BACKPLAY FOR NEURAL MACHINE TRANSLATION

Manikanta Srikar Yellapragada
New York University
msy290@nyu.edu

May 17, 2019

1 Abstract

Current log likelihood training methods for Neural machine translation (NMT) are limited due to their inconsistency between training and testing as models are expected to generate tokens based on its previous outputs during testing and use the ground truth tokens during training. Reinforcement learning (RL) can be used to bridge the gap between training and inference of NMT and allows us to optimize on evaluation metric such as BLEU during training time. A popular practice for using RL for NMT is to combine it with monolingual data and maximum likelihood estimate [Wu et al., 2018]. This report will focus on the challenges that are faced in training without using log likelihood and monolingual data by using Backplay [Resnick et al., 2018], which has been proven effective in environments with sparse rewards.

2 Introduction

Recurrent neural networks have shown impressive results on various sequence generation problems like machine translation [Bahdanau et al., 2014]. They are usually trained to maximize the likelihood of tokens in the target sentence, given source sentence and previous ground truth target tokens as input with teacher forcing, where ground truth words are fed back to the model. This causes problems like exposure bias where the ground truth words are not available during inference. In such cases, if the decoder is fed its own output, one bad prediction in the sequence would lead to divergence from the right trajectory. The token level objective function used during training is not consistent with sequence level evaluation metrics such as BLEU.

To address these inconsistencies, RL methods have been adopted for sequence to sequence objectives. Algorithms like scheduled sampling [Bengio et al., 2015] and DAgger [Ross et al., 2011] were designed to overcome the problem of exposure bias by gradually exposing the model with its own outputs during training (imitation learning). Using policy gradients in the training objective has a similar effect because the gradient update is done on the log likelihood of model's output. The inconsistency with evaluation metrics is also solved because the optimization is done on the sentence level reward (sentence BLEU) that is closer to the corpus BLEU.

When RL is used for NMT, for generating an entire sequence, there is only one terminal reward available. This would make the RL training inefficient due to the sparsity of rewards, because the agent needs to take many actions to generate a complete sentence, but only one reward is available for all those actions. To address this, we use Backplay [Resnick et al., 2018] which has been proven effective in environments with sparse rewards. We start with the case where the agent has to generate the last word in the target sentence, and move backwards, until the agent only receives the source sentence and has to generate the entire target sentence.

3 Background

This section introduces the attention based sequence to sequence framework for Neural machine translation and the setting of NMT as an RL problem.

3.1 Neural machine translation

A sequence to sequence model is a model that takes a sequence of items and outputs another sequence of items. In neural machine translation, a sequence is a series of words which outputs a series of words. NMT models the conditional probability $p(y/x)$ of translating a source sentence $(x_1, x_2, ...x_n)$ to a target sentence $(y_1, y_2, ...y_m)$. Attention-based networks refer to the context vector c throughout the translation process [Bahdanau et al., 2014]. NMT consists of an encoder which takes the source sentence and computes a context vector c = $(c_1, c_2, ...c_n)$, and a decoder which generates the target sentence one word at a time and hence decomposes the conditional probability as:

$$\log p(y/x) = \sum_{j=1}^{m} \log p(y_j \mid \{y_1, y_2, ...y_{j-1}\}, c)$$

with an RNN, the conditional probability of decoding each word y_j is

$$p(y_j \mid \{y_1, y_2, ...y_{j-1}\}, c) = g(y_{j-1}, h_j, c)$$

where g is the nonlinearity, h_j is the RNN hidden state. Given N training pairs $\{x^i, y^i\}_{i=1}^{N}$, the maximum likelihood training objective can be framed as

$$\begin{aligned} L_{mle} &= \sum_{i=1}^{N} \log p(y^i/x^i) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{m} p(y_j^i \mid \{y_1^i, y_2^i, ...y_{j-1}^i\}, x^i) \end{aligned}$$

3.2 NMT as an RL problem

Reinforcement learning (RL) is used to bridge the gap between training and inference of NMT and allows us to optimize on evaluation metric such as BLEU during training time. The idea is

to introduce rewards to encourage model outputs that would not only obtain a high reward, but also perfect translation. Rewards can be simulated with sentence level BLEU scores that would encourage the model to generate samples with high BLEU score. The training objective would be to maximize the expected reward for all model outputs.

NMT model can be viewed as the agent, that interacts with the environment, which gives a pair of source sentence and previously generated tokens at every step. The conditioned RNN acts as a stochastic policy that generates actions (a word to pick from target language vocabulary). Reward is observed once the agent completes generating the target sequence. The goal of RL training is to maximize the expected reward. We use a technique called Backplay [Resnick et al., 2018] , in which the agent is started at the end of a demonstration, and the policy is learnt in this setup. The agent is then moved to the starting point backward, until the agent is trained only on the initial state of the task. Backplay requires fewer number of samples vastly decreases the exploration time relative to standard RL and performs relative to reward shaping and behavioral cloning.

4 Related work

The problem of credit assignment was addressed by reward shaping in [Bahdanau et al., 2016], in which for each element in the output, reward is the difference between BLEU score for partial output including and excluding the element. They propose an actor critic method for generating sequences using RL in which they replace the reward obtained by the environment by a reward given by a critic that is trained to imitate the original reward which allows optimization for BLEU.

[Shen et al., 2015] propose minimal risk training for NMT in which evaluation metrics are used as loss function with the assumption that optimization of model parameters would minimize the expected loss on training data. Their approach allows arbitrary sentence level loss functions which are not necessarily differentiable. Their approach is to replace the domain over which the task score expectation is defined with a small subset of it, sample multiple times and average the gradients over this subset.

[Wu et al., 2018]'s study provides a comparison on several important factors in RL training and a method to make use of monolingual data. The study finds that using RL leads to marginal improvements over well tuned baselines, when used along with MLE and monolingual data. They leverage not only target-side monolingual data, but also source-side monolingual data. Their performance gain from reward shaping is quite small. They also combine the MLE and RL objectives to stabilize the RL training process. However, the largest portions of improvement come from leveraging additional monolingual data. [Nguyen et al., 2017] describe a RL algorithm that improves neural machine translation from simulated human feedback. They use advantage actor critic framework in which a critic network predicts the expected future reward for each element reward from the environment (BLEU) is only given at the end of the sequence.

5 Proposed Approach

Reinforcement learning is used to bridge the gap between training and inference of NMT, by optimizing on evaluation metric (sentence BLEU) at training time. NMT model is the *agent* which interacts with the *environment* (which gives a pair of the source sentence and all the previously generated tokens). The agent will pick an action, ie a word from the target language vocabulary according to the policy. Reward is 0 if the selected action is not $< eos >$ and smoothed BLEU score computed between the source and the generated sentence otherwise.

Since this is a very hard exploration problem, we plan to use backplay [Resnick et al., 2018]. Rather than making the decoder generate the entire target sentence, we start at the end of demonstration where the goal is to generate the last token, given the first n-1 tokens. We then move backwards until we reach the initial state where only the $< sos >$ token is present. Since backplay helps with exploration, this would make the problem simpler than generating the target sentence from scratch.

Since bleu score is only valid for sentences of length greater than 4, backplay training was started from last 4 words, ie in the initial stage, observation was the source sentence and last 4 words of the target sentence. The agent would pick an action (next word in the target sentence) from the action space (set of all possible words in the target vocabulary). Reward was sentence bleu score at the end of a demonstration and 0 everywhere else. Initially, the objective function used was PPO loss (value loss + action loss + entropy loss), but later on log likelihood was also used during training. Evaluation metric was corpus bleu score on the validation data. Dataset used was IWSLT German to English.

To reduce variance, only part of the data collected was used for forward pass. If number of parallel processes collecting data is p, and the number of steps or interactions with the environment per epoch is k, the total data collected in an epoch would be km. Let B be the maximum amount that can be fit in a batch. Out of the km data collected, only a small amount, divided into a few batches, say n is used for forward pass of the model. The remaining $km - nB$ data is thrown away and this process is repeated. The intuition behind throwing away data is that the data collected would be highly correlated and doing backprop on it would have the same effect as using a high learning rate.

6 Training details

The agent's policy is determined by a 2 layer LSTM with 512 hidden units in the encoder and decoder. The hidden layer output of the decoder was passed through a linear layer of 512 dimensions which would act as the value network. The agent is trained using Proximal Policy Update [Schulman et al., 2017]. Training for all the experiments was done on Nvidia k40 GPU for 2 days each. Hyperparameters can be found in table 1.

7 Experiments with backplay

Initial experiments were done with a much simpler setup in which the training data was reduced to 100 sentences, and the action space was restricted only to the words occurring in the target sentences used for training. The purpose of this experiment was to check if the model can overfit on the training data. When reward was 0 everywhere else and sentence level bleu score at the end of a generation, the model was unable to overfit. Possible reasons for this could be due credit assignment, sparsity of rewards and large action space.

To deal with this, we used a dense reward scheme. At step t, let $g = (g_1, g_2, ...g_t)$ be the target sentence generated by the decoder and $y = (y_1, y_2, ...y_t)$ be the ground truth target sentence. Let L be the total number of missing words in the target sentence. Reward at that step would be

$$Reward = \begin{cases} \frac{len(g)}{L} & g_t == y_t \\ 0 & otherwise \end{cases}$$

A transition from last n missing words to n+1 missing words was made when the model was able to remember the last n words well. This was determined by a threshold on the reward. This change in rewards brought significant improvement in the number of words the model could remember or overfit. When 3, 10, 50, and 100 sentences were used as the training data, the model could memorize the last 35, 20, 6, and 3 words respectively. A plot showing the number of words the model could remember vs number of sentences used for training is shown in Figure 1.

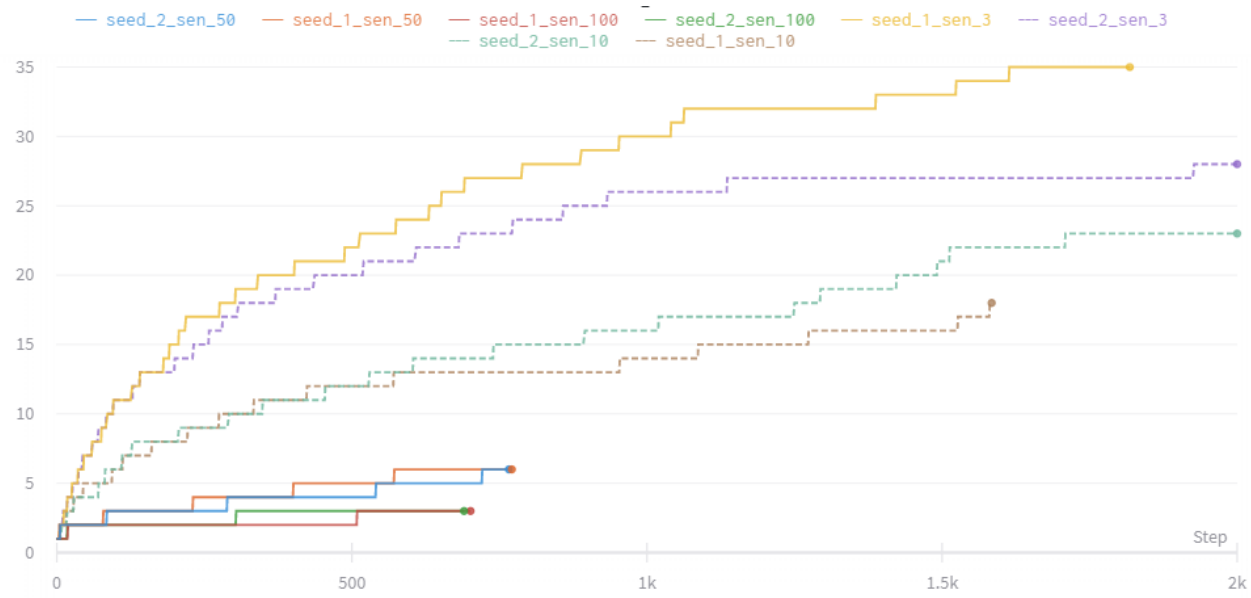
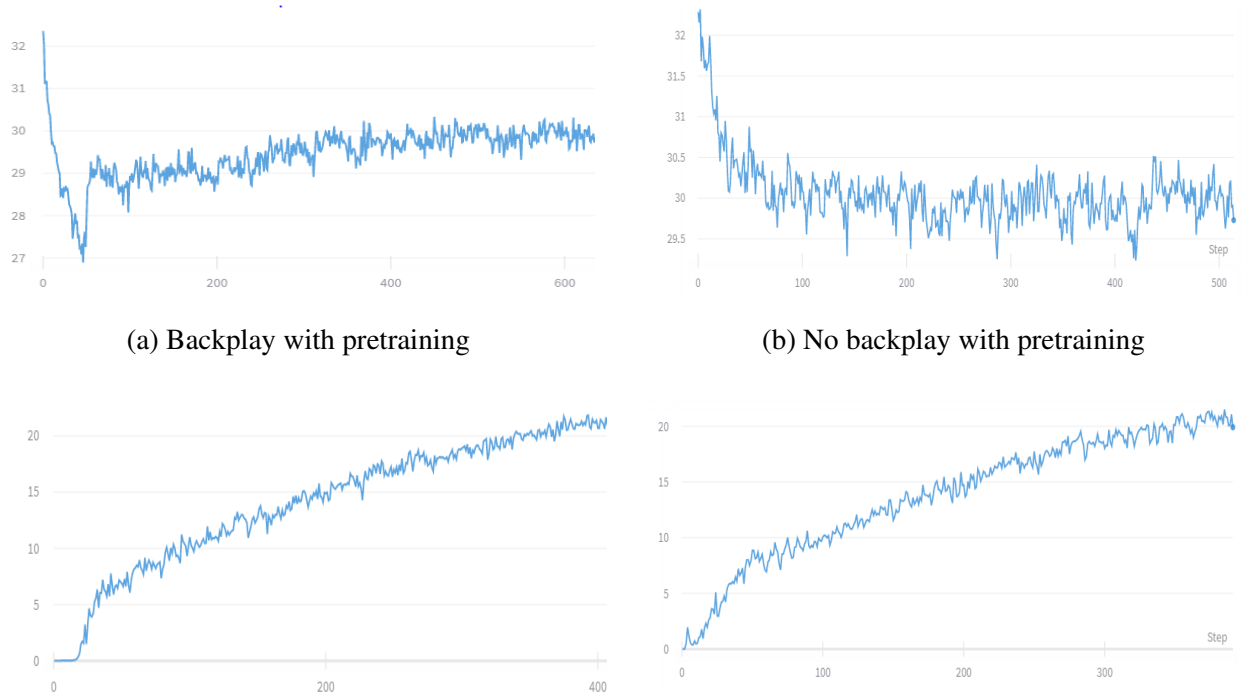


Figure 1: X axis - Training epochs; Y axis - Number of words missing

Even though the model could remember a lot of words, it was unable to overfit perfectly on the training data. After 2000 epochs, the model wasn't able to memorize the entire sentence for 100 training examples and a restricted action space. Due to this, we added log likelihood loss



(a) Backplay with pretraining

(b) No backplay with pretraining

(c) Backplay without pretraining

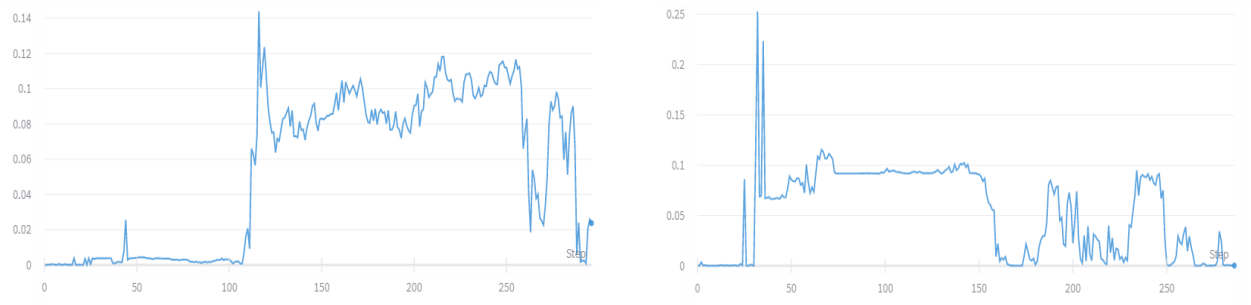
(d) No backplay without pretraining

Figure 2: Corpus bleu scores on validation data when log likelihood was used.

term to the training objective and used a model pretrained on the same data by regular supervised training. The reward used was sentence level bleu score at the end of a generation. Sentence level bleu score only make sense for sentences of length greater than 3, we started the backplay training where 4 words are missing from the target and moved backwards.

Previously we used a threshold on the reward to make a transition. Since reward scheme has been modified, we also changed the way backplay was done. A transition was made every n training epochs (usually 20-50), and the number of words missing would increase by m (usually 1-5) (both n and m were hyperparameters). The training objective was a linear combination of RL loss and Log likelihood loss. We observed that the performance of the models was similar irrespective of using backplay. A reason for this could be that log likelihood loss was doing most of the work during training. The plots of these experiments showing corpus bleu score on validation data is shown in Figure 2.

When log likelihood loss was not used and the rest of the setup was kept same, we observed that the model failed to generalize on the validation data and bleu score was always less than 1. Plots of these experiments is shown in Figure 3.



(a) Backplay without pretraining

(b) No backplay without pretraining

Figure 3: Corpus bleu scores on validation data without log likelihood loss.

Hyperparameter	Value	Description
discount factor γ	0.99	Discount factor for PPO
Learning rate	7e-4	Learning rate for adam optimizer
num processes	100	Number of training CPU processes to use
Num steps	150	Number of forward steps in A2C
ppo epochs	4	Number of ppo epochs
ppo batch size	512	Batch size for parameter update
Clip parameter	0.2	PPO clip parameter
value loss coeff	0.5	Value loss coefficient
Entropy coeff	0.01	Distribution entropy coefficient
Threshold	0.9	Reward threshold for transition
Tau τ	0.95	Gae parameter

Table 1: RL hyperparameters

8 Conclusion

Since this is a hard problem for the existing RL algorithms, it doesn't work well without supervison even when a method for dealing with sparse rewards was used. If Monolingual data and other forms of pretraining are used with Backplay, there is a possibility of successfully training an NMT system without supervised loss.

9 Acknowledgements

I would like to thank Kyunghyun Cho, Roberta Raileanu and Cinjon Resnick for their discussions and guidance along the way.

References

[Bahdanau et al., 2016] Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2016). An actor-critic algorithm for sequence prediction. *arXiv*

preprint arXiv:1607.07086.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[Bengio et al., 2015] Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

[Nguyen et al., 2017] Nguyen, K., Daumé III, H., and Boyd-Graber, J. (2017). Reinforcement learning for bandit neural machine translation with simulated human feedback. *arXiv preprint arXiv:1707.07402*.

[Resnick et al., 2018] Resnick, C., Raileanu, R., Kapoor, S., Peysakhovich, A., Cho, K., and Bruna, J. (2018). Backplay:" man muss immer umkehren". *arXiv preprint arXiv:1807.06919*.

[Ross et al., 2011] Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Shen et al., 2015] Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2015). Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

[Wu et al., 2018] Wu, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. (2018). A study of reinforcement learning for neural machine translation. *arXiv preprint arXiv:1808.08866*.