Text Generation using Relational Memory and Transformer Discriminators

Adam Paslawski, Justin Wu, Ariyo Vahdat
University of Toronto

Abstract

Applying GANs to natural language generation is a challenging issue that continues to be explored today. For our final project, we propose an extension to the RelGAN model [9] for the task of language generation. Our proposed extension is to replace the CNN discriminator with a transformer. The results of our study found no significant improvement in our model compared to RelGAN.

1 Introduction

Language generation is a task that has been of great interest in the domain of unsupervised learning. Generative adversarial networks (GANs) are a two-network system that generate novel instances of data given a dataset and have seen great success in computer vision [5, 2]. However, GANs struggle to generate discrete sequences like sentences due to the differentiability issue with discrete data. The objective of our study is to improve upon previous iterations of text generating GANs by modifying the architecture of the discriminator network. In this paper we propose architectural changes to RelGAN [9] by using a transformer for the discriminator to improve the coherence of generated novel sentences.

2 Related Works

2.1 Differentiability Issue

Using GANs for representational learning on discretized data like text has historically been difficult due to an indifferentiable loss landscape. Backpropogation requires a differentiable loss function and discrete data structures can have arbitrarily large changes in the loss landscape for a given perturbation of the data. As a result, solutions need to take the discretized text and transform it to a continuous space. Solutions have included: score based gradient methods and the Gumbel-Softmax distribution [4]. No solutions that we could find currently address this issue in a directly robust manner[3].

2.2 GANs and Text Generation

Seq-GAN [17] was one of the first attempts to extend GANs to the task of text generation. It used a policy gradient method to overcome the differentiability issue with an RNN generator and CNN as a discriminator. SeqGAN performed well on short Chinese poems but showed limited success on longer sequences. This was later improved upon by TextGan, which used a soft-argmax function to overcome the differentiability issue and did so with a smaller variance amongst gradient estimates. Leveraging a CNN discriminator and an LSTM generator, it showed improvements over Seq-GAN [17] as measured by their BLEU scores [13], but did not significantly improve the coherence of generated novel sentences. RelGAN [9] quantitatively improved upon its predecessors by including a discriminator architecture with multiple CNN's to capture multiple aspects of the input sentence.

3 Method

3.1 RelGAN

RelGAN provides three major contributions to natural language generation using GANs. First, it makes use of a relational memory architecture [12] in the generator, replacing the traditionally used LSTM architecture. Relational memory compartmentalizes information as a fixed number of memory slots, and allows interaction between these memory slots with the self-attention mechanism [12]. Secondly, RelGAN solves the issue of non-differentiability in GANs by using Gumbel-Softmax relaxation [9]. Thirdly, RelGAN adds a tunable parameter called inverse temperature which allows the trade-off between sample quality and diversity to be controlled.

RelGAN's generator is defined as such. We say M_t is the model's memory at time t, and we assume that each of its rows is a memory slot. Given H heads, we have H sets of queries, keys, and values. The queries, keys, and values are defined as $Q_t^{(h)} = M_t W_q^{(h)}$, $K_t^{(h)} = [M_t; x_t] W_k^{(h)}$, and $V_t^{(h)} = [M_t; x_t] W_v^{(h)}$ where [;] is a row-wise concatenation. The updated memory $\tilde{M}_{t+1}$ is

$$\tilde{M}_{t+1} = [\tilde{M}_{t+1}^{(1)} : ... : \tilde{M}_{t+1}^{(H)}], \tilde{M}_{t+1}^{(h)} = \sigma(\frac{Q_t^{(h)} K_t^{(h)T}}{\sqrt{d_k}}) V_t^{(h)}$$

$\sigma(\cdot)$ denotes the softmax function, performed row-wise, d_k is the column dimension of key $K_t^{(h)}$, and [:] denotes column-wise concatenation. [9]

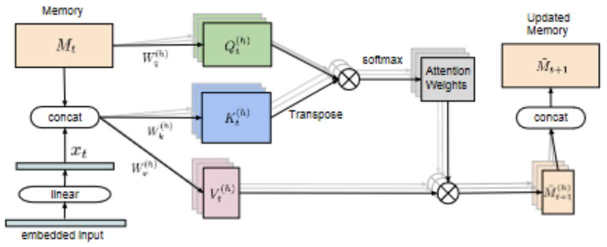


Figure 1: Memory Updating Mechanism using Self-Attention (from [9])

RelGAN also proposes a novel discriminator architecture. Multiple embedding vectors of input sentences are generated and passed into a series of convolutional layers. The motivation behind this approach is that the embedding vectors should capture different features in the input data [9]. Finally, fully connected layers are used to transform the CNN outputs to a single logit.

3.2 Proposed Extensions

Our primary extension to the RelGAN model is a modification to the discriminator architecture. Replacing the CNN architecture used in RelGAN, we propose a transformer-based architecture. Like in the CNN discriminator, either a one-hot encoded sentence from the dataset or the softmax activations from the generator are passed to the discriminator as input. This sequence of vectors are linearly transformed into a sequence of encoding vectors. To give the transformer context about positional information, we do element-wise addition on these encoding vectors with positional encoding vectors. These positional encoding vectors are constructed using sin and cos functions, as specified by Vaswani et al. [14]. We pass these encoding vectors through several transformer encoder layers with multihead attention. Between each transformer encoder layer, we use layer normalization, as used by Wei Yu et al. [16] The output of the transformer module is flattened and passed through several fully connected layers with ReLU activation, followed by a single sigmoid output logit.

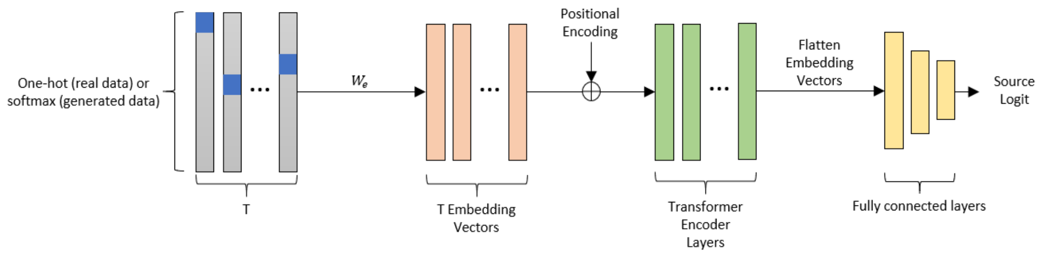

One-hot (real data) or softmax (generated data)
T
W_e
T Embedding Vectors
Positional Encoding
Transformer Encoder Layers
Flatten Embedding Vectors
Fully connected layers
Source Logit

Figure 2: Proposed Discriminator Architecture

Convolutional layers, which are used in RelGAN, are good at capturing local interactions [16], but have difficulty with global interactions. The motivation behind using transformers in the discriminator is to leverage its ability to draw global dependencies [14]. We posit that in sequential data such as text, these global dependencies are more relevant than local features in predicting if data is real or synthetic

Additionally, TextGAN and RelGAN extract several features from the input sentences using convolution, before feeding them into fully connected layers [15, 9]. We also use this idea of extracting features from the sentences, except instead of using convolution, we use multihead-attention [14]. Each head corresponds to some feature that can be extracted from the embedded input and intermediate transformer outputs.

Lastly, we made changes to the discriminator training by adding a gradient penalty to the generator loss. The gradient penalty is minimized when the norm of the discriminator gradients is 1 [6]. This gradient penalty is meant to improve training stability by preventing exploding and vanishing gradients [6].

3.3 Training Techniques

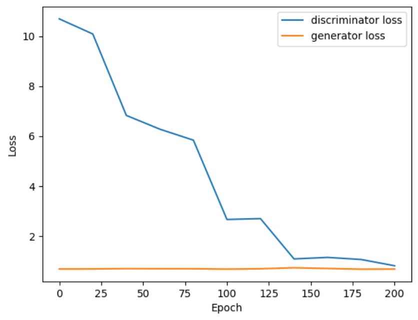
The GAN loss function used for training both RelGAN and OurGAN is Relativistic standard GAN (RSGAN) [7], which was found to be the most effective GAN loss for training RelGAN (in comparison to standard GAN loss [5] and hinge loss [10], [18]) [9]. The generators of both models were pre-trained using MLE for several epochs.

4 Experiments

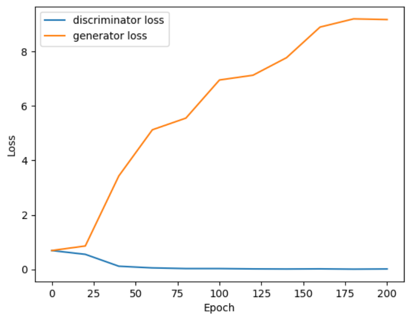
Identical tests were performed on both RelGAN as well as the extended model. For consistency and to accurately compare the results obtained in this study with a baseline, the trials are a re-creation of the trials described by Nie et. al. The dataset is the COCO image captions dataset [1], and the evaluation metric is BLEU scores [11]. Also, to save on compute resources and training time, we used a truncated version of the COCO dataset, with 5,000 training and 5,000 test samples.

Model	BLEU-2	BLEU-3	BLEU-4	BLEU-5
RelGAN	0.736	0.488	0.283	0.166
OurGAN	0.71	0.465	0.282	0.177

Figure 3: RelGAN and OurGAN BLEU scores after 200 epochs



(a) OurGAN



(b) RelGAN

Figure 4: Generator and discriminator losses over training

OurGAN saw a much higher initial discriminator loss that decreased over the training process. This is likely due to the gradient penalty applied to the discriminator. Given random initialization, the initial value of the gradient penalty is expected to be high; we cannot draw meaningful conclusions by comparing discriminator loss. However, note that the generator losses trend differently between OurGAN and RelGAN. OurGAN's curve suggests the generator is not being challenged too much, while RelGAN's curve suggests the generator is struggling to keep up with its discriminator.

5 Discussion

5.1 Experiment Results

Our hypothesis for our experiments was an increase in coherence of sentences by using a transformer attention mechanism to capture more aspects of input sentences. Our findings did not show a significant improvement in performance qualitatively (Figure 5) or quantitatively (Figure 3) over the results of RelGAN.

RelGAN
a broken white kitchen with a ceiling fan .
a bathroom that has white towels on the floor .
a kitten is laying on a motorcycle in a small clear day .

(a) RelGAN Generated Sentences

OurGAN
man in a kitchen preparing at a light gathering out
a bathroom with a tub and a tan sink and need head .
a crowd of bathroom being remolded floor and sink .

(b) OurGAN Generated Sentences

Figure 5: OurGAN and RelGAN Generated Sentences

However, our experiments are proof of concept that using multi-headed attention in the discriminator is capable of similar performance to RelGAN. Limited access to compute may have restrained the performance of our model. Further experimentation with larger datasets and larger compute and further hyper parameter tuning could result in OurGAN outperforming RelGANgan in the future.

6 Conclusion

In this paper we proposed an architectural extension to RelGAN to improve the coherence of generated sentences. By replacing the CNN based discriminator architecture with a transformer we leveraged multi-headed attention to increase our models ability to learn multiple aspects of sentences in the Coco dataset. The result was a proof of concept for multi-headed attention in text generating GANs. We saw an insignificant improvement in performance as measured by the BLEU score of our model with no significant improvement qualitatively to sentence coherence. For future work we would like to experiment with larger computational resources to better take advantage of the transformer architecture and improve generator performance.

References

[1] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server, 2015.

[2] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks, 2015.

[3] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Adversarially regularized autoencoders, 2015.

[4] Ben Poole Eric Jang, Shixiang Gu. Categorical reparameterization with gumbel-softmax, 2017.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[6] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[7] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missingfrom standard gan, 2018.

[8] William Lam. Textgan-pytorch, 2021.

[9] Weili Nie, Nina Narodytska, and Ankit Patel. RelGAN: Relational generative adversarial networks for text generation. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJedV3R5tm`.

[10] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers usingvariational divergence minimization, 2016.

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://www.aclweb.org/anthology/P02-1040`.

[12] Adam Santoro, Ryan Faulkner, David Raposo, Jack W. Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy P. Lillicrap. Relational recurrent neural networks. *CoRR*, abs/1806.01822, 2018. URL `http://arxiv.org/abs/1806.01822`.

[13] Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. Bleurt: Learning robust metrics for text generation, 2020.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[15] Kai Fan Zhi Chen Ricardo Henao Dinghan Shen Lawrence Carin Yizhe Zhang, Zhe Gan. Adversarial feature matching for text generation, 2015.

[16] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018. URL `http://arxiv.org/abs/1804.09541`.

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

[18] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2019.

7 Appendix

7.1 Github

We provide a GitHub repository containing our code here: `https://github.com/JustinLokHinWu/TextGAN-PyTorch`. The ReadMe provides instructions on how to run the training, as well as a list of the files we have created. This repo was forked from https://github.com/williamSYSU/TextGAN-PyTorch [8] - credit goes to the original contributers of this repository for the RelGAN implementation, as well as the codebase for OurGAN. Implementation was done using Visual Studio Live Share.

7.2 Contributions

Adam Paslawski: Researched history of GANs for text generation, implemented model, debugged code, performed experiments, contributed to design decisions and discussions, contributed to all sections of the report.
Justin Wu: Researched related works and promising architectures, implemented model, conducted experiments, contributed to model and experiment sections of the report.
Ariyo Vahdat: Researched RelGAN and other subsequent works. Contributed to implementation and troubleshooting model. Ran experiments. Contributed to abstract, methods, and discussion section in report.