Amortized Noisy Channel Neural Machine Translation

NYU ML²

Richard Yuanzhe Pang[1] He He[1] Kyunghyun Cho[1,2,3]

[1] New York University [2] Genentech [3] CIFAR Fellow

1 Background and Goal

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**)
- p_f: forward translator
  - models p(*target*-lang sentence | *source*-lang sentence)
- p_r: reverse translator
  - models p(*source*-lang sentence | *target*-lang sentence)

Inference: For each source sentence $\boldsymbol{x}$, (1) do beam search with beam size 50—100 (**SLOW!**); (2) rerank using the following objective and pick the top-ranked translation

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

Used in many top/winning models in WMT competitions

Can we train a **new network** such that if we do **greedy decoding** using the new network, the translations will maximize $R(\boldsymbol{y}) = \log p_f(\boldsymbol{y} \mid \boldsymbol{x}) + \gamma \log p_r(\boldsymbol{x} \mid \boldsymbol{y})$?

Criteria for successful amortization

Inference speed
Successful if the inference is faster than BSR. 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/BLEURT)
Successful if the BLEURT of our translations are similar to the BLEURT by BSR.

3 Discussion

- "BSR → high BLEURT" doesn't imply "higher reward → higher BLEURT"
- KD/IL-generated translations are similar (in terms of corpus-level BLEU); they are different from Q-generated translations, possibly due to how reverse reward is presented to KD/IL vs. Q
- Q learning also applies to text generation (we trained Q from scratch!) – rarely used in NLG; but Q learning doesn't do well when the source sentence is long (> 80 tokens) possibly due to the optimization difficulty given by the sparse reverse reward

2 Methods and Results

Approach 1: knowledge distillation (KD)

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})

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

Approach 2: one-step deviation imitation learning (IM)

Call our new network A. To train A, intuitively:
Use cross entropy to…
- match the t-th step distribution of A *and* the t-th step distribution of p_f
- match onehot($\boldsymbol{x}_t$) and the t-th step distribution of p_r (p_r is a function of A)

Approach 3: Q learning *adapted from DQN used to train Atari games*

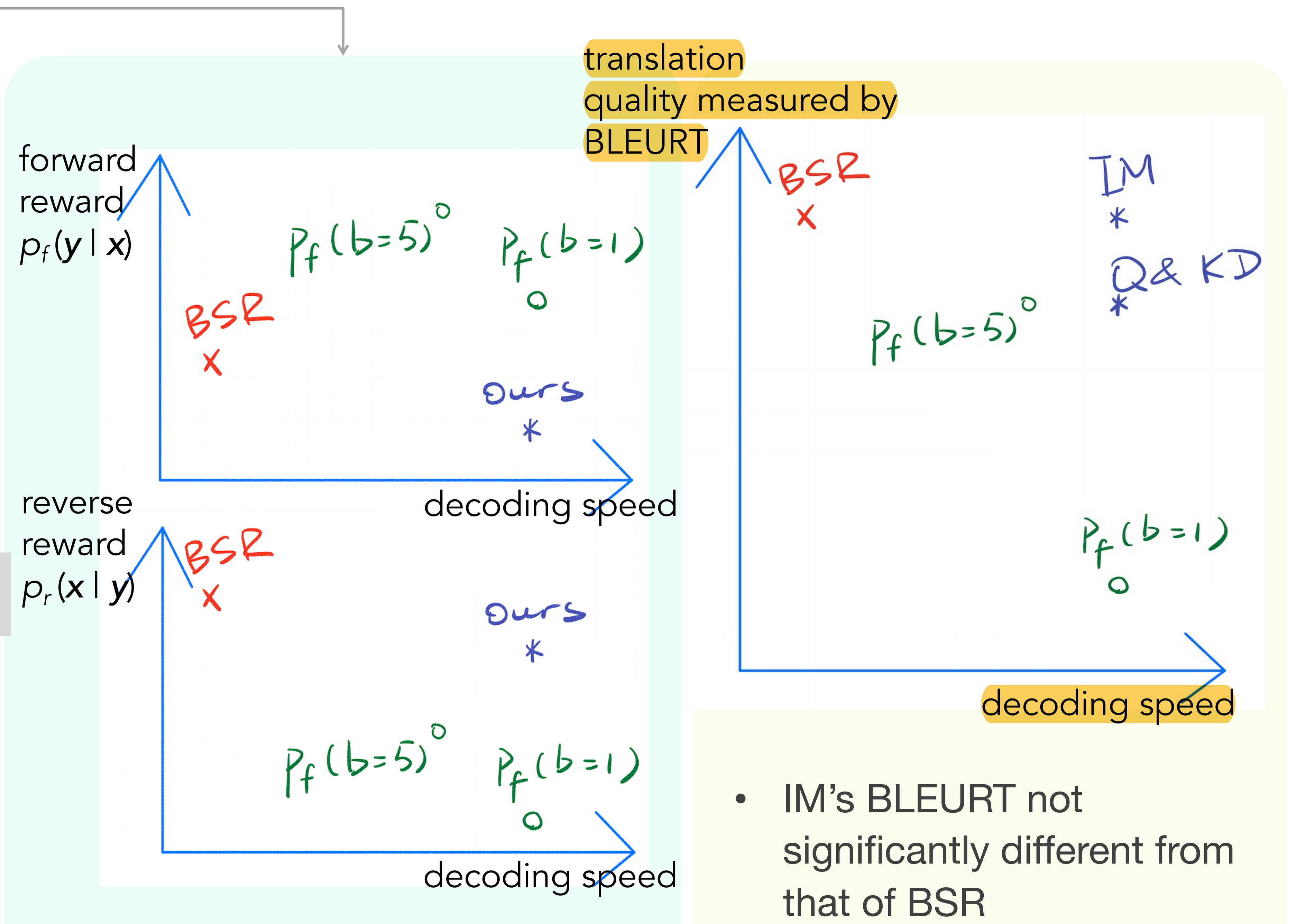
Want: Q ("future return" – higher is better);
Define: $s_t = (\boldsymbol{y}_{<t}, \boldsymbol{x})$, $a_t = y_t$

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

Given p_f, p_r, 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
| $\text{argmin}_\phi\ [\ Q_\phi(s_t, a_t) - R_t\ ]^2$
| Update Q'_ϕ: Q'_ϕ <– Q_ϕ every K steps

Results intuitively…



- Our approach: lower forward reward
- …higher reverse reward than p_f (b=5) but lower than BSR

- IM's BLEURT not significantly different from that of BSR
- IM's BLEURT significantly higher than that of p_f (b=5)