Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Toward Diverse Text Generation with Inverse Reinforcement Learning

Zhan Shi, Xinchi Chen, Xipeng Qiu*, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University

Abstract

Text generation is a crucial task in NLP. Recently, several adversarial generative models have been proposed to improve the exposure bias problem in text generation. Though these models gain great success, they still suffer from the problems of reward sparsity and mode collapse. In order to address these two problems, in this paper, we employ inverse reinforcement learning (IRL) for text generation. Specifically, the IRL framework learns a reward function on training data, and then an optimal policy to maximum the expected total reward. Similar to the adversarial models, the reward and policy function in IRL are optimized alternately. Our method has two advantages: (1) the reward function can produce more dense reward signals. (2) the generation policy, trained by "entropy regularized" policy gradient, encourages to generate more diversified texts. Experiment results demonstrate that our proposed method can generate higher quality texts than the previous methods.

1 Introduction

Text generation is one of the most attractive problems in NLP community. It has been widely used in machine translation, image captioning, text summarization and dialogue systems.

Currently, most of the existing methods [Graves, 2013] adopt auto-regressive models to predict the next words based on the historical predictions. Benefiting from the strong ability of deep neural models, such as long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997], these auto-regressive models can achieve excellent performance. However, they suffer from the so-called *exposure bias* issue [Bengio *et al.*, 2015] due to the discrepancy distribution of histories between the training and inference stage. In training stage, the model predicts the next word according to ground-truth histories from the data distribution rather than its own historical predictions from the model distribution.

Recently, some methods have been proposed to alleviate this problem, such as scheduled sampling [Bengio *et al.*, 2015], Gibbs sampling [Su *et al.*, 2018] and adversarial models, including SeqGAN [Yu *et al.*, 2017], RankGAN [Lin *et al.*, 2017], MaliGAN [Che *et al.*, 2017] and LeakGAN [Guo *et al.*, 2017]. Following the framework of generative adversarial networks (GAN) [Goodfellow *et al.*, 2014], the adversarial text generation models use a discriminator to judge whether a given text is real or not. Then a generator is learned to maximize the reward signal provided by the discriminator via reinforcement learning (RL). Since the generator always generates a entire text sequence, these adversarial models can avoid the problem of exposure bias.

Inspired of their success, there are still two challenges in the adversarial model.

The first problem is *reward sparsity*. The adversarial model depends on the ability of the discriminator, therefore we wish the discriminator always correctly discriminates the real texts from the "generated" ones. Instead, a perfect discriminator increases the training difficulty due to the sparsity of the reward signals. There are two kinds of work to address this issue. The first one is to improve the signal from the discriminator. RankGAN [Lin *et al.*, 2017] uses a ranker to take place of the discriminator, which can learn the relative ranking information between the generated and the real texts in the adversarial framework. MaliGAN [Che *et al.*, 2017] develops normalized maximum likelihood optimization target to alleviate the reward instability problem. The second one is to decompose the discrete reward signal into various sub-signals. LeakGAN [Guo *et al.*, 2017] takes a hierarchical generator, and in each step, generates a word using leaked information from the discriminator.

The second problem is the *mode collapse*. The adversarial model tends to learn limited patterns because of mode collapse. One kind of methods, such as TextGAN [Zhang *et al.*, 2017], uses feature matching [Salimans *et al.*, 2016; Metz *et al.*, 2016] to alleviate this problem, it is still hard to train due to the intrinsic nature of GAN. Another kind of methods [Bayer and Osendorfer, 2014; Chung *et al.*, 2015; Serban *et al.*, 2017; Wang *et al.*, 2017] introduces latent random variables to model the variability of the generated sequences.

To tackle these two challenges, we propose a new method to generate diverse text via inverse reinforcement learning (IRL) [Ziebart *et al.*, 2008]. Typically, the text generation can be regarded as an IRL problem. Each text in the training

*Corresponding Author, xpqiu@fudan.edu.cn

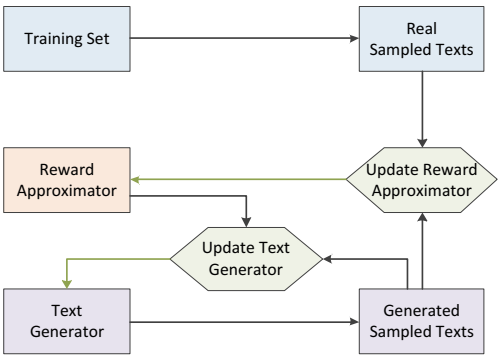

Figure 1: IRL framework for text generation.

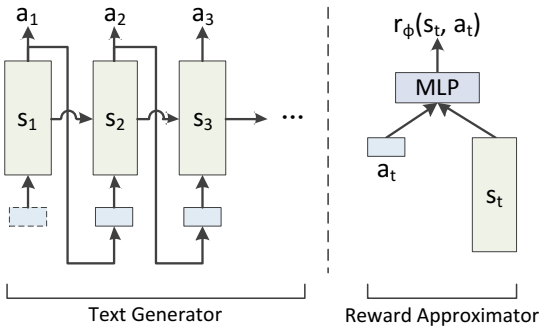

Figure 2: Illustration of text generator and reward approximator.

data is generated by some experts with an unknown reward function. There are two alternately steps in IRL framework. Firstly, a reward function is learned to explain the expert behavior. Secondly, a generation policy is learned to maximize the expected total rewards. The reward function aims to increase the rewards of the real texts in training set and decrease the rewards of the generated texts. Intuitively, the reward function plays the similar role as the discriminator in SeqGAN. Unlike SeqGAN, the reward function is an instant reward of each step and action, thereby providing more dense reward signals. The generation policy generates text sequence by sampling one word at a time. The optimized policy be learned by "entropy regularized" policy gradient [Finn *et al.*, 2016], which intrinsically leads to a more diversified text generator.

The contributions of this paper are summarized as follows.

- We regard text generation as an IRL problem, which is a new perspective on this task.
- Following the maximum entropy IRL [Ziebart *et al.*, 2008], our method can improve the problems of reward sparsity and mode collapse.
- To better evaluate the quality of the generated texts, we propose three new metrics based on BLEU score, which is very similar to precision, recall and F_1 in traditional machine learning task.

2 Text Generation via Inverse Reinforcement Learning

Text generation is to generate a text sequence $x_{1:T} = x_1, x_2, \cdots, x_T$ with a parameterized auto-regressive probabilistic model $q_\theta(x)$, where x_t is a word in a given vocabulary $\mathcal{V}$. The generation model $q_\theta(x)$ is learned from a given dataset $\{x^{(n)}\}_{n=1}^N$ with an underlying generating distribution p_{data}.

In this paper, we formulate text generation as inverse reinforcement learning (IRL) problem. Firstly, the process of text generation can be regarded as Markov decision process (MDP). In each timestep t, the model generates x_t according a policy $\pi_\theta(a_t|s_t)$, where s_t is the current state of the previous prediction $x_{1:t}$ and a_t is the action to select the next word x_{t+1}. A text sequence $x_{1:T} = x_1, x_2, \cdots, x_T$ can be formulated by a trajectory of MDP $\tau = \{s_1, a_1, s_2, a_2 ..., s_T, a_T\}$.

Therefore, the probability of $x_{1:T}$ is

$$q_\theta(x_{1:T}) = q_\theta(\tau) = \prod_{t=1}^{T-1} \pi_\theta(a_t = x_{t+1}|s_t = x_{1:t}), \quad (1)$$

where the state transition $p(s_{t+1} = x_{1:t+1}|s_t = x_{1:t}, a_t = x_{t+1}) = 1$ is deterministic and can be ignored.

Secondly, the reward function is not explicitly given for text generation. Each text sequence $x_{1:T} = x_1, x_2, \cdots, x_T$ in the training dataset is formulated by a trajectory τ by experts from the distribution $p(\tau)$, and we have to learn a reward function that explains the expert behavior.

Concretely, IRL consists of two phases: (1) estimate the underlying reward function of experts from the training dataset; (2) learn an optimal policy to generate texts, which aims to maximize the expected rewards. These two phases are executed alternately. The framework of our method is as shown in Figure 1.

2.1 Reward Approximator

Following the framework of maximum entropy IRL [Ziebart *et al.*, 2008], we assume that the texts in training set are sampled from the distribution $p_\phi(\tau)$,

$$p_\phi(\tau) = \frac{1}{Z} \exp(R_\phi(\tau)), \quad (2)$$

where $R_\phi(\tau)$ an unknown reward function parameterized by ϕ, $Z = \int_\tau \exp(R_\phi(\tau))d\tau$ is the partition function.

The reward of trajectory $R_\phi(\tau)$ is a parameterized reward function and assumed to be summation of the rewards of each steps $r_\phi(s_t, a_t)$:

$$R_\phi(\tau) = \sum_t r_\phi(s_t, a_t), \quad (3)$$

where $r_\phi(s_t, a_t)$ is modeled a simple feed-forward neural network as shown in Figure 2.

Objective of Reward Approximator

The objective of the reward approximator is to maximize the log-likehood of the samples in the training set:

$$\mathcal{J}_r(\phi) = \frac{1}{N} \sum_{n=1}^N \log p_\phi(\tau_n) = \frac{1}{N} \sum_{n=1}^N R_\phi(\tau_n) - \log Z, \quad (4)$$

where τ_n denotes the n_{th} sample in the training set D_{train}.

Thus, the derivative of $\mathcal{J}_r(\phi)$ is:

$$\nabla_\phi \mathcal{J}_r(\phi) = \frac{1}{N}\sum_n \nabla_\phi R_\phi(\tau_n) - \frac{1}{Z}\int_\tau \exp(R_\phi(\tau))\nabla_\phi R_\phi(\tau)\mathrm{d}\tau$$
$$= \mathbb{E}_{\tau\sim p_{data}}\nabla_\phi R_\phi(\tau) - \mathbb{E}_{\tau\sim p_\phi(\tau)}\nabla_\phi R_\phi(\tau). \quad (5)$$

Intuitively, the reward approximator aims to increase the rewards of the real texts and decrease the trajectories drawn from the distribution $p_\phi(\tau)$. As a result, $p_\phi(\tau)$ will be an approximation of p_{data}.

Importance Sampling Though it is quite straightforward to sample $\tau \sim p_\phi(\tau)$ in Eq. (5), it is actually inefficient in practice. Instead, we directly use trajectories sampled by text generator $q_\theta(\tau)$ with importance sampling. Concretely, Eq. (5) is now formalized as:

$$\nabla_\phi \mathcal{J}_r(\phi) \approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\phi R_\phi(\tau_i) - \frac{1}{\sum_j w_j}\sum_{j=1}^{M} w_j \nabla_\phi R_\phi(\tau'_j), \quad (6)$$

where $w_j \propto \frac{\exp(R_\phi(\tau_j))}{q_\theta(\tau_j)}$. For each batch, we sample N texts from the train set and M texts drawn from q_θ.

2.2 Text Generator

The text generator uses a policy $\pi_\theta(a|s)$ to predict the next word one by one. The current state s_t can be modeled by LSTM neural network as shown in Figure 2. For $\tau = \{s_1, a_1, s_2, a_2 ..., s_T, a_T\}$,

$$\mathbf{s}_t = \mathrm{LSTM}(\mathbf{s}_{t-1}, \mathbf{e}_{a_{t-1}}), \quad (7)$$
$$\pi_\theta(\mathbf{a}_t|s_t) = \mathrm{softmax}(\mathbf{W}\mathbf{s}_t + \mathbf{b}), \quad (8)$$

where $\mathbf{s}_t$ is the vector representation of state s_t; $\mathbf{a}_t$ is distribution over the vocabulary; $\mathbf{e}_{a_{t-1}}$ is the word embedding of a_{t-1}; θ denotes learnable parameters including $\mathbf{W}$, $\mathbf{b}$ and all the parameters of LSTM.

Objective of Text Generator

Following "entropy regularized" policy gradient [Williams, 1992; Nachum *et al.*, 2017], the objective of text generator is to maximize the expected reward plus an entropy regularization.

$$\mathcal{J}_g(\theta) = \mathbb{E}_{\tau\sim q_\theta(\tau)}[R_\phi(\tau)] + H(q_\theta(\tau)) \quad (9)$$

where $H(q_\theta(\tau)) = -\mathbb{E}_{q_\theta(\tau)}[\log q_\theta(\tau)]$ is an entropy term, which can prevent premature entropy collapse and encourage the policy to generate more diverse texts.

Intuitively, the "entropy regularized" expected reward can be rewrite as

$$\mathcal{J}_g(\theta) = -\mathrm{KL}(q_\theta(\tau)||p_\phi(\tau)) + \log Z, \quad (10)$$

where $Z = \int_\tau \exp(R_\phi(\tau))d\tau$ is the partition function and can be regarded as a constant unrelated to θ. Therefore, the objective is also to minimize the KL divergence between the text generator $q_\theta(\tau)$ and the underlying distribution $p_\phi(\tau)$.

Thus, the derivative of $\mathcal{J}_g(\theta)$ is

$$\nabla_\theta \mathcal{J}_g(\theta) = \sum_t \mathbb{E}_{\pi_\theta(a_t|s_t)}\nabla_\theta \log \pi_\theta(a_t|s_t)$$

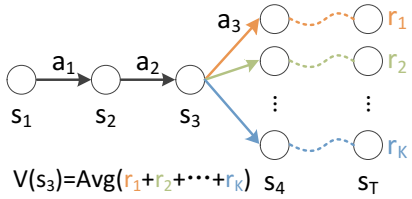

Figure 3: MCMC sampling for calculating the expected total reward at each state.

$$[R_\phi(\tau_{t:T}) - \log \pi_\theta(a_t|s_t) - 1]. \quad (11)$$

where $R_\phi(\tau_{t:T})$ denotes the reward of partial trajectory $\tau_t, \cdots, \tau_T$. For obtaining lower variance, $R(\tau_{t:T})$ can be approximately computed by

$$R_\phi(\tau_{t:T}) \approx r_\phi(s_t, a_t) + V(s_{t+1}), \quad (12)$$

where $V(s_{t+1})$ denotes the expected total reward at state s_{t+1} and can be approximately computed by MCMC. Figure 3 gives an illustration.

2.3 Why Can IRL Alleviate Mode Collapse?

GANs often suffer from mode collapse, which is partially caused by the use of Jensen-Shannon (JS) divergence. There is a reverse KL divergence $\mathrm{KL}(q_\theta(\tau)||p_{data})$ in JS divergence. Since the p_{data} is approximated by training data, the reverse KL divergence encourages $q_\theta(\tau)$ to generate safe samples and avoid generating samples where the training data does not occur. In our method, the objective is $\mathrm{KL}(q_\theta(\tau)||p_\phi(\tau))$. Different from GANs, we use $p_\phi(\tau)$ in IRL framework instead of p_{data}. Since $p_\phi(\tau)$ never equals to zero due to its assumption, IRL can alleviate the model collapse problem in GANs.

3 Training

The training procedure consists of two steps: (I) reward approximator update step (**r-step**) and (II) text generator update step (**g-step**). These two steps are applied iteratively as described in Algorithm (1).

Initially, we have r_ϕ with random parameters and π_θ with pre-trained parameters by maximum log-likelihood estimation on D_{train}. The r-step aims to update r_ϕ with π_θ fixed. The g-step aims to update π_θ with r_ϕ fixed.

4 Experiment

To evaluate the proposed model, we experiment on three corpora: the synthetic oracle dataset [Yu *et al.*, 2017], the COCO image caption dataset [Chen *et al.*, 2015] and the IMDB movie review dataset [Diao *et al.*, 2014]. Furthermore, we also evaluate the performance by human on the image caption dataset and the IMDB corpus. Experimental results show that Our method outperforms the previous methods. Table 1 gives the experimental settings on the three corpora.

4.1 Synthetic Oracle

The synthetic oracle dataset is a set of sequential tokens which are regraded as simulated data comparing to the real-

Algorithm 1 IRL for Text Generation

repeat
  Pretrain π_θ on D_{train} with MLE
  for n_r epochs in r-step **do**
    Drawn $\tau^{(1)}, \tau^{(2)}, \cdots, \tau^{(i)}, \cdots, \tau^{(N)} \sim p_{data}$
    Drawn $\tau'^{(1)}, \tau'^{(2)}, \cdots, \tau'^{(j)}, \cdots, \tau'^{(M)} \sim q_\theta$
    Update $\phi \leftarrow \phi + \alpha \nabla_\phi \mathcal{J}_r(\phi)$
  end for
  for n_g batches in g-step **do**
    Drawn $\tau^{(1)}, \tau^{(2)}, \cdots, \tau^{(i)}, \cdots, \tau^{(N)} \sim q_\theta$
    Calculate expected reward $R_\phi(\tau_{t:T})$ by MCMC
    Update $\theta \leftarrow \theta + \beta \nabla_\theta \mathcal{J}_g(\theta)$
  end for
until Convergence

Hyper-Parameters	Synthetic Oracle L = 20	Synthetic Oracle L = 40	COCO & IMDB
Text Generator			
- Embedding dimension	32	64	128
- Hidden layer dimension	32	64	128
- Batch size	64		128
- Optimizer & lr rate	Adam, 0.005		Adam, 0.005
Reward Approximator			
- Drop out	0.75	0.45	0.75
- Batch size	64		1024
- Optimizer & lr rate	Adam, 0.0004		Adam, 0.0004

Table 1: Configurations on hyper-parameters.

world language data. It uses a randomly initialized LSTM [1] as the oracle model to generate 10000 samples of length 20 and 40 respectively as the training set for the following experiments.

The oracle model, which has an intrinsic data distribution P_{oracle}, can be used to evaluate the sentences generated by the generative models. The average negative log-likelihood(NLL) is usually conducted to score the quality of the generated sequences [Yu *et al.*, 2017; Guo *et al.*, 2017; Lin *et al.*, 2017]. The lower the NLL score is, the better token sequences we have generated.

Training Strategy In experiments, we find that the stability and performance of our framework depend on the training strategy. Figure 4 shows the effects of pretraining epochs. It works best in generating texts of length 20 with 50 epochs of MLE pretraining, and in generating texts of length 40 with 10 epochs of pretraining.

Figure 5 shows that the proportion of $n_r : n_g$ in Algorithm 1 affects the convergence and final performance. It implies that sufficient training on the approximator in each iteration will lead to better results and convergence. Therefore, we take $n_r : n_g = 10 : 1$ as our final training configuration.

[1] The synthetic data and the oracle LSTM are publicly available at https://github.com/LantaoYu/SeqGAN and https://github.com/CR-Gjx/LeakGAN

Length	MLE	SeqGAN	RankGAN	LeakGAN	IRL	Ground Truth
20	9.038*	8.736*	8.247*	7.038*	**6.913**	5.750
40	10.411*	10.310*	9.958*	7.197*	**7.083**	4.071

Table 2: The overall NLL performance on synthetic data. "Ground Truth" consists of samples generated by the oracle LSTM model. Results with * are reported in their papers.

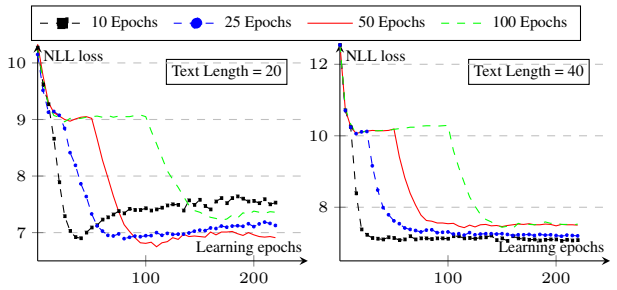

Figure 4: Learning curves with different pretrain epochs (10, 25, 50, 100 respectively) on texts of length 20 and 40.

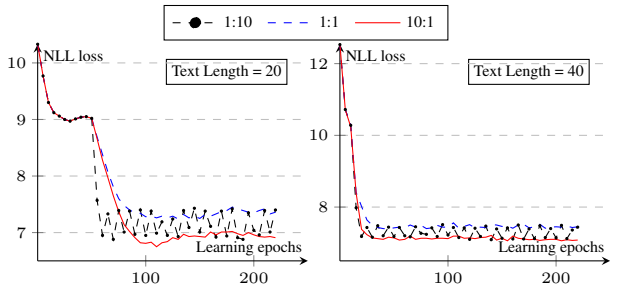

Figure 5: Learning curves with different training equilibriums between text generator and reward approximator on texts of length 20 and 40. The proportion in the legend means $n_r : n_g$.

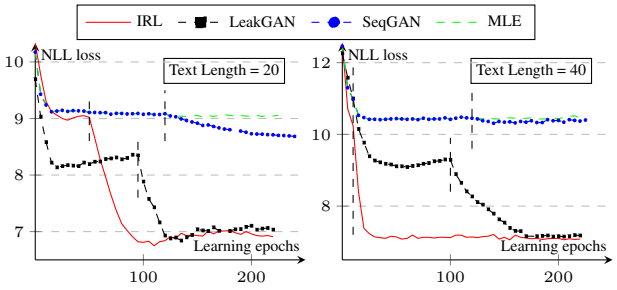

Figure 6: Learning curves of different methods on the synthetic data of length 20 and 40 respectively.The vertical dashed line indicates the end of the pre-training of SeqGAN, LeakGAN and our method respectively. Since RankGAN didn't publish code, we cannot plot the result of RankGAN.

Results Table 2 gives the results. We compare our method with other previous state-of-the-art methods: maximum likelihood estimation (MLE), SeqGAN, RankGAN and LeakGAN. The listed ground truth values are the average NLL of the training set. Our method outperforms the previous state-of-the-art results (6.913 and 7.083 on length of 20 and 40

respectively). Figure 6 shows that Our method convergences faster and obtains better performance than other state-of-art methods.

Analysis Our method performs better due to the instant rewards approximated at each step of generation. It addresses the reward sparsity issue occurred in previous methods. Thus, the dense learning signals guide the generative policy to capture the underlying distribution of the training data more efficiently.

4.2 COCO Image Captions

The image caption dataset [Chen *et al.*, 2015] consists of image-description pairs. The length of captions is between 8 and 20. Following LeakGAN [Guo *et al.*, 2017], for preprocessing, we remove low frequency words (less than 10 times) as well as the sentences containing them. We randomly choose 80,000 texts as training set, and another 5,000 as test set. The vocabulary size of the dataset is 4,939. The average sentence length is 12.8.

New Evaluation Measures on BLEU To evaluate different methods, we employ BLEU score to evaluate the qualities of the generated texts.

- *Forward BLEU (BLEU$_F$)* uses the testset as reference, and evaluates each generated text with BLEU score.
- *Backward BLEU (BLEU$_B$)* uses the generated texts as reference, and evaluates each text in testset with BLEU score.
- *BLEU$_{HA}$* is the harmonic average value of BLEU$_F$ and BLEU$_B$.

Intuitively, BLEU$_F$ aims to measure the precision (quality) of the generator, while BLEU$_B$ aims to measure the recall (diversity) of the generator.

The configurations of three proposed valuation measures are shown in Table 3.

Metrics	Evaluated Texts	Reference Texts
BLEU$_F$	Generated Texts	Test Set
BLEU$_B$	Test Set	Generated Texts
BLEU$_{HA}$	$\frac{2\times \text{BLEU}_F \times \text{BLEU}_B}{\text{BLEU}_F + \text{BLEU}_B}$	

Table 3: Configurations of BLEU$_F$, BLEU$_B$ and BLEU$_{HA}$.

BLEU$_F$ For BLEU$_F$, we sample 1000 texts for each method as evaluated texts. The reference texts are the whole test set. We list the BLEU$_F$ scores of different frameworks and ground truth as shown in first subtable of Table 4. Surprisingly, it shows that results of LeakGAN beat the rest, even the ground truth (LeakGAN has averagely 10 points higher than the ground truth). It may due to the mode collapse which frequently occurs in GAN. The text generator is prone to generate safe text patterns but misses many other patterns. Therefore, BLEU$_F$ is failing to measure the diversity of the generated sentences.

Metrics	MLE	SeqGAN	RankGAN	LeakGAN	IRL	Ground Truth
BLEU$_F$-2	0.798	0.821	0.850*	**0.914**	0.829	0.836
BLEU$_F$-3	0.631	0.632	0.672*	**0.816**	0.662	0.672
BLEU$_F$-4	0.498	0.511	0.557*	**0.699**	0.586	0.598
BLEU$_F$-5	0.434	0.439	0.544*	**0.632**	0.542	0.557
BLEU$_B$-2	0.801	0.682	-	0.790	**0.868**	0.869
BLEU$_B$-3	0.622	0.542	-	0.605	**0.718**	0.710
BLEU$_B$-4	0.551	0.513	-	0.549	**0.660**	0.649
BLEU$_B$-5	0.508	0.469	-	0.506	**0.609**	0.601
BLEU$_{HA}$-2	0.799	0.745	-	0.847	**0.848**	0.852
BLEU$_{HA}$-3	0.626	0.584	-	**0.695**	0.689	0.690
BLEU$_{HA}$-4	0.523	0.512	-	0.615	**0.621**	0.622
BLEU$_{HA}$-5	0.468	0.454	-	0.562	**0.574**	0.578

Table 4: Results on COCO image caption dataset. Results of RankGAN with * are reported in [Guo *et al.*, 2017]. Results of MLE, SeqGAN and LeakGAN are based on their published implementations.

Metrics	MLE	SeqGAN	LeakGAN	IRL	Ground Truth
BLEU$_F$-2	0.652	0.683	**0.809**	0.788	0.791
BLEU$_F$-3	0.405	0.418	**0.554**	0.534	0.539
BLEU$_F$-4	0.304	0.315	**0.358**	0.352	0.355
BLEU$_F$-5	0.202	0.221	0.252	**0.262**	0.258
BLEU$_B$-2	0.672	0.615	0.730	**0.755**	0.785
BLEU$_B$-3	0.495	0.451	0.483	**0.531**	0.534
BLEU$_B$-4	0.316	0.299	0.318	**0.347**	0.357
BLEU$_B$-5	0.226	0.209	0.232	**0.254**	0.258
BLEU$_{HA}$-2	0.662	0.647	0.767	**0.771**	0.788
BLEU$_{HA}$-3	0.445	0.434	0.516	**0.533**	0.537
BLEU$_{HA}$-4	0.310	0.307	0.337	**0.350**	0.356
BLEU$_{HA}$-5	0.213	0.215	0.242	**0.258**	0.258

Table 5: Results on IMDB Movie Review dataset. Results of MLE, SeqGAN and LeakGAN are based on their published implementations. Since RankGAN didn't publish code, we cannot report the results of RankGAN on IMDB.

BLEU$_B$ For BLEU$_B$, we sample 5000 texts for each method as reference texts. The evaluated texts consist 1000 texts sampled from the test set. The BLEU$_B$ of each method is listed in the second block of Table 4. Intuitively, the higher the BLEU$_B$ score is, the more diversity the generator gets. From Table 4, our method outperforms the other methods, which implies that our method generates more diversified texts than the other methods. As we have analyzed before, the diversity of our method may be derived from "entropy regularization" policy gradient.

BLEU$_{HA}$ Finally, BLEU$_{HA}$ takes both generation quality and diversity into account and the results are shown in the last block of Table 4. The BLEU$_{HA}$ reveals that our work gains better performance than other methods.

Models	COCO	IMDB
MLE	(1) A girl sitting at a table in front of medical chair. (2) The person looks at a bus stop while talking on a phone.	(1) If somebody that goes into a films and all the film cuts throughout the movie. (2) Overall, it is what to expect to be she made the point where she came later.
SeqGAN	(1) A man holding a tennis racket on a tennis court. (2) A woman standing on a beach next to the ocean.	(1) The story is modeled after the old classic "B" science fiction movies we hate to love, but do. (2) This does not star Kurt Russell, but rather allows him what amounts to an extended cameo.
LeakGAN	(1) A bathroom with a toilet , window , and white sink. (2) A man in a cowboy hat is milking a black cow.	(1) I was surprised to hear that he put up his own money to make this movie for the first time. (2) It was nice to see a sci-fi movie with a story in which you didn't know what was going to happen next.
IRL (This work)	(1) A woman is standing underneath a kite on the sand. (2) A dog owner walks on the beach holding surfboards.	(1) Need for Speed is a great movie with a very enjoyable storyline and a very talented cast. (2) The effects are nothing spectacular, but are still above what you would expect, all things considered.

Table 6: Case study. Generated texts from different models on COCO image caption and IMDB movie review datasets.

Corpora	MLE	SeqGAN	LeakGAN	IRL	Ground Truth
COCO	0.205	0.450	0.543	**0.550**	0.725
IMDB	0.138	0.205	0.385	**0.463**	0.698

Table 7: Results of Turing test. Samples of MLE, SeqGAN and LeakGAN are generated based on their published implementations. Since RankGAN didn't publish code, we cannot generate samples of RankGAN.

4.3 IMDB Movie Reviews

We use a large IMDB text corpus [Diao *et al.*, 2014] for training the generative models as long-length text generation. The dataset is a collection of 350K movie reviews. We select sentences with the length between 17 and 25, set word frequency at 180 as the threshold of frequently occurred words and remove sentences with low frequency words. Finally we randomly choose 80000 sentences for training and 3000 sentences for testing with the vocabulary size at 4979 and the average sentence length is 19.6.

IMDB is a more challenging corpus. Unlike sentences in COCO Image captions dataset, which mainly contains simple sentences, e.g., sentences only with the subject-predicate structure, IMDB movie reviews are comprised of various kinds of compound sentences. Besides, the sentence length of IMDB is much longer than that of COCO.

We also use the same metrics ($BLEU_F$, $BLEU_B$, $BLEU_{HA}$) to evaluate our method. The results in Table 5 show our method outperforms other models.

4.4 Turing Test and Case Study

The evaluation metrics mentioned above are still not sufficient for evaluating the quality of the sentences because they just focus on the local statistics, ignoring the long-term dependency characteristic of language. So we have to conduct a Turing Test based on scores by a group of people. Each sentence will get 1 point when it is viewed as a real one, otherwise 0 point. We perform the test on frameworks of MLE, SeqGAN, LeakGAN and our method on COCO Image captions dataset and IMDB movie review dataset.

Practically, we sample 20 sentences by each generator from different methods, and for each sentence, we ask 20 different people to score it. Finally, we compute the average score for each sentence, and then calculate the average score for each method according to the sentences it generate.

Table 6 shows some generated samples of our and the baseline methods. These samples are what we have collected for people to score.

The results in Table 7 indicate that the generated sentences of our method have better quality than those generated by MLE, SeqGAN and LeakGAN, especially for long texts.

5 Related Work

Text generation is a crucial task in NLP which is widely used in a bunch of NLP applications. Text generation is more difficult than image generation since texts consist of sequential discrete decisions. Therefore, GAN fails to back propagate the gradients to update the generator. Recently, several methods have been proposed to alleviate this problem, such as Gumbel-softmax GAN [Kusner and Hernández-Lobato, 2016], RankGAN [Lin *et al.*, 2017], TextGAN [Zhang *et al.*, 2017], LeakGAN [Guo *et al.*, 2017], etc.

SeqGAN [Yu *et al.*, 2017] addresses the differentiation problem by introducing RL methods, but still suffers from the problem of reward sparsity. LeakGAN [Guo *et al.*, 2017] manages the reward sparsity problem via Hierarchical RL methods. Joji Toyama [2018] designs several reward functions for partial sequence to solve the issue. However, the generated texts of these methods still lack diversity due to the mode collapse issue. In this paper, we employ IRL framework [Finn *et al.*, 2016] for text generation. Benefiting from its inherent instant reward learning and entropy regularization, our method can generate more diverse texts.

6 Conclusions & Future Work

In this paper, we propose a new method for text generation by using inverse reinforcement learning (IRL). This method alleviates the problems of reward sparsity and mode collapse in the adversarial generation models. In addition, we propose three new evaluation measures based on BLEU score to better evaluate the generated texts.

In the future, we would like to generalize the IRL framework to the other NLP tasks, such as machine translation, summarization, question answering, etc.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by the National Key Research and Development Program of China (No. 2017YFB1002104), Shanghai Municipal Science and Technology Commission (No. 17JC1404100 and 16JC1420401), and National Natural Science Foundation of China (No. 61672162).

References

[Bayer and Osendorfer, 2014] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv*, 2014.

[Bengio *et al.*, 2015] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179, 2015.

[Che *et al.*, 2017] Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv*, 2017.

[Chen *et al.*, 2015] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv*, 2015.

[Chung *et al.*, 2015] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *NIPS*, pages 2980–2988, 2015.

[Diao *et al.*, 2014] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *SIGKDD*, pages 193–202. ACM, 2014.

[Finn *et al.*, 2016] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, pages 49–58, 2016.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[Graves, 2013] Alex Graves. Generating sequences with recurrent neural networks. *arXiv*, 2013.

[Guo *et al.*, 2017] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. *arXiv*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Joji Toyama, 2018] Kotaro Nakayama Yutaka Matsuo Joji Toyama, Yusuke Iwasawa. Toward learning better metrics for sequence generation training with policy gradient. *ICLR submission*, 2018.

[Kusner and Hernández-Lobato, 2016] Matt J Kusner and José Miguel Hernández-Lobato. GANs for sequences of discrete elements with the gumbel-softmax distribution. *arXiv*, 2016.

[Lin *et al.*, 2017] Kevin Lin, Dianqi Li, Xiaodong He, Mingting Sun, and Zhengyou Zhang. Adversarial ranking for language generation. In *NIPS*, pages 3158–3168, 2017.

[Metz *et al.*, 2016] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv*, 2016.

[Nachum *et al.*, 2017] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *arXiv*, 2017.

[Salimans *et al.*, 2016] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, pages 2234–2242, 2016.

[Serban *et al.*, 2017] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301, 2017.

[Su *et al.*, 2018] Jinyue Su, Jiacheng Xu, Xipeng Qiu, and Xuanjing Huang. Incorporating discriminator in sentence generation: a gibbs sampling method. In *AAAI*, 2018.

[Wang *et al.*, 2017] Heng Wang, Zengchang Qin, and Tao Wan. Text generation based on generative adversarial nets with latent variable. *arXiv*, 2017.

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

[Yu *et al.*, 2017] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858, 2017.

[Zhang *et al.*, 2017] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *arXiv*, 2017.

[Ziebart *et al.*, 2008] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.