Objective Function for MT

Before, we talked about sequence to sequence models

\[ \ell = -\log p(u^* | \tilde{x}) \]  

- Doesn’t include issue of decoding
Objective Function for MT

Before, we talked about sequence to sequence models

$$\ell = -\log p(u^* | \tilde{x}) + \log \sum_{u \in U(x)} p(u | \tilde{x})$$  \hspace{1cm} (1)

- Doesn’t include issue of decoding
- So normalize by decoder hypotheses
- But this isn’t the right objective function
Why we need Reinforcement Learning

- We know the right answer (oracle)
- We want to reach that answer
- Decoding may not know how to produce it
- Search problem: reinforcement learning
- Learn how to generate correct sequence
Reward

Expected BLEU score $\mathbb{E}_{p_\theta(y|x)}[R(y)] = \ell \equiv \sum_{u \in U(x)} \text{BLEU}(t, u) \frac{p(u|x)}{\sum_{u' \in U(x)} p(u'|x)}$ (2)

- Policy gradient lets us optimize parameters of policy $\theta$
  $\nabla_\theta \text{RL} = \mathbb{E}_{p_\theta(y|x)}[R(y) \nabla_\theta \log p_\theta(y|x)]$ (3)

- REINFORCE estimates gradient of reward with one sample for each input
  $\tilde{\nabla}_\theta \text{RL} = R(\tilde{y}) \nabla_\theta \log p_\theta(\tilde{y}|x), \quad \tilde{y} \sim p_\theta(y|x)$ (4)
Reward

Expected BLEU score $\mathbb{E}_{p_\theta(y|x)} [R(y)] =$

$$\ell \equiv \sum_{u \in U(x)} \text{BLEU}(t, u) \frac{p(u|x)}{\sum_{u' \in U(x)} p(u'|x)}$$  \hspace{1cm} (2)

- Policy gradient lets us optimize parameters of policy $\theta$

$$\nabla_\theta \text{RL} = \mathbb{E}_{p_\theta(y|x)} [R(y) \nabla_\theta \log p_\theta(y|x)]$$  \hspace{1cm} (3)

- REINFORCE estimates gradient of reward with one sample for each input

$$\tilde{\nabla}_\theta \text{RL} = R(\tilde{y}) \nabla_\theta \log p_\theta(\tilde{y}|x), \quad \tilde{y} \sim p_\theta(y|x)$$  \hspace{1cm} (4)

- approximate the policy gradient with either multinomial sampling from the softmax-normalized outputs of the NMT model, or by beam search

- The two objectives are trained either sequentially (e.g., supervised pre-training before reinforced fine-tuning, or alternating batches) or
Sounds Good … What’s the Catch?

- Variance of gradient estimator can prevent convergence
  - Baseline: Subtract empirical average from reward
  - Actor-critic: try to imitate original reward
  - Number of samples for gradient hugely important: over-sample

- Reward shaping
  - Only get reward at end of sentence
  - For token $t$, $R(y_t) = R(y_1:t) - R(y_1:t-1)$ of removing token

- Advantage Actor Critic: learn critic for each element

- Monolingual Data
  - generate pseudo-sources for the available target data
  - models get even better when the pseudo-sources are of low quality
  - like denoising auto-encoders
Sounds Good … What’s the Catch?

- Variance of gradient estimator can prevent convergence
  - Baseline: Subtract empirical average from reward
  - Actor-critic: try to imitate original reward
  - Number of samples for gradient hugely important: over-sample
- Reward shaping
  - Only get reward at end of sentence
  - For token \( t \), \( R(y_t) = R(y_{1:t}) - R(y_{1:t-1}) \) of removing token
  - Advantage Actor Critic: learn critic for each element
Sounds Good . . . What’s the Catch?

- Variance of gradient estimator can prevent convergence
  - Baseline: Subtract empirical average from reward
  - Actor-critic: try to imitate original reward
  - Number of samples for gradient hugely important: over-sample

- Reward shaping
  - Only get reward at end of sentence
  - For token $t$, $R(y_t) = R(y_{1:t}) - R(y_{1:t-1})$ of removing token
  - Advantage Actor Critic: learn critic for each element

- Monolingual Data
  - generate pseudo-sources for the available target data
  - models get even better when the pseudo-sources are of low quality
  - like denoising auto-encoders
Where to go next

- Disagree on environment, state, where reward comes from
- Bandit structured prediction may be better fit
- Improve bias of search: imitation learning mixes model and reference
- Use cheaper references
- Use real-world applications and true interactions