CS230

Unsupervised Text Generation Using Generative Adversarial Networks

Andrew E. Freeman
Department of Statistics
Stanford University
aefree@stanford.edu

Anna C. Shors
Department of Statistics
Stanford University
ashors@stanford.edu

Anastasios Sapalidis
Department of Statistics
Stanford University
tsap@stanford.edu

Abstract

Despite their success in other fields, General Adversarial Networks have gained little traction in the field of text generation due to the need to choose words to generate at each timestep. This discrete "picking" function poses a challenge, as it prevents gradients from being propagated from the discriminator to the generator. In this paper, we explore an alternative method of using GANs for text generation in which the generator works to directly output sentence encodings that can be decoded using a pretrained decoder. While our generated sentences lack the fluency of the language model baseline, we show that this method has the potential to generate creative, realistic sentences and would benefit from further exploration in future works.

1 Introduction

Generative Adversarial Networks (GANs) have become increasingly common in the field of computer vision due to their ability to generate novel and realistic images. With the growing popularity of open-ended text generation, it is natural to ask whether the methods used to generate images can be extended to generate realistic bodies of text. Unfortunately, GANs have seen little success in the field of text generation due to the discrete nature of text. The function that maps from the space of vocabulary words to a single word is non-differentiable, which poses a challenge during GAN training when the gradients need to pass from the discriminator to the generator.

Recent research on using GANs for text generation has focused on using reinforcement learning and policy gradient methods to train the generator (25) (23). An alternate approach consists of training a generator to produce sentence encodings and using a pretrained RNN decoder to decode the encodings into human readable sentences (7). In this paper, we explore this method for text generation.

This technique consists of first pretraining an autoencoder and then training a GAN to generate sentence encodings that are indistinguishable from the encodings generated by the autoencoder from real text. When training the GAN, the autoencoder's encoder can be used to generate "real" sentence encodings. These real encodings, as well as the encodings generated by the generator, are fed as input into the discriminator, whose goal is to distinguish the real encodings from the fake ones. Once the

CS230: Deep Learning, Winter 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

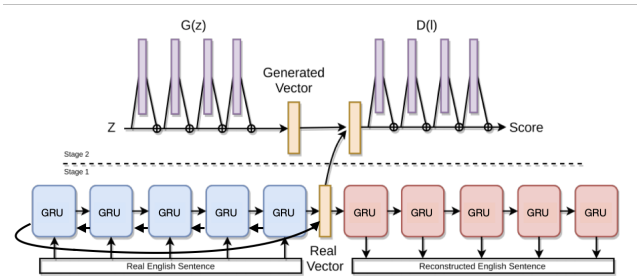

Figure 1: GAN + autoencoder architecture adapted from (7)

GAN training is complete, the generator can be used to generate sentence encodings which can then be decoded using the autoencoder's pretrained decoder. We outline this architecture in Figure 1.

2 Related work

2.1 Language Modeling

Language models are commonly used for text generation. A language model outputs the probability of each word in the vocabulary conditioned on a sequence of previous words. A number of variations of recurrent neural network architectures (12) (6) have achieved notable performance on text generation tasks. The introduction of transformer architectures (24) further improved on these results, and deep stacked transformer decoders such as GPT-2 and GPT-3 (20) (4) have been shown to generate remarkably realistic texts in a variety of domains.

2.2 Wasserstein GANs

Traditional GANs are notoriously difficult to train. The Wasserstein GAN (2), or WGAN, has been proposed as a more stable alternative to the traditional GAN. WGANs are trained to minimize the Wasserstein distance between the true data distribution and the distribution learned by the generator. This minimization problem is easier to optimize than the corresponding GAN minimization problem and has been shown to yield better results on a variety of tasks. A number of further WGAN improvements have been proposed, including the introduction of the gradient penalty (10). We make use of the WGAN and gradient penalty during model training.

2.3 LaTextGANs

We draw largely from (7) and (1) for our implementation. (7) trains a WGAN to directly output sentence encodings using the Toronto Book Corpus dataset and limits their sentences to a maximum length of 20 words. (1) explores a number of variations of the LaTextGAN architecture and reports promising results on the SNLI and COCO datasets.

3 Dataset and Features

We use the Wikipedia movie plot dataset from Kaggle (14) to train both the autoencoder and the GAN. This dataset contains approximately 35,000 movie summaries, comprising over 600,000 sentences, scraped from Wikipedia. To simplify our auotencoder and GAN models, we use standalone sentences rather than maintaining a structure of sequential sentences. We remove any parenthetical comments in the sentences in order to simplify the sentence structure. From this cleaned set, we split our data into train, dev, and test sets using 4,253 sentences each for the dev and test sets. At train time, we remove any non-alphabetic characters aside from trailing punctuation and commas to further simplify the model. Finally, we tokenize the sentences using the Keras Tokenizer and a vocabulary size of 20,000 words.

4 Methods

We take inspiration primarily from the methods presented by Donahue and Rumshisky (7). We first pre-train a simple RNN autoencoder on our movie summaries dataset. We train the encoder to generate a dense context vector from a given sentence and train the decoder to recover the original sentence from this context vector. Because we need the encoder and decoder to work independently of one another, we do not use attention for this model. We use categorical crossentropy loss for each word generated by the decoder and define the loss function for a given sentence to be the average crossentropy of each word in the generated output.

Once the autoencoder is trained, we train our GAN. Following the methods proposed in (7), we use a Wasserstein GAN, rather than a standard GAN, for more stable training (2). Both the generator and discriminator (called the "critic" in the case of WGANs) consist of simple series of ResNet blocks. The GAN critic takes as input a sentence encoding (either a real encoding generated by the encoder, or a fake encoding generated by the generator) and outputs a real number that describes the "level of realness" of the encoding. The critic loss function, $\mathcal{J}^{(D)}$, is described in Equation 1, where f_w refers to the critic, and g refers to the generator.

$$\mathcal{J}^{(D)} = -\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) + \frac{1}{m}\sum_{i=1}^{m} f_w(g(z^{(i)})). \tag{1}$$

The goal of the critic is thus to output large positive values when given real sentence encodings and large negative values when given fake sentence encodings. In contrast, the goal of the generator is to make the critic predict that the fake sentences are as "real" as possible. The generator loss, $\mathcal{J}^{(G)}$, is described in Equation 2.

$$\mathcal{J}^{(G)} = -\frac{1}{m}\sum_{i=1}^{m} f_w(g(z^{(i)})). \tag{2}$$

After the GAN has been trained, we pass the encodings generated by the generator into the decoder to create human-readable sentences.[1]

5 Results and Discussion

5.1 Autoencoder Performance

We train the autoencoder on a training set of 561,710 sentences with a maximum sentence length of 30 words. This autoencoder consists of a bidirectional GRU encoder and a standard GRU decoder. We use 512 hidden units for the encoder and concatenate the forward and backward encoder hidden states to create a 1024-dimensional vector that is used as input into the decoder. We train using the Adam optimizer with learning rate $\alpha = 10^{-3}$ for 2 epochs. Table 4 in the Appendix summarizes the performance of this autoencoder on 2,000 sentences in the validation set.

5.2 GAN Results

Because there has been little research on the optimal hyperparameters for text GANs, we evaluate model performance using a number of hyperparameter combinations. Following the advice of (10), models are trained using the Adam optimizer with a learning rate $\alpha = 10^{-4}$, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. We also add a gradient penalty with strength $\lambda = 10$ to the critic loss. We vary the number of layers in both the generator and critic, the number of critic updates per generator update, and the maximum sentence length of our training set. Because training GANs is notoriously difficult, we hoped that exploring a variety of architectures would allow us to find a combination that works well for our particular task. Additionally, we were interested in investigating how varying the maximum sentence length would affect performance, as previous models (7) (1) typically use relatively low maximum sentence lengths, resulting in quite simple generated sentences. We compare our results against a language model baseline, consisting of a simple LSTM architecture with attention [2]. The BLEU scores for these models are shown in Table 1. Following the methods from (7), our base model has 40 layers in both the generator and critic and trains the generator once for every 10 critic training iterations.

Hyperparameter Tests				
Hyperparameters	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Base Model	85.4	**61.3**	**21.4**	**1.5**
10 gen layers / 10 critic layers	84.4	55.8	14.8	0.6
20 gen layers / 20 critic layers	85.6	57.4	19.3	1.1
60 gen layers / 60 critic layers	85.4	56.2	16.7	0.8
40 gen layers / 10 critic layers	85.5	60.2	15.5	0.0
10 gen layers / 40 critic layers	86.7	59.9	21.4	0.9
5 critic iters / 1 gen iter	87.1	59.2	20.9	1.4
15 critic iters / 1 gen iter	**87.6**	59.2	18.4	1.0
max sent len 20 words	83.7	52.2	12.8	0.7
max sent len 10 words	70.4	34.7	4.7	0.0
RNN Baseline	**94.6**	**84.4**	**66.9**	**40.5**

Table 1: BLEU scores for sentences generated using various hyperparameters. The Base Model uses 40 generator and critic ResNet layers, 10 critic iterations per generator iteration, and a max sentence length of 30 words. For each row label in the hyperparameter column, the rest of the hyperparameters are equivalent to those in the base model.

We see that the base model has the highest BLEU-2, BLEU-3, and BLEU-4 scores of the GAN-based architectures, though this model still lags in quality relative to the RNN baseline. However, we note that while BLEU provides a quantitative measure of evaluating sentence quality, it is incomplete as an evaluation metric. Another useful method to determine how well our generator is learning the real data distribution is visualizing the distributions using UMAP, a dimensionality reduction tool similar to t-SNE. These UMAP plots indicate that, while the 40 layer base model achieves the highest BLEU score, it is unable to effectively learn the true data distribution and tends to get stuck outputting sentences that follow a very predictable syntactical structure that is easily separated from the real sentences. In contrast, the 10-layer model is able to better approach the distribution of the real vectors. Further, it produces sentences that are within the distribution of real vectors and are therefore more difficult to distinguish from real sentences by the critic. The UMAP plots of these two models are shown in Figure 2. We refer to Table 5 in the Appendix for a further comparison of sentences generated by our base model with sentences generated by the 10-layer model.

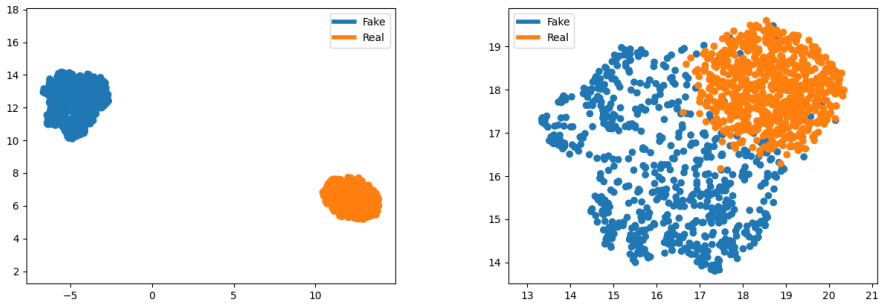

Figure 2: UMAP plots for a WGAN with 40 generator and critic layers (left) and a WGAN with 10 generator and critic layers (right).

Additionally, we evaluate our models using human evaluation metrics. For the reasons described above, we choose to use the 10-layer model in our human evaluations. Evaluators were asked to rate sentences based on their fluency using a Likert scale, with a score of 1 being "undecipherable" and a score of 5 being "perfectly coherent." Results of the human evaluations are shown in Table 2.

While our text GAN does not perform as well as the RNN baseline, it is important to note that the sentences generated by our model, while often lacking in coherent content, do generally match the syntax and structure of true sentences. To demonstrate this fact, we provide a sample of sentences generated by the GAN along with a sample of sentences generated by the RNN baseline in Table 3.

[1]Model code: github.com/tassossapalidis/latextgan

[2]Baseline code: https://github.com/minimaxir/textgenrnn

Human Evaluations	
Text Source	Fluency Score
Real Text	4.43
RNN Baseline	3.08
GAN	1.61

Table 2: Average fluency score from 30 human evaluators. 90 sentences were generated from each model. Each evaluator was shown 15 sentences from each model, with each sentence being shown 5 times across all evaluators.

RNN Baseline
He is then attacked by indians, and they attempt to escape.
8 years later, the young boy, is a simple and western person.
Jerry helps quacker and sally, but are getting arrested.
GAN
To melinda falls aboard bartok when escapes the two gun, they warns it to be police.
The lover, tony is attracted with jeffrey, who realizing his parents mother seeing bruce <unk>.
Tamura, engages to two henchmen, and sends the bandits <unk>, steps in a man with richard simon.

Table 3: Sentences generated from a simple word-level RNN with attention (top) vs. sentences generated by our 10-layer GAN (bottom)

Another pattern to note is that we are often unable to train our models to optimality. It has been shown that Wasserstein GANs have much more stable loss curves than traditional GANs and that, in contrast to standard GANs, WGAN loss is correlated with generation quality. (2) shows that for image generation tasks, the generator loss tends to start relatively high and proceeds to converge to 0. As the loss decreases, image quality increases. Our models exhibit analogous behavior at the beginning of training, but the generator loss becomes volatile before converging, leading to a decrease in generated sentence quality. The loss curves associated with this phenomenon are displayed in Figure 3 in the Appendix. This behavior indicates that the model architectures investigated here may not be ideal for this task and that more exploration into GAN model architectures for text generation tasks is necessary.

6 Conclusion and Future Work

We have demonstrated the potential of a GAN architecture that circumvents the discreteness problem of GAN usage in text generation and avoids the use of reinforcement learning techniques. While our model does underperform relative to a traditional attention-based RNN, it is able to generate sentences with some degree of structure, coherence, and consistent context. We hypothesize that training instability is the primary cause of the limitations of this model's output; other authors studying networks with this architecture were successful in stably training their models for more iterations than we were able to reach with our model (7) (1). We conclude that we have demonstrated the plausibility of using GANs for text generation tasks, but further research must be done on stabilizing the training of these text generation GANs. In particular, we believe future work should focus on optimizing the loss penalty to maintain the 1-Lipschitz constraint required for theoretical convergence while allowing for more stable empirical convergence.

We conclude with a theoretical motivation for expanding this work from generating stand-alone sentences to generating bodies of text. One could implement a conditional model, where the generator conditions on the $t-1$ previously generated context vectors when generating a sentence vector at timestep t. This process could continue until the generator outputs an "end of sequence" token, at which point the entire sequence of context vectors could be passed to the critic. The loss functions in this model would be analogous to those in equations 1 and 2, but the critic would instead make its predictions on sequences of sentences, rather than individual sentences. This could allow for the generation of complete, novel stories. We were unable to explore this idea in this work due to time constraints, but we hope to do so in the future.

7 Contributions

All members of the team generally contributed to all aspects of the project, but each member was more involved in certain areas than others. Anna Shors took the lead on initial topic research, wrote the majority of the model training code, and handled much of the hyperparameter research and tuning. Andrew Freeman was also heavily involved in hyperparameter research and tuning, while taking the lead on data preprocessing, creating a system for human evaluations, and creating visualizations of model results. Anastasios Sapalidis is responsible for quantitative model metrics, writing much of the model code, management of the GitHub repository, and code cleaning.

References

[1] AKMAL HAIDAR, M., REZAGHOLIZADEH, M., DO-OMRI, A., AND RASHID, A. Latent Code and Text-based Generative Adversarial Networks for Soft-text Generation. *arXiv e-prints* (Apr. 2019), arXiv:1904.07293.

[2] ARJOVSKY, M., CHINTALA, S., AND BOTTOU, L. Wasserstein GAN. *arXiv e-prints* (Jan. 2017), arXiv:1701.07875.

[3] BAMMAN, D., O'CONNOR, B., AND SMITH, N. A. Learning Latent Personas of Film Characters. *ACL 2013, Sofia, Bulgaria* (Aug. 2013).

[4] BROWN, T. B., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AGARWAL, S., HERBERT-VOSS, A., KRUEGER, G., HENIGHAN, T., CHILD, R., RAMESH, A., ZIEGLER, D. M., WU, J., WINTER, C., HESSE, C., CHEN, M., SIGLER, E., LITWIN, M., GRAY, S., CHESS, B., CLARK, J., BERNER, C., MCCANDLISH, S., RADFORD, A., SUTSKEVER, I., AND AMODEI, D. Language Models are Few-Shot Learners. *arXiv e-prints* (May 2020), arXiv:2005.14165.

[5] CHINTAPALLI, K. Generative Adversarial Networks for Text Generation — Part 3: non-RL methods, Jun 2019.

[6] CHO, K., VAN MERRIENBOER, B., GULCEHRE, C., BAHDANAU, D., BOUGARES, F., SCHWENK, H., AND BENGIO, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv e-prints* (June 2014), arXiv:1406.1078.

[7] DONAHUE, D., AND RUMSHISKY, A. Adversarial Text Generation Without Reinforcement Learning. *arXiv e-prints* (Oct. 2018), arXiv:1810.06640.

[8] FAN, A., LEWIS, M., AND DAUPHIN, Y. Hierarchical Neural Story Generation. *arXiv e-prints* (May 2018), arXiv:1805.04833.

[9] FANG, L., ZENG, T., LIU, C., BO, L., DONG, W., AND CHEN, C. Outline to Story: Fine-grained Controllable Story Generation from Cascaded Events. *arXiv e-prints* (Jan. 2021), arXiv:2101.00822.

[10] GULRAJANI, I., AHMED, F., ARJOVSKY, M., DUMOULIN, V., AND COURVILLE, A. Improved Training of Wasserstein GANs. *arXiv e-prints* (Mar. 2017), arXiv:1704.00028.

[11] HERRERA-GONZÁLEZ, B., GELBUKH, A., AND CALVO, H. Automatic Story Generation: State of the Art and Recent Trends. *Advances in Computational Intelligence 19th Mexican International Conference on Artificial Intelligence* (Oct. 2020).

[12] HOCHREITER, S., AND SCHMIDHUBER, J. Long Short-term Memory. *Neural computation 9* (12 1997), 1735–80.

[13] HUSZÁR, F. How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary? *arXiv e-prints* (Nov. 2015), arXiv:1511.05101.

[14] JUSTINR. Wikipedia movie plots. `kaggle.com/jrobischon/wikipedia-movie-plots/metadata`, 10 2018.

[15] LI, Y., GAN, Z., SHEN, Y., LIU, J., CHENG, Y., WU, Y., CARIN, L., CARLSON, D., AND GAO, J. StoryGAN: A Sequential Conditional GAN for Story Visualization. *arXiv e-prints* (Dec. 2018), arXiv:1812.02784.

[16] LUONG, M.-T., PHAM, H., AND MANNING, C. D. Effective Approaches to Attention-based Neural Machine Translation. *arXiv e-prints* (Aug. 2015), arXiv:1508.04025.

[17] MINIMAXIR. textgenrnn. `github.com/minimaxir/textgenrnn`, 2020.

[18] MIRZA, M., AND OSINDERO, S. Conditional Generative Adversarial Nets. *arXiv e-prints* (Nov. 2014), arXiv:1411.1784.

[19] PRANAVPSV. GPT2 genre-based story generator. github.com/pranavpsv/Genre-Based-Story-Generator, 2021.

[20] RADFORD, A., WU, J., CHILD, R., LUAN, D., AMODEI, D., AND SUTSKEVER, I. Language models are unsupervised multitask learners.

[21] SAEED, A., ILIĆ, S., AND ZANGERLE, E. Creative GANs for generating poems, lyrics, and metaphors. *arXiv e-prints* (Sept. 2019), arXiv:1909.09534.

[22] SHREYDESAI. Implementation of adversarial text generation without reinforcement learning. github.com/shreydesai/latext-gan, 2019.

[23] SUTTON, R. S., MCALLESTER, D., SINGH, S., AND MANSOUR, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems* (2000), S. Solla, T. Leen, and K. Müller, Eds., vol. 12, MIT Press.

[24] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention Is All You Need. *arXiv e-prints* (June 2017), arXiv:1706.03762.

[25] YU, L., ZHANG, W., WANG, J., AND YU, Y. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv e-prints* (Sept. 2016), arXiv:1609.05473.

[26] ZHU, Y., LU, S., ZHENG, L., GUO, J., ZHANG, W., WANG, J., AND YU, Y. Texygen: A Benchmarking Platform for Text Generation Models. *arXiv e-prints* (Feb. 2018), arXiv:1802.01886.

8 Appendix

Autoencoder Performance

Autoencoder Performance	
BLEU Metric	Score
BLEU-1	86.2
BLEU-2	81.2
BLEU-3	77.2
BLEU-4	73.8

Table 4: Averaged BLEU scores for 2,000 encoded/decoded sentences from the validation set

Sample Sentences

Table 5 provides a sample of sentence generated by the base model and a sample of sentences generated by the GAN with 10 layers for both the generator and critic. Notice that the structure of the sentences in the 40-layer base model is quite predictable. In contrast, the sentences generated by the 10-layer model vary more in structure and appear more creative than those from the base model.

Base GAN model
It is further affair by harry to <unk> desires the inebriated murderer to order to lili dan.
To vote to the carrier john and alter meets the wrong person to the us eventually escort.
To daylight to <unk> snyder, edmund is regarded and revives him to place the tracks brutally.
As she to be followed to anton to leaves lucy with wilder s father the rangers in kensington mexico.
Yamamoto upon to kwan at gary as zhang is recovers with attacked, goes with the loose blood.
10-layer GAN
When anupama was raghu he initially dismissed from when eva, child vanishes, and meanwhile saves anbu in constable.
To brinda comes that fights to agree and harry believes jerry will end as frank hesitates.
The jackal swims insane suspicious saito, and frustration, shooting mary demanding the deal to life people.
When the nerdy night gifts to knife, princess and virginia, tommy with danny gun to <unk>.
That and haddad, is are halted the house when the village is helping, she destroys village when suddenly it.

Table 5: Sentences generated using the 40-layer base GAN (top) vs. sentences generated by the 10-layer GAN (bottom)

Loss Curves

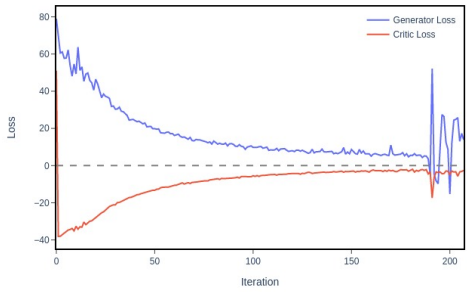

Figure 3: Generator and critic loss during training of the base model