Paper Reading: Neural Machine Translation with Gumbel-Greedy Decoding (AAAI 2018)

Jiatao Gu, Daniel Jiwoong Im, Victor O.K. Li

The University of Hong Kong

Motivation

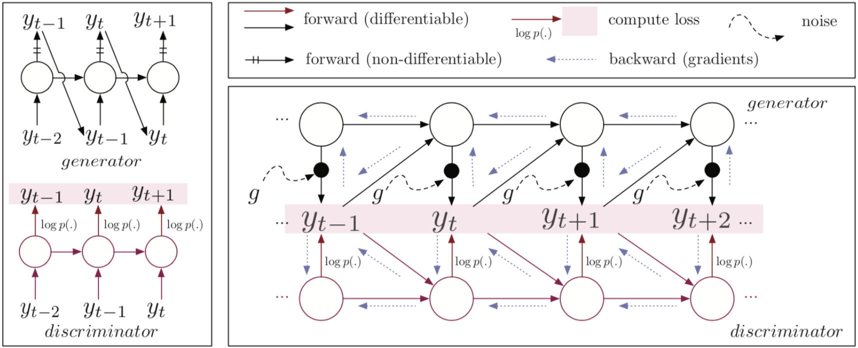
- Once the model is trained, the most probable output which maximum the log-likelihood during training cannot be properly found at the test time.
- Avoid solving the maximum a posteriori problem over translation sentences at test phase.

Discriminator-Generator framework

- NMT-discriminator: measures the log-likelihood at word level-$log_{p_\theta}(Y \mid X)$ given the source sentence X and a translation Y.
- NMT-generator: generates the translation by taking the output of the word as an input to next step recursively, $Y = G_\theta(X)$, given the source sentence X. (G is usually a search-based method).
- NMT discriminator's score:

$$J(\phi) = \mathbb{E}_{Y \sim G_\phi} \log p_\theta (Y|X)$$

Discriminator, Generator and Gumbel-Greedy Decoding



(a) (b)

Gumbel-Greedy Decoding

- The Gumbel-Max Trick:

$$y = \operatorname{argmax}(g + a), g \sim \text{Gumbel i.i.d.}$$

  Where each element in g can be computed using the inverse transform sampling of an auxiliary random uniform variable $u_i\ U(0,1)$, $g_i = -log(-log(u_i))$

- Gumbel-Softmax Relaxation:

$$\hat{y} = \operatorname{softmax}((g + a)/\tau), g \sim \text{Gumbel i.i.d.}$$

  Where $\tau \in (0, \infty)$. The softmax function approaches argmax operations as $\tau \longrightarrow 0$, and it becomes uniform when $\tau \longrightarrow \infty$.

- Stright-Throught(ST) Gumbel: During the forward phase, use the Gumbel-max, while computing the gradient of the Gumbel-softmax.

- Arbitrary Decoding Algorithms as Gumbel-Greedy Decoding:

$$y = \operatorname{argmax}\left(g + a\right), g \sim Q$$

$$g_i^* = \begin{cases} g', & y_i \text{is selected} \\ g' - \log\left[1 + e^{g' - \tilde{g}_i}\right], & \text{otherwise} \end{cases} \quad (12)$$

where

$$\tilde{g}_i = -\log(-\log(u_i)) + a_i, u_i \sim \mathcal{U}(0, 1)$$

and the "top-gumbel"

$$g' = -\log(-\log(u)) + \log\left(\sum_i \exp(a_i)\right), u \sim \mathcal{U}(0, 1)$$

Gumbel-Greedy Decoding Algorithm

- With Regularization:

$$\mathbb{E}_{G_\phi}\left[\log p_\theta\left(Y|X\right)\right] - \mathbb{E}_{G_\phi}\left[\log p_{\phi'}\left(Y|X\right)\right] \quad (13)$$

where we use ϕ' to represent a copy of the current parameters ϕ and make it as a "discriminator". Note that gradients w.r.t ϕ will not flow into ϕ'.

- Adversarial Learning:

$$\mathbb{E}_{D}\left[\log p_\theta\left(Y|X\right)\right] - \mathbb{E}_{G}\left[\log p_{\theta}\left(Y|X\right)\right] \quad (14)$$

where D is the empirical distribution of real translation. In practice, we alternate the training of the generator and the discriminator iteratively.

Algorithm 1 Gumbel-Greedy Decoding

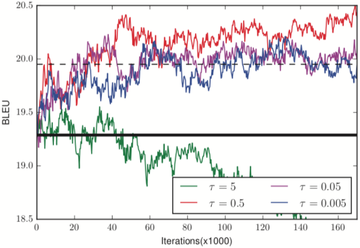
Require: discriminator p_θ, generator G_ϕ, $N_d \geq 0$, $N_g > 0$

Train θ using MLE/REINFORCE on training set D;
Initialize ϕ using θ;
Shuffle D twice into D_θ and D_ϕ
while stopping criterion is not met **do**
  for $t = 1 : N_g$ **do** *// learn the generator*
    Draw a translation pair: $(X, \_) \sim D_\phi$;
    Obtain $Y, \hat{Y} = \textsc{GumbelDec}(G, X)$
    Compute forward pass $\sim X, Y$ with Eq. 13
    Compute backward pass $\sim X, \hat{Y}$, update ϕ
  for $t = 1 : N_d$ **do** *// learn the discriminator*
    Draw a translation pair: $(X, Y^*) \sim D_\theta$;
    Obtain $Y, \_ = \textsc{GumbelDec}(G, X)$
    Compute forward pass $\sim X, Y, Y^*$ with Eq. 14
    Compute backward pass $\sim X, Y, Y^*$, update θ

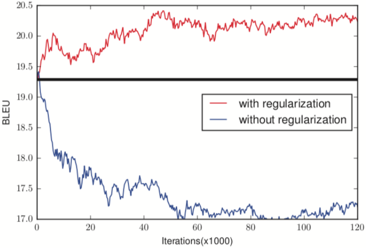
Function: $\textsc{GumbelDec}(G, X)$

if $G =$ 'sampling' **then**
  Sample $g \sim$ Gumbel i.i.d.
  Obtain $Y, \hat{Y}$ with Eq. 7 and Eq. 8
else
  Obtain $Y = G(X)$
  Infer g with Eq. 12
  Obtain $\hat{Y}$ with Eq. 8
Return $Y, \hat{Y}$

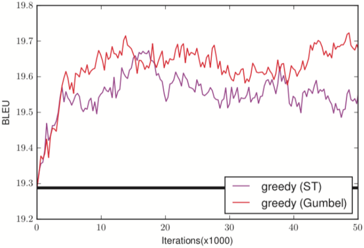
Experiments



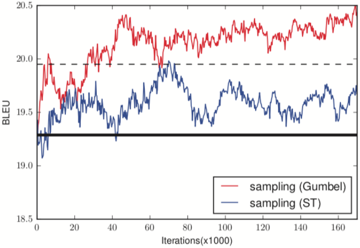
(a) comparison with variant temperature



(b) comparison w/o entropy regularization



(c) comparison of greedy decoding



(d) comparison of sampling

	Model	DE-EN	EN-DE	CS-EN	EN-CS
Greedy	MLE	21.63	18.97	18.90	14.49
	RL	22.56	19.32	19.45	15.02
	GGD	**23.27**	**19.81**	**20.62**	**16.04**
Beam	MLE	24.46	21.33	21.20	16.20
	RL	25.12	**22.13**	21.92	17.02
	GGD	**25.32**	21.97	**22.47**	**17.64**

Table 1: The greedy decoding and the beam-search performance of models trained with GGD-GAN against MLE and REINFORCE (referred to RL). BLEU scores are calculated on the test sets.

Conclusion

- Use the Gumbel-Softmax reparameterization trick to make the generative network differentiable and can be trained through standard stochastic gradient methods.