Sequence Generation with Optimal-Transport-Enhanced Reinforcement Learning

Liqun Chen, Ke Bai, Chenyang Tao, Yizhe Zhang, Guoyin Wang, Wenlin Wang, Ricardo Henao, Lawrence Carin
Duke University

Abstract

Reinforcement learning (RL) has been widely used to aid training in language generation. This is achieved by enhancing standard maximum likelihood objectives with user-specified reward functions that encourage global semantic consistency. We propose a principled approach to address the difficulties associated with RL-based solutions, namely, high-variance gradients, uninformative rewards and brittle training. By leveraging the optimal transport distance, we introduce a regularizer that significantly alleviates the above issues. Our formulation emphasizes the preservation of semantic features, enabling end-to-end training instead of *ad-hoc* fine-tuning, and when combined with RL, it controls the exploration space for more efficient model updates. To validate the effectiveness of the proposed solution, we perform a comprehensive evaluation covering a wide variety of NLP tasks: machine translation, abstractive text summarization and image caption, with consistent improvements over competing solutions.

Introduction

Sequence generation is one of the central research topics in natural language processing (NLP) studies, covering a broad spectrum of applications including machine translation (Sutskever, Vinyals, and Le, 2014; Cho et al., 2014; Bahdanau, Cho, and Bengio, 2015), abstractive summarization (Rush, Chopra, and Weston, 2015; Chopra, Auli, and Rush, 2016), image captioning (Vinyals et al., 2015; Xu et al., 2015), and style transfer (Shen et al., 2017; Prabhumoye et al., 2018).

Maximum likelihood estimation (MLE) is employed widely for training a sequence generation model. Specifically, given the sequential nature of language, MLE training follows an autoregressive design that maximizes the expected log-likelihood of a word conditioned on its previous context, *i.e.*, the information encoded by the already *observed* input sequence Cho et al. (2014). This practice leads to the well-known *exposure bias* issue, where the lack of exposure of the model to generated sequences during the training phase leads to the fast degeneration of semantic consistency for long sequences at test time (Bengio et al., 2015). This has motivated development of regularization schemes that involve the model-generated sequence at training time, resulting in marked performance gains (Ranzato et al., 2016; Norouzi et al., 2016; Rennie et al., 2017).

Among various empirical attempts to regularize MLE, the use of reinforcement learning (RL) techniques has been investigated extensively (Ranzato et al., 2016; Norouzi et al., 2016; Rennie et al., 2017; Paulus, Xiong, and Socher, 2018; Wu et al., 2016). RL leverages a set of principled learning algorithms, such as REINFORCE Williams (1992) and TRPO Schulman et al. (2015), to optimize the model over desirable but often non-differentiable performance metrics. In RL-based approaches these metrics are termed *reward functions*. Relative to the likelihood, reward functions allow the model to be informed by specilized performance metrics that can be understood as regularizers to the standard MLE objective Tan et al. (2018).

Crucial to the success of RL-regularized training is the choice of the reward function. Current practice largely falls into two categories: i) *static rewards*, such as phrase-matching-based metrics like BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and CIDEr Vedantam, Lawrence Zitnick, and Parikh (2015), routinely used in the evaluation of language models; and more recently, ii) *dynamic rewards* that track changes of the model in feature space during training, and are typically implemented via adversarial learning (Yu et al., 2017; Chen et al., 2018).

Despite encouraging results Rennie et al. (2017); Paulus, Xiong, and Socher (2018), limitations of both types of rewards have been widely recognized. Static rewards usually serve as imperfect proxies for human evaluation, resulting in undesirable biases (Wang et al., 2018). While being more objective, adversarial supervision in dynamic rewards relies on the delicate balance of a mini-max objective, which can be easily undermined in practice Arjovsky and Bottou (2017); Zhang et al. (2017); Chen et al. (2018). Both types of rewards can incur destabilization when training, manifested as mode-dropping and gradient-vanishing. Consequently, careful fine tuning is often required in the training of RL-regularized NLP models.

Optimal transport (OT) has emerged recently as a powerful learning framework, demonstrating success on some of the frontier challenges in machine learning, such as natural image generation Salimans et al. (2018) and graph learning Xu et al. (2019). OT-based learning generally reformu-

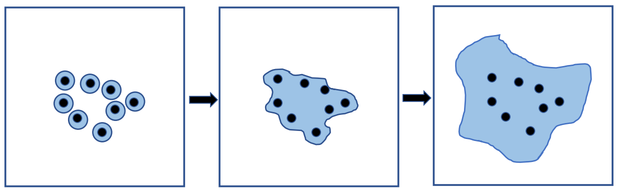

Figure 1: Effective exploration space for different training objectives. Black dots represent reference words, blue areas represent the effective exploration space, and the white background is the potential exploration space. Left: Exploration space in MLE, that only searches for words similar to the references. Middle: Optimal transport distance that tries to match words dynamically regardless of positioning in the sentence, so the exploration space is a set containing all reference words. Right: RL further pushes the exploration boundary via more flexible objectives.

lates problems into a cost-dependent (discrete) distribution-matching problem, which can be solved efficiently by iterative schemes like the Sinkhorn algorithm Cuturi (2013) or adversarial updates Arjovsky, Chintala, and Bottou (2017b). Applications of OT approaches in the context of NLP were pioneered by the work of Kusner et al. (2015) in document classification, and more recently Chen et al. (2019) demonstrated its effectiveness and robustness for sequence generation tasks through dynamic 1-gram matching.

Motivated by their success, this work presents a new training strategy that fuses the best aspects of RL- and OT-based learning for sequence modeling. Empirical evidence suggests that RL-learning poses negative influence early in training, a direct consequence of the uninformative and sparse reward it receives, resulting in a blind exploration performed in an enormous (discrete) policy space Tan et al. (2018). The gains of RL only manifest after a long burn-in period, thus yielding steady and gradual improvements during the fine-tuning stage. Alternatively, OT-regularization leads to swift convergence, *i.e.*, model performance increases quickly and plateaus after a short burn-in period Chen et al. (2018). Such complementary learning dynamics motivates us to take full advantage of both approaches via a carefully designed scheduled learning plan, that systematically explores the policy space in a cost-effective fashion (see Figure 1).

The proposed optimal-transport-enhanced RL (OTRL) framework features the following contributions: i) Analysis of the different regimes of RL and OT learning, motivating an annealed-schedule learning strategy that integrates RL and OT regularizations. ii) Demonstration of how the additional OT loss serves to adaptively regularize the RL exploration in policy space, leading to stabilized training dynamics. iii) Extensive experiments demonstrating the superiority of the proposed OTRL framework compared to strong baselines that leverage RL and OT separately.

RL for Sequence Generation

Reinforcement learning (RL) can be used as a means for evaluating and optimizing model parameters over flexible performance metrics in NLP tasks. Examples of these metrics are BLEU, ROUGE and CIDEr. Any dynamic (autoregressive) generative model for sequence generation can be viewed as an *agent* that interacts with an environment, *i.e.*, word (token) sequences. Well-known examples of these generative models are the long-short term memory (LSTM) cell Hochreiter and Schmidhuber (1997) and the Transformer Vaswani et al. (2017).

Predictions from the agent that result in the (unobserved) next word of the sequence constitute the *action* of an executed *policy* p_θ, where θ denotes the parameters of the agent, *i.e.*, the generative model, $\mathcal{M}(\cdot, \theta)$. During training, once the agent has reached the end of the sequence, a *reward* is issued. The reward function $r(\cdot)$, *e.g.*, an evaluation metric such as BLEU Papineni et al. (2002), is calculated by comparing the sequence generated by the model with a reference, *i.e.*, the ground-truth sequence. The loss function in RL-based training can be written as

$$\mathcal{L}_{\mathrm{RL}}(\theta) = -\mathbb{E}_{\boldsymbol{y}\sim p_\theta}[\boldsymbol{r}(\boldsymbol{y}, \boldsymbol{y}^*)] \,, \qquad (1)$$

where $\boldsymbol{y}$ is the sequence sampled from p_θ, $\boldsymbol{y}^*$ is the reference, and the expectation is wrt policy p_θ.

In scenarios where either sequence sampling or the reward function are not differentiable, we can use the REINFORCE algorithm (Williams, 1992) to approximate the gradient of (1). Specifically, we can approximate the gradient of $\mathcal{L}_{\mathrm{RL}}(\theta)$ as:

$$\nabla_\theta \mathcal{L}_{\mathrm{RL}}(\theta) = -\mathbb{E}_{\boldsymbol{y}\sim p_\theta}[\boldsymbol{r}(\boldsymbol{y}, \boldsymbol{y}^*) \nabla_\theta \log p_\theta(\boldsymbol{y})], \qquad (2)$$

where the expectation is approximated by Monte Carlo sampling from p_θ, *i.e.*, the probability of each generated word, resulting in actions manifested as each word of sequence $\boldsymbol{y}$.

To decrease the variance of the gradient of (1), a baseline (function) to the reward can be added to (2), while preserving the unbiasedness of the gradient estimation Paisley, Blei, and Jordan (2012). This baseline may be an arbitrary function depending on the history of $\boldsymbol{y}$ not its future, *i.e.*, a parameterized value-function whose purpose is to estimate the unbiased future reward Ranzato et al. (2016). The *self-critic* sequence training algorithm Rennie et al. (2017), considers the baseline function $b(\cdot, \cdot)$ as the reward collected at test time, which implies $b \triangleq r(\boldsymbol{y}, \hat{\boldsymbol{y}})$, where $\hat{\boldsymbol{y}}$ is the greedily decoded sequence whose elements are obtained from the generator at step t via $\hat{\boldsymbol{y}}_t = \arg\max_{\boldsymbol{y}_t} p_\theta(\boldsymbol{y}_t | \hat{\boldsymbol{y}}_{<t})$. Note the baseline is obtained at the end of the generating process and is not dependent of $\boldsymbol{y}^*$ (the ground-truth sequence). Aided by the baseline, we can re-formulate (2) as

$$\nabla_\theta \mathcal{L}(\theta) = -\mathbb{E}_{\boldsymbol{y}\sim p_\theta}[(\boldsymbol{r}(\boldsymbol{y}, \boldsymbol{y}^*) - b(\boldsymbol{y})) \nabla_\theta \log p_\theta(\boldsymbol{y})] \,, \qquad (3)$$

where $r(\cdot, \cdot)$ and $b(\cdot, \cdot)$ compare the generated sequence with the reference and greedily decoded sequence, respectively. As a result, we seek to maximize the reward and at the same time minimize the difference between the most likely, greedily decoded sequence, and samples from p_θ. Intuitively, if the evaluation metric value of the sampled sentence $\boldsymbol{y}$ is larger than that of the greedily decoded sentence $\hat{\boldsymbol{y}}$, minimizing the loss is equivalent to maximizing the probability of the greedily chosen words Rennie et al. (2017).

OT for Semantic Matching

Optimal transport (OT) can provide a dynamic (independent of position) 1-gram matching scheme for semantic matching Chen et al. (2019). In a nutshell, it computes the moving distance between two distributions, under an optimal transport plan, which can be learned. Below we consider leveraging OT as a means of expanding the exploration space for sequence generation tasks, thus improving the performance of generative models relative to their MLE counterparts.

In an optimal transport approach for sequence generation, we need to evaluate distances between words, that here we perform in word embedding space, to learn both the parameters of the model θ and the embedding matrix $\mathbf{E}$. In particular, we use the word embeddings predicted by the model, given by $\tilde{\boldsymbol{z}}_t = \mathbf{E}^T \tilde{\boldsymbol{w}}_t$, where $\mathbf{E} \in \mathbb{R}^{V \times d}$ is the word embedding matrix, V is the vocabulary size, d is the dimensionality of the embedding vector, and $\tilde{\boldsymbol{w}}_t$ is the word generated by the model by sampling from a Gumbel-softmax design (Jang, Gu, and Poole, 2016; Maddison, Mnih, and Teh, 2017). Under Gumbel-softmax, outputs from the sequence generation model $\boldsymbol{v}_t$ produce $\tilde{\boldsymbol{w}}_t = \text{softmax}(\frac{\boldsymbol{v}_t + \boldsymbol{g}}{\tau})$, where τ is the temperature parameter, and elements of $\boldsymbol{g}$ are sampled i.i.d. from Gumbel$(0, 1)$ (Maddison, Mnih, and Teh, 2017; Jang, Gu, and Poole, 2016).

This Gumbel-softmax formulation is used for convenience since directly applying the non-differentiable $\arg\max$ operator to obtain (discrete) one-hot represented tokens, as above in the RL case, results in the inability to back-propagate gradients through the model to obtain parameter updates (Zhang et al., 2017; Yu et al., 2017). By applying the Gumbel-softmax trick, we can collect embeddings for the entire predicted sequence $\mathbf{Z}_g = \{\tilde{\boldsymbol{z}}_i\}_{i=1}^n$. Similarly we denote the reference sequence embeddings as $\mathbf{Z}_r = \{\boldsymbol{z}_j\}_{j=1}^m$, but using the ground-truth one-hot-encoded input sequence $\boldsymbol{y}^* = \{\boldsymbol{w}_t\}_{t=1}^m$. Here n and m denote the lengths of the embedded sequences $\mathbf{Z}_g$ and $\mathbf{Z}_r$, respectively. Using $\mathbf{Z}_r$ and $\mathbf{Z}_g$, we can compute the sequence-level optimal transport loss between the reference (ground-truth) and model predictions using the inexact proximal point method for optimal transport (IPOT) algorithm (Xie et al., 2018). We use the cosine similarity as the cost function: $C_{ij} = C(\tilde{\boldsymbol{z}}_i, \boldsymbol{z}_j) = 1 - \frac{\tilde{\boldsymbol{z}}_i^T \boldsymbol{z}_j}{\|\tilde{\boldsymbol{z}}_i\| \|\boldsymbol{z}_j\|}$. The optimal transport distance is then formulated as:

$$\begin{aligned} \mathcal{L}_{\text{OT}} = \min_{\mathbf{T}_{ij}} & \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{T}_{ij} \mathbf{C}_{ij} \,, \\ \text{subject to} & \sum_{i} \mathbf{T}_{ij} = d_j, \ \ \forall j \in [1, m] \\ & \sum_{j} \mathbf{T}_{ij} = d_i, \ \ \forall i \in [1, n] \,, \end{aligned} \tag{4}$$

where $\mathbf{T}_{ij}$ is the transport matrix, $d_i = \frac{a_i}{\sum_j a_j}$, a_i is the number of times the i-th word appears in the generated sentence, and similarly for d_j and ground-truth sequence. The loss in (4) can be optimized with the IPOT algorithm Xie et al. (2018), outlined in Algorithm 1. Note that an iterative algorithm can be problematic for building the computational graph in modern deep learning frameworks; however, it can be proved by the *envelope algorithm* that the gradient flow through the transport matrix $\mathbf{T}_{ij}$ can be ignored Carter (2001). Therefore, this algorithm can be executed efficiently as gradients only depend on the cost function $C(\cdot, \cdot)$ (cosine similarity).

Algorithm 1 IPOT algorithm.

```
 1: Input: Embeddings $\mathbf{Z}_g = \{\tilde{\boldsymbol{z}}_i\}_i^n$, $\mathbf{Z}_r = \{\boldsymbol{z}_j\}_j^m$ and
    generalized step size $1/\beta$.
 2: $\boldsymbol{\sigma} = \frac{1}{m}\mathbf{1_m}$; $\mathbf{T}^{(1)} = \mathbf{1_n}\mathbf{1_m}^\top$
 3: $C_{ij} = C(\tilde{\boldsymbol{z}}_i, \boldsymbol{z}'_j)$; $\mathbf{A}_{ij} = \exp\left(-\frac{C_{ij}}{\beta}\right)$
 4: for $t = 1, 2, 3 \ldots$ do
 5:     $\mathbf{Q} = \mathbf{A} \odot \mathbf{T}^{(t)}$ // $\odot$ is the Hadamard product
 6:     for $k = 1, \ldots, K$ do // $K = 1$ in practice
 7:         $\boldsymbol{\delta} = \frac{1}{n\mathbf{Q}\boldsymbol{\sigma}}$; $\boldsymbol{\sigma} = \frac{1}{m\mathbf{Q}^\top\boldsymbol{\delta}}$
 8:     end for
 9:     $\mathbf{T}^{(t+1)} = \text{diag}(\boldsymbol{\delta})\mathbf{Q}\text{diag}(\boldsymbol{\sigma})$
10: end for
11: Return $\mathbf{T}, \mathbf{C} = \{C_{ij}\}$
```

Optimal-Transport-Enhanced RL

OT-only sequence modeling solution considered in Chen et al. (2019) basically describes a dynamic 1-gram matching scheme, whose training is generally stable. However, the diversity of natural language largely comes from the compositional complexity of k-grams, which simple 1-gram cannot capture. It is challenging to extended OT to dynamic k-gram matching. Specifically, we will have $\binom{n}{k} \times \binom{m}{k}$ distinct combinations of k-gram pairs to match between two sentences of length n and m. On the other hand, RL-only solutions suffer weak learning signals due to high gradient variance, though it is capable of matching long phrases, *i.e.*, k-grams for arbitrary k.

So motivated, we combine OT and RL to improve models for sequence generation. OT helps stabilizing training while encouraging semantic consistency, while RL helps to capture consistency in long phrases. The proposed OT-enhanced RL (OTRL) approach is constructed as a hybrid algorithm. The complete loss function for the model is written as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{MLE}} + \lambda_2 \mathcal{L}_{\text{OT}} + \lambda_3 \mathcal{L}_{\text{RL}} \,, \tag{5}$$

where $\mathcal{L}_{\text{MLE}}$ denotes the standard maximum likelihood objective of the sequence generation model of choice.

In practice, the sequence generation model will be first trained with the most restrictive loss, *i.e.*, $\mathcal{L}_{\text{MLE}}$, to get a good initial starting point. Then we gradually expand the exploration space by introducing the optimal transport loss $\mathcal{L}_{\text{OT}}$. This allows the model to search words during the generation process that account for both dynamic position matching and key word preservation. Finally we add the RL-based self-critic loss $\mathcal{L}_{\text{RL}}$, to fine-tune the model. This procedure can be implemented by parameter annealing. For instance, we can decrease λ_1 and increase λ_2 and λ_3 gradually when training. A similar annealing strategy is also employed in MIXER (Ranzato et al., 2016). Detailed discussion about the choice of the hyper-parameters is provided in Section .

Algorithm 2 OTRL for Seq2Seq learning.

```
1: Input: batch size ($B$), paired input and output sequences ($\mathbf{X}, \mathbf{Y}$), hyper-parameters ($\lambda_1, \lambda_2, \lambda_3$)
2: Build Seq2Seq model $\mathcal{M}(\cdot; \theta)$ and embedding matrix $\mathbf{E}$.
3: for iteration $= 1, \ldots$ MaxIter do
4:     $\lambda_1', \lambda_2', \lambda_3' = \text{AnnealingScheme}(\lambda_1, \lambda_2, \lambda_3)$
5:     for $i = 1, \ldots, B$ do
6:         Draw $\boldsymbol{x}_i, \boldsymbol{y}_i \sim (\mathbf{X}, \mathbf{Y})$ as:
7:         $\boldsymbol{x}_i = \{\tilde{\boldsymbol{w}}_{i,t}\}, \boldsymbol{y}_i = \{\boldsymbol{w}_{i,t}\}$
8:         // Compute output vectors from model:
9:         $\{\boldsymbol{v}_{i,t}\} = \mathcal{M}(\boldsymbol{x}_i; \theta)$
10:        // Encode model belief:
11:        $\hat{\boldsymbol{w}}_{i,t} = \text{Gumbel-softmax}(\boldsymbol{v}_{i,t})$
12:        // Compute embedding vectors:
13:        $\mathbf{Z}_{r,i} = \{\mathbf{E}^T \boldsymbol{w}_{i,t}\}, \mathbf{Z}_{g,i} = \{\mathbf{E}^T \tilde{\boldsymbol{w}}_{i,t}\}$
14:        // Decode greedily (test time):
15:        $\boldsymbol{v}_i' = \mathcal{M}(\boldsymbol{x}_i, \theta)$
16:        // Convert model outputs to words:
17:        $\hat{\boldsymbol{y}}_i = \text{argmax}[\text{softmax}(\boldsymbol{v}_i')]$
18:        $\boldsymbol{y}_i = \text{argmax}[\text{softmax}(\boldsymbol{v}_i)]$
19:    end for
20:    // Update $\mathcal{M}(\cdot; \theta)$ by optimizing:
21:    $\sum_{i=1}^{m} [\lambda_1' \mathcal{L}_{\text{MLE}}(\boldsymbol{x}_i, \boldsymbol{y}_i; \theta) + \lambda_2' \mathcal{L}_{\text{OT}}(\mathbf{Z}_{r,i}, \mathbf{Z}_{g,i}) + \lambda_3' \mathcal{L}_{\text{RL}}(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i)]$
22: end for
```

Related Work

RL in sequence generation A variety of RL approaches have been proposed to further improve sequence generation over MLE-based learning. Ranzato et al. (2016) used the REINFORCE algorithm (Williams, 1992) and Bahdanau et al. (2017) applied the actor-critic algorithm. These two approaches only use a RL-inspired loss to fine-tune the model pre-trained via MLE. In Ding and Soricut (2017), they proposed the softmax policy gradient (SPG) algorithm to allow for end-to-end training without a pre-training phase. Further, the reward augmented maximum algorithm likelihood (RAML) (Norouzi et al., 2016) is a hybrid RL method borrowing ideas from MLE and policy gradient. All these algorithms can be understood as some special cases of entropy-regularized policy optimization (ERPO) (Tan et al., 2018). To improve the performance of RL-based methods, recent work revisits traditional variance-reduction techniques, *i.e.*, offsetting the reward with a greedy roll-out baseline (2). Our work considers a new framework that seeks to execute systematic policy exploration space via OT, to decrease the variance of the policy optimization.

GAN in sequence generation The generative adversarial network (GAN) (Goodfellow et al., 2014) has achieved success in many computer vision tasks. Several works in NLP have also tried to incorporate GAN-inspired losses for text generation Yu et al. (2017); Zhang et al. (2017); Chen et al. (2018). Specifically, a discriminator is designed to compare the generated sequences with references to tell which ones are real or synthesized. This implies that the feedback from the discriminator into the generator is a sequence-based (global) loss. However, the discriminator of GAN is often hard to train for discrete observations. This happens because it is very easy for the discriminator to differentiate between real and synthetic sequences, and consequently the gradient from the discriminator can easily vanish. Further, there is still no principled way to avoid the so called *mode-collapsing* problem in GAN training (Arjovsky, Chintala, and Bottou, 2017a). Alternatively, our framework, unlike GAN, is robust to misspecification and does not require an additional design for the discriminator.

Optimal transport in NLP Optimal transport has been widely applied in computer vision applications (Rubner, Tomasi, and Guibas, 2000), and recently it has been applied to NLP tasks. In Kusner et al. (2015), the authors propose *word mover's distance* (WMD) for document classification and word embedding refinement, while in Alvarez-Melis and Jaakkola (2018) there is a goal of aligning the word embedding space with Gromov-Wasserstein distance (Mémoli, 2011) in an unsupervised way. Most recently, Chen et al. (2019) used OT as a sequence-level loss for sequence generation, as a regularization term in MLE-based learning. Here, optimal transport is used initially to refine word embeddings and capture key phrases, and later to regularize the RL objective.

Experiments

Datasets and setup

We consider three tasks to evaluate our model: *i*) **Machine translation**: the commonly used English-German translation dataset and English-Vietnamese translation dataset, IWSLT 2014 (Cettolo et al., 2014), which was also applied in Ranzato et al. (2016); Norouzi et al. (2016); Chen et al. (2019). We employ the same pre-processing in Ranzato et al. (2016); Norouzi et al. (2016). The EN-DE dataset has 146K/7K/7K paired sentences for training/validation/testing, respectively. The vocabulary size for English is 23K, and 32K for German. For EN-VI dataset, we train the model on IWSLT15 with 133K paired sentences, and test on IWSLT 2012 news dataset (NT12) and 2013 news dataset (NT13) following the same setup as Chen et al. (2019). *ii*) **Abstractive summarization**: two different datasets are employed for the summarization task, English gigawords (Graff et al., 2003) and CNN/Dailymail Hermann et al. (2015); Nallapati et al. (2016), which are standard summarization benchmarks. The English gigawords corpus consists of 200K/8K/2K source and target sentence pairs in training/validation/testing, respectively. The CNN/Daily-mail dataset has 287,113 training pairs, 13,368 validation pairs and 11,490 testing pairs. *iii*) **Image captioning**: we also consider an image captioning task with the COCO dataset (Lin et al., 2014), which contains 123,287 images in total and each image is annotated with at least 5 captions. Following the split method proposed in (Karpathy and Fei-Fei, 2015), 113,287 images are used for training and 5,000 images are used for validation and testing.

For fair comparison, we use Texar Hu et al. (2018) as our codebase to implement all proposed methods and baselines, and to compared models for IWSLT 2014 and gigawords.

Method	ROUGE-1	ROUGE-2	ROUGE-L
SummaRuNNer (Nallapati, Zhai, and Zhou, 2017)	39.6	16.2	35.3
Pointer-Generator See, Liu, and Manning (2017)	36.44	15.66	33.42
Pointer-Generator + Coverage See, Liu, and Manning (2017)	39.53	17.28	36.38
Saliency + Entailment rewardPasunuru and Bansal (2018)	40.43	18.00	37.10
Key information guide network Li et al. (2018)	38.95	17.12	35.68
Sentence Rewriting Chen and Bansal (2018)	40.88	17.80	38.54
Inconsistency loss Hsu et al. (2018)	40.68	17.97	37.13
ML + RL Paulus, Xiong, and Socher (2018)	39.87	15.82	36.90
ML + Intra-Attention Paulus, Xiong, and Socher (2018)	38.30	14.81	35.49
Pointer-Generator	36.25	16.17	33.41
Pointer-Generator + Coverage Penalty	39.12	17.35	36.12
Transformer + Copy + Coverage Penalty	39.25	17.54	36.45
Bottom-Up Summarization	41.22	**18.68**	38.34
Pointer-Generator + OT	39.75	17.87	38.22
Pointer-Generator + Coverage Penalty + OT	40.28	18.17	38.51
Pointer-Generator+ Coverage Penalty + OTRL	**41.40**	18.22	**38.86**

Table 1: Results of abstractive summarization on CNN/Daily-mail. First section is the MLE-training baseline with Seq2Seq model. Models in the second section are results trained using OpenNMT codebase Gehrmann, Deng, and Rush (2018). The last section shows the results of our baselines and the proposed model.

Method	CIDEr	BLEU-4	BLUE-3	BLEU-2	BLEU-1	ROUGE	METEOR
Soft Attention (Xu et al., 2015)	-	24.3	34.4	49.2	70.7	-	23.9
Hard Attention (Xu et al., 2015)	-	25.0	35.7	50.4	71.8	-	23.0
Show & Tell (Vinyals et al., 2015)	85.5	27.7	-	-	-	-	23.7
ATT-FCN (You et al., 2016)	-	30.4	40.2	53.7	70.9	-	24.3
SCN-LSTM (Gan et al., 2017)	101.2	33.0	43.3	56.6	72.8	-	25.7
Adaptive Attention (Lu et al., 2017)	108.5	33.2	43.9	58.0	74.2	-	26.6
MLE	106.3	34.3	45.3	59.3	75.6	55.2	26.2
MLE+OT	107.9	**34.8**	46.1	60.1	76.2	55.6	26.5
MLE+SC	110.4	33.4	45.8	61.4	**79.5**	56.0	26.6
MLE+OTRL	**111.8**	34.4	**46.6**	**61.8**	79.3	**56.2**	**26.8**

Table 2: Results on the testing set on COCO-caption dataset on the model with the best evaluation loss. The first section are results from previous image caption models. The second section are results of our baselines and the proposed model.

The performance of our implementation is at least comparable with the reported results. Both encoder and decoder are one-layer LSTMs with 256 hidden states. The word embedding dimension is also set to 256. Dropout is also applied with rate 0.2. We use Adam optimization with learning rate 0.001 and batch size $= 64$ for training.

For the CNN/Daily-mail experiment, we preprocess the data using the same method proposed in See, Liu, and Manning (2017). We use the most recent extractor-abstractor framework Chen and Bansal (2018); Gehrmann, Deng, and Rush (2018), consisting of two parts, extractor and abstractor. We implement the extractor following the structure from Chen and Bansal (2018), and modify the abstractor using our OTRL loss function without changing the network architecture, in comparison to the OpenNMT implementation Klein et al.. The word embedding size is set to 128. The encoder and decoder are also both one-layer LSTMs with 256 hidden states.

The image captioning experiment is implemented based on a public[1] Pytorch implementation (Paszke et al., 2017). We pre-train the image feature extractor using the bottom-up attention model (Anderson et al., 2018). The setup of the caption decoder is the same as self-critic model (Rennie et al., 2017), where the image feature is fed into a one-layer LSTM. For the LSTM, the hidden state dimension and the dimension of word vectors are both set to 512. All experiments are performed on one NVIDIA TITAN X GPU.

Neural machine translation

The commonly applied metric BLEU score Papineni et al. (2002) is used to evaluate the performance of different models. To assess variation, we test each model 5 times with

[1] https://github.com/ruotianluo/self-critical.pytorch

different random seeds and calculate the mean and standard deviation for each test run. The results in Table 3 show that our model results in consistent improvement over the baselines, across all three datasets.

Abstractive summarization

ROUGE-1, -2 and -L scores Lin (2004) are employed to evaluate the performance of the abstractive summarization model.

English gigawords Table 4 shows performance comparisons of seven baselines and our model. We implement all baseline models on Texar Hu et al. (2018) under the same setup as ours. Our model achieves the highest score on the test dataset on ROUGE-1, -2, -L metrics consistently. Generated examples are found in the Appendix.

CNN/Daily-mail We also run our model on the large summarization dataset CNN/Daily-mail. Table 1 shows ROUGE score results on different models. We compare our model with the most recent works, *e.g.*, Gehrmann, Deng, and Rush (2018); Chen and Bansal (2018). Our OTRL framework achieves better ROUGE-1 and ROUGE-L score on this dataset, indicating that our model's predictions can better capture the semantic meaning from the source paragraph.

Image captioning

For the image caption task, the OT loss between the generated sentences and the ground truth sentences is employed to regularize training. Image features are extracted from fast R-CNN Ren et al. (2015) following the same setup as (Anderson et al., 2018), which is denoted as bottom-up features.

We perform our comparisons based on two different scenarios: teacher-forcing MLE objective with attention mechanism and the self-critic RL objective. In the teacher-forcing setup, we select the checkpoint which has the minimal loss on the validation set from each model. In the self-critic setup, we select the checkpoint with the largest evaluation CIDEr score Vedantam, Lawrence Zitnick, and Parikh (2015) from each model. We then evaluate all models using different metrics on the testing set. The split of the training, evaluation and test set is the same as in Karpathy and Fei-Fei (2015), for fair comparison, and the detailed results are shown in Table 2. We report BLEU-k (k from 1 to 4) (Papineni et al., 2002), CIDEr (Vedantam, Lawrence Zitnick, and Parikh, 2015), and METEOR (Banerjee and Lavie, 2005) scores. It is observed that by employing optimal transport, under two different setups, our OTRL framework consistently outperforms all baselines and compared methods.

Analysis

Hyper-parameter setup

For hyper-parameters $\lambda_1, \lambda_2, \lambda_3$, we use parameter grid-search to find the best setup for different tasks. In machine translation and abstractive summarization, We set $\lambda_1 = 0.9, \lambda_2 = 0.1, \lambda_3 = 0$ as our initialization. Then we gradually decrease λ_1 to 0.7, and increase λ_3 to 0.2. This annealing process starts after the 7-th epoch. Note that the annealing process is implemented as a linear decay function provided in Tensorflow (Abadi et al., 2016)/Pytorch (Paszke et al., 2017). For the image-captioning task, the training is divided into two phases. In the first phase, we set $\lambda_1 = 0.9, \lambda_2 = 0.1, \lambda_3 = 0$ and choose the best on the evaluation set as the initializer for the second phase, where $\lambda_1 = 0, \lambda_2 = 0.1, \lambda_3 = 1$. The annealing trace plot is provided in the Appendix.

Ablation study

To evaluate the stability of different frameworks, in Figure 2 we show the convergence curve of the ROUGE-1 F1 score on the testing dataset for each training epoch. We test different methods on the English gigawords dataset with the same sequence generation architecture. The num-

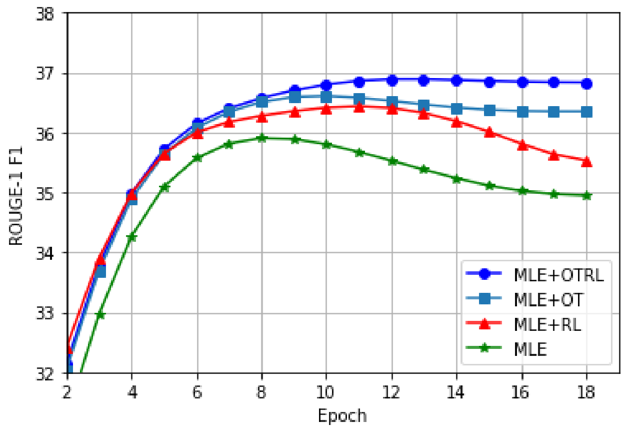

Figure 2: Convergence curve of validation ROUGE-1 F1 score *vs.* training epochs on English gigawords dataset.

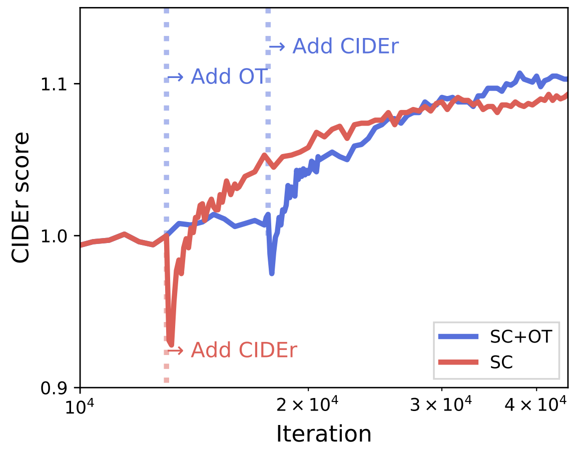

Figure 3: Convergence curve of validation CIDEr score *vs.* training iterations on MS-COCO dataset. The number of epochs is 50.

Model	EN-DE	EN-VI NT12	EN-VI NT13
MLE	26.44 ± 0.18	23.76 ± 0.17	26.10 ± 0.07
Scheduled Sampling Bengio et al. (2015)	24.11 ± 0.10	23.71 ± 0.14	26.11 ± 0.12
RAML Norouzi et al. (2016)	27.22 ± 0.14	24.24 ± 0.16	26.42 ± 0.17
MIXER Ranzato et al. (2016)	26.53 ± 0.11	23.81 ± 0.18	26.33 ± 0.17
SPG Ding and Soricut (2017)	26.62 ± 0.05	24.27 ± 0.11	26.51 ± 0.09
ERPO Tan et al. (2018)	27.82 ± 0.11	24.56 ± 0.21	26.88 ± 0.13
Seq2Seq-OT Chen et al. (2019)	27.79 ± 0.12	24.46 ± 0.11	26.91 ± 0.13
MLE+OTRL	$\mathbf{28.13 \pm 0.11}$	$\mathbf{24.85 \pm 0.13}$	$\mathbf{27.11 \pm 0.18}$

Table 3: Results of machine translation for EN-DE and EN-VI.

Method	ROUGE-1	ROUGE-2	ROUGE-L
MLE	36.11 ± 0.21	16.39 ± 0.16	32.32 ± 0.19
RAMLNorouzi et al. (2016)	36.30 ± 0.04	16.69 ± 0.20	32.49 ± 0.17
SPG Ding and Soricut (2017)	36.48 ± 0.24	16.84 ± 0.26	32.79 ± 0.26
MIXERRanzato et al. (2016)	36.34 ± 0.23	16.61 ± 0.25	32.57 ± 0.15
Scheduled Sampling Bengio et al. (2015)	36.59 ± 0.12	16.79 ± 0.22	32.77 ± 0.17
ERPOTan et al. (2018)	36.72 ± 0.29	16.99 ± 0.17	32.95 ± 0.33
Seq2Seq-OT Chen et al. (2019)	36.82 ± 0.25	17.22 ± 0.16	33.15 ± 0.23
MLE+OTRL	$\mathbf{37.2 \pm 0.19}$	$\mathbf{17.63 \pm 0.15}$	$\mathbf{33.76 \pm 0.13}$

Table 4: Results of text summarization on gigawords dataset.

ber of training epochs is set to 18. For the MLE baseline model, the ROUGE score gradually decays, presumably due to over-fitting. OT training is more stable and the ROUGE is smoothly improved. However, MLE+OT does not perform well, probably because the exploration space for OT+MLE is still limited. The implementation of the RL+MLE model is based on Texar's ERPO (Tan et al., 2018). This training method converges fast but suffers from over-fitting. This indicate that after approximately 12 epochs, MLE eventually dominates the training procedure since the RL signal can no longer provide any useful information but just overfits the training data. With the help of both OT and RL, our OTRL is stable and has the best performance among other baselines, and the testing ROUGE-1 improvement curve is still smooth.

Exploration space analysis

Since it is difficult to directly visualize the changing of the exploration space, here we conduct an experiment on image captioning, to show how the CIDEr score changes on the validation dataset with respect to the loss switching in training objective. As shown in the Figure 3, we initialize both experiments (red and blue) by training with the same MLE objective. Note that training under the MLE objective has converged according to the validation loss. The red line experiment then switches the training loss from MLE to self-critic (SC). Consequently, its CIDEr score on the validation set drops significantly, indicating the exploration space becomes larger , so that the performance drops dramatically after the switching. For the blue line experiment, the objective is regularized with OT. When the MLE+OT loss switched to OTRL, the performance does not drop significantly, and the blue line converges faster than the red line with the same hyper-parameter setup. This phenomenon implies that the exploration space of the blue line is presumably controlled with the help of OT, resulting in a smaller variance than the red line, thus a more efficient training.

Conclusions

We have presented a new framework to control (regularize) the exploration space of policy optimization. This also provides a new way to reduce the variance of RL approaches in NLP tasks, so that RL-based sequence generation models can be further improved. We evaluate our new sequence level loss on three NLP tasks. The consistent improvement in machine translation, abstractive summarization and image captioning strongly support our motivations.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments. This research was supported in part by DARPA, DOE, NIH, ONR, NSF and RTI Internal research & development funds.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283.

Alvarez-Melis, D., and Jaakkola, T. S. 2018. Gromov-wasserstein alignment of word embedding spaces. In *EMNLP*.

Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; and Zhang, L. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*.

Arjovsky, M., and Bottou, L. 2017. Towards principled methods for training generative adversarial networks. In *ICLR*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017a. Wasserstein gan. In *arXiv preprint arXiv:1701.07875*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017b. Wasserstein generative adversarial networks. In *ICML*.

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A.; and Bengio, Y. 2017. An actor-critic algorithm for sequence prediction. *ICLR*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Banerjee, S., and Lavie, A. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL Workshop*.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Carter, M. 2001. *Foundations of mathematical economics*. MIT Press.

Cettolo, M.; Niehues, J.; Stüker, S.; Bentivogli, L.; and Federico, M. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, 57.

Chen, Y.-C., and Bansal, M. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *ACL*.

Chen, L.; Dai, S.; Tao, C.; Shen, D.; Gan, Z.; Zhang, H.; Zhang, Y.; and Carin, L. 2018. Adversarial text generation via feature-mover's distance. In *NIPS*.

Chen, L.; Zhang, Y.; Zhang, R.; Tao, C.; Gan, Z.; Zhang, H.; Li, B.; Shen, D.; Chen, C.; and Carin, L. 2019. Improving sequence-to-sequence learning via optimal transport. In *ICLR*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Chopra, S.; Auli, M.; and Rush, A. M. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*.

Ding, N., and Soricut, R. 2017. Cold-start reinforcement learning with softmax policy gradient. In *NIPS*.

Gan, Z.; Gan, C.; He, X.; Pu, Y.; Tran, K.; Gao, J.; Carin, L.; and Deng, L. 2017. Semantic compositional networks for visual captioning. In *CVPR*.

Gehrmann, S.; Deng, Y.; and Rush, A. M. 2018. Bottom-up abstractive summarization. In *EMNLP*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Graff, D.; Kong, J.; Chen, K.; and Maeda, K. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *NIPS*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*.

Hsu, W.-T.; Lin, C.-K.; Lee, M.-Y.; Min, K.; Tang, J.; and Sun, M. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *ACL*.

Hu, Z.; Shi, H.; Yang, Z.; Tan, B.; Zhao, T.; He, J.; Wang, W.; Qin, L.; Wang, D.; et al. 2018. Texar: A modularized, versatile, and extensible toolkit for text generation. *arXiv preprint arXiv:1809.00794*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with Gumbel-softmax. In *ICLR*.

Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. M. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Kusner, M.; Sun, Y.; Kolkin, N.; and Weinberger, K. 2015. From word embeddings to document distances. In *ICML*.

Li, C.; Xu, W.; Li, S.; and Gao, S. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *NAACL*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Lu, J.; Xiong, C.; Parikh, D.; and Socher, R. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *CVPR*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*.

Mémoli, F. 2011. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics* 11(4):417–487.

Nallapati, R.; Zhou, B.; Gulcehre, C.; Xiang, B.; et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Nallapati, R.; Zhai, F.; and Zhou, B. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Norouzi, M.; Bengio, S.; Jaitly, N.; Schuster, M.; Wu, Y.; Schuurmans, D.; et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *NIPS*.

Paisley, J.; Blei, D.; and Jordan, M. 2012. Variational bayesian inference with stochastic search. In *ICML*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Pasunuru, R., and Bansal, M. 2018. Multi-reward reinforced summarization with saliency and entailment. In *NAACL*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Paulus, R.; Xiong, C.; and Socher, R. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Prabhumoye, S.; Tsvetkov, Y.; Salakhutdinov, R.; and Black, A. W. 2018. Style transfer through back-translation. *ACL*.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*.

Rennie, S. J.; Marcheret, E.; Mroueh, Y.; Ross, J.; and Goel, V. 2017. Self-critical sequence training for image captioning. In *CVPR*.

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The earth mover's distance as a metric for image retrieval. In *IJCV*.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Salimans, T.; Zhang, H.; Radford, A.; and Metaxas, D. 2018. Improving GANs using optimal transport. In *ICLR*.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *ICML*.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: summarization with pointer-generator networks. In *ACL*.

Shen, T.; Lei, T.; Barzilay, R.; and Jaakkola, T. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Tan, B.; Hu, Z.; Yang, Z.; Salakhutdinov, R.; and Xing, E. 2018. Connecting the dots between mle and rl for sequence generation. *arXiv preprint arXiv:1811.09740*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Vedantam, R.; Lawrence Zitnick, C.; and Parikh, D. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Wang, X.; Chen, W.; Wang, Y.-F.; and Wang, W. Y. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*.

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

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *TACL*.

Xie, Y.; Wang, X.; Wang, R.; and Zha, H. 2018. A fast proximal point method for Wasserstein distance. In *arXiv:1802.04307*.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A. C.; Salakhutdinov, R.; Zemel, R. S.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Xu, H.; Luo, D.; Zha, H.; and Carin, L. 2019. Gromov-wasserstein learning for graph matching and node embedding. In *ICML*.

You, Q.; Jin, H.; Wang, Z.; Fang, C.; and Luo, J. 2016. Image captioning with semantic attention. In *CVPR*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Zhang, Y.; Gan, Z.; Fan, K.; Chen, Z.; Henao, R.; Shen, D.; and Carin, L. 2017. Adversarial feature matching for text generation. In *ICML*.