Adaptive Prior-Dependent Correction Enhanced Reinforcement Learning for Natural Language Generation

Wei Cheng[1], Ziyan Luo[2], Qiyue Yin [3*]
[1] Technology and Data Center, JD.com , Beijing, China
[2] University of California, San Diego
[3] Institute of Automation, Chinese Academy of Sciences, Beijing, China
weicheng5993@foxmail.com, discat@foxmail.com, qyyin@nlpr.ia.ac.cn

Abstract

Natural language generation (NLG) is an important task with various applications like neural machine translation (NMT) and image captioning. Since deep-learning-based methods have issues of exposure bias and loss inconsistency, reinforcement learning (RL) is widely adopted in NLG tasks recently. But most RL-based methods ignore the deviation ignorance issue, which means the model fails to understand the extent of token-level deviation well. It leads to semantic incorrectness and hampers the agent to perform well. To address the issue, we propose a technique called adaptive prior-dependent correction (APDC) to enhance RL. It leverages the distribution generated by computing the distances between the ground truth and all other words to correct the agent's stochastic policy. Additionally, some techniques on RL are explored to coordinate RL with APDC, which requires a reward estimation at every time step. We find that the RL-based NLG tasks are a special case in RL, where the state transition is deterministic and the afterstate value equals the Q-value at every time step. To utilize such prior knowledge, we estimate the advantage function with the difference of the Q-values which can be estimated by Monte Carlo rollouts. Experiments show that, on three tasks of NLG (NMT, image captioning, abstractive text summarization), our method consistently outperforms the state-of-the-art RL-based approaches on different frequently-used metrics.

Introduction

Natural language generation (NLG) is a promising task in natural language processing that aims to generate a piece of new text. It has a wide range of applications, including neural machine translation (NMT), image captioning, text summarization and so on.

Since NLG is a sequence prediction task, it usually adopts *maximum likelihood estimation* (MLE) with teacher-forcing technique (Cho et al. 2014; Williams and Zipser 1989). MLE training maximizes the log-likelihood of each word conditioned on its previous context, but the model is evaluated using a sequence-level metric like BLEU (Papineni et al. 2002), ROUGE (Lin 2004) and CIDEr (Vedantam, Lawrence Zitnick, and Parikh 2015) which are non-differentiable. This *loss inconsistency* issue hampers the algorithms to optimize the sequence as a whole. The teacher-forcing technique trains the model to predict the next word given the previous ground truth words as input, but at the test stage, the model utilizes the previously generated words instead of the ground-truth as the input to generate the following sentence. It results that the model has never been exposed to its own predictions. It is well known as the *exposure bias* issue. Besides, the MLE-based approaches treat all incorrect outputs equally during training (Li et al. 2019).

To address the *loss inconsistency* and *exposure bias* issues, reinforcement learning (RL) methods have been adopted to train NLG models to avoid these two issues. For example, policy gradient and actor-critic methods (Bahdanau et al. 2017; Rennie et al. 2017; Ranzato et al. 2016; Chen et al. 2020a; Wu et al. 2018) are applied to this task. Unlike MLE, which maximizes the log-likelihood, RL-based methods optimize the reward function. The reward function can either be differentiable and non-differentiable, thus the non-differentiable sequence-level metrics like BLEU can be properly optimized. RL-based methods also solve the *exposure bias* issue as the training sentences are generated by the agent rather than using the ground-truth as the instruction.

However, these RL-based training methods can not solve the issue that all incorrect outputs are treated equally well, which we call it *deviation ignorance* in our RL-based method. It means that, the models may fail to understand how much the prediction distribution deviates from a prior distribution related to the ground-truth at token-level. Since in RL training method for NLG, the objective aims to maximize the reward, such as BLEU and ROUGE, by estimating gradient. However, this reward assigns unfair scores to different incorrect model outputs, which means that all incorrect token-level outputs are treated equally in a sequence during training. In some cases, a huge deviation between the predicted token and the ground-truth token can directly cause semantically incorrect results. For instance, the ground-truth sentence is "the boy is eating an apple". It is clearly better for the prediction sentence to be "the kid is eating an apple" or "the boy is having an apple" rather than "the cat is eating an apple" or "the dog is eating an apple". But the metrics, such as BLEU and ROUGE assign the

*Corresponding author

same scores for these different prediction sentences, since these evaluation metrics are based on recall and precision, which treat all the incorrect token equally. Whereas, the token "kid" is more semantically similar to "boy" than to "cat" or "dog", which results in the total semantic incorrectness of the sentences. The extent of incorrectness of the wrong predictions should be aware by the models, which also provides important guidance on the training of models.

To alleviate the issue of *deviation ignorance*, we proposed an *adaptive prior-dependent correction* (APDC) objective for the RL training method. APDC can adaptively correct the deviated prediction distribution with an adaptive Kullback-Leibler (KL) divergence penalty term for the RL training objective. The KL divergence is computed by two probability distributions. The first distribution is the model training prediction output and the second distributions is a prior distribution computed by the well-trained word embedding of the ground-truth. It can both reduce the token-level deviation and the bias when the models cannot predict the ground-truth word. Because merely optimizing metrics like CIDEr and METEOR cannot fully leverage such important prior information, and metrics like BLEU and ROUGE cannot even distinguish the extent of deviation. Additionally, the adaptive mechanism helps the algorithm to choose to pay how much attention to the token-level distribution accuracy.

To make RL work well with APDC, further algorithmic improvement is needed. We find that the RL-based NLG tasks are a special case in RL, where the *afterstate* (Sutton and Barto 2018) value equals the state-action value (Q-value) at every time step. In such a case, the state transition is deterministic after the action is given, but other tasks such as Atari Games (Mnih et al. 2013) and StarCraft II (Vinyals et al. 2019), can have different next states with a certain state transition probability. Such RL tasks can utilize this kind of prior knowledge to produce a more efficient learning method (Sutton and Barto 2018), but most of the previous RL method (Bahdanau et al. 2017; Chen et al. 2018; Ranzato et al. 2016; Rennie et al. 2017; Wu et al. 2018) for NLG ignored this characteristic. However, adaptively deciding the extent of token-level correction in APDC needs a token-level reward. Thus, we estimate the Q-value on each step using K Monte Carlo rollouts. Using the aforementioned prior knowledge, we further estimate an advantage function merely based on Q-values to reduce the reward variance and assign every token with an instant feedback.

In this work, our contributions are summarized as follows:

- To address the *deviation ignorance* issue, we propose a novel technique, *adaptive prior-dependent correction* (APDC) to enhance RL on *natural language generation* (NLG). APDC adaptively corrects the stochastic policy using the distances of the embeddings between the ground truth and other words, where "adaptively" means our correction is self-regulating according to the need.
- We explore *advantage-function-weighted policy gradient* (APG) method to coordinate with APDC, which requires a reward estimation at each step.
- Our experiments cover three major tasks in NLG (neural machine translation, image captioning, and abstractive text summarization). The results show that our method consistently outperforms the state-of-the-art RL-based approaches in a wide spectrum of applications and has great generalizability.

Related Work

Recent efforts on incorporating RL to standard DL-based methods solve the aforementioned two issues (DL-based related works can be seen in Appendix). MIXER (Ranzato et al. 2016) combines optimizing with XENT loss and REINFORCE algorithm (Williams 1992) to enable it directly optimize the non-differentiable metrics. Self-critical sequence training (SCST) (Rennie et al. 2017) also leverages a policy gradient algorithm, but it inventively uses a self-critical method in its baseline which is calculated by the algorithm at the inference stage. SPIDEr (Liu et al. 2017) explores another way of reward estimation, using the rollout algorithm to estimate the Q-value of each action. Based on SCST, Chen et al. (2018) introduces the temporal-difference (TD) learning method. Optimal-Transport-Enhanced RL (OTRL) (Chen et al. 2020b) introduces Optimal Transport (OT) to RL to stabilize training, which can be applied to the sequence generation problem.

Background

Training with XENT Loss

Traditionally, DL-based methods employ *maximum likelihood estimation* (MLE) with teacher-forcing technique (Cho et al. 2014; Williams and Zipser 1989), which maximizes the log-likelihood by lowering the cross-entropy (XENT) loss during the training stage, while evaluate the model using a sequence-level metric like BLEU (Papineni et al. 2002), ROUGE (Lin 2004) and CIDEr (Vedantam, Lawrence Zitnick, and Parikh 2015) which are non-differentiable.

In traditional LSTM decoder (Vinyals et al. 2015), based on MLE, the task is to maximize the conditional possibility given that Z is the output of the encoder:

$$p(W \mid Z)=\prod_{t=0}^{T} p\left(w_t \mid w_{0:t-1}, Z\right) \tag{1}$$

where w_t is the word at time step t, $w_{0:t-1}$ is the previously generated words, W is the generated sentence $w_{0:T}$.

Thus, the XENT loss objective can be defined as:

$$L_{MLE}(\theta)=-\sum_{t=0}^{T} \log p_\theta\left(w_t^* \mid w_{0:t-1}^*, Z\right) \tag{2}$$

where w_t^* is the ground-truth word at step t and $w_{0:t-1}^*$ is the previous ground-truth words. By lowering the objective above, the algorithm maximizes the log-likelihood.

Problem Formulation

The *natural language generation* (NLG) problem can be formulated as a *Markov Decision Process* (MDP), which is modeled by the five-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$. In the five-tuple, S is the state space, A the action space, T the transition function $\mathcal{T}: \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_+$, and $\mathcal{R}: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function. Below are the components of the five-tuple:

- **State** $\mathcal{S}$. The state s_t at time step t consists of the encoding of the previously generated sentence $w_{0:t-1}$ and the encoder's output Z:

$$s_t = (w_0, w_1, ..., w_{t-1}, Z) \tag{3}$$

- **Action** $\mathcal{A}$. An action is a word to be generated, namely $a_t = w_t$, in the action space $\mathcal{A} \in \mathbb{R}^n$. The action space $\mathcal{A}$ of our agent is a large discrete action space, which contains all the available words in the dictionary with the length n.
- **Transition function** $\mathcal{T}$. It defines the probability of transition from the current state s_t to the next state s_{t+1}. Once the agent takes an action (i.e. selects a word), the transition is determined:

$$s_{t+1} = (s_t, a_t) \tag{4}$$

According to Equations (3) and (4), we have:

$$P(s_{t+1}|s_t, a_t) \equiv 1 \tag{5}$$

- **Reward** $\mathcal{R}$. The environment gives a instant feedback $R(s, a)$ to the agent when it takes an action $a \in \mathcal{A}$ at a state $s \in \mathcal{S}$ to evaluate the action a at state s. In our tasks, the reward is computed by comparing the generated sequence to corresponding ground-truth sequences. The reward is defined as follows:

$$r_t = \begin{cases} 0 & 0 \le t < T \\ r & t = T \end{cases} \tag{6}$$

where r is the score computed using the evaluation metrics, which are non-differentiable sequence-level metrics such as BLEU or CIDEr, and T is the final time step.

- **Discount factor** γ. The discount factor $\gamma \in [0, 1]$ is the hyper-parameter representing the trade-off between the instant feedback and the long-term yield. Specifically, when $\gamma = 0$, the agent only sees the instant reward. When $\gamma = 1$, all the future reward is seen by the agent.

Method

Training with Advantage-function-weighted Policy Gradient

Reinforcement learning (RL) can be used as a method for optimizing model parameters over flexible performance metrics, such as BLEU, ROUGE, and CIDEr, in NLG tasks. In RL-based NLG tasks, the language generative models, such as Transformer and LSTM, can be viewed as an agent that interacts with an environment, e.g. words (tokens) and source sentences for NMT, words and images for image captioning. The parameters of the model, θ, define a policy π_θ. The execution of the policy results in an "action", the prediction of the next token. After executing the action, the agent updates its internal state. Once the end-of-sequence (EOS) has been reached, the agent observes a final "reward" r such as BLEU. Details can be seen in the previous section. The whole architecture of our proposed method: APG with APDC is shown in Figure 1.

Policy gradient training In RL, the agent aims to maximize the cumulative rewards $\mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^{t-1} r\right]$ with discount factor γ by estimating the policy gradient $\nabla_\theta L_{RL}(\theta)$ and updating its parameters, instead of maximum likelihood estimation. In policy gradient methods, the expected gradient can be approximated using a single Monte-Carlo sample $(a_0, a_1, ..., a_{t-1})$ from π_θ, and the gradient $\nabla_\theta L_{RL}(\theta)$ can be calculated as follows:

$$\nabla_\theta L_{RL}(\theta) = \sum_{t=0}^{T} \mathbb{E}_\pi \left[\phi_t \nabla_\theta log \left(\pi_\theta \left(a_t|s_t\right)\right)\right] \tag{7}$$

where ϕ can be many formulas, such as $R = r - r_{baseline}$ and $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$ (advantage function). $Q^\pi(s_t, a_t)$ is the Q-value (state-action value) function and $V^\pi(s_t)$ is the state value function. The policy gradient method has high variance on the gradient estimation, but using a baseline can decrease the variance of gradient estimation, and the expected gradient remains unchanged (Sutton and Barto 2018). Among the formulas, the advantage function yields almost the lowest possible variance, though in practice, the advantage function is not known and must be estimated (Schulman et al. 2016). Therefore, we use the advantage function instead of other formulas, which is different from (Rennie et al. 2017; Ranzato et al. 2016; Wu et al. 2018; Chen et al. 2020b), and the gradient $\nabla_\theta L_{RL}(\theta)$ can be calculated with the advantage function:

$$\nabla_\theta L_{RL}(\theta) = \sum_{t=0}^{T} \mathbb{E}_\pi[A^\pi(s_t, a_t)\nabla_\theta log(\pi_\theta(a_t|s_t))] \tag{8}$$

Advantage function estimation We use the generalized advantage estimator (GAE) (Schulman et al. 2016) to estimate the advantage function. It can be calculated as follows:

$$A^\pi(s_t, a_t) = \sum_{l=1}^{\infty} (\gamma\lambda)^l (r_t + \gamma V^\pi(s_{t+l+1}) - V^\pi(s_{t+l})) \tag{9}$$

Here we use GAE(γ, 0) ($l = 0$), it can be calculated as follows:

$$A^\pi(s_t, a_t) = r_t + \gamma V^\pi(s_{t+l+1}) - V^\pi(s_{t+l}) \tag{10}$$

Therefore, we need to estimate the value function $V^\pi(s_t)$ to estimate the advantage function. In RL, according to the definition of Q-value and value function, the Q-value function can be calculated as follows:

$$Q^\pi(s_t, a_t) = r_t + \gamma \sum_{s_{t+1} \in S} P(s_{t+1}|s_t, a_t) V^\pi(s_{t+1}) \tag{11}$$

where $P(s_{t+1}|s_t, a_t)$ is the state transition probability. Due to the determinacy of the state transition, the probability $P(s_{t+1}|s_t, a_t) \equiv 1$. Here we set $\gamma = 1$ for our NLG tasks, which is the same as recent work. Thus the process of calculating the value function can be simplified as follows:

$$V^\pi(s_{t+1}) = Q^\pi(s_t, a_t) - r_t \tag{12}$$

According to Equation (6) and (12), the estimated advantage function can be written as:

$$\tilde{A}^\pi(s_t, a_t) = \tilde{Q}^\pi(s_t, a_t) - \tilde{Q}^\pi(s_{t-1}, a_{t-1}) \tag{13}$$

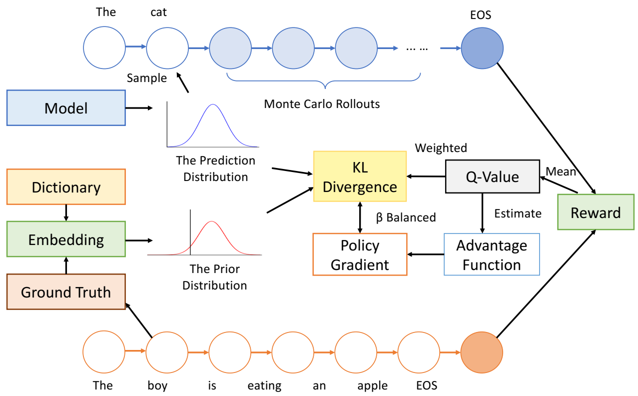

Figure 1: Architecture of our approach: APG with APDC.

Where $\tilde{A}^{\pi}(s_t, a_t)$ and $\tilde{Q}^{\pi}(s_t, a_t)$ are the estimations of $A^{\pi}(s_t, a_t)$ and $Q^{\pi}(s_t, a_t)$ .Thus, we only need to estimate the Q-value function to calculate the advantage function.

Q-value function estimation Applying RL techniques in NLG tasks may suffer from the large action space issue (typically the case in NLG tasks with large vocabularies) (Bahdanau et al. 2017), which leads to inaccurate Q-value estimation. It can be alleviated by pruning the actions with very small possibilities to take. The agent samples from the actions with top N probability (N is much less than the vocabulary size). Here we use same method as (Liu et al. 2017) to sample actions in each step and generate complete sentences for estimating Q-value:

$$\tilde{Q}^{\pi}(s_t, a_t) = \frac{1}{k}\sum_{k=1}^{k} R[a_{0:t}; (a_{t+1:T})_k] \tag{14}$$

k $a_{t+1:T}$ are sampled from the current policy, and compute the average of the k final rewards to estimate Q-value.

Adaptive Prior-Dependent Correction Enhanced Reinforcement Learning

Most RL training methods for NLG tasks ignore the issue of *deviation ignorance* which ignores the similarity between the correct and incorrect predictions, and treats all incorrect predictions (deviation) equally. The most straightforward way to solve this issue is to improve the reward (evaluation metric), and some existing metrics such as METEOR and CIDEr can alleviate this issue to some extent. But the main drawback of this solution is the lack of flexibility, and it is difficult for us to design a common reward for all NLG tasks. Another drawback is that, it cannot make full use of the prior knowledge. Therefore, we proposed APDC that is a more flexible method to alleviate *deviation ignorance* issue for NLG tasks.

Prior-dependent correction (PDC) with KL divergence To alleviate this issue, we enhance the RL objective $L_{RL}(\theta)$ with an additional objective $L_{KL}(\theta)$ which makes better use of the prior knowledge to capture the deviation between incorrect prediction of the model and the ground truth. We use Kullback-Leibler (KL) divergence to measure how well the predicted distribution $p_\theta(w_t)$ matches the prior distribution of ground truth $p^*(w_t)$ at each time step during training. The KL divergence is calculated as follows:

$$L_{KL}(\theta) = \sum_{t=0}^{T} KL\left[p^*(w_t)||p_\theta(w_t)\right] \tag{15}$$

where $p_\theta(w_t) = p_\theta(w_t \mid w_0, w_1, \ldots, w_{t-1}, Z)$, and it is the predicted distribution of the model (where $w_t \sim p_\theta(w_t)$ is a random variable) and $p^*(w_t)$ is a prior distribution related to ground truth at time step t, which will be introduced later. In this way, the agent can capture the deviation and update its parameters "unequally" for different incorrect predictions. Meanwhile, it adds a constraint that the stochastic policy not to far from the prior distribution $p^*(w_t)$.

The prior distribution $p^*(w_t)$ In NLG tasks, one of the priors we can use is the pre-trained word embedding. Therefore, we pre-train the word embedding. Then we utilize the embedding to obtain the prior distribution. Making full use of the prior knowledge of word embeddings, the prior distribution reflects the similarity between words. The aforementioned prior distribution $p^*(w_t)$ is calculated as:

$$p^*(w_t) = \sigma\left(cos\_sim\left(emb\left(w_t^*\right), emb\left(w_t\right)\right)\right) \tag{16}$$

where $\sigma(x)$ is the SoftMax function, $cos\_sim(x, y)$ stands for the cosine similarity between vector x and y, $emb(w)$ represents the pre-trained word embedding of the token w, and w_t^* is the ground-truth word of the time step t.

Thus, the probability of the ground-truth word in $p^*(w_t)$ is the highest, and the more similar the semantic meaning to the ground truth, the higher the probability of the word.

Adaptive prior-dependent correction (APDC) Additionally, PDC is a token-level objective which is suffer from the issue of generation diversity and and short-sighted (Li et al. 2015), but the superiority of RL is that, it can optimize the evaluation metric at the sequence level to avoid these issues, but PDC may undermine the superiority. Therefore, PDC needs a sequence-level adaptive factor to adjust how much the token-level objective affects sequence-level training. Incorporating such idea, we modify the above-mentioned KL divergence to make it self-regulating:

$$L_{KL}(\theta) = \sum_{t=0}^{T} e^{-\alpha \tilde{A}^{\pi}(s_t, a_t)} KL\left[p^*(w_t)||p_\theta(w_t)\right] \tag{17}$$

Here we use $e^{-\alpha \tilde{A}^{\pi}(s_t, a_t)}$ as the adaptive factor to ensure it is a positive value and monotonically decreasing, where α is a hyper-parameter to adjust the scale of the adaptive factor. The $A^{\pi}(s_t, a_t)$ measures how much a token is better than the others for entire sentence at sequence level. If the $A^{\pi}(s_t, a_t)$ is high, it means the action may also be reasonable at sequence level, and it is less necessary to correct the output distribution, and vice versa. This mechanism can also be interpreted that, the influence of the PDC will be weaken if the action has larger expected cumulative rewards compared to other actions, and vice versa. It helps the algorithm choose to pay how much attention to the token-level distribution accuracy.

The final objective for RL is written as follows:

$$L(\theta) = -L_{RL}(\theta) + \beta L_{KL}(\theta) \tag{18}$$

where β is a hyper-parameter to adjust the scale of the L_{KL} term. We train the model by minimize $L(\theta)$.

Experiments

We evaluate our APDC enhanced RL training method on three common language generation tasks: neural machine translation (NMT), image captioning, and abstractive text summarization (summarization for short).

For the three tasks, we use fastText (Bojanowski et al. 2017) to pre-train the embedding. The pre-training settings are the same as the original paper of fastText. In RL training, we use the BLEU as the reward for NMT, CIDEr for image captioning and ROUGE-L for summarization (as most RL training methods typically do). All the experimental results obtained are based on the optimal settings.

Neural Machine Translation

Neural machine translation (NMT) is an approach to machine translation that uses a deep neural network to predict the likelihood of a sequence of words, modeling the entire sentences in a single integrated model.

Datasets We evaluated our RL training method for NMT on commonly used machine translation datasets: WMT14 (Bojar et al. 2014) English-German (En-De), WMT17 (Ondrej et al. 2017) English-Chinese (En-Zh), and WMT17 Chinese-English (Zh-En). For a fair comparison, we employ the same pre-processing in (Vaswani et al. 2017) for WMT14 En-De dataset, and the same pre-processing in (Wu et al. 2018) for WMT17 En-Zh and Zh-En datasets. We chose the Transformer (Vaswani et al. 2017) as the base model for all methods. For Zh-En and En-Zh translations, we adopt the Transformer big setting. For En-De translation, we adopt the Transformer base setting. These settings are the same as used in the original paper of Transformer.

Experiment configuration We compare our method for NMT with state-of-the-art methods and several variants of our and their methods. Here are the configurations.

- **Transformer**. We use the Transformer (Vaswani et al. 2017) trained with XENT loss as a baseline, which is the method in the original paper and achieves state-of-the-art translation performance in several datasets.
- **MIXER** (Ranzato et al. 2016) uses seq2seq (Bahdanau, Cho, and Bengio 2014) as the NMT model. For a fair comparison, we implement Transformer+MIXER for the three language pairs. We do not implement MIXER enhanced with APDC for NMT, image captioning, and summarization, as the reward is not obtained until the sequence is completed.
- **RL4NMT**. We re-implement RL4NMT (Wu et al. 2018) based on their open-source code.
- **APG+PDC**. Advantage-function-weighted Policy Gradient (APG) is our proposed RL algorithm. PDC is APDC without the adaptive factor. APG+PDC means APG trained with PDC. We will not re-describe these in the following sections since the settings are the same.
- **RL4NMT+PDC**. RL4NMT uses reward shaping instead of estimating a Q-value in each time step, which calculates BLEU score with incomplete sentences $w_{1:t}$ instead of $w_{1:T}$, and then uses BLEU at time t minus BLEU at time $t-1$ as the reward at time t. The reward used in (Wu et al. 2018) is a token-level reward, but our adaptive factor of APDC is a sentence-level expected reward. Therefore we implement RL4NMT enhanced with PDC and RL4NMT+APDC is not applicable here.

Experimental results We compare our method with one MLE training and two RL training methods, which are strong baselines. Table 1 shows the results of comparing APDC with these strong baselines. For the language pair En-De, the result we obtain is very close to those in the original paper of Transformer. For language pairs En-Zh and Zh-En, the results we obtain are very close to those in the original paper of RL4NMT. The results show that APG+APDC outperforms the strong baselines for all language pairs, which validates the advancement of our method.

Image Captioning

Image captioning, also called image description, is a task that aims to automatically generate natural language descriptions according to the content observed in an image.

Datasets We evaluate our proposed method on the MSCOCO dataset (Lin et al. 2014), which is a standard benchmark for image captioning. We follow the split method proposed in (Karpathy and Fei-Fei 2015) for MSCOCO.

Experiment configuration We compare our method for image captioning with several state-of-the-art methods and their variants.

- **Top-down**. Top-down (Anderson et al. 2018) used faster R-CNN (Ren et al. 2015) to detect objects in the image, and a Top-down attention mechanism to dynamically attend to these object features. They also implement Top-down model trained with SCST (Top-down+SCST) and XENT loss (Top-down+MLE). Top-down+MLE is used as the pre-trained model of all RL training methods. In this paper, we follow the same setup as (Anderson et al. 2018) for extracting image features and decoder LSTM for all methods.
- **MIXER**. MIXER uses a convolutional neural network (CNN) pre-trained on the ImageNet to extract image features. For a fair comparison, we implement the Top-down model trained with MIXER, and obtain a better result than MIXER with original settings.

Method	En-De	En-Zh	Zh-En
Transformer+MLE (Vaswani et al. 2017)	27.30	34.12	24.29
MIXER (Ranzato et al. 2016)	27.43	34.38	24.59
RL4NMT (Wu et al. 2018)	27.52	34.46	24.70
APG	27.63	34.54	24.81
APG+PDC	27.70	34.56	24.90
RL4NMT+PDC	27.81	34.62	24.94
APG+APDC	**28.03**	**34.91**	**25.28**

Table 1. BLEU score of NMT for En-De, En-Zh, and Zh-En.

Method	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr
SCST (Att2all) (Rennie et al. 2017)	-	34.2	26.7	55.7	114.0
OTRL (Chen et al. 2020a)	79.3	34.4	26.8	56.2	111.8
Top-Down+MLE (Anderson et al. 2018)	77.2	36.2	27.0	56.4	113.5
Top-Down+SCST (Anderson et al. 2018)	79.8	36.3	27.7	56.9	120.1
Top-Down+MIXER (Ranzato et al. 2016)	78.4	36.2	27.4	56.5	115.6
Top-Down+SCST+APDC	79.9	36.6	27.8	56.9	120.2
APG	80.1	36.4	28.0	56.9	120.2
APG+PDC	80.3	36.7	28.4	57.3	121.2
APG+APDC (METEOR)	80.1	36.5	29.2	57.1	119.7
APG+APDC	**80.8**	**37.9**	**28.9**	**58.1**	**123.6**

Table 2. Performance of image captioning on the MSCOCO Karpathy test split.

- **SCST**. Self-critical sequence training (SCST) (Rennie et al. 2017) is a state-of-the-art RL training method for image captioning. It regards the sequence from the inference algorithm as a baseline in reward to reduce the variance of gradient estimation. To validate the effectiveness of APDC, we also implement SCST+APDC, where the advantage function of adaptive factor for SCST is $r_t - r_{baseline}$ (in SCST, the function of $(r_t - r_{baseline})$ used is similar to the advantage function in our method).
- **OTRL**. Optimal-Transport-Enhanced RL (OTRL) (Chen et al. 2020a) combines RL and Optimal-Transport learning (Chen et al. 2019) and obtains the state-of-the-art performance on MSCOCO dataset. They also use Top-down as the baseline model.

Experimental results beta We compare our method with one MLE training and several RL training methods, which are strong baselines. We report BLEU-1, BLEU-4, CIDEr, ROUGE-L and METEOR scores. The results are summarized in Table 2. It is observed that, under several different setups, our method consistently outperforms all baselines.

Abstractive Text Summarization

Abstractive text summarization condenses a piece of text to a shorter one containing the primary information.

Datasets We use the CNN/Daily Mail (Hermann et al. 2015; Nallapati et al. 2016) dataset to evaluate our proposed method, which is a standard summarization benchmark. We use the same pre-processing as (See, Liu, and Manning 2017), and use the Pointer-Generator networks (See, Liu, and Manning 2017), which is a state-of-the-art summarization method, as the base model for our RL training method.

Experiment configuration We compare our method for summarization with the following methods.

- **Pointer-Generator+Coverage.** Pointer-Generator (See, Liu, and Manning 2017) proposes a hybrid pointer-generator network that can copy words from the source text via pointing, and the coverage mechanism with coverage loss. For summarization, ROUGE-L score is the reward for all RL training methods.
- **MIXER**. For a fair comparison, we implement Pointer-Generator+Coverage model trained with MIXER, and obtain a better result than MIXER with original settings.
- **OTRL**. OTRL also uses Pointer-Generator+Coverage as their baseline, and evaluate their method on CNN/Daily Mail dataset with the same data pre-processing as ours.

Experimental results We compare our method with the baseline and most recent works. The results are summarized in Table 3. We report ROUGE-1, ROUGE-2 and ROUGE-3 scores. APG+APDC achieves better ROUGE scores on CNN/Daily Mail dataset. It indicates that our method can better capture the semantic meaning from the source text.

Analysis

From Figure 2 and Tables 1,2 and 3, it is observed that APG outperforms other RL methods. There are two main reasons: (1) All of the policy gradient methods have a large variance of the gradient estimation, and the advantage function yields almost the lowest possible variance. APG used advantage

Method	ROUGE-1	ROUGE-2	ROUGE-L
Pointer (See, Liu, and Manning 2017)	39.53	17.28	36.38
MIXER (Ranzato et al. 2016)	39.78	17.91	37.15
OTRL (Chen et al. 2020a)	41.40	18.22	38.86
APG	41.51	18.34	38.93
APG+PDC	41.60	18.48	39.02
APG+APDC	**42.73**	**18.81**	**39.85**

Table 3. Results of abstractive text summarization on CNN/Daily Mail dataset. "Pointer" means the method Pointer-Generator+Coverage.

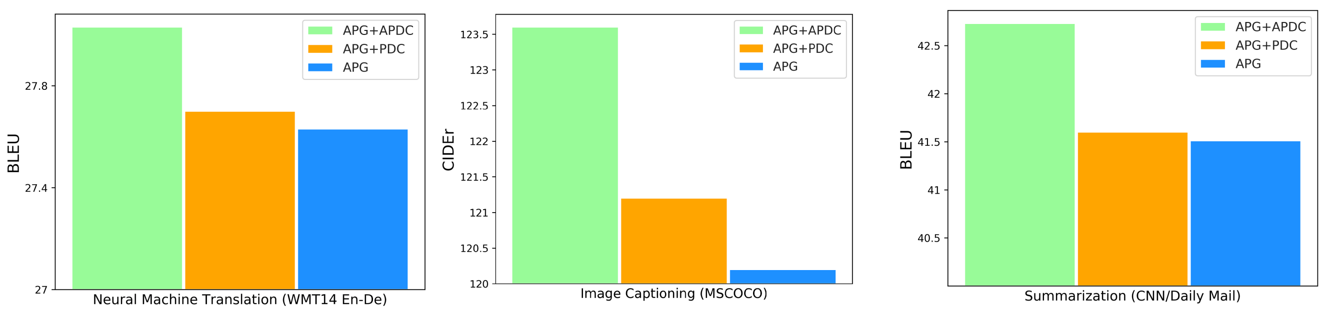

Figure 2: Ablation study on the three tasks.

function instead of $(r - r_{baseline})$ used in SCST, MIXER, RL4NMT and OTRL. It also alleviates the *sparse reward* and *late feedback* issues. (2) MIXER, RL4NMT and OTRL use a parameterized estimator to estimate baseline, which may introduce estimation bias, while in APG, the advantage function is estimated by the non-parametric Q_t and Q_{t-1} which are the expected cumulative rewards estimated by K Monte Carlo rollouts at time step $t-1$.

We have analyzed that BLEU or ROUGE score as the reward in RL will bring *deviation ignorance* problems. To some extent, using the reward of CIDEr or METEOR score can alleviate the issue. But merely optimizing these metrics cannot make full use of the prior distribution to correct the bias of the agent's stochastic policy. These metrics are also inflexible, as most of the works on NMT and summarization adopts BLEU and ROUGE. In the experiments of image captioning, the results show that optimizing CIDEr or METEOR can alleviate the issue of *deviation ignorance*, but using RL enhanced with APDC achieves better performance.

Ablation study To evaluate the effectiveness of different components, we compare the results of applying APG, APG+PDC, and APG+APDC on the three tasks. The results also show that APG+PDC (other RL+PDC) outperform the APG (other RL), since PDC alleviates the *deviation ignorance* issue. However, as Figure 2 shows, the improvement of PDC is not significant. Since PDC is a token-level objective, but the advantage of RL is that the model can be trained at the sequence level, and PDC weakens this advantage. Therefore, PDC needs a sequence-level adaptive factor to adjust how much the token-level objective affects sequence-level training. The results that APG+APDC significantly outperforms other methods also proves the importance of the adaptive mechanism.

We compare our method with one MLE training and several RL training methods, which are strong baselines. We report BLEU-1, BLEU-4, CIDEr, ROUGE-L and METEOR scores. The results are summarized in Table 2. It is observed that, under several different setups, our method consistently outperforms all baselines.

Conclusion

In this work, we leverage reinforcement learning (RL) in natural language generation (NLG) tasks to solve the *exposure bias* and *loss inconsistency* issues of the deep-learning-based methods. We propose a novel technique: *adaptive prior-dependent correction* (APDC) to further address the *deviation ignorance* issue that former RL-based approaches on NLG seldom study. Furthermore, we combine some advantage function estimation techniques which utilize the prior knowledge that the *afterstate* value equals the Q-value at every time step. We utilize *advantage-function-weighted policy gradient* (APG) to work well with APDC. Enhancing APG with APDC can strike a balance between token-level and sequence-level optimization. Our experiments show that, on three tasks, our method consistently outperforms the state-of-the-art approaches on different frequently-used metrics.

Acknowledgments

This work is funded by the National Natural Science Foundation of China (Grand No. 61876181) and Beijing Nova

Program of Science and Technology under Grand No. Z191100001119043 and in part by the Youth Innovation Promotion Association, CAS.

References

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6077–6086.

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A. C.; and Bengio, Y. 2017. An Actor-Critic Algorithm for Sequence Prediction. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5: 135–146.

Bojar, O.; Buck, C.; Federmann, C.; Haddow, B.; Koehn, P.; Leveling, J.; Monz, C.; Pecina, P.; Post, M.; Saint-Amand, H.; et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, 12–58.

Chen, H.; Ding, G.; Zhao, S.; and Han, J. 2018. Temporal-difference learning with sampling baseline for image captioning. AAAI.

Chen, L.; Bai, K.; Tao, C.; Zhang, Y.; Wang, G.; Wang, W.; Henao, R.; and Carin, L. 2020a. Sequence Generation with Optimal-Transport-Enhanced Reinforcement Learning. In *AAAI*, 7512–7520.

Chen, L.; Bai, K.; Tao, C.; Zhang, Y.; Wang, G.; Wang, W.; Henao, R.; and Carin, L. 2020b. Sequence Generation with Optimal-Transport-Enhanced Reinforcement Learning. In *AAAI*.

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 *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

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. *arXiv preprint arXiv:1406.1078* .

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, 1693–1701.

Karpathy, A.; and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3128–3137.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, B. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Li, Z.; Wang, R.; Chen, K.; Utiyama, M.; Sumita, E.; Zhang, Z.; and Zhao, H. 2019. Data-dependent gaussian prior objective for language generation. In *International Conference on Learning Representations*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

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 *European conference on computer vision*, 740–755. Springer.

Liu, S.; Zhu, Z.; Ye, N.; Guadarrama, S.; and Murphy, K. 2017. Improved Image Captioning via Policy Gradient optimization of SPIDEr. *2017 IEEE International Conference on Computer Vision (ICCV)* doi:10.1109/iccv.2017.100. URL http://dx.doi.org/10.1109/ICCV.2017.100.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* .

Nallapati, R.; Zhou, B.; Gulcehre, C.; Xiang, B.; et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Ondrej, B.; Chatterjee, R.; Christian, F.; Yvette, G.; Barry, H.; Matthias, H.; Philipp, K.; Qun, L.; Varvara, L.; Christof, M.; et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Second Conference onMachine Translation*, 169–214. The Association for Computational Linguistics.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2016. Sequence Level Training with Recurrent Neural Networks. In Bengio, Y.; and LeCun, Y., eds., *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. URL http://arxiv.org/abs/1511.06732.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Rennie, S. J.; Marcheret, E.; Mroueh, Y.; Ross, J.; and Goel, V. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7008–7024.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In Bengio, Y.; and LeCun, Y., eds., *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1073–1083.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

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 *Advances in neural information processing systems*, 5998–6008.

Vedantam, R.; Lawrence Zitnick, C.; and Parikh, D. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4566–4575.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575(7782): 350–354.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3156–3164.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.

Williams, R. J.; and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2): 270–280.

Wu, L.; Tian, F.; Qin, T.; Lai, J.; and Liu, T.-Y. 2018. A Study of Reinforcement Learning for Neural Machine Translation. In *EMNLP*.