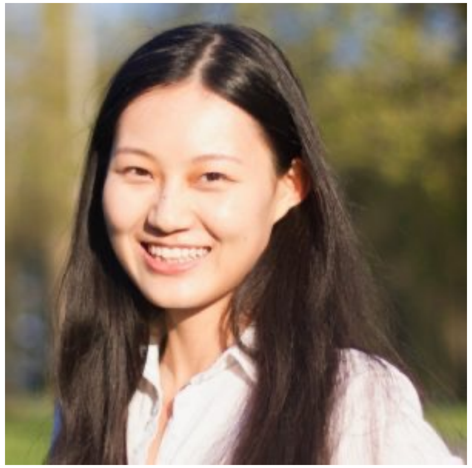
NYU ML² Machine Learning for Language

Amortized Noisy Channel Neural Machine Translation

Richard Yuanzhe Pang

He He

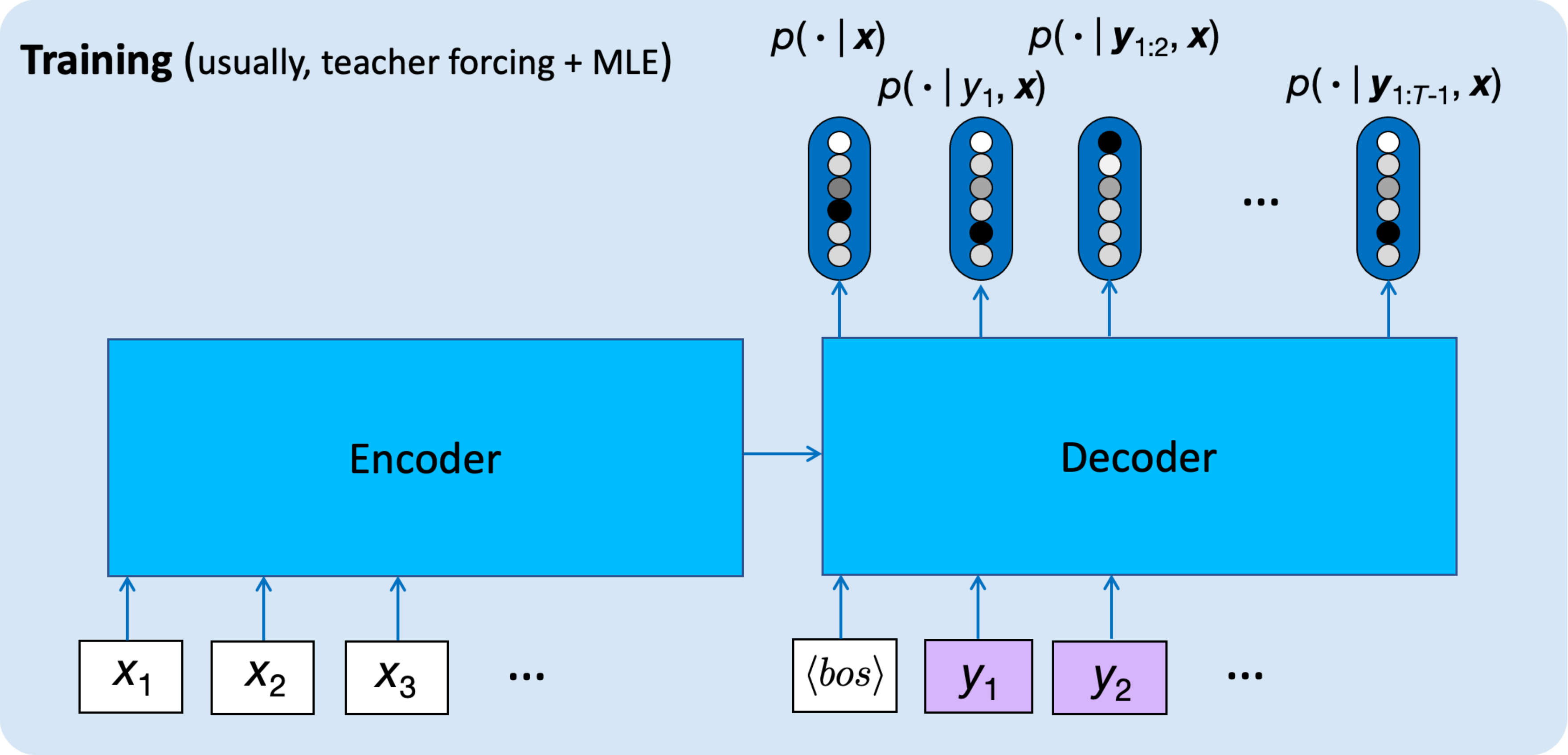
Kyunghyun Cho

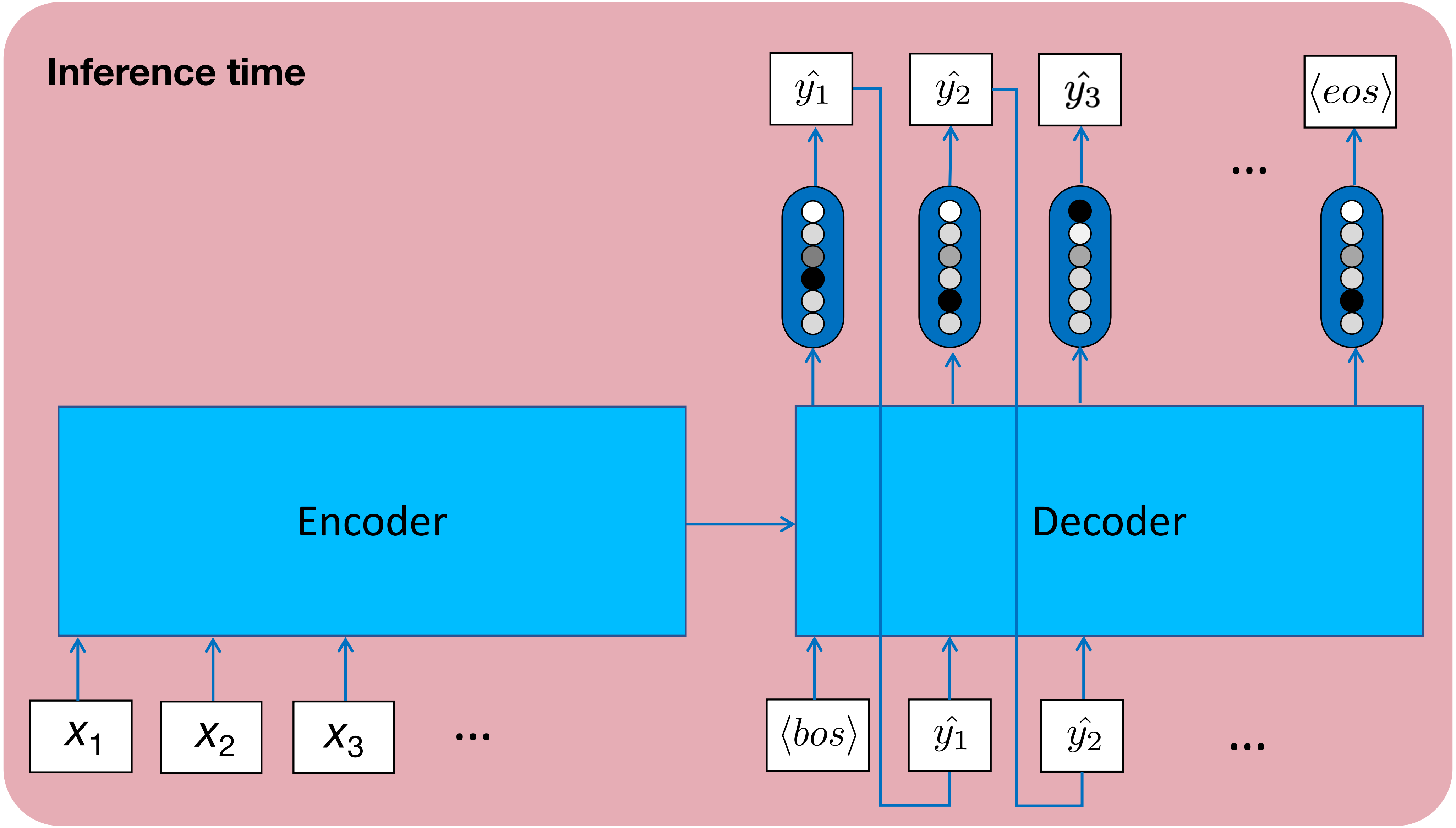


Background 1: conditional text generation

We usually model the distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ where $\boldsymbol{x} = (x_1, x_2, \dots, x_{T_s})$ is the source sequence, and $\boldsymbol{y} = (y_1, y_2, \dots, y_T)$ is the target sequence. Autoregressive factorization:

$$\log p(\boldsymbol{y} \mid \boldsymbol{x}) = \log p(y_1 \mid \boldsymbol{x}) + \log p(y_2 \mid y_1, \boldsymbol{x}) + \log p(y_3 \mid \boldsymbol{y}_{1:2}, \boldsymbol{x}) + \dots + \log p(y_T \mid \boldsymbol{y}_{1:T\text{-}1}, \boldsymbol{x})$$




Inference time
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
...
$\langle eos \rangle$
Encoder
Decoder
x_1
x_2
x_3
...
$\langle bos \rangle$
$\hat{y}_1$
$\hat{y}_2$
...

Background 2: regular NMT vs. noisy channel NMT

Naïve decoding based on the forward translator

Training: train p_f using (X, Y)

Inference: greedy decoding or beam search with small beam size (e.g., b=5)

One way of **noisy channel** decoding: beam search and rerank (BSR)

Training: train p_f and p_r using (X, Y)

Inference: For each source sentence, (1) do beam search using p_f with beam size 50—100; (2) rerank using the following objective and pick the top-ranked translation

$$\log p_f(\boldsymbol{y} \mid \boldsymbol{x}) + \gamma \log p_r(\boldsymbol{x} \mid \boldsymbol{y}) + \gamma' \log p_{lm}(\mathbf{y})$$

Used in many top/winning models in WMT competitions!

Goal

The generated sentences (from a new network) would maximize

$$R(\boldsymbol{y}) = \log p_f(\boldsymbol{y} \mid \boldsymbol{x}) + \gamma \log p_r(\boldsymbol{x} \mid \boldsymbol{y})$$

while using **the same inference time as greedily decoding from $\boldsymbol{p_f}$.**

How to examine if amortization is successful?

Inference speed

- Successful if the inference is faster. Guaranteed.

Translation reward

- Successful if the forward rewards of the generated sentences are comparable to the forward rewards by BSR, and the reverse rewards are comparable to the reverse rewards by BSR.

Translation quality (approximated by BLEURT)

- Successful if the BLEURT of our translations are similar to the BLEURT by BSR.

Approach 1: Knowledge distillation

Training

- Step 1: train p_f using (X, Y)
- Step 2: generate pseudo-corpus Y_{pseudo} by BSR
- Step 3: train p_{KD} using (X, Y_{pseudo})

Inference

- Greedily decode from p_{KD}

Effectively minimizing the KL-div between the distribution induced by the pseudo-corpus obtained through BSR and our model distribution

Approach 2: 1-step-deviation imitation learning

Want: $A_\phi(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ *A* has the same architecture as p_f

$$\min_\phi \sum_{\boldsymbol{x}} \left[\sum_{t=1}^{T} E_t^f(\boldsymbol{x}, \boldsymbol{y}; \phi) + \gamma \sum_{t'=1}^{|\boldsymbol{x}|} E_{t'}^r(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) \right]$$

Approach 2: 1-step-deviation imitation learning

Want: $A_\phi(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$

$$\min_\phi \sum_{\boldsymbol{x}} \left[\sum_{t=1}^{T} E_t^f(\boldsymbol{x}, \boldsymbol{y}; \phi) + \gamma \sum_{t'=1}^{|\boldsymbol{x}|} E_{t'}^r(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) \right]$$

$$E_t^f(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) = -A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})^\top \log p_f(\cdot \mid A_\phi(\cdot|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<1}), \ldots, A_\phi(\cdot|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}), \boldsymbol{x})$$

$$\hat{y}_t = \arg\max_{v \in \mathcal{V}} A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})$$

or the sequences gen by BSR

Approach 2: 1-step-deviation imitation learning

Want: $A_\phi(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$

$$\min_\phi \sum_{\boldsymbol{x}} \left[\sum_{t=1}^{T} E_t^f(\boldsymbol{x}, \boldsymbol{y}; \phi) + \gamma \sum_{t'=1}^{|\boldsymbol{x}|} E_{t'}^r(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) \right]$$

$$E_t^f(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) = -A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<t})^\top \log p_f(\cdot \mid A_\phi(\cdot|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<1}), \ldots, A_\phi(\cdot|\boldsymbol{x}, \hat{\boldsymbol{y}}_{<t}), \boldsymbol{x})$$

$$E_t^r(\boldsymbol{x}, \hat{\boldsymbol{y}}; \phi) = -\mathrm{onehot}(\boldsymbol{x}_t)^\top \log p_r(\cdot \mid \boldsymbol{x}_{<t}, A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<1}), \ldots, A_\phi(\cdot \mid \boldsymbol{x}, \hat{\boldsymbol{y}}_{<T}))$$

Background 3: RL in text generation

Eval $$\mathbb{E}_{\boldsymbol{y} \sim p_\theta} \sum_{t=1}^{T} r(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x})$$

Background 3: RL in text generation

Eval $$\mathbb{E}_{\boldsymbol{y}\sim p_\theta} \sum_{t=1}^{T} r(y_t \mid \underline{\boldsymbol{y}_{1:t-1}, \boldsymbol{x}})$$

action

policy reward state

RL $$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta} \sum_{t=1}^{T} r(a_t, s_t)$$

Conditional text generation can be considered as a sequential decision-making process.

- At each time step t, the policy π_θ takes an action a_t in V, transits to the next state s_{t+1}, and receives a reward r_t.

Approach 3: Q learning

Want: Q ("future return" – higher is better);

Define: $s_t = (\boldsymbol{y}_{<t}, \boldsymbol{x})$, $a_t = y_t$,

$r_t = \log p_f\,(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$, if $t < T$

$= \log p_f\,(y_T \mid \boldsymbol{y}_{<T}, \boldsymbol{x}) + \gamma \log p_r\,(\boldsymbol{x} \mid \boldsymbol{y})$, if $t = T$

Given p_f, p_r, and a parallel translation dataset D. Initialize Q_ϕ and Q'_ϕ by p_f.

while *not converged* **do**

Collect training trajectories, and sample a minibatch B

Compute target R_t:

if $t < T$, then $R_t = r_t + \max\_\{a_{t+1}\}\ Q'_\phi(s_{t+1}, a_{t+1})$

if t = T, then $R_t = r_T$

Update ϕ (using gradient descent) by the objective

$\mathrm{argmin}_\phi\ [\, Q_\phi(s_t, a_t) - R_t\,]^2$

Update Q'_ϕ: $Q'_\phi \leftarrow Q_\phi$ every K steps

Want: Q (“future return” – higher is better);

Define: $s_t = (\boldsymbol{y}_{<t}, \boldsymbol{x})$, $a_t = y_t$,

$r_t = \log p_f\,(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$, if $t < T$

$= \log p_f\,(y_T \mid \boldsymbol{y}_{<T}, \boldsymbol{x}) + \gamma \log p_r\,(\boldsymbol{x} \mid \boldsymbol{y})$, if $t = T$

How to examine if amortization is successful?

Inference speed

- Successful if the inference is faster. Guaranteed.

Translation reward

- Successful if the forward rewards of the generated sentences are comparable to the forward rewards by BSR, and the reverse rewards are comparable to the reverse rewards by BSR.

Translation quality (approximated by BLEU)

- Successful if the BLEU of our translations are similar to the BLEU by BSR.

(1) Inference speed

	beam size (*b*)
p_f	**1**
p_f	5
BSR	50—100
Knowledge distillation	**1**
Imitation learning	**1**
Q learning	**1**

Given that the architecture for each experiment is the same, inference speed of approaches 1—3 is 50—100x of BSR.

(2) Translation reward (forward / reverse)

	beam size (b)	IWSLT14 De-En	WMT16 Ro-En	WMT14 De-En
p_f	**1**	-9.1 / -35.4	-9.5 / -41.0	-11.0 / -31.5
p_f	5	-8.6 / -34.2	-9.0 / -40.2	-10.4 / -29.9
BSR	50—100	-9.4 / -25.7	-10.0 / -29.7	-10.7 / -23.6
Knowledge distillation	**1**	-13.8 / -28.0	-17.2 / -35.4	-14.8 / -24.0
Imitation learning	**1**	-13.3 / -27.9	-17.2 / -34.3	-14.6 / -23.6
Q learning	**1**	-13.7 / -29.9	-11.6 / -39.1	-14.4 / -24.9

Compared to p_f, our approaches have lower forward reward, but higher reverse reward

(2) Translation reward (forward / reverse)

	beam size (b)	IWSLT14 De-En	WMT16 Ro-En	WMT14 De-En
p_f	1	-9.1 / -35.4	-9.5 / -41.0	-11.0 / -31.5
p_f	5	-8.6 / -34.2	-9.0 / -40.2	-10.4 / -29.9
BSR	50—100	-9.4 / -25.7	-10.0 / -29.7	-10.7 / -23.6
Knowledge distillation	**1**	-13.8 / -28.0	-17.2 / -35.4	-14.8 / -24.0
Imitation learning	**1**	-13.3 / -27.9	-17.2 / -34.3	-14.6 / -23.6
Q learning	**1**	-13.7 / -29.9	-11.6 / -39.1	-14.4 / -24.9

Compared to p_f, our approaches have lower forward reward, but higher reverse reward

Compared to BSR, our approaches have lower forward & reverse rewards

(3) BLEURT

	beam size (b)	IWSLT14 De-En	WMT16 Ro-En	WMT14 De-En
p_f	1	62.40 (0.04)	61.14 (0.10)	64.83 (0.10)
p_f	5	63.21 (0.07)	61.42 (0.15)	65.79 (0.08)
BSR	50—100	64.15 (0.05)	62.67 (0.13)	66.32 (0.12)
Knowledge distillation	1	63.88 (0.04)	61.78 (0.10)	66.00 (0.07)
Imitation learning	1	63.94 (0.13)	62.35 (0.16)	66.14 (0.08)
Q learning	1	63.25 (0.07)	61.70 (0.18)	65.92 (0.14)

Using BLEURT-20-D12, imitation learning scores > p_f scores

(3) BLEURT

	beam size (b)	IWSLT14 De-En	WMT16 Ro-En	WMT14 De-En
p_f	1	62.40 (0.04)	61.14 (0.10)	64.83 (0.10)
p_f	5	63.21 (0.07)	61.42 (0.15)	65.79 (0.08)
BSR	50—100	64.15 (0.05)	62.67 (0.13)	66.32 (0.12)
Knowledge distillation	**1**	63.88 (0.04)	61.78 (0.10)	66.00 (0.07)
Imitation learning	**1**	63.94 (0.13)	62.35 (0.16)	66.14 (0.08)
Q learning	**1**	63.25 (0.07)	61.70 (0.18)	65.92 (0.14)

Using BLEURT-20-D12, imitation learning scores not significantly lower than BSR scores

Discussion

- "BSR → high BLEU" doesn't imply "higher reward → higher BLEU/BLEURT"
- Comparing three approaches, knowledge distillation and imitation learning generations are similar, but they are different from Q learning generations.
- The Q network in Q learning is trained from scratch!
  - The BLEU/BLEURT scores are competitive to at least "p_f (beam 5)" but lower than BSR
  - Generates repetitive sequences when the source is long