Reinforcement Learning For Sequence Learning in Neural Network

September 28, 2018

Yanjun Gao
PSU NLP Lab

- Problems
- RL Framework in Neural Network
- Sequence Learning with RL Framework

Research Highlight - Reinforcement Learning in semantic representation

- Why using RL in semantic representation?
  - Most semantic representations are supervised learning. What if the model gets a sentence it has never seen before?
  - Sometimes the approaches are highly relied on the evaluation: how to address this with a good performance? Does one representation fit to all tasks?
  - RL is a hot topic; -> more attentions and publications!

- Why using RL in semantic representation is hard?
  - RL needs definitions of "agent," "states", "actions", "rewards", "policy"; it is a really hard process to formulate a semantic representation problem to RL problem; semantic representation is really different from dialogue modeling, auto-driving, robotics,etc.
  - Tradeoff between exploration and exploitation
  - Training data: where to get the training data is always a big quesiton;
  - Accuracy and performance: does RL always improve performance?
  - Other difficulties: optimizations, etc.

A RL framework for parametric neural network (Williams, 1992)

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3-4 (1992): 229-256.

learning rate factor

reinforcement baseline

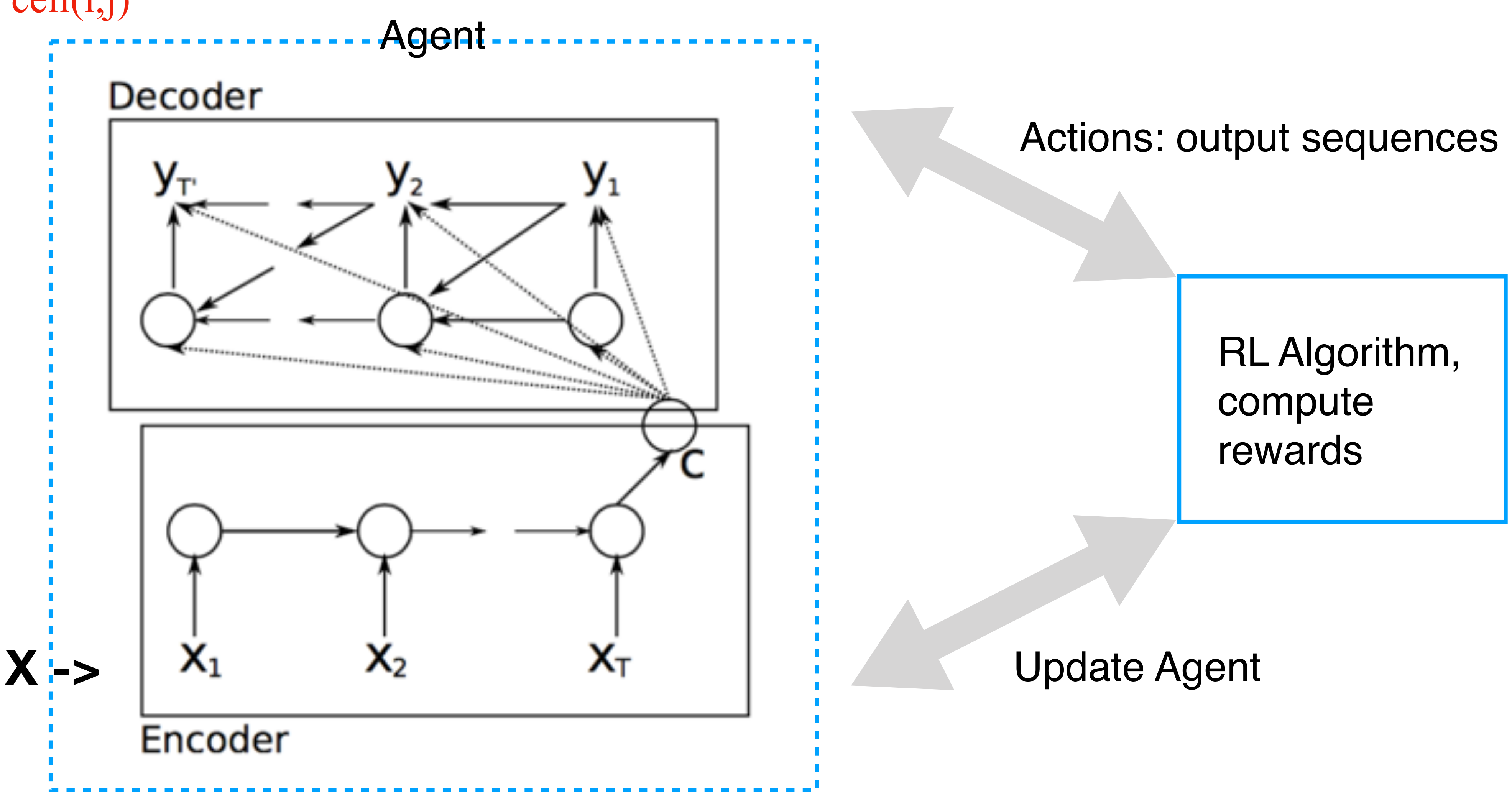
$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij},$$

changes of weight in each cell(i,j)

loss function

$$e_{ij} = \partial \ln g_i / \partial w_{ij}$$

characteristic eligibility of w(i,j)



Goal: learn a set of parameters that generate the best output (maximum rewards)

Could be plugged into any reward function, any networks

A RL framework for parametric neural network (Williams, 1992)

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3-4 (1992): 229-256.

y_i: output of ith unit of the network;

x_i, pattern of input to that unit i.

The output y_i is drawn from the distribution depending on x_i and weights w_ij.

W: weight matrix of all w_ij;

w_i: collection of all parameters on i_th unit.

$$\text{for each } i \text{ let } g_i(\xi, \mathbf{w}^i, \mathbf{x}^i) = \Pr\{y_i = \xi | \mathbf{w}^i, \mathbf{x}^i\}.$$

A RL framework for parametric neural network (Williams, 1992)

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3-4 (1992): 229-256.

Then we define a *stochastic semilinear unit* to be one whose output y_i is drawn from some given probaility distribution whose mass function has single parameter p_i:

$$p_i = f_i(s_i), \tag{1}$$

f_i: differentiable squashing function,
e.g. logistic function

$$s_i = \mathbf{w}^{i^{\mathrm{T}}} \mathbf{x}^i = \sum_j w_{ij} x_j, \tag{2}$$

A RL framework for parametric neural network (Williams, 1992)

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3-4 (1992): 229-256.

When y_i follows Bernoulli distribution, we define Bernoulli semilinear unit with parameter p_i, where all possible output values are 0 and 1. Thus:

$$g_i(\xi, \mathbf{w}^i, \mathbf{x}^i) = \begin{cases} 1 - p_i & \text{if } \xi = 0 \\ p_i & \text{if } \xi = 1, \end{cases}$$

Alternatively,

$$y_i = \begin{cases} 1 & \text{if } \sum_j w_{ij} x_j + \eta > 0 \\ 0 & \text{otherwise,} \end{cases}$$

A RL framework for parametric neural network (Williams, 1992)

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3-4 (1992): 229-256.

Given a parameter to be adapted p_i = Pr(y_i=1), follows Bernouli distribution

$$g_i(y_i, p_i) = \begin{cases} 1 - p_i & \text{if } y_i = 0 \\ p_i & \text{if } y_i = 1, \end{cases} \tag{4}$$

Define characteristic eligibility for parameter p_i:

$$\frac{\partial \ln g_i}{\partial p_i}(y_i, p_i) = \begin{cases} -\frac{1}{1-p_i} & \text{if } y_i = 0 \\ \frac{1}{p_i} & \text{if } y_i = 1 \end{cases} = \frac{y_i - p_i}{p_i(1 - p_i)}, \tag{5}$$

Define characteristic eligibility for weight w_ij:

$$\frac{\partial \ln g_i}{\partial w_{ij}}(y_i, \mathbf{w}^i, \mathbf{x}^i) = \frac{y_i - p_i}{p_i(1 - p_i)} f_i{}'(s_i) x_j, \tag{6}$$

A RL framework for parametric neural network (Williams, 1992)

Williams, Ronald J. "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine learning* 8.3-4 (1992): 229-256.

Set f_i as logistic function, b_ij = 0,

$$\frac{\partial \ln g_i}{\partial w_{ij}}(y_i, \mathbf{w}^i, \mathbf{x}^i) = (y_i - p_i)x_j. \tag{7}$$

Through some substituions, we have the form:

$$\Delta w_{ij} = \alpha r(y_i - p_i)x_j,$$

Adopt “reinforcement comparison” in (Sutton 1984), r_hat is an adpative estimate of upcoming reinforcement based on past experience:

$$\Delta w_{ij} = \alpha(r - \overline{r})(y_i - p_i)x_j, \tag{9}$$

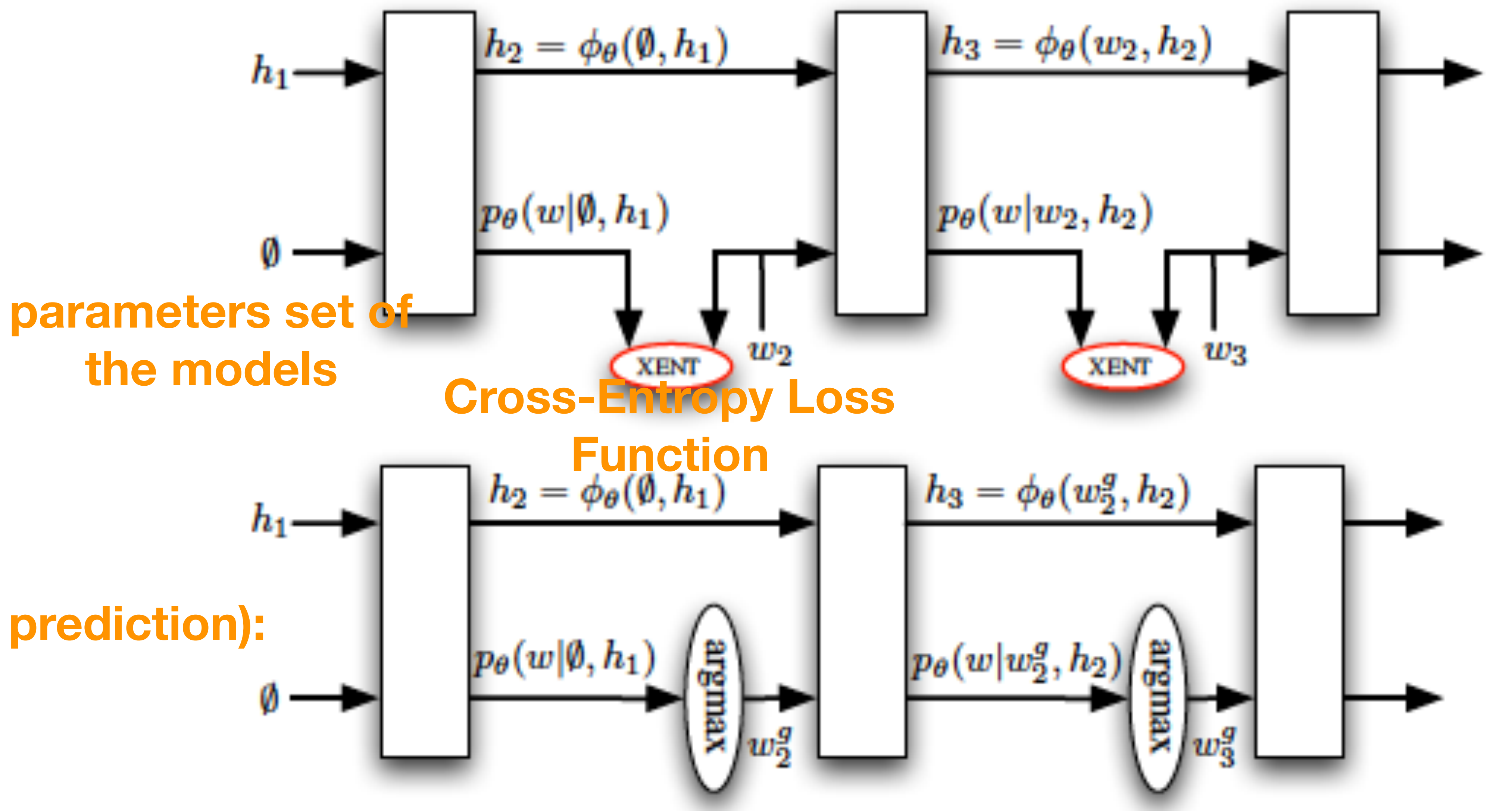
Sequence Level Training with Recurrent Neural Networks

Ranzato, Marc'Aurelio, et al. "Sequence level training with recurrent neural networks." *arXiv preprint arXiv: 1511.06732* (2015).

Typical Elman RNN without RL

Training:

Testing (model prediction):



$$\mathbf{h}_{t+1} = \sigma(M_i \mathbf{1}(w_t) + M_h \mathbf{h}_t + M_c \mathbf{c}_t), \quad (3)$$

$$\mathbf{o}_{t+1} = M_o \mathbf{h}_{t+1}, \quad (4)$$

$$w_{t+1} \sim \text{softmax}(\mathbf{o}_{t+1}), \quad (5)$$

Sequence Level Training with Recurrent Neural Networks

Ranzato, Marc'Aurelio, et al. "Sequence level training with recurrent neural networks." *arXiv preprint arXiv:1511.06732* (2015).

Mixed Incremental Cross-Entropy Reinforce (MIXER)

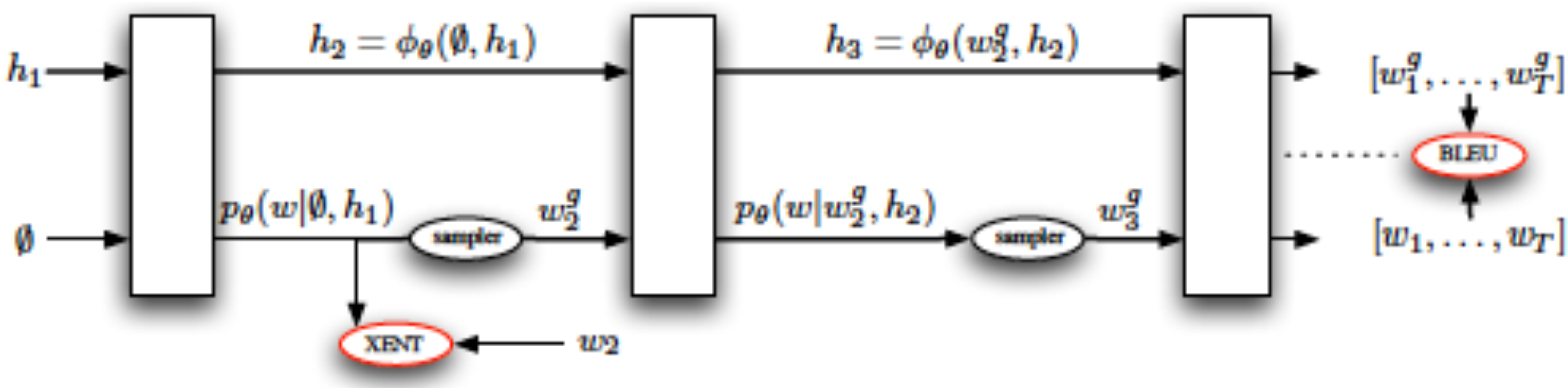


Figure 4: Illustration of MIXER. In the first s unrolling steps (here $s = 1$), the network resembles a standard RNN trained by XENT. In the remaining steps, the input to each module is a sample from the distribution over words produced at the previous time step. Once the end of sentence is reached (or the maximum sequence length), a reward is computed, e.g., BLEU. REINFORCE is then used to back-propagate the gradients through the sequence of samplers. We employ an annealing schedule on s, starting with s equal to the maximum sequence length T and finishing with $s = 1$.

Sequence Level Training with Recurrent Neural Networks

Ranzato, Marc'Aurelio, et al. "Sequence level training with recurrent neural networks." *arXiv preprint arXiv: 1511.06732* (2015).

Mixed Incremental Cross-Entropy Reinforce (MIXER)

$$L_\theta = -\sum_{w_1^g,\ldots,w_T^g} p_\theta(w_1^g,\ldots,w_T^g)\, r(w_1^g,\ldots,w_T^g) = -\mathbb{E}_{[w_1^g,\ldots w_T^g]\sim p_\theta} r(w_1^g,\ldots,w_T^g), \quad (9)$$

$$\frac{\partial L_\theta}{\partial \theta} = \sum_t \frac{\partial L_\theta}{\partial \mathbf{o}_t}\frac{\partial \mathbf{o}_t}{\partial \theta} \quad (10)$$

$$\frac{\partial L_\theta}{\partial \mathbf{o}_t} = \left(r(w_1^g,\ldots,w_T^g) - \bar{r}_{t+1}\right)\left(p_\theta(w_{t+1}|w_t^g, \mathbf{h}_{t+1}, \mathbf{c}_t) - \mathbf{1}(w_{t+1}^g)\right), \quad (11)$$

Sequence Level Training with Recurrent Neural Networks

Ranzato, Marc'Aurelio, et al. "Sequence level training with recurrent neural networks." *arXiv preprint arXiv:1511.06732* (2015).

Data: a set of sequences with their corresponding context.
Result: RNN optimized for generation.
Initialize RNN at random and set N^{XENT}, $N^{\text{XE+R}}$ and Δ;
for $s = T, 1, -\Delta$ **do**
  if $s == T$ **then**
    train RNN for N^{XENT} epochs using XENT only;
  else
    train RNN for $N^{\text{XE+R}}$ epochs. Use XENT loss in the first s steps, and REINFORCE (sampling from the model) in the remaining $T - s$ steps;
  end
end

Algorithm 1: MIXER pseudo-code.

first (T-delta) steps, train XENT

Rest of (T-s) steps, anneal output to a stable sequence

Sequence Level Training with Recurrent Neural Networks

Ranzato, Marc'Aurelio, et al. "Sequence level training with recurrent neural networks." *arXiv preprint arXiv: 1511.06732* (2015).

TASK	XENT	DAD	E2E	MIXER
summarization	13.01	12.18	12.78	**16.22**
translation	17.74	20.12	17.77	**20.73**
image captioning	27.8	28.16	26.42	**29.16**

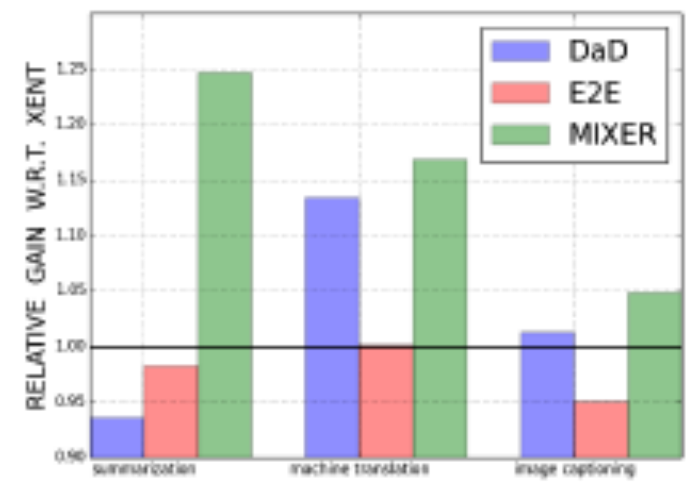


Figure 5: Left: BLEU-4 (translation and image captioning) and ROUGE-2 (summarization) scores using greedy generation. Right: Relative gains produced by DAD, E2E and MIXER on the three tasks. The relative gain is computed as the ratio between the score of a model over the score of the reference XENT model on the same task. The horizontal line indicates the performance of XENT.