ARAML: A Stable Adversarial Training Framework for Text Generation

Pei Ke, Fei Huang*, Minlie Huang, Xiaoyan Zhu
Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems
Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
kepei1106@outlook.com, f-huang18@mails.tsinghua.edu.cn
aihuang@tsinghua.edu.cn, zxy-dcs@tsinghua.edu.cn

Abstract

Most of the existing generative adversarial networks (GAN) for text generation suffer from the instability of reinforcement learning training algorithms such as policy gradient, leading to unstable performance. To tackle this problem, we propose a novel framework called Adversarial Reward Augmented Maximum Likelihood (ARAML). During adversarial training, the discriminator assigns rewards to samples which are acquired from a stationary distribution near the data rather than the generator’s distribution. The generator is optimized with maximum likelihood estimation augmented by the discriminator’s rewards instead of policy gradient. Experiments show that our model can outperform state-of-the-art text GANs with a more stable training process.

1 Introduction

Natural text generation, as a key task in NLP, has been advanced substantially thanks to the flourish of neural models (Bengio et al., 2003; Mikolov et al., 2010). Typical frameworks such as sequence-to-sequence (seq2seq) have been applied to various generation tasks, including machine translation (Sutskever et al., 2014) and dialogue generation (Vinyals and Le, 2015). The standard paradigm to train such neural models is maximum likelihood estimation (MLE), which maximizes the log-likelihood of observing each word in the text given the ground-truth proceeding context (Graves, 2013).

Although widely used, MLE suffers from the exposure bias problem (Bengio et al., 2015; Ranzato et al., 2016): during test, the model sequentially predicts the next word conditioned on its previous generated words while during training conditioned on ground-truth words. To tackle this problem, generative adversarial networks (GAN) with reinforcement learning (RL) training approaches have been introduced to text generation tasks (Yu et al., 2017; Che et al., 2017; Lin et al., 2017; Fedus et al., 2018; Guo et al., 2018; Shi et al., 2018; Xu et al., 2018), where the discriminator is trained to distinguish real and generated text samples to provide reward signals for the generator, and the generator is optimized via policy gradient (Yu et al., 2017).

However, recent studies have shown that potential issues of training GANs on discrete data are more severe than exposure bias (Semeniuta1 et al., 2018; Caccia et al., 2018). One of the fundamental issues when generating discrete text samples with GANs is training instability. Updating the generator with policy gradient always leads to an unstable training process because it’s difficult for the generator to derive positive and stable reward signals from the discriminator even with careful pre-training (Che et al., 2017). As a result, the generator gets lost due to the high variance of reward signals and the training process may finally collapse (Li et al., 2017).

In this paper, we propose a novel adversarial training framework called Adversarial Reward Augmented Maximum Likelihood (ARAML) to deal with the instability issue of training GANs for text generation. At each iteration of adversarial training, we first train the discriminator to assign higher rewards to real data than to generated samples. Then, inspired by reward augmented maximum likelihood (RAML) (Norouzi et al., 2016), the generator is updated on the samples acquired from a stationary distribution with maximum likelihood estimation (MLE), weighted by the discriminator’s rewards. This stationary distribution is designed to guarantee that training samples are surrounding the real data, thus the exploration space of our generator is indeed restricted by the
MLE training objective, resulting in more stable training. Compared to other text GANs with RL training techniques, our framework acquires samples from the stationary distribution rather than the generator’s distribution, and uses RAML training paradigm to optimize the generator instead of policy gradient. Our contributions are mainly as follows:

- We analyze the fundamental issue of current GANs for text generation from the perspectives of training instability.
- We propose a novel framework called Adversarial Reward Augmented Maximum Likelihood (ARAML), which incorporates stable RAML training into adversarial training paradigm. Experimental results on three text generation tasks show the effectiveness of our method.

2 Related Work

Recently, text generation has been widely studied with neural models trained with maximum likelihood estimation (Graves, 2013). However, MLE tends to generate universal text (Li et al., 2016). Various methods have been proposed to enhance the generation quality by refining the objective function (Li et al., 2016; Mou et al., 2016) or modifying the generation distribution with external information like topic (Xing et al., 2017), sentence type (Ke et al., 2018), emotion (Zhou et al., 2018a) and knowledge (Zhou et al., 2018b).

As mentioned above, MLE suffers from the exposure bias problem (Bengio et al., 2015; Ranzato et al., 2016). Thus, reinforcement learning has been introduced to text generation tasks such as policy gradient (Ranzato et al., 2016) and actor-critic (Bahdanau et al., 2017). (Norouzi et al., 2016) proposed an efficient and stable approach called Reward Augmented Maximum Likelihood (RAML), which connects the log-likelihood and expected rewards to incorporate MLE training objective into RL framework.

Since some text generation tasks have no explicit metrics to be directly optimized, adversarial training has been applied to generating discrete text samples with a discriminator to learn a proper reward. For instance, SeqGAN (Yu et al., 2017) devised a discriminator to distinguish the real data and generated samples, and a generator to maximize the reward from the discriminator via policy gradient. Other variants of GANs have been proposed to improve the generator or the discriminator. To improve the generator, MaliGAN (Che et al., 2017) developed a normalized maximum likelihood optimization target for the generator to stably model the discrete sequences. LeakGAN (Guo et al., 2018) guided the generator with reward signals leaked from the discriminator at all generation steps to deal with long text generation task. MaskGAN (Fedus et al., 2018) employed an actor-critic architecture to make the generator fill in missing text conditioned on the surrounding context, which is expected to mitigate the problem of mode collapse. As for the discriminator, RankGAN (Lin et al., 2017) replaced traditional discriminator with a ranker to learn the relative ranking information between the real texts and generated ones. Inverse reinforcement learning (Shi et al., 2018) used a trainable reward approximator as the discriminator to provide dense reward signals at each generation step. DPGAN (Xu et al., 2018) introduced a language model based discriminator and regarded cross-entropy as rewards to promote the diversity of generation results.

The most similar works to our model are RAML (Norouzi et al., 2016) and MaliGAN (Che et al., 2017): 1) Compared with RAML, our model adds a discriminator to learn the reward signals instead of choosing existing metrics as rewards. We believe that our model can adapt to various text generation tasks, particularly those without explicit evaluation metrics. 2) Unlike MaliGAN, we acquire samples from a fixed distribution near the real data rather than the generator’s distribution, which is expected to make the training process more stable.

3 Model

3.1 Task Definition and Model Overview

Text generation can be formulated as follows: given the real data distribution $P_{data}(X)$, the task is to train a generative model $G_{\theta}$ where $P_{G_{\theta}}(X)$ can fit $P_{data}(X)$ well. In this formulation, $X = x_1x_2\cdots x_m$ and $x_t (1 \leq t \leq m)$ denotes a word in the vocabulary $\mathcal{V}$.

Figure 1 shows the overview of our model ARAML. This adversarial training framework consists of two phases: 1) The discriminator is trained to assign higher rewards to real data than to generated data. 2) The generator is trained on the samples acquired from a stationary distribu-
tion with reward augmented MLE training objective. This training paradigm of the generator indeed constrains the search space with the MLE training objective, which alleviates the issue of unstable training.

### 3.2 Discriminator

The discriminator $D_\phi$ aims to distinguish real data and generated data like other GANs. Inspired by Least-Square GAN (Mao et al., 2017), we devise the loss function as follows:

$$
L_{D_\phi} = \frac{1}{2} E_{X \sim P_{data}(X)} [(D_\phi(X) - 1)^2] + \frac{1}{2} E_{X \sim P_{G_\theta}(X)} [(D_\phi(X))^2]
$$

This loss function forces the discriminator to assign higher rewards to real data than to generated data, so the discriminator can learn to provide more proper rewards as the training proceeds.

### 3.3 Generator

The training objective of our generator $G_\theta$ is derived from the objective of other discrete GANs with RL training method:

$$
L_{RL,\theta} = -E_{X \sim P_{data}(X)}[r_\phi(X)] - \tau \mathbb{H}(P_{G_\theta}(X))
$$

where $r_\phi(X)$ denotes the rewards from the discriminator $D_\phi$ and the entropy regularized term $\mathbb{H}(P_{G_\theta}(X))$ encourages $G_\theta$ to generate diverse text samples. $\tau$ is a temperature hyper-parameter to balance these two terms.

As mentioned above, discrete GANs suffer from the instability issue due to policy gradient, thus they are consequently difficult to train. Inspired by RAML (Norouzi et al., 2016), we introduce an exponential payoff distribution $Q_\phi(X)$ to connect RL loss with RAML loss:

$$
Q_\phi(X) = \frac{1}{Z} \exp(r_\phi(X)/\tau) \quad (3)
$$

where $Z = \sum_X \exp(r_\phi(X)/\tau)$. Thus, we can rewrite $L_{RL,\theta}$ with $P_{G_\theta}(X)$ and $Q_\phi(X)$ as follows:

$$
L_{RL,\theta} = \tau KL(P_{G_\theta}(X)||Q_\phi(X)) + constant \quad (4)
$$

Following RAML, we remove the constant term and optimize the KL divergence in the opposite direction:

$$
L_{RAML,\theta} = KL(Q_\phi(X)||P_{G_\theta}(X)) = -E_{X \sim Q_\phi(X)}[\log P_{G_\theta}(X)] - \mathbb{H}(Q_\phi(X)) = -E_{X \sim Q_\phi(X)}[\log P_{G_\theta}(X)] + constant \quad (5)
$$

where $\mathbb{H}(Q_\phi(X))$ is a constant in the training phase of the generator. It has been proved that $L_{RL,\theta}$ and $L_{RAML,\theta}$ are equivalent up to their first order Taylor approximations, and they have the same global optimum (Norouzi et al., 2016). $L_{RAML,\theta}$ can be trained in a MLE-like fashion but sampling from the distribution $Q_\phi(X)$ is intractable in the adversarial setting, because $Q_\phi(X)$ varies with the discriminator $D_\phi$. Thus, we introduce importance sampling to separate sampling process from $D_\phi$ and obtain the final loss function:

$$
L_{G_\theta} = -E_{X \sim P_{s}(X)}[W_\phi(X) \log P_{G_\theta}(X)] \quad (6)
$$
where \( P_s(X) \) denotes a stationary distribution and \( W_\phi(X) \propto Q_\phi(X) / P_s(X) \). To optimize this loss function, we first construct the fixed distribution \( P_s(X) \) to get samples, and devise the proper reward function \( r_\phi(X) \) to train the generator in a stable and effective way.

3.3.1 Sampling

We construct the distribution \( P_s \) based on \( P_{data} \):

\[
P_s(X) = \mathbb{E}_{X \sim P_{data}}[P_s(X|X)]
\]  

(7)

In this way, \( P_s(X_s|X) \) can be designed to guarantee that \( P_s(X) \) is near \( P_{data}(X) \), leading to a more stable training process. To obtain a new sample \( X_s \) from a real data sample \( X \), we can design three steps which contain sampling an edit distance \( d \), the positions \( \{p_1, p_2, \cdots, p_d\} \) for substitution and the new words \( \{w_1, w_2, \cdots, w_d\} \) filled into the corresponding positions. Thus, \( P_s(X_s|X) \) can be decomposed into three terms:

\[
P_s(X_s|X) = P(d, p, w|X) = P(d|X)P(p|X,d)P(w|X,d,p)
\]  

(8)

The first step is to sample an edit distance based on a real data sample \( X \), where \( X = x_1x_2 \cdots x_m \) is a sequence of length \( m \). The number of sentences which have the edit distance \( e \) to some input sentence can be computed approximately as below:

\[
c(e, m) = \binom{m}{e} \cdot (|\mathcal{V}| - 1)^e
\]  

(9)

where \( c(e, m) \) denotes the number of sentences which have an edit distance \( e \in \{0, 1, \cdots, m\} \) to a sentence of length \( m \), and \( |\mathcal{V}| \) indicates the size of vocabulary. We then follow (Norouzi et al., 2016) to re-scale the counts by \( \exp\{-e/\tau\} \) and do normalization, so that we can sample an edit distance \( d^* \) from:

\[
P(d = d^*|X) = \frac{\exp\{-d^*/\tau\}c(d^*, m)}{\sum_{e=0}^{m} \exp\{-e/\tau\}c(e, m)}
\]  

(10)

where \( \tau \), as a temperature hyper-parameter, restricts the search space surrounding the original sentence. Larger \( \tau \) brings more samples with long edit distances.

The next step is to select positions for substitution based on the sampled edit distance \( d^* \). Intuitively, we can randomly choose \( d^* \) distinct positions in \( X \) to be replaced by new words. The probability of choosing the position \( p^* \) is calculated as follows:

\[
P(p = p^*|X, d = d^*) = \frac{d^*}{m}
\]  

(11)

Following this sampling strategy, we can obtain the position set \( \{p_1, p_2, \cdots, p_d\} \). This strategy approximately guarantees that the edit distance between a new sentence and the original sentence is \( d^* \).

At the final step, our model determines new words for substitution at each sampled position \( p_j (j = 1, 2, \cdots, d^*) \). We can formulate this sampling process from the original sequence \( X \) to a new sample \( X_s \) as a sequential transition \( X = X_0 \rightarrow X_1 \rightarrow \cdots \rightarrow X_{d^*} = X_s \). At each step from \( X_{j-1} \) to \( X_j (j = 1, \cdots, d^*) \), we first sample a new word \( w_j \) from the distribution \( P(w|X_{j-1}, p = p_j) \), then replace the old word at position \( p_j \) of \( X_{j-1} \) to obtain \( X_j \). The whole sampling process can be decomposed as follows:

\[
P(w|X, d = d^*, p = \{p_1, p_2, \cdots, p_{d^*}\}) = \prod_{j=1}^{d^*} P(w_j|X_{j-1}, p = p_j)
\]  

(12)

There are two common sampling strategies to model \( P(w|X_{j-1}, p = p_j) \), i.e. random sampling and constrained sampling. Random sampling strategy samples a new word \( w_j \) according to the uniform distribution over the vocabulary \( \mathcal{V} \) (Norouzi et al., 2016), while constrained sampling strategy samples \( w_j \) to maximize the language model score of the target sentence \( X_j \) (Su et al., 2018; Miao et al., 2019). Here, we adopt constrained sampling in our model and compare the performances of two strategies in the experiment.

3.3.2 Training

We devise the reward function \( r_\phi(X) \) according to the discriminator’s output \( D_\phi(X) \) and the stationary distribution \( P_s(X) \):

\[
r_\phi(X) = \tau \cdot [\log P_s(X) + D_\phi(X)]
\]  

(13)

Intuitively, this reward function encourages the generator to generate sentences with large sampling probability and high rewards from the discriminator. Thus, the weight of samples \( W_\phi(X) \) can be calculated as follows:

\[
W_\phi(X) \propto \frac{Q_\phi(X)}{P_s(X)} \propto \exp \{D_\phi(X)\}
\]  

(14)
So far, we can successfully optimize the generator’s loss \( L_{G_\theta} \) via Equation 6. This training paradigm makes our generator avoid possible variances caused by policy gradient and get more stable reward signals from the discriminator, because our generator is restricted to explore the training samples near the real data.

**Algorithm 1** Adversarial Reward Augmented Maximum Likelihood

**Require:**
- Total adversarial training iterations: \( N_{\text{iters}} \)
- Steps of training generator: \( G_{\text{steps}} \)
- Steps of training discriminator: \( D_{\text{steps}} \)

1. Pre-train the generator \( G_\theta \) with MLE loss
2. Generate samples from \( P_{G_\theta} \)
3. Pre-train the discriminator \( D_\phi \) via Eq.(1)
4. Construct \( P_s(X) \) via Eq.(7) - Eq.(12)
5. for each \( s = 1, 2, ..., N_{\text{iters}} \) do
   6. for each \( j = 1, 2, ..., G_{\text{steps}} \) do
      7. Update \( G_\theta \) via Eq.(6)
     8. end for
   9. for each \( k = 1, 2, ..., D_{\text{steps}} \) do
      10. Update \( D_\phi \) via Eq.(1)
11. end for
12. end for

3.4 Extension to Conditional Text Generation

We have shown our adversarial training framework for text generation tasks without an input. Actually, it can also be extended to conditional text generation tasks like dialogue generation. Given the data distribution \( P_{\text{data}}(C, X) \) where \( C, X \) denote contexts and responses respectively, the objective function of ARAML’s generator can be modified as below:

\[
L_{G_\theta} = -E_{(C, X) \sim P_{\text{data}}(C, X)} \left[ E_{X_s \sim P_s(X_s|C, X)} \left[ W_\phi(C, X_s) \log P_{G_\theta}(X_s|C) \right] \right]
\]

(15)

where \( W_\phi(C, X_s) \propto \exp\{D_\phi(C, X_s)\} \) and \( D_\phi(C, X_s) \) is trained to distinguish whether \( X_s \) is the true response to \( C \).

3.5 Comparison with RAML and MaliGAN

The most similar works to our framework are RAML (Norouzi et al., 2016) and MaliGAN (Che et al., 2017). The main difference among them is the training objective of their generators. We have shown different objective functions in Table 1. For comparison, we use the form with no input for all the three models.

Our model is greatly inspired by RAML, which gets samples from a non-parametric distribution \( Q(X) \) constructed based on a specific reward. Compared to RAML, our reward comes from a learnable discriminator which varies as the adversarial training proceeds rather than a specific reward function. This difference equips our framework with the ability to adapt to the text generation tasks with no explicit evaluation metrics as rewards.

Our model is also similar to MaliGAN, which gets samples from the generator’s distribution. In MaliGAN’s training objective, \( G_\theta' \) also indicates the generator’s distribution but it’s used in the sampling phase and fixed at each optimization step. The weight of samples \( W_s(X) \propto \frac{D_\phi(X)}{1-D_\phi(X)} \). Different from our model, MaliGAN acquires samples from the generator’s distribution \( P_{G_\theta'} \), which usually brings samples with low rewards even with careful pre-training for the generator, leading to training instability. Instead, our framework gets samples from a stationary distribution \( P_s \) around real data, thus our training process is more stable.

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Objective of Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAML</td>
<td>( L_{G_\theta} = -E_{X \sim Q(X)} \log P_{G_\theta}(X) )</td>
</tr>
<tr>
<td>MaliGAN</td>
<td>( L_{G_\theta} = -E_{X \sim P_{G_\theta}(X)} \log P_{G_\theta}(X) )</td>
</tr>
<tr>
<td>ARAML</td>
<td>( L_{G_\theta} = -E_{X \sim P(X)} \log P_{G_\theta}(X) )</td>
</tr>
</tbody>
</table>

Table 1: Training objectives of generators for RAML, MaliGAN and ARAML.

4 Experiment

4.1 Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Amount(Train/Test)</th>
<th>Vocabulary</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCO</td>
<td>80,000/5,000</td>
<td>4,839</td>
<td>12.8</td>
</tr>
<tr>
<td>EMNLP2017</td>
<td>49,996/10,000</td>
<td>5,721</td>
<td>27.8</td>
</tr>
<tr>
<td>WeiboDial</td>
<td>100,000/5,000</td>
<td>7,998</td>
<td>7.3/10.8</td>
</tr>
</tbody>
</table>

Table 2: Statistics of COCO, EMNLP2017 WMT and WeiboDial. The average lengths 7.3/10.8 of WeiboDial indicate the lengths of posts and responses, respectively.

We evaluated ARAML on three datasets: COCO image caption dataset (Chen et al., 2015), EMNLP2017 WMT dataset\(^1\) and WeiboDial single-turn dialogue dataset (Qian et al., 2017).

\(^1\)http://statmt.org/wmt17/translation-task.html
2018). COCO and EMNLP2017 WMT are the common benchmarks with no input to evaluate the performance of discrete GANs, and we followed the existing works to preprocess these datasets (Shi et al., 2018; Guo et al., 2018). WeiboDial, as a dialogue dataset, was applied to test the performance of our model with input trigger. We simply removed post-response pairs containing low-frequency words and randomly selected a subset for our training/test set. The statistics of three datasets are presented in Table 2.

4.2 Baselines

We compared our model with MLE, RL and GAN baselines. Since COCO and EMNLP2017 WMT don’t have input while WeiboDial regards posts as input, we chose the following baselines respectively:

**MLE**: a RNN model trained with MLE objective (Graves, 2013). Its extension, Seq2Seq, can work on the dialogue dataset (Sutskever et al., 2014).

**SeqGAN**: The first text GAN model that updates the generator with policy gradient based on the rewards from the discriminator (Yu et al., 2017).

**LeakGAN**: A variant of SeqGAN that provides rewards based on the leaked information of the discriminator for the generator (Guo et al., 2018).

**MaliGAN**: A variant of SeqGAN that optimizes the generator with a normalized maximum likelihood objective (Che et al., 2017).

**IRL**: This inverse reinforcement learning method replaces the discriminator with a reward approximator to provide dense rewards (Shi et al., 2018).

**RAML**: A RL approach to incorporate MLE objective into RL training framework, which regards BLEU as rewards (Norouzi et al., 2016).

**DialogGAN**: An extension of SeqGAN tuned to dialogue generation task with MLE objective added to the adversarial objective (Li et al., 2017).

**DPGAN**: A variant of DialogGAN which uses a language model based discriminator and regards cross-entropy as rewards (Xu et al., 2018).

Note that MLE, SeqGAN, LeakGAN, MaliGAN and IRL are the baselines on COCO and EMNLP2017 WMT, while MLE, RAML, DialogGAN, and DPGAN on WeiboDial. The original codes are used to test the baselines.

4.3 Implementation Details

The implementation details of our model are shown in Table 3. For COCO / EMNLP2017, the generator is a LSTM unit (Hochreiter and Schmidhuber, 1997) with 128 cells, and the discriminator is implemented based on (Yu et al., 2017). For WeiboDial, the generator is an encoder-decoder structure with attention mechanism, where both the encoder and the decoder consist of a two-layer GRU (Cho et al., 2014) with 128 cells. The discriminator is implemented based on (Tao et al., 2018). The language model used in the constrained sampling of ARAML is implemented in the same setting as the generators, and is pretrained on the training set of each dataset. The codes and the datasets are available at https://github.com/kepei1106/ARAML.

As for the details of the baselines, the generators of all the baselines except LeakGAN are the same as ours. Note that the generator of LeakGAN consists of a hierarchical LSTM unit, thus we followed the implementation in the original paper. In terms of the differences, the discriminators of GAN baselines are implemented based on the original papers. Other hyper-parameters of baselines including batch size, learning rate, and pre-training epochs, were set based on the original codes, because the convergence of baselines is sensitive to these hyper-parameters.

4.4 Language Generation on COCO and EMNLP2017 WMT

We adopted forward/reverse perplexity (Zhao et al., 2018) and Self-BLEU (Zhu et al., 2018) to evaluate the quality of generated texts. Forward perplexity (PPL-F) indicates the perplexity on the generated data provided by a language model trained on real data to measure the fluency of generated samples. Reverse perplexity (PPL-R) switches the roles of generated data and real data.
to reflect the discrepancy between the generated distribution and the data distribution. Self-BLEU (S-BLEU) regards each sentence in the generated collection as hypothesis and the others as reference to obtain BLEU scores, which evaluates the diversity of generated results.

Results are shown in Table 4. LeakGAN performs best on forward perplexity because it can generate more fluent samples. As for reverse perplexity, our model ARAML beats other baselines, showing that our model can fit the data distribution better. Other GANs, particularly LeakGAN, obtain high reverse perplexity due to mode collapse (Shi et al., 2018), thus they only capture limited fluent expressions, resulting in large discrepancy between the generated distribution and data distribution. ARAML also outperforms the baselines in terms of Self-BLEU, indicating that our model doesn’t fall into mode collapse with the help of the MLE training objective and has the ability to generate more diverse sentences.

We also provide standard deviation of each metric in Table 4, reflecting the stability of each model’s performance. Our model ARAML nearly achieves the smallest standard deviation in all the metrics, indicating that our framework outperforms policy gradient in the stability of adversarial training.

### 4.5 Dialogue Generation on WeiboDial

Dialogue evaluation is an open problem and existing works have found that automatic metrics have low correlation to human evaluation (Liu et al., 2016; Novikova et al., 2017; Chaganty et al., 2018). Thus, we resorted to manual evaluation to assess the generation quality on WeiboDial. We randomly sampled 200 posts from the test set and collected the generated results from all the models. For each pair of responses (one from ARAML and the other from a baseline, given the same input post), five annotators were hired to label which response is better (i.e., win, lose or tie) in terms of **grammaticality** (whether a response itself is grammatical and logical) and **relevance** (whether a response is appropriate and relevant to the post). The two metrics were evaluated independently.

The evaluation results are shown in Table 5. To measure the inter-annotator agreement, we calculated Fleiss’ kappa (Fleiss, 1971) for each pairwise comparison where results show moderate agreement ($0.4 \leq \kappa \leq 0.6$). We also conducted sign test to check the significance of the differences.

As shown in Table 5, ARAML performs significantly better than other baselines in all the cases. This result indicates that the samples surrounding true responses provide stable rewards for the generator, and stable RAML training paradigm significantly enhances the performance in both metrics.

#### 4.6 Further Analysis on Stability

![Figure 2: PPL-F/PPL-R curves of ARAML, SeqGAN, LeakGAN, MaliGAN and IRL in the training process. The shade area indicates the standard deviation at each data point. The dotted vertical lines separate pre-training and adversarial training phases (50 for ARAML, IRL and MaliGAN, 80 for SeqGAN and LeakGAN).](image-url)

To verify the training stability, we conducted experiments on COCO many times and chose the best 5 trials for SeqGAN, LeakGAN, IRL, MaliGAN and ARAML, respectively. Then, we presented the forward/reverse perplexity in the train-
Table 5: Human evaluation on WeiboDial. The scores represent the percentages of Win, Lose or Tie when our model is compared with a baseline. κ denotes Fleiss’ kappa (all are moderate agreement). The scores marked with * mean p-value < 0.05 and ** indicates p-value < 0.01 in sign test.

<table>
<thead>
<tr>
<th>Model</th>
<th>Grammaticality</th>
<th>Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Win (%)</td>
<td>Lose (%)</td>
</tr>
<tr>
<td>ARAML vs. MLE</td>
<td>56.5**</td>
<td>36.5</td>
</tr>
<tr>
<td>ARAML vs. RAML</td>
<td>54.5**</td>
<td>37.5</td>
</tr>
<tr>
<td>ARAML vs. DialogGAN</td>
<td>75.5**</td>
<td>13.5</td>
</tr>
<tr>
<td>ARAML vs. DPGAN</td>
<td>57.5**</td>
<td>36.0</td>
</tr>
</tbody>
</table>

Table 6: PPL-F, PPL-R and S-BLEU of ARAML with random sampling (ARAML-R) and constrained sampling (ARAML-C) on COCO.

Table 6 shows the results. It’s obvious that random sampling hurts the model performance except Self-BLEU-1, because it indeed allows low-quality samples available to the generator. Exploring these samples degrades the quality and diversity of generated results. Despite the worse performance on automatic metrics, random sampling doesn’t affect the training stability of our framework. The standard deviation of ARAML-R is still smaller than other GAN baselines.

4.8 Case Study

Table 7 presents the examples generated by the models on COCO. We can find that other baselines suffer from grammatical errors (e.g. “in front of flying her kite” from MLE), repetitive expressions...
Table 7: Examples of generated sentences on COCO. Grammatical errors are in red, while blue text represents repetitive expressions and green part indicates incoherent statements.

<table>
<thead>
<tr>
<th>Model</th>
<th>Generated Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLE</td>
<td>A little girl sitting on a beach in front of flying her kite at the beach. A little boy standing in a room next to a desk.</td>
</tr>
<tr>
<td>SeqGAN</td>
<td>A man sitting on a bench with snowboard in the background. A brown gray cat is in the corner of a street.</td>
</tr>
<tr>
<td>LeakGAN</td>
<td>A person that is holding something while another kid is standing in the water. A room with a television, mantle, and a chair.</td>
</tr>
<tr>
<td>MaliGAN</td>
<td>A man with a shirt on holding one large pink giant and white kite. A couple and vases are outside on the bed.</td>
</tr>
<tr>
<td>IRL</td>
<td>A group of people wearing helmet sitting down on a cell phone. A group of people sitting in the middle of tracks.</td>
</tr>
<tr>
<td>ARAML</td>
<td>A man is wearing a hat and holding a toothbrush as he stands on the grass of a field. A boy reading a book on a sofa in a room.</td>
</tr>
</tbody>
</table>

Table 8: Examples of generated responses on WeiboDial.

Table 8 shows the generated examples on WeiboDial. It’s obvious that other baselines don’t capture the topic word “late” in the post, thus generate irrelevant responses. ARAML can provide a response that is grammatical and closely relevant to the post.

5 Conclusion

We propose a novel adversarial training framework to deal with the instability problem of current GANs for text generation. To address the instability issue caused by policy gradient, we incorporate RAML into the adversarial training paradigm to make our generator acquire stable rewards. Experiments show that our model performs better than several state-of-the-art GAN baselines with lower training variance, yet producing better performance on three text generation tasks.

Acknowledgments

This work was supported by the National Science Foundation of China (Grant No. 61936010/61876096) and the National Key R&D Program of China (Grant No. 2018YFC0830200). We would like to thank THUNUS NExT Joint-Lab for the support.

References


