TextGAIL: Generative Adversarial Imitation Learning for Text Generation

Qingyang Wu 1, Lei Li 2, Zhou Yu 1,
1University of California, Davis, 2ByteDance,
{wilwu, joyu}@ucdavis.edu, lileilab@bytedance.com

Abstract

Exposure bias is one of the most challenging problems in text generation with maximum likelihood estimation (MLE) based methods. Previous studies try to tackle this problem by applying Generative Adversarial Networks. However, lacking sufficient language priors makes those methods perform worse than MLE-based methods. To address this problem, we propose TextGAIL, a generative adversarial imitation learning framework with large pretrained language models RoBERTa and GPT-2. We also leverage proximal policy optimization, and efficiently utilize human demonstrations to stabilize the generator’s training. We evaluate TextGAIL on a story ending generation task on the ROCStories dataset. We introduce the Perception score to evaluate our model. Experiments show that with the help of the large pretrained language models, our proposed method can generate highly diverse and appropriate story endings. We release the code and generated outputs.

1 Introduction

Text generation is gaining popularity in research recently. The most widely used approach for neural text generation is to maximize the joint probability of the target text sequence (Bengio et al., 2000), which is also referred as Maximum Likelihood Estimation (MLE). However, MLE suffers from the exposure bias problem caused by the training-inference discrepancy. During training, the model is trained on the ground truth, but during inference, the model needs to autoregressively predict the next word conditioned to its own previously generated words. This discrepancy leads to dull and repetitive sentences (Welleck et al., 2019; Wiseman and Rush, 2016; Serban et al., 2017). As a consequence, solving the exposure bias problem becomes important to improve the text generation quality.

One direction of reducing exposure bias is to utilize Generative Adversarial Networks (GAN). The main idea is to alternately train a discriminator to distinguish real samples from generated samples, and train the generator to improve its generated samples against the discriminator. Along this direction, there have been many studies that achieve certain progress. Nevertheless, there are increasing criticisms towards text GANs. Caccia et al. (2019) claims that GAN generated text is substantially worse than the ones generated by MLE. In the mean time, the recent large generative pretrained language models have greatly improved the quality of MLE generations (Radford and Sutskever, 2018; Radford et al., 2019).

We want to explore the possibility of applying GAN to the pretrained language models to solve the exposure bias problem while preserving stable results. Thus, we propose TextGAIL, a generative adversarial training framework that exploits the large-scale pretrained language models RoBERTa (Liu et al., 2019) and GPT-2 (Radford et al., 2019). However, traditional text GAN methods suffer from high variance of gradients. To reduce variance, we leverage the recent advancements in reinforcement learning (RL): generative imitation learning (GAIL) (Ho and Ermon, 2016), proximal policy optimization (PPO) (Schulman et al., 2017), and efficient use human demonstrations (Paine et al., 2019) to help optimize the generator.

To validate our model, we apply TextGAIL on one difficult language generation task - story ending generation, which is based on the ROCStories Corpora (Mostafazadeh et al., 2016). Story ending generation is a more open-ended task compared to some other language generation tasks, such as machine translation (Sutskever et al., 2014) or abstractive text summarization (Nallapati et al., 2016). It cannot use automatic evaluation metrics that are based on n-gram overlap such as BLEU (Papineni...
et al., 2002), because one story can have various diverse endings. To remedy this problem, we introduce Perception score to assess the text quality. It approximately measures the similarity between the real and generated samples’ distributions.

Our contributions of the paper are as follows: (1) Compared to the previous text GAN approaches, our method exploits the large-scale pretrained language models. (2) Further, we leverage generative adversarial imitation learning (GAIL), proximal policy optimization (PPO), and other techniques to stabilize the training of the generator. (3) Experimental results show that our method can generate highly diverse results with better logic and commonsense compared to MLE, suggesting the importance of large pretrained language models in improving the adversarial training performance.

2 Related Work

Exposure bias can lead to problems such as neural text degeneration by generating generic and repetitive generations. Even after the emergence of large-scale pretrained language model GPT-2, this problem is still prevalent. Many works tried to use Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) to solve the exposure bias problem caused by MLE (Ranzato et al., 2016; Welleck et al., 2019).

SeqGAN (Yu et al., 2017) is the very first paper which adopts the adversarial training idea at text generation. As text sequence is discrete, SeqGAN applies REINFORCE (Williams, 1992), which is a policy gradient algorithm, to train the generator with a reward defined by the discriminator’s prediction on the generated sample. The alternative is to apply Gumbel Softmax to relax discrete distribution to have a continuous approximation which enables the gradient back-propagation from the discriminator to the generator (Kusner and Hernández-Lobato, 2016). However, these two methods have very high variance, and are highly unstable in training. We tried to implement them on GPT-2, but failed to reach convergence.

There are many other text GANs (Ke et al., 2019; Che et al., 2017; Guo et al., 2018; Lin et al., 2017; Nie et al., 2019) that tried to solve the stability problem. The main difference is that we leverage the large pretrained language models for generative adversarial training. In specific, we use RoBERTa as the discriminator, and GPT-2 as the generator. We show the importance of large-scale pretraining to help improve the quality of text generations.

Besides, we utilize the recent Reinforcement Learning (RL) achievements, specifically Imitation Learning (IL) to combine the benefits of reinforcement learning and supervised learning. Behavior Cloning (BC) (Rahmatizadeh et al., 2018) is a simple IL method which is equivalent to MLE. However, BC is not generalizable to unseen data, due to its compounding error (Ross and Bagnell, 2010). Inverse Reinforcement Learning (IRL) (Ziebart et al., 2008) inversely learns the reward function on human demonstrations to solve this problem, but in practice it is computationally expensive. Ho and Ermon (2016) proposed generative adversarial imitation learning (GAIL) to improve IRL. We apply GAIL to text generation, and use proximal policy optimization (PPO) (Schulman et al., 2017), and efficient use of human demonstrations (Paine et al., 2019) to further stabilize the training.

Figure 1: Overall architecture of TextGAIL. The green arrow represents the samples from human demonstrations, while the yellow arrow represents the samples from the generator. \( p \) is the ratio between human demonstrations and generated samples during training.

3 Methods

In this section, we first introduce the main generative adversarial imitation learning framework. Then we explain the details of the paired discriminator and text generator, which are augmented with the large pretrained language models - RoBERTa and GPT-2. We further combine efficient use of human demonstrations to stabilize the training. In the end, we summarize the entire training process. The overall architecture is shown in Figure 1.

3.1 Generative Adversarial Imitation Learning

The training framework is mostly based on generative adversarial imitation learning (GAIL). The
framework consists of a generator $p_θ$ and a discriminator $D_φ$, which are parameterized with $θ$ and $φ$, respectively. The goal of the generator is to output sequences similar to human demonstrations. Meanwhile, the discriminator needs to distinguish the real from the generated sequences, and provide a single sparse reward for each generated sequence.

Here, we only consider the conditional generation setting. We have a context $x$ and want to generate the corresponding target sequence $y$. $y$ can either be given from ground truth in the dataset as real data or sampled from the generator $p_θ$ as fake data. GAIL finds a saddle point where the generator and discriminator satisfy the following expression:

$$
\min_{p_θ} \max_{D_φ} \mathbb{E}_{y \sim p^*} [D_φ(x, y)] + \mathbb{E}_{y \sim p_θ} [1 - D_φ(x, y)]
$$

(1)

where $p^*$ represents the real data distribution.

### 3.2 Paired Discriminator

The discriminator aims to distinguish between the real and generated samples. Traditional discriminator utilizes logistic loss (sigmoid), but it saturates quickly after the model learns the difference between the real and the generated samples. Moreover, logistic loss causes higher variance as it is hard for the model to calibrate its prediction based on a single estimate.

A better alternative to perform prediction is to utilize softmax cross-entropy instead of logistic loss. To calculate softmax cross-entropy, for each context $x$, we need one paired real target $y^*$ and a generated target $y$ as the inputs. The objective is to assign higher probabilities to the real target. Then we train the discriminator with cross-entropy loss:

$$
L_D(φ) = -\mathbb{E}_{y \sim p_θ, y^* \sim p^*} [\log D_φ(x, y^*)]
$$

(2)

$D(x, y^*)$ is the probability prediction after softmax for the real sample $y^*$. The paired discriminator can calibrate the predictions close to the real samples, which avoids overconfidence in logistic output. The probability of fake sequence $D(x, y)$ will be used as reward to optimize the generator.

Zellers et al. (2019) shows that the paired discriminator yields higher accuracy in fake news detection. When applied in the adversarial setting, the model can also be viewed as an importance sampling version of GAN that reduces variance compared to the traditional GANs (Lin, 2017).

### 3.3 Text Generator

For the generator, we begin by defining the probability of a text sequence as the joint probability of all the tokens:

$$
p_θ(y_{1:T}|x) = \prod_{t=0}^{T} p_θ(y_t|y_{<t}, x)
$$

(3)

where $y_{1:T}$ is a text sequence. $T$ is the sequence length, and $u_t$ is the word at the time step $t$. We sample from this distribution with Monte Carlo search to acquire the generated sequences.

Previous GAN methods have used policy gradient to directly optimize the generator with generated outputs which maximizes the expected reward below:

$$
\mathbb{E}[\hat{R}(x, y)] = \mathbb{E}_{y \sim p_θ} [1 - D_φ(x, y)]
$$

(4)

Directly optimizing this objective suffers from high variance of gradients, because the $D_φ$ is not stationary.

As a solution to reduce high variance, the original GAIL employs trust region policy optimization (TRPO) (Schulman et al., 2015), as it is crucial to ensure that $p_{θ_{t+1}}$ does not move too far away from $p_θ$. However, TRPO needs to compute natural gradient which is computationally expensive. As an alternative, we replace it with a more recent and stable method, proximal policy optimization (PPO) (Schulman et al., 2017). Compared to TRPO, PPO is easier to implement and generalize. PPO has better sample complexity in practice as well.

PPO applies importance sampling by the likelihood ratio between the current and old policy for $y \sim p_{θ_{old}}(\cdot|x)$:

$$
r(θ) = \frac{p_θ(y|x)}{p_θ(y|x)}
$$

(5)

Then it maximizes the expected reward by optimizing the following surrogate:

$$
L_{policy}(θ) = \min \left\{ r(θ) \hat{A}_y, \text{clip}(r(θ), 1 - ϵ, 1 + ϵ) \hat{A}_y \right\}
$$

(6)

where $\hat{A}_y$ is the advantage term that controls the update. This surrogate serves the same purpose as TRPO to have a trust region constraint on the gradient update. It will prevent the generator from moving too far away from the pretrained language model.
In addition, we add the KL divergence between the generator and another pretrained language model in replacement of the traditional maximum entropy regularizer. The final generator objective can be written as the following:

\[
L_G(\theta) = \mathbb{E}_{y \sim p_{\theta}(\cdot|x)} \left[ L_{\text{policy}}(\theta) + \beta \text{KL}(q|p_{\theta}) \right]
\]

(7)

\(q\) is another pretrained language model. The benefit for this replacement is to add entropy bonus to stimulate more exploration of the generator. Also it limits the generator distribution from changing too much from the pretrained language model \(q\).

### 3.4 Efficient Use of Human Demonstrations

Another problem is that GAIL alone has difficulty in solving complex exploration tasks (Paine et al., 2019). Especially in text generation, it faces a series of challenges, such as sparse rewards, latent knowledge, and highly diverse context. Fortunately, the recent work of DeepMind’s Recurrent Replay Distributed DQN from Demonstrations (R2D3) (Paine et al., 2019) introduced another imitation learning method that efficiently uses human demonstrations to solve hard exploration problems in RL. We apply this method to the TextGAIL framework.

During the training of the generator, we introduce a hyper-parameter \(p\) to control the ratio between the sampling from human demonstrations and generated sequences. Moreover, we force the advantage value for human demonstrations to always be a constant across the training.

\[
\hat{A}_y = \begin{cases} 
\text{constant} & y \in \text{human demos}, \\
\hat{R} & \text{otherwise},
\end{cases}
\]

(8)

\(\hat{R}\) is the reward after normalization. By fixing the human demonstrations’ advantage to be a constant, it stabilizes the training to update the generator distribution to be close to the real data distribution.

### 3.5 Training

Finally, we present the algorithm of TextGAIL. First, the generator is trained with part of the training set using MLE as warm-up. We also include a replay buffer to store temporary generated outputs and human demonstrations. After the replay buffer is full, the discriminator would assign rewards to all the context-target pairs in the replay buffer. Next, we normalize all the rewards in the buffer to have \(\hat{R}\). We use a constant reward for all human demonstrations. We update the generator \(p_{\theta}\) with PPO using the replay buffer. In the meantime, we update the discriminator \(D_{\phi}\) with the real and generated pairs. We repeat the process until the model over-fits. The summary of the training process is illustrated in Algorithm 1.

**Algorithm 1 TextGAIL**

1: **Initialize:** Collect human demonstrations
2: Warm-up the generator \(p_{\theta}\)
3: Replay Buffer \(B\)
4: for \(i = 1, 2, 3, \ldots\) do
5: Sample \(p\) proportion of human demonstrations \(y\)
6: Sample \(1-p\) proportion of generator outputs \(y \sim p(\cdot|x)\)
7: Put all sampled \((x, y)\) pairs into \(B\)
8: Collect rewards using discriminator \(D\) for all \((x, y) \in B\)
9: Normalize all the rewards to get \(\hat{R}\)
10: Replace rewards for human demonstrations with a constant
11: Update the discriminator \(\phi\) with Eq. 2
12: Update the generator \(\theta\) using the PPO with Eq. 7 and Eq. 8
13: Clear Buffer \(B\)
14: end for

### 4 Perception Score

Traditional n-gram matching metrics such as BLEU and ROUGE are not sufficient to evaluate open-ended text generation tasks. More and more studies rely on using human evaluators to determine the final quality of generated text. However, fully using human evaluation instead of automatic evaluation is expensive and inconvenient in practice.

We introduce **Perception score**, which is a simple automatic evaluation metric for text generation built on the top of fake text detector. We report the accuracy of detecting fake text as the Perception score. The assumption of this approach is that the pretrained language models such as BERT have acquired plenty of prior knowledge during pretraining. We can use them to detect not only the statistical difference, but also the semantic errors of the generated text. If it has low accuracy in detecting generated text, it means that the real and generated text have very similar distribution. Also, to reduce variance, we only train the discriminator with one epoch on a fixed dataset, where the generated samples are acquired from greedy sampling.
Figure 2: An example of story ending generation in TextGAIL. D is the discriminator. G is the generator.

for deterministic outputs.

However, the drawback for this method is that it relies on the researchers themselves to train this fake text detector. Its result can be manipulated easily and misleading. It is better to use it as a self-rating rather than a metric for comparison. However, while Perception score is still a preliminary version of the metric and has limitations, we believe it merits further exploration to fill in the blank of text evaluation metrics.

5 Experiments

We first introduce the dataset, and the automatic evaluation metrics we used to evaluate TextGAIL. Then we describe the implementation details.

5.1 Dataset

We evaluate TextGAIL on the ROCStories dataset (Mostafazadeh et al., 2016). The task is to generate an open-ended ending for a four-sentence story. Different from unconditional or other more constrained text generation tasks, remembering the training set will not lead to good performance in this task, as every story is unique. In ROCStories, the training set contains 98,162 five-sentence stories. The fifth sentence in the story is considered to be the ending of the story. Both the validation set and the test set have 1,871 samples. Unlike the training set, these stories have another wrong ending constructed adversarially by human. These adversarial endings are fluent and relevant but contain commonsense mistakes. We show an example in Figure 2, where the blue box is the context, the green one is the right ending, and the yellow one is wrong ending. We also demonstrate if the yellow is generated by the generator, how the discriminator would assign the reward.

5.2 Automatic Evaluation Metrics

Because multiple story ending can be appropriate under the same story context, n-gram-based reference matching metrics such as BLEU and ROUGE cannot reflect the generation quality. Therefore, rather than using the traditional automatic evaluation metrics, we decide to evaluate both the overall quality and diversity of the generation using three metrics: Perception Score, sentence level repetition and unique n-gram.

**Perception score (Perception):** We use a large pretrained language model RoBERTa-base as a classifier to distinguish between the real and generated samples. We train the classifier for one epoch on the validation set, and use greedy sampling to sample outputs from the generator. Then we deploy the classifier to the test set, and report the accuracy as the final score. The resulting score approximately reflects the similarity of the real and generated distributions.

**Sentence-level repetition (Sent-REP 2-grams):** We use sentence-level repeated 2-grams to evaluate the repetition in the corpus, which usually accounts for the dull and generic sentences. We first collect all the 2-grams that appear more than once in the training set. Then we count the number of occurrences of those 2-grams in the generated story endings. Sent-REP 2-grams can capture the sentence-level repetition, as those repetitive and generic sentences patterns often occur more than once in the corpus. Finally, we report the result normalized with the total number of the 2-grams in all the story endings.

**Unique n-grams (Unique n-grams):** We also follow Xu et al. (2018) to use the number of unique n-grams to evaluate language diversity. We report the number of unique 1-gram and unique 2-grams.

5.3 Implementation Details

TextGAIL takes the advantage of large pretrained language models. The generator uses the GPT-2 base (117M parameters) model, while the discriminator uses the RoBERTa-base (125M parameters) model. One can use any other pretrained language model for the generator and discriminator. We choose these two models because they share the same vocabulary, which saves us the effort from extra tokenization. We warm up the generator with 20% of data in the training set. We apply nucleus sampling (Holtzman et al., 2019) in decoding to avoid low probability words being sampled. Dur-
### Results and Analysis

We first show the automatic evaluation, and then the human evaluation results. In both results, TextGAIL performs better compared against MLE. Finally, we conduct case study and error analysis to interpret the results from TextGAIL.

#### 6.1 Automatic Evaluation Results

We want to test if TextGAIL is better than simply fine-tuning GPT-2 with MLE in the story ending generation task. As mentioned previously, n-gram match metrics scores such as BLEU are not applicable. We focus on reporting the Perception score as an approximate quality metric, and the three diversity metrics: Sent-Rep 2-grams, Unique 1-gram, and Unique 2-grams. For Perception score, we run the experiments five times, and for other metrics, three times to report the mean and variance.

The results are shown in Table 1. TextGAIL performs well in terms of diversity, which is reflected by the three diversity metrics. Our results are considered even more diverse than the ground truth written by human. This suggests that TextGAIL has utilized the pretrained language models to alleviate the exposure bias problem. However, we still need other metrics to ensure the overall quality such as logic is not affected. We use Perception Score to approximately measure the similarity between the real and generated samples distributions. TextGAIL achieves lower score than MLE, meaning that the TextGAIL generated outputs are more similar to the human written ground truth. Theoretically, human written story endings would have Perception Score 0.5, which is the lower bound.

The overall automatic evaluation results suggest that TextGAIL generations are more diverse and similar to the human written ground truth.

For the ablation study, we replace the pretrained discriminator RoBERTa-base with another randomly initialized discriminator. Though the adversarial training process is still stable, we observe severe performance drop which is also reflected by the automatic evaluation scores. This observation matches some previous criticisms that GAN generated text is worse than MLE. Through our experiments, we suspect that this is possibly due to the fact that without pretraining, the discriminator does not have sufficient prior knowledge to distinguish the real and generated samples from the generator, and therefore does not provide helpful reward signal to the generator. Different from the previous text GANs, our method exploits the large pretrained language models and achieves much better performance than MLE.

#### 6.2 Human Evaluation Results

As automatic evaluation metrics are limited, we conduct human evaluation here. We randomly select 40 samples from the test set. Each sample contains one story body and two endings. We ask Amazon Mechanical Turkers to vote which ending fits the story body better. To guarantee Turkers voting quality, we also mix 10 human-written right endings and wrong endings pairs from the validation set, where the answers are obvious for this task. If the workers answer any one of those pairs wrong, we will exclude their results. In the end,
Example 1

| Context: | Example 2 |
|----------|-----------|
| I wanted to buy a video game console. I asked my parents, and they came up with an idea. They said if I did my chores, I would be given money to save. I did my chores without being asked every week for a whole summer. | Ben went to the DMV to get his License. The instructor gave Ben a passing grade at the end. Excited, Ben calls up his father to tell him the good news. Ben father never picked up, he died in a car accident that day. |

| Ground Truth: | MLE: |
|--------------|------|
| My parents gave me enough money to buy the console. | (1) Now I have the video game console I asked for. (2) It was an awesome idea. (3) The next week, I had to buy a new gaming console! |

| TextGAIL: |
|-----------|
| (1) I bought a PlayStation 4 to play with my parents. (2) I was so happy when my parents gave me a Wii U. (3) When I got my console, I played my favorite video games. |

| Ground Truth: | Ben was devastated. |
|--------------|--------------------|
| Ben regretted going to the DMV. | Ben regrets going to the DMV. |
| Ben mourns the loss of his father but also the passing of a great man. | Ben was happy that his dad was alive. |
| It seems like too much to bear. |

Table 3: Examples of generated results from MLE and TextGAIL. We sample three samples to reduce bias.

we collect 10 unique and valid votes for each test sample, which results in 400 votes in total.

As shown in Table 2, TextGAIL significantly outperforms GPT-2 fine-tuned with MLE, by having 158% more votes. This proves TextGAIL can not only generate more diverse story endings, but also achieves better overall quality judged by humans. It also suggests that TextGAIL achieves better commonsense reasoning than MLE. In the Case Study Section, we show what leads to the huge improvement over MLE. We also compare the results of TextGAIL with ground truth ending that is written by humans. Nevertheless, even though TextGAIL improves the text generation quality compared to MLE, human written endings are still better (52.20%). In the error analysis section, we demonstrate the main discrepancy that contributes to our model’s imperfection.

6.3 Case Study

To better understand the difference between MLE and TextGAIL outputs, we carefully analyze the generated results. Here, we choose two example stories and generate their endings by randomly sample three times with nucleus sampling and to reduce observation bias. The two example stories are shown in Table 3. These two examples showcase the two main types of errors from MLE.

One major problem of MLE’s output is being generic and repetitive, which is also referred as neural text degeneration (Welleck et al., 2019). In Example 1, MLE generated outputs lack of details and are universal. The generated output (2) appears in other story contexts. Meanwhile, TextGAIL can generate detailed named entities such as "PlayStation 4" and "Wii U", and have better logic than MLE. Surprisingly, those named entities never appeared in the training data. They probably come from the large-scale corpus in pretraining GPT-2 and RoBERTa.

Another problem of MLE is the lack of commonsense. The Example 2 reflects this problem. All three MLE generated outputs do not make sense to humans. They either are against the factual information in the context, or do not follow the progression of the story. In contrast, the three TextGAIL generated samples appear to be much more logical. This difference suggests that the discriminator can distinguish the real and generated samples by using commonsense. Such commonsense later helps to improve the generated endings’ quality. In Section 7, we specifically analyze the discriminator’s capability of common sense reasoning after adversarial training.

6.4 Error Analysis

However, we need to be aware of TextGAIL’s limitations, as it is not perfect in terms of following commonsense. We can also observe its limitation from human evaluation results when comparing against the ground truth. We have conducted error analysis of those failed results from human
Ari spends $20 a day on pickles. He decides to make his own to save money. He puts the pickles in brine. Ari waits 2 weeks for his pickles to get sour.

Right Ending: Ari opens the jar to find perfect pickles.  
Generated Ending: When his pickles are rotten, he dumps it.

Table 4: An example of generated endings that contain commonsense error.

evaluation. In Table 4, an example of generated endings that contain commonsense error is shown. For human, we know that pickles’ taste is sour, but it does not mean it is rotten. However, the machine cannot correctly understand the difference between “sour” and “rotten”, but probably based on its learned prior knowledge, it makes the association and generate the wrong ending. “Sour” sometimes does have association with “rotten”, but here “sour” is a property of “pickles”. This result shows that commonsense reasoning is still a very hard task in text generation. We will examine more direct approaches such as using knowledge graph to tackle this problem in the future.

7 Unsupervised Commonsense Reasoning

In order to verify our hypothesis that the discriminator learns the difference between the real and generated samples by using commonsense reasoning, we evaluate our learned discriminator on the ROCStories’ correct ending prediction task. As mentioned, the test set contains two candidate endings: one is right and the other is wrong. The difference is that the wrong ending contains commonsense error. For this reason, this dataset is often used for commonsense reasoning. We use the learned discriminator to classify which one is the right one. Ideally, the discriminator has captured the commonsense errors to distinguish real samples from generated samples, and therefore it should tell the difference on a separate data. We compare our discriminator with two baselines. One baseline, GPT-2+MLE, uses a language model to assign a probability to each ending to see which one has a higher probability (Trinh and Le, 2018). Here, we use GPT-2 fine-tuned with MLE as the language model. The other baseline, RoBERTa+Fine-tune, is to train RoBERTa as a text classifier on the validation set with 1,872 pairs of right and wrong endings, and then predict on the test set. The results are shown in Table 5:

| Models                                | Supervised? | Accuracy(%) |
|---------------------------------------|-------------|-------------|
| RoBERTa+Fine-tune w/ extra data       | ✔           | 92.8 ± 0.29 |
| GPT-2 + MLE*                          |             | 69.6 ± 0.35 |
| TextGAIL discriminator                |             | 79.1 ± 0.76 |
| TextGAIL discriminator with random weights |             | 51.2 ± 0.85 |

Table 5: Results of unsupervised commonsense classification. *We use the language model’s sentence joint probability to do classification.

criminator can accomplish story ending prediction without any supervision to a certain degree. Our model achieves 79.1% accuracy which is much better than the other unsupervised baseline GPT-2+MLE (69.6%). However, our model still is not as good as supervised method such as RoBERTa+Fine-tune (92.8%). We also test the discriminator with randomly initialized weights instead of RoBERTa, and find that without the pretrained language model, it is impossible to accomplish unsupervised commonsense reasoning (51.2%). The results are consistent with our hypothesis that the discriminator uses commonsense from the pretrained RoBERTa to distinguish the real and generated samples.

8 Conclusion

Despite the success on image generation, GANs have not yet achieved considerable progress on text generation. One of the difficulties is the instability of applying GANs on discrete data. Also, we observe that another reason may be the lack of sufficient language priors for the discriminator.

To tackle those problems, we propose a novel adversarial training framework - TextGAIL, which leverages the large pretrained language models. We further utilize generative adversarial imitation learning, proximal policy optimization, and efficient use of human demonstrations to solve the instability problem of generator’s training. Also, we introduce a new automatic evaluation metric Perception score which uses the RoBERTa as a fake text classifier to assess the quality of generated sentences. Experiment shows that TextGAIL can generate not only more diverse, but more logical and reasonable text. This paper extends the exploration of adversarial training on text generation. We plan to apply TextGAN to other text generation tasks such dialogue generation and text summarization in the future.
References

Yossha Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA, pages 932–938. MIT Press.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. CoRR, abs/1811.02549.

Tong Che, Yanran Li, Ruixiang Zhang, R. Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. CoRR, abs/1702.07983.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial networks. CoRR, abs/1406.2661.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 5141–5148. AAAI Press.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 4565–4573.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. CoRR, abs/1904.09751.

Pei Ke, Fei Huang, Minlie Huang, and Xiaoyan Zhu. 2019. ARAML: A stable adversarial training framework for text generation. CoRR, abs/1908.07195.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. GANS for sequences of discrete elements with the gumbel-softmax distribution. CoRR, abs/1611.04051.

Kevin Lin, Dianqi Li, Xiaodong He, Ming-Ting Sun, and Zhengyou Zhang. 2017. Adversarial ranking for language generation. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pages 3155–3165.

Min Lin. 2017. Softmax GAN. CoRR, abs/1704.06191.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. CoRR, abs/1907.11692.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. CoRR, abs/1604.01696.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos Santos, Çağlar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016, pages 280–290. ACL.

Weili Nie, Nina Narodytska, and Ankit Patel. 2019. Relgan: Relational generative adversarial networks for text generation. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Tom Le Paine, Çağlar Gülçehre, Bobak Shahriari, Misha Denil, Matthew D. Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil C. Rabinowitz, Duncan Williams, Gabriel Barth-Maron, Ziyu Wang, Nando de Freitas, and Worlds Team. 2019. Making efficient use of demonstrations to solve hard exploration problems. CoRR, abs/1909.01387.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA, pages 311–318. ACL.

Alec Radford and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislaau Bööntö, and Sergey Levine. 2018. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In 2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018, pages 3758–3765. IEEE.

Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.
Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010, volume 9 of JMLR Proceedings, pages 661–668. JMLR.org.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of JMLR Workshop and Conference Proceedings, pages 1889–1897. JMLR.org.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. CoRR, abs/1707.06347.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In (Singh and Markovitch, 2017), pages 3295–3301.

Satinder P. Singh and Shaul Markovitch, editors. 2017. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pages 3104–3112.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. CoRR, abs/1806.02847.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. CoRR, abs/1908.04319.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8:229–256.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 1296–1306. The Association for Computational Linguistics.

Jingjing Xu, Xu Sun, Xuancheng Ren, Junyang Lin, Bingzhen Wei, and Wei Li. 2018. DP-GAN: diversity-promoting generative adversarial network for generating informative and diversified text. CoRR, abs/1802.01345.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In (Singh and Markovitch, 2017), pages 2852–2858.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. CoRR, abs/1905.12616.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum entropy inverse reinforcement learning. In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008, pages 1433–1438. AAAI Press.