Adversarial Text Generation Without Reinforcement Learning

David Donahue
University of Massachusetts Lowell
david_donahue@student.uml.edu

Anna Rumshisky
University of Massachusetts Lowell
arum@cs.uml.edu

Abstract

Generative Adversarial Networks (GANs) have experienced a recent surge in popularity, performing competitively in a variety of tasks, especially in computer vision. However, GAN training has shown limited success in natural language processing. This is largely because sequences of text are discrete, and thus gradients cannot propagate from the discriminator to the generator. Recent solutions use reinforcement learning to propagate approximate gradients to the generator, but this is inefficient to train. We propose to utilize an autoencoder to learn a low-dimensional representation of sentences. A GAN is then trained to generate its own vectors in this space, which decode to realistic utterances. We report both random and interpolated samples from the generator. Visualization of sentence vectors indicate our model correctly learns the latent space of the autoencoder. Both human ratings and BLEU scores show that our model generates realistic text against competitive baselines.

1 Introduction

Over the past several years, deep learning models have provided large performance gains on many tasks requiring language generation, from machine translation (Johnson et al., 2016) to dialogue agents (Serban et al., 2016) to summarization (Rush et al., 2015) and question answering (Weissenborn et al., 2017). A recent 2018 survey paper (Gatt and Krahmer, 2018) includes the discussion of neural approaches to natural language generation (NLG) in three out of four chapters dedicated to the current state-of-the-art methods.

The probabilistic neural language model (NLM) is one important example. NLMs have long been utilized for language generation, by predicting sequence probabilities from learned word representations (Bengio et al., 2001). These models generate text without being conditioned on any input, by outputting a distribution over the vocabulary at each time step which is sampled and given as input at the next time step. More recently, variational autoencoders (VAEs) have been used for sentence generation (Bowman et al., 2015). VAE models enforce a prior distribution on the latent output of the encoder, to smooth the space. Random points selected in this space then decode to valid sentences. However, the latent space is not always uniform (Makhzani et al., 2015), and generated examples cannot be conditioned on input features.

Generative Adversarial Networks were recently proposed as a method for image generation (Goodfellow et al., 2014). GANs have been very successful in computer vision, where they have been applied to a variety of tasks from image captioning (Zhang et al., 2017a) to image super-resolution (Ledig et al., 2016). Interestingly, while GANs have shown exceptional promise for generating realistic data, applying them to text has proved very difficult, largely because text is discrete, and thus gradients cannot propagate from the discriminator to the generator.

Developing methods that overcome this obstacle and leverage GANs for text generation is the focus of this paper. Autoregressive models such as the RNN produce a sequence one token at time, by sampling from a generated distribution over the vocabulary at each time step. This sampling occurs at the final layer of the model. However, introducing variation at the final layer can hinder higher-level sentence planning (Serban et al., 2017). In contrast, GANs insert variation starting from the input layer, which encourages the model to generate in a top-down manner. This is one motivation for the long-term application of GANs to NLG. In addition, Adversarial training of a GAN happens at the sequence level, instead of at the word level. This may encourage greater textual

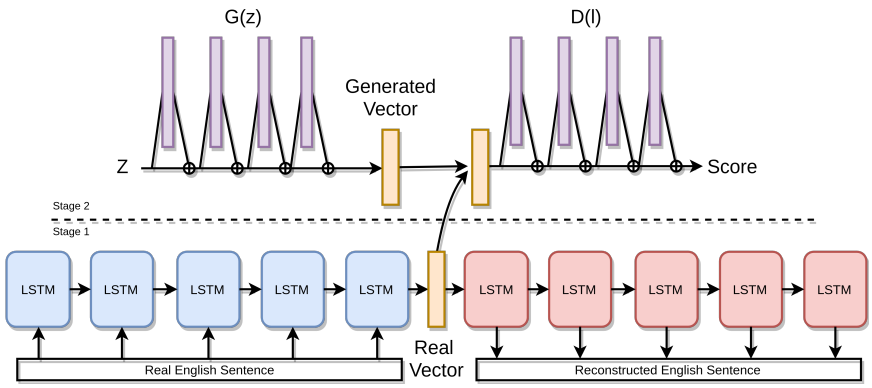

Figure 1: LaTextGAN model architecture. The discriminator network $D(l)$ receives sentence representations produced by the fully-trained autoencoder, and from the generator network $G(z)$.

coherence.

It is not currently possible to train a GAN to produce text directly using standard optimization, as text is discrete and thus gradients cannot be passed from discriminator to generator during training. To overcome optimization difficulties involving the discrete text output of the generator, Yu et al. (2017) utilize reinforcement learning by directly applying policy gradients to the generator. Zhang et al. (2017b) use a soft-argmax approximation with a convolutional network discriminator to smooth policy gradients to the generator, and pretrain the discriminator on sentence permutations to speed-up convergence. Li et al. (2017) train a discriminator to score only partially decoded sequences. To reduce the instability of sequence prediction with recurrent neural networks, Lamb et al. (2016) use an adversarial network to encourage similar system behavior during training and prediction. In that work, the adversarial model acts as a training regularizer for a RNN decoder, but is not used during prediction.

Given recent developments in the literature, we propose a Generative Adversarial Network model for sentence generation which does not require reinforcement learning. To overcome the discrete nature of text, we utilize an autoencoder (AE) to encode sentences into smooth sentence representations. A generator network is then trained to generate its own sentence representations in the learned latent space. Each sentence vector produced by the generator is then passed through the AE decoder, which decodes to the nearest sentence. We evaluate our system against multiple baselines and show our generated sentences score well on both human and automatic methods.

2 LaTextGAN for Sequence Generation

We introduce LaTextGAN (latent-space GAN for text) for the purpose of generating discrete sequences. In this paper, we focus specifically on the construction and application of LaTextGAN to the unconditional generation of English sentences. Figure 1 contains a diagram of our proposed model. We utilize an autoencoder component which learns a dense low-dimensional representation of text. A generator network is utilized to produce additional points in this latent variable space, which decode to valid sentences. As is typical for Generative Adversarial Networks, a discriminator network is trained to classify real and generated sentences from their latent representations. The generator attempts to fool the discriminator by generating more realistic sentence representations.

2.1 Textual Autoencoder

Autoencoders are designed to learn a low-dimensional representation of text by using an encoder network to compress information about each sentence into a finite vector. A decoder network is tasked with reconstructing the input representation from the vector. We utilize a Long-Short Term Memory (LSTM) network for both the encoder and decoder (Hochreiter and Schmidhuber, 1997). The LSTM network reads each sentence sequentially, one word at a time.

During sentence reconstruction, the decoder takes the encoder latent representation and the previous hidden state as input and produces a probability distribution which is used to select the word at that time-step. We use greedy sampling in our autoencoder, and select the highest probabil-

ity word at each timestep.

2.2 GAN Architecture Overview

It seems natural to model both the generator and discriminator using standard fully-connected networks. However, randomly-initialized fully-connected layers are notoriously harder to train as layer depth increases. To mitigate gradient instability associated with these networks, we instead represent the generator and discriminator each using a ResNet architecture (He et al., 2016).

2.3 Training Procedure

To aid in training quality, we adopt the Improved Wasserstein GAN network from Gulrajani et al. (2017), a modification of the original Wasserstein GAN (Arjovsky et al., 2017) which formulates the training objective as:

$$\max_{\theta} E_{z\ p(z)}[f_w(g_\theta(z)] - E_{x\ p(x)}[f_w(x)] \quad (1)$$

for discriminator (critic) f_w and generator g_θ. Gulrajani et al. (2017) also apply this training objective to large ResNet architectures, reassuring their applicability to this task.

3 Evaluation

3.1 Toronto Book Corpus

While large corpora are usually biased toward particular domains, books offer a variety of genres and dialects. While it is impossible to obtain sentences that exactly match the distribution of the English language (datasets are always biased toward particular domains), books offer a wide variety of sentences from different genres and dialects. Characters have different backgrounds and appear in numerous environments or time periods. For this reason, we elected to train LaTextGAN on the Toronto Book Corpus, a collection of books known for both sentence quality and quantity (Zhu et al., 2015). We select two million sentences from the corpus for our training set.

We use the neural language model (NLM) and the variational autoencoder (VAE) as baselines.

3.2 Human Discriminators

Evaluation of generative models remains a difficult endeavor. Traditionally, sentence quality has been evaluated with metrics such as METEOR and BLEU score (Papineni et al., 2002). However, these metrics often fail to evaluate the higher-level meaning of generated sentences. We propose an empirical evaluation using humans as discriminators, to differentiate between generated sentences and sentences from the dataset. To perform the evaluation, a pair of real and generated sentences is presented in random order to the participant. The most "realistic" sentence is selected from the pair, or both are selected as realistic. If both are selected as about equally realistic, this counts as a draw. See Table 2 for results.

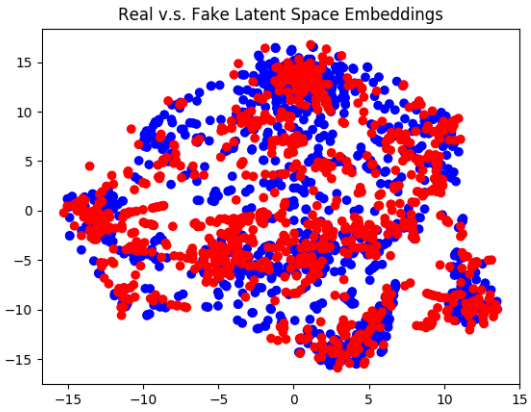

Figure 2: Plot of GAN-generated sentence vectors (red) and genuine sentence vectors (blue) produced by the autoencoder. Dimensionality reduced using t-SNE.

3.3 BLEU Score

For automatic evaluation, we calculate the BLEU-4 score of system-generated sentences against a validation set of 10,000 held-out sentences from the Toronto corpus. The variational autoencoder performed best on this metric, followed by LaTextGAN and the neural language model.

3.4 Plotted Sentences

A quality generator should produce sentence representations that lie in similar neighborhoods to real sentences within the autoencoder-learned latent space. To examine how our generator output compares, we plot both real and generated sentences using t-SNE (Maaten and Hinton, 2008). To examine the input space of our generator, we select two random points within the input z-space of the generator, and decode sentence vectors at evenly spaced intervals between them. Points that are close in the latent space should decode to similar sentences.

LaTextGAN Sampled Sentences	LaTextGAN Interpolations in Input Space
He lifted his hand.	It was as late when everyone else was waiting.
He arrived at the subject.	They told them we ate all.
The music hit by an instant and released her hair.	They knew how soon these.
You ve always been in the world to get my friends.	They found them then.
The clock pulled his head away.	You keep them then.
Jill pointed at the door and stared at them blankly .	They found her hands.
Egwene finished by the dwarf.	They sat beside her.
He knew she would be able to keep her alive.	They sat on captain.
I thought you would be there.	He looked at her.
The words he could feel.	He watched her.

Table 1: Left: LaTextGAN sentences decoded using random Gaussian vectors as input. Right: LaTextGAN sentences decoded by moving linearly through input space.

Model	More Realistic	Less Realistic	Equally Realistic	BLEU Score
LaTextGAN	**13.9%**	55.6%	30.5%	0.678
NLM	12.2%	46.6%	**41.2%**	0.643
VAE	6.0%	81.6%	12.4%	**0.688**

Table 2: Human evaluation of model-generated sentences as more realistic, less realistic, or equally realistic with respect to real English sentences. BLEU-4 score calculated on a held-out validation set.

4 Discussion

The LaTextGAN system scored the highest percentage of sentences that were rated as more realistic than a given English sentence from the dataset. However, the neural language model scored a higher draw rate. Judging by the dataset quality, it is possible that LaTextGAN is more likely to generate high quality sentences, while the neural language model is more likely to generate decent sentences that the evaluators deemed about as real as sentences from the dataset (medium quality, high probability). While the VAE had the highest BLEU score, it was not competitive with the other models for human-rated realism. Our system performs well on both human and automatic evaluation.

Unlike VAEs, the LaTextGAN input space is forced to be a Gaussian distribution, providing theoretical guarantees for the validity of high-probability regions of the space. We show sentences sampled from this space in Table 1. Interpolated sentences reveal smooth transformations from one sentence to another. However, not all adjacent sentences are semantically similar (e.g. "They sat on captain" and "He looked at her") even if they share similar syntactic structure. Improving the GAN input space still remains one possible research direction. An encoder which maps generated sentence vectors back into Gaussian vectors could improve the smoothness and potential applications of the space (Ulyanov et al., 2017).

The structure of LaTextGAN allows for more interesting evaluation techniques. Figure 2 contains sentence embeddings produced by our model, alongside genuine sentence embeddings encoded using the autoencoder. LaTextGAN-generated sentence embeddings cover every region of high-probability. This provides evidence that LaTextGAN does not suffer from mode-collapse as is often observed in GAN models.

5 Conclusion

We propose LaTextGAN as a new generative adversarial architecture for text, capable of being trained without reinforcement signals of any kind. We demonstrate that our GAN performs well on both human evaluation and BLEU-4 scoring, and achieves the highest percentage of sentences which rate higher quality than real sentences sampled from the dataset. Plotted embeddings indicate that LaTextGAN has properly modeled the latent space of a trained autoencoder, while sentence interpolations show that the LaTextGAN generator smoothly encodes sentences in the input space. We expect this model to be useful in downstream tasks such as dialogue generation where it would potentially increase response diversity.

In short, reinforcement learning poses a significant barrier-to-entry for the application of GAN models in natural language generation. We introduce a model which removes this barrier, with the aim of inspiring more widespread use of GANs outside of computer vision.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2016. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

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 *AAAI*, pages 3295–3301.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2017. It takes (only) two: Adversarial generator-encoder networks. *arXiv preprint arXiv:1704.02304*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural qa as simple as possible but not simpler. *arXiv preprint arXiv:1703.04816*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient.

Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. 2017a. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE Int. Conf. Comput. Vision (ICCV)*, pages 5907–5915.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017b. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*.

A Model Implementation

Here we report model parameters and implementation decisions non-central to the paper. It is well-documented that GANs suffer from a number of convergence problems (Goodfellow et al., 2014). To solve these issues, Arjovsky et al. (2017) propose the Wasserstein GAN (WGAN), which uses the Earth-Mover distance metric for optimization, and clips the discriminator weights during training. They show the Earth-Mover distance provides stable gradients for all points in the solution space. Gulrajani et al. (2017) introduce a regularization of the WGAN which encourages the norm of discriminator gradients to approach unity. They show the regularization increases the capacity of larger models, including ResNets. We choose to implement a WGAN model with this regularization to avoid convergence issues associated with the vanilla GAN architecture.

Empirically, we choose the cell sizes for the encoder and decoder LSTMs of our autoencoder to be 100 and 600, respectively. We apply a dropout of 0.5 to the encoder output during training. The application of dropout improved the quality of generated sentences. Input words to the encoder are represented by 200-dimensional word embeddings, learned during training.

ResNet architectures can be deeper than standard fully-connected layers. We implement the generator and discriminator ResNets using 40 layers each. Each layer is of the form $F(x) = H(x)+x$ where $H(x)$ is the learned residual layer function. Note that for a learned fully-connected function $G(x)$, a residual layer can learn $H(x) = G(x) - x$, giving residual layers an equivalent capacity to standard dense layers. In both generator and discriminator layers, we choose to implement each residual layer as

$$H(x) = relu(x \cdot W_1 + b_1) \cdot W_2 + b_2 \quad (2)$$

for learned weight matrices W_1, W_2 and biases b_1, b_2. For simplicity, all layers have the same dimension of 100.

We implement our NLM as an LSTM decoder network with the same size (600-dimensional cell state) and configuration as the decoder of our autoencoder network, for fair comparison. We sample words from their probabilities at each timestep to decode each full sequence. For our variational autoencoder, we select the encoder and decoder LSTM networks to be 600-dimensional, and apply KL annealing to enhance training (Bowman et al., 2015). 200-dimensional word embeddings are learned as input. We use learning rates of 10^{-3} and $5{\cdot}10^{-4}$ for the NLM and VAE respectively and train for 5 epochs.

As is suggested in the paper, we update the discriminator N times per update to the generator. In practice, we cannot train a fully optimal discriminator, so we use N=10. Adam optimizer is used for optimization of all models for its fast convergence properties (Kingma and Ba, 2014). We use a learning rate $\alpha = 5 \cdot 10^{-4}$ for the autoencoder and $\alpha = 1 \cdot 10^{-4}$ for both the generator and discriminator. We train the autoencoder and WGAN for 5 and 15 epochs, respectively. For the stochastic input to the generator $p(z)$, we sample vectors from a multivariate Gaussian distribution.

To improve the quality of our data, we filter vocabulary words which appear less than 5 times and remove all sentences which contain these rare words. Sentences are restricted to a maximum length of 20 words.

For human evaluation, we assembled 1,000 sentence pairs for human evaluation (one third per model), and selected two unaffiliated evaluators to label sentence pairs. Each pair was labeled once. The order of pairs were randomized, and the real and generated sentence within each pair was randomized. All pairs were completed in a few hours, highlighting this method as a feasible human evaluation for future generative models. BLEU-4 score was calculated using a held-out validation set of 10,000 sentences, and was calculated using weights (0.25, 0.25, 0.25, 0.25) for 1, 2, 3, and 4-grams respectively. BLEU scores were calculated for 1000 sentences generated from each model, and averaged.

To create sentence interpolations, two Gaussian vectors v_1 and v_2 were sampled as starting points in the input space. We then sample N-1 points linearly between them, each point calculated as

$$v_i = v_1 + (v_2 - v_1)/N * i \quad (3)$$

for all intermediate points $i = 1, 2, 3...N-1$. Each of these points is used as input to the generator and decoded to produce a sentence.