Text Generation using Generative Adversarial Training

Xuerong Xiao
Department of Electrical Engineering
Stanford University
xuerong@stanford.edu

Abstract

Generative models reduce the need of acquiring laborious labeling for the dataset. Text generation techniques can be applied for improving language models, machine translation, summarization, and captioning. This project experiments on different recurrent neural network models to build generative adversarial networks for generating texts from noise. The trained generator is capable of producing sentences with certain level of grammar and logic.

1 Introduction

Training and sampling from generative models is a great test of the ability to represent high-dimensional probability distributions. Current supervised deep learning models have shown excellent results in various areas including natural language processing and computer vision. However, large amount of annotations and labeling can sometimes be difficult to obtain. Unsupervised or semi-supervised learning have now drawn a lot of research attention because they can reduce or eliminate the number of labels. Generative models can work with multimodal outputs and can be trained with missing data and can provide predictions on missing data [1].

Recurrent neural networks (RNNs) based models have been used widely for generative tasks such as language modeling, machine translation, speech recognition, and image captioning. Being able to capture important features in the time series is the most notable ability of the RNNs. In the generative RNN models, words from the previous time steps are input to the next time step iteratively. However, due to the fact that only training data is seen by the model during training, the statistics of the hidden states during optimization may be shifted during sampling. As a result, the generated sentences may not be grammatically correct.

Generative adversarial networks (GANs) have been successfully applied in computer vision [2-4] to generate high-resolution and realistic images. In a GANs framework, there are two adversaries, a discriminator and a generator, playing against each other in a game. Both players are represented by a differentiable function with a set of parameters. The generator forges samples that are intended to be from the same distribution as the training data. The discriminator is a binary classifier striving to distinguish the generated fake samples from the real ones.

Despite the tremendous interest of using GANs for image generation, the counterparts in natural language processing have not been comparable. In this project, we ask the question of whether the adversarial training strategy in GANs would be applied to or improve the generation of texts. We investigate different recurrent architectures of the generator and the discriminator. We evaluate the model based on the grammar of the generated sentences. The generator in GANs is initialized by training a generator in standard supervised training. The generated texts preserve certain level of grammar structures and logic.

2 Related Work

There are different kinds of deep generative models that choose the parameters to maximize the likelihood of the training data. GANs can be made to do so [1] and has some advantage over other generative models. Compared with Boltzmann machines and nonlinear independent component analysis, GANs have few restrictions on the generator function, and no Markov chains are needed. Variational autoencoders performs maximum likelihood inference on the global parameters and variational inference on the latent variables. Although a generative method, VAE makes weak assumptions about the model and can learn parameters efficiently via back propagation [5]. Compared with variational autoencoders, GANs were designed to be unbiased and the generated samples are of better quality, at least in the realms of vision.

RNNs possess both a complex internal state representation and nonlinear transition functions. RNN-based language models can capture long-term dependencies [6]. We train the GANs using gated recurrent units (GRUs) as both the generator and the discriminator, compared with vanilla RNNs. Unlike image data, text generation is inherently discrete, which makes the gradient from the discriminator difficult to back-propagate to the generator. In this project, we provide the discriminator with the intermediate hidden vectors of the generator to make the network differentiable, similar to [7].

Professor Forcing [7] is a recently published technique of training RNNs using an adversarial objective. One of their conclusions is that adversarial objective has a regularizing effect and helps convergence in RNN training. The discriminator looks at the statistics of the behavior, forcing the generator to behave the same when it is constrained by the data and when it is left generating outputs alone. SeqGAN [8] models the data generator as a stochastic policy in reinforcement learning, bypassing the generator differentiation problem by directly performing the gradient policy update. The reward signal comes from the discriminator judged on a complete sequence and passed to the intermediate state-action steps using Monte Carlo search. Wasserstein GAN (WGAN) [9] provides rigorous theoretical analysis and improves the performance of GANs using the Earth Mover (EM) distance. It minimizes a reasonable and efficient approximation of the EM distance. Training WGANs and does not require careful design of the network architecture. Although addressed by WGAN, another difficulty in training GANs is investigated in this project. That is, maintaining the balance in training of the discriminator and the generator to prevent one from overpowering the other.

3 Approach

3.1 Training setup

In this project, we primarily use GRUs as building blocks and use vanilla RNNs as comparisons. GRUs are computationally cheaper than LSTMs but have comparable performance in modeling sequences. The operational stages in a GRU are (1) update gate $z_t = \sigma(W_z x_t + U_z h_{t-1})$, (2) reset gate $r_t = \sigma(W_r x_t + U_r h_{t-1})$, (3) new memory $\widetilde{h_t} = \tanh(r_t \circ U h_{t-1} + W x_t)$, and (4) hidden state $h_t = (1 - z_t) \circ \widetilde{h_t} + z_t \circ h_{t-1}$.

At each time step of the recurrent model, the network takes x_t from the input sequence and y_t from the output sequence from the previous step, and updates the hidden state h_t as a function of the previous hidden state h_{t-1} and of (x_t, y_t). It then computes a probability distribution over the next element conditioned on the previous elements. Discrete output is generated by adding a softmax layer on top of the hidden state. The output sequence y is generated by the generator (parametrized by θ_g) with input sequence x according to the distribution $P_{\theta_g}(y|x)$.

Word outputs from the generator are not differentiable because they are discrete. However, GANs require differentiability of the generators. In this project, similar to what is described in [7], we attempt to add a summarizing function $B\ (x, y, \theta_g)$ on top of the hidden states of

the generator and of the input embedding, and the output of the function is the input to the discriminator (parametrized by θ_d). The summarizing function is differentiable and carries the gradient back to the generator. The model during training is illustrated in Figure 1 below.

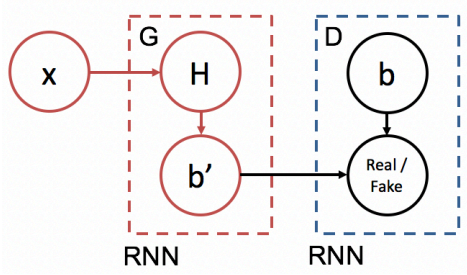

Figure 1: GAN model during training. x: input embedding; G: generator; H: hidden states of generator; b': output of B on H; D: discriminator; b: output of B on x and y from training

The discriminator is trained to maximize the likelihood of correctly distinguish between the outputs of the summarizing function on the input embedding and on the hidden states. The loss function is [7]:

$$J^{(D)} = -\frac{1}{2}E_{(x,y)\sim data}\left[\log\left(D\left(B\left(x,y,\theta_g\right),\theta_d\right)\right)\right] - \frac{1}{2}E_{y\sim P_{\theta_g}}\left[\log\left(1-D\left(B\left(x,y,\theta_g\right),\theta_d\right)\right)\right] \quad (1)$$

Instead of a zero-sum game, the loss function of the generator is:

$$J^{(G)} = -\frac{1}{2}E_{(x,y)\sim data}\left[\log\left(P_{\theta_g}\right)\right] \quad (2)$$

3.2 Training process

A standard supervised training is used to initialize the generator in the GANs. Input sequences are fed into a generative RNN with the same architecture as the generator. This initializing generator aims to predict the next element in the current time step, hence its label is the same sequence but shifted one time step ahead.

When the training loss of the generator is below a certain value, the training of the generator is paused, and the training of the discriminator starts. When the training loss of the discriminator is below a certain value, the training of the discriminator is paused, and the training of the generator starts. The process is repeated until a certain number of epochs is reached.

Both the generator and the discriminator are either composed by GRUs or regular RNNs in different experiments. The generator and the discriminator have comparable size. The discriminator has one fully connect layer on top of the hidden states, and a sigmoid layer to output probabilities. Both networks are trained with minibatch stochastic gradient descent via Adam optimization. All the work is implemented in TensorFlow.

4 Experiments

4.1 Dataset

Project Gutenberg is a collection of free electronic books. We download 4 of those English books and combine them into a dataset. The prepared dataset contains 20k or 30k words for training, and 5k words held out for testing. The sentences are shuffled before the dataset splitting. The models are word-level models. In preprocessing, miscellaneous symbols are removed.

4.2 Hyperparameter tuning

We experiment with different combinations of GRUs and RNNs for both the generator and the discriminator, with different hidden sizes. GRUs generally work better with better text generation. We also experiment with the size of the training data. It contains either 20k words or 30k words. The number of layers of the recurrent models are fixed to be 2. The batch size is 50 for both networks. The sequence length is 30. The learning rate starts at 0.002 with a decay rate of 0.9. Adam optimization determines the adaptive learning rate and the momentum. Generator training pauses when loss is below 0.5, and discriminator training pauses when its loss is below 0.3. All training stops at 50 epochs.

In WGAN, weights are clamped to a fixed box after each gradient update. In this project, we clip the gradients at 5 to achieve similar goals. We engineer the cost function of the generator to avoid gradient vanishing problem as in the zero-sum game. We use one-sided label smoothing [1] to avoid gradient exploding in the generator.

During test phase, the noise input to the generator is sampled from the conditional distribution produced by the RNN at each time step. We also experiment with this latent size. The generated results, together with the results from seq2seq models with similar RNN architectures as the generator, are summarized in Table 1.

4.3 Results

For the best performing model (GRUs with 30k training words, 256 hidden units and 20 latent size), the training process is illustrated in the loss of the supervised generator, the loss of the generator in GAN and the loss of the discriminator, as in Figure 2 and Figure 3 below.

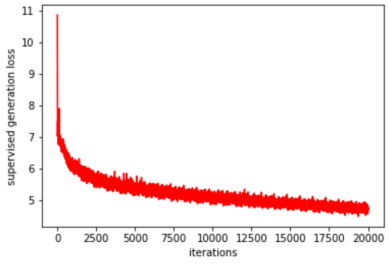

Figure 2: Supervised training loss of a generator used to initialize the generator in GANs

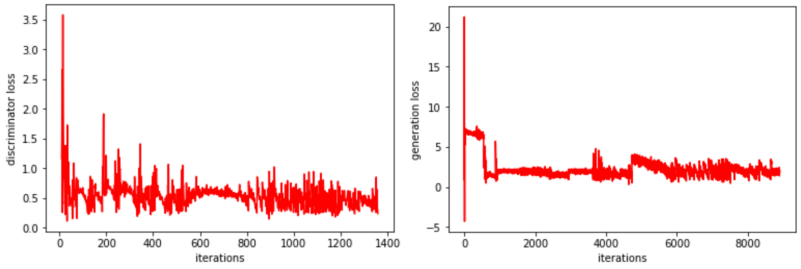

Figure 2: Training loss of the discriminator and the generator in GANs

The training of the supervised model converges. The loss curves of the discriminator and generator in GANs are not as smooth due to loss clipping.

Table 1: Sample text generation in different architectures

G	D	Vocab size	Hidden size	Latent size	Generated examples from GAN	Generated examples from seq2seq
RNN	RNN	20k	256	10	... for thaw presided . in thaw course oesophagus thaw yawned , 1787, thaw unit statements ...	…recognized lesion lesion glycosuria caustics caustics caustics…
GRU	RNN	20k	256	10	... and thaw emperor warts nostrils titles be titles be able titles be titles be titles ...	…requirements pleased mentioned prove as header suspend brotherly mihiel …
RNN	GRU	20k	256	10	... he warts nostrils titles be able titles restoring his brows and thaw sam timber he had nostrils yeomen ...	…needles razor charity thirtieth issued blunt an volition oesophagus a portfolio…
GRU	GRU	20k	256	10	... thaw conservative oesophagus thaw sentences at thaw american programs warts accomplished ...	…his brilliant and , and thaw . tunic and thaw wept thaw oesophagus plantation . tunic…
GRU	GRU	30k	256	10	... what was the signs of the european war fenchurch street . thank you . you are set and follows are advice in your hand ...	…was he 's mary churches and he california to nowhere , to unconcerned…
GRU	GRU	30k	128	10	... the incubation period is a pure complicated degree of lymphangitis ...	…the prisoners of was of . of apologize arranged the rebels wearing of judge , was equally same distribution that…
GRU	GRU	30k	256	20	... from giving a of fenchurch street. thank you you have made your statement very very very much to remember ...	…get into vouchsafe from defending from happening from helped…

From Table 1, we see that architectures with GRUs generally perform better than those with RNNs. The generation from GANs generally perform better than those without adversarial training. With larger training data, the generated texts from GANs are more structured. With smaller hidden size, the influence is not very obvious. This is also the case with different latent sizes, since the other parameters already lead to fair generation.

5 Conclusion

This projects aims at building text generators using generative adversarial networks. The investigated discriminators and generators are recurrent models, and the generator is initialized by a model trained via supervised learning. The generated texts from GANs can have correct grammar and logic in some models and are generally better than the texts

generated without adversarial training. The changes from regular RNNs to GRUs and from smaller to larger training set give rise to most notable improvement on generated texts. More experiments need to be conducted to draw conclusions on the influence of hidden sizes and latent sizes.

In the future, the training set needs to be larger to achieve better results, since the current training set is still small compared to published work on text generation. Also, more hyperparameter searching can be performed. For instance, CNNs, bi-directional RNNs, or current models with more fully connected layers and nonlinearities can be used for discriminators. In order to improve the current model, GANs can be combined with a good language model to eliminate the repetitiveness in the generated texts. Moreover, a more quantitative metrics needs to be implemented to better evaluate the results. Reinforcement learning and WGAN can also be further studied and incorporated into text generation.

References

[1] Goodfellow, I. J. (2016). NIPS 2016 Tutorial: Generative adversarial networks.

[2] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014b). Generative adversarial networks. In *NIPS*'2014.

[3] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[4] Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., and Metaxas, D. (2016). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1612.03242*.

[5] Kingma, D. P., and Welling, M. (2014). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)* 2014.

[6] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, September 26-30, 2010, pp. 1045-1048.

[7] Lamb, A., Goyal, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor Forcing: A new algorithm for training recurrent networks. ArXiv e-prints, Oct. 2016.

[8] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2016). SeqGAN: Sequence generative adversarial nets with policy gradient. ArXiv e-prints, Dec. 2016.

[9] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. ArXiv e-prints, Jan. 2017.