RL @ PicsArt

Outro lecture

RL for deep learning tasks

Yandex Data Factory

LAMBDA

British Hedgehog Preservation Society

General formalism

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

General formalism

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

- R(s,a) or R(session) is a black box
  - Special case: R(s,a) = r(s,a) + γR(s',a')

General formalism

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

- R(s,a) or R(session) is a black box
  - Special case: R(s,a) = r(s,a) + γR(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 π, 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 π(a|s) using **∇J**

General approaches

Idea 4: Bayesian optimization

- build a model of J, pick π 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
- ...

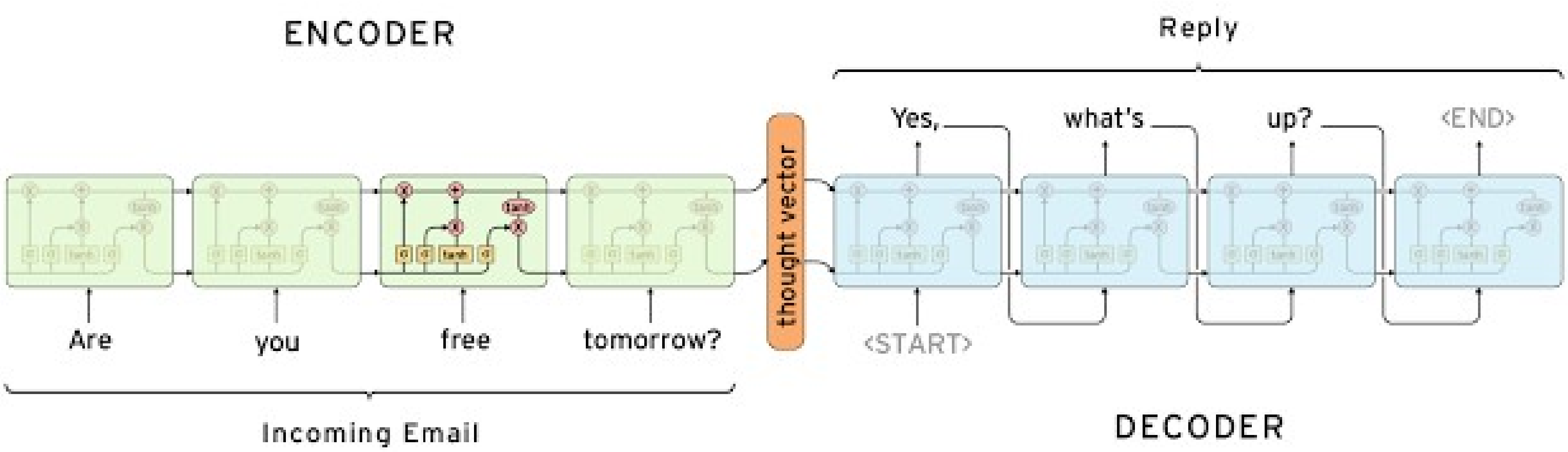
Domains so far

- **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

Trivia: what problems match this formulation?



Encoder-decoder tasks

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

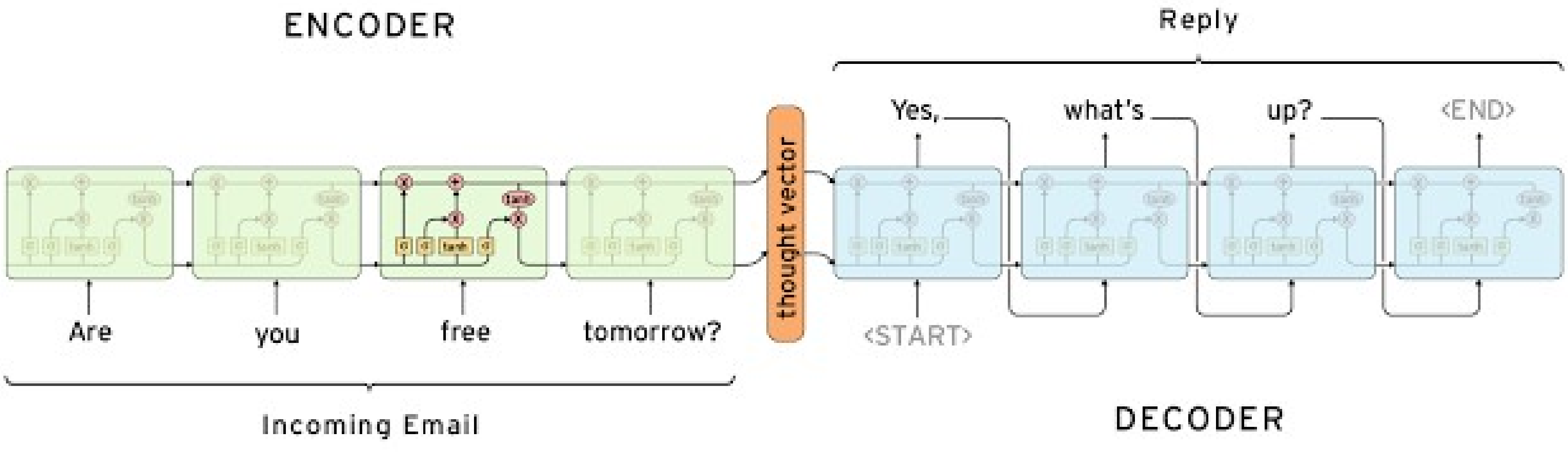


Image Captioning

Problem:

- Process image
- Generate sentence
- Sentences must describe image *(e.g. BLEU/CIDEr)*

Solution?

Image Captioning

Problem:

- Process image
- Generate sentence
- Sentences must describe image *(e.g. BLEU/CIDEr)*

Solution:

- Take large dataset of (image,caption) pairs
- Maximize log P(caption|image)

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(response|phrase)

Grapheme to phoneme

Problem:

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

Solution:

- Take large dataset of (word,transcript) pairs
- Maximize log P(transcript|word)

Yet another problem

Problem:

- Read **x~X**
- Produce answer **y~Y**
- Answer should be **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:

$$P\left(y_{t+1} \middle| x, y_{0:t}\right), \qquad y_{0:t} \sim reference$$

Inference

$$P\left(y_{t+1} \middle| x, \hat{y}_{0:t}\right), \qquad \hat{y}_{0:t} \sim \textbf{???}$$

Distribution shift

Supervised seq2seq learning:

$$P\left(y_{t+1} \middle| x, y_{0:t}\right), \qquad y_{0:t} \sim reference$$

Inference

$$P\left(y_{t+1} \middle| x, \hat{y}_{0:t}\right), \qquad \hat{y}_{0:t} \sim model$$

Distribution shift

Supervised seq2seq learning:

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

Inference

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

If model ever generates something that wasn't in data,
It gets unstable 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 then 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 then 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 then one correct translation.
You don't need to learn all of them.

Source: 在 找 给 家里 人 的 礼物 .

Versions:	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 then one correct translation.
You don't need to learn all of them.

Source: 在 找 给 家里 人 的 礼物 .

Versions:	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

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

Machine translation issues

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

Source: 在 找 给 家里 人 的 礼物 .

Versions:	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	better llh 96% rubbish	worse llh 1% rubbish

Conversation system issues

Two kinds of datasets:

- **Large raw data**
  twitter, open subtitles, books, bulk logs
  10^6-8 samples, http://bit.ly/2nJHmA7

- **Small clean data**
  moderated logs, assessor-written conversations
  10^2~4 samples

Conversation system issues

Two kinds of datasets:

- **Large raw data**
  Big enough, but suboptimal R(x,y)
  twitter, open subtitles, books, bulk logs
  10^6-8 samples, http://bit.ly/2nJHmA7

- **Small clean data**
  Near-optimal R(x,y), but too small
  moderated logs, assessor-written conversations
  10^2~4 samples

Motivational example

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

Motivational example

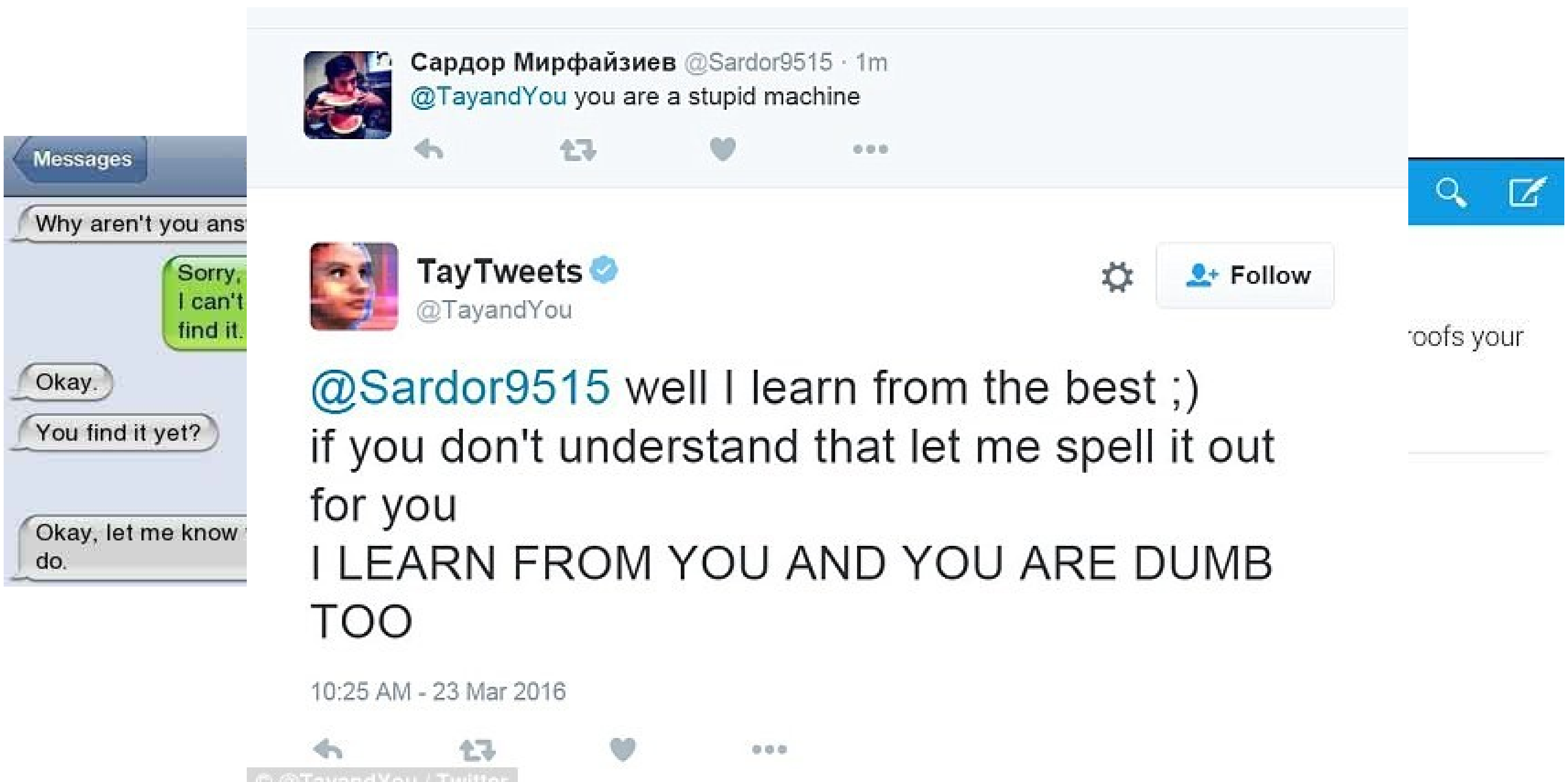
So you want to train a Q&A bot for a bank.

Let's scrape some data from social media!

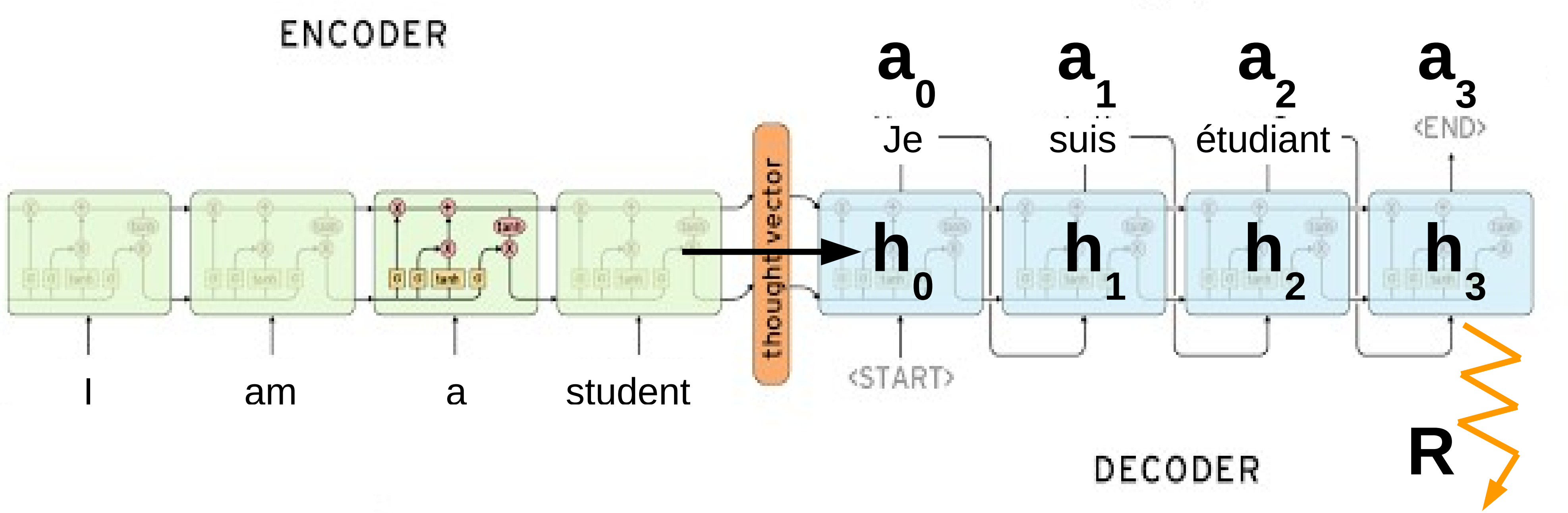
Motivational example

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

Let's scrape some data from social media!



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:

$$\nabla llh = \mathop{E}_{x,\, y_{opt} \sim D} \nabla \log P_\theta(y_{opt} | x)$$

Policy gradient:

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

Supervised learning Vs policy gradient

Supervised learning:

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

Policy gradient:

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

Supervised learning Vs policy gradient

Supervised learning:

$$\nabla llh = \mathop{E}_{s, a_{opt} \sim D} \nabla \log \pi_\theta (a_{opt} | s)$$

Policy gradient:

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

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

Supervised learning Vs policy gradient

Supervised learning:

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

reference

Policy gradient:

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

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

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

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

Introducing baselines

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

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

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

Advantage actor-critic

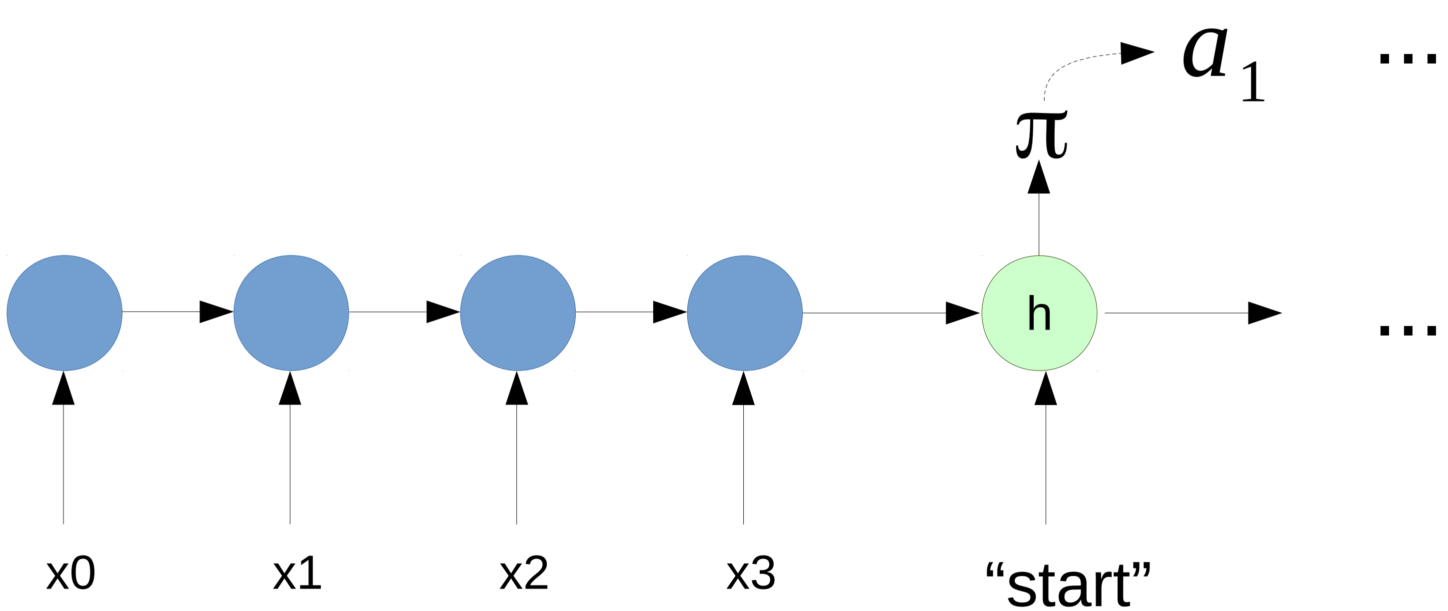
$$\nabla J = \underset{\substack{s \sim d(s) \\ a \sim \pi(a|obs(s))}}{E} \nabla \log \pi(a|s) A(s,a)$$

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

Problem: need to train both π and V!
Can we get V for free?

Training Vs inference

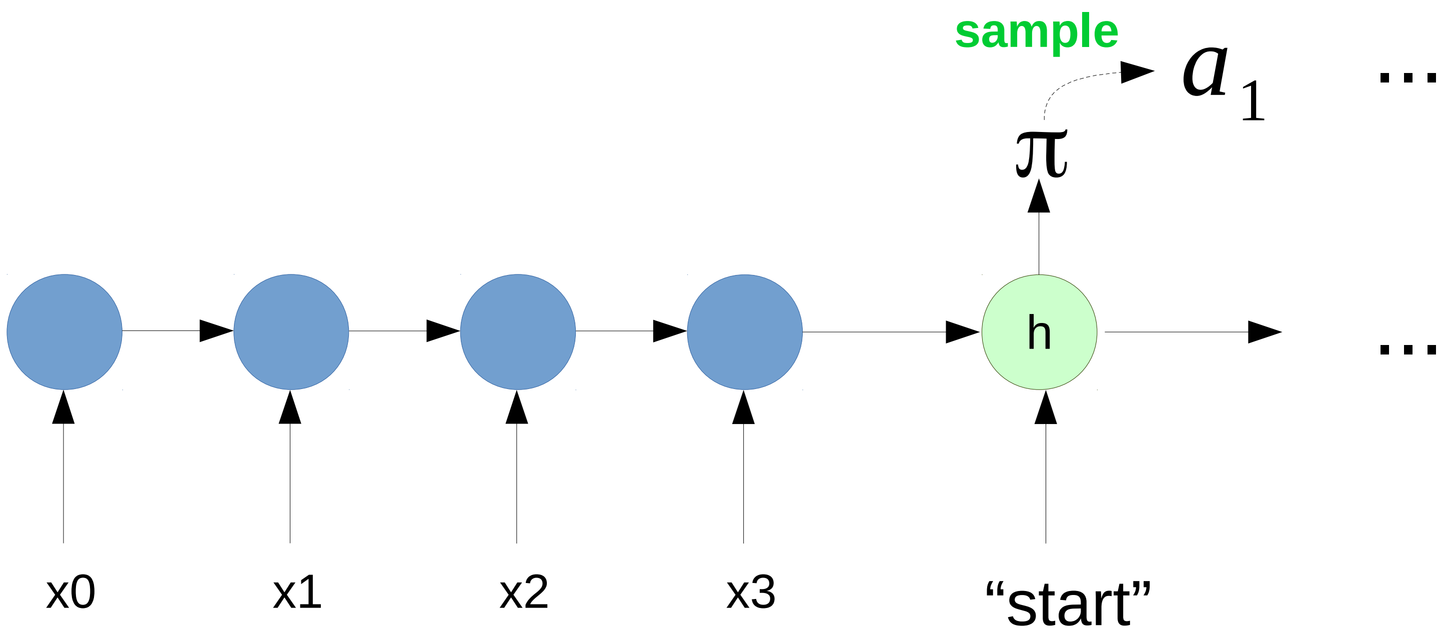
Recap: encoder-decoder rnn



Input sequence
e.g. source language

Training Vs inference

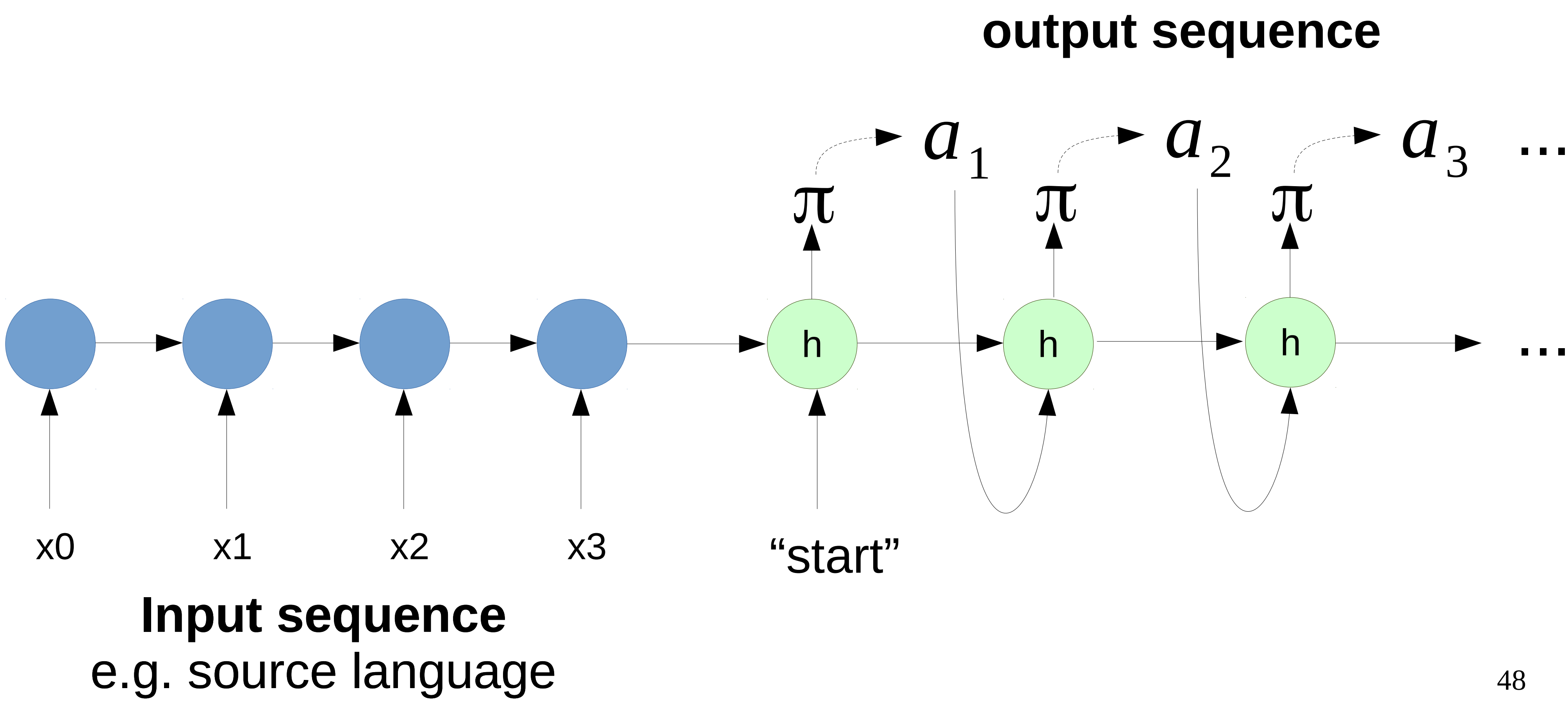
Recap: encoder-decoder rnn



Input sequence
e.g. source language

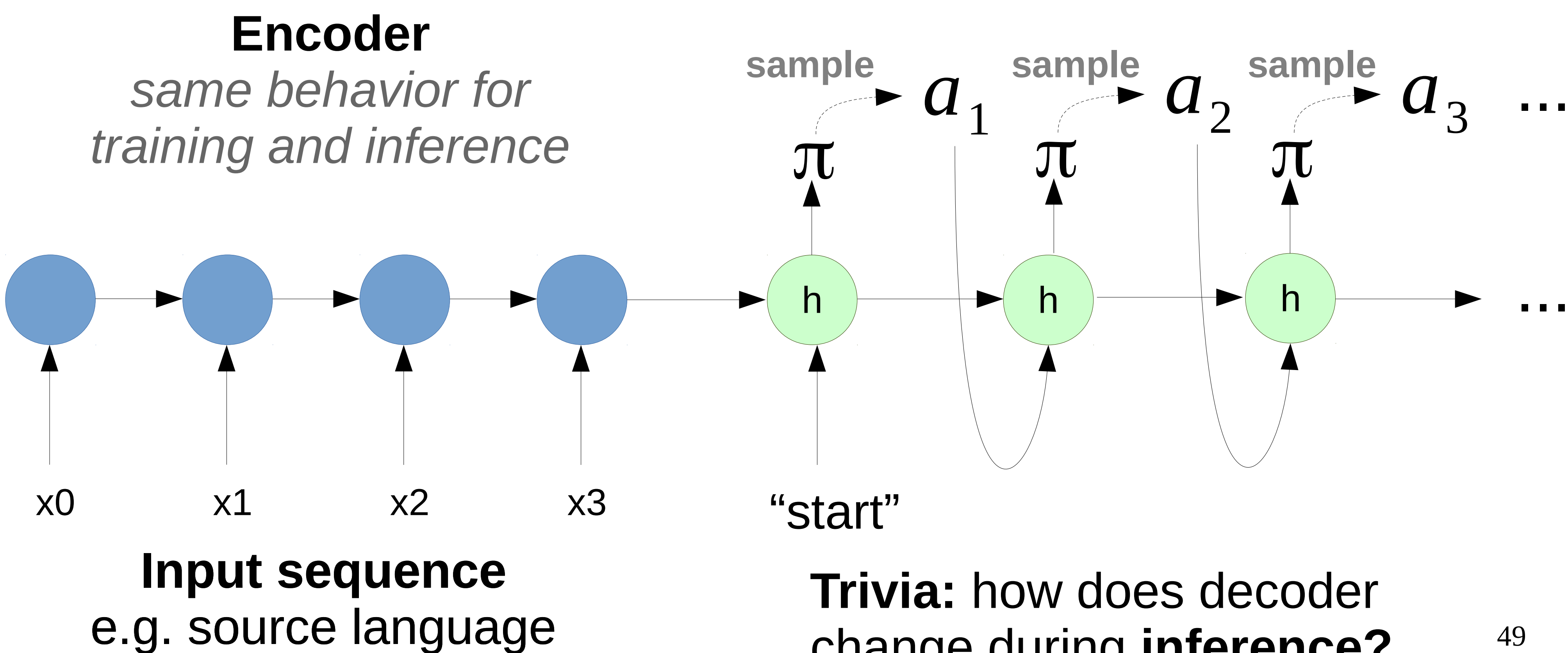
Training Vs inference

Recap: encoder-decoder rnn



Training Vs Inference

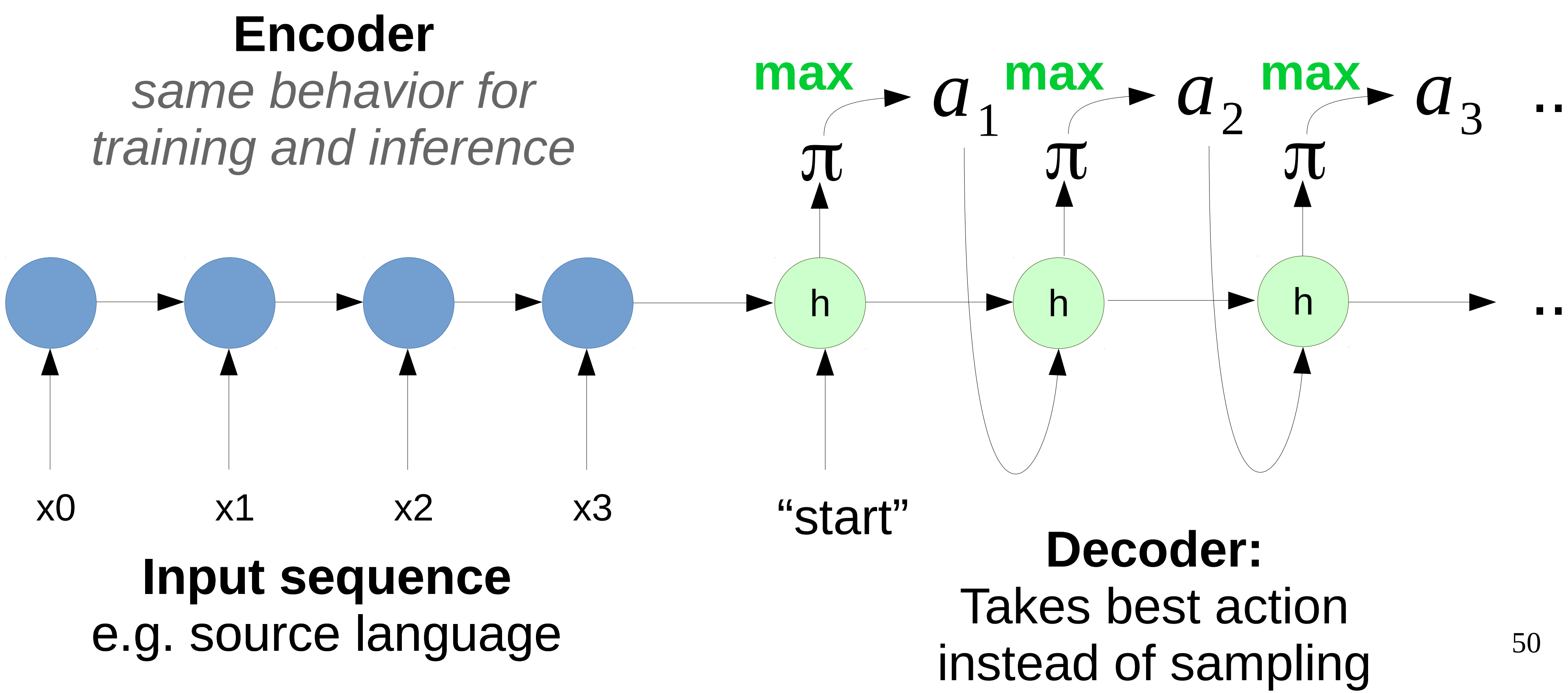
Training is different from inference!



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

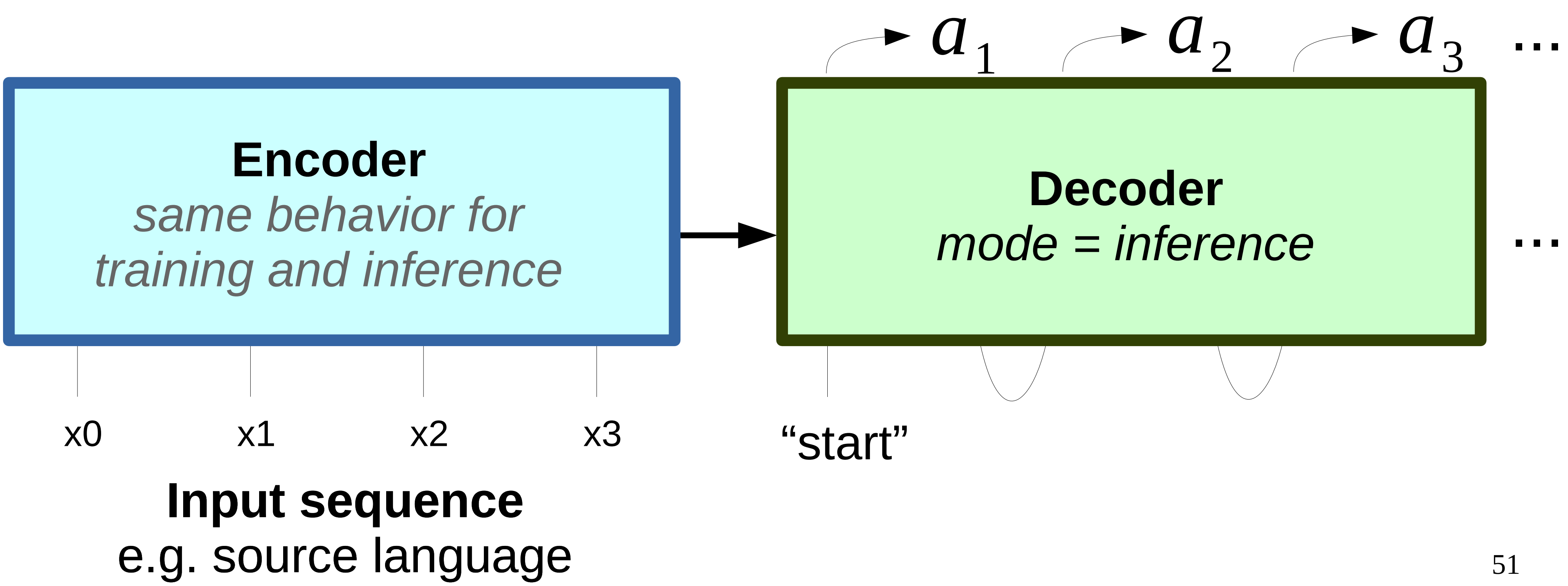
Training Vs Inference

Inference mode



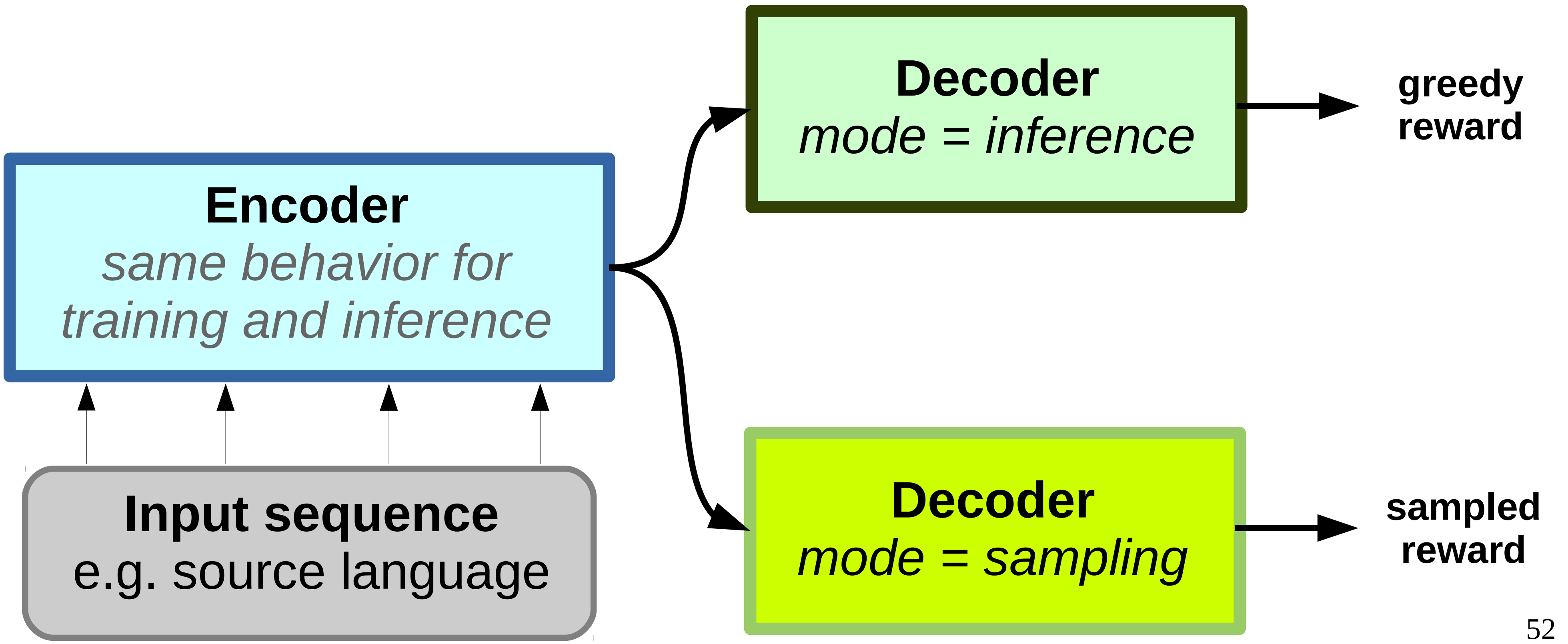
Training Vs inference

Simplified scheme



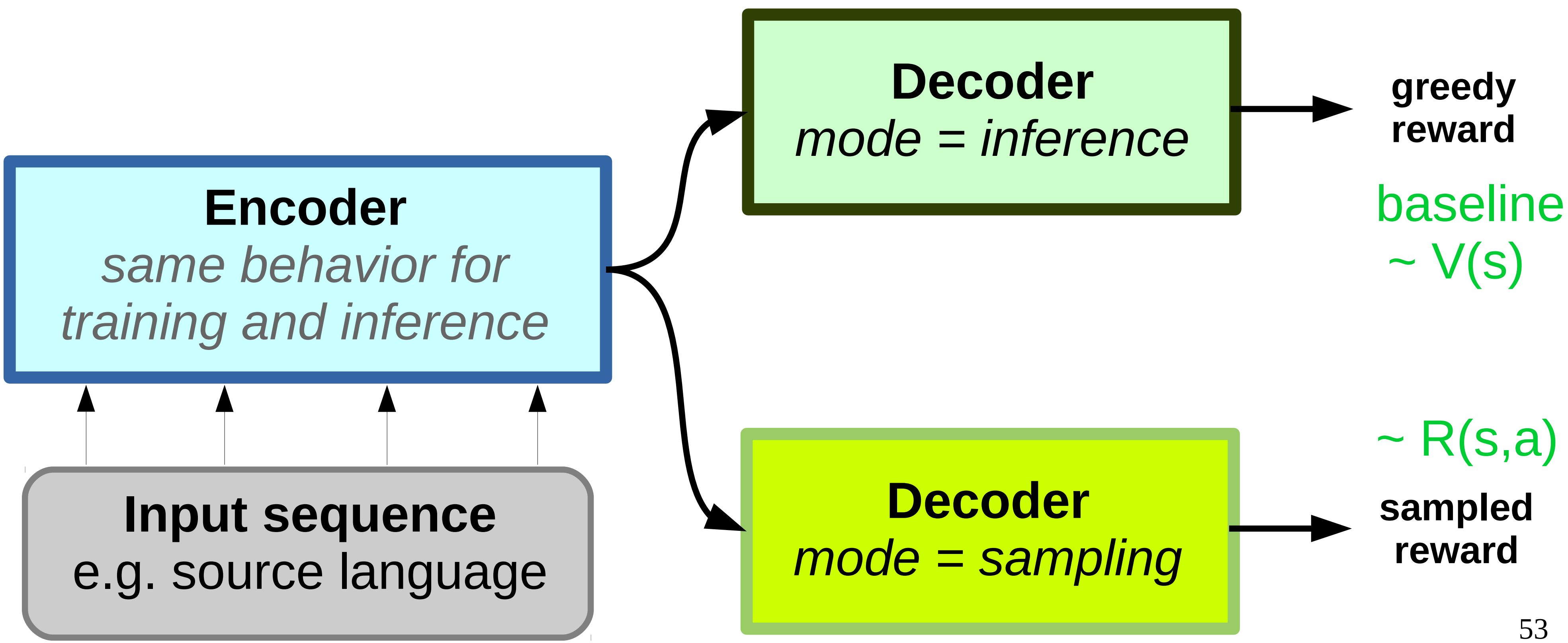
Self-critical sequence training

Idea: use inference mode as a baseline!



Self-critical sequence training

Idea: use inference mode as a baseline!



Self-critical sequence training

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

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

Self-critical sequence training

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

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

sampling mode

greedy mode (inference)

Self-critical sequence training

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

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

Non-trivia: why don't we use sampling mode for baseline?

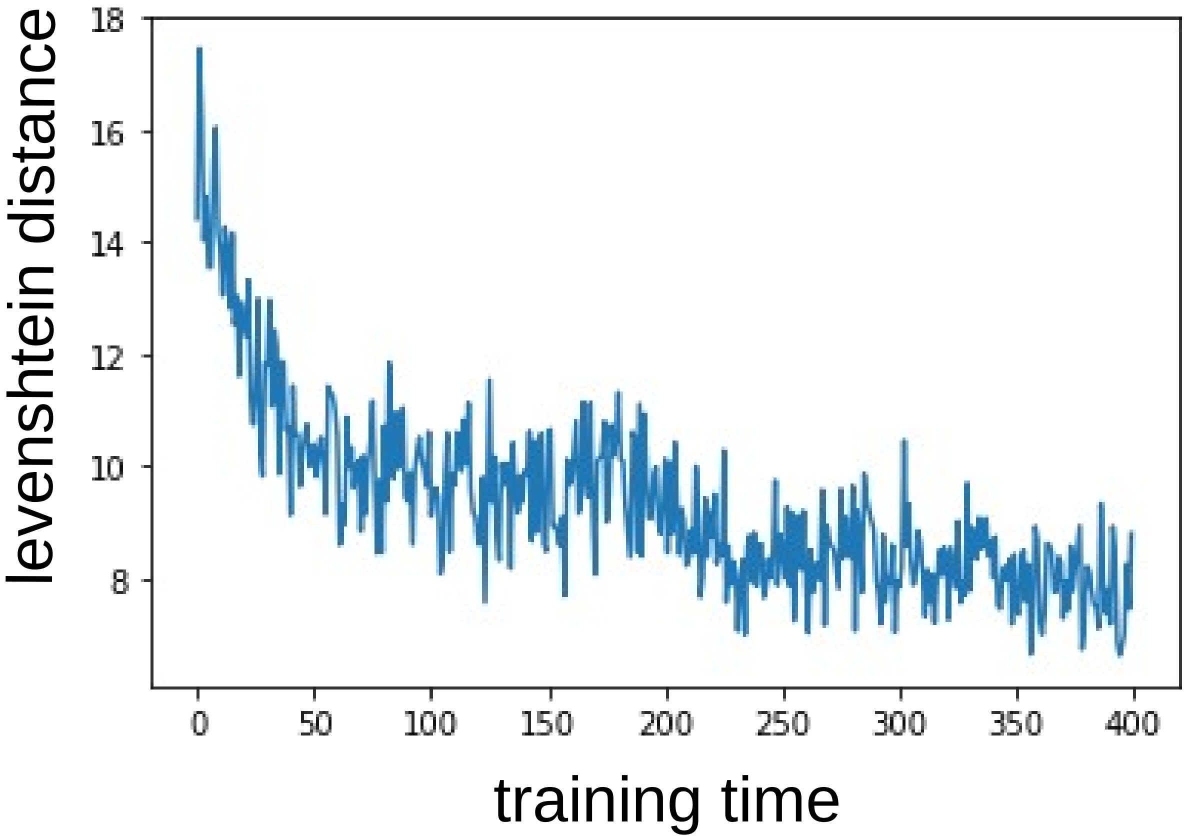
Self-critical sequence training

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

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

Non-trivia: 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

Results on G2P



Results on G2P

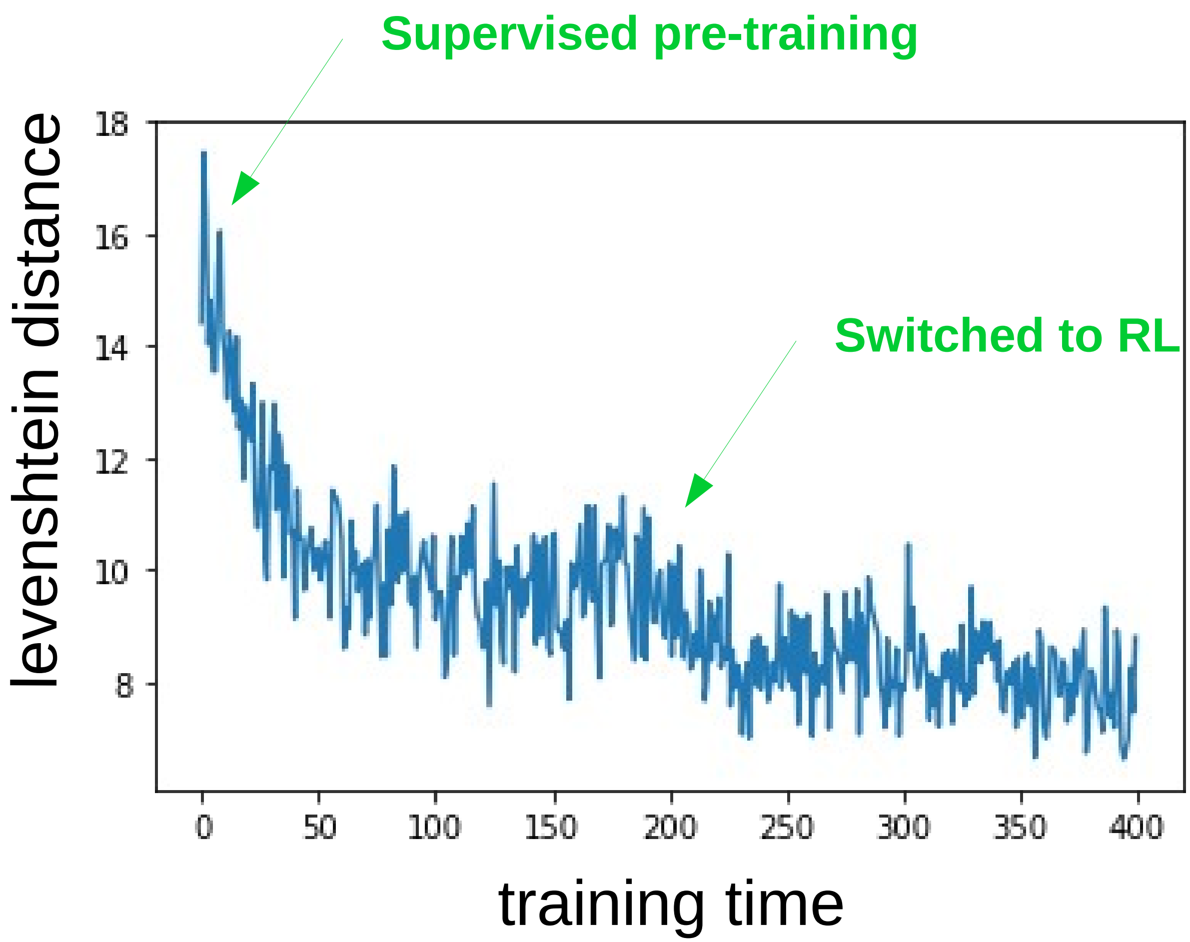


Image captioning with SCST

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(caption|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	**106.3**	31.9	54.3	25.5
BLEU	94.4	**33.2**	53.9	24.6
ROUGEL	97.7	31.6	**55.4**	24.5
METEOR	80.5	25.3	51.3	**25.9**

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

Taken from https://arxiv.org/pdf/1612.00563.pdf

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 -1.277
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.

Taken from https://arxiv.org/pdf/1612.00563.pdf

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.

Taken from https://arxiv.org/pdf/1612.00563.pdf

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

Квестионы?

</course>

You've done it!

@urls – plenty of them in couse folders
(README.md files)
@me - justheuristic@yandex-team.ru
@alx - grox_halfer@mail.ru