SEQUENCE LEVEL TRAINING WITH DEEP REINFORCEMENT LEARNING ALGORITHMS

Harmanpreet Singh, Akshay Singh Rana
Department of Computer Science
McGill University, Canada
{harmanpreet.singh,akshay.singhrana}@mail.mcgill.ca

ABSTRACT

In this paper we present the comparative study of various policy-gradient methods to train deep end-to-end systems directly on non-differentiable metrics for the task at hand. Additionally, we investigate how reinforcement learning helps mitigate the problem of exposure bias in seq2seq models, in particular machine translation. The experimental results show that the Self-Critic method demonstrates 6.97% improvement in BLEU score on Portuguese to English translation. [1]

1 INTRODUCTION

Recent studies have shown that reinforcement learning as an effective approach for improving the performance of sequence-to-sequence (seq2seq) models in. The underlying framework for all these models is usually a deep neural network comprising an encoder and a decoder. However, such seq2seq models suffer from the problems of inconsistency between train/test performance measurement and exposure bias. The cross-entropy loss is used for training the model whereas the test set is evaluated on different metrics such as BLEU, ROGUE etc. The token-level objective function during training is inconsistent with sequence-level evaluation metrics during testing. By using reinforcement algorithm, we can address this issue by directly optimizing the metric during training as the one used at test time and significant gains in performance can be realized. Second problem arises during training, the decoder uses ground-truth input from previous time step, to calculate its current output state and uses it to generate the next action. This approach has been called "Teacher-Forcing". However, at the test time, the decoder completely relies on the previously generated action from the model distribution to predict the next action, since the ground truth data is not available anymore. Therefore, in summary, the input to the decoder is from the ground-truth during training, but the input comes from the model distribution during model testing. This exposure bias results in error accumulation during the output generation at test time, since the model has never been exclusively exposed to its own predictions during training. To avoid the exposure bias problem, we need to remove the ground-truth dependency during training. There are multiple ways to address this issue and a lot of work has gone into it.

In this project we consider the problem of Neural Machine Translation (NMT) using Reinforcement Learning (RL). To address the above inconsistencies , RL algorithms have been adopted to optimize sequence-level objectives. For example, policy optimization methods such as REINFORCE (Ranzato et al., 2015) and Actor-Critic (Bahdanau et al., 2016) are leveraged for sequence generation tasks including NMT. In machine translation community, a similar method is proposed with the name "minimum risk training" (Shen et al., 2015). All these works demonstrate the effectiveness of RL techniques for NMT models. However, all these approaches have been tried on LSTM models, we tried to replicate the same approaches in a transformer model and compared REINFORCE, REINFORCE-Critic and Self-Critic methods on a low-resource machine translation problem. We also tried implementing Actor-Critic methods for NMT (Bahdanau et al., 2016) with the transformer networks but since these methods relied on partial rewards from critic at each time-step, it wasn't efficiently feasible to integrate them with transformers where the training takes place parallelly across the entire sequence.

[1] The source code is available at: https://github.com/harmanpreet93/nmt-with-rl

2 RELATED WORK

Several approaches to overcoming the exposure bias problem described in the previous section have recently been proposed. One way to handle this situation is through the scheduled sampling method proposed in (Bengio et al., 2015). In scheduled sampling, the model is first pre-trained using cross-entropy loss and will subsequently and slowly replace the ground-truth with a sampled action from the model. Another line of work proposes "Professor-Forcing" (Lamb et al., 2016) technique that uses adversarial training to encourage the dynamics of the recurrent network to be the same when training conditioned on ground truth previous words and when sampling freely from the network. Problems of non-differentiable evaluation metric have been approached before that aim to approximate the gradient of the expected score. One such approach is "Direct Loss Minimization" (Hazan et al., 2010) in which the inference procedure is adapted to take both the model likelihood and task-specific score into account.

There have been works incorporating techniques from reinforcement learning to address both the exposure bias and non-differentiable task metric issues. In other recent RL-inspired work on sequence prediction, (Ranzato et al., 2015) trained a translation model by gradually transitioning from maximum likelihood learning into optimizing BLEU or ROUGE scores using the REINFORCE algorithm. The major limitation of the approach is that the expected gradient computed using mini-batches under REINFORCE typically exhibit high variance, and without proper context-dependent normalization, is typically unstable. Moreover, these methods does not exploit the availability of the ground-truth like the critic network does. (Bahdanau et al., 2016) trained an additional network called the critic to output the value of each token, which they define as the expected task-specific score that the network will receive if it outputs the token and continues to sample outputs according to its probability distribution. Furthermore, they use the main model to train the sequence prediction network, which is referred to as the actor and it utilizes Temporal-Difference (TD) returns at each timestamp. They also experimented with the original REINFORCE algorithm and improved its baseline network by comparing it with the ground truth instead of relying on the estimated linear regressor for rewards, called Reinforce-Critic. We believe that estimating the baseline while comparing it with the ground truth is helpful to reduce the variance of the agent. While their method requires an additional neural network, called a critic model, to predict the expected reward and stabilize the objective function gradients, (Rennie et al., 2017) designed a self-critical sequence training method that does not require this critic model and lead to further improvements on image captioning tasks. additionally, we explored different ways to merge the Cross-Entropy and reinforcement learning loss using a simple weighted addition inspired from Mixed Incremental Cross-Entropy Reinforce (MIXER) (Ranzato et al., 2015). Following sections describe these methods in more detail.

3 METHODOLOGY

We use self-attention based transformer model for our task of machine translation. They exhibit the ability to attend to different positions of the input sequence to compute a representation of that sequence using self-attention to draw global dependencies between input and output. In this section we first discuss about Cross Entropy 3.1 as the standard objective used in sequence to sequence model. Later we introduce how sequence generation can be reinforcement learning problem 3.2 and elaborate on its algorithms in Sections 3.2.1, 3.2.2 and 3.2.3. We finally devise a hybrid objective function in section 3.3.

3.1 CROSS ENTROPY WITH TEACHER FORCING

Cross-entropy loss (XENT) maximizes the probability of the observed sequence according to the model. We define $y^* = \{y_1^*, y_2^*, ...y_T^*\}$ as the target sequence for a given input sequence x. Let θ denote the parameters of the model. The maximum-likelihood training objective is the minimization of the following loss:

$$\mathrm{L}_{\mathrm{XENT}} = -\sum_{t=1}^{T} \log p_\theta(y_t^* | y_1^*, y_2^*, ...y_{t-1}^*, x) \tag{1}$$

3.2 SEQUENCE GENERATION AS REINFORCEMENT LEARNING PROBLEM

In order to apply the REINFORCE algorithm (Williams, 1992; Zaremba Sutskever, 2015) to the problem of sequence generation we cast our problem in the reinforcement learning (RL) framework (Sutton Barto, 1988). Our model can be viewed as an agent, which interacts with the external environment (the words and the context vector it sees as input at every time step). The parameters of this agent defines a policy, whose execution results in the agent picking an action. In the sequence generation setting, an action refers to predicting the next word in the sequence at each time step. Once the agent has reached the end of a sequence, it observes a reward i.e. BLEU score in our case which is also the evaluation metric. During training the agent observes a reward r computed by an evaluation metric by comparing the generated sequence to corresponding ground-truth sequences. The goal of training is to minimize the negative expected reward:

$$L(\theta) = -\mathbb{E}_{y^s \sim p_\theta}[r(y^s)]$$

where $y^s = y_1^s, y_2^s, ...y_T^s$ and y^s is the word sampled from the model at the time step t. It is impossible to exactly maximize L_θ due to the large vocabulary size. Therefore in practice, we approximate the above expectation via sampling $\hat{y}$ from the policy p_θ, leading to the objective as maximizing:

$$\hat{L}(\theta) \approx -r(y^s), y^s \sim p_\theta$$

3.2.1 POLICY GRADIENT WITH REINFORCE

In order to compute the gradient $\nabla_\theta L(\theta)$, we use the REINFORCE algorithm. REINFORCE is based on the observation that the expected gradient of a non-differentiable reward function can be computed as follows:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{y^s \sim p_\theta}[r(y^s)\nabla_\theta \log p_\theta(y^s)]$$

In practice the expected gradient can be approximated using a single Monte-Carlo sample $y^s = y_1^s, y_2^s, ...y_T^s$ from p_θ, for each training example in the mini-batch:

$$\nabla_\theta L(\theta) \approx -r(y^s)\nabla_\theta \log p_\theta(y^s)$$

3.2.2 REINFORCE WITH BASELINE

The policy gradient given by REINFORCE can be generalized to compute the reward associated with an action value *relative* to a reference reward or *baseline* b:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{y^s \sim p_\theta}[(r(y^s) - b)\nabla_\theta \log p_\theta(y^s)]$$

Since we can prove that $\mathbb{E}_{y^s \sim p_\theta}[b\nabla_\theta \log p_\theta(y^s)] = 0$, implies that the baseline can be an arbitrary function, since it does not depend on action y^s. This shows that the baseline does not change the expected gradient, but importantly, it can reduce the variance of the gradient estimate. For each training case, we again approximate the expected gradient with a single sample $y^s \sim p_\theta$:

$$\nabla_\theta L(\theta) \approx -(r(y^s) - b)\nabla_\theta \log p_\theta(y^s)$$

Using the chain rule, and the parametric model of p_θ, we have:

$$\nabla_\theta L(\theta) = \sum_{t=1}^{T} \frac{\partial L(\theta)}{\partial s_t} \frac{\partial s_t}{\partial \theta},$$

where s_t is the input to the softmax function. Using REINFORCE with a baseline b the estimate of the gradient of $\frac{\partial L(\theta)}{\partial s_t}$ is given by:

$$\frac{\partial L(\theta)}{\partial s_t} \approx (r(y^s) - b)(p_\theta(y_t|y_1, ..y_{t-1}, x) - 1_{y_t^s})$$

For this training algorithm, we produce two separate output sequences at each training iteration: y^s, which is obtained by sampling from the $p(y_t^s|y_1^s, y_2^s, ...y_{t-1}^s, x)$ probability distribution at each decoding time step, and $\hat{y}$, the baseline output, obtained by maximizing the output probability distribution at each time step, essentially performing a greedy search. We define $r(y)$ as the reward function for an output sequence y, comparing it with the ground truth sequence y^* with the evaluation metric of our choice.

$$\mathrm{L_{RL}} = -(r(y^s) - r(\hat{y})) \sum_{t=1}^{T} \log p(y_t^s|y_1^s, y_2^s, ...y_{t-1}^s, x) \tag{2}$$

We can see that minimizing L_{RL} is equivalent to maximizing the conditional likelihood of the sampled sequence y^s if it obtains a higher reward than the baseline $\hat{y}$, thus increasing the reward expectation of our model.

3.2.3 SELF-CRITIC MODEL

The above techniques rely on estimating future reward by outputs from another trainable network which can also be unstable. (Rennie et al., 2017) designed a self-critical sequence training (SCST) method that does not require this critic model and lead to further improvements on image captioning tasks. SCST is a REINFORCE algorithm that, rather than estimating the reward signal, or how the reward signal should be normalized, utilizes the output of its own test-time inference algorithm to normalize the rewards it experiences. As a result, only samples from the model that outperform the current test-time system are given positive weight, and inferior samples are suppressed. Finally, SCST is self-critical, and so avoids all the inherent training difficulties associated with actor-critic methods, where a second "critic" network must be trained to estimate value functions, and the actor must be trained on estimated value functions rather than actual rewards. Like the original paper, we also focus on scenario of greedy decoding.

(Paulus et al., 2017) has used SCST in the Abstractive Summarization to achieve delta improvements. To the best of our knowledge, Self-critic model has not been explored in the NMT tasks and we are the first ones to explore this with transformer networks, while using two different hybrid learning objective function. We are also comparing it with existing approaches for REINFORCE, and REINFORCE-Critic.

3.3 HYBRID LEARNING OBJECTIVE

There are various ways to combine two different losses i.e. $\mathrm{L_{XENT}}$ and $\mathrm{L_{RL}}$. (Ranzato et al., 2015) used an incremental scheduling algorithm called 'MIXER'. In this method, the model is trained with the cross-entropy loss for $\mathrm{N_{XENT}}$ epochs using the ground-truth sequences. This ensures that the model starts off with a much better policy than random because now the model can focus on promising regions of the search space. Then, they used an annealing schedule in order to gradually teach the model to produce stable sequences. Therefore, after the initial $\mathrm{N_{XENT}}$ epochs, they continue training the model for $\mathrm{N_{XENT+RL}}$ epochs, such that, for every sequence, they use the $\mathrm{L_{RL}}$ for the first (T$-\delta$) steps, and the REINFORCE algorithm for the remaining δ steps. The MIXER model was successfully used on a variety of tasks such as text summarization, image captioning, and machine translation. We took inspiration from this method and trained our model initially on cross-entropy loss before using the hybrid loss.

Another way to handle the transition from using $\mathrm{L_{XENT}}$ and $\mathrm{L_{RL}}$ is to use the following combined loss by combining equations [1] and [2]:

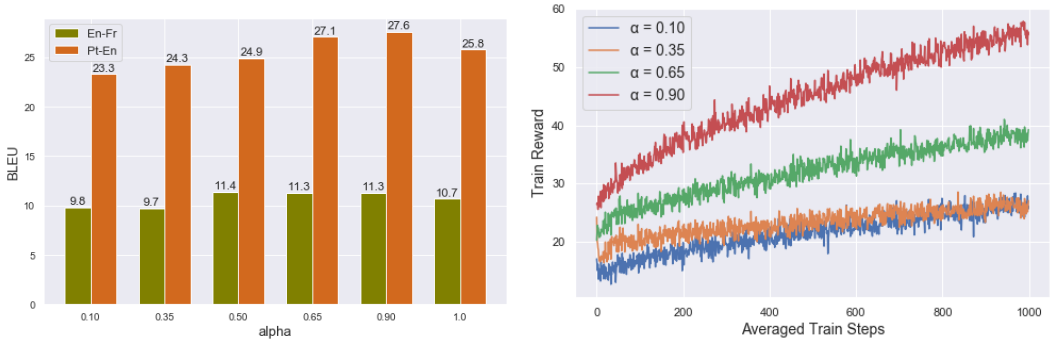
$$\mathrm{L_{HYBRID}} = \alpha \mathrm{L_{XENT}} + (1 - \alpha)\mathrm{L_{RL}} \tag{3}$$

where α is a scaling factor accounting for the difference in magnitude between L_{XENT} and L_{RL} We will empirically evaluate how different values of α impact the final translation accuracy.

Table 1: Quantitative BLEU score results for various models evaluated on the test set. The model with HYBRID objective function was initially trained on L_{XENT} for first 25 epochs. The models were trained until convergence with $\alpha = 0.75$ except L_{XENT} for which the $\alpha = 1$

Training Strategy	Pt-En	En-Fr
L_{XENT}	25.8	10.7
REINFORCE + L_{HYBRID}	25.4	10.1
REINFORCE-Critic + L_{HYBRID}	26.9	**11.3**
Self-Critic + L_{HYBRID}	**27.6**	**11.4**

Figure 1: 2(a): Results of different weights α to combine XENT and RL objectives in equation 3 on held-out validation sets. 2(b): Training reward averaged over 1000 training steps for different alpha values.



4 Experiments and Results

4.1 Datasets

For our translation experiment, we evaluate the mixed-objective learning on two datasets, 10k parallel samples for English-French, and 50k samples on Portuguese-English dataset. We chose to investigate low-resource machine translation scenario to clearly distinguish influence of hybrid loss function on our datasets. We report widely used BLEU score metrics to measure quality of translations.

4.2 Implementation Details

We follow the smallest architecture stated in the original transformer paper Experiments were conducted using the transformer encoder-decoder model (Vaswani et al., 2017) with vocab size of 20k each for both datasets. The transformer encoder and decoder are composed of a stack of N = 4 identical layers with embedding dimension size of 128 and 8 attention heads. We first run maximum-likelihood (ML) training and select the best performing architecture. Since our L_{XENT} loss based model saturates at around 20-25 epochs, we initially train it for 25 epochs.

Next, we initialize our model with the best ML parameters and we compare reinforcement learning (RL) with our mixed-objective learning (ML+RL), following our objective functions in Equation 3. Normalized version of the sacreBLEU score was utilized as the reinforcement learning reward. For both ML and hybrid training, we use the teacher forcing algorithm. Our intentions were not to reproduce the state of the art performances, but instead measure the impact of RL on sequence to sequence tasks. In particular, we want to know whether the ML+RL training objective did improve readability compared to RL. We perform human evaluation to ensure that our increase in BLEU scores is also followed by an increase in human readability and quality. Some example translations generated by our model are shown in appendix table 2 for reference.

4.3 RESULT DISCUSSIONS

For L_{HYBRID}, we tried different α values, and 2(a) show that a value between 0.5 to 0.75 seems to be appropriate in both our translation task. Although the model were trained on the original cross entropy loss for 25 epochs. This ensures that the model starts off with a much better policy than random because now the model can focus on promising regions of the search space. For all our Reinforcement models, the sampling was done using a Multinomial distribution. REINFORCE without baseline was expected to perform poor as it has high variance. A single return is returned after taking actions(choosing next words) along the entire sequence length. To overcome this, instead of using a estimated baseline function as in (Ranzato et al., 2015), we made extension to the baseline by leveraging the extra information available in the ground-truth output as suggested by (Bahdanau et al., 2016). We basically use a critic model as a function of predicted and true values to estimate the expected return with REINFORCE. As shown in the table 1, it performed much better than the REINFORCE alone. Also it exceeded the model trained with only cross entropy loss by a good margin of 1.1 BLEU Score in Pt-En and 0.7 in En-Fr. Please note that the En-Fr was trained only on 10k pair of samples and Pt-En on 50k parallel samples. The same approach could not be used for temporal difference Actor-Critic algorithm because unlike in LSTM based networks, the transformers relies on self attention and are trained parallelly across all tokens in the sequence. Hence, we could not implement an efficient way to receive reward after each time-step. Another approach of reinforcement learning where the rewards from the critic model is estimated by the reward signal of its own test-time inference algorithm. We got a significant improvement of 1.8 BLEU from the model trained only on cross-entropy for Pt-En and 0.7 improvement for En-Fr. We believe that self-critic performs as good as the REINFORCE-critic but without the inherent training difficulties associated with the critic model. Our results for machine translation are shown in table 1. We observe that using hybrid objective function helps our model achieve better BLEU scores on both the datasets.

5 CONCLUSION AND FUTURE WORK

Self-Critical Sequence Training (SCST) has all the advantages of REINFORCE algorithms, as it directly optimizes the true, sequence-level, evaluation metric, but avoids the usual scenario of having to learn a (context-dependent) estimate of expected future rewards as a baseline. Since the SCST baseline is based on the test-time estimate under the current model, SCST is forced to improve the performance of the model under the inference algorithm used at test time. This encourages training/test time consistency like the maximum likelihood-based approaches "Data as Demonstrator" (Venkatraman et al., 2015), "Professor Forcing" (Lamb et al., 2016), but importantly, it can directly optimize sequence metrics. Also, it is not easy to make RL practically effective given quite a few widely acknowledged limitations of RL method such as high variance of gradient estimation and objective instability. We also observed that giving more weightage to the Reinforcement objective function does not guarantee an increase in quality and readability of the output. It is possible to game such discrete metrics and increase their score without an actual increase in readability or relevance. Therefore, more research and work needs to be done for the smooth integration into supervised problems.

STATEMENT OF CONTRIBUTION

This paper was jointly written and edited by all the authors. Code and experiments were performed cooperatively.

REFERENCES

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Tamir Hazan, Joseph Keshet, and David A McAllester. Direct loss minimization for structured prediction. In *Advances in neural information processing systems*, pp. 1594–1602, 2010.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pp. 4601–4609, 2016.

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7008–7024, 2017.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

A Appendix

PORTUGUESE - ENGLISH TRANSLATIONS

Table 2: Sample PT-EN translations with Self-Critic Model

Source	e isso significava que ela sabia que tinha apenas uma oportunidade para recolher os seus dados.
Machine translation	and that meant that she knew she had just an opportunity to collect the data.
Human	and that meant she knew she only had one shot at collecting her data.
Source	o que aconteceu é o mesmo que num acelerador de partículas porém a muito grande , grande escala.
Machine translation	what happened is the same as in a particle accelerator but far , big scale.
Human	what 's happened is a particle accelerator at a huge , huge scale.
Source	encontrar uma nova cultura também iniciou o meu hábito de leitura comparada.
Machine translation	finding a new culture also started my habit of reading from reading.
Human	encountering a new culture also started my habit of comparative reading.