RL and GAN for Sentence Generation and Chat-bot

Hung-yi Lee

Outline

- Policy Gradient
- SeqGAN
  - Two techniques: MCMC, partial
  - Experiments: SeqGAN and dialogue
- Original GAN
  - MadliGAN
  - Gumbel

Review: Chat-bot

- Sequence-to-sequence learning

Training data:

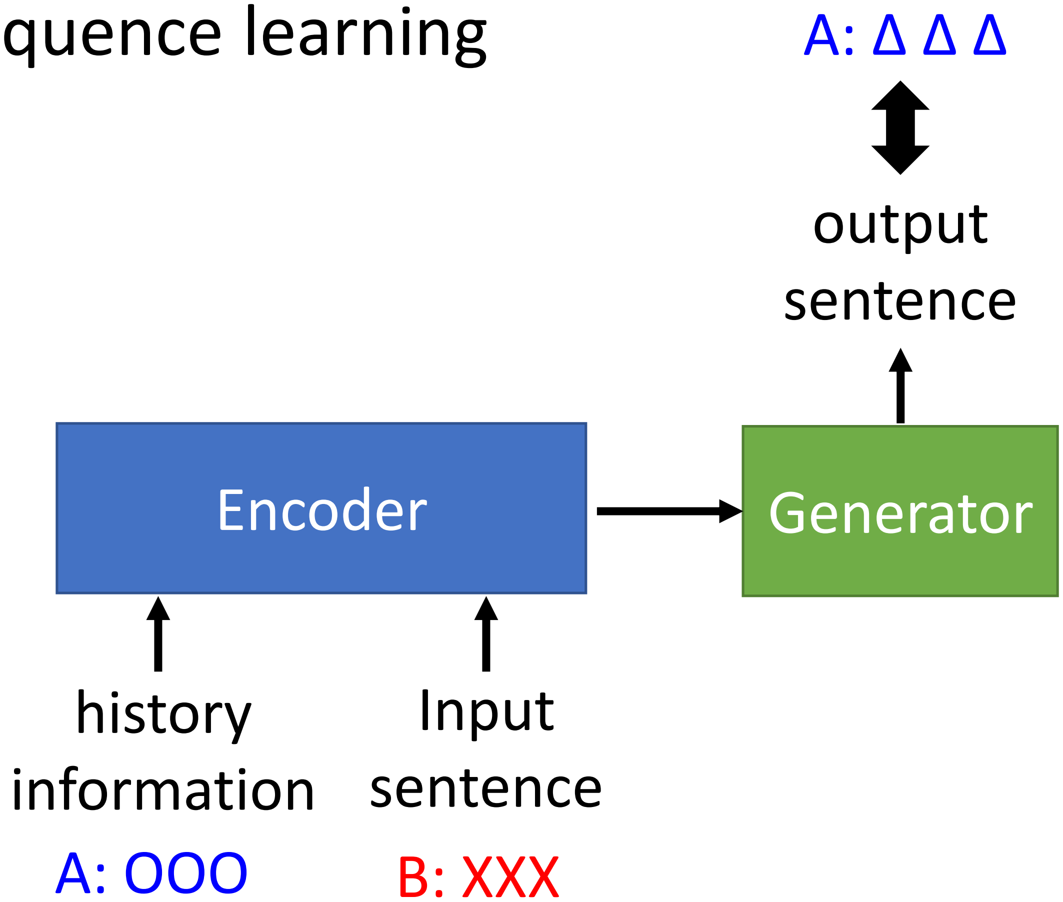
⋮

A: OOO

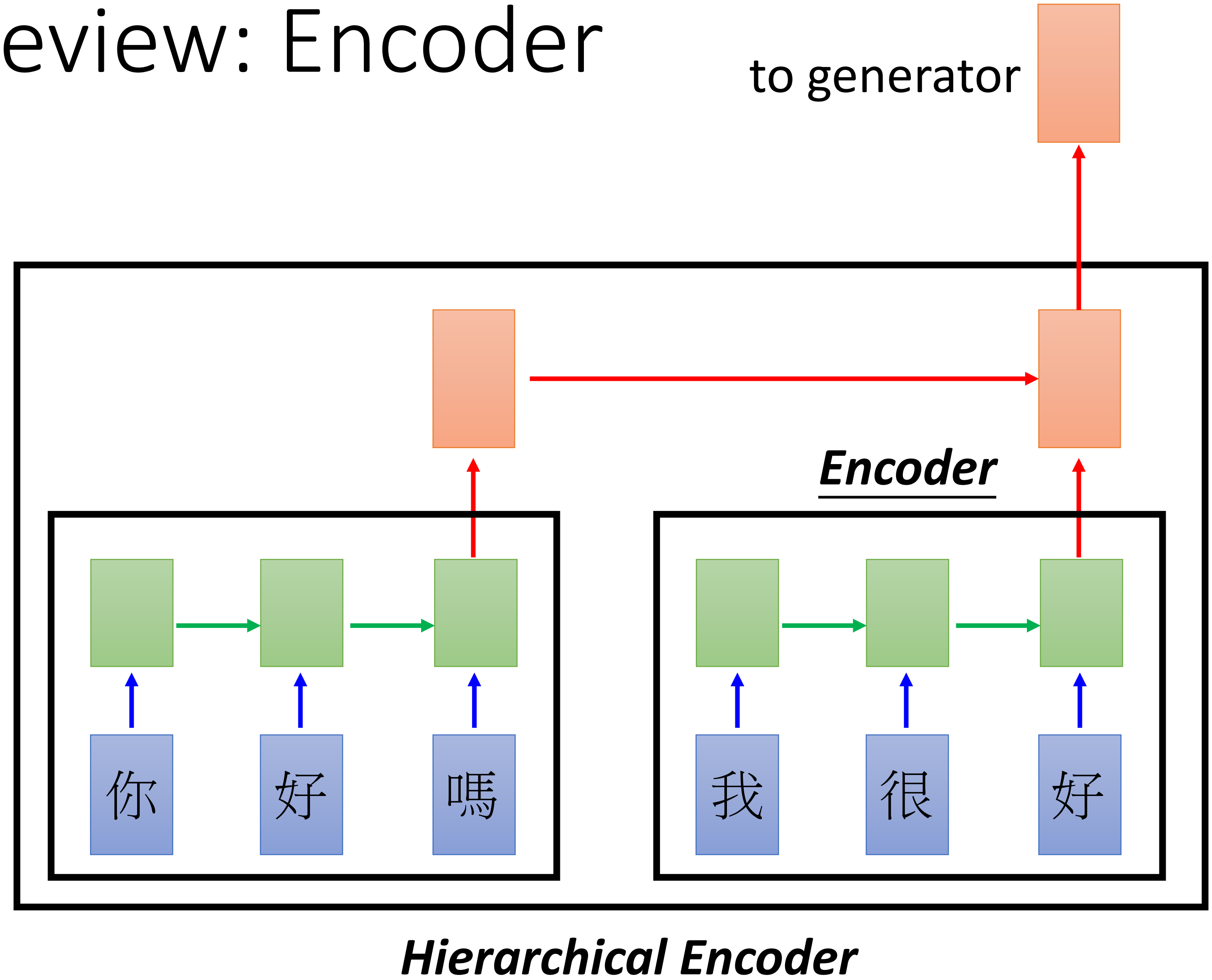
B: XXX

A: Δ Δ Δ

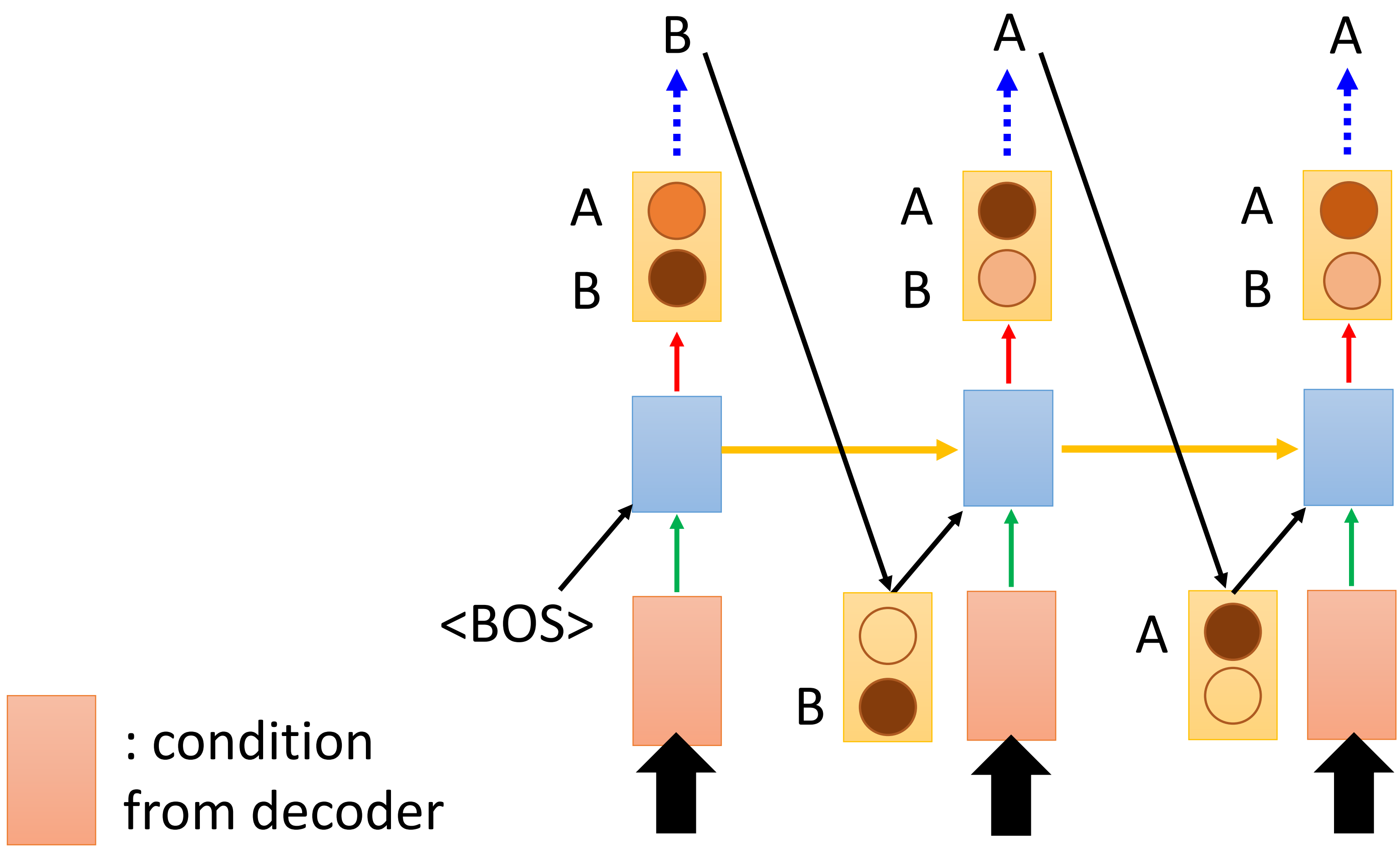
⋮



Review: Encoder



Review: Generator



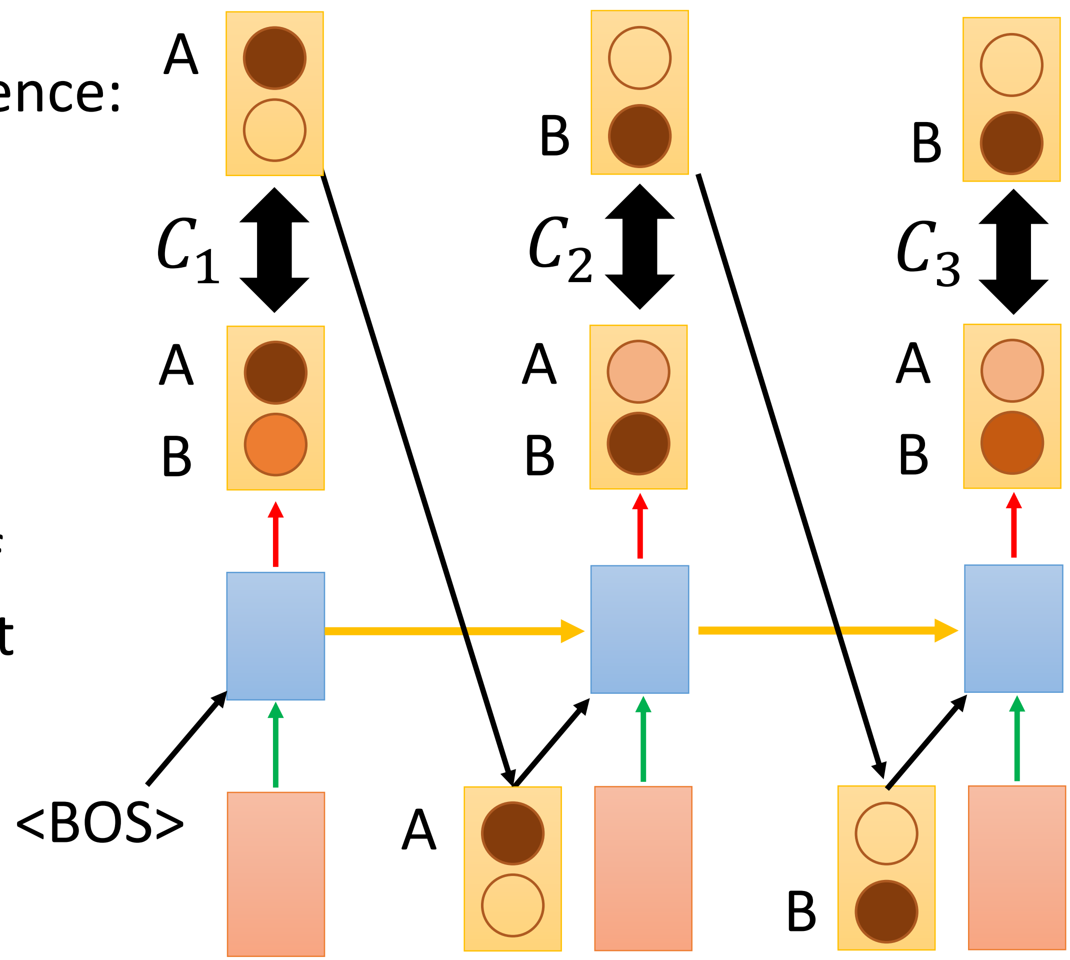
can be different with attention mechanism

Review: Training Generator

Reference:

$$C = \sum_t C_t$$

Minimizing cross-entropy of each component



: condition from decoder

Review: Training Generator

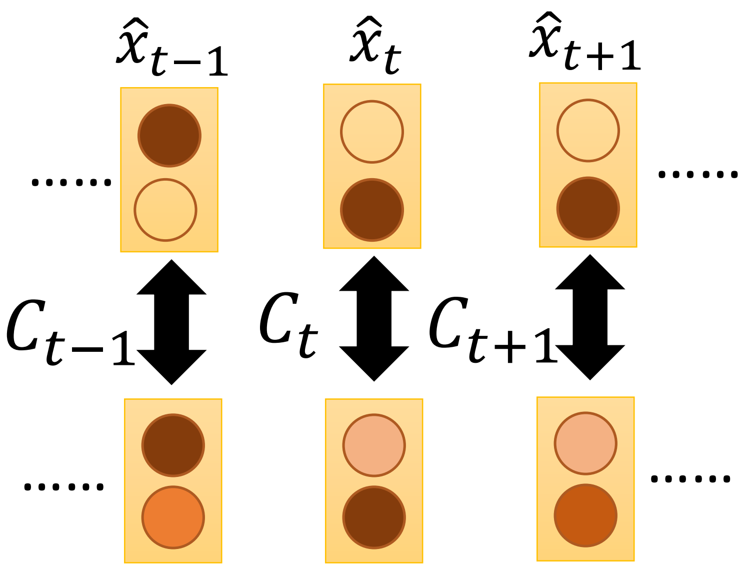
Training data: $(h, \hat{x})$

h: input sentence and history/context

$\hat{x}$: correct response (word sequence)

$\hat{x}_t$: t-th word, $\hat{x}_{1:t}$: first t words of $\hat{x}$

$$C = \sum_t C_t$$



generator output

$$C_t = -logP_\theta(\hat{x}_t|\hat{x}_{1:t-1}, h)$$

$$C = -\sum_t logP(\hat{x}_t|\hat{x}_{1:t-1}, h)$$

$$= -logP(\hat{x}_1|h)P(\hat{x}_t|\hat{x}_{1:t-1}, h)$$

$$\cdots P(\hat{x}_T|\hat{x}_{1:T-1}, h)$$

$$= -logP(\hat{x}|h)$$

Maximizing the likelihood of generating $\hat{x}$ given h

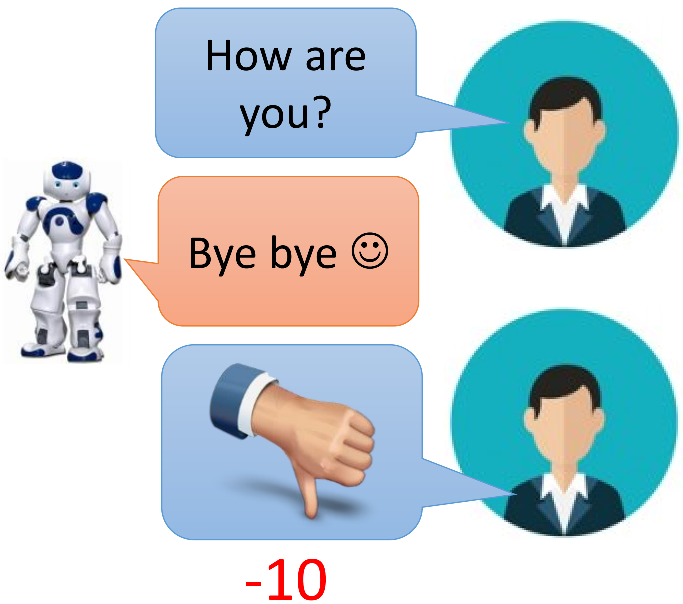
RL for
Sentence Generation

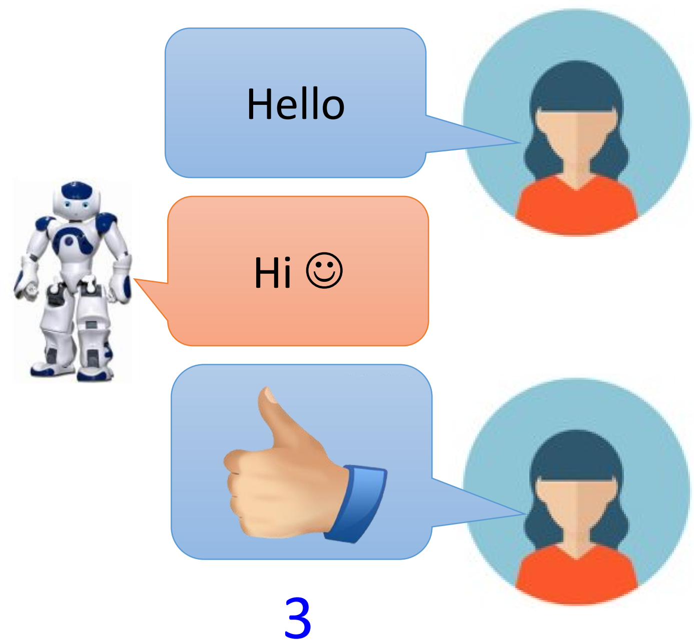
Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, Dan Jurafsky, "Deep Reinforcement Learning for Dialogue Generation", EMNLP 2016 https://image.freepik.com/free-vector/variety-of-human-avatars_23-2147506285.jpg

http://www.freepik.com/free-vector/variety-of-human-avatars_766615.htm

Introduction

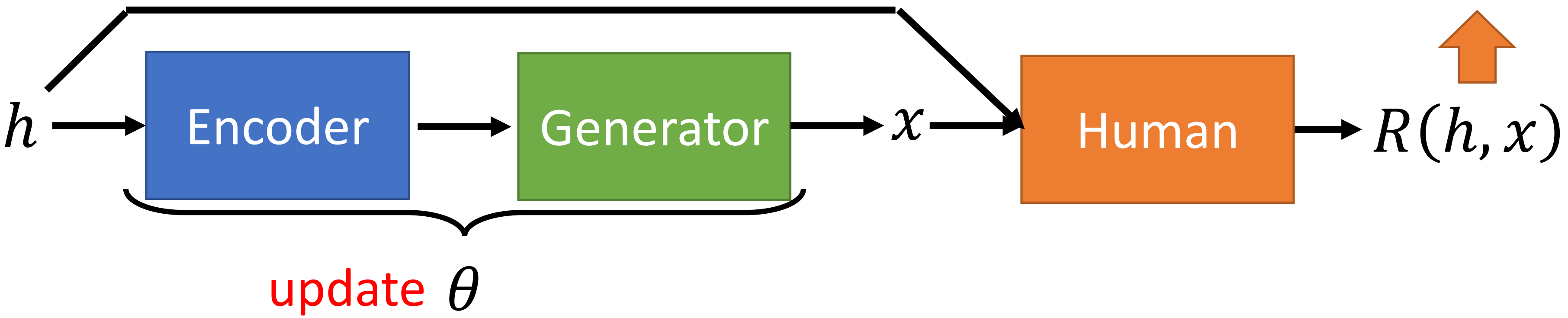
- Machine obtains feedback from user





- Chat-bot learns to maximize the ***expected reward***

Maximizing Expected Reward



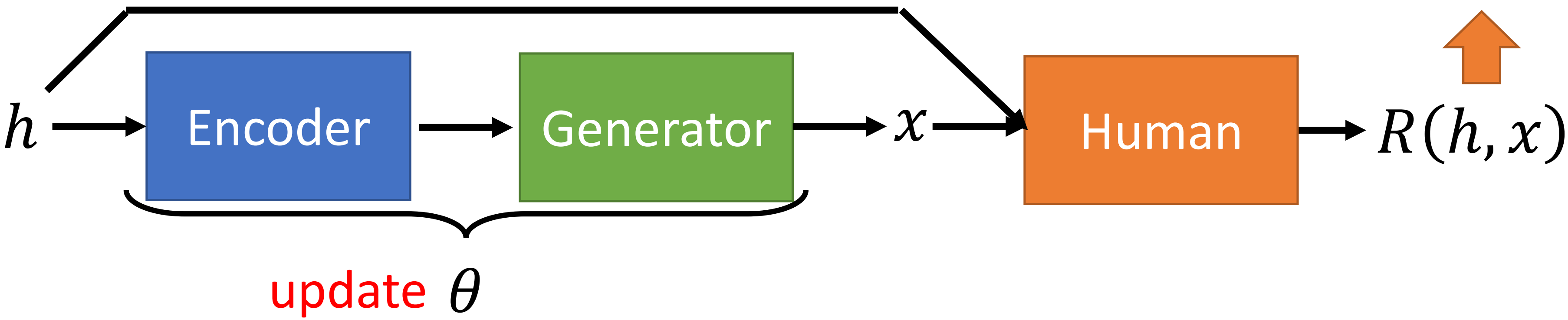
$\theta^* = arg\max_{\theta} \bar{R}_\theta$ Maximizing expected reward

$$\bar{R}_\theta = \sum_h P(h) \sum_x R(h,x)\, P_\theta(x|h)$$

Randomness in generator

Probability that the input/history is h

Maximizing Expected Reward



$$\theta^* = arg\max_{\theta} \bar{R}_\theta$$ Maximizing expected reward

$$\bar{R}_\theta = \sum_h P(h) \sum_x R(h,x)\, P_\theta(x|h) = E_{h\sim P(h)}\left[E_{x\sim P_\theta(x|h)}[R(h,x)]\right]$$

$$= E_{h\sim P(h),x\sim P_\theta(x|h)}[R(h,x)] \approx \frac{1}{N}\sum_{i=1}^{N} R\left(h^i, x^i\right)$$

Sample: $(h^1, x^1), (h^2, x^2), \cdots, (h^N, x^N)$

Where is θ?

Policy Gradient

$$\boxed{\frac{dlog\big(f(x)\big)}{dx} = \frac{1}{f(x)}\frac{df(x)}{dx}}$$

$$\bar{R}_\theta = \sum_h P(h) \sum_x R(h,x)\, P_\theta(x|h) \approx \frac{1}{N}\sum_{i=1}^{N} R\big(h^i, x^i\big)$$

$$\nabla \bar{R}_\theta = \sum_h P(h) \sum_x R(h,x) \nabla P_\theta(x|h) \quad \approx \frac{1}{N}\sum_{i=1}^{N} R\big(h^i, x^i\big) \nabla log P_\theta(x|h)$$

Sampling

$$= \sum_h P(h) \sum_x R(h,x) P_\theta(x|h) \boxed{\frac{\nabla P_\theta(x|h)}{P_\theta(x|h)}}$$

$$= \sum_h P(h) \sum_x R(h,x) P_\theta(x|h) \boxed{\nabla log P_\theta(x|h)}$$

$$= E_{h\sim P(h), x\sim P_\theta(x|h)}[R(h,x)\nabla log P_\theta(x|h)]$$

Policy Gradient

- Gradient Ascent

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_{\theta} \approx \frac{1}{N} \sum_{i=1}^{N} R\left(h^i, x^i\right) \nabla log P_{\theta}\left(x^i | h^i\right)$$

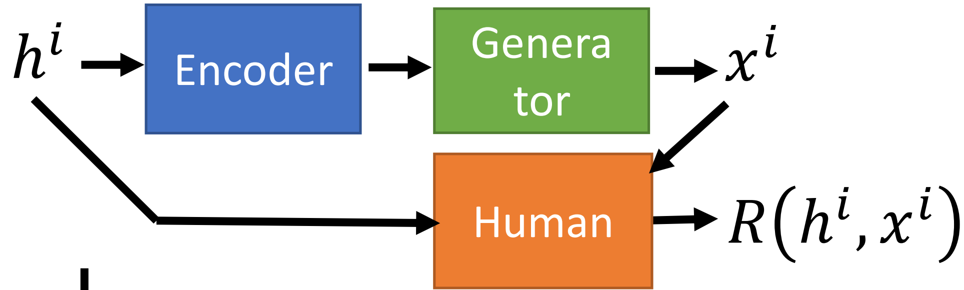
$R\left(h^i, x^i\right)$ is positive

➡ After updating θ, $P_{\theta}\left(x^i | h^i\right)$ will increase

$R\left(h^i, x^i\right)$ is negative

➡ After updating θ, $P_{\theta}\left(x^i | h^i\right)$ will decrease

Implementation



	Maximum Likelihood	Reinforcement Learning
Objective Function	$\frac{1}{N}\sum_{i=1}^{N} logP_\theta(\hat{x}^i \vert h^i)$	$\frac{1}{N}\sum_{i=1}^{N} R(h^i, x^i) logP_\theta(x^i \vert h^i)$
Gradient	$\frac{1}{N}\sum_{i=1}^{N} \nabla logP_\theta(\hat{x}^i \vert h^i)$	$\frac{1}{N}\sum_{i=1}^{N} R(h^i, x^i) \nabla logP_\theta(x^i \vert h^i)$
Training Data	$\{(h^1, \hat{x}^1), \dots, (h^N, \hat{x}^N)\}$ $R(h^i, \hat{x}^i) = 1$	$\{(h^1, x^1), \dots, (h^N, x^N)\}$ Sampling as training data weighted by $R(h^i, x^i)$

Implementation

θ^0 can be well pre-trained from $\{(h^1, \hat{x}^1), \ldots, (h^N, \hat{x}^N)\}$

θ^t

(h^1, x^1) $R(h^1, x^1)$

(h^2, x^2) $R(h^2, x^2)$

$\vdots$ $\vdots$

(h^N, x^N) $R(h^N, x^N)$

New Objective:

$$\frac{1}{N}\sum_{i=1}^{N} R(h^i, x^i) log P_\theta(x^i | h^i)$$

$$\theta^{t+1} \leftarrow \theta^t + \eta \boxed{\nabla \bar{R}_{\theta^t}}$$

$$\frac{1}{N}\sum_{i=1}^{N} R(h^i, x^i) \nabla log P_{\theta^t}(x^i | h^i)$$

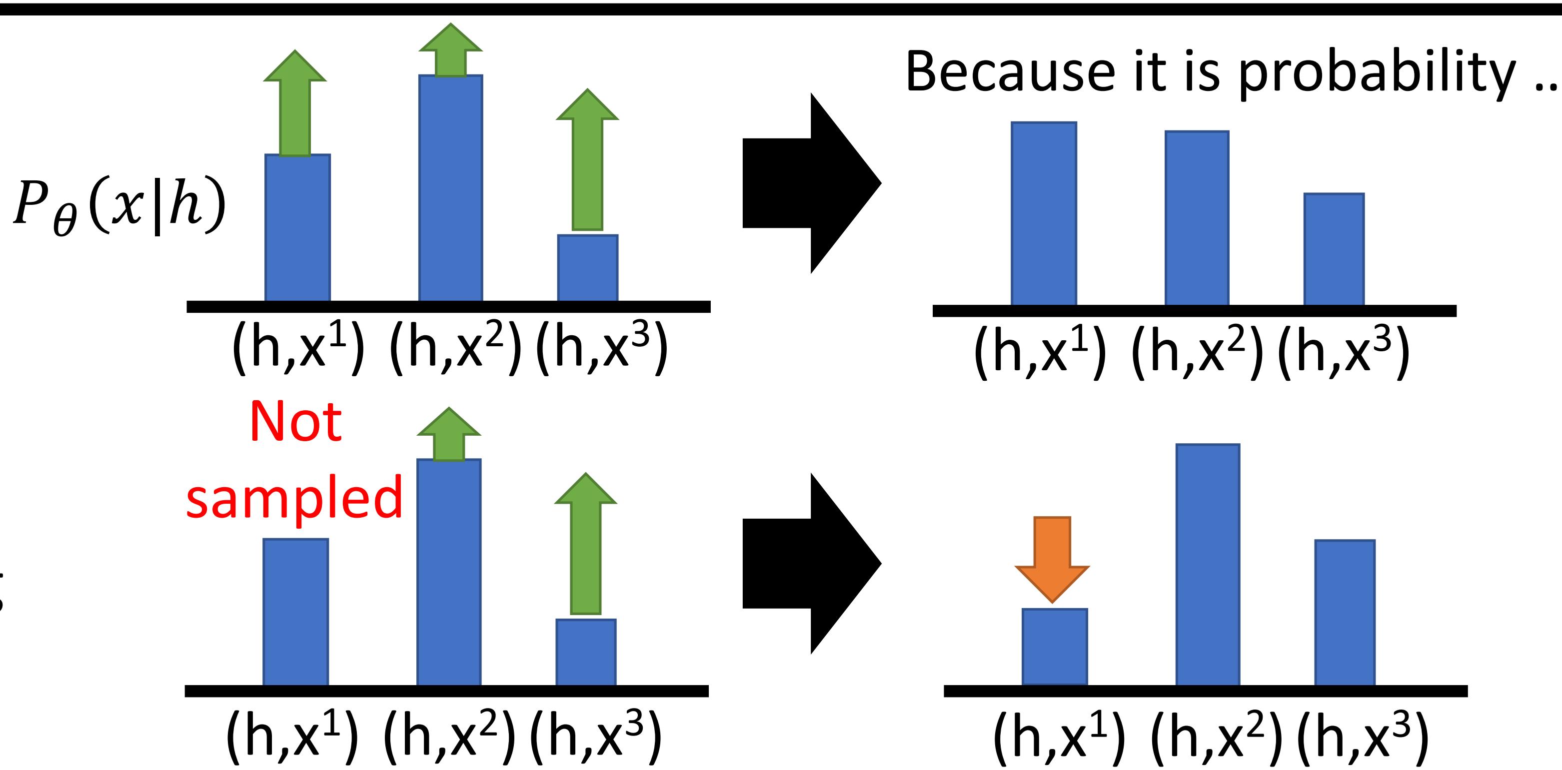
Add a Baseline

If $R\left(h^i, x^i\right)$ is always positive

$$\frac{1}{N}\sum_{i=1}^{N} R\left(h^i, x^i\right) log \nabla P_\theta\left(x^i | h^i\right)$$

Ideal case

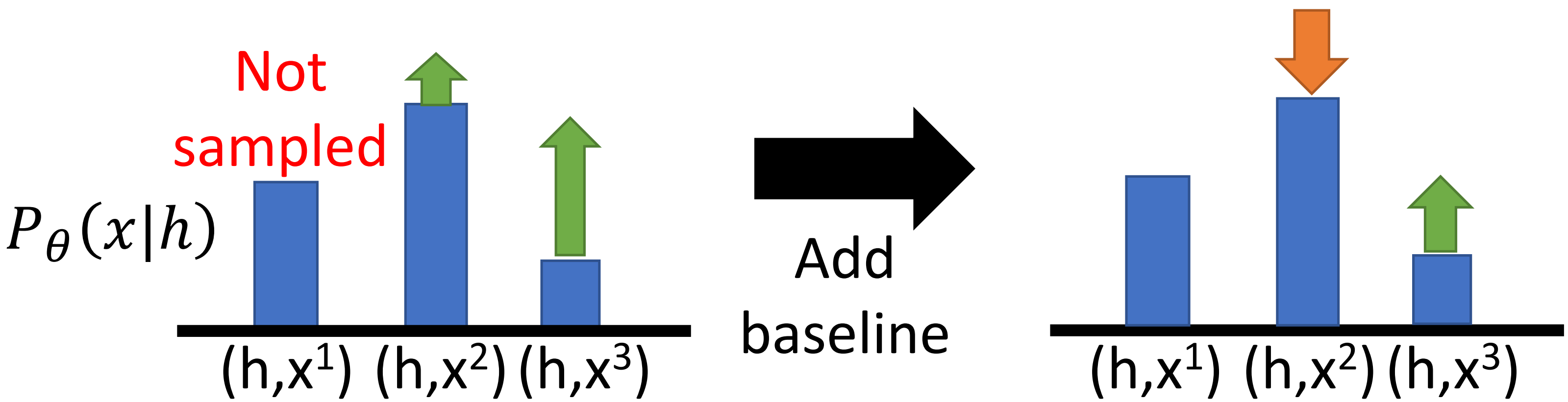
Due to Sampling



Add a Baseline

If $R\left(h^i, x^i\right)$ is always positive

$$\frac{1}{N}\sum_{i=1}^{N} R\left(h^i, x^i\right) log\nabla P_\theta\left(x^i|h^i\right) \Rightarrow \frac{1}{N}\sum_{i=1}^{N}\left(R\left(h^i, x^i\right)\boxed{-b}\right) log\nabla P_\theta\left(x^i|h^i\right)$$



There are several ways to obtain the baseline b.

Alpha GO style training !

- Let two agents talk to each other

How old are you?

See you.

See you.

See you.

How old are you?

I am 16.

I though you were 12.

What make you think so?

Using a pre-defined evaluation function to compute R(h,x)

Example Reward

- The final reward R(h,x) is the weighted sum of three terms $r_1(h,x)$, $r_2(h,x)$ and $r_3(h,x)$

$$R(h,x) = \lambda_1 r_1(h,x) + \lambda_2 r_2(h,x) + \lambda_3 r_3(h,x)$$

Ease of answering

不要成為句點王

Information Flow

說點新鮮的

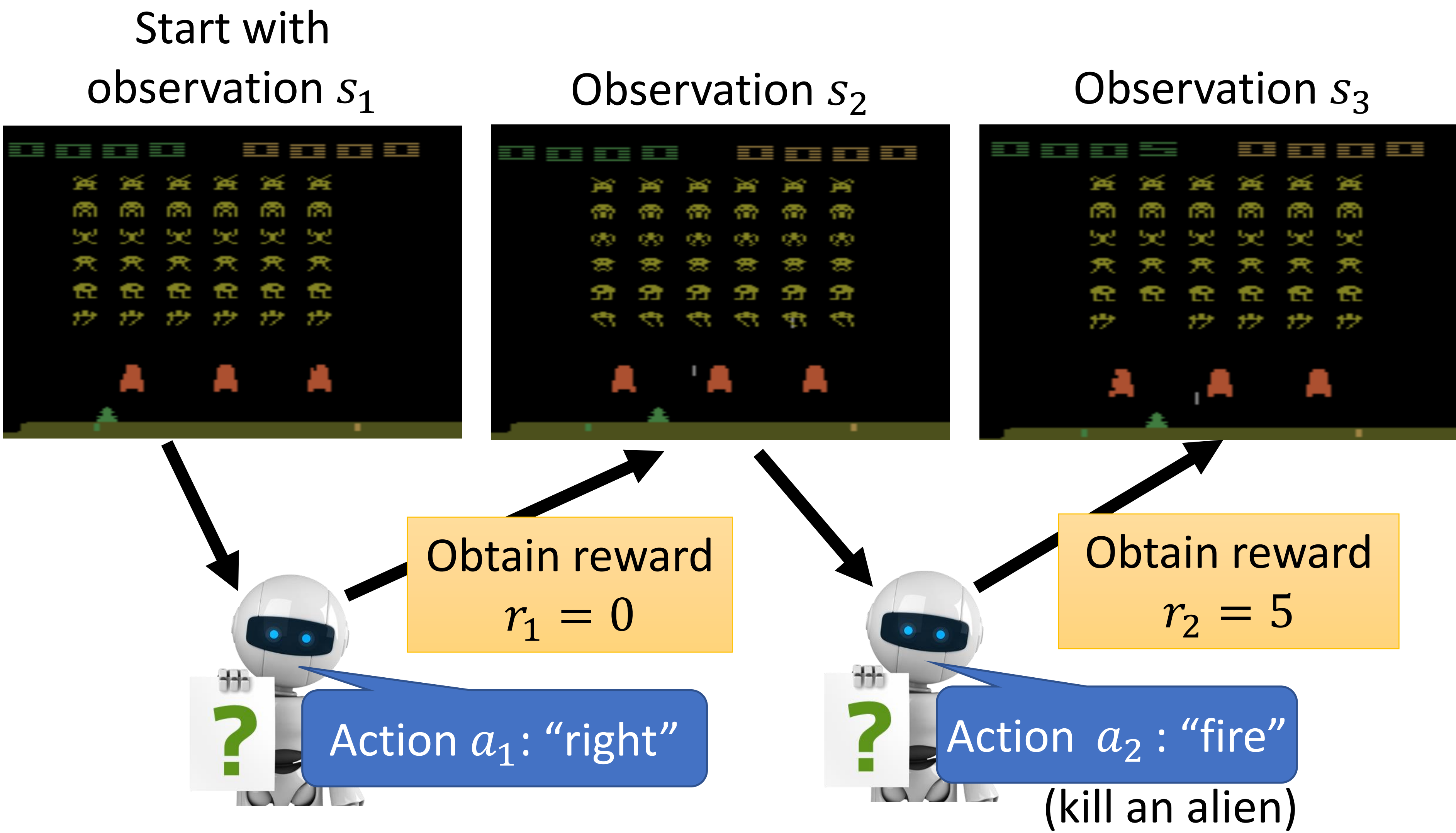
Semantic Coherence

不要前言不對後語

Example Results

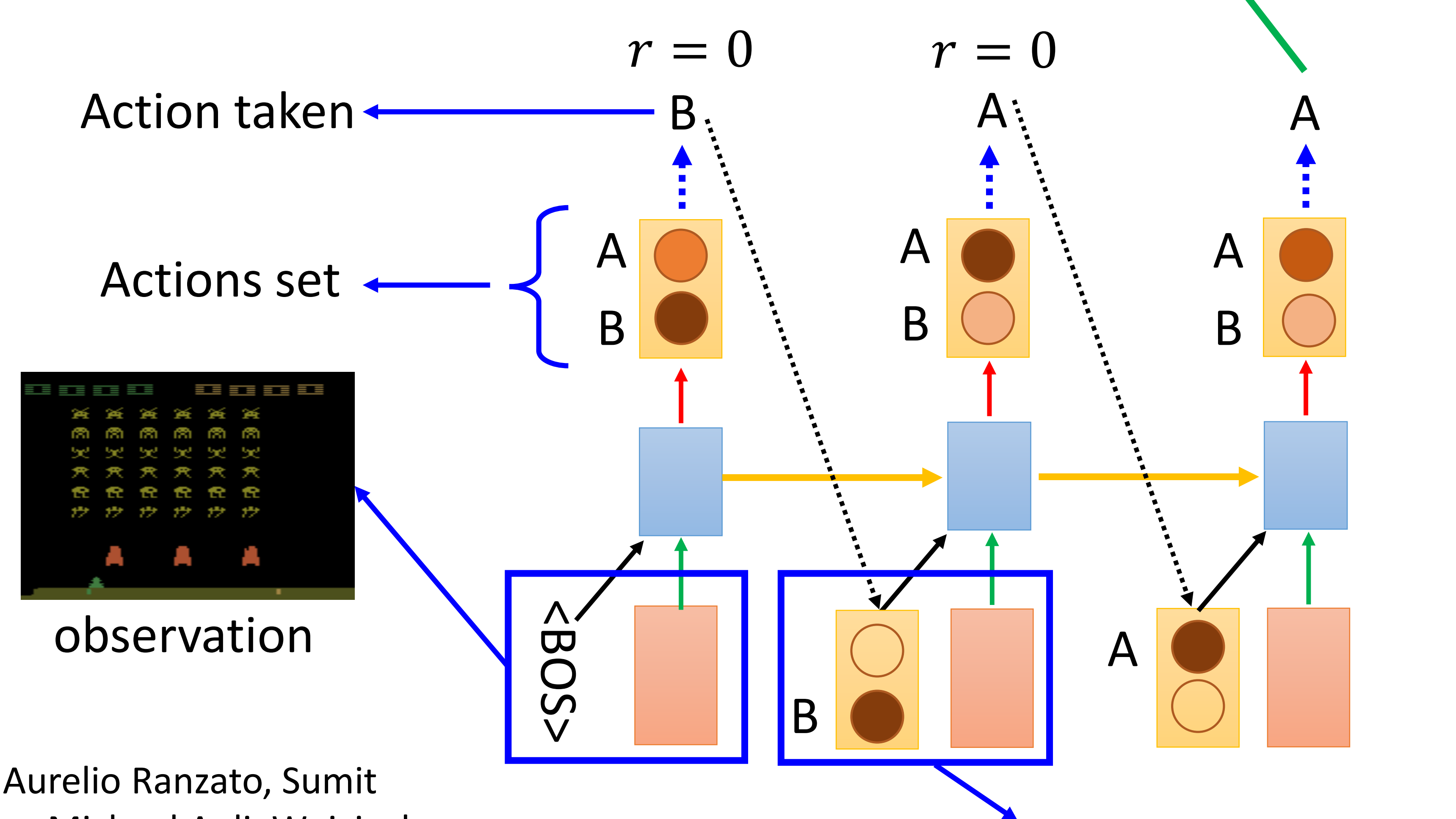
Baseline mutual information model (Li et al. 2015)	Proposed reinforcement learning model
...	...
...	...

Reinforcement learning?



Reinforcement learning?

$reward$:
R(“BAA”, reference)



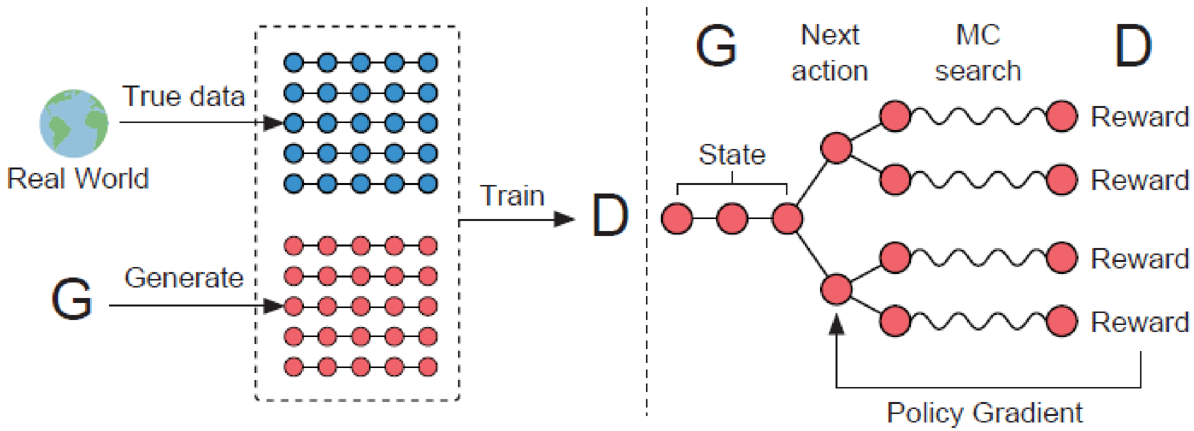
The action we take influence the observation in the next step

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba, “Sequence Level Training with Recurrent Neural Networks”, ICLR, 2016

Reinforcement learning?

- One can use any advanced RL techniques here.
- For example, actor-critic
  - Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, Yoshua Bengio. "An Actor-Critic Algorithm for Sequence Prediction." ICLR, 2017.

SeqGAN



Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu, "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient", AAAI, 2017

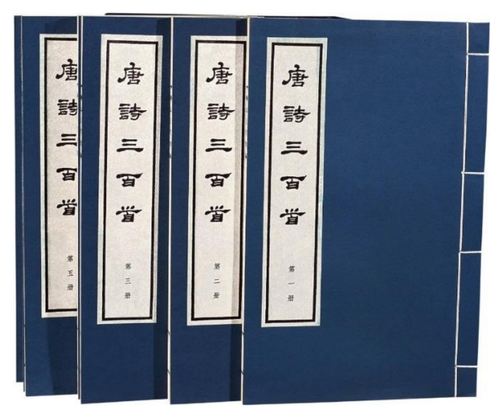
Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, Dan Jurafsky, "Adversarial Learning for Neural Dialogue Generation", arXiv preprint, 2017

Basic Idea – Sentence Generation

code z sampled from prior distribution → **Generator** → sentence x

Sampling from RNN at each time step also provides randomness

sentence x → **Discriminator** → Real or fake



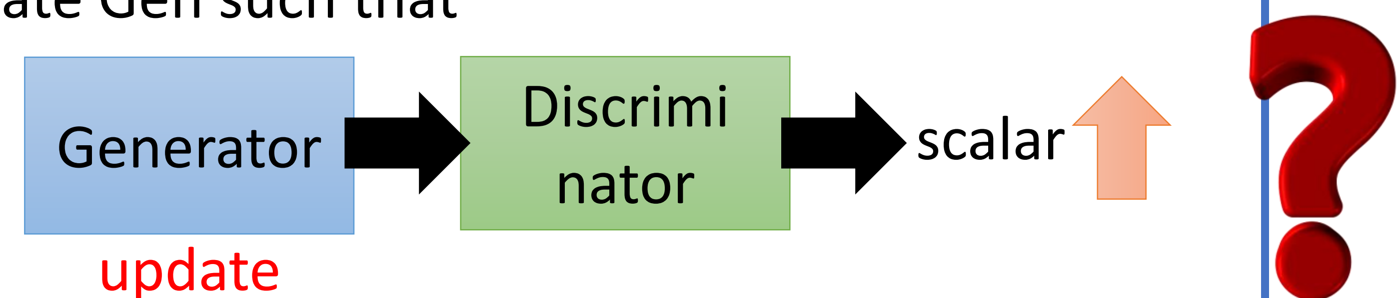
Original GAN

Algorithm – Sentence Generation

- Initialize generator Gen and discriminator Dis
- In each iteration:

  - Sample real sentences x from database
  - Generate sentences $\tilde{x}$ by Gen
  - Update Dis to increase $Dis(x)$ and decrease $Dis(\tilde{x})$

  - Update Gen such that



http://www.nipic.com/show/3/83/3936650kd7476069.html

Basic Idea – Chat-bot

Input sentence/history h → [En | De — Chatbot] → response sentence x

Input sentence/history h → Discriminator → Real or fake

response sentence x → Discriminator

human dialogues → Discriminator

<u>Conditional GAN</u>

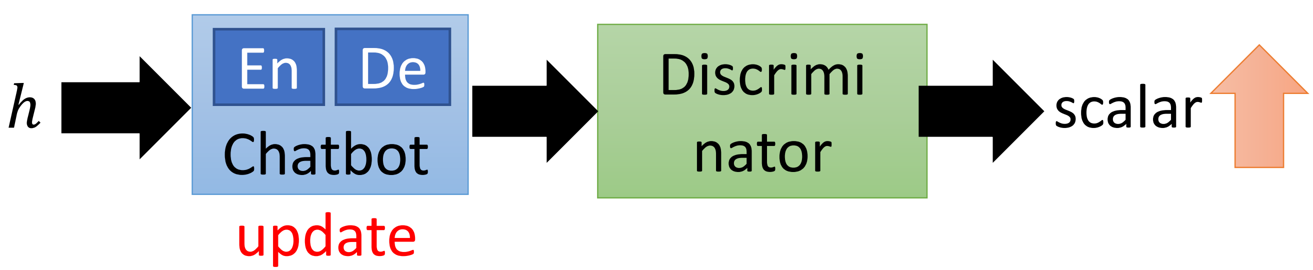
Training data:

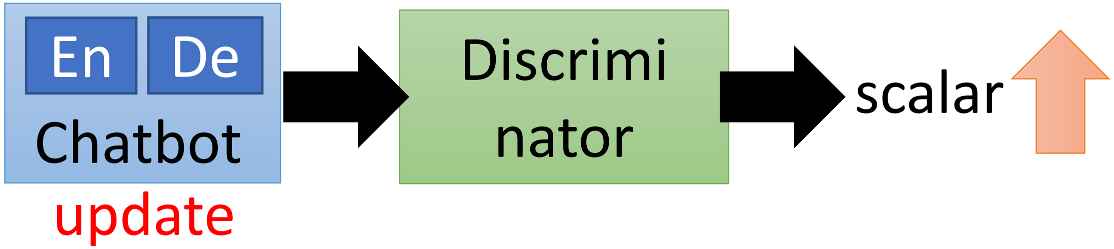
h
A: OOO
B: XXX

x
A: ∆ ∆ ∆

Algorithm – Chat-bot

- Initialize generator Gen and discriminator Dis
- In each iteration:
  - Sample real history h and sentence x from database
  - Sample real history h' from database, and generate sentences $\tilde{x}$ by Gen(h')
  - Update Dis to increase $Dis(h, x)$ and decrease $Dis(h', \tilde{x})$
  - Update Gen such that

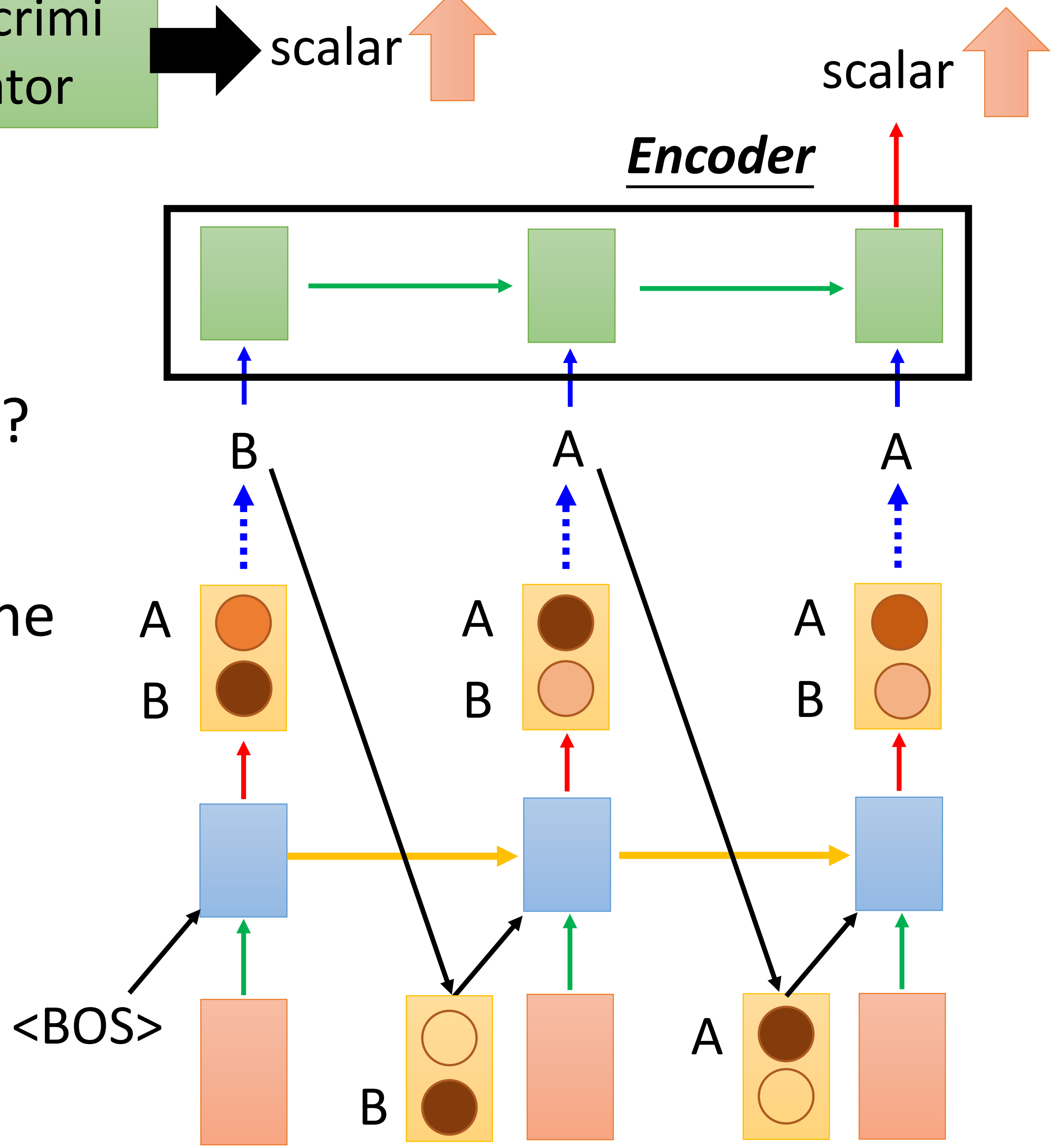




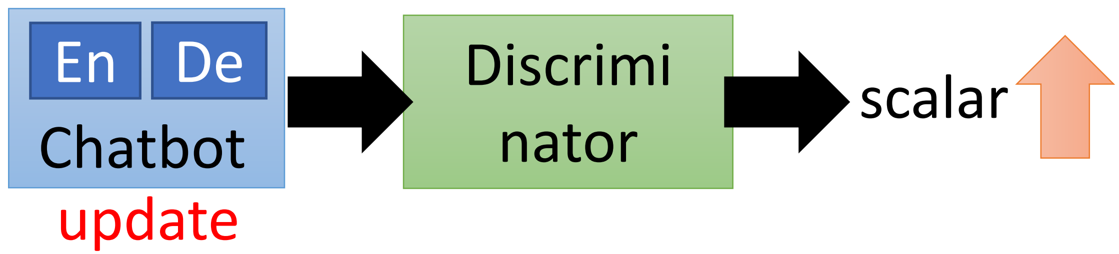
Can we do backpropogation?

Tuning generator will not change the output.

Alternative: improved WGAN



Reinforcement Learning



- Consider the output of discriminator as reward
  - Update generator to increase discriminator = to get maximum reward

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{i=1}^{N} \underbrace{\overbrace{D\left(h^i, x^i\right)}^{\text{reward}}}_{\text{Discriminator Score}} - b\big)\nabla log P_\theta\left(x^i | h^i\right)$$

- Different from typical RL
  - The discriminator would update

Reward for Every Generation Step

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{i=1}^{N}\left(D\left(h^i, x^i\right) - b\right)\nabla log P_\theta\left(x^i | h^i\right)$$

h^i = "What is your name?"

x^i = "I don't know"

$D\left(h^i, x^i\right) - b$ is negative

Update θ to decrease $\log P_\theta\left(x^i | h^i\right)$

$$log P_\theta\left(x^i | h^i\right) = log P\left(x_1^i | h^i\right) + log P\left(x_2^i | h^i, x_1^i\right) + log P\left(x_3^i | h^i, x_{1:2}^i\right)$$

$P\left("I" | h^i\right)$

h^i = "What is your name?"

x^i = "I am John"

$D\left(h^i, x^i\right) - b$ is positive

Update θ to increase $\log P_\theta\left(x^i | h^i\right)$

$$log P_\theta\left(x^i | h^i\right) = log P\left(x_1^i | h^i\right) + log P\left(x_2^i | h^i, x_1^i\right) + log P\left(x_3^i | h^i, x_{1:2}^i\right)$$

$P\left("I" | h^i\right)$

Reward for Every Generation Step

h^i = “What is your name?” x^i = “I don’t know”

$$logP_\theta\left(x^i|h^i\right) = logP\left(x_1^i|h^i\right) + logP\left(x_2^i|h^i, x_1^i\right) + logP\left(x_3^i|h^i, x_{1:2}^i\right)$$

$$P\left("I"|h^i\right) \quad P\left("don't"|h^i, "I"\right) \quad P\left("know"|h^i, "I\ don't"\right)$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(Q\left(h^i, x_{1:t}^i\right) - b\right)\nabla logP_\theta\left(x_t^i|h^i, x_{1:t-1}^i\right)$$

Method 1. Monte Carlo (MC) Search

Method 2. Discriminator For Partially Decoded Sequences

Monte Carlo Search

- How to estimate $Q\left(h^i, x_{1:t}^i\right)$?

$$\underset{h^i}{Q(\text{"}What\ is\ your\ name?\text{"}}, \underset{x_1^i}{\text{"}I\text{"}})$$

A roll-out generator for sampling is needed

$x^A =$ I am John $\quad D\left(h^i, x^A\right) = 1.0$

$x^B =$ I am happy $\quad D\left(h^i, x^B\right) = 0.1$

$x^C =$ I don't know $\quad D\left(h^i, x^C\right) = 0.1$

$x^D =$ I am superman $\quad D\left(h^i, x^D\right) = 0.8$

avg

$Q\left(h^i, \text{"}I\text{"}\right) = 0.5$

Rewarding Partially Decoded Sequences

- Training a discriminator that is able to assign rewards to both fully and partially decoded sequences
  - Break generated sequences into partial sequences

h="What is your name?", x="I am john"

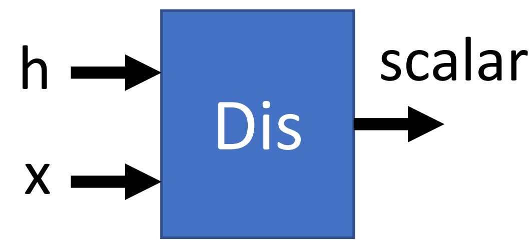
h="What is your name?", x="I am"

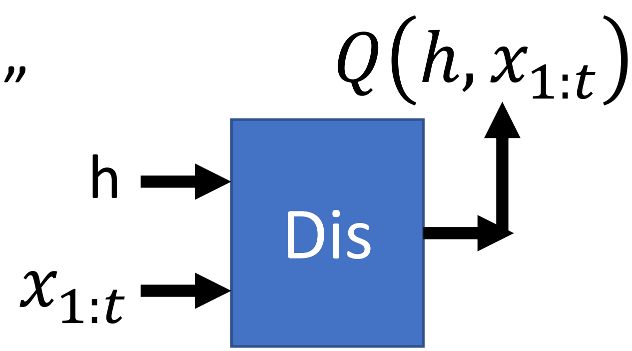
h="What is your name?", x="I"

h="What is your name?", x="I don't know"

h="What is your name?", x="I don't"

h="What is your name?", x="I"





Teacher Forcing

- The training of generative model is unstable
  - This reward is used to promote or discourage the generator's own generated sequences.
  - Usually It knows that the generated results are bad, but does not know what results are good.
- Teacher Forcing

Training Data for SeqGAN: $\{(h^1, x^1), \ldots, (h^N, x^N)\}$

Obtained by sampling

weighted by $D(h^i, x^i)$

Adding more Data: $\{(h^1, \hat{x}^1), \ldots, (h^N, \hat{x}^N)\}$ Real data

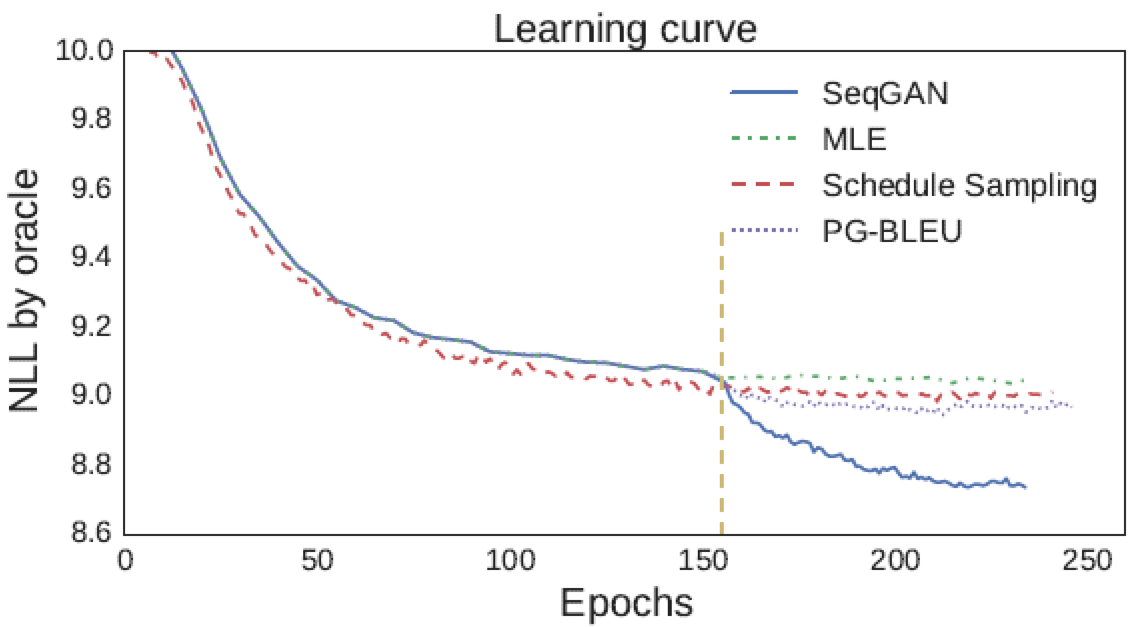
Consider $D(h^i, \hat{x}^i) = 1$

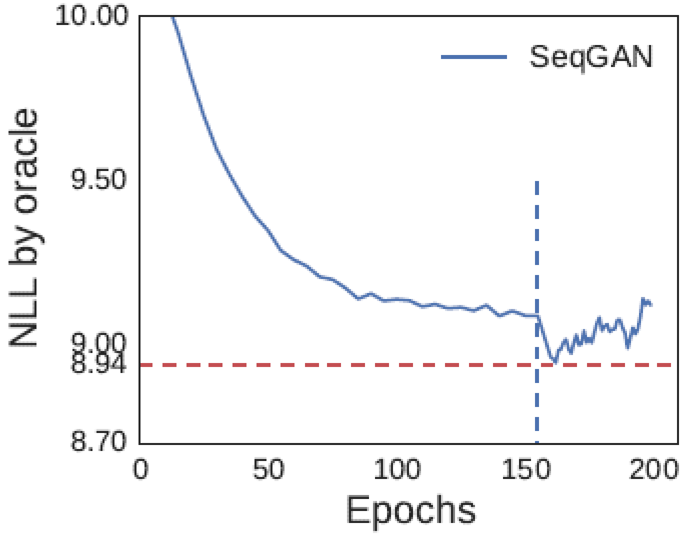
Experiments in paper

- Sentence generation: Synthetic data
  - Given an LSTM
  - Using the LSTM to generate a lot of sequences as "real data"
  - Generator learns from the "real data" by different approaches
  - Generator generates some sequences
  - Using the LSTM to compute the negative log likelihood (NLL) of the sequences
    - Smaller is better

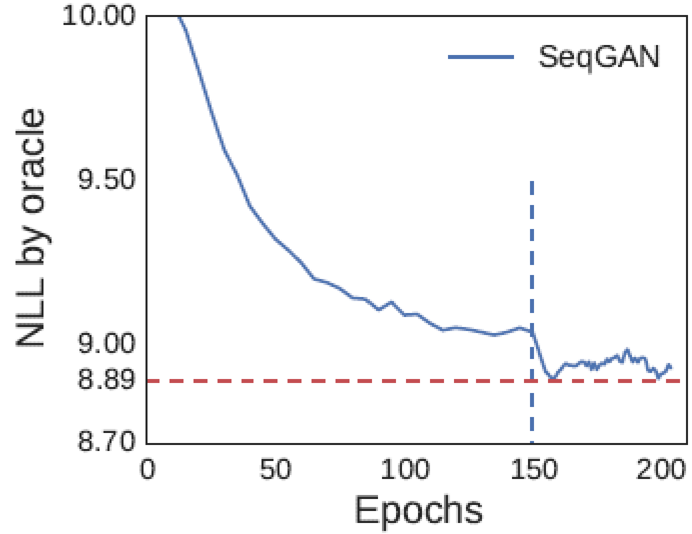
Experiments in paper
- Synthetic data

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	**8.736**
p-value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	

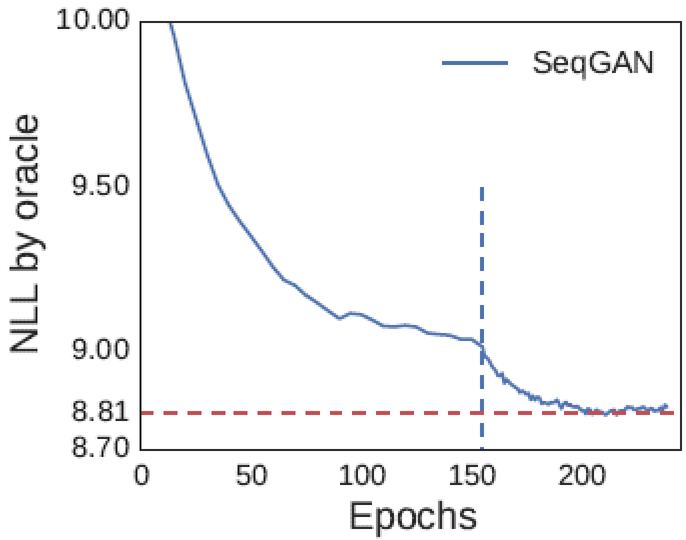




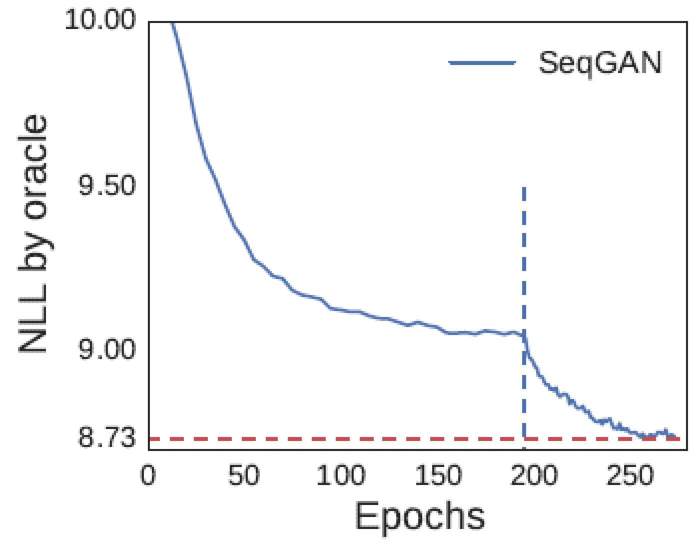
(a) g-steps=100, d-steps=1, k=10



(b) g-steps=30, d-steps=1, k=30



(c) g-steps=1, d-steps=1, k=10



(d) g-steps=1, d-steps=5, k=3

Experiments in paper - Real data

Table 2: Chinese poem generation performance comparison.

Algorithm	Human score	p-value	BLEU-2	p-value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	**0.5356**		**0.7389**	
Real data	0.6011		0.746	

Table 3: Obama political speech generation performance.

Algorithm	BLEU-3	p-value	BLEU-4	p-value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	**0.556**		**0.427**	

Table 4: Music generation performance comparison.

Algorithm	BLEU-4	p-value	MSE	p-value
MLE	0.9210	$< 10^{-6}$	22.38	0.00034
SeqGAN	**0.9406**		**20.62**	

Results - Chat-bot

Setting	adver-win	adver-lose
single-turn	0.62	0.18
multi-turn	0.72	0.10

Input	tell me ... how long have you had this falling sickness ?
Vanilla-MLE	i 'm not a doctor .
Vanilla-Sample	well everything you did was totally untrue .
REINFORCE	i don 't know how long it 's been .
REGS Monte Carlo	A few months, I guess .
Input	so i had the doctors test sammy 's response to conditioning .
Vanilla-MLE	sammy wrote the test sammy wrote the test .
Vanilla-Sample	objects pick objects up objects objects objects objects objects
REINFORCE	i 'm not sure that 's relevant .
REGS Monte Carlo	so he took the pills .
Input	what are you going to do ?
Vanilla-MLE	i 'm going to the movies .
Vanilla-Sample	get him outta here first !
REINFORCE	i 'm going to get you .
REGS Monte Carlo	i 'm going to get you a new car .
Input	they fear your power your intellect .
Vanilla-MLE	you 're the only one who knows what 's going on .
Vanilla-Sample	when they are conquered and you surrender they will control all of us .
REINFORCE	i 'm afraid i 'm not ready yet .
REGS Monte Carlo	i 'm not afraid of your power .

To Learn More …

Algorithm – MaliGAN

Maximum-likelihood Augmented Discrete GAN

- Initialize generator Gen and discriminator Dis
- In each iteration:
  - Sample real sentences x from database
  - Generate sentences $\tilde{x}$ by Gen
  - Update Dis to maximize

$$\sum_{x} logD(x) + \sum_{\tilde{x}} log\left(1 - D(\tilde{x})\right)$$

  - Update Gen by gradient

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{r_D\left(x^i\right)}{\sum_{i=1}^{N} r_D\left(x^i\right)} - b\right)\nabla logP_\theta\left(x^i\right)$$

$$D\left(h^i, x^i\right)$$

$$r_D\left(x^i\right) = \frac{D\left(x^i\right)}{1 - D\left(x^i\right)}$$

To learn more ……

- Professor forcing
  - Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, Yoshua Bengio, “Professor Forcing: A New Algorithm for Training Recurrent Networks”, NIPS, 2016
- Handling discrete output by methods other than policy gradient
  - MaliGAN, Boundary-seeking GAN
  - Yizhe Zhang, Zhe Gan, Lawrence Carin, “Generating Text via Adversarial Training”, Workshop on Adversarial Training, NIPS, 2016
  - Matt J. Kusner, José Miguel Hernández-Lobato, “GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution”, arXiv preprint, 2016