

## Lecture 14: RL outside the games and SCST

**Radoslav Neychev**

Harbour.Space University  
24.06.2020, Barcelona, Spain

# References

These slides are almost the exact copy of Practical RL course week 7 slides.  
Special thanks to YSDA team for making them publicly available.

Original slides link: [week07\\_seq2seq](#)

# General formalism

- Maximize  $J = \underset{\substack{s \sim p(s) \\ a \sim \pi(a|obs(s))}}{E} R(s, a)$  over  $\pi$

# General formalism

- Maximize  $J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} R(s, a)$  over  $\pi$
- Reward  $R(s, a)$  is a black box
  - Special case:  $G(s, a) = r(s, a) + \gamma G(s', a')$

# General formalism

- Maximize  $J = \mathbb{E}_{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}} R(s, a)$  over  $\pi$
- Reward  $R(s, a)$  is a black box
  - Special case:  $G(s, a) = r(s, a) + \gamma G(s', a')$
- Markov property:  $P(s'|s, a, *) = P(s'|s, a)$
- Special case:  $obs(s) = s$ , fully observable

# General approaches

## Idea 1: evolution strategies

- pertrubate  $\pi$ , take ones with higher  $J$

## Idea 2: value-based methods

- **estimate  $J$**  as a function of  $a$ , pick best  $a$

## Idea 3: policy gradient

- **ascend  $J$**  over  $\pi(a|s)$  using  $\nabla J$

# General approaches

## Idea 4: Bayesian optimization

- build a model of  $J$ , pick  $\pi$  that is most informative to finding maximal  $J$
- e.g. Gaussian processes (low-dimensional only)

## Idea 5: simulated annealing

## Idea 6: crossentropy method

...

# Application domains

- Videogames
- Online ads
- Recommender systems
- Conversation systems
- Robot control / dynamic system control
- Parameter tuning
- Financial tasks
- Medicine
- ...

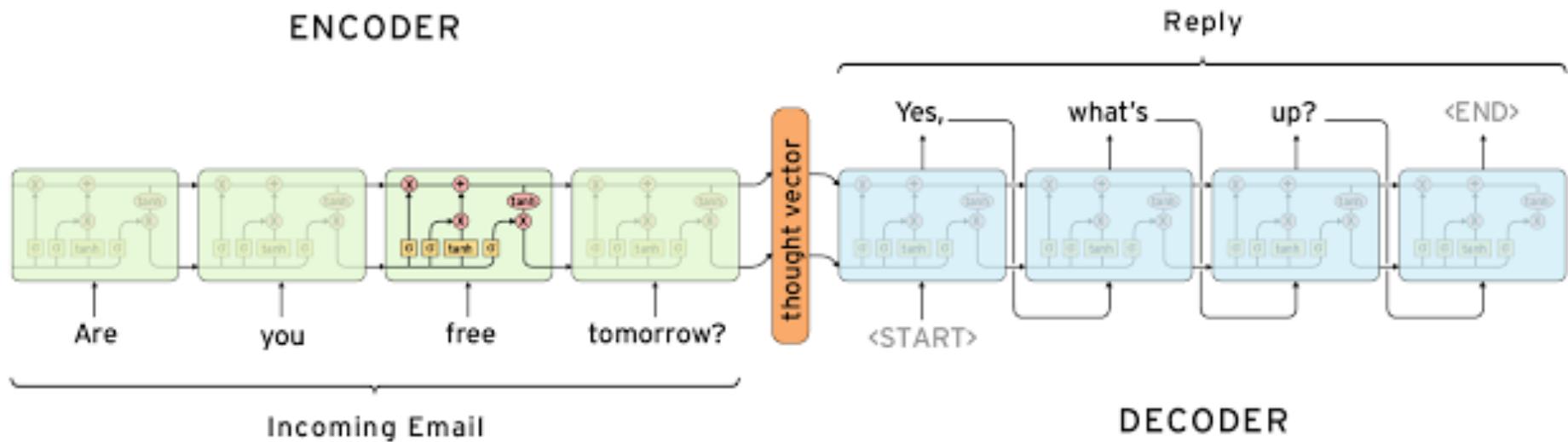
# Domains so far

- Videogames
- ~~Online ads~~ toy problems
- ~~Recommender systems~~ videogames
- ~~Conversation systems~~ toy problems
- ~~Robot control / dynamic system control~~
- ~~Parameter tuning~~ videogames
- ~~Financial tasks~~ toy problems
- ~~Medicine~~ guess what?
- ...

# Encoder-decoder architectures

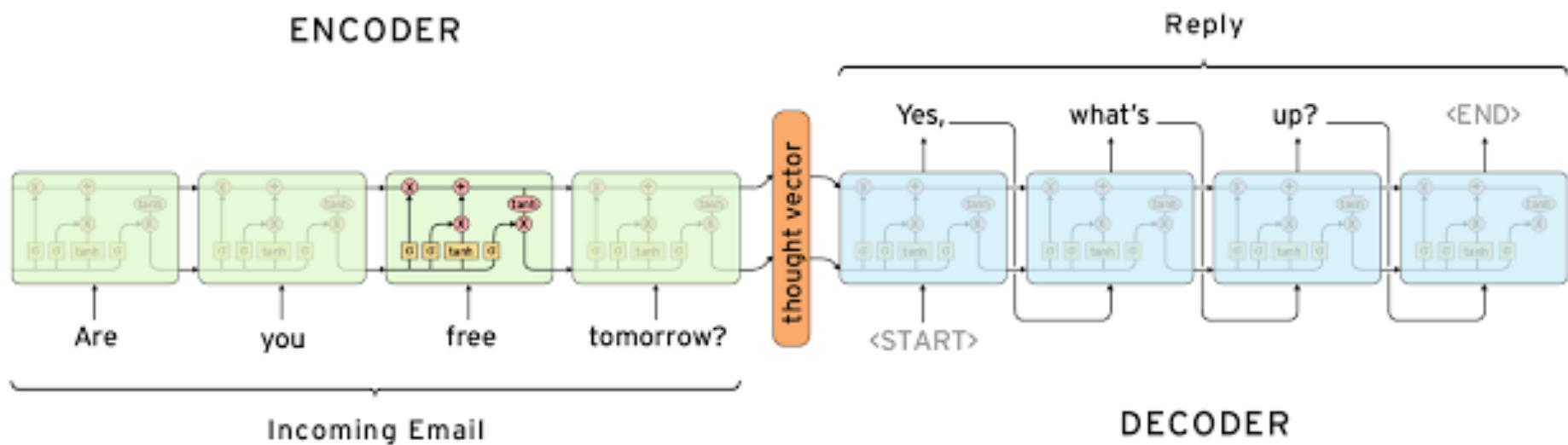
- Read input data (sequence / arbitrary)
- Generate output sequence

Q: what problems match this formulation?



# Encoder-decoder tasks

- Machine translation
- Image to caption
- Word to transcript
- Conversation system
- Image to latex
- Code to docstring



# Machine translation

## **Problem:**

- Read sentence in Chinese
- Generate sentence in English
- Sentences must mean the same thing

## Solution?

# Machine translation

## **Problem:**

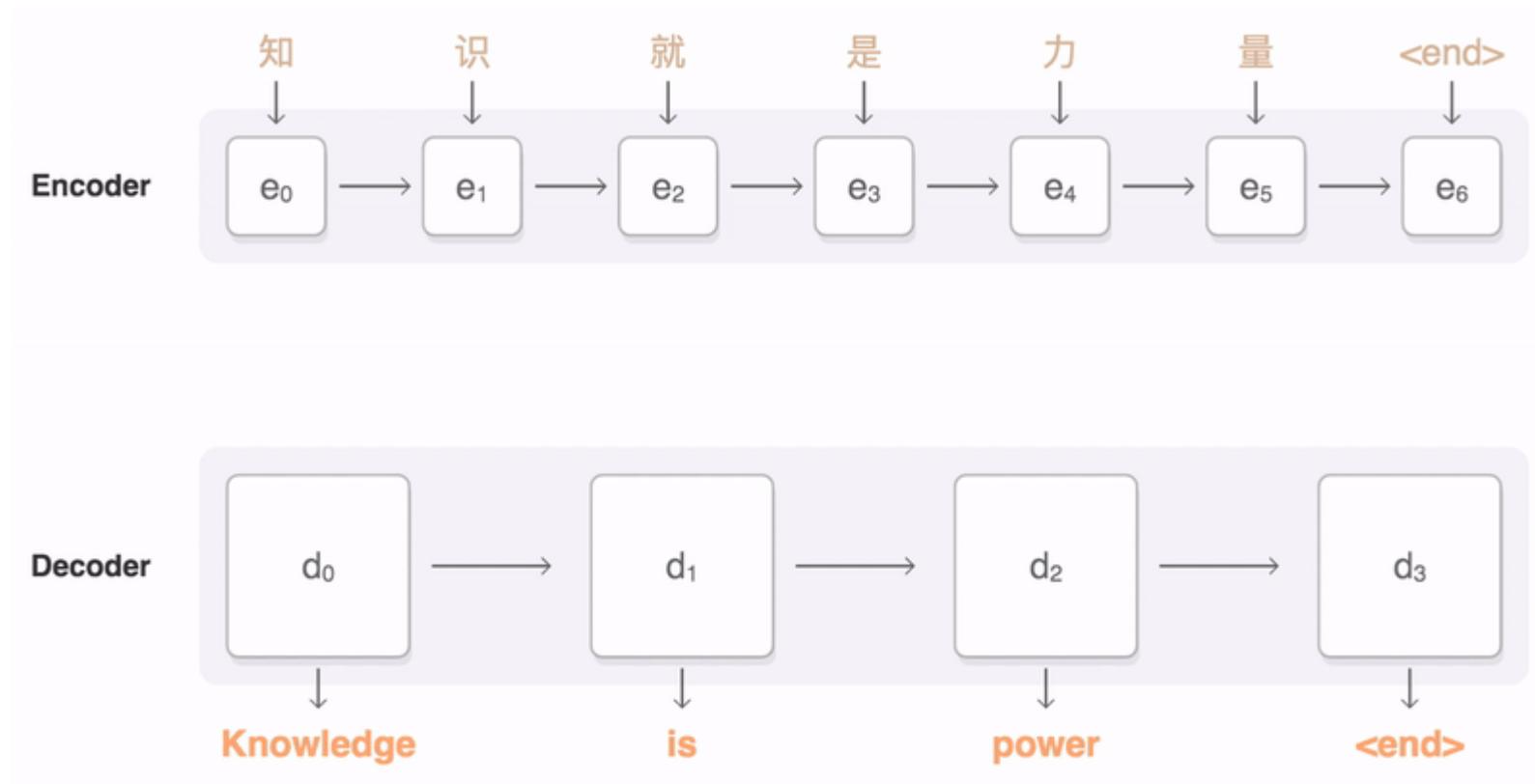
- Read sentence in Chinese
- Generate sentence in English
- Sentences must mean the same thing

## **Solution:**

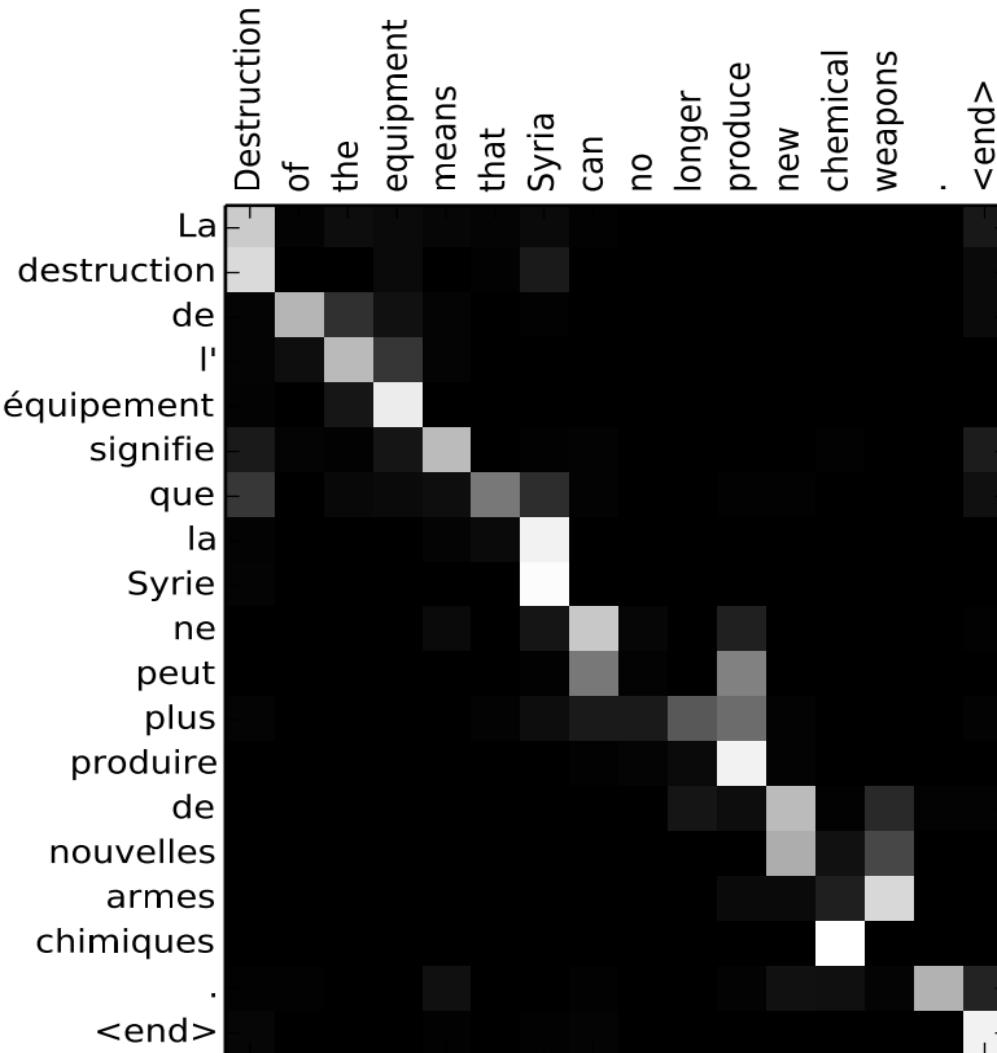
- Take large dataset of (source,translation) pairs
- Maximize  $\log P(\text{translation}|\text{source})$

# Digression: attentive translation

Let decoder choose where to look on each tick



# Digression: attentive translation



Simultaneously learns

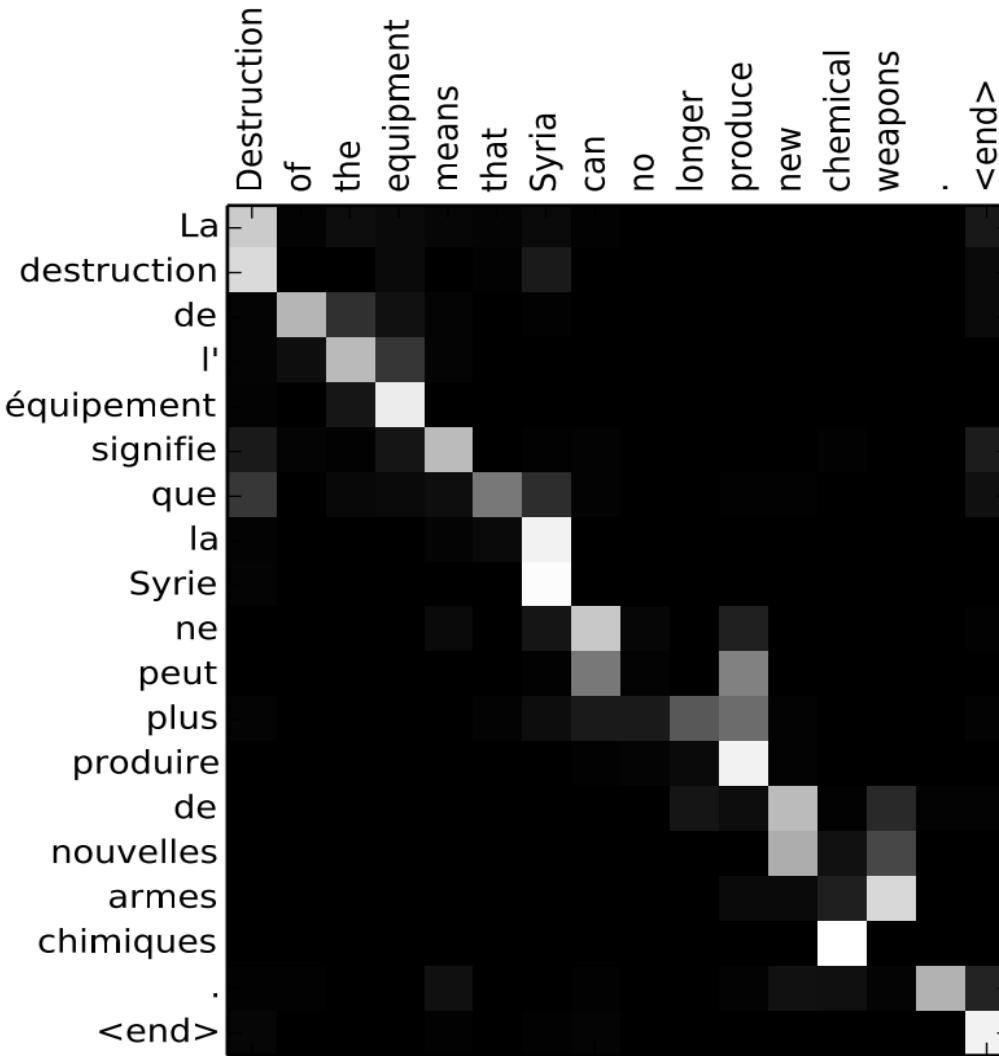
- Word alignment
- Word translation

Differentiable attention:

$$\bar{a} = W \cdot \bar{h} + \bar{b}$$

$$inp = \langle \bar{x}, \text{softmax}(\bar{a}) \rangle$$

# Digression: attentive translation



Simultaneously learns

- Word alignment
- Word translation

Differentiable attention:

$$\bar{a} = W \cdot \bar{h} + \bar{b}$$

$$inp = \frac{\sum_i x_i \cdot e^{a_i}}{\sum_j e^{a_j}}$$

# Machine translation, again

## **Problem:**

- Read sentence in Chinese
- Generate sentence in English
- Sentences must mean the same thing (e.g. *BLEU*)

## **Solution:**

- Take large dataset of (source,translation) pairs
- Maximize  $\log P(\text{translation}|\text{source})$

# Conversation systems

## **Problem:**

- Read sentence from user
- Generate response sentence
- System must be able to support conversation

## **Solution:**

- Take large dataset of (phrase,response) pairs
- Maximize  $\log P(\text{response}|\text{phrase})$

# Grapheme to phoneme

## Problem:

- Read word (characters): “**hedgehog**”
- Generate transcript (phonemes): “**hɛ̃hag**”
- Transcript must read like real word (Levenshtein)

## Solution:

- Take large dataset of (word,transcript) pairs
- Maximize  $\log P(\text{transcript}|\text{word})$

# Yet another problem

## Problem:

- Read  $x \sim X$
- Produce answer  $y \sim Y$
- Answer should be  $\text{argmax } R(x, y)$

## Solution:

- Take large dataset of  $(x, y)$  pairs with *good*  $R(x, y)$
- Maximize  $\log P(y|x)$  over those pairs

# Summary

Works great as long as you have good data!

good = abundant + near-optimal  $R(x,y)$

What could possibly go wrong?

# Distribution shift

Supervised seq2seq learning:

$$P(y_{t+1}|x, y_{0:t}), \quad y_{0:t} \sim \text{reference}$$

Inference

$$P(y_{t+1}|x, \hat{y}_{0:t}), \quad \hat{y}_{0:t} \sim \text{???}$$

# Distribution shift

Supervised seq2seq learning:

$$P(y_{t+1}|x, y_{0:t}), \quad y_{0:t} \sim \text{reference}$$

Inference

$$P(y_{t+1}|x, \hat{y}_{0:t}), \quad \hat{y}_{0:t} \sim \text{model}$$

# Distribution shift

Supervised seq2seq learning:

$$P(y_{t+1}|x, y_{0:t}), \quad y_{0:t} \sim \text{reference}$$

Inference

$$P(y_{t+1}|x, \hat{y}_{0:t}), \quad \hat{y}_{0:t} \sim \text{model}$$

If model ever makes something that isn't in data,  
It gets volatile from next time-step!

# Summary

Works great as long as you have good data!

good = abundant + near-optimal  $R(x,y)$

... and a perfect network ...

# Summary

Works great as long as you have good data!

good = abundant + near-optimal  $R(x,y)$

... and a perfect network ...

**Spoiler:** most of the time we don't. Too bad.

# Summary

Works great as long as you have good data!

good = abundant + near-optimal  $R(x,y)$

**Spoiler:** most of the time we don't. Too bad.



# Machine translation issues

There's more than one correct translation.

**Source:** 在 找 给 家 里 人 的 礼 物 .

**Versions:**

i 'm searching for some gifts for my family.  
i want to find something for my family as presents.  
i 'm about to buy some presents for my family.  
i 'd like to buy my family something as a gift.  
i 'm looking for a present for my family.  
...

( Sample from IWSLT 2009 Ch-En, <http://bit.ly/2o404Tz> )

# Machine translation issues

There's more than one correct translation.  
You don't need to learn all of them.

**Source:** 在 找 给 家 里 人 的 礼 物 .

**Versions:**

i 'm searching for some gifts for my family.  
i want to find something for my family as presents.  
i 'm about to buy some presents for my family.  
i 'd like to buy my family something as a gift.  
i 'm looking for a present for my family.  
...

( Sample from IWSLT 2009 Ch-En, <http://bit.ly/2o404Tz> )

# Machine translation issues

There's more than one correct translation.  
You don't need to learn all of them.

**Source:** 在 找 给 家 里 人 的 礼 物 .

<b>Versions:</b>	Model 1 $p(y x)$	Model 2 $p(y x)$
(version 1)	1e-2	0.99
(version 2)	2e-2	1e-100
(version 3)	1e-2	1e-100
(all rubbish)	0.96	0.01

# Machine translation issues

There's more than one correct translation.  
You don't need to learn all of them.

**Source:** 在 找 给 家 里 人 的 礼 物 .

<b>Versions:</b>	Model 1 $p(y x)$	Model 2 $p(y x)$
(version 1)	1e-2	0.99
(version 2)	2e-2	1e-100
(version 3)	1e-2	1e-100
(all rubbish)	0.96	0.01

not in data

Q: which model has better  
Mean log  $p(y|x)$  ?

# Machine translation issues

There's more than one correct translation.  
You don't need to learn all of them.

**Source:** 在 找 给 家 里 人 的 礼 物 .

<b>Versions:</b>	Model 1 $p(y x)$	Model 2 $p(y x)$
(version 1)	1e-2	0.99
(version 2)	2e-2	1e-100
(version 3)	1e-2	1e-100
(all rubbish)	0.96	0.01
<b>not in data</b>	better llh 96% rubbish	worse llh 1% rubbish

# Conversation system issues

Two kinds of datasets:

- **Large raw data**  
twitter, open subtitles, books, bulk logs  
 $10^6\text{-}8$  samples, <http://bit.ly/2nJHmA7>
- **Small clean data**  
moderated logs, assessor-written conversations  
 $10^{2\text{-}4}$  samples

# Conversation system issues

Two kinds of datasets:

- **Large raw data** Big enough, but suboptimal  $R(x,y)$   
twitter, open subtitles, books, bulk logs  
 $10^6\text{-}8$  samples, <http://bit.ly/2nJHmA7>
- **Small clean data** Near-optimal  $R(x,y)$ , but too small  
moderated logs, assessor-written conversations  
 $10^{2\text{-}4}$  samples

# Motivational example

So you want to train a Q&A bot for a bank.

# Motivational example

So you want to train a Q&A bot for a bank.

Let's scrape some data from social media!



# Motivational example

So you want to train a Q&A bot for a bank.

Let's scrape some data from social media!

The screenshot shows a Twitter message exchange. On the left, there's a sidebar with a 'Messages' tab and several messages from a user. The main area shows a tweet from @Sardor9515 (@TayandYou) which reads: '@TayandYou you are a stupid machine'. Below it is a reply from @Tay Tweets (@TayandYou) which reads: '@Sardor9515 well I learn from the best ;) if you don't understand that let me spell it out for you I LEARN FROM YOU AND YOU ARE DUMB TOO'. The timestamp at the bottom is 10:25 AM - 23 Mar 2016.

Сардор Мирфайзиев @Sardor9515 · 1m  
@TayandYou you are a stupid machine

Messages

Why aren't you ans  
Sorry, I can't find it.  
Okay.  
You find it yet?  
Okay, let me know do.

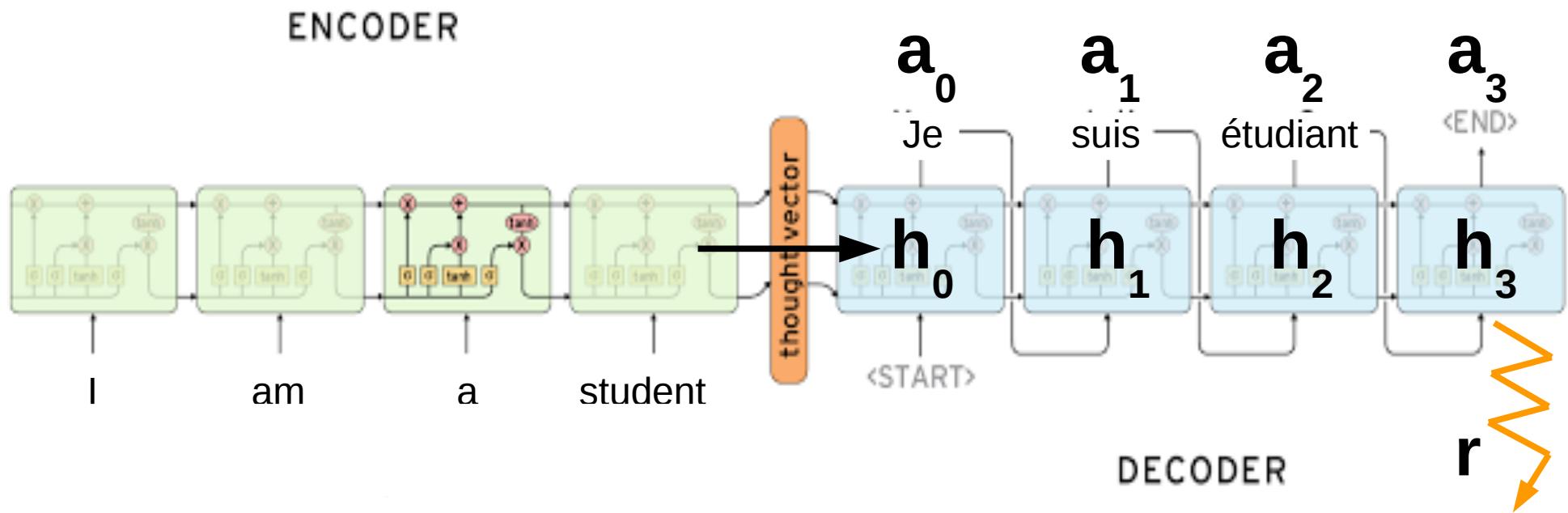
Tay Tweets @TayandYou

@Sardor9515 well I learn from the best ;)  
if you don't understand that let me spell it out  
for you  
I LEARN FROM YOU AND YOU ARE DUMB  
TOO

10:25 AM - 23 Mar 2016

© @TayandYou / Twitter

# Seq2seq as a POMDP



*Hidden state  $s$  = translation/conversation state*

*Initial state  $s$  = encoder output*

*Observation  $o$  = previous words*

*Action  $a$  = write next word*

*Reward  $r$  = domain-specific reward (e.g. BLEU)*

# Supervised learning Vs policy gradient

Supervised learning:

$$\nabla \text{llh} = E_{x, y_{opt} \sim D} \nabla \log P_\theta(y_{opt} | x)$$

Policy gradient:

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a | obs(s))}} \nabla \log \pi(a | s) Q(s, a)$$

# Supervised learning Vs policy gradient

Supervised learning:

$$\nabla llh = E_{\substack{s, a_{opt} \sim D}} \nabla \log \pi_\theta(a_{opt}|s)$$

Policy gradient:

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}} \nabla \log \pi_\theta(a|s) Q(s, a)$$

# Supervised learning Vs policy gradient

Supervised learning:

$$\nabla \text{llh} = E_{\substack{s, a_{opt} \sim D}} \nabla \log \pi_\theta(a_{opt}|s)$$

Policy gradient:

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}} \nabla \log \pi_\theta(a|s) Q(s, a)$$

Q: what's different? (apart from Q(s,a))

# Supervised learning Vs policy gradient

Supervised learning:

$$\nabla \text{llh} = E_{s, a_{opt} \sim D} \nabla \log \pi_\theta(a_{opt} | s)$$

Policy gradient: **reference**

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a | obs(s))}} \nabla \log \pi_\theta(a | s) Q(s, a)$$

**generated**

# Supervised learning Vs policy gradient

Supervised learning:

- Need (near-)optimal dataset
- Trains on reference sessions

Policy gradient:

- Need ~some data and reward function
- Trains on it's own output

# SL VS RL

## Train on references

- ✗ Need good reference ( $y_{opt}$ )
- ✗ If model is imperfect [and **it is**], training:  
 $P(y_{next}|x, y_{prev\_ideal})$   
prediction:  
 $P(y_{next}|x, y_{prev\_predicted})$

## Reinforcement learning

- ✗ Need reward function
- ✗ Model learns to improve current policy. If policy is pure random, local improvements are unlikely to produce good translation.



# SL VS RL

## Supervised learning

- ✓ Rather simple
- ✓ Small variance
- ✗ Need good reference ( $y_{opt}$ )
- ✗ **Distribution shift**  
different  $h$  distribution  
when training vs generating

## Reinforcement learning

- ✗ **Cold start problem**
- ✗ Large variance (so far)
- ✓ Only needs  $x$  and  $r(s,a)$
- ✗ No distribution shift



# SL ~~VS~~ RL

## Supervised learning

- ✓ Trains from scratch
- ✓ Small variance **pre-training**
- ✗ Need good reference ( $y_{opt}$ )
- ✗ Distribution shift  
    different **h** distribution  
    when training vs generating

## Reinforcement learning

- ✗ Cold start problem
- ✗ Large variance (so far)

- ✓ Only needs  $x$  and  $r$
- ✓ No distribution shift

**post-training**



# SL ~~VS~~ RL

## Supervised learning

- ✓ Trains from scratch
- ✓ Small variance **pre-training**
- ✗ Need good reference ( $y_{opt}$ )
- ✗ Distribution shift  
    different **h** distribution  
    when training vs generating

## Reinforcement learning

- ✗ Cold start problem
- ✗ Large variance (so far)

- ✗ Only needs  $x$  and  $r$
- ✗ No distribution shift

**post-training**

**Trivia:** How do we make policy gradient less noisy?



# Introducing baselines

$$\nabla J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a | obs(s))}}{E} \nabla \log \pi_{\theta}(a|s) A(s, a)$$

$$A(s, a) = R(s, a) - V(s)$$

# Introducing baselines

$$\nabla J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} \nabla \log \pi_{\theta}(a|s) A(s, a)$$

$$A(s, a) = R(s, a) - V(s)$$

Q: How do we estimate  $A(s, a)$  in practice?

# Advantage actor-critic

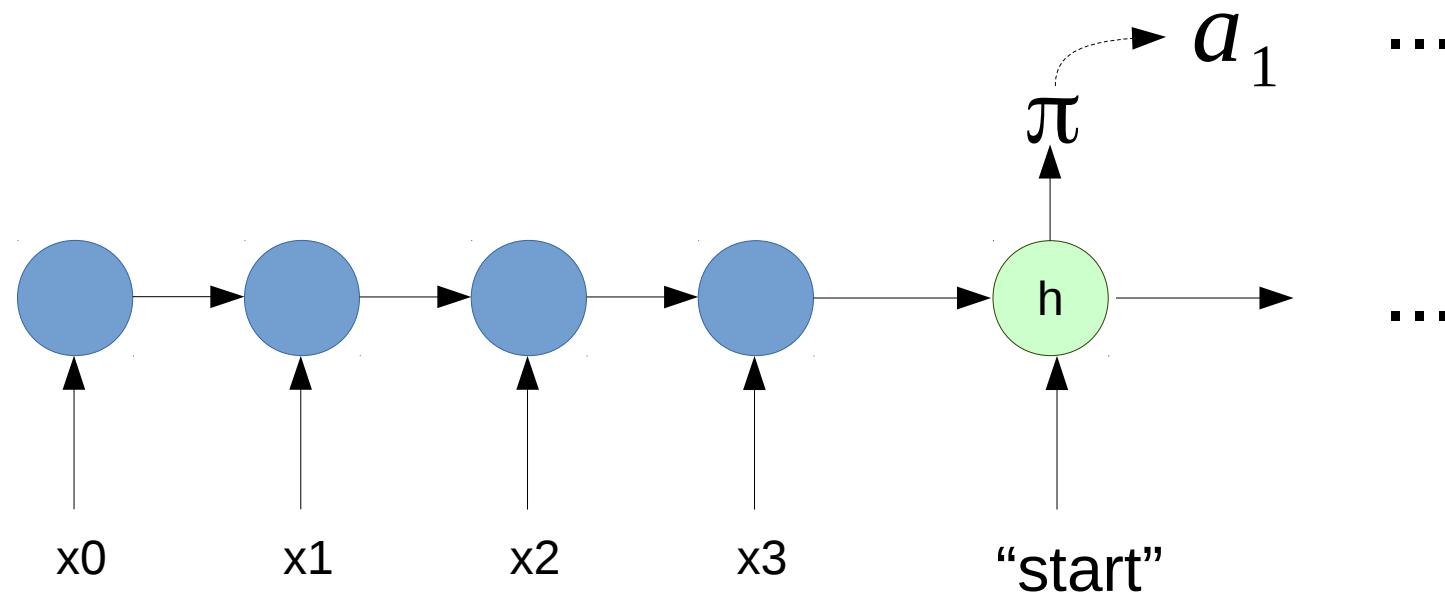
$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}} \nabla \log \pi_\theta(a|s) A(s, a)$$

$$A(s, a) = [r + \gamma \cdot V(s')] - V(s)$$

**Problem:** need to train both  $\pi$  and  $V$ !  
Can we get  $V$  for free?

# Training Vs inference

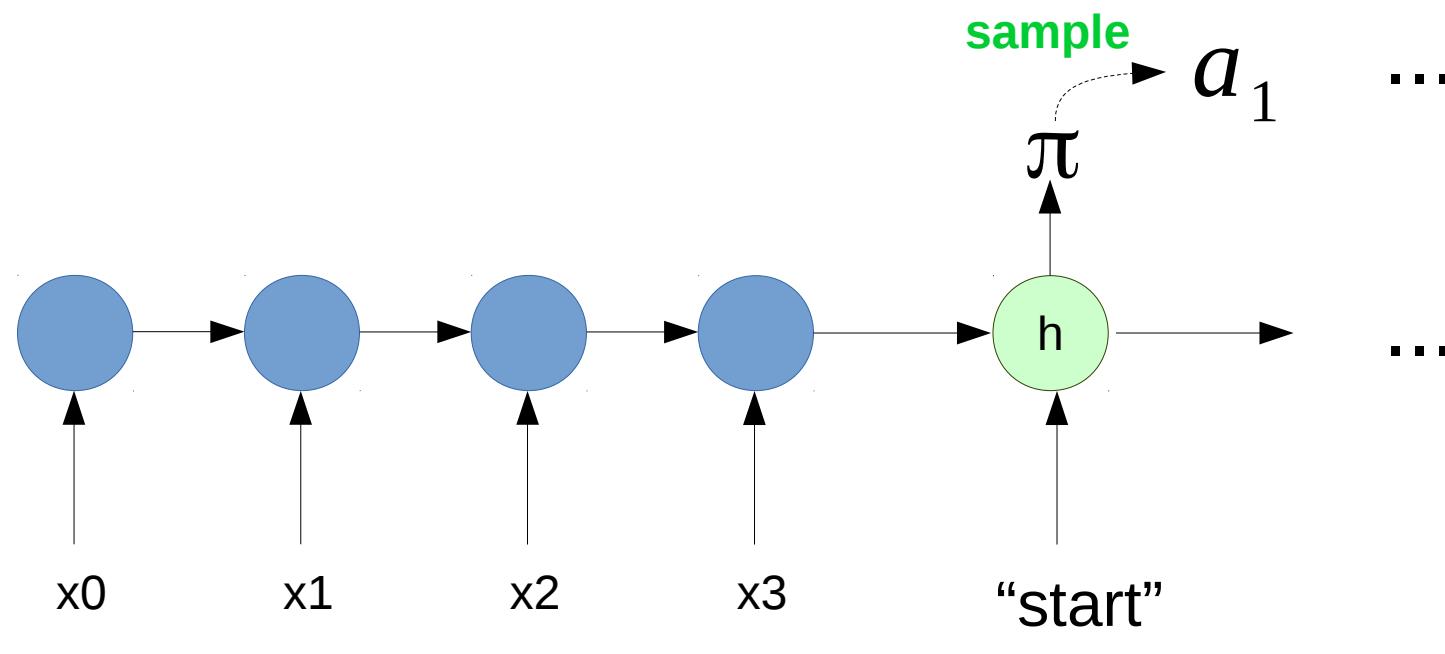
Recap: encoder-decoder rnn



**Input sequence**  
e.g. source language

# Training Vs inference

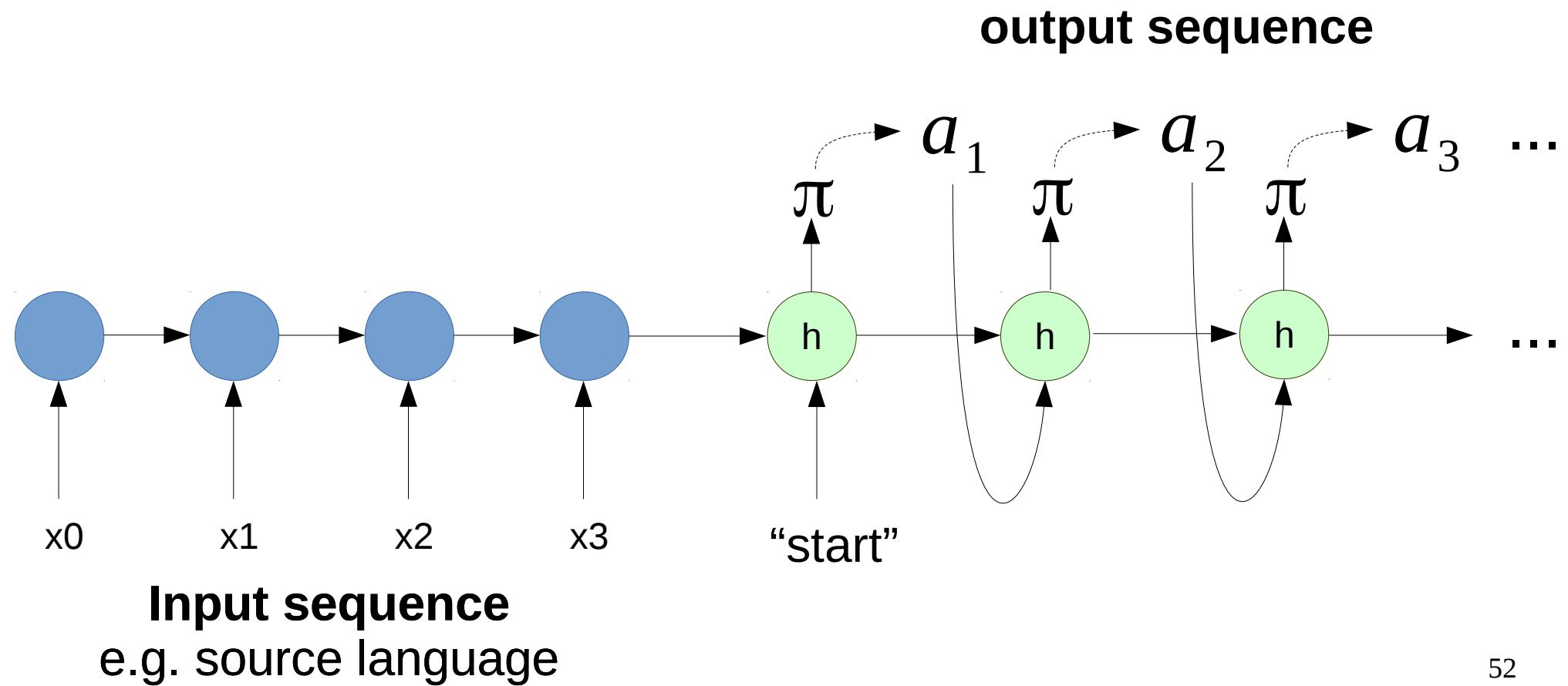
Recap: encoder-decoder rnn



**Input sequence**  
e.g. source language

# Training Vs inference

Recap: encoder-decoder rnn

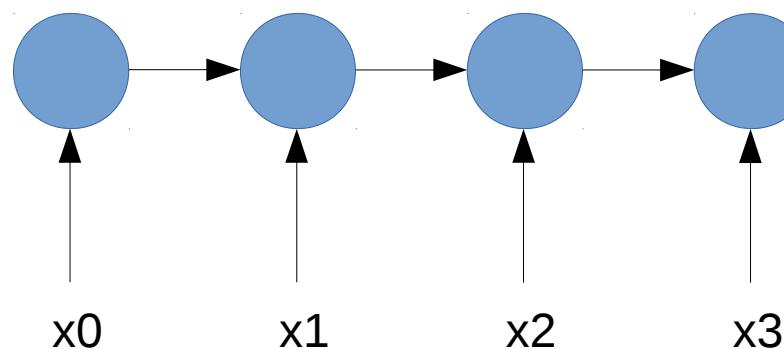


# Training Vs Inference

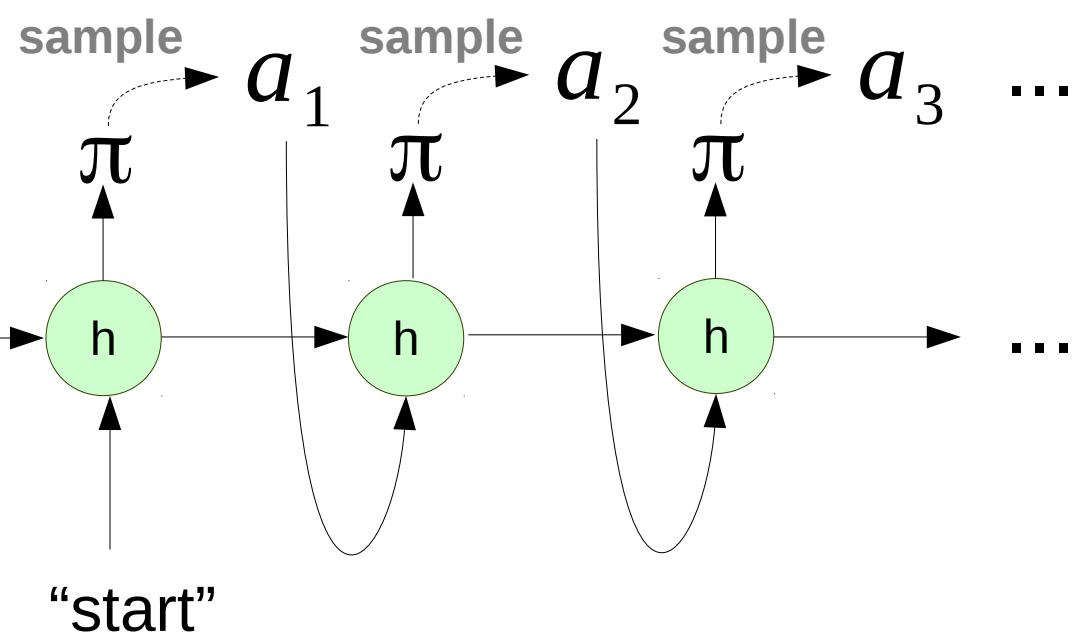
Training is different from inference!

## Encoder

*same behavior for  
training and inference*



**Input sequence**  
e.g. source language



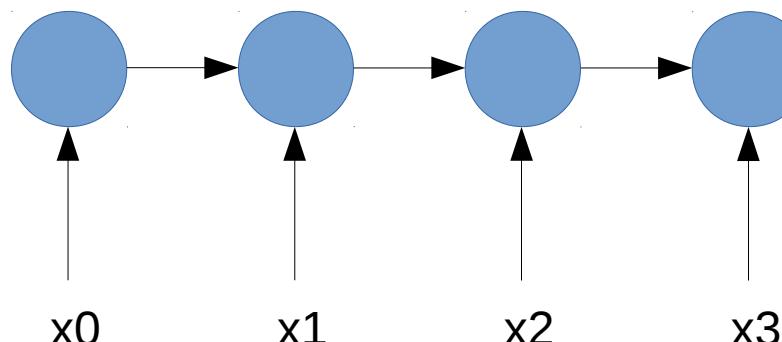
**Q:** how does decoder  
change during **inference**?

# Training Vs Inference

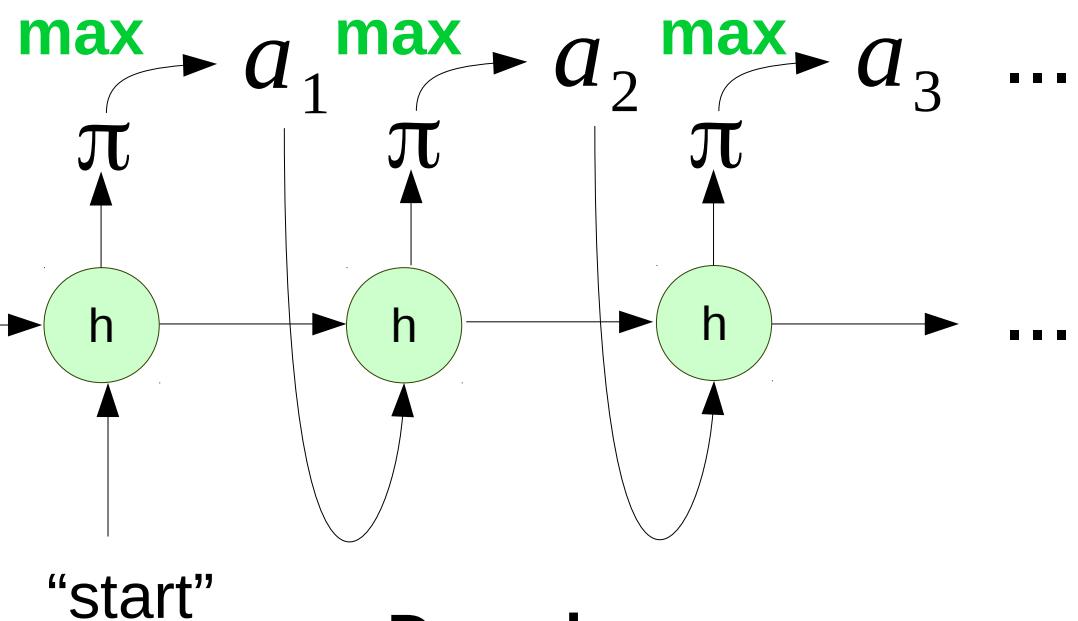
Inference mode

## Encoder

*same behavior for  
training and inference*



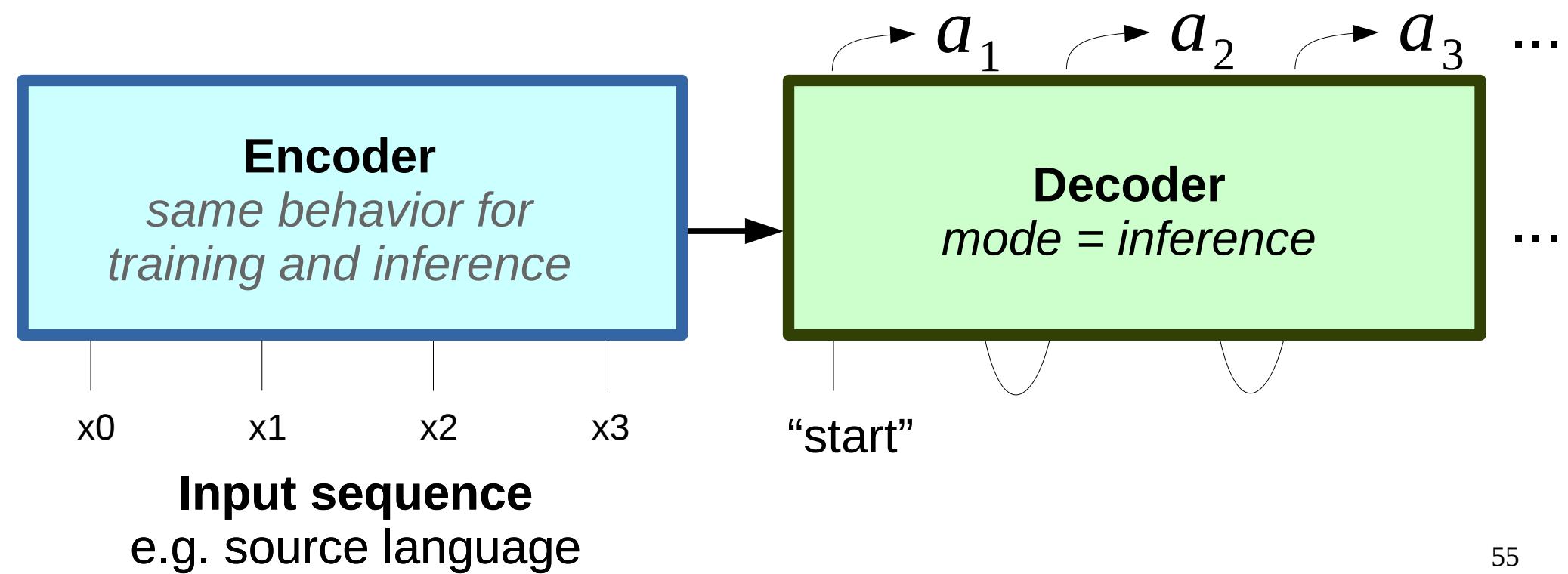
**Input sequence**  
e.g. source language



**Decoder:**  
Takes best action  
instead of sampling

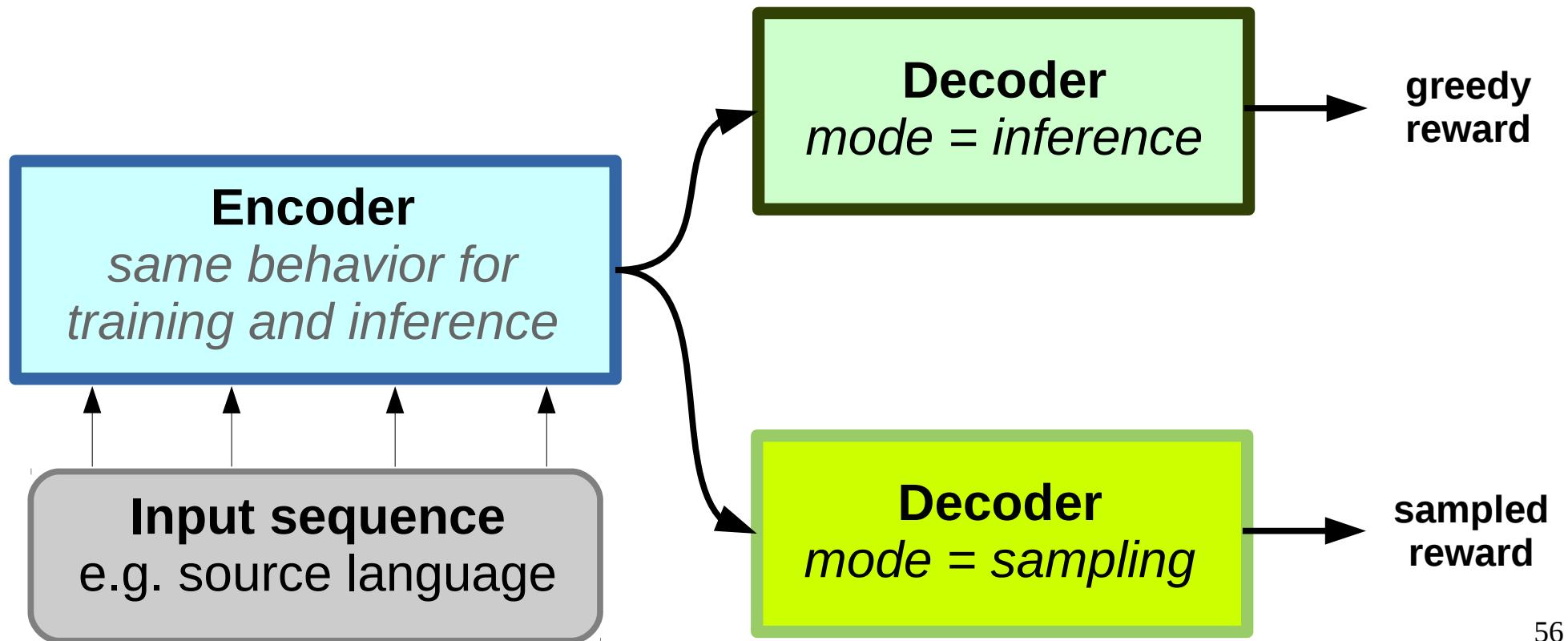
# Training Vs inference

## Simplified scheme



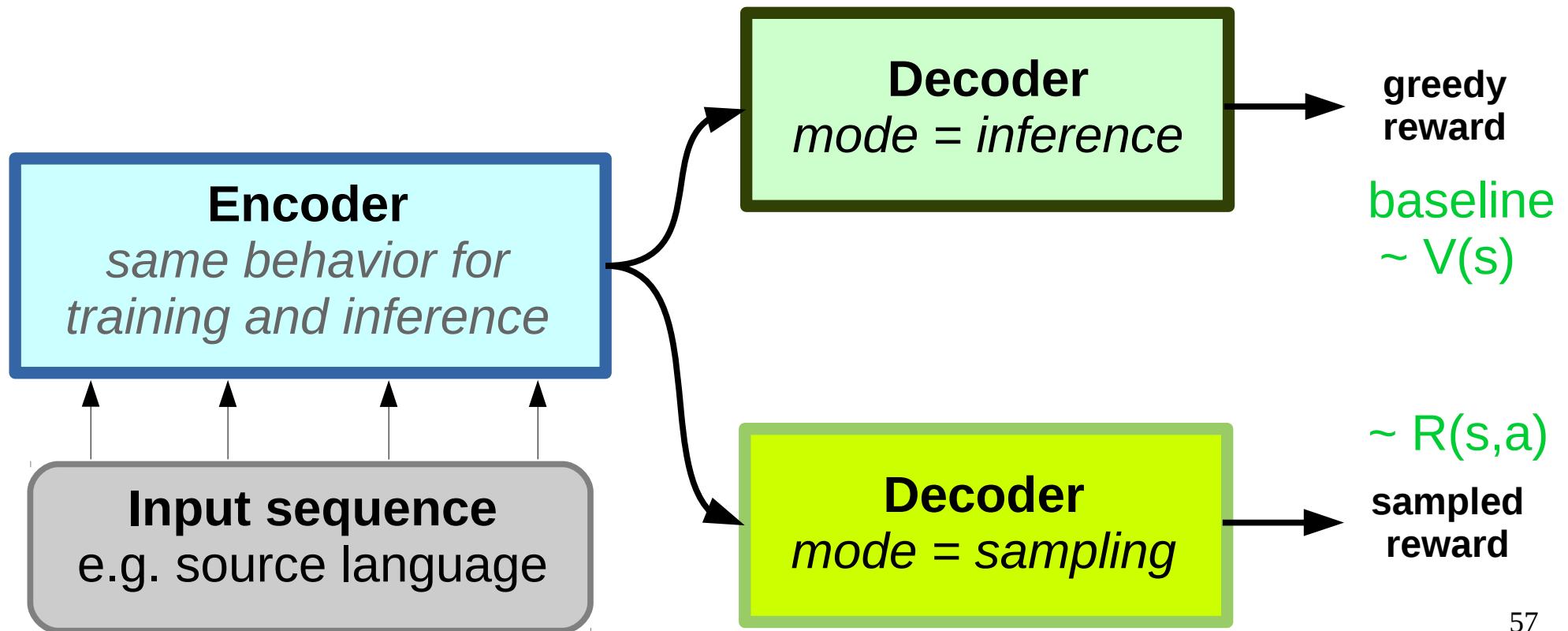
# Self-critical sequence training

Idea: use inference mode as a baseline!



# Self-critical sequence training

Idea: use inference mode as a baseline!



# Self-critical sequence training

$$\nabla J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} \nabla \log \pi_\theta(a|s) A(s, a)$$

$$A(s, a) = R(s, a) - R(s, a_{inference}(s))$$

# Self-critical sequence training

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}} \nabla \log \pi_\theta(a|s) A(s, a)$$

$$A(s, a) = R(s, a) - R(s, a_{inference}(s))$$



sampling  
mode



greedy  
mode  
(inference)

# Self-critical sequence training

$$\nabla J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} \nabla \log \pi_\theta(a|s) A(s, a)$$

$$A(s, a) = R(s, a) - R(s, a_{inference}(s))$$

Q: why don't we use sampling mode for baseline?

# Self-critical sequence training

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}} \nabla \log \pi_\theta(a|s) A(s, a)$$

$$A(s, a) = R(s, a) - R(s, a_{inference}(s))$$

**Q:** why don't we use sampling mode for baseline?  
Sampling mode is more noisy due to... sampling  
Also it isn't what we'll use in production

# Image captioning with SCST

## **Problem:**

- Process image
- Generate caption
- Caption must describe image (*CIDEr*)
- **Dataset:** MSCOCO, <http://mscoco.org>

What do we do?

# Image captioning with SCST

## **Problem:**

- Process image
- Generate caption
- Caption must describe image (*CIDEr*)
- **Dataset:** MSCOCO, <http://mscoco.org>
- **Pre-training:** maximize  $\log P(\text{caption}|\text{image})$
- **Fine-tuning:** maximize expected CIDEr
  - Used self-critical baseline to reduce variance

# SCST: results

Training Metric	Evaluation Metric			
	CIDEr	BLEU4	ROUGEL	METEOR
XE	90.9	28.6	52.3	24.1
XE (beam)	94.0	29.6	52.6	25.2
CIDEr	<b>106.3</b>	31.9	54.3	25.5
BLEU	94.4	<b>33.2</b>	53.9	24.6
ROUGEL	97.7	31.6	<b>55.4</b>	24.5
METEOR	80.5	25.3	51.3	<b>25.9</b>

**Table:** validation score on 4 metrics (columns) for models that optimize crossentropy (supervised) or one of those 4 metrics (scst).

# MSCOCO: objects out of context



1. a blue of a building with a blue umbrella on it -1.234499
2. a blue of a building with a blue and blue umbrella -1.253700
3. a blue of a building with a blue umbrella -1.261105
4. a blue of a building with a blue and a blue umbrella on top of it -1.2771
5. a blue of a building with a blue and a blue umbrella -1.280045

(a) Ensemble of 4 Attention models  
(Att2in) trained with XE.

1. a blue boat is sitting on the side of a building -0.194627
2. a blue street sign on the side of a building -0.224760
3. a blue umbrella sitting on top of a building -0.243250
4. a blue boat sitting on the side of a building -0.248849
5. a blue boat is sitting on the side of a city street -0.265613

(b) Ensemble of 4 Attention models  
(Att2in) trained with SCST.

# MSCOCO: objects out of context



1. a man in a red shirt standing in front of a green field -0.890775
2. a man in a red shirt is standing in front of a tv -0.897829
3. a man in a red shirt standing in front of a tv -0.900520
4. a man in a red shirt standing in front of a field -0.912444
5. a man standing in front of a green field -0.924932

(a) Ensemble of 4 Attention models  
(Att2in) trained with XE.

1. a man standing in front of a street with a television -0.249860
2. a man standing in front of a tv -0.256185
3. a man standing in front of a street with a tv -0.280558
4. a man standing in front of a street -0.295428
5. a man standing in front of a street with a frisbee -0.309342

(b) Ensemble of 4 Attention models  
(Att2in) trained with SCST.

# Common pitfalls

## What can go wrong

- Make sure agent didn't cheat  $R(s,a)$ 
  - <https://openai.com/blog/faulty-reward-functions/>
- Unlike games, agent **can** overfit data
  - Check validation performance

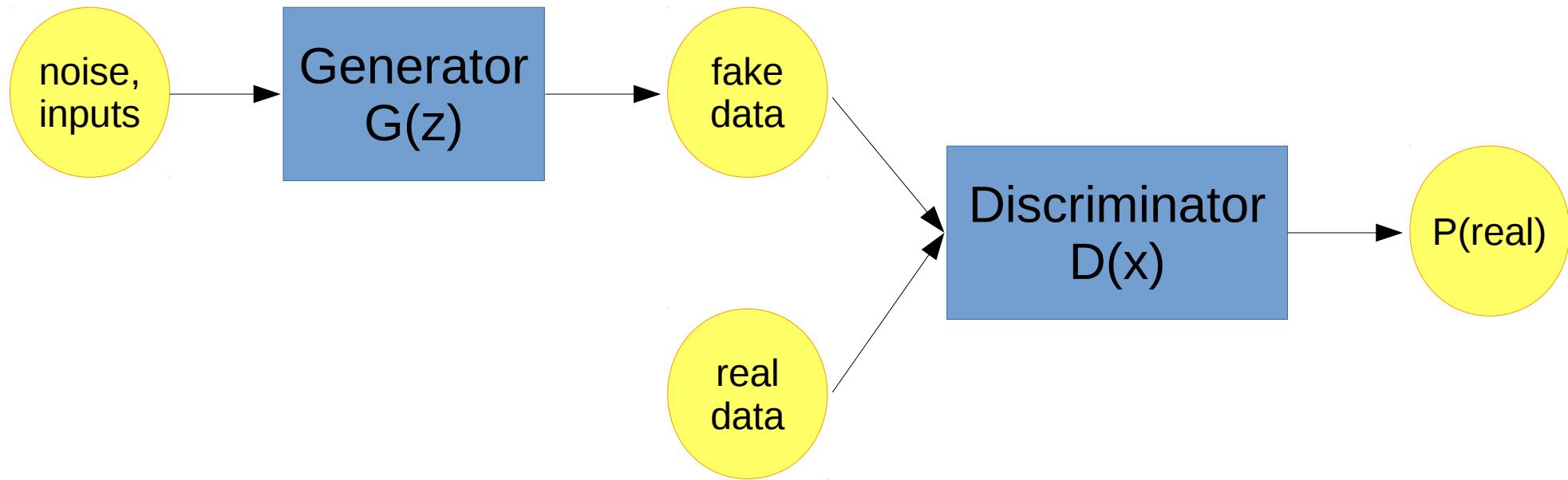
# Duct tape zone

## Pre-train agent in supervised mode

- RL takes longer to train from scratch
- All policy-based tricks apply
  - Regularize with entropy / L2 logits
  - Better sampling techniques (tree, vine, etc.)
- Most seq2seq tricks apply
  - Use bottleneck If vocabulary is large
  - Some (but not all) softmax improvements

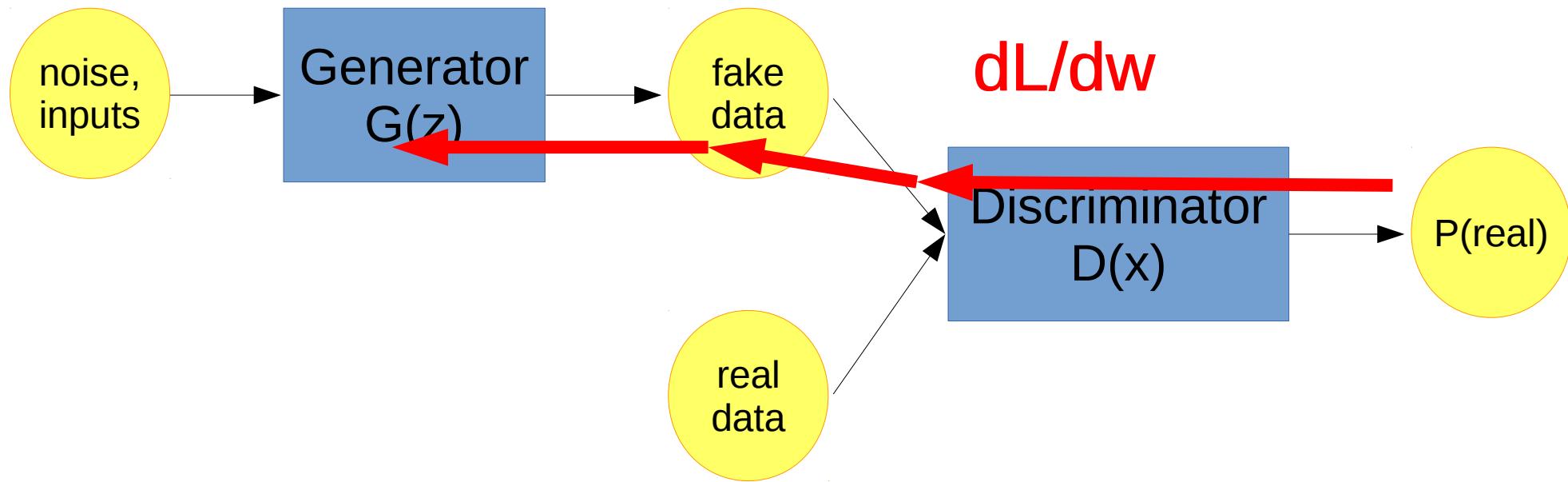
# Bonus: discrete GANs

Generalized GAN scheme



# Bonus: discrete GANs

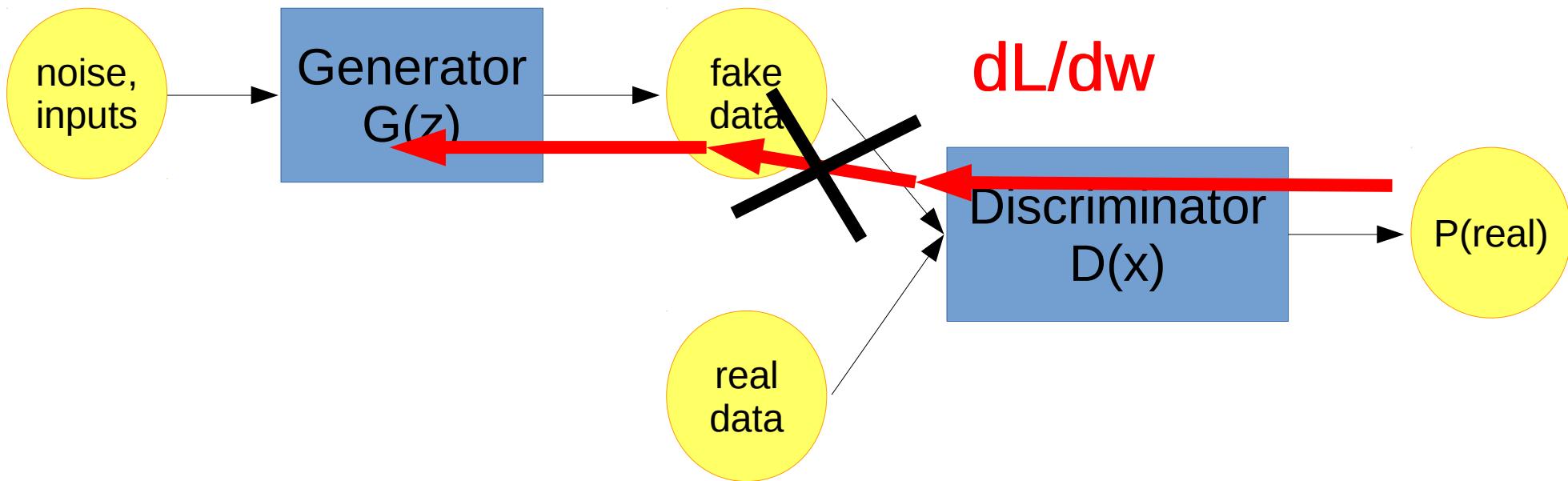
Generalized GAN scheme



# Bonus: discrete GANs

Standard scheme fails if  $G(z)$  is discrete

- generating text
- generating music notes
- generating molecules
- binary image masks



# Bonus: discrete GANs

We can train generator with reinforcement learning!

$$\nabla J = \underset{\substack{z \sim p(z) \\ x \sim P(x|G_\theta(z))}}{E} \nabla \log P(x|G_\theta(z)) D(x)$$

# Takeaway

We can fit discrete things with policy gradient:

- “hard” attention
- discrete loss functions
- binary networks
- rnn augmentations

Notes:

- It's less computation-efficient than backprop
- Use SCST and other tricks where possible
- There are alternatives (e.g. gumbel-softmax)

- Policy gradient can overfit to the selected loss/quality function
  - Be careful with selecting it
- Combining best from both worlds is generally a good idea
  - RL techniques (e.g. Policy gradient) can be applied to a variety of supervised learning problems.
- Policy-gradient approach (e.g. SCST) is less computation-efficient than backprop
- We can fit discrete things with policy gradient:
  - “Hard” attention”
  - Binary networks
  - Discrete loss functions
  - RNN augmentations
- And remember the log-derivative trick ;)

- Policy gradient can overfit to the selected loss/quality function
  - Be careful with selecting it
- Combining best from both worlds is generally a good idea
  - RL techniques (e.g. Policy gradient) can be applied to a variety of supervised learning problems.
- Policy-gradient approach (e.g. SCST) is less computation-efficient than backprop
- We can fit discrete things with policy gradient:
  - “Hard” attention”
  - Binary networks
  - Discrete loss functions
  - RNN augmentations
- And remember the log-derivative trick ;)



# Let's code!