Toward Diverse Text Generation with Inverse Reinforcement Learning

RL Reading Group
Yanjun Gao
04/04/2019

- Introduction to Inverse Reinforcement Learning
- Background: Text Generation using GAN
- Problems and Motivation of this paper
- Architecture and Algorithms
- Experiments and Results
- Discussion

Introduction to Inverse Reinforcement Learning

penn state university

Learning agents for uncertain environments (extended abstract)

COLT 1998

Stuart Russell*
Computer Science Division
University of California
Berkeley, CA 94720
russell@cs.berkeley.edu

Algorithms for Inverse Reinforcement Learning

ICML 2000

Andrew Y. Ng ANG@CS.BERKELEY.EDU
Stuart Russell RUSSELL@CS.BERKELEY.EDU
Computer Science Division, U.C. Berkeley, Berkeley, CA 94720 USA

AAAI 2008

Maximum Entropy Inverse Reinforcement Learning

Brian D. Ziebart, Andrew Maas, J.Andrew Bagnell, and **Anind K. Dey**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bziebart@cs.cmu.edu, amaas@andrew.cmu.edu, dbagnell@ri.cmu.edu, anind@cs.cmu.edu

penn state university

We always assume that the reward function is fixed and known, however …

> It seems clear, however, that *in examining animal and human behaviour we must consider the reward function as an unknown to be ascertained*. The reasons for this are straightforward:
>
> - The specification of a given reward function is an empirical hypothesis and may turn out to be wrong. For example, it was assumed initially that horses' gait selection for a given speed was determined by energetic economy (Hoyt & Taylor, 1981); this turns out not to be the case (Farley & Taylor, 1991).
> - The parameters of a *multiattribute* reward function can surely not be determined *a priori*; e.g., for running, attributes might be speed, efficiency, stability against perturbations, wear and tear on muscles, tendons, and bones, etc. How are these to be weighted and combined?

—- Russell, 1998

Introduction to Inverse Reinforcement Learning

The inverse reinforcement learning (IRL) problem can be characterized informally as follows (Russell, 1998):

Given 1) measurements of an agent's behavior over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to that agent; 3) if available, a model of the environment.

Determine the reward function being optimized.

—- Russell, 1998; Ng, Russell, 2000

- Exposure Bias

A *distribution discrepancy* issue in conventional Seq2Seq models due to the *maximum likelihood prediction*: the model is trained on distribution given the training set (words drawn from training set), trained to predict next word given the ground-truth input, but will generate the output sequence at test time by one word at a time, and keeps feeding in the generated sequence back to the model. The model makes the prediction based on a distribution it might never observe in training before; *"This process is very brittle because the model was trained on a different distribution of inputs, namely, words drawn from the data distribution, as opposed to words drawn from the model distribution. As a result the errors made along the way will quickly accumulate."*

— Marc' Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. "SEQUENCE LEVEL TRAINING WITH RECURRENT NEURAL NETWORKS." ICLR 2016

— Bengio, Samy, et al. "Scheduled sampling for sequence prediction with recurrent neural networks." Advances in Neural Information Processing Systems. 2015.

Background: Text Generation using GAN

- Exposure Bias

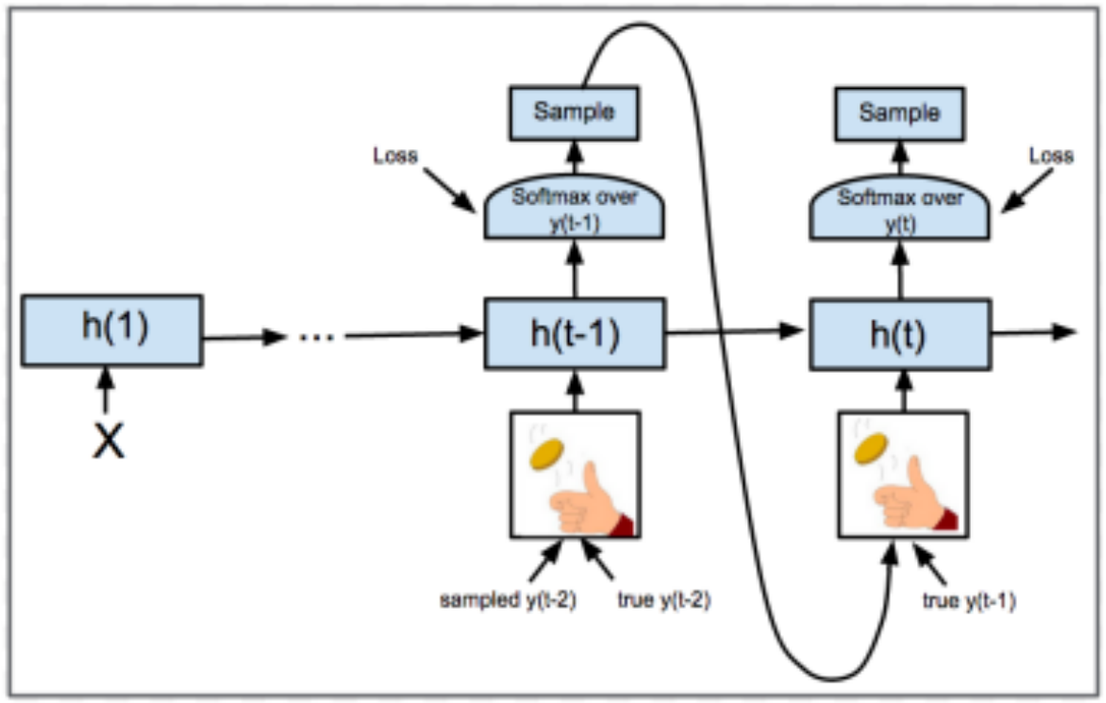
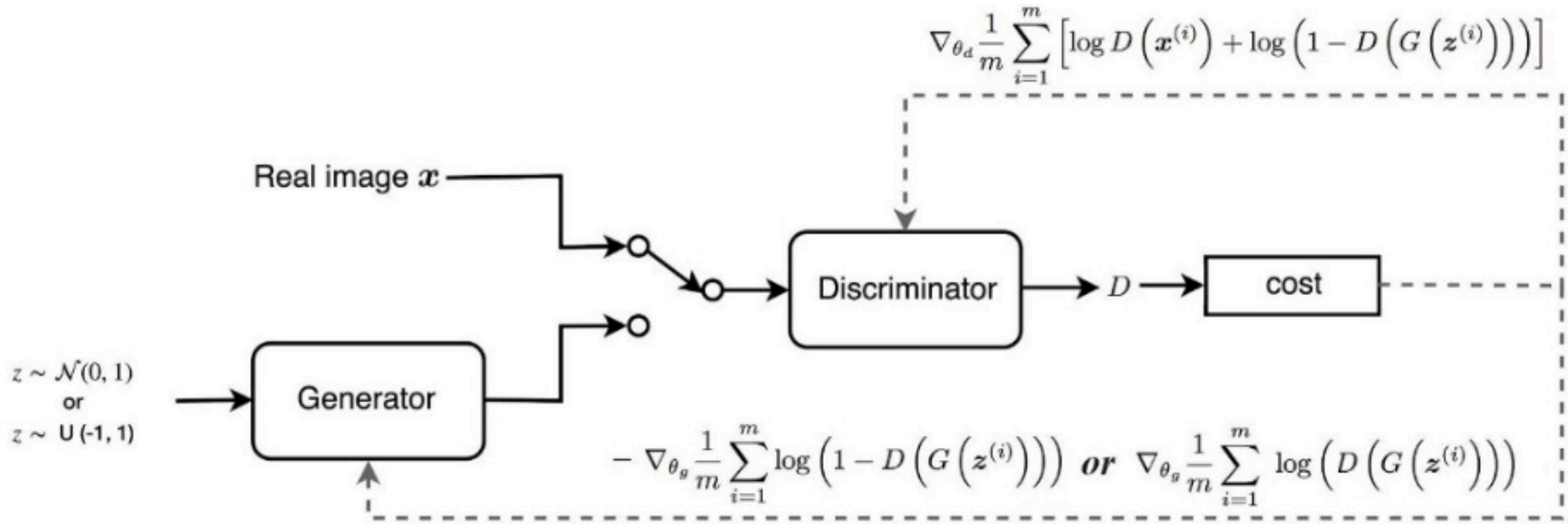


Figure 1: Illustration of the Scheduled Sampling approach, where one flips a coin at every time step to decide to use the true previous token or one sampled from the model itself.

Bengio, Samy, et al. "Scheduled sampling for sequence prediction with recurrent neural networks." Advances in Neural Information Processing Systems. 2015.

Background: Text Generation using GAN

A quick intro to GAN

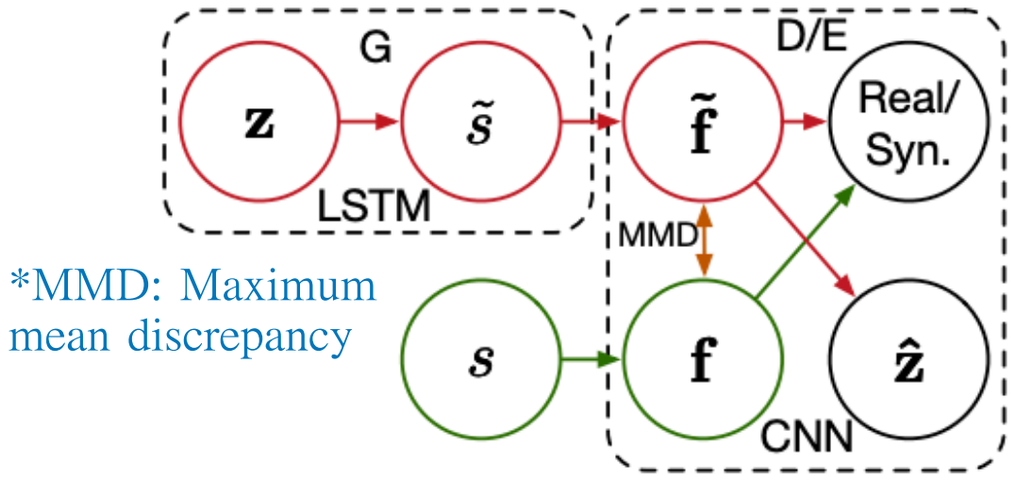


$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Background: Text Generation using GAN

Why GAN could help solve this problem? A sip of two Text GAN architectures

The idea is similar to sampling the data, but in a more advanced way



*MMD: Maximum mean discrepancy

Figure 1. Model scheme of TextGAN. Latent codes z are fed through a generator $G(\cdot)$, to produce synthetic sentence $\tilde{s}$. Synthetic and real sentences ($\tilde{s}$ and s) are fed into a binary discriminator $D(\cdot)$, for real *vs.* fake (synthetic) prediction, and also for latent code reconstruction $\hat{z}$. $\tilde{\boldsymbol{f}}$ and $\boldsymbol{f}$ represent features of $\tilde{s}$ and s, respectively.

Zhang, Yizhe, et al. "Adversarial feature matching for text generation." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

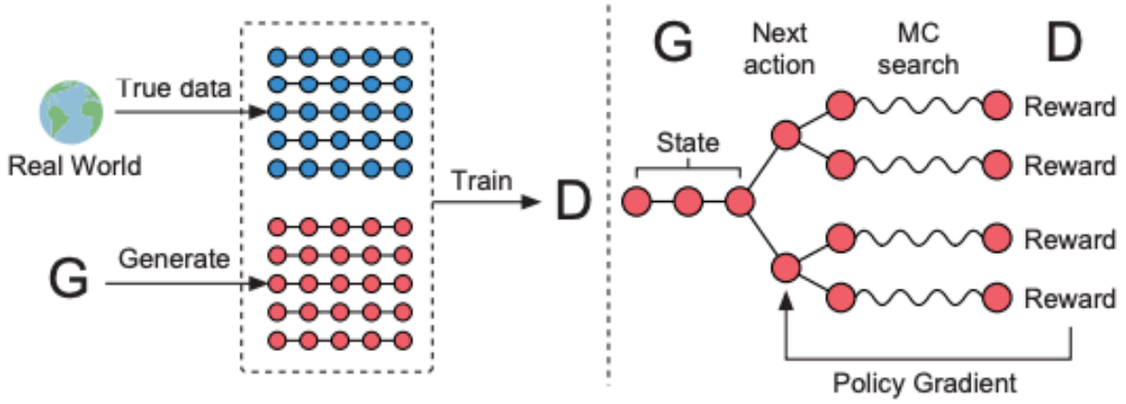


Figure 1: The illustration of SeqGAN. Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

Yu, Lantao, et al. "Seqgan: Sequence generative adversarial nets with policy gradient." Thirty-First AAAI Conference on Artificial Intelligence. 2017.

Problems and Motivation

- Reward Sparsity

We hope that the discriminator will always correctly discriminate the “fake” from real text, but this case does not happen often in practice. Therefore it is dangerous to totally rely on the discriminator for the reward signal.

- Mode Collapse

One of the most difficult problems to be solved in GAN. The generator tries so hard to fool the discriminator the most that it could be totally independent of its own distribution, and then collapse to a single point where the objective is the achieved without further sampling at all.

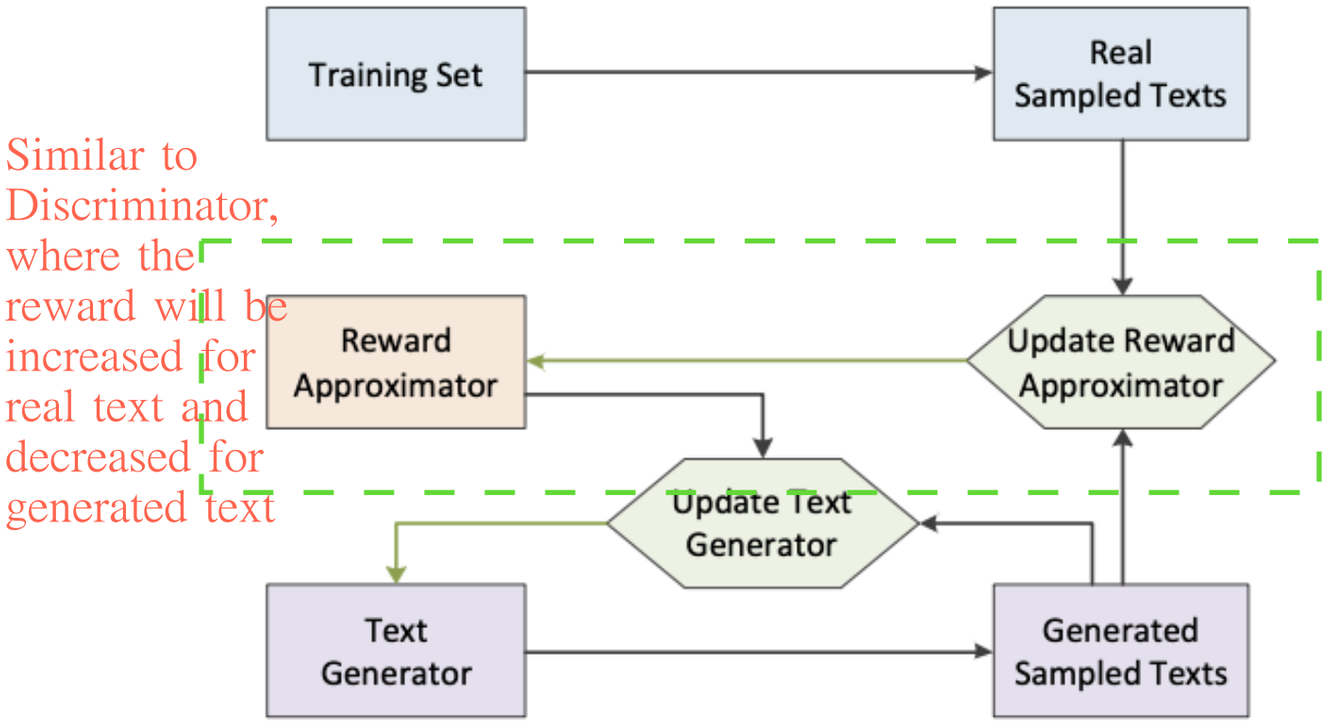
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

$$x^* = argmax_x D(x)$$

Problems and Motivation

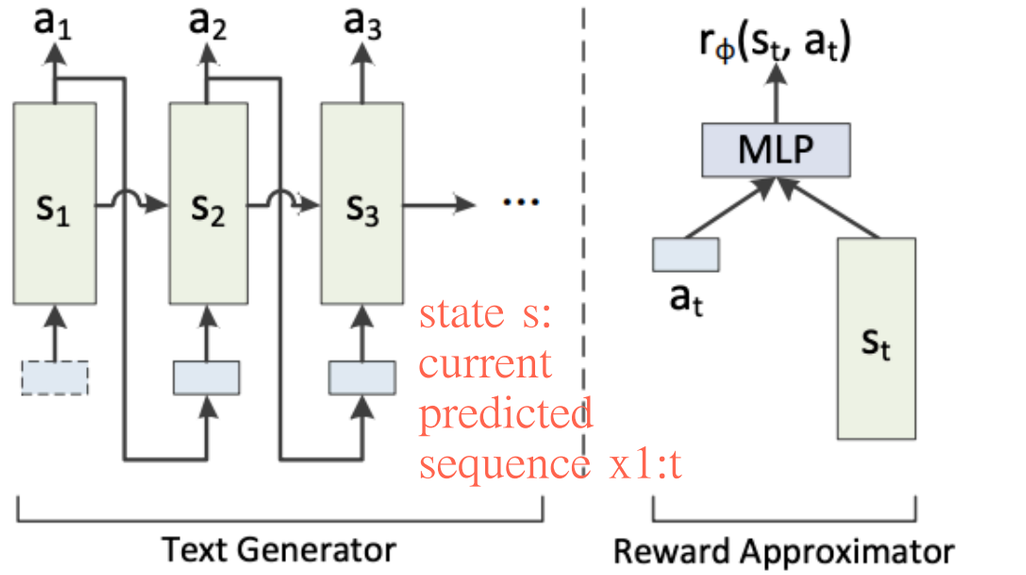
- We regard text generation as an IRL problem, which is a new perspective on this task.
- Following the maximum entropy IRL [Ziebart *et al.*, 2008], our method can improve the problems of reward sparsity and mode collapse.
- To better evaluate the quality of the generated texts, we propose three new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

Architectures and Algorithms



Similar to Discriminator, where the reward will be increased for real text and decreased for generated text

Figure 1: IRL framework for text generation.



action a: to select the next token xt+1

state s: current predicted sequence x1:t

Figure 2: Illustration of text generator and reward approximator.

1. esimate the reward function from training dataset $p(\tau)$
   Need to learn a trajectory tau from training set that explains this behaviour

2. Learn optimal policy to generate the text

generative model

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

MDP trajectory:

$$\tau = \{s_1, a_1, s_2, a_2 ..., s_T, a_T\}$$

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 $p_\phi(\tau)$,

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

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 $r_\phi(s_t, a_t)$:

$$R_\phi(\tau) = \sum_t r_\phi(s_t, a_t), \tag{3}$$

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

partition function: helps the reward function converges in fininte numbers of state, as proved in Ziebart et.al 2008.

Objective of Reward Approximator

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

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

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

Reward Approximator

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}.

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 $\tau = \{s_1, a_1, s_2, a_2 ..., s_T, a_T\}$,

$$\mathbf{s}_t = \text{LSTM}(\mathbf{s}_{t-1}, \mathbf{e}_{a_{t-1}}), \tag{7}$$

$$\pi_\theta(\mathbf{a}_t|s_t) = \text{softmax}(\mathbf{W}\mathbf{s}_t + \mathbf{b}), \tag{8}$$

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.

Text Generator

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.

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

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

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

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

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

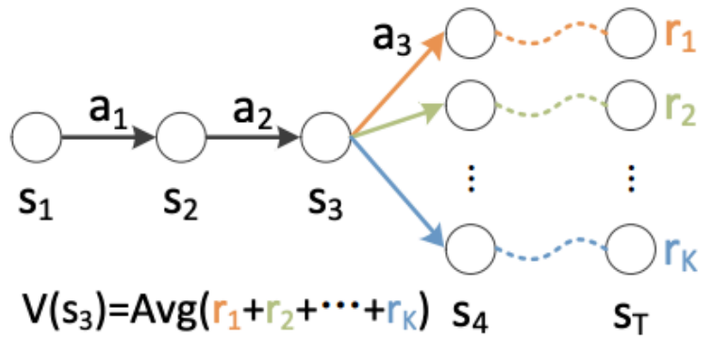


Figure 3: MCMC sampling for calculating the expected total reward at each state.

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

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

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

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.

Architectures and Algorithms

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

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.

Experiments and Results

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	$\frac{2\times\text{BLEU}_\text{F}\times\text{BLEU}_\text{B}}{\text{BLEU}_\text{F}+\text{BLEU}_\text{B}}$	

Table 3: Configurations of BLEU_F, BLEU_B and BLEU_HA.

Experiments and Results

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.

Experiments and Results

Turing test and Human score(0,1)

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.

References

— Shi, Zhan, et al. "Toward diverse text generation with inverse reinforcement learning." Proceedings of the 27th International Joint Conference on Artificial Intelligence. AAAI Press, 2018.

— Marc’ Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. "SEQUENCE LEVEL TRAINING WITH RECURRENT NEURAL NETWORKS.” ICLR 2016

— Bengio, Samy, et al. "Scheduled sampling for sequence prediction with recurrent neural networks." Advances in Neural Information Processing Systems. 2015.

— Russell, Stuart J. "Learning agents for uncertain environments." COLT. Vol. 98. 1998.

— Ng, Andrew Y., and Stuart J. Russell. "Algorithms for inverse reinforcement learning." Icml. Vol. 1. 2000.

— Zhang, Yizhe, et al. "Adversarial feature matching for text generation." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

— Yu, Lantao, et al. "Seqgan: Sequence generative adversarial nets with policy gradient." Thirty-First AAAI Conference on Artificial Intelligence. 2017.

— Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

— https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b