MINIMUM DIVERGENCE VS. MAXIMUM MARGIN: AN EMPIRICAL COMPARISON ON SEQ2SEQ MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequence to sequence (seq2seq) models have become a popular framework for neural sequence prediction. While traditional seq2seq models are trained by Maximum Likelihood Estimation (MLE), much recent work has made various attempts to optimize evaluation scores directly to solve the mismatch between training and evaluation, since model predictions are usually evaluated by a task specific evaluation metric like BLEU or ROUGE scores instead of perplexity. This paper for the first time puts this existing work into two categories, a) minimum divergence, and b) maximum margin. We introduce a new training criterion based on the analysis of existing work, and empirically compare models in the two categories. Our experimental results show that training criteria based on the idea of minimum divergence can usually work better than maximum margin methods, on both the tasks of machine translation and sentence summarization.

1 INTRODUCTION

Sequence to sequence (seq2seq) models (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) are a powerful end-to-end solution to sequence prediction since they allow a mapping between two sequences of different lengths. However, Maximum Likelihood Estimation (MLE) which is used as a standard training criterion in seq2seq has a series of drawbacks:

- Training and evaluation mismatch: during training, we maximize the log likelihood, while during inference, the model is evaluated by a different metric such as BLEU or ROUGE;
- MLE fails to assign proper scores to different model outputs, which means that all incorrect outputs are treated equally.

Similar to the SEARN (Daumé et al., 2009) and the DAGGER (Ross et al., 2011) in traditional models, many reinforcement learning (RL) techniques have been extended to seq2eq models recently to solve the mismatch between training and inference. The key idea is to replace MLE with a criterion that gives task-specific scores to different model outputs.

Recent approaches to taking the evaluation metric into training criteria generally fall into one of the following two categories:

1. *minimum divergence*: Minimum divergence methods minimize some divergence functions between the model output distribution and the probability distribution given by the ground truth, and by optimizing the divergence between the two distributions, the model output distribution will ideally be equal to the ground truth distribution.
2. *maximum margin*: During inference, in structured prediction, the goal is to simply ensure good structures have the highest predicted score. That is, margin based learning differs from *minimum divergence* learning in the aspect that it only requires that the predicted score of the correct output should be higher than that of the others by a margin.

The main contribution of this paper can be summarized as: based on the analysis of existing work, we introduce a novel training criterion for seq2seq learning and show its effectiveness.

Based on our experimental results on the tasks of machine translation and sentence summarization, we conclude that some models that take the evaluation metric into consideration can improve over a strong MLE baseline by a large margin, and *minimum divergence* methods usually get better results than *maximum margin* based approaches.

2 RELATED WORK

Seq2seq models were proposed and improved by Kalchbrenner & Blunsom (2013) and Sutskever et al. (2014). With the attention mechanism (Bahdanau et al., 2015; Luong et al., 2015), seq2seq models achieve better results than traditional models in many fields including machine translation (Bahdanau et al., 2015; Luong et al., 2015) and text summarization (Chopra et al., 2016).

Before seq2seq models, previous work on optimizing task specific evaluation metric scores genrally includes SEARN (Daumé et al., 2009), DAGGER (Ross et al., 2011), Minimum Error Rate Training (MERT) (Och, 2003), and softmax-margin criterion (Gimpel & Smith, 2010).

Seq2seq model uses MLE as the training objective, which also has the problem of training and evaluation mismatch. Ranzato et al. (2016) took evaluation metric into training of seq2seq models and proposed the mixed incremental cross entropy REINFORCE (MIXER) training strategy, which is similar to the idea of Minimum Risk Training (MRT) (Smith & Eisner, 2006; Li & Eisner, 2009; Ayana et al., 2016; Shen et al., 2016). MIXER used decoder hidden states to predict the bias term to reduce the variance, while MRT renormalized the predicted probabilities.

Inspired by MIXER, Bahdanau et al. (2017) extended the actor critic algorithm (Konda & Tsitsiklis, 2000; Sutton et al., 2000) to the seq2seq framework. Sokolov et al. (2016) and Kreutzer et al. (2017) proposed the pairwise preference learning framework, which enables the model to learn from the difference between a pair of samples generated by the model. Another popular model in the form of *divergence minimization* is the smoothed version of MERT algorithm proposed by Och (2003). The smoothed MERT algorithm also minimizes the expected loss over samples from model outputs and is similar to MRT essentially. Norouzi et al. (2016) proposed the Reward Augmented Maximum Likelihood (RAML) which also optimizes the divergence by data augmentation, and RAML was further combined with maximum likelihood via the α-divergence recently (Koyamada et al., 2017).

In contrast to the above *divergence minimization* based approaches, Wiseman & Rush (2016) used a margin based training objective and their search optimization process requires performing parameter updates or inference each time a new word is generated. One main drawback of *maximum margin* is that given a sample subset of an arbitrary size, only two samples of the subset can contribute to the gradient. The softmax-margin criterion proposed by Gimpel & Smith (2010) can allievate this problem, in which all samples are used in loss and gradient computation. However, Edunov et al. (2018) shows that the softmax-margin criterion still cannot reach the performance of risk minimization. As the latest work, Edunov et al. (2018) is very similar to ours. However, our interests are in finding better training criteria on a consistant setting, while they mainly focused on absolute performance improvement, using techniques like combination with token level loss, online vs. offline candidate generation, etc.

3 SEQUENCE TO SEQUENCE MODEL

This paper considers the seq2seq model from Bahdanau et al. (2015) and Luong et al. (2015), which consists of an RNN encoder and an RNN decoder. Due to the gradient vanishing problem, the vanilla RNN cell is replaced with a Gated Recurrent Unit (GRU) (Cho et al., 2014) or Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). A bidirectional RNN is used as encoder, with one RNN reading the source sequence from beginning to end, and the other RNN reading in reversed order. The bidirectional RNN encoder encodes the input sequence and gets a sequence of hidden states by concatenating the hidden states of each word in the two directions, which will be used as attention information in the decoder RNN.

The decoder RNN is initialized by passing the two final hidden states of the encoder to a feed forward network. The mixed information used as input in the first step $\boldsymbol{m}_0$ is initialized by a zero vector. In decoding step i, using the input feeding approach described in Luong et al. (2015), the decoder RNN reads the embedding of the word $\boldsymbol{w}_i$ and the mixed information $\boldsymbol{m}_{i-1}$, to generate the hidden

state of current step $\boldsymbol{h}_i$:

$$\boldsymbol{h}_i = f_1(concat(\boldsymbol{w}_i\,,\,\boldsymbol{m}_{i-1}),\,\boldsymbol{h}_{i-1}),$$

where f_1 is the RNN cell. The decoder then generates the attention context $\boldsymbol{c}_i$ from encoder using $\boldsymbol{h}_i$. Since there are many different attention mechanisms, we omit the details here. After getting $\boldsymbol{c}_i$, the model mixes $\boldsymbol{c}_i$ and $\boldsymbol{h}_i$ together by a feedforward network f_2 to generate $\boldsymbol{m}_i$, which is used as the mixed information in the next step:

$$\boldsymbol{m}_i = f_2(\boldsymbol{c}_i, \boldsymbol{h}_i)$$

Finally, $\boldsymbol{m}_i$ is passed to a feedforward network to generate $\boldsymbol{o}_i$, and the softmax layer outputs the probability for each word at step i. Suppose the target word in step i is v', the probability of the target word in step i is given by

$$p(\boldsymbol{y}_i = v' \,|\, \boldsymbol{x}, \boldsymbol{y}_{1:i-1}) = \frac{exp(\boldsymbol{o}_{iv'})}{\sum_v exp(\boldsymbol{o}_{iv})}.$$

For the sequence $\boldsymbol{y} = \langle \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_l \rangle$, the probability $p(\boldsymbol{y} \,|\, \boldsymbol{x})$ is

$$p(\boldsymbol{y} \,|\, \boldsymbol{x}) = p(\boldsymbol{y}_1 \,|\, \boldsymbol{x})\, p(\boldsymbol{y}_2 \,|\, \boldsymbol{x}, \boldsymbol{y}_1) ... p(\boldsymbol{y}_l \,|\, \boldsymbol{x}, \boldsymbol{y}_{1:l-1}).$$

Suppose $\boldsymbol{y}^*$ is the correct output, MLE actually maximizes the log likelihood of $p(\boldsymbol{y}^* \,|\, \boldsymbol{x})$:

$$\mathcal{L}_{\mathbf{MLE}} = -\log p(\boldsymbol{y}^* \,|\, \boldsymbol{x}) = -\sum_{i=1}^{l} \log p(\boldsymbol{y}_i^* \,|\, \boldsymbol{x}, \boldsymbol{y}_{<i}^*)$$

For a detail explanation, please refer to Luong et al. (2015). Now we describe the algorithm to construct a subset $\mathbf{S}$ from all possible output sequences which will be used in Section 4 & 5. In seq2seq model, the *n-best* list can only be approximated by performing beam search, as used in Wiseman & Rush (2016), or Monte Carlo (MC) sampling, as used in Ayana et al. (2016) and Shen et al. (2016). While using beam search may produce more accurate results, the time required for generating a candidate set of the same size as MC sampling using beam search is too long. In this paper, we construct the approximated *n-best* list S using MC sampling proposed in Shen et al. (2016) , and for completeness, we describe the sampling approach in Appendix 8.1 in details. Please note that the sampling algorithm does not guarantee the size of the sample subset, since it is inefficient to force the model to generate an exact number of distinct samples using GPU.

4 Minimum Divergence

In this section, we consider training criteria in the form of the following equation:

$$\mathcal{L} = D(\, p_1 \,||\, p_2 \,),$$

where p_1 and p_2 are two probability distributions given by the model or the ground truth, and D is a divergence function, for example, the Kullback Leibler (KL) divergence. The baseline training criterion MLE can also be seen as minimum divergence training, where the cross entropy between the model output distribution and the onehot encoded ground truth is minimized. Now we will discuss training criteria in which the ground truth distribution contains information of evaluation metric scores.

4.1 Reward Augmented Maximum Likelihood

Suppose that $\{\boldsymbol{x}, \boldsymbol{y}^*\}$ is a given input-output pair, the model learns parameters $\boldsymbol{\theta}$ and gives predition $p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$ on the output space $\mathbf{Y}$, the *exponentiated payoff distribution* (Norouzi et al., 2016) $q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$ is defined as follows:

$$q(\boldsymbol{y} \,|\, \boldsymbol{y}^*) = \frac{1}{Z(\boldsymbol{y}^*)}\ \exp[\, r(\boldsymbol{y}, \boldsymbol{y}^*)/\tau \,], \tag{1}$$

where $Z(\boldsymbol{y}^*) = \sum_{\boldsymbol{y} \in \mathbf{Y}} \exp[\, r(\boldsymbol{y}, \boldsymbol{y}^*)/\tau \,]$ is the normalization factor. τ is a hyperparameter that controls the smoothness of the distribution, and $r(\boldsymbol{y}, \boldsymbol{y}^*)$ is the score of prediction $\boldsymbol{y}$ given by the evaluation metric, e.g. BLEU (Papineni et al., 2002), or ROUGE (Lin, 2004)

The training objective of Reward augmented Maximum Likelihood (RAML) (Norouzi et al., 2016) is defined as

$$\mathcal{L}_{\mathbf{RAML}} = \mathbb{E}_{q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)}[\, \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})\,], \tag{2}$$

in which, the gradient is:

$$\mathbb{E}_{q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)}[\, \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})\,] = \sum_{\boldsymbol{y} \in \mathbf{Y}} q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)\ \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}) \tag{3}$$

Equation (3) requires computing $q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$ for all $\boldsymbol{y}$ and performing gradient desent on the whole $\mathbf{Y}$. However, due to the huge search space of $\mathbf{Y}$ and the high computational cost of $r(\boldsymbol{y}, \boldsymbol{y}^*)$, Equation (3) cannot be directly computed in an efficient way. Norouzi et al. (2016) proposed using stratified sampling. One can sample from negative edit distance or Hamming distance and then do importance reweighting to reweight the samples with task specific scores. After sampling from $q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$, the model is trained to maximize $\log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$.

Another choice is to construct a sample subset $\mathbf{S}$ from the model outputs using MC sampling, and use Equation (1) to compute the probability $q'(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$ in the sample subset. The training criterion is then changed to

$$\mathcal{L}'_{\mathbf{RAML}} = \sum_{\boldsymbol{y} \in \mathbf{S}} q'(\boldsymbol{y} \,|\, \boldsymbol{y}^*)\ \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}).$$

4.2 Minimum Risk Training

We first introduce the REINFORCE algorithm (Williams, 1992), which is a famous policy gradient method in RL, and then compare it with the Minimum Risk Training (MRT) (Smith & Eisner, 2006; Li & Eisner, 2009; Ayana et al., 2016; Shen et al., 2016). In REINFORCE, given a state $\boldsymbol{x}$, an action $\boldsymbol{y}$ under state $\boldsymbol{x}$, and model parameters $\boldsymbol{\theta}$, the policy gradient $\boldsymbol{g}$ is

$$\boldsymbol{g} = (r(\boldsymbol{y}) - \mathcal{B}(\boldsymbol{x}))\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}), \tag{4}$$

where $r(\boldsymbol{y})$ is the reward of action $\boldsymbol{y}$, and $\mathcal{B}(\boldsymbol{x})$ is the *baseline* or *control variate* that is used to reduce the variance (without introducing bias).

In MRT, the training objective is the expectation of evaluation metric scores taken with respect to the model output distribution:

$$\mathcal{L}_{\mathbf{MRT}} = -\mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})}[\, r(\boldsymbol{y}, \boldsymbol{y}^*)\,]. \tag{5}$$

Also due to the huge search space, Equation (5) can only be estimated by sampling from the model outputs. Suppose $\mathbf{S}$ is a sample subset from $\mathbf{Y}$, $p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$ is replaced by $p'(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}; \alpha)$:

$$p'(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}; \alpha) = \frac{p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})^{\alpha}}{\sum_{\boldsymbol{y} \in \mathbf{S}} p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})^{\alpha}}, \tag{6}$$

where α is a hyperparameter that controls the smoothness of $p'(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}; \alpha)$, we will later show that the α here is essentially the same as τ in RAML.

The gradient of $\mathcal{L}_{\mathbf{MRT}}$ can be derived as:

$$-\nabla_{\boldsymbol{\theta}}\, \mathbb{E}_{p'(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}, \alpha)}[\, r(\boldsymbol{y}, \boldsymbol{y}^*)\,] = -\alpha\, \mathbb{E}_{p'(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}, \alpha)}[\, \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})(r(\boldsymbol{y}, \boldsymbol{y}^*) - B)\,],$$

with

$$B = \mathbb{E}_{p'(\boldsymbol{y'} \,|\, \boldsymbol{x}; \boldsymbol{\theta}, \alpha)}[\, r(\boldsymbol{y'}, \boldsymbol{y}^*)\,]. \tag{7}$$

If we regard the whole seq2seq framework as a one-step Markov Decision Process (MDP), in which the input sequence $\boldsymbol{x}$ is the state, the output sequence $\boldsymbol{y}$ is the action, and the evaluation metric score r is the reward, both REINFORCE and MRT actually optimize the same training criterion. The difference between them is that REINFORCE requires only single sample and is unbiased, while MRT has multiple samples and the probability renormalization in Equation (6) introduces bias. The *expected loss* objective in Kreutzer et al. (2017) is also similar (and closer) to REINFORCE which uses one sample to estimate the gradient.

4.3 COMPARISON OF RAML & MRT

The KL divergence is defined as:

$$D_{\mathbf{KL}}(p_1(x)||p_2(x)) = \mathbb{E}_{p_1(x)}[\log p_1(x)] - \mathbb{E}_{p_1(x)}[\log p_2(x)]. \tag{8}$$

Starting from RAML, in terms of KL divergence, Equation (2) can be rewritten as:

$$\mathcal{L}_{\mathbf{RAML}} = D_{\mathbf{KL}}(q(\boldsymbol{y} \,|\, \boldsymbol{y}^*) \,||\, p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})) - \sum_{\boldsymbol{y}} q(\boldsymbol{y} \,|\, \boldsymbol{y}^*) \log q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$$

The gradient of the second term with respect to model parameters $\boldsymbol{\theta}$ is zero, thus the second term can be regarded as a constant. The training criterion of RAML is essentially minimizing the KL divergence from $q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$ to $p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$.

Meanwhile, the training criterion of MRT in Equation (5) can be rewritten as:

$$\begin{aligned}
\mathcal{L}_{\mathbf{MRT}} &= -\tau\, \mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})}[\, r(\boldsymbol{y}, \boldsymbol{y}^*)/\tau \,] \\
&= -\tau\, \mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})}[\, \log q(\boldsymbol{y} \,|\, \boldsymbol{y}^*) \,] - \tau\, \mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})}[\, \log Z(\boldsymbol{y}^*) \,] \\
&= -\tau\, \mathbb{E}_{p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})}[\, \log q(\boldsymbol{y} \,|\, \boldsymbol{y}^*) \,] - \tau \log Z(\boldsymbol{y}^*),
\end{aligned}$$

in which, the second term is again a constant as it has no gradient with respect to $\boldsymbol{\theta}$. Thus we have:

$$\mathcal{L}_{\mathbf{MRT}} = \tau\, D_{\mathbf{KL}}(p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})||q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)) - \tau \sum_{\boldsymbol{y}} p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}) \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}) + Const, \tag{9}$$

in which, the second term is the negative Shannon Entropy of $p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$. Increasing the entropy can prevent the model from generating too deterministic results, which is called the maximum entropy regularizer (Williams & Peng, 1991). Equation (9) actually shows that MRT minimizes the KL divergence from $p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$ to $q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$.

By comparing the above two methods, we find that both RAML and MRT are minimizing the KL divergence between the model output distribution and the *exponentiated payoff distribution*, but with different directions of $D_{\mathbf{KL}}$. Now we will discuss the effect of hyperparameter τ in RAML and α in MRT.

KL divergence has a property that it is minimized if and only if the two probability distributions are equal. Ideally, when the KL divergence in RAML is minimized, we have:

$$\begin{aligned}
p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}) &= q(\boldsymbol{y} \,|\, \boldsymbol{y}^*) \\
\log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}) &= r(\boldsymbol{y}, \boldsymbol{y}^*)/\tau + Const \\
r(\boldsymbol{y}, \boldsymbol{y}^*) &= \tau\, \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}) + Const.
\end{aligned}$$

And for MRT, we similarly have:

$$r(\boldsymbol{y}, \boldsymbol{y}^*) = \alpha\, \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}) + Const. \tag{10}$$

It turns out that the hyperparameter τ in RAML and α in MRT have the same effect.

4.4 HELLINGER DISTANCE MINIMIZATION

Now we introduce another popular distance metric, the Hellinger distance (Hellinger, 1909), which also belongs to the family of Csiszár f-divergence (Csiszár, 1963) and has been used in graphical models (Beykikhoshk et al., 2015) and clustering algorithms (Abdel-Azim, 2016; Ji et al., 2018). Given two probability distributions p_1 and p_2, let f and g denote the probability density functions of p_1 and p_2. The square of the Hellinger distance $H(p_1, p_2)$ is defined as:

$$H^2(p_1, p_2) = \frac{1}{2} \int (\sqrt{f(x)} - \sqrt{g(x)})^2\, dx.$$

The training criterion using squared Hellinger distance is:

$$\mathcal{L}_{\mathbf{Hellinger}} = \sum_{\boldsymbol{y} \in \mathbf{S}} (\, \sqrt{p'(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})} - \sqrt{q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)}\,)^2,$$

with $p'(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta})$ defined in Equation (6) and $q(\boldsymbol{y} \,|\, \boldsymbol{y}^*)$ defined in Equation (1).

5 MAXIMUM MARGIN

Let Δ denote the loss given by the evaluation metric:

$$\Delta = r(\boldsymbol{y}^*, \boldsymbol{y}^*) - r(\boldsymbol{y}, \boldsymbol{y}^*),$$

and let F be the predicted score of the model:

$$F = \tau \ \log p(\boldsymbol{y} \,|\, \boldsymbol{x}; \boldsymbol{\theta}).$$

Different from minimizing the expected risk, another choice of training criterion is to optimize the evaluation score of the sequence with the highest predicted score:

$$\arg\min_{\boldsymbol{\theta}} \ \Delta(\arg\max_{\boldsymbol{y}} \ F(\boldsymbol{y}, \boldsymbol{\theta}), \boldsymbol{y}^*). \tag{11}$$

For each input sequence $\boldsymbol{x}$, we first do inference to find the output $\boldsymbol{y}$ that has the highest predicted score, then we train the model to maximize the score of $\boldsymbol{y}$ given by evaluation metric.

Equation (11) includes an $\arg\max$ and the dependency of r on $\boldsymbol{y}$ is complex, and also, the gradient of such complex function is usually not defined, thus optimizing it directly on a deep neural network is not computationally tractable. This problem has been widely discussed in the training process of Structural Support Vector Machine (SSVM) (Tsochantaridis et al., 2004; Joachims, 2006; Joachims et al., 2009). One solution is to find some surrogates that are relatively easier to optimize. Tsochantaridis et al. (2004) proposed using margin and slack rescaling instead.

Margin Rescaling is based on the following inequality:

$$\Delta(\boldsymbol{y}, \boldsymbol{y}^*) \leq \max_{\boldsymbol{y}}(\Delta(\boldsymbol{y}, \boldsymbol{y}^*) + F(\boldsymbol{y})) - F(\boldsymbol{y}^*) \tag{12}$$

Similarly, Slack Rescaling is:

$$\Delta(\boldsymbol{y}, \boldsymbol{y}^*) \leq \Delta(\boldsymbol{y}, \boldsymbol{y}^*) \ \max_{\boldsymbol{y}}(1 + F(\boldsymbol{y}) - F(\boldsymbol{y}^*)) \tag{13}$$

As Equation (13) is a tighter bound on $\Delta(\boldsymbol{y}, \boldsymbol{y}^*)$, slack rescaling generally gives better results than margin rescaling. Both Equation (12) and Equation (13) can be optimized by subgradient descent. Let $\widehat{\boldsymbol{y}}$ denote the sample that gives the $\max$ value. The subgradient for margin rescaling is:

$$\nabla_{\boldsymbol{\theta}} \tau \log p(\widehat{\boldsymbol{y}}) - \nabla_{\boldsymbol{\theta}} \tau \log(p(\boldsymbol{y}^*)),$$

and similarly, the subgradient for slack rescaling is:

$$\Delta(\widehat{\boldsymbol{y}}, \boldsymbol{y}^*)(\nabla_{\boldsymbol{\theta}} \tau \log p(\widehat{\boldsymbol{y}}) - \nabla_{\boldsymbol{\theta}} \tau \log p(\boldsymbol{y}^*)).$$

Since doing *loss augmented inference* on such a huge search space is again not realistic, approximate inference is still needed. Equation (12) also has the name of structured hinge loss when used in SSVM. One may have found that Equation (13) is the same as the training criterion in Wiseman & Rush (2016). Besides, to make a fair comparison with other training criteria and make it applicable on large datasets, we replace the beam search with sample subset.

6 EXPERIMENTS

We evaluate the training criteria on two tasks in total. The first task is machine translation on a small dataset to explore the hyperparameter impact. The second task is sentence summarization using large datasets.

For all the experiments, we use LSTM as RNN cell. The attention mechanism used in our experiments is global attention with input feeding (Luong et al., 2015). All the predicted sequences for evaluation are generated using greedy search, since greedy search can be easily batched and is much faster than beam search.

6.1 MACHINE TRANSLATION

Due to space limit, experimental results on hyperparameter effects and training details are provided in Appendix 8.2.1.

Data We use the IWSLT 2014 German-English translation dataset, with the same splits as Ranzato et al. (2016) and Wiseman & Rush (2016), which contains about 153K training sentence pairs, 7K validation sentence pairs and 7K test sentence pairs. Sentences in the dataset are first tokenized and then converted into lowercase. During training, the maximum length for both inputs and outputs is restricted to 50. Words that appear less than three times are replaced with a UNK token.

Models The encoder is a single-layer bidirectional LSTM with 256 hidden units for either direction, and the decoder LSTM also has 256 hidden units. The size of word embedding for both encoder and decoder is 256. We use a dropout rate (Srivastava et al., 2014) of 0.2 to avoid overfitting. The gradient is clipped when its norm exceeds 1.0 to prevent gradient explosion and stabilize learning (Pascanu et al., 2013). The batch size is set to 32 and the training set is shuffled at each new epoch. All models are trained with the Adam optimizer (Kingma & Ba, 2015). The sentence level BLEU score used as reward or loss during training is smoothed by adding 1 to both numerator and denominator, as suggested by Chen & Cherry (2014).

Results For the two max margin methods, τ is set to 1.0×10^{-3}. We use the sample subset version of RAML, and τ for RAML is also set to 1.0×10^{-3}. For Hellinger loss, $\alpha = 5.0 \times 10^{-4}$ and $\tau = 0.5$. The final results on the test set are described in Table 1. Max margin methods here improve the MLE baseline less significant than Wiseman & Rush (2016) did, since the search optimization algorithm is not used in our experiments. The training criterion of squared Hellinger distance gives the highest BLEU score on the test set.

Table 1: BLEU scores on IWSLT German-English translation evaluated on the test set.

Criterion	BLEU	Improvement
MLE	26.03	0.00
margin	26.22	+0.19
slack	26.32	+0.29
RAML	26.34	+0.31
MRT	27.35	+1.32
Hellinger	**27.75**	**+1.72**

6.2 SENTENCE SUMMARIZATION

The task of sentence summarization is: given a long sentence or a passage, the model will generate a short sentence that summarizes the long text. For evaluation, following Rush et al. (2015), we use the full length F1 variant of ROUGE score (Lin, 2004) as metric. Training details are described in Appendix 8.2.2.

Data We use the Gigaword corpus with the same preprocessing steps as in Rush et al. (2015). In the Gigaword dataset, the first sentence of each article is regarded as the source sequence, and the title of the article as the target sequence. The training set consists of 3.7M sequence pairs, and the vocabulary size for both article and title is 50K, with the out-of-vocabulary words replaced by a unified UNK token. The whole dataset is then converted into lowercase without tokenization. During training, we use the first 2K sequences of the dev corpus as validation set, and the size of the test set is also 2K. The maximum length for both input and output sequence is restricted to 50.

Models The encoder is a single layer bidirectional LSTM with 1000 hidden units for either direction. The single layer decoder LSTM also has 1000 dimmensions. The word embedding size for both encoder and decoder is 500. Dropout is not used in the sentence summarization task, and the gradient is clipped when its norm exceeds 1.0. The batch size is set to 128. The training set is shuffled at each new epoch, and we prefetch 20 batches and sort them by the length of target sequence to form new batches to speed up training. We use ROUGE-2 as the reward or loss function.

Results Table 2 shows the results on the validation set and the test set. We report ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-S4 score. MRT outperforms the MLE baseline by a large margin, but is still slightly lower than the newly introduced Hellinger loss.

Table 2: Results on Gigaword sentence summarization task

Criterion	ROUGE-1		ROUGE-2		ROUGE-L		ROUGE-S4	
	valid	test	valid	test	valid	test	valid	test
MLE	48.41	34.20	24.14	15.22	45.08	31.84	22.62	14.00
margin	48.43	34.20	24.16	15.24	45.09	31.85	22.65	14.00
slack	48.40	34.20	24.12	15.25	45.08	31.83	22.63	14.00
RAML	48.43	34.20	24.16	15.24	45.10	31.84	22.66	14.00
MRT	50.41	35.63	26.42	16.96	47.05	33.24	24.47	15.31
Hellinger	**51.02**	**35.84**	**26.88**	**17.03**	**47.71**	**33.47**	**25.12**	**15.54**

7 DISCUSSION

For training criteria in the form of *divergence minimization*, according to our experimental results, MRT outperforms RAML. As mentioned before, MRT and RAML minimize the KL divergence in two different directions. By the definition of KL divergence in Equation (8), it is only minimized when $p_1 = p_2$. Optimizing the KL divergence in either of the two directions both guarantee consistency, i.e., the training procedure can ultimately recover the true probability distribution. For more explanation on the properties of KL divergence, we refer the readers to Huszár (2015).

For a model that minimizes $D_{\mathbf{KL}}(p_1||p_2)$, the model will sample from p_1 and maximize $\log p_2$, which means that if x has a high probability in p_1, it will also have a high probability in p_2, but if has a low probability in p_1, $p_2(x)$ may still be high.

Thus in RAML which samples from *exponentiated payoff distribution* q and maximizes $\log p$, sequences that have a high evaluation score will also have a high predicted probability. However, samples that have a low q, e.g., a low BLEU or ROUGE, may still have a high p.

In contrast, MRT will first sample from model outputs and ensure that a high BLEU or ROUGE score will be assigned to sequences that have a high probability in model output distribution p, at the cost that sequences that have low probabilities in p may still have high evaluation scores. This can explain the difference between RAML and MRT.

The squared Hellinger distance can be regarded as the squared error between $p^{\frac{1}{2}}$ and $q^{\frac{1}{2}}$, which is essentially a regression loss, while the KL divergence usually works better in classification problems. By optimizing the KL divergence, the model is trained to find the candidate that has the highest probability and pays less attention to samples that have low probabilities, while by optimizing the squared Hellinger distance, the model learns to predict the evaluation metric score for every sample in the candidate set and pays equal attention to all of them. Since seq2seq models generate a predicted sequence word by word, the target sequence which has the highest evaluation metric score (sample from q and maximize $\log p$) or the highest predicted probability (sample from p and maximize $\log q$) may not be reached by approximate inference algorithms like greedy search or beam search. In this case, a model trained by optimizing the KL divergence may not work as expected, however, a model trained by optimizing the squared Hellinger distance can still make predictions normally since it pays equal attention to samples with low probabilities during training.

For *maximum margin* methods, slack rescaling is a tighter bound and the experimental result is slightly higher than margin rescaling on our small scale experiments. However, slack rescaling still cannot achieve the same level of performance of MRT and Hellinger distance. The reason may be that the upper bound is still not tight enough. In the training process of *maximum margin* methods, only the target sequence and a negative sample selected from the candidate set are used to update model parameters, which may also explain their performance.

REFERENCES

Gamil Abdel-Azim. New hierarchical clustering algorithm for protein sequences based on hellinger distance. *Appl Math*, 10(4):1541–9, 2016.

Ayana, Shiqi Shen, Zhiyuan Liu, and Maosong Sun. Neural headline generation with minimum risk training. *arXiv:1604.01904*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *ICLR*, 2017.

Adham Beykikhoshk, Ognjen Arandjelovic, Dinh Phung, and Svetha Venkatesh. Discovering topic structures of a temporally evolving document corpus. *arXiv preprint arXiv:1512.08008*, 2015.

Boxing Chen and Colin Cherry. A systematic comparison of smoothing techniques for sentence-level bleu. In *ACL*, 2014.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, pp. 103, 2014.

Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*, 2016.

Imre Csiszár. Eine informationstheoretische ungleichung und ihre anwendung auf den beweis der ergodizitat von markoffschen ketten. *Magyar. Tud. Akad. Mat. Kutató Int. Közl*, 8:85–108, 1963.

Hal Daumé, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 2009.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, et al. Classical structured prediction losses for sequence to sequence learning. In *NAACL-HLT*, 2018.

Kevin Gimpel and Noah A Smith. Softmax-margin training for structured log-linear models. 2010.

Ernst Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 136:210–271, 1909.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

F. Huszár. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *ArXiv e-prints*, 2015.

Wenying Ji, Simaan M AbouRizk, Osmar R Zaïane, and Yitong Li. Complexity analysis approach for prefabricated construction products using uncertain data clustering. *Journal of Construction Engineering and Management*, 144(8):04018063, 2018.

Thorsten Joachims. Training linear svms in linear time. In *KDD*, 2006.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, 2013.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *NIPS*, 2000.

Sotetsu Koyamada, Yuta Kikuchi, Atsunori Kanemura, Shin-ichi Maeda, and Shin Ishii. Neural sequence model training via α-divergence minimization. In *ICML Workshop*, 2017.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. Bandit structured prediction for neural sequence-to-sequence learning. In *ACL*, 2017.

Zhifei Li and Jason Eisner. First-and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*, 2009.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop*, 2004.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. Reward augmented maximum likelihood for neural structured prediction. In *NIPS*, 2016.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *ACL*, 2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *ICLR*, 2016.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *ACL*, 2016.

David A Smith and Jason Eisner. Minimum risk annealing for training log-linear models. In *ACL*, 2006.

Artem Sokolov, Julia Kreutzer, Stefan Riezler, and Christopher Lo. Stochastic structured prediction under bandit feedback. In *NIPS*, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 2000.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 1991.

Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, 2016.

8 APPENDIX

8.1 THE SAMPLING ALGORITHM

For completeness, we describe the MC sampling algorithm proposed by Shen et al. (2016) and Ayana et al. (2016) here.

Algorithm 1 The sampling approach to constructing the approximated *n-best* list

Input:
source sequence $\boldsymbol{x}$, model parameters $\boldsymbol{\theta}$, target sequence $\boldsymbol{y}^*$
sample size m, maximum length l
Output:
A set of samples $\mathbf{S}$ and corresponding probability $\mathbf{P}$

```
S = {y*}; P = {p(y*)};
for i in 1, ..., m − 1 do
    sample a sequence word by word given x and θ, stop sampling when length is l or meeting
    ⟨end-of-sequence⟩ symbol, and finally store the result and probability in y and q respectively
    if y in S then
        continue
    end if
    S = S ∪ {y}
    P = P ∪ {q}
end for
return S, P
```

8.2 EXPERIMENTS

8.2.1 MACHINE TRANSLATION

The MLE baseline is trained with a learning rate of 3.0×10^{-4}. The model is trained for 20 epochs in total, and is evaluated every 1000 steps on the validation set.

τ for RAML For RAML, since we use BLEU instead of Hamming distance as evaluation metric, we first show the BLEU scores on the validation set for hyperparameter τ in Table 3. RAML-I refers to importance reweighting and RAML-S refers to sample subset RAML, as mentioned in Section 4.1. Experimental results show that the BLEU score is not very sensitive to hyperparameter τ although τ varies from 1.0×10^{-7} to 1.0×10^{-1}, and setting τ larger will decrease the BLEU score compared with the MLE baseline. We also find that RAML-S tends to give slightly better results than RAML-I. The learning rate for RAML is 1.0×10^{-6}.

τ	RAML-I	RAML-S
1.0×10^{-7}	29.25	29.28
1.0×10^{-5}	29.27	29.30
1.0×10^{-3}	29.26	29.32
1.0×10^{-1}	29.28	29.32

Table 3: Tokenized BLEU scores on IWSLT 2014 German-English translation evaluated on validation set using RAML criterion.

α for MRT For MRT, since Shen et al. (2016) did several experiments to study the effect of hyperparameters, the original hyperparameters are strictly followed in our experiments. Hyperparameter α is set to 5.0×10^{-3} and the sample size is 100. The best BLEU score of MRT on the validation set is 30.55. The learning rate is set to 1.0×10^{-5}.

α and τ for Hellinger loss As shown in Table 4, $\alpha = 5.0 \times 10^{-4}$, $\tau = 0.5$ produces the best result. The highest BLEU score on the validation set is 31.13, which is 0.58 higher than MRT. The learning rate used in Hellinger loss is also 1.0×10^{-5}.

α \ τ	0.1	0.5	1.0
5.0×10^{-3}	29.33	29.71	29.22
5.0×10^{-4}	29.91	**31.13**	31.01
5.0×10^{-5}	30.75	30.83	31.00

Table 4: Tokenized BLEU scores on IWSLT 2014 German-English translation evaluated on the validation set using squared Hellinger distance with different α and τ as training criterion.

Sample size for Hellinger loss For squared Hellinger loss, the effect of sample size is shown in Table 5. A larger sample size generally gives a higher BLEU score, however, due to the limit of GPU memory, we cannot set sample size to a larger value.

sample size	20	50	100
BLEU	29.51	30.30	**31.13**

Table 5: Tokenized BLEU scores on IWSLT 2014 German-English translation evaluated on the validation set using squared Hellinger Distance with different sample sizes.

τ **for max margin** For the two max margin training criteria, the learning rate is 1.0×10^{-7}. Table 6 shows the effect of τ: according to the results on the validation set, τ is not very sensitive to the BLEU scores. As shown in Table 6, BLEU scores given by slack rescaling are usually 0.1 higher than for margin rescaling, and we use $\tau = 1.0 \times 10^{-3}$ for both of the two models.

τ	Slack	Margin
1.0×10^{-1}	29.16	29.23
1.0×10^{-3}	29.33	29.25
1.0×10^{-5}	29.35	29.16

Table 6: Tokenized BLEU scores on IWSLT 2014 German-English translation evaluated on the validation set using max margin criteria.

8.2.2 SENTENCE SUMMARIZATION

Details We use ROUGE-2 as reward or loss function. For the MLE baseline, we train the model for 7 epochs in total. For the first 5 epochs, we use Adam with a learning rate of 3.0×10^{-4}, we then use SGD with a learning rate of 0.1, and decay the learning rate by a factor of 0.5 at the beginning of epoch 7.

For the other training criteria, we use the hyperparameters with the best performance described in Section 6.1.