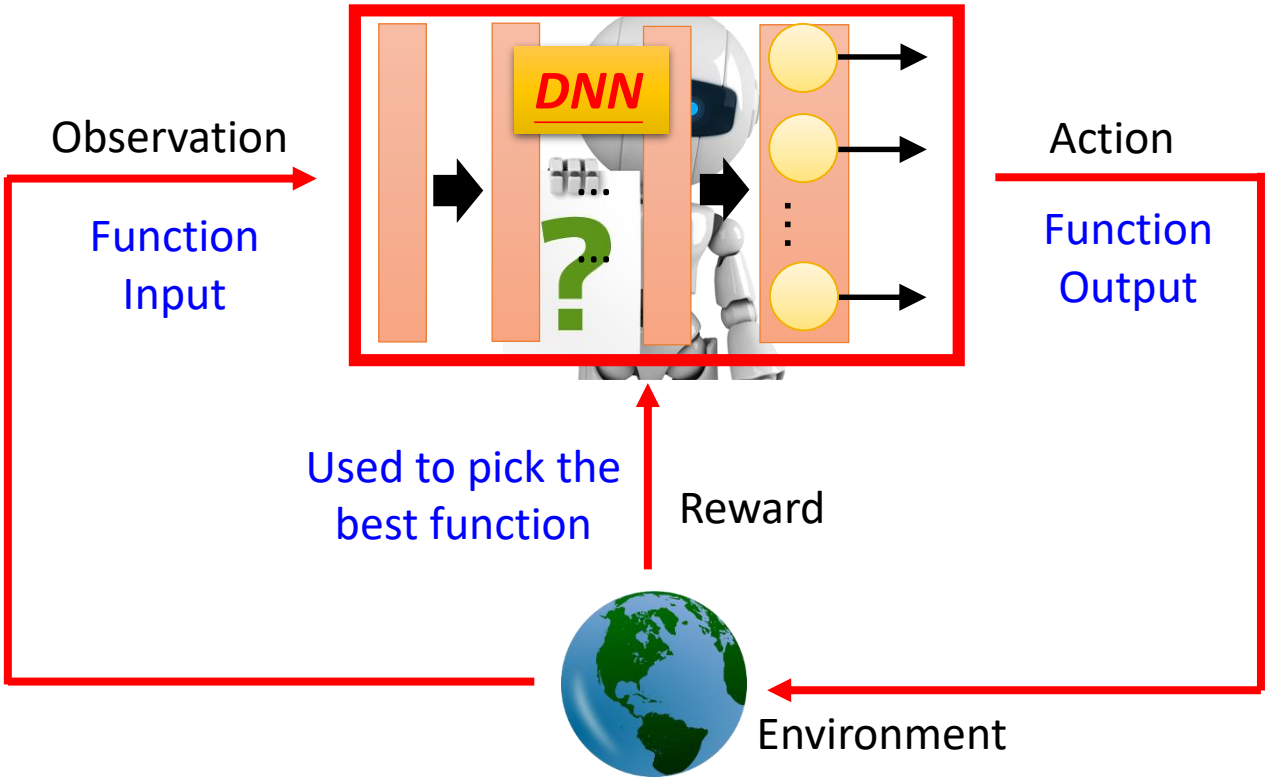
The user simulator is usually required for dialogue system training before deployment



Sequential Decision Making

- Goal: select actions to maximize total future reward
  - Actions may have long-term consequences
  - Reward may be delayed
  - It may be better to sacrifice immediate reward to gain more long-term reward

Deep Reinforcement Learning



Reinforcement Learning Approach

Value-Based

Policy-Based

Model-Based

Major Components in an RL Agent

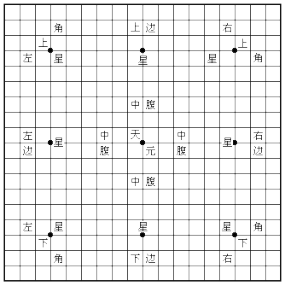
- An RL agent may include one or more of these components
  - **Policy**: agent's behavior function
  - **Value function**: how good is each state and/or action
  - **Model**: agent's representation of the environment

Policy

- A policy is the agent's behavior
- A policy maps from state to action
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a) = P(a \mid s)$

Value Function

- A value function is a prediction of future reward (with action *a* in state *s*)
- Q-value function gives expected total reward
  - from state s and action a
  - under policy π
  - with discount factor γ



$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a]$$

- Value functions decompose into a Bellman equation

$$Q^{\pi}(s,a) = \mathbb{E}_{s',a'}[r + \gamma Q^{\pi}(s',a') \mid s, a]$$

Optimal Value Function

- An optimal value function
  - is the maximum achievable value

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) = Q^{\pi^*}(s,a)$$

  - allows us to act optimally

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

  - informally maximizes over all decisions

$$\begin{aligned} Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \ldots \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

  - decompose into a Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

Reinforcement Learning Approach

- Policy-based RL
  - Search directly for optimal policy π^*

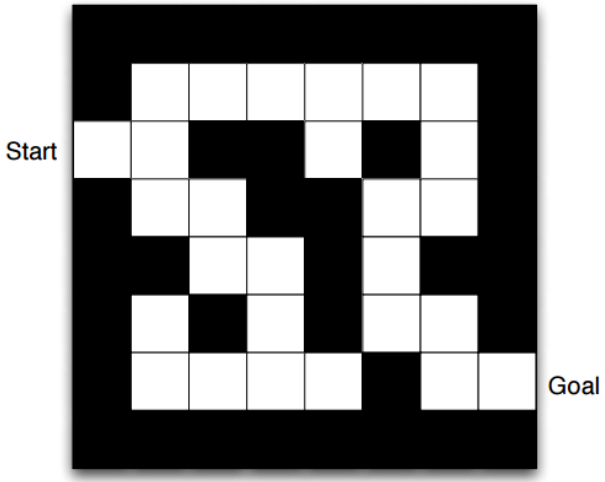
π^* is the policy achieving maximum future reward

- Value-based RL
  - Estimate the optimal value function $Q^*(s, a)$

$Q^*(s, a)$ is maximum value achievable under any policy

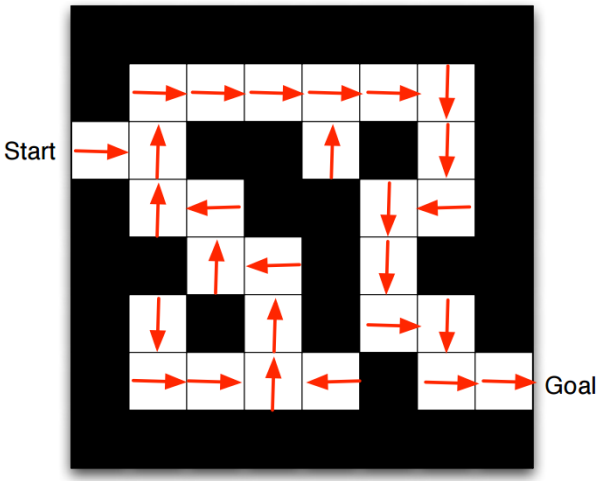
- Model-based RL
  - Build a model of the environment
  - Plan (e.g. by lookahead) using model

Maze Example



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: agent's location

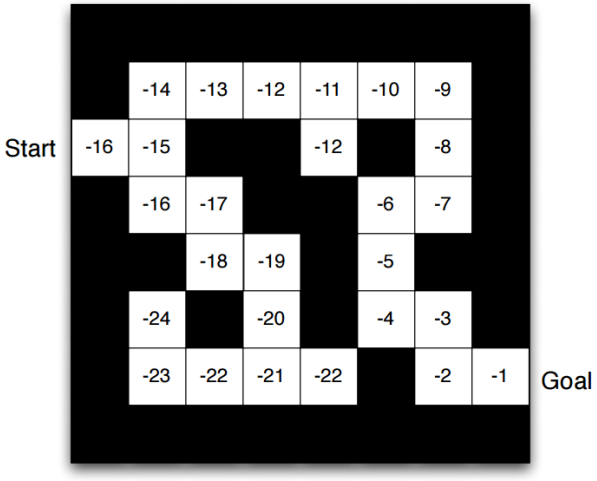
Maze Example: Policy



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: agent's location

Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function



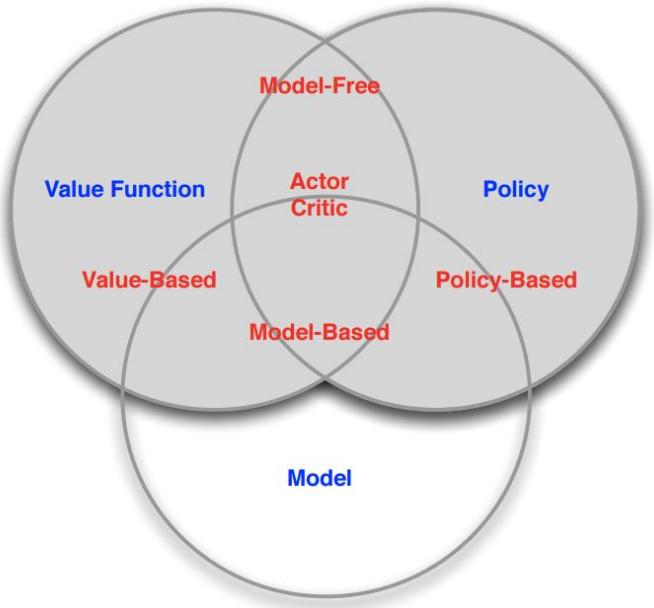
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: agent's location

Numbers represent value $Q_\pi(s)$ of each state s

Categorizing RL Agents

- Value-Based
  - No Policy (implicit)
  - Value Function
- Policy-Based
  - Policy
  - No Value Function
- Actor-Critic
  - Policy
  - Value Function
- Model-Free
  - Policy and/or Value Function
  - No Model
- Model-Based
  - Policy and/or Value Function
  - Model

RL Agent Taxonomy



Value-Based Deep RL

Dynamic Programming

Monte-Carlo

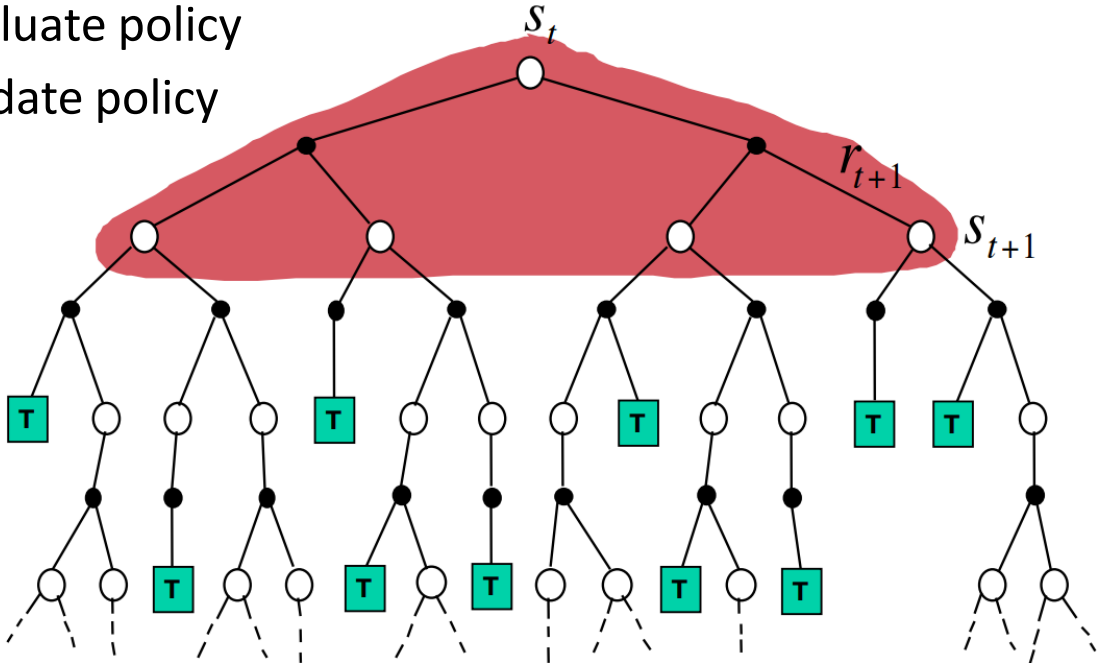
Temporal-Difference

Q-Learning

Dynamic Programming

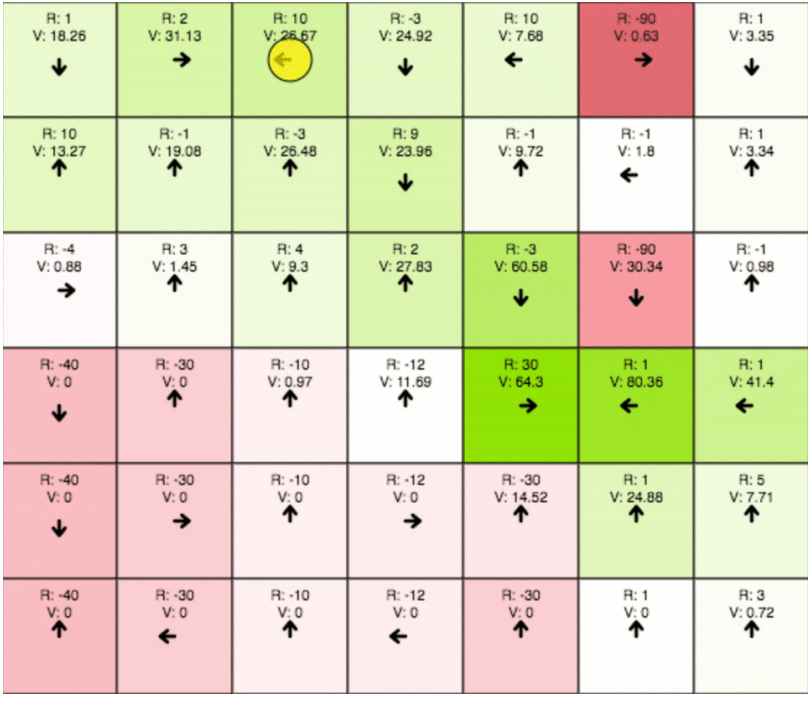
- Model-based
  - Evaluate policy
  - Update policy

$$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$$



Dynamic Programming

- GridWorld example



Monte-Carlo RL

- Characteristics
  - Learn from *complete* episodes of experience
  - Model-free: no knowledge of MDP transitions / rewards
  - Value = mean return
- MC policy
  - Goal: learn v_π from episodes under policy π

$$s_1, a_1, r_1, ..., s_k \sim \pi$$

  - Return is the total discounted reward

$$G_t = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{T-1} r_T$$

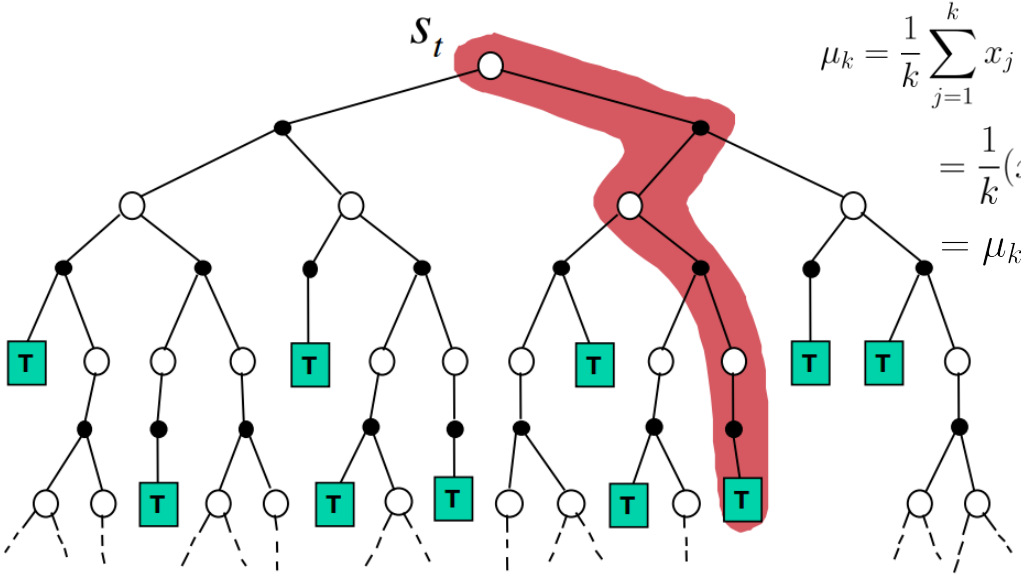
  - Value function is the expected return

$$v_\pi(s_t) = \mathbb{E}[G_t \mid s_t]$$

Monte-Carlo

- Model-free prediction

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$



$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right)$$
$$= \frac{1}{k}(x_k + (k-1)\mu_{k-1})$$
$$= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

Temporal-Difference RL

- Characteristics
  - Learn from *incompele* episodes of experience
  - Model-free: no knowledge of MDP transitions / rewards
  - Update a guess toward a guess
- TD policy
  - Goal: learn v_π online from experience under policy π
  - Value function is updated toward *estimated* return

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

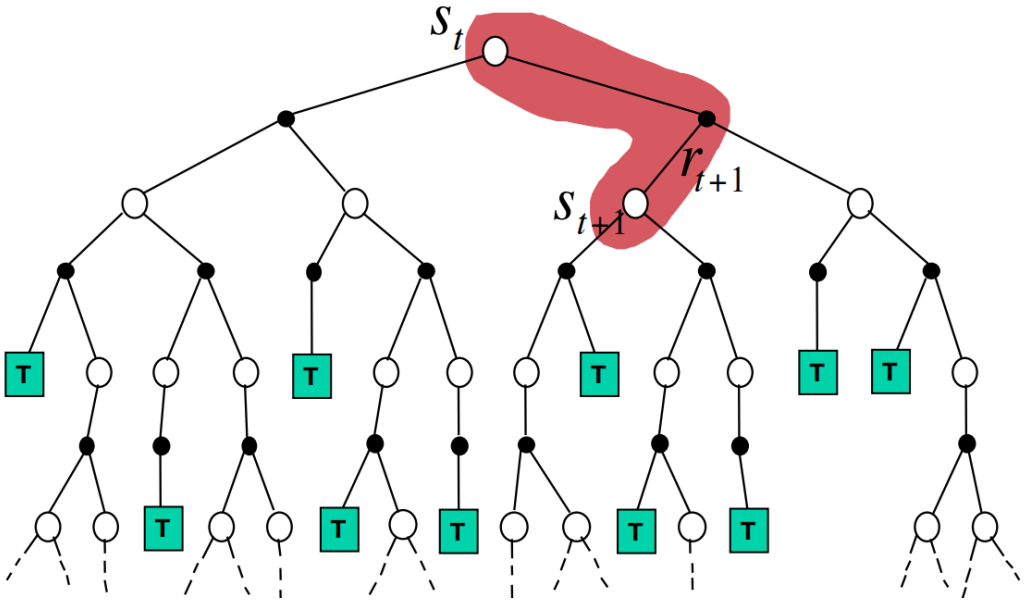
$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

TD target

Temporal-Difference

- Model-free prediction

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

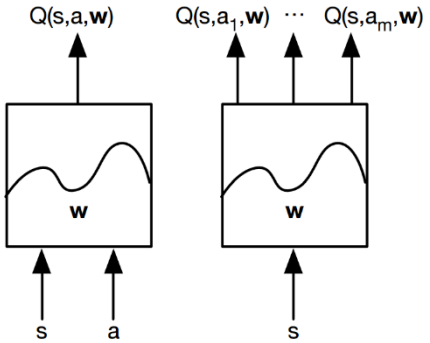


Q-Learning – Value Function Approximation

- Value functions are represented by a *lookup table*

$$Q(s,a) \quad \forall s,a$$

  - too many states and/or actions to store
  - too slow to learn the value of each entry individually
- Values can be estimated with *function approximation*

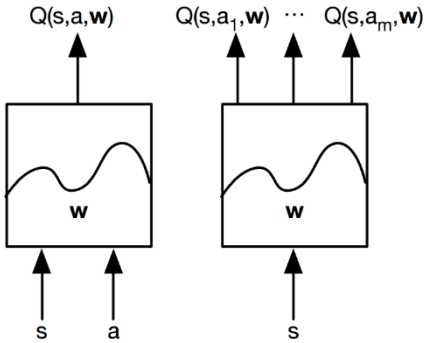


Q-Networks

- Q-networks represent value functions with weights w

$$Q(s, a, w) \approx Q^*(s, a)$$

  - generalize from seen states to unseen states
  - update parameter w for function approximation



Q-Learning

- Goal: estimate optimal Q-values
  - Optimal Q-values obey a Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s',a') \mid s,a]$$

learning target

  - *Value iteration* algorithms solve the Bellman equation

$$Q_{i+1}(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q_i(s',a') \mid s,a]$$

Deep Q-Networks (DQN)

- Represent value function by deep Q-network with weights w

$$Q(s, a, w) \approx Q^*(s, a)$$

- Objective is to minimize mean square error (MSE) loss by SGD

$$L(w) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right)^2\right]$$

learning target

- Leading to the following Q-learning gradient

$$\frac{\partial L(w)}{\partial w} = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right) \frac{\partial Q(s, a, w)}{\partial w}\right]$$

Issue: naïve Q-learning oscillates or diverges using NN due to:
1) correlations between samples 2) non-stationary targets

Stability Issues with Deep RL

- Naive Q-learning oscillates or diverges with neural nets
  1. Data is sequential
     - Successive samples are correlated, non-iid (independent and identically distributed)
  2. Policy changes rapidly with slight changes to Q-values
     - Policy may oscillate
     - Distribution of data can swing from one extreme to another
  3. Scale of rewards and Q-values is unknown
     - Naive Q-learning gradients can be unstable when backpropagated

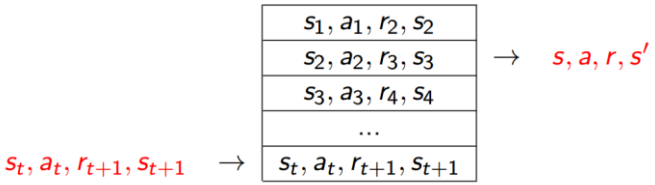
Stable Solutions for DQN

- DQN provides a stable solutions to deep value-based RL
  1. Use experience replay
     - Break correlations in data, bring us back to iid setting
     - Learn from all past policies
  2. Freeze target Q-network
     - Avoid oscillation
     - Break correlations between Q-network and target
  3. Clip rewards or normalize network adaptively to sensible range
     - Robust gradients

Stable Solution 1: Experience Replay

- To remove correlations, build a dataset from agent's experience
  - Take action at according to ϵ-greedy policy
  - Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in replay memory D
  - Sample random mini-batch of transitions (s, a, r, s') from D

small prob for exploration



  - Optimize MSE between Q-network and Q-learning targets

$$L(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a', w) - Q(s, a, w)\right)^2\right]$$

Stable Solution 2: Fixed Target Q-Network

- To avoid oscillations, fix parameters used in Q-learning target
  - Compute Q-learning targets w.r.t. old, fixed parameters w^-

$$r + \gamma \max_{a'} Q(s', a', w^-)$$

  - Optimize MSE between Q-network and Q-learning targets

$$L(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w)\right)^2\right]$$

  - Periodically update fixed parameters $w^- \leftarrow w$

Stable Solution 3: Reward / Value Range

- To avoid oscillations, control the reward / value range
  - DQN clips the rewards to [−1, +1]
    - Prevents too large Q-values
    - Ensures gradients are well-conditioned

Other Improvements: Double DQN

- Nature DQN

$$L(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w)\right)^2\right]$$

- Double DQN: remove upward bias caused by $\max_a Q(s, a, w)$
  - Current Q-network w is used to select actions
  - Older Q-network w^- is used to evaluate actions

$$L(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma Q(s', \arg\max_{a'} Q(s', a', w), w^-) - Q(s, a, w)\right)^2\right]$$

Other Improvements: Prioritized Replay

- Prioritized Replay: weight experience based on surprise
  - Store experience in priority queue according to DQN error

$$\left| r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w) \right|$$

Other Improvements: Dueling Network

- Dueling Network: split Q-network into two channels

$$Q(s,a) = V(s,v) + A(s,a,w)$$

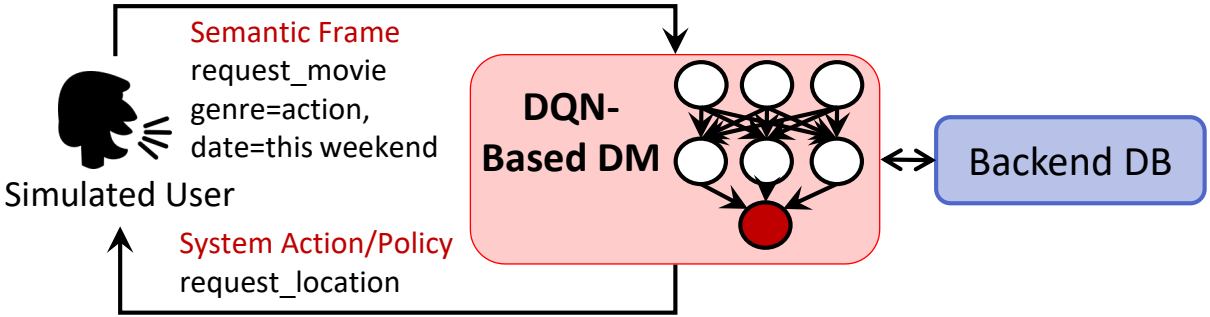
  - Action-independent value function $V(s,v)$
    - Value function estimates how good the state is
  - Action-dependent advantage function $A(s,a,w)$
    - Advantage function estimates the additional benefit

DQN for Dialogue Management (Li et al., 2017)

- Goal: end-to-end learning of values $Q(s, a)$ from interactions

$$L(w) = \mathbb{E}_{s,a,r,s' \sim D}\left[\left(r + \gamma \max_{a'} Q(s', a', w^{-}) - Q(s, a, w)\right)^2\right]$$

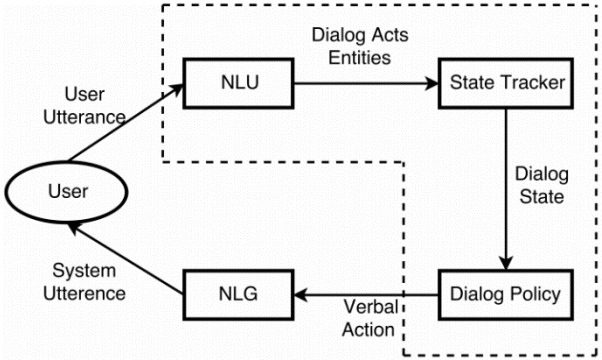
  - Input: state is the combination of user history observation, previous system action, database returned results
  - Output: $Q(s, a)$ for all available system action a
  - Reward: -1 per turn; large reward for successful task

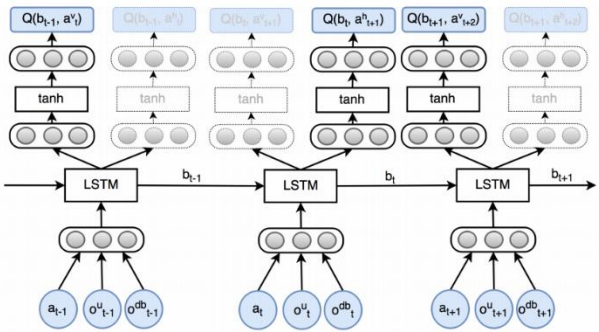


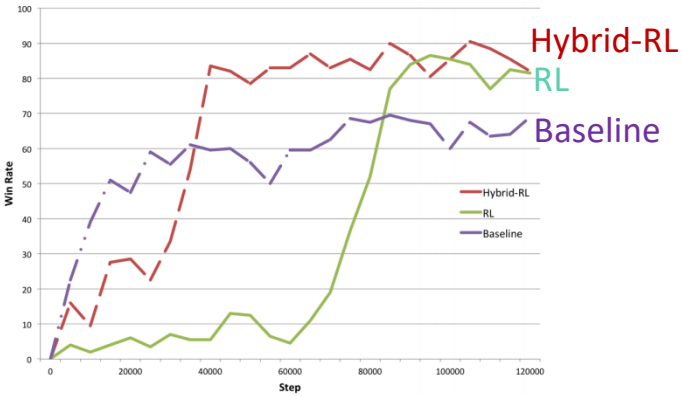
E2E RL-Based System (Zhao and Eskenazi, 2016)

http://www.aclweb.org/anthology/W/W16/W16-36.pdf#page=19

- Joint learning
  - NLU, DST, Dialogue Policy
- Deep RL for training
  - Deep Q-network
  - Deep recurrent network

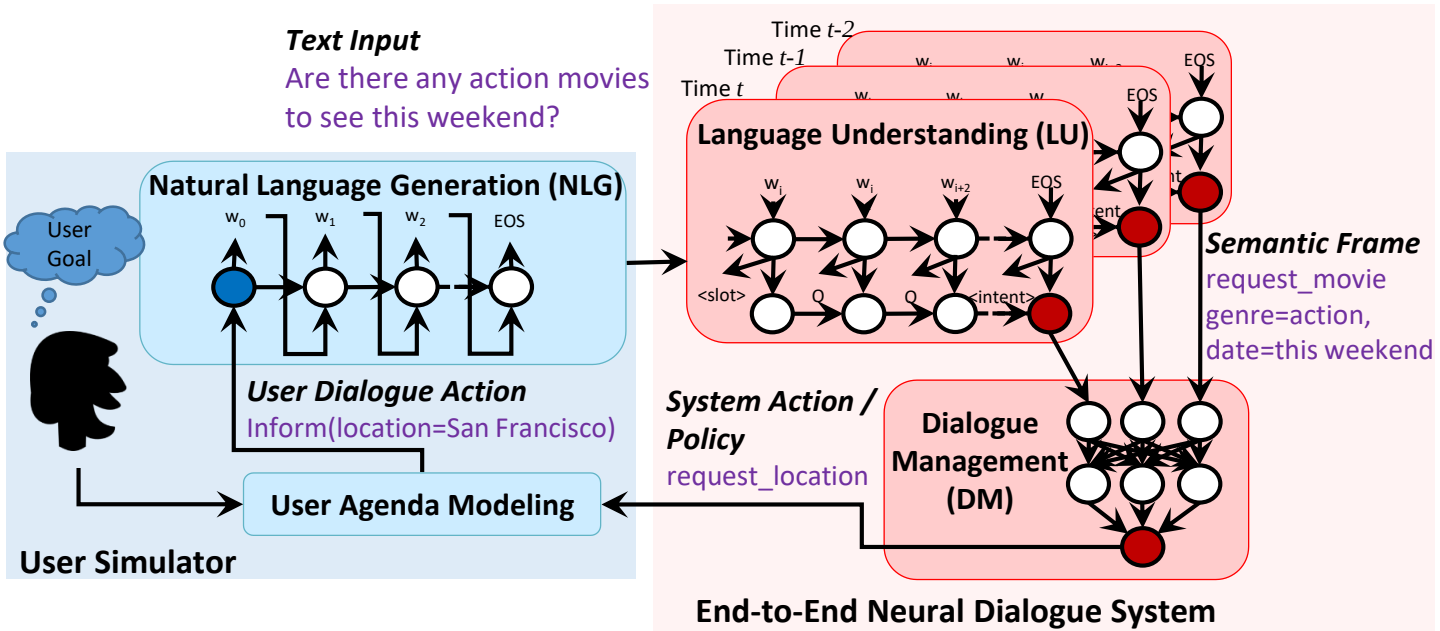






E2E Task-Completion Bot (TC-Bot) (Li et al., 2017)

https://arxiv.org/abs/1703.01008



Idea: supervised learning for each component and reinforcement learning for end-to-end training the neural dialogue system

E2E Task-Completion Bot (TC-Bot) (Li et al., 2017)

https://arxiv.org/abs/1703.01008

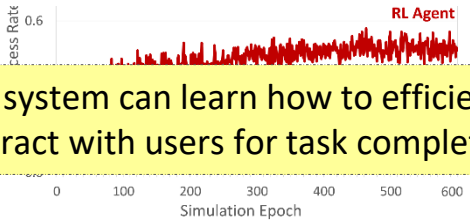
- User goal: ***Two tickets for "the witch" tomorrow 9:30 PM at regal meridian 16, Seattle.***

RULE BASED SYSTEM

User: Can I get 2 tickets for the witch?
Agent: Which theater do you want?
User: Regal meridian 16
Agent: What time would you like to see it?
User: 9:30 pm
Agent: What date would you like to see it?
User: Tomorrow!
Agent: How many people are you?
User: Just two adults.
Agent: What is your location?
User: Seattle!
Agent: What kind of video format do you like?
User: Oh, any kind is okay.
Agent: Do you have any price constraint?
User: No, I don't care.
Agent: Great - I was able to purchase 2 tickets for you to see the witch tomorrow at regal meridian 16 theater in Seattle at 9:30 pm.
User: Thank you.
Agent: Thank you!

REINFORCEMENT LEARNING SYSTEM

User: Can I get 2 tickets for the witch?
Agent: What date would you like to see it?
User: Tomorrow
Agent: Which theater do you prefer?
User: Regal meridian 16
Agent: What is your location?
User: Seattle
Agent: What time would you like to see it?
User: 9:30 pm.
Agent: Great - I was able to purchase 2 tickets for you to see the witch tomorrow at regal meridian 16 theater in Seattle at 9:30 pm.
User: Thanks.
Agent: Thanks!



The system can learn how to efficiently interact with users for task completion

Policy-Based Deep RL

Estimate How Good An Agent's Behavior is

Deep Policy Networks

- Represent policy by deep network with weights u

$$a = \pi(a \mid s, u) \qquad a = \pi(s, u)$$

stochastic policy — deterministic policy

- Objective is to maximize total discounted reward by SGD

$$L(u) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid \pi(\cdot, u)\right]$$

Policy Gradient

- The gradient of a stochastic policy $\pi(a \mid s, u)$ is given by

$$\frac{\partial L(u)}{\partial u} = \mathbb{E}_s \left[\frac{\partial \log \pi(a \mid s, u)}{\partial u} Q^{\pi}(s, a) \right]$$

- The gradient of a deterministic policy $\pi(s, u)$ is given by

$$\frac{\partial L(u)}{\partial u} = \mathbb{E}_s \left[\frac{\partial Q^{\pi}(s, a)}{\partial a} \frac{\partial a}{\partial u} \right] \qquad a = \pi(s, u)$$

How to deal with continuous actions

Actor-Critic (Value-Based + Policy-Based)

- Estimate value function $Q(s, a, w) \approx Q^{\pi}(s, a)$
- Update policy parameters u by SGD
  - Stochastic policy

$$\frac{\partial L(u)}{\partial u} = \mathbb{E}_s \left[\frac{\partial \log \pi(a \mid s, u)}{\partial u} Q(s, a, w) \right]$$

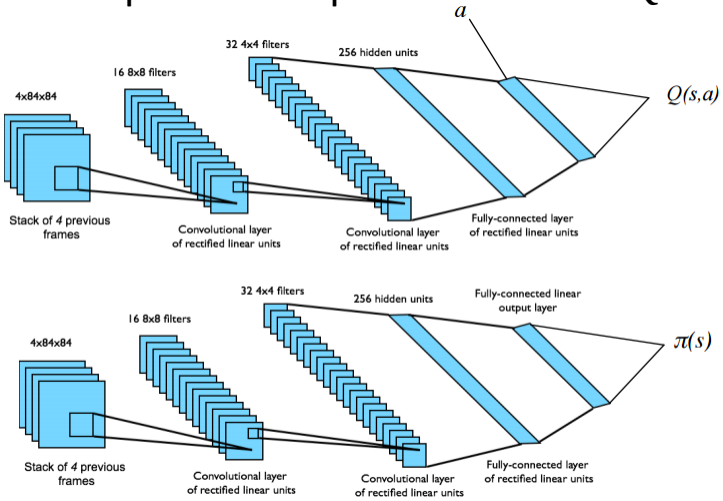
  - Deterministic policy

$$\frac{\partial L(u)}{\partial u} = \mathbb{E}_s \left[\frac{\partial Q(s, a, w)}{\partial a} \frac{\partial a}{\partial u} \right]$$

Q-networks tell whether a policy is good or not
Policy networks optimize the policy accordingly

Deterministic Deep Policy Gradient

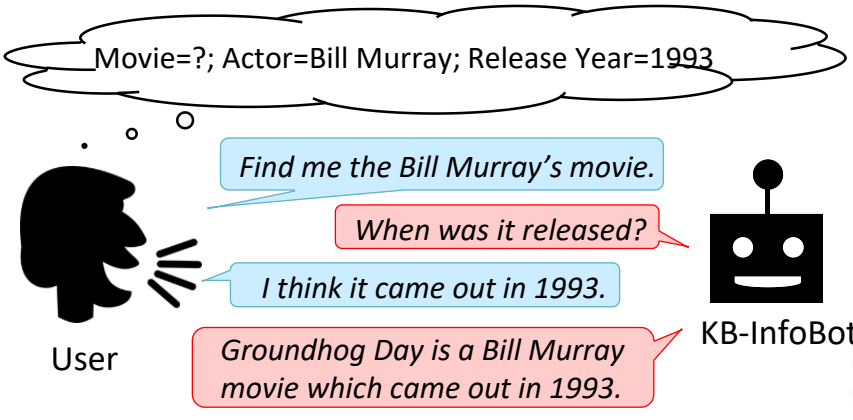
- Goal: end-to-end learning of control policy from pixels
  - Input: state is stack of raw pixels from last 4 frames
  - Output: two separate CNNs for Q and π



Lillicrap et al., "Continuous control with deep reinforcement learning," arXiv, 2015.

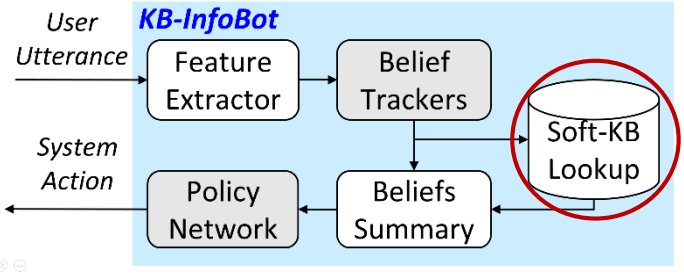
E2E RL-Based Info-Bot (Dhingra et al., 2016)

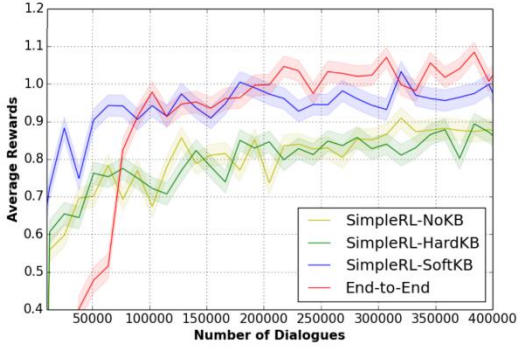
https://arxiv.org/abs/1609.00777



Knowledge Base *(head, relation, tail)*

(Groundhog Day, actor, Bill Murray)
(Groundhog Day, release year, 1993)
(Australia, actor, Nicole Kidman)
(Mad Max: Fury Road, release year, 2015)





Idea: differentiable database for propagating the gradients

Dialogue Management Evaluation

- Metrics
  - Turn-level evaluation: system action accuracy
  - Dialogue-level evaluation: task success rate, reward

Concluding Remarks

- **Dialogue policy optimization** of DM solves MDP via RL
- Value-based
  - Dynamic programming
  - Monte-Carlo
  - Temporal-difference
  - Q-learning → DQN
- Policy-based
  - Deep policy gradient
- Actor-critic