ANLP Presentation 1109

Deep Reinforcement Learning in NLP

Zehao Dou
PhD-Department of Statistics and Data Science, Yale Univ.
zehao.dou@yale.edu

Paper List

Paper 1: Using Reinforcement Learning to Build a Better Model of Dialogue State

Paper 2: A Comparative Study of Reinforcement Learning Techniques on Dialogue Management

Paper 3: Deep Reinforcement Learning for Dialogue Generation

Paper 4: Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning

Paper 5: Single Agent vs. Multi Agent Techniques for Concurrent Reinforcement Learning of Negotiation Dialogue Policies.

Deep Reinforcement Learning

- Supervised Learning: Prediction

  Example: Classification task on ImageNet.

- Unsupervised Learning: Underlying distribution learning

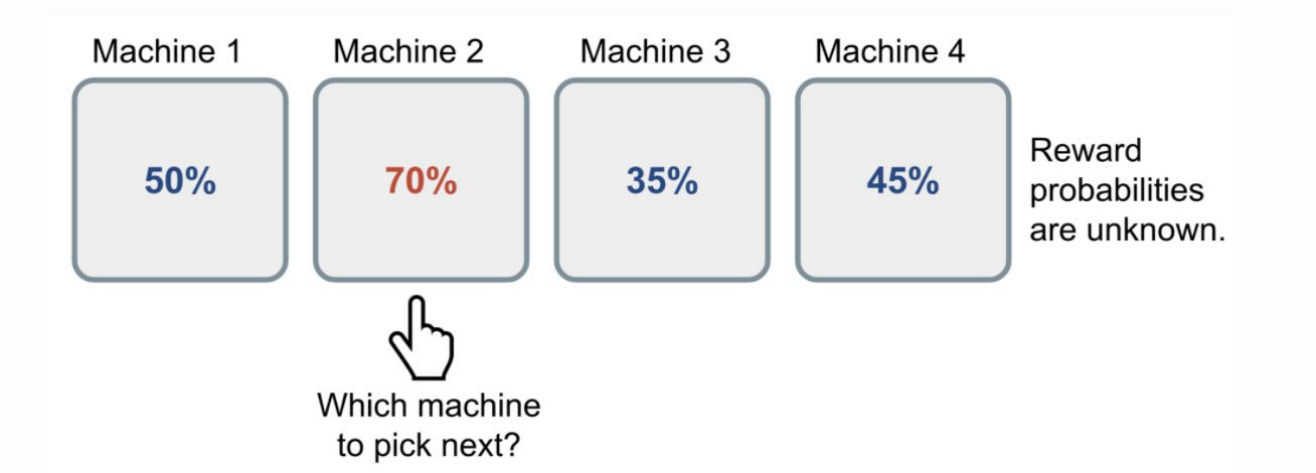
  Example: PCA, Generating human face images by GAN.

- Reinforcement Learning: Decision

  Example: Atari Games, Robotics, Autonomous Driving

Multi-armed Bandits

- Exploration and Exploitation Dilemma



Atari Games

https://www.atari.com/atari-games/

- OpenAI Gym

  Demo: https://github.com/shivaverma/OpenAIGym

  RL Algorithms: https://github.com/gsurma/atari

- Tabular Setting

Special Thanks

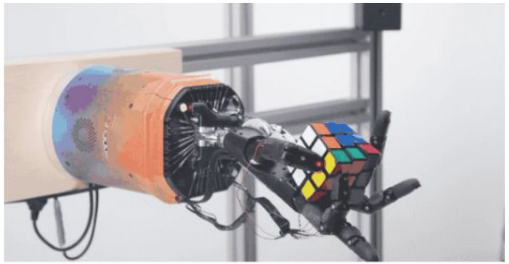
Special thanks to Zhuoran Yang, an assistant professor from the Department of Statistics and Data Science, who gave me permission to use part of his images originally used in his job talk.

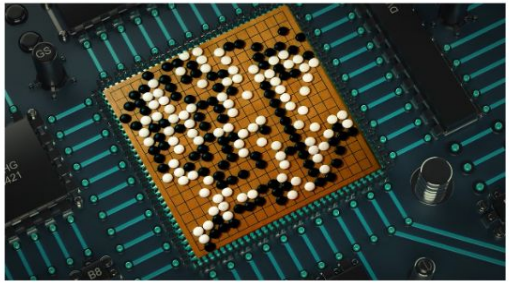
Special thanks to Yaodong Yang, an assistant professor from KCL, who gave me permission to use his MARL image originally used in his DeepMind talk.

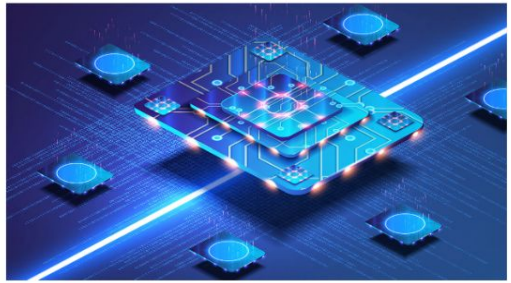
Successes of Deep RL 😄

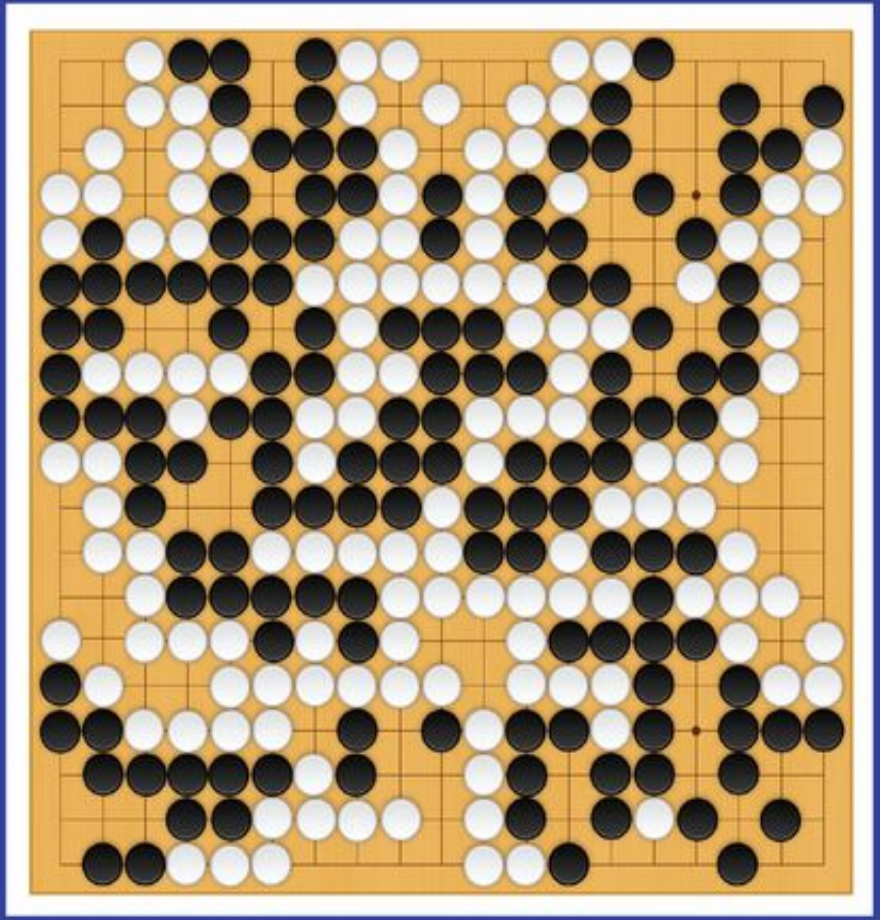
Representation + Decision-Making → Human-Level AI

- Play: Atari, Go, Poker, StarCraft
- Control: robotics (dexterous, swarm), vehicle
- Interact: recommendation, chatbot, rideshare
- Design: chip placement, program synthesis









THE ULTIMATE GO CHALLENGE

GAME 1 OF 3

23 MAY 2017

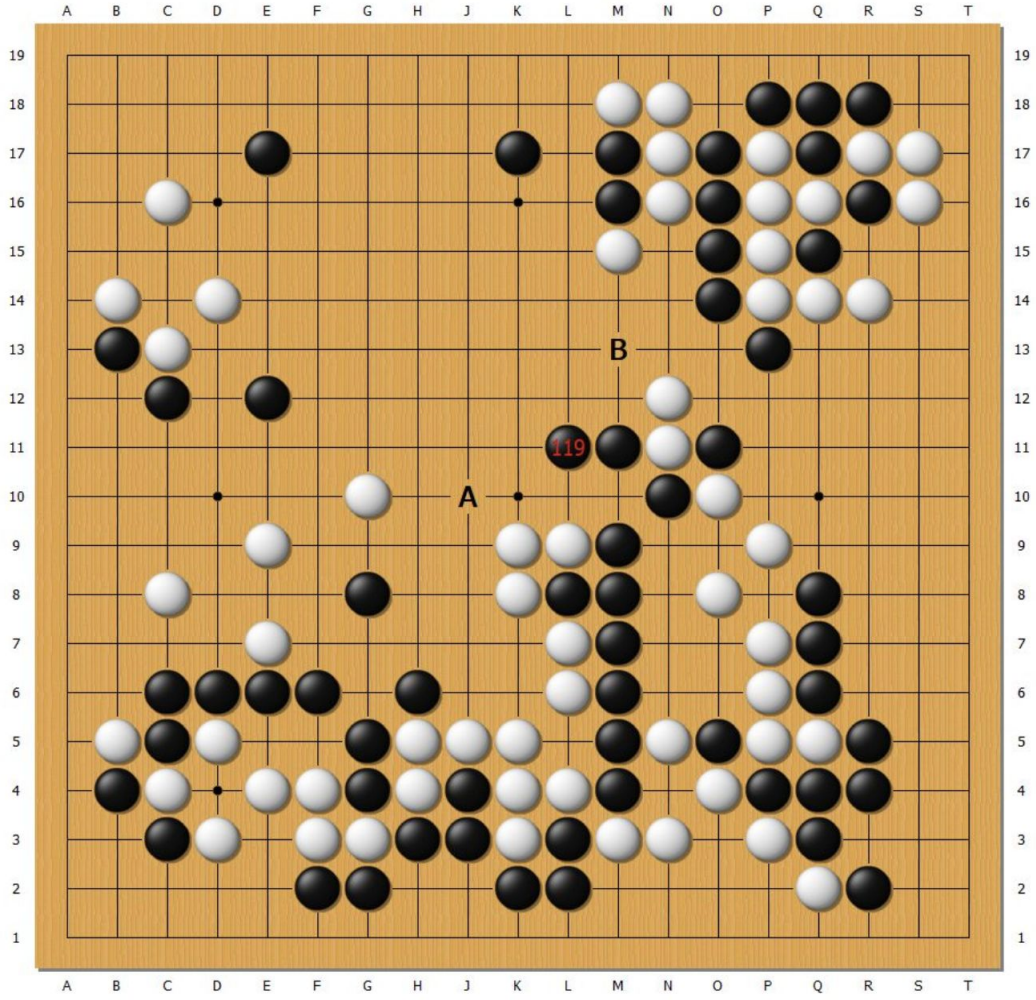
AlphaGo

Winner of Match 1

vs

Ke Jie

RESULT W+0.5


A B C D E F G H J K L M N O P Q R S T
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
A
B
119

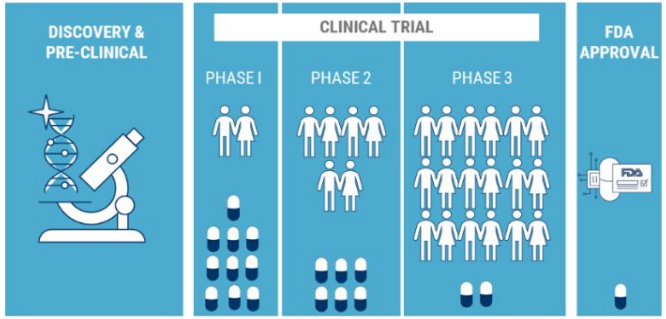
Challenges of Deep RL ☹️

- AlphaGo: 3×10^7 games of self-play (data), 40 days of training (computation)
- AlphaStar: 2×10^2 years of self-play, 44 days of training
- Rubik's Cube: 10^4 years of simulation, 10 days of training

Massive Data Needed → Insufficient Data

Massive Computation → Lack of Convergence

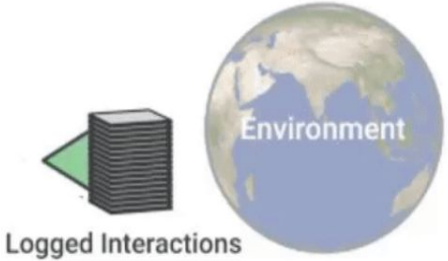
Insufficient Data, Lack of Convergence → **Unreliable AI**



Offline RL vs Online RL

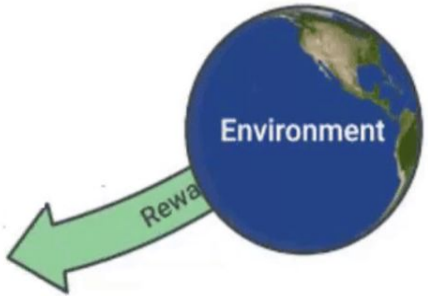
Offline RL

Learn from datasets
No interactions w/ env.
Cannot explore, only exploit

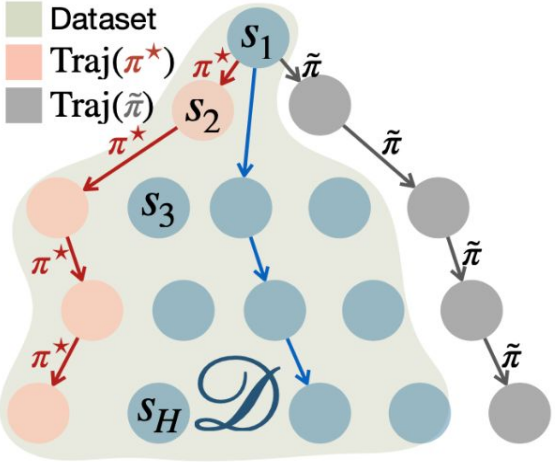


Online RL

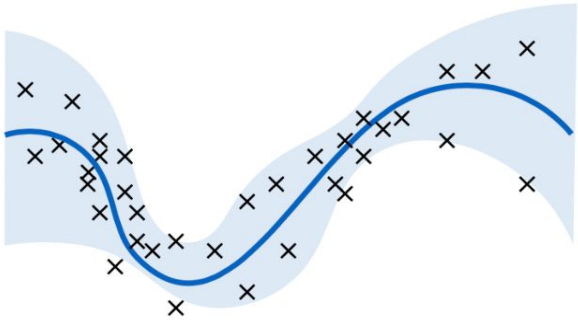
Learn from interactions
Collect a good dataset
Exploration vs. exploitation



Offline RL: Lack of Coverage & Uncertainty



Offline: Learn from Datasets

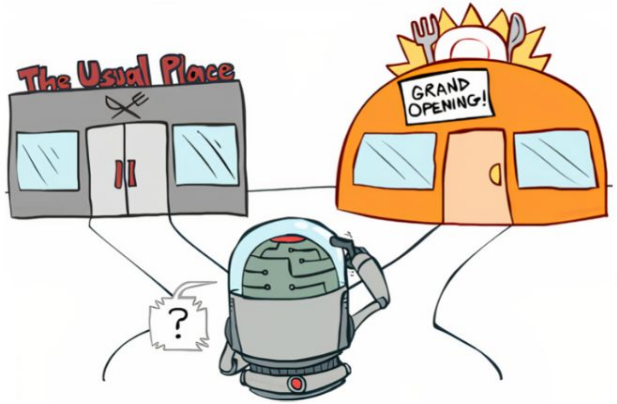


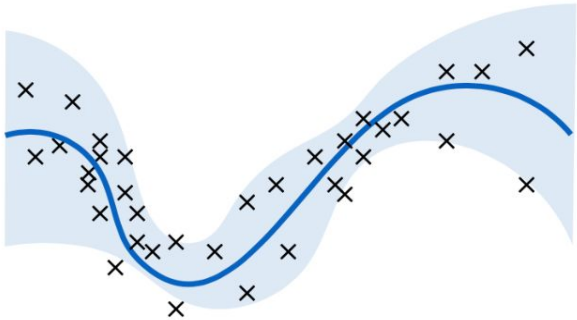
How to assess estimation uncertainty based on arbitrarily collected data?

Arbitrary Dataset → **Offline RL** → **Function Estimation**

How to maximally exploit offline datasets with insufficient coverage?

Online RL: Exploration & Uncertainty



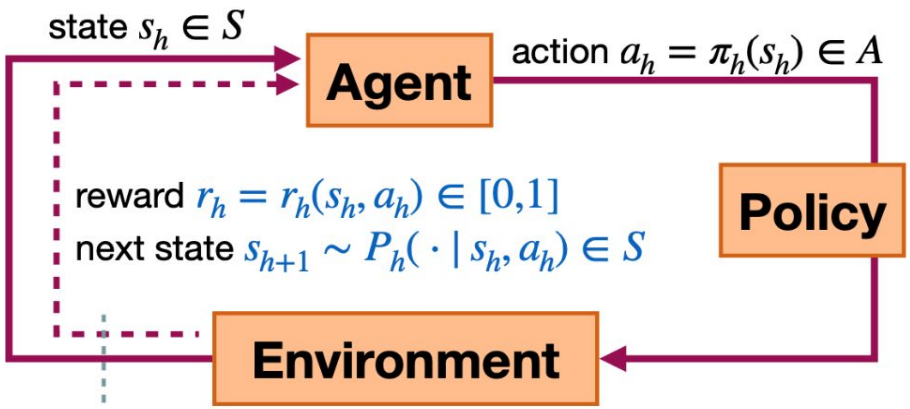


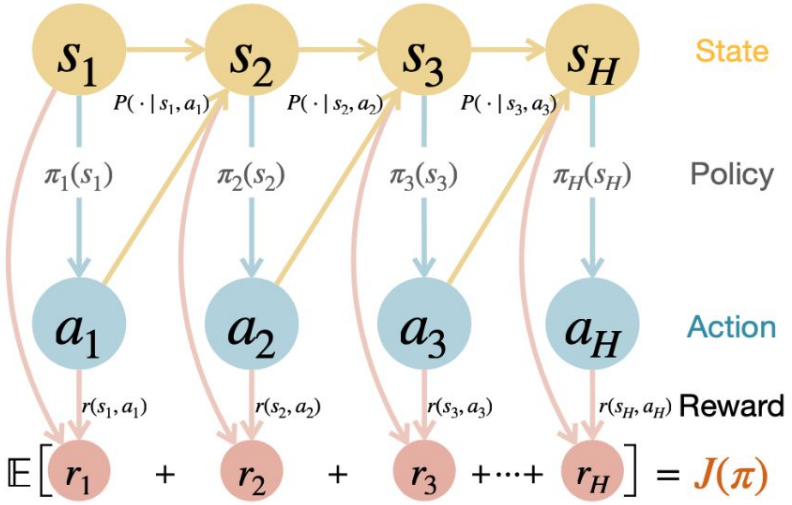
How to assess estimation uncertainty based on adaptively acquired data?

Deep Exploration — **Online RL** — **Function Estimation**

How to construct exploration incentives tailored to function approximators?

Markov Decision Process (MDP)





- S: infinite state space
- A: finite action space
- $r : S \times A \to [0,1]$: unknown reward function
- $P(\cdot \mid s, a) \in \Delta(S)$: unknown transition kernel
- H: finite horizon —— terminate when $h = H$

- $\pi = \{\pi_h\}_{h \in [H]} : S \to A$: policy —— $a_h = \pi_h(s_h)$
- $\mathrm{Traj}(\pi) = \{(s_h, a_h, r_h)\}_{h \in [H]}$: trajectory of π
- $J(\pi) = \mathbb{E}_\pi[\sum_{h=1}^{H} r_h] \in [0,H]$: expected total reward
- $\pi^\star = \mathrm{argmax}_\pi J(\pi) : S \to A$: optimal policy

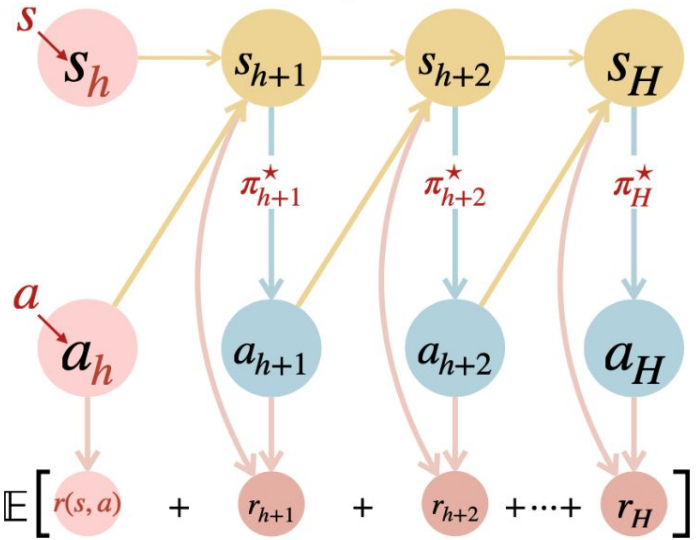
Dynamic Programming & Bellman Equation

- $\pi^\star$ is greedy w.r.t $Q^\star = \{Q_h^\star\}_{h\in[H]}$:

  $\pi_h^\star(s) = \text{argmax}_{a\in A}\ Q_h^\star(s,a)\ \forall s \in S$

$$Q_h^\star(s,a) = r(s,a) + \mathbb{E}_{s_{h+1}\sim P}\left[\max_{a'\in A} Q_{h+1}^\star(s_{h+1},a')\right]$$

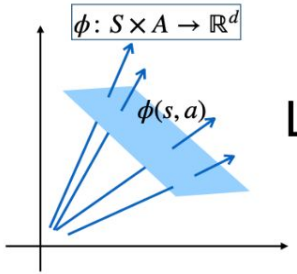
Bellman Operator $\mathbb{B}$: $\mathbb{B}Q_{h+1}^\star$



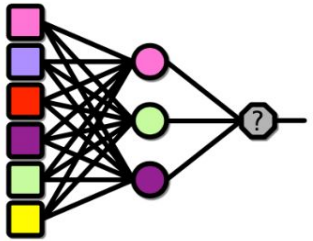
$= Q_h^\star(s,a)$

- $Q_h^\star = \mathbb{B}Q_{h+1}^\star$ $(Q_{H+1}^\star = \mathbf{0})$: Bellman equation
- RL with function approximation:

$\mathcal{F} = \{f: S\times A \to \mathbb{R}\}$ approximates $Q_h^\star$



Linear



NN

Deep Q-learning (DQN)

- Experience Replay: All the state-action pairs (s,a,s',R) are stored in an experience replay buffer. During policy update, samples are drawn at random from replay memory so each tuple can be used multiple times.
- Periodically updated policy: the target policy is updated periodically, instead of for each iteration, so that the training trajectory can be more stable and it overcomes short-term oscillations.

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

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode $= 1, M$ **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $\left(\phi_t, a_t, r_t, \phi_{t+1}\right)$ in D

    Sample random minibatch of transitions $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^{-}\right) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

Policy Gradient Method

- Instead of parameterizing Q function, we parameterize the policy itself. Policy Gradient Method is designed to optimize the policy in order to maximize the potential cumulative rewards.

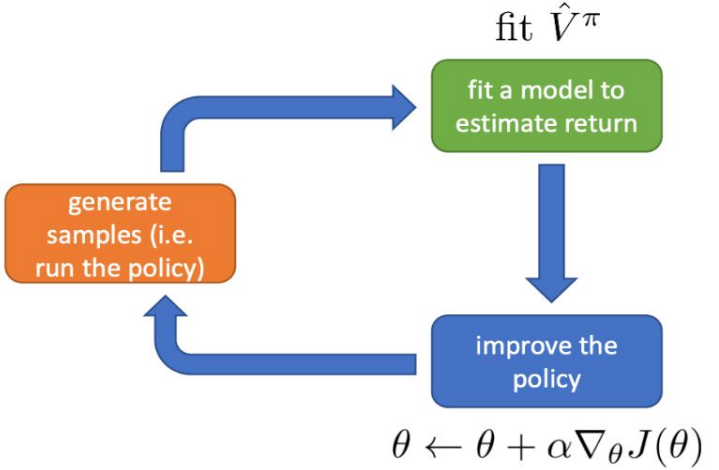
$$
\begin{aligned}
\mathcal{J}(\theta) &= \mathbb{E}_{\pi_\theta}[r] = \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) R(s,a) \\
\nabla \mathcal{J}(\theta) &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \nabla \pi(a|s;\theta) Q_\pi(s,a) \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \frac{\nabla \pi(a|s;\theta)}{\pi(a|s;\theta)} Q_\pi(s,a) \\
&= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \nabla \ln \pi(a|s;\theta) Q_\pi(s,a) \\
&= \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s;\theta) Q_\pi(s,a)]
\end{aligned}
$$

Picture from Lily Log
https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

An actor-critic algorithm

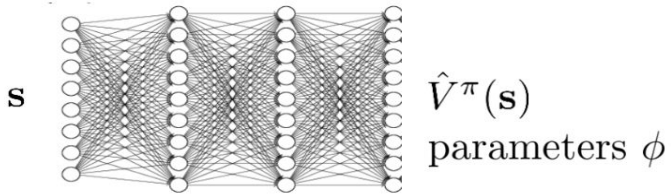
batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha\nabla_\theta J(\theta)$



$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}^\pi_\phi(\mathbf{s}_{i,t+1})$$

$$\mathcal{L}(\phi) = \frac{1}{2}\sum_i \left\| \hat{V}^\pi_\phi(\mathbf{s}_i) - y_i \right\|^2$$



$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t\right]$$

Picture from http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_5_actor_critic_pdf

Reinforcement Learning for language understanding

Sequential decision-making problem for text understanding:

- E.g., Conversation, task-completion, text-based games…
- Agent observes state as a string of text at a time t, e.g., state-text s(t).
- Agent also knows a set of possible actions, each describes as a string text, e.g, action-texts
- Agent tries to understand the “state text” and all possible “action texts”, and takes to right action — right means maximizing the long term reward.
- Then, the environment state transits to a new state, agent receives an immediate reward.

Blog link:
https://medium.com/analytics-vidhya/take-a-peek-at-deep-reinforcement-learning-for-nlp-a8e37fbd7640

Unbounded action space in RL for NLP:

Not only the state space is huge, but the action space is also huge too. Action is characterized by unbounded natural language descriptions. For example, if say to the model “Hey! how are you doing? I was just waiting for F.R.I.E.N.D.S to stream but the power is gone”. Well, this input text from me is the state-space for the model(quite heavy), and the action space is every text combination available(or infinity). This problem for such a huge action space was still the problem in Deep-Q-Network. Then Deep Reinforcement Relevance Network(DRRN) got proposed.

Deep Reinforcement Relevance Network(DRRN):

The idea of DRRN is to project both the state and action into a continuous space(as vectors). Q-function is a relevance function of the state vector and action vector.

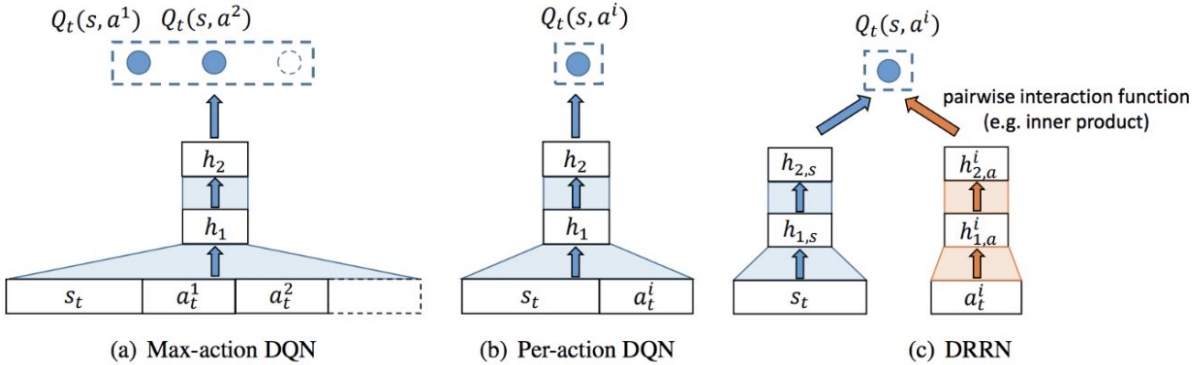


Figure 1: Different deep Q-learning architectures: Max-action DQN and Per-action DQN both treat input text as concantenated vectors and compute output Q-values with a single NN. DRRN models text embeddings from state/action sides separately, and use an interaction function to compute Q-values.

Multi-Agent Reinforcement Learning in Games

Great advantages have been made in 2019!

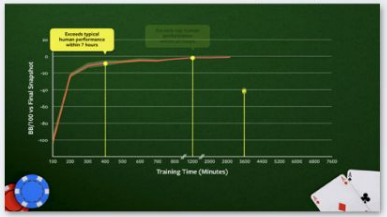
Jan 2016 | Dec 2017 | July 2018 | Jan 2019 | Apr 2019 | July 2019 | Sep 2019

AlphaGO Series

technique of single-agent decision-making is mature

AlphaStar (DeepMind)

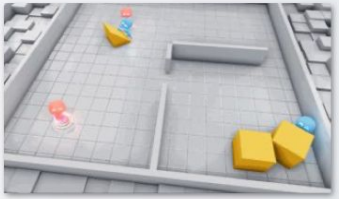
Pluribus Poker (FAIR)



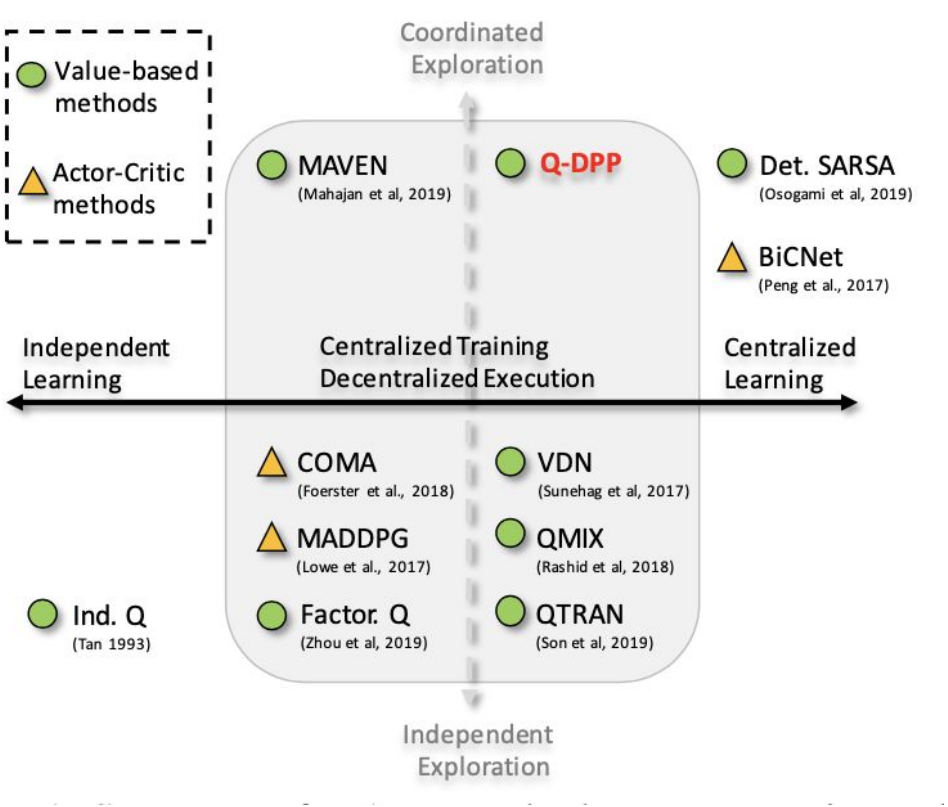
Capture-the-flag (DeepMind)

Dota2 (OpenAI)

Hide and Seek (OpenAI)



techniques of multi-agent decision-making is getting mature !



Picture from: arxiv.org/pdf/2006.01482.pdf

Required Paper 1: Using Reinforcement Learning to Build a Better Model of Dialogue State (ACL 2006)

Joel R. Tetreault
University of Pittsburgh
Learning Research and Development Center
Pittsburgh PA, 15260, USA
tetreaul@pitt.edu

Diane J. Litman
University of Pittsburgh
Department of Computer Science &
Learning Research and Development Center
Pittsburgh PA, 15260, USA
litman@cs.pitt.edu

https://aclanthology.org/E06-1037.pdf

- The purpose of this paper is to using RL to construct a better dialogue state, which means the features that can represent the true state of the speaker.

Introducing the Corpus

Content: Annotated corpus of 20 human-computer spoken dialogue tutoring sessions. Each session consists of an interaction with one student over 5 different physics problems.

Process: After given the physics problem, students write out a short essay first to answer this question. Then, our computer needs to interact with the students to help them understand their mistakes or confused concepts within several rounds of dialogues. After answering all the questions raised by the software. Finally, students officially fix their initial answer and write a new one.

Introducing the Corpus

Score Measuring: The system will give a score to the students' initial answers as well as their final answers. (Pre-test score vs Post-test score). We can calculate the normalized learning gain:

$$NLG = \frac{post - pre}{1 - pre}$$

Introducing the Corpus

State Extraction from Dialogue (Student): Manually annotated by experts.

State	Parameters
Student Move	Shallow (S) Novel & As & Deep (O)
Certainty	Certain, Uncertain, Neutral
Frustration	Frustrated (F), Neutral (N),
Correctness	Correct (C), Incorrect (I) Partially Correct (PC)
Percent Correct	50-100% (High), 0-50% (Low)
Concept Repetition	Concept is not repeated (0), Concept is repeated (R)

Introducing the Corpus

State Extraction from Dialogue (Tutor)

Action	Parameters
Tutor Feedback Act	Positive, Negative
Tutor Question Act	Short Answer Question (SAQ) Complex Answer Question (CAQ)
Tutor State Act	Restatement, Recap, Hint Expansion, Bottom Out

Main Punchline

What features best approximate the student state? What are the best actions for tutor to consider?

Case	Tutor Moves	Example Turn
Feed	Pos	"Super."
Mix	Pos, SAQ	"Good. What is the direction of that force relative to your fi st?"
NonFeed	Hint, CAQ	"To analyze the pumpkin's acceleration we will use Newton's Second Law. What is the defi nition of the law?"

Experimental Results

By adding one feature after another, we get:

#	State	Baseline	New Policy
1	certain:C	**NonFeed**	N: NonFeed F: **Feed**
2	certain:IPC	NonFeed	N: NonFeed F: NonFeed
3	neutral:C	**Feed**	N: Feed F: **NonFeed**
4	neutral:IPC	**Mix**	N: Mix F: **NonFeed**
5	uncertain:C	NonFeed	N: NonFeed F: NonFeed
6	uncertain:IPC	NonFeed	N: NonFeed F: NonFeed

#	State	Baseline	New Policy
1	certain:C	NonFeed	H: NonFeed L: NonFeed
2	certain:IPC	NonFeed	H: NonFeed L: NonFeed
3	neutral:C	**Feed**	H: Feed L: **NonFeed**
4	neutral:IPC	**Mix**	H: Mix L: **NonFeed**
5	uncertain:C	**NonFeed**	H: **Mix** L: NonFeed
6	uncertain:IPC	NonFeed	H: NonFeed L: NonFeed

Discussion

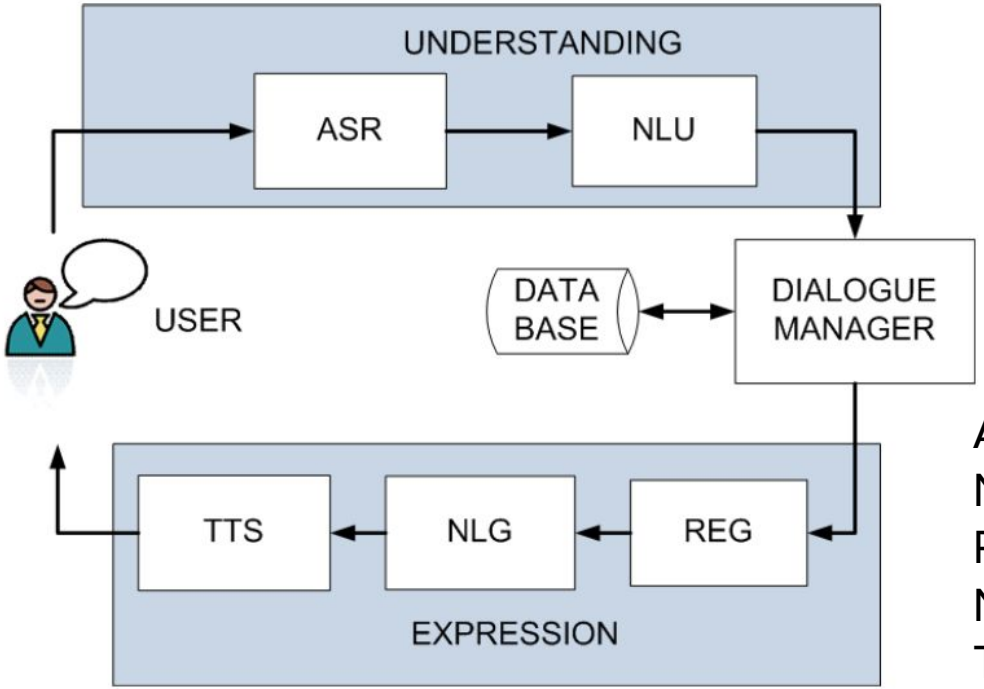
- Why do we need to extract certain states from the students' answers? We know that in Deep Reinforcement Learning, we allow super large state space. Why don't we just use the sentences themselves as the dialogue states?
- Besides the certainty and frustration, can you think of any other emotional or non-emotional features that can be annotated manually and can really help our dialogue generation/ analysis?
- Do you think only 100 dialogues are far not enough to build a reasonable model?

Required Paper 2: A Comparative Study of Reinforcement Learning Techniques on Dialogue Management (ACL 2012)

Alexandros Papangelis
NCSR "Demokritos",
Institute of Informatics
& Telecommunications
and
Univ. of Texas at Arlington,
Comp. Science and Engineering
alexandros.papangelis@mavs.uta.edu

https://aclanthology.org/E12-3003.pdf

- The purpose of this paper is to compare different Reinforcement Learning algorithms on adaptive dialogue system (ADS) building.



- Travel Planning
- Customer Service
- Museum Guide
- Telephone Service

ASR: Automatic Speech Recognition
NLU: Natural Language Understanding
REG: Referring Expression Generation
NLG: Natural Language Generation
TTS: Text to Speech

Achieve Adaptation

- Adjust to current user's personality, behaviour, mood, needs and to the environment in general.
- Adaptive NLG: Use RL to find a policy according to which the system will decide how to present information. (Statistical Planning Problem)

  (How to express your idea adaptively?)

- Adaptive REG: Use RL to choose one of the three strategies (jargon, tutorial or descriptive)

  (How to adaptively choose ideas to express?)

Achieve Adaptation

- Adaptive TTS: The authors propose a model that sorts paraphrases with respect to predictions of which sounds more natural.
- Adaptive Dialogue Management: The Main Challenge.

Results

Algorithm	Model	Policy	Iteration
SARSA(λ)	No	On	Value
LS-SARSA(λ)	No	On	Policy
Q Learning	No	Off	Value
Q(λ)	No	Off	Value
Actor Critic - QV	No	On	Policy
IAC	No	On	Policy
NAC	No	On	Policy
DynaSARSA(λ)	Yes	On	Value
DynaQ	Yes	Off	Value
DynaQ(λ)	Yes	Off	Value
DynaAC-QV	Yes	On	Policy

Algorithm	Average Reward
SARSA(λ)	-10.5967
LS-SARSA(λ)	-14.3439
Q Learning	-14.8888
Q(λ)	-63.7588
Actor Critic - QV	-15.9245
IAC	-10.5000
NAC	**-5.8273**
DynaSARSA(λ)	-11.9758
DynaQ	-14.7270
DynaQ(λ)	-17.1964
DynaAC-QV	-58.4576

Discussion

- What advantages and disadvantages do these adaptive algorithms / models have? In a long term, do you think they do more good than harm for the human beings?
- What kind of improvements can we apply to these models in order to prevent users being trap in an “information cocoon” ?
- In our daily lives, are there any examples of “information cocoon”? Are there any ways for us to protect ourselves?
- Technically, the authors simplify this problem to a slot filling problem. Is this kind of simplification reasonable?

Additional Paper 1: Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning

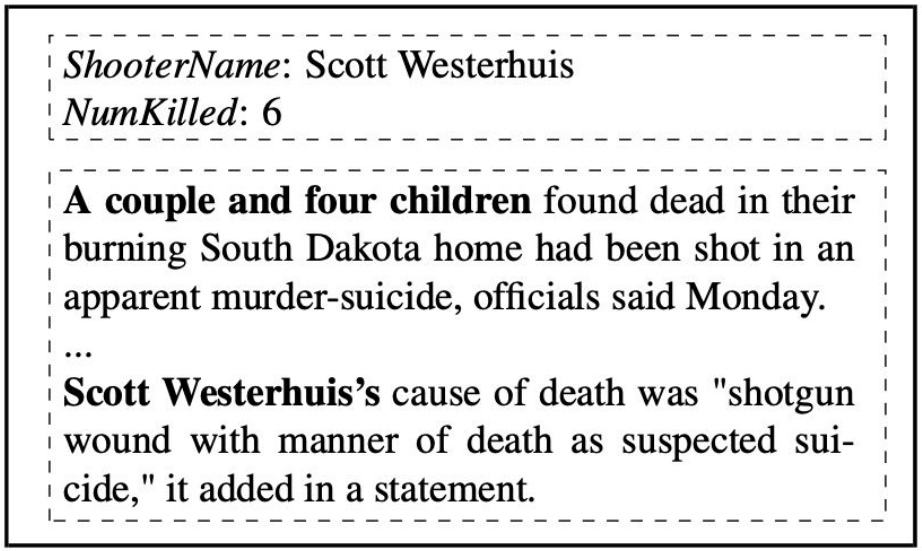
Karthik Narasimhan
CSAIL, MIT
karthikn@mit.edu

Adam Yala
CSAIL, MIT
adamyala@mit.edu

Regina Barzilay
CSAIL, MIT
regina@csail.mit.edu

https://arxiv.org/pdf/1606.01541.pdf

Additional Paper 1: Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning



- Information Extraction System
- The purpose of this paper is to use a RL framework where our model learns to select optimal actions based on contextual information, when a large training corpus is not available.

Main Challenges

1. Performing event coreference, i.e. retrieving suitable articles describing the same type of incidents.
2. Reconciling the entities extracted from these different documents.

To address these challenges, the authors use a Reinforcement Learning approach to combine query formulation, extraction from new sources, and value reconciliation .

Model Formulation

- States: The states of the MDP incorporate these pieces of information:

  1. Confidence scores of current and newly extracted entity values.

  2. One-hot encoding of matches between current and new values.

  3. Unigram/tf-idf counts of context words around the entity values.

  4. Tf-idf similarity between the original article and the new article.

Model Formulation

- Actions: At each step, the agent is required to take two actions.

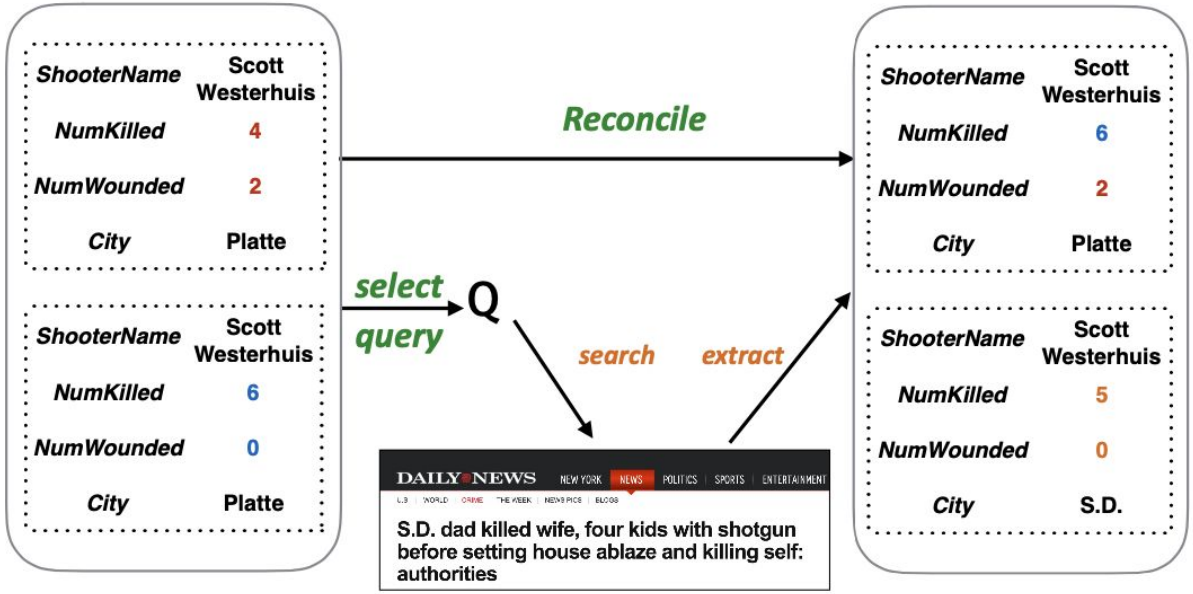
  1. A reconciliation decision d. (1) Accept the current entity's value. (2) Accept all entity values. (3) Reject all values. (4) Stop.

  2. A query decision q. It is used to choose the next query from a set of automatically generated alternatives in order to retrieve the next article.

Model Formulation

- Reward function:

  Maximizing the final extraction accuracy, while minimizing the number of queries.



State 1

State 2

Current Values:

ShooterName → Scott Westerhuis
NumKilled → 4
NumWounded → 2
City → Platte

New Values:

ShooterName → Scott Westerhuis
NumKilled → 6
NumWounded → 0
City → Platte

State:

$\langle 0.3, 0.2, 0.5, 0.1,$ ← currentConf
$0.4, 0.6, 0.2, 0.4,$ ← newConf
$1, 0, 0, 1, 0, 1, 1, 0,$ ← matches
$0.2, 0.3, ..., 0.1, 0.5,$ ← contextWords
$0.65\rangle$ ← document tf-idf similarity

Algorithm 1 MDP framework for Information Extraction (Training Phase)

Initialize set of original articles X
for $x_i \in X$ **do**
  for each query template T^q **do**
    Download articles with query $T^q(x_i)$
    Queue retrieved articles in Y_i^q
for $epoch = 1, M$ **do**
  for $i = 1, |X|$ **do** //episode
    Extract entities e_0 from x_i
    $e_{cur} \leftarrow e_0$
    q $\leftarrow 0$, r $\leftarrow 0$ //query type, reward
    while Y_i^q not empty **do**
      Pop next article y from Y_i^q
      Extract entities e_{new} from y
      Compute tf-idf similarity $\mathcal{Z}(x_i, y)$
      Compute context vector $\mathcal{C}(y)$
      Form state s using e_{cur}, e_{new}, $\mathcal{Z}(x_i, y)$ and $\mathcal{C}(y)$
      Send (s, r) to agent
      Get decision d, query q from agent
      if q == "end_episode" **then break**
      $e_{prev} \leftarrow e_{cur}$
      $e_{cur} \leftarrow$ *Reconcile*(e_{cur}, e_{new}, d)
      $r \leftarrow \sum_{\text{entity } j} \text{Acc}(e_{cur}^j) - \text{Acc}(e_{prev}^j)$
    Send (s_{end}, r) to agent

Algorithm 2 Training Procedure for DQN agent with ϵ-greedy exploration

Initialize experience memory $\mathcal{D}$
Initialize parameters θ randomly
for $episode = 1, M$ **do**
  Initialize environment and get start state s_1
  for $t = 1, N$ **do**
    if $random() < \epsilon$ **then**
      Select a random action a_t
    else
      Compute $Q(s_t, a)$ for all actions a
      Select $a_t = \text{argmax } Q(s_t, a)$
    Execute action a_t and observe reward r_t and new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample random mini batch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    $$y_j = \begin{cases} r_j, & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta_t), & \text{else} \end{cases}$$
    Perform gradient descent step on the loss $\mathcal{L}(\theta) = (y_j - Q(s_j, a_j; \theta))^2$
    if s_{t+1} == s_{end} **then break**

Additional Paper 2: Deep Reinforcement Learning for Dialogue Generation

Jiwei Li[1], Will Monroe[1], Alan Ritter[2], Michel Galley[3], Jianfeng Gao[3] and Dan Jurafsky[1]
[1]Stanford University, Stanford, CA, USA
[2]Ohio State University, OH, USA
[3]Microsoft Research, Redmond, WA, USA
{jiweil,wmonroe4,jurafsky}@stanford.edu, ritter.1492@osu.edu
{mgalley,jfgao}@microsoft.com

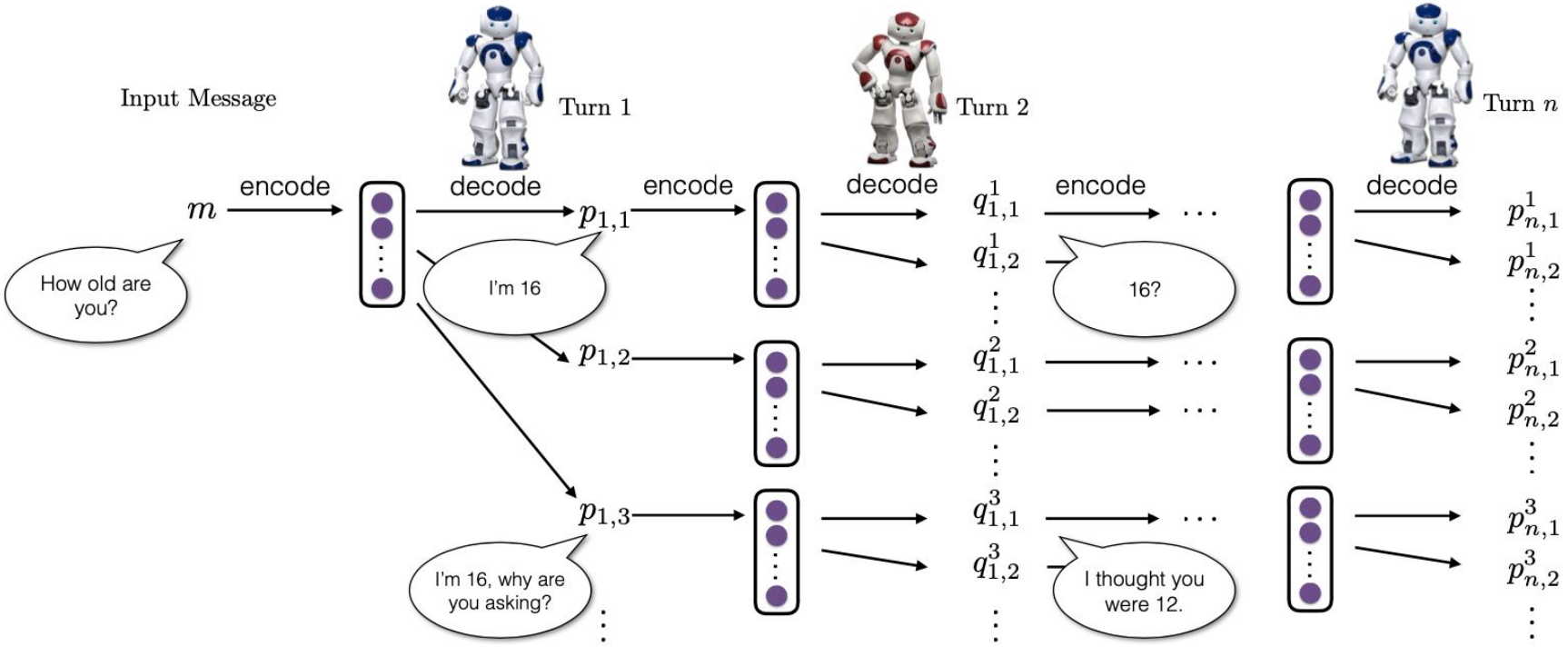
https://arxiv.org/pdf/1603.07954.pdf

- The purpose of this paper is to use deep reinforcement learning method to improve the existing dialogue generation models so that they can be longer-sighted and focus more on their influence on future outcomes, since the current models tend to be short-sighted.

Baseline mutual information model (Li et al. 2015)	Proposed reinforcement learning model
A: Where are you going? (1) B: I'm going to the restroom. (2) A: See you later. (3) B: See you later. (4) A: See you later. (5) B: See you later. (6)	A: Where are you going? (1) B: I'm going to the police station. (2) A: I'll come with you. (3) B: No, no, no, no, you're not going anywhere. (4) A: Why? (5) B: I need you to stay here. (6) A: I don't know what you are talking about. (7) ...
A: how old are you? (1) B: I'm 16. (2) A: 16? (3) B: I don't know what you are talking about. (4) A: You don't know what you are saying. (5) B: I don't know what you are talking about . (6) A: You don't know what you are saying. (7) ...	A: How old are you? (1) B: I'm 16. Why are you asking? (2) A I thought you were 12. (3) B: What made you think so? (4) A: I don't know what you are talking about. (5) B: You don't know what you are saying. (6) ... ...

Model Formulation

- State: The authors only denote the previous two dialogue turns as part of the dialogue history and they transform them to a vector representation by using an LSTM encoding model.
- Action: The dialogue sentence to generate. Notice that the action space is infinite since we can generate sequences with arbitrary length.
- Policy: It takes the form of an LSTM encoder-decoder. Given the state (the encoding version of previous two dialogue turns), it outputs the next sentence. Our policy is parameterized by LSTM architecture so that we can use actor-critic or other policy-based methods to find the optimal policy.


Input Message
Turn 1
Turn 2
Turn n
m
encode
decode
encode
decode
encode
decode
How old are you?
I'm 16
16?
I'm 16, why are you asking?
I thought you were 12.
$p_{1,1}$
$p_{1,2}$
$p_{1,3}$
$q^1_{1,1}$
$q^1_{1,2}$
$q^2_{1,1}$
$q^2_{1,2}$
$q^3_{1,1}$
$q^3_{1,2}$
$p^1_{n,1}$
$p^1_{n,2}$
$p^2_{n,1}$
$p^2_{n,2}$
$p^3_{n,1}$
$p^3_{n,2}$

Model Formulation

- Reward Design:

  Ease of Answering (Schegloff and Sacks, 1973; Allwood et al, 1992). Minimize the probability that the current sentence will lead to a dialogue black hole in the following step.

$$r_1 = -\frac{1}{N_{\mathbb{S}}} \sum_{s \in \mathbb{S}} \frac{1}{N_s} \log p_{\text{seq2seq}}(s|a)$$

  Information Flow We don't want repetitive sentences. So we minimize the cosine similarity between the representations of adjacent responses.

$$r_2 = -\log \cos(h_{p_i}, h_{p_{i+1}}) = -\log \cos \frac{h_{p_i} \cdot h_{p_{i+1}}}{\|h_{p_i}\| \|h_{p_{i+1}}\|}$$

Model Formulation

- Reward Design:

  Semantic Coherence We don't want high-rewarding responses which are ungrammatical or not coherent. The first term is to maximize the probability of generating the response a given the previous two dialogue turns. The second term is to maximize the backward probability of the previous dialogue conditioned on the response.

$$r_3 = \frac{1}{N_a} \log p_{\text{seq2seq}}(a|q_i, p_i) + \frac{1}{N_{q_i}} \log p^{\text{backward}}_{\text{seq2seq}}(q_i|a)$$

Results

Model	# of simulated turns
SEQ2SEQ	2.68
mutual information	3.40
RL	4.48

Model	Unigram	Bigram
SEQ2SEQ	0.0062	0.015
mutual information	0.011	0.031
RL	0.017	0.041

A: What's your name ?
B: Daniel.
A: How old are you ?
B. Twelve. What's your name ?
A. Daniel.
B: How old are you ?
A: Twelve. What's your name ?
B: Daniel.
A: How old are you ?
B ...

Results

Input Message	Mutual Information Model	RL
How old are you?	I'm 16.	I'm 16, why are you asking?
What's your full name?	I have no idea.	What's yours?
I don't want to go home tonight.	Really?	Why?
Do you have any feelings for me?	I don't know what you are talking about.	I don't want to hurt your feelings.
How much time do you have here?	Not long enough. Sorry, sir.	Ten seconds.
Shall we get started?	Of course!	Yes. We've got a lot of work to do here.
Do you play football?	No, i don't	Yes. I love football!
We'd have to talk to him.	I mean, he's a good guy	About what ?
How come you never say it?	Because I don't want to hurt you.	I don't think it's a good idea to say it.

Addition Paper 3. Single-Agent vs. Multi-Agent Techniques for Concurrent Reinforcement Learning of Negotiation Dialogue Policies (ACL 2014)

Kallirroi Georgila, Claire Nelson, David Traum
University of Southern California Institute for Creative Technologies
12015 Waterfront Drive, Playa Vista, CA 90094, USA
{kgeorgila,traum}@ict.usc.edu

https://aclanthology.org/P14-1047.pdf

Main Punchline

- The purpose of this paper is to propose concurrent learning using multi-agent RL, without any need for simulated users to train against or corpora to learn from.

Main Approaches to Learn Dialogue Policies

1. Learn against a simulated user (SU), which is a model that simulates the behavior of a real user (Georgila et al., 2006; Schatzmann et al., 2006).
2. Learn directly from a corpus (Henderson et al., 2008; Li et al., 2009).
3. Learn via live interaction with human users (Singh et al., 2002).
4. Concurrent Learning of the system policy and the SU policy using multi-agent RL techniques.

Experimental Setup

Two agents negotiate about how to share resources. For simplicity, we call them apples and oranges. They have different goals and they are constrained of imperfect information about each other. They do not know each other’s reward function or degree of rationality.

	Agent 1	Agent 2
apple	300	200
orange	200	300

Agent 1: offer-2-2 (I offer you 2 A and 2 O)

Agent 2: offer-3-0 (I offer you 3 A and 0 O)

Agent 1: offer-0-3 (I offer you 0 A and 3 O)

Agent 2: offer-4-0 (I offer you 4 A and 0 O)

Agent 1: accept (I accept your offer)

Finally, after conducting experiments the authors conclude that, Q-learning fails to converge while PHC and PHC-WoLF always converge and perform similarly. Also, gradually decreasing exploration rates are required for convergence.

Discussion

- What else can be contained into our reward function, besides semantic coherence, information flow, grammatical correctness?
- If we do not want to manually design reward function, what else can we do to run the Reinforcement Learning framework?

Relevant Links

http://www.wildml.com/2016/10/learning-reinforcement-learning/

https://www.intel.com/content/www/us/en/developer/tools/frameworks/overview.html

https://adeshpande3.github.io/adeshpande3.github.io/Deep-Learning-Research-Review-Week-2-Reinforcement-Learning

http://www.kdnuggets.com/2016/11/deep-learning-research-review-reinforcement-learning.html

https://github.com/dennybritz/reinforcement-learning

https://ayearofai.com/lenny-1-robots-reinforcement-learning-ae6e69ff4cf0

https://magenta.tensorflow.org/2016/11/09/tuning-recurrent-networks-with-reinforcement-learning

https://blog.altoros.com/reinforcement-learning-with-tensorflow-deep-q-networks-a-monitored-session-and-a3c.html

https://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning

Relevant Links

https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0

https://medium.com/@awjuliani/super-simple-reinforcement-learning-tutorial-part-1-fd544fab149

https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-1-5-contextual-bandits-bff01d1aad9c

https://medium.com/@awjuliani/super-simple-reinforcement-learning-tutorial-part-2-ded33892c724

https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-3-model-based-rl-9a6fe0cce99

https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-4-deep-q-networks-and-beyond-8438a3e2b8df

https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflow-part-5-visualizing-an-agents-thoughts-and-actions-4f27b134bb2a

https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-6-partial-observability-and-deep-recurrent-q-68463e9aeefc

https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-7-action-selection-strategies-for-exploration-d3a97b7cceaf

https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2

Relevant Links

https://www.oreilly.com/learning/introduction-to-reinforcement-learning-and-openai-gym

http://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/

http://neuro.cs.ut.ee/deep-reinforcement-learning-with-neon/

https://medium.com/towards-data-science/reinforcement-learning-w-keras-openai-actor-critic-models-f084612cfd69

https://www.oreilly.com/ideas/reinforcement-learning-for-complex-goals-using-tensorflow

https://www.analyticsvidhya.com/blog/2016/12/getting-ready-for-ai-based-gaming-agents-overview-of-open-source-reinforcement-learning-platforms/

https://medium.com/machine-learning-for-humans/reinforcement-learning-6eacf258b265

http://www.datasciencecentral.com/profiles/blogs/reinforcement-learning-part-3-challenges-considerations

http://www.datasciencecentral.com/profiles/blogs/reinforcement-learning-and-ai

https://www.intelnervana.com/deep-reinforcement-learning-with-neon/

https://github.com/ShangtongZhang/reinforcement-learning-an-introduction

Relevant Links

https://www.windowsdevcenter.com/learning/introduction-to-reinforcement-learning-and-openai-gym

http://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

http://videolectures.net/rldm2015_silver_reinforcement_learning/

https://www.oreilly.com/learning/introduction-to-reinforcement-learning-and-openai-gym

https://buzzrobot.com/5-ways-to-get-started-with-reinforcement-learning-b96d1989c575

http://pytorch.org/tutorials/_downloads/reinforcement_q_learning.py

http://pytorch.org/tutorials/_downloads/reinforcement_q_learning.ipynb

https://blog.openai.com/deep-reinforcement-learning-from-human-preferences/

http://mostafadehghani.com/2017/08/31/some-highlights-of-mila-deep-learning-and-reinforcement-learning-summer-schools-2017/

https://sigmoidal.io/alphago-how-it-uses-reinforcement-learning-to-beat-go-masters/

https://deepmind.com/blog/deep-reinforcement-learning/