Improved Transformer-Based Implicit Latent GAN with Multi-headed Self-attention for Unconditional Text Generation

Fuji Ren, Ziyun Jiao, Xin Kang

Improved Transformer-based Implicit Latent GAN with Multi-headed Self-attention for Unconditional Text Generation

Fuji Ren[1] , Ziyun Jiao[2], Xin Kang[3]

[1] University of Electronic Science and Technology of China, China
[2,3] Tokushima University, Tokushima, Japan
[1] renfuji@uestc.edu.cn
[2]c501947010@tokushima-u.ac.jp
[3]kang-xin@is.tokushima-u.ac.jp

Abstract. Generative Adversarial Network (GAN) is widely used in computer vision, such as image generation and other tasks. In recent years, GAN has also been developed in the field of unconditional text generation. In this work, we improve TILGAN for unconditional text generation by refactoring the generator. In short, we use Multi-headed Self-attention to replace the Linear layer and BN layer to endow the generator with better text generation capabilities. Our model consists of three components: a transformer autoencoder, a Multi-headed Self attention based generator and a linear based discriminator. The encoder in transformer autoencoder is used to generate the distribution of real samples, and the decoder is used to decode real or generated sentence vector into text. The loss functions for autoencoder and GAN are cross entropy and KL divergence, respectively. On the MS COCO dataset, the proposed model has achieved a better BLEU score than TILGAN. Our ablation experiments also proved the effectiveness of the proposed generator network for unconditional text generation.

Keywords: TILGAN, Self-attention, GAN, Unconditional Text Generation.

1 Introduction

1.1 Generative Adversarial Network(GAN) for unconditional text generation

A generative adversarial network (GAN) [1] can be leaned in an unsupervised way by letting two neural networks play against each other. GAN includes a Generator and a Discriminator, where the goal of the generator is to generate fake samples that can fool the discriminator, and the goal of the discriminator to distinguish between the real and fake samples. In the end, the Generator and the Discriminator reach a Nash equilibrium in the process of playing against each other. In this way, learning GAN models can essentially be thought of as a minimax game, with the objective function given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim Pdata(x)}[\log D(x)] + \mathbb{E}_{z\sim Pz(z)}[\log(1-D(G(x)))] \quad (1)$$

where x represents the real sample and, z represents the random noise. The goal of the Generator is:

$$\arg maxP(D(G(z))) \tag{2}$$

and the goal of the Discriminator is:

$$\arg maxP(D(x)) - P(D(G(z))) \tag{3}$$

In the field of computer vision, GANs rapidly become the hotspot in recent years due to its superior performance. There are some problems when extending the idea of GAN to text generation. In Eq. (2), $G(z)$ generates samples through the '$\arg max$' (this process also calls sampling). Because this operation in text generation is non-derivable process, gradients cannot transfer properly between the generator and the discriminator, which prohibits the normal gradient based training.

For the above problems, text GANs have proposed some effective solutions, such as reinforcement learning (RL) for sequence generation, Gumbel–Softmax relaxation [2], and Wasserstein Distance [3].

At present, GANs for text generation have been able to generate fluent text. GANs are often used in unconditional text generation. In some tasks that need to control the generation direction, such as machine translation, dialogue generation, text summarization, etc., gaps remain between GANs and Seq2seq architecture. Therefore, this work only involves unconditional text generation. And most of the evaluation datasets used for unconstrained text generation include the COCO Captions, EMNLP2017 WMT, Chinese Poems, etc.

1.2 Research objective and content

In this work, we propose a new generator architecture based on Multi-headed Self-attention and linear layer. The overall structure of GAN is improved from TILGAN [4], We rebuilt the generator architecture with Multi-headed Self-attention to make the generator obtain better text generation capabilities. Our model consists of a transformer autoencoder, a generator with Multi-headed Self-attention and a linear-based discriminator. We use the Wasserstein distance or Kullback-Leibler (KL) divergence as the GAN's loss functions. The encoder in transformer autoencoder is used to generate the distribution of real samples, and the decoder is used to decode real sample encoding or generated sample encoding into text. The loss function of autoencoder is cross entropy. The detailed model structure and parameters can be found in Chapter 3. We experiment on the MS COCO dataset. On the MS COCO dataset, the proposed model has achieved a better BLEU score than TILGAN. Through the ablation experiments, we prove that the proposed generator has better ability for unconditional text generation. And the details can be found in Chapter 4. Chapter 5 presents a discussion of the results and the conclusions at last.

2 Related Works

For the above problems to text GANs, researchers have proposed many excellent models in recent years, which can be divided into the following categories:

(1) Using REINFORCE algorithm. This method focuses on dealing with non-differentiable problems caused by discrete data by considering RL methods or reformulating problems in continuous space [5].
A typical representative model using this method is SeqGAN [6]. For the problem that the generator is difficult to transfer gradients, authors regard the entire GAN as a reinforcement learning system and use the Policy Gradient algorithm to update the parameters of the Generator. For the problem that it is difficult for discriminator to evaluate non-complete sequences, the authors draw on the idea of Monte Carlo tree search (MCTS), so that the discriminator can evaluate incomplete sequences at any time.
The LeakGAN [7], which is improved on SeqGAN, also uses the REINFORCE algorithm for training. Different from SeqGAN, the author additionally "leaks" some high-level information of the discriminator to the generator to help the generator to complete the generation task. Specifically, in addition to the reward given by the discriminator, the generator can additionally obtain the high-level feature representation of the discriminator at each moment. In this way, the generation of long texts will be more accurate and varied.

(2) Using Gumbel–Softmax relaxation. Gumbel-Softmax relaxation was first proposed for reparameterization of categories. The improvement goal applied to GAN can be considered to design a more "powerful" softmax, which can replace the sampling operation in the original GAN.
The typical representative network is RelGAN [5]. For the problem that the generator is difficult to transfer gradients, RelGAN utilizes Gumbel-Softmax relaxation to simplify the model, thus replacing reinforcement learning heuristics. At the same time, RelGAN uses relational memory on the generator, which makes it have stronger expression ability and better generation ability on long text. And RelGAN uses multi-layer word vector representation on the discriminator to make the generated text more diverse. Experiments show that RelGAN achieves very good results in the quality and diversity of the generated text.

(3) Using Wasserstein Distance or KL divergence.
The typical representative network is Wasserstein GAN(WGAN). For the problem that the generator is difficult to transfer gradients, Wasserstein Distance can directly calculate the distance between the real and the generated sample distribution, so there is no non-derivable problem. WGAN completely solves the problem of unstable GAN training, and no longer needs to carefully balance the training degree of the generator and the discriminator. It is worth noting that the proposal of WGAN is not aimed at solving the problems faced by GAN in text generation.
Benefit from the idea of WGAN and the extensive use of Transformer autoencoder, the idea of GAN in text generation can be slightly changed, that is, the output of the generator is not necessarily a sentence, but also a sentence vector

in the latent space. Correspondingly, the task of the discriminator has also changed, that is, from judging whether the current sentence is true, to judging whether the current sentence vector is true. Therefore, TILGAN is proposed. Before TILGAN training, the author trains a Transformer-based auto-encoder on the real corpus. After the training, the sentence vectors in the real corpus through the encoder of the auto-encoder will be used as real data, while the generated sentence vectors will be used as fake data.

In addition to the above three categories, there are some other excellent models, such as RankGAN [8], MaskGAN [9], CatGAN [10], etc., which will not be repeated here.

3 Model Architecture

3.1 Overall Framework

The overall framework of our model is shown in Figure 1. The model receives a random noise ε under a Gaussian distribution and takes in real text samples from a corpus X. Through the generator network G_θ, random noise ε is transformed into the generated sentence vector Z. Through the Transformer Encoder, the real text sample is transformed into Z. $\widetilde{X}$ and $\widehat{X}$ represent the sentences obtained by Z and Z through the Transformer Decoder, respectively.

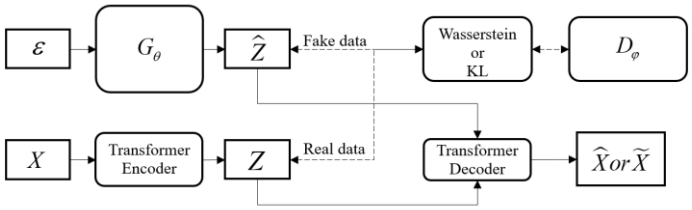

Fig. 1. The overall framework

The proposed GAN framework can be divided into three parts: the Transformer auto-encoder, the Generator and the Discriminator.

The Transformer Encoder is used to generate the distribution of real samples, and the Decoder is used to decode sentence vector into text. The loss function of autoencoder is cross entropy. The task of Transformer is to minimize the gap between $\widetilde{X}$ and X to ensure the accuracy of the real sentence vector distributions.

The Discriminator consists of three linear layers and two BN layers, with the ReLU activation function for each layer. The loss function of GAN is Wasserstein distance or KL divergence. The goal of the generator is to minimize the distance between the generated sentence vector and the sentence vector of the real sample. On the other hand, the discriminator tries to maximize the distance between the real data and the fake data.

3.2 Multi-headed Self attention Based Generator

Different from the stacking of linear layers and BN layers of the TILGAN generator, we use multi-head self-attention to build the generator. The proposed generator framework is shown in Figure 2, where $\otimes$ means the dot product, and the two linear layers are used for reshaping.

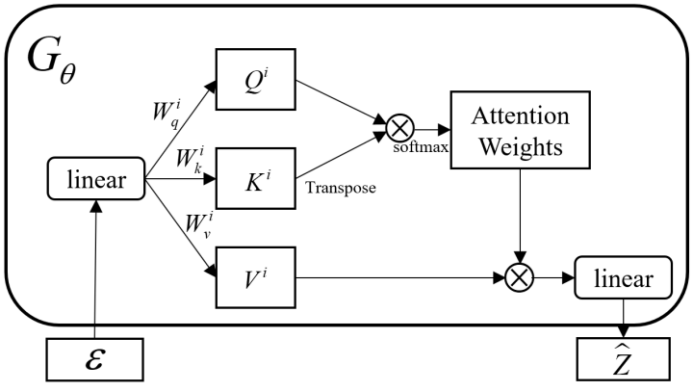

Fig. 2. The proposed generator framework

Formally, we employ $L(\varepsilon)$ to represent the processed noise ε through the linear layer. By setting the number of attention heads to I, we can get I sets of queries, keys and values. For each attention head, we have:

$$\begin{cases} Q^i = L(\varepsilon)W_q^i \\ K^i = L(\varepsilon)W_k^i \\ V^i = L(\varepsilon)W_v^i \end{cases} \tag{4}$$

Accordingly, we can get the sentence vector Z by:

$$Z = L'(\sigma(\frac{Q^i(K^i)^T}{\sqrt{d_k}})V^i) \tag{5}$$

where σ is the softmax function and, d_k is the column dimension of the keys.

3.3 Training Details

Limited by hardware equipment, we set the batch size to 64. We used the Adam [11] optimizer, the learning rate of the GAN is 1×10^{-4}. And the learning rate of auto-encoder is 0.12. The current loss function is KL divergence, in the future work, we may change the loss function.

4 Experiments

4.1 Evaluation Metrics

In this work, we use two metrics to evaluate the models. The first metric is bilingual evaluation understudy(BLEU-test) [12]. This score indicates how similar the candidate text is to the reference text. The BLEU-test value is in the range of [0, 1], and a larger BLEU-test value indicates a better generation result. In general, the BLEU score could provide an overall assessment of model quality.

The second metric is Self-BLEU [13]. Self-BLEU is a diversity metric, by calculating the similarity between one generated sentence and the whole remaining generation. A lower the Self-BLEU score is, indicates a higher diversity we can obtain in the generated texts [4].

4.2 Microsoft COCO: Common Objects in Context

In order to test our model, we firstly conduct experiments on MSCOCO [14]. All the preprocessing steps are same as Chen et al. (2018) [15]. The details of the dataset are shown in Table 1.

Table 1. The details of MS COCO

Dataset	MS COCO
Vocab_Size	27842
Average_len	10.4
Train sentence Num	120K
Test sentence Num	10K

Similar with TILGAN, we set the Transformer autoencoder with 2 layers, 4 heads, and 512 hidden dimensions. In addition, we set the generator with 4 heads, 256 head size, 32 hidden dimensions. All the sentences will be padded to the maximum length during training. Then the BLEU scores on MSCOCO dataset are shown in Table 2. The proposed model has achieved significantly better performance compared to the existing models in BLEU-2, 3 and 4 and Self-BLEU-2 and 3, The results suggest that our text generation model is generally more effective on the MSCOCO dataset than the existing models.

Table 2. The BLEU scores on MSCOCO. For BLEU-test, the higher the better. For Self-BLEU, the lower the better.

Method	BLEU-test				Self-BLEU		
	B2%	B3%	B4%	B5%	B2%	B3%	B4%
SEQGAN [6]	82.0	60.4	36.1	21.1	80.7	57.7	27.8
RANKGAN [8]	85.2	63.7	38.9	24.8	82.2	59.2	28.8
LEAKGAN [7]	92.2	79.7	60.2	41.6	91.2	82.5	68.9

GSGAN [16]	81.0	56.6	33.5	19.7	78.5	52.2	23.0
WGAN [3]	73.0	53.8	34.2	12.5	90.4	80.9	69.0
TILGAN [4]	96.7	90.3	77.2	**53.2**	61.6	35.6	**9.9**
OUR MODEL	**98.6**	**92.8**	**79.9**	42.0	**54.8**	**27.0**	12.1

4.3 Ablation Experiment

To indicate that our changes to the generator are effective, we also conduct ablation experiment on MSCOCO . We keep all model parameters the same with TILGAN except the generator (including learning rate, model structure, number of autoencoder layers, number of hidden layers, etc.). The only difference is the generator. The BLEU-3 curve is shown in Figure 3. The overfitting part is not shown in the figure.

Through the curves, we find that our generator converges much faster than TILGAN. The results show that our generator has better text generation ability. Compared with the original generator, our model can achieve better results on large datasets as well as for long text generation.

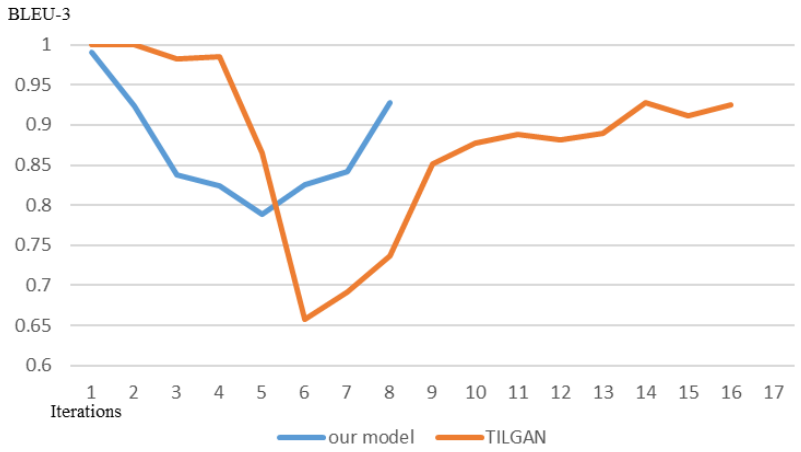

Fig. 3. Ablation Experiment

5 Conclusion and Future work

This paper proposes an improved model for text generation, we use Multi-headed Self-attention to replace the Linear layer and BN layer to make the generator obtain better text generation capabilities. Compared with the existing models, our model has higher evaluation scores and diverse sample on MSCOCO dataset. In the future work, we will continue to conduct experiments on other datasets, while looking for the best model parameters to obtain better performance.

Acknowledgments

This research has been supported by JSPS KAKENHI Grant Number 19K20345.

References

1. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks [J] Commun. ACM 2020, 63, 139–144.
2. Kusner, M.J.; Hernández-Lobato, J.M. GANs for sequences of discrete elements with the gumbel-softmax distribution. arXiv 2016, arXiv:1611.04051.
3. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein gan. arXiv 2017, arXiv:1701.07875.
4. Diao S, Shen X, Shum K, et al. TILGAN: Transformer-based Implicit Latent GAN for Diverse and Coherent Text Generation [C] Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. 2021: 4844-4858.
5. Nie, W.; Narodytska, N.; Patel, A. Relgan: Relational generative adversarial networks for text generation [C] In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
6. Yu, L.; Zhang, W.; Wang, J.; Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient [C] In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
7. Guo, J.; Lu, S.; Cai, H.; Zhang, W.; Yu, Y.; Wang, J. Long text generation via adversarial training with leaked information. arXiv 2017, arXiv:1709.08624.
8. Juefei-Xu, F.; Dey, R.; Boddeti, V.N.; Savvides, M. Rankgan: A maximum margin ranking gan for generating faces [C] In Proceedings of the Asian Conference on Computer Vision, Perth, Australia, 2–6 December 2018; Springer: Cham, Switzerland, 2018; pp. 3–18.
9. Fedus, W.; Goodfellow, I.; Dai, A.M. MaskGAN: Better text generation via filling in the ____. arXiv 2018, arXiv:1801.07736.
10. Liu Z, Wang J, Liang Z. Catgan: Category-aware generative adversarial networks with hierarchical evolutionary learning for category text generation [C] Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 8425-8432.
11. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. arXiv 2014, arXiv:1412.6980.
12. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. Bleu: A method for automatic evaluation of machine translation [C] In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002
13. Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A Benchmarking Platform for Text Generation Models [C] In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pages 1097–1100.
14. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context [C] In European conference on computer vision, pages 740–755. Springer
15. Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. 2018. Adversarial Text Generation via Feature Mover's Distance [C] In Advances in Neural Information Processing Systems, pages 4666–4677.
16. Wu H Y, Chen Y L. Graph sparsification with generative adversarial network [C] 2020 IEEE International Conference on Data Mining (ICDM). IEEE, 2020: 1328-1333.