TextKD-GAN: Text Generation using Knowledge Distillation and Generative Adversarial Networks

Md. Akmal Haidar and Mehdi Rezagholizadeh
{md.akmal.haidar, mehdi.rezagholizadeh}@huawei.com

Huawei Noah's Ark Lab, Montreal Research Center, Montreal, Canada

Abstract. Text generation is of particular interest in many NLP applications such as machine translation, language modeling, and text summarization. Generative adversarial networks (GANs) achieved a remarkable success in high quality image generation in computer vision, and recently, GANs have gained lots of interest from the NLP community as well. However, achieving similar success in NLP would be more challenging due to the discrete nature of text. In this work, we introduce a method using knowledge distillation to effectively exploit GAN setup for text generation. We demonstrate how autoencoders (AEs) can be used for providing a continuous representation of sentences, which is a smooth representation that assign non-zero probabilities to more than one word. We distill this representation to train the generator to synthesize similar smooth representations. We perform a number of experiments to validate our idea using different datasets and show that our proposed approach yields better performance in terms of the BLEU score and Jensen-Shannon distance (JSD) measure compared to traditional GAN-based text generation approaches without pre-training.

Keywords: Text generation · Generative adversarial networks · Knowledge distillation.

1 Introduction

Recurrent neural network (RNN) based techniques such as language models are the most popular approaches for text generation. These RNN-based text generators rely on maximum likelihood estimation (MLE) solutions such as teacher forcing [11] (i.e. the model is trained to predict the next item given all previous observations); however, it is well-known in the literature that MLE is a simplistic objective for this complex NLP task [15]. MLE-based methods suffer from exposure bias [20], which means that at training time the model is exposed to gold data only, but at test time it observes its own predictions.

However, GANs which are based on the adversarial loss function and have the generator and the discriminator networks suffers less from the mentioned problems. GANs could provide a better image generation framework comparing to the traditional MLE-based methods and achieved substantial success in the field of computer vision for generating realistic and sharp images. This great success motivated researchers to apply its framework to NLP applications as well.

GANs have been exploited recently in various NLP applications such as machine translation [24,25], dialogue models [15], question answering [26], and natural language generation [7,20,19,13,28,29]. However, applying GAN in NLP is challenging due to the discrete nature of the text. Consequently, back-propagation would not be feasible for discrete outputs and it is not straightforward to pass the gradients through the discrete output words of the generator. The existing GAN-based solutions can be categorized according to the technique that they leveraged for handling the problem of the discrete nature of text: Reinforcement learning (RL) based methods, latent space based solutions, and approaches based on continuous approximation of discrete sampling. Several versions of the RL-based techniques have been introduced in the literature including Seq-GAN [27], MaskGAN [5], and LeakGAN [8]. However, they often need pre-training and are computationally more expensive compared to the methods of the other two categories. Latent space-based solutions derive a latent space representation of the text using an AE and attempt to learn data manifold of that space [13]. Another approach for generating text with GANs is to find a continuous approximation of the discrete sampling by using the Gumbel Softmax technique [14] or approximating the non-differentiable argmax operator [28] with a continuous function.

In this work, we introduce TextKD-GAN as a new solution for the main bottleneck of using GAN for text generation with knowledge distillation: a technique that transfer the knowledge of softened output of a teacher model to a student model [9]. Our solution is based on an AE (Teacher) to derive a smooth representation of the real text. This smooth representation is fed to the TextKD-GAN discriminator instead of the conventional one-hot representation. The generator (Student) tries to learn the manifold of the softened smooth representation of the AE. We show that TextKD-GAN outperforms the conventional GAN-based text generators that do not need pre-training. The remainder of the paper is organized as follows. In the next two sections, some preliminary background on generative adversarial networks and related work in the literature will be reviewed. The proposed method will be presented in section 4. In section 5, the experimental details will be discussed. Finally, section 6 will conclude the paper.

2   Background

Generative adversarial networks include two separate deep networks: a generator and a discriminator. The generator takes in a random variable, z following a distribution $P_z(z)$ and attempt to map it to the data distribution $P_x(x)$. The output distribution of the generator is expected to converge to the data distribution during the training. On the other hand, the discriminator is expected to discern real samples from generated ones by outputting zeros and ones, respectively. During training, the generator and discriminator generate samples and classify them, respectively by adversarially affecting the performance of each other. In this regard, an adversarial loss function is employed for training [6]:

$$\min_G \max_D V(D,G) = E_{x \sim P_x(x)}[\log D(x)] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

This is a two-player minimax game for which a Nash-equilibrium point should be derived. Finding the solution of this game is non-trivial and there has been a great extent of literature dedicated in this regard [22].
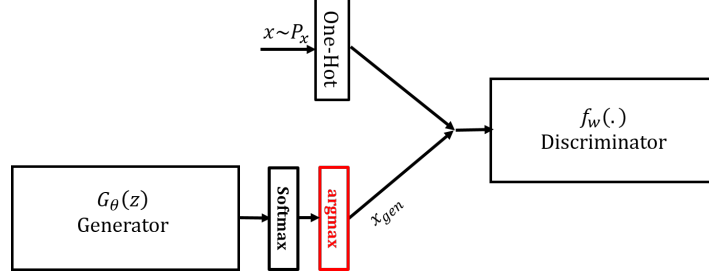

Fig. 1: Simplistic text generator with GAN

As stated, using GANs for text generation is challenging because of the discrete nature of text. To clarify the issue, Figure 1 depicts a simplistic architecture for GAN-based text generation. The main bottleneck of the design is the *argmax* operator which is not differentiable and blocks the gradient flow from the discriminator to the generator.

$$\min_G E_{z \sim P_z(z)}[\log(1 - D(\text{argmax}(\text{softmax}(G(z)))))] \tag{2}$$

2.1 Knowledge Distillation

Knowledge distillation has been studied in model compression where knowledge of a large cumbersome model is transferred to a small model for easy deployment. Several studies have been studied on the knowledge transfer technique [9,21]. It starts by training a big teacher model (or ensemble model) and then train a small student model which tries to mimic the characteristics of the teacher model, such as hidden representations [21], it's output probabilities [9], or directly on the generated sentences by the teacher model in neural machine translation [12]. The first teacher-student framework for knowledge distillation was proposed in [9] by introducing the softened teacher's output. In this paper, we propose a GAN framework for text generation where the generator (Student) tries to mimic the reconstructed output representation of an auto-encoder (Teacher) instead of mapping to a conventional one-hot representations.

2.2 Improved WGAN

Generating text with pure GANs is inspired by improved Wasserstein GAN (IWGAN) work [7]. In IWGAN, a character level language model is developed

based on adversarial training of a generator and a discriminator without using any extra element such as policy gradient reinforcement learning [23]. The generator produces a softmax vector over the entire vocabulary. The discriminator is responsible for distinguishing between the one-hot representations of the real text and the softmax vector of the generated text. The IWGAN method is described in Figure 2. A disadvantage of this technique is that the discriminator is able to tell apart the one-hot input from the softmax input very easily. Hence, the generator will have a hard time fooling the discriminator and vanishing gradient problem is highly probable.
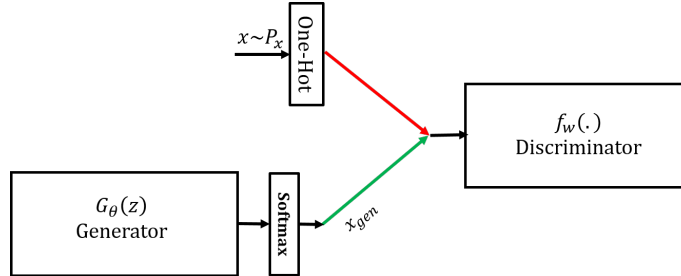

Fig. 2: Improved WGAN for text generation

3   Related Work

A new version of Wasserstein GAN for text generation using gradient penalty for discriminator was proposed in [7]. Their generator is a CNN network generating fixed-length texts. The discriminator is another CNN receiving 3D tensors as input sentences. It determines whether the tensor is coming from the generator or sampled from the real data. The real sentences and the generated ones are represented using one-hot and softmax representations, respectively.

A similar approach was proposed in [20] with an RNN-based generator. They used a curriculum learning strategy [2] to produce sequences of gradually increasing lengths as training progresses. In [19], RNN is trained to generate text with GAN using curriculum learning. The authors proposed a procedure called teacher helping, which helps the generator to produce long sequences by conditioning on shorter ground-truth sequences.

All these approaches use a discriminator to discriminate the generated softmax output from one-hot real data as in Figure 2, which is a clear downside for them. The reason is the discriminator receives inputs of different representations: a one-hot vector for real data and a probabilistic vector output from the generator. It makes the discrimination rather trivial.

AEs have been exploited along with GANs in different architectures for computer vision application such as AAE [17], ALI [4], and HALI [1]. Similarly,

AEs can be used with GANs for generating text. For instance, an adversarially regularized AE (ARAE) was proposed in [13]. The generator is trained in parallel to an AE to learn a continuous version of the code space produced by AE encoder. Then, a discriminator will be responsible for distinguishing between the encoded hidden code and the continuous code of the generator. Basically, in this approach, a continuous distribution is generated corresponding to an encoded code of text.

4   Methodology

AEs can be useful in denoising text and transferring it to a code space (encoding) and then reconstructing back to the original text from the code. AEs can be combined with GANs in order to improve the generated text. In this section, we introduce a technique using AEs to replace the conventional one-hot representation [7] with a continuous softmax representation of real data for discrimination.

4.1   Distilling output probabilities of AE to TextKD-GAN generator

As stated, in conventional text-based discrimination approach [7], the real and generated input of the discriminator will have different types (one-hot and softmax) and it can simply tell them apart. One way to avoid this issue is to derive a continuous smooth representation of words rather than their one-hot and train the discriminator to differentiate between the continuous representations. In this work, we use a conventional AE (Teacher) to replace the one-hot representation with softmax reconstructed output, which is a smooth representation that yields smaller variance in gradients [9]. The proposed model is depicted in Figure 3. As seen, instead of the one-hot representation of the real words, we feed the softened reconstructed output of the AE to the discriminator. This technique would makes the discrimination much harder for the discriminator. The GAN generator (Student) with softmax output tries to mimic the AE output distribution instead of conventional one-hot representations used in the literature.
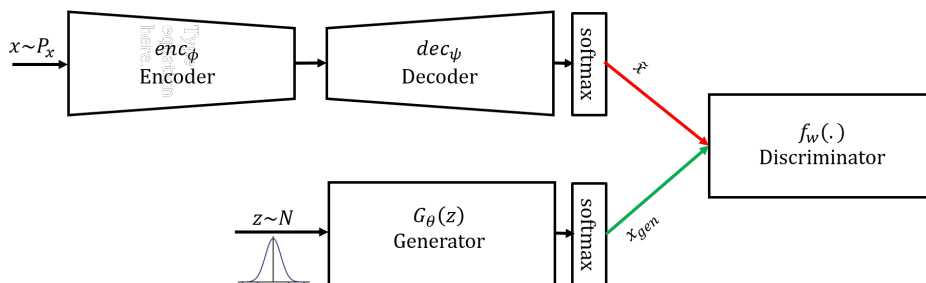

Fig. 3: TextKD-GAN model for text generation

4.2   Why TextKD-GAN should Work Better than IWGAN

Suppose we apply IWGAN to a language vocabulary of size two: words x_1 and x_2. The one-hot representation of these two words (as two points in the Cartesian coordinates) and the span of the generated softmax outputs (as a line segment connecting them) is depicted in the left panel of Figure 4. As evident graphically, the task of the discriminator is to discriminate the points from the line connecting them, which is a rather simple very easy task.

Now, let's consider the TextKD-GAN idea using the two-word language example. As depicted in Figure 4 (Right panel), the output locus of the TextKD-GAN decoder would be two red line segments instead of two points (in the one-hot case). The two line segments lie on the output locus of the generator, which will make the generator more successful in fooling the discriminator.
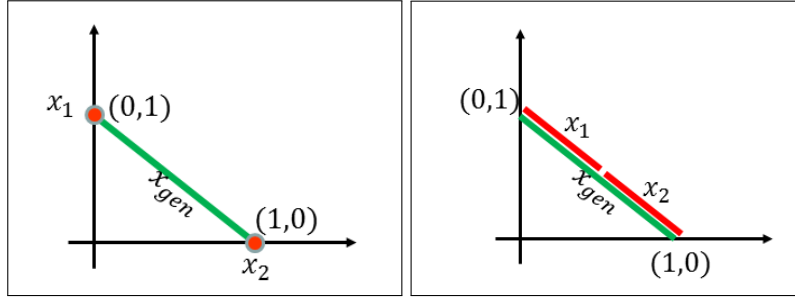

Fig. 4: Locus of the input vectors to the discriminator for a two-word language model; Left panel: IWGAN, Right panel: TextKD-GAN.

4.3   Model Training

We train the AE and TextKD-GAN simultaneously. In order to do so, we break down the objective function into three terms: (1) a reconstruction term for the AE, (2) a discriminator loss function with gradient penalty, (3) an adversarial cost for the generator. Mathematically,

1) $\min_{(\phi,\psi)} L_{AE}(\phi,\psi) = \min_{(\phi,\psi)} ||x - \text{softmax}(\text{dec}_\psi(\text{enc}_\phi(x)))||^2$

2) $\min_{w\in W} L_{\text{discriminator}}(w) =$

$\min_{w\in W} -E_{x\sim P_x}[f_w(\text{dec}_\psi(\text{enc}_\phi(X)))] + E_{z\sim P_z}[f_w(G(z))] + \lambda_2 E_{\hat{x}\sim P_{\hat{x}}}[(||\nabla_{\hat{x}} f_w(\hat{x})||_2 - 1)^2]$

3) $\min_\theta L_{\text{Gen}}(\theta) = -\min_\theta E_{z\sim P_z}[f_w(G(z))].$

$$(3)$$

These losses are trained alternately to optimize different parts of the model. We employ the gradient penalty approach of IWGAN [7] for training the discriminator. In the gradient penalty term, we need to calculate the gradient norm of random samples $\hat{x} \sim P_{\hat{x}}$. According to the proposal in [7], these random samples can be obtained by sampling uniformly along the line connecting pairs of generated and real data samples:

$$[\hat{x} \sim P_{\hat{x}}] \leftarrow \alpha \; [x \sim P_x] + (1 - \alpha) \; [x_{gen} \sim G(z)] \tag{4}$$

The complete training algorithm is described in 1.

Algorithm 1 TextKD-GAN for text generation.

Require: The Adam hyperparameters α, β_1, β_2, the batch size m. Initial AE parameters (encoder (ϕ_0), decoder ψ_0), discriminator parameters w_0 and initial generator parameters θ_0

1: **for** number of training iterations **do**
      AE Training:
2:     Sample $\{x^{(i)}\}_{i=1}^m \sim P_x$ and compute code-vectors $c^i = enc_\phi(x^i)$
3:     and reconstructed text $\{\tilde{x}^i\}_{i=1}^m$.
4:     Backpropagate reconstruction loss $L_{AE}(\phi, \psi)$.
5:     Update with $(\phi, \psi) \leftarrow Adam(L_{AE}(\phi, \psi), \alpha, \beta_1, \beta_2)$.
      Train the discriminator:
6:     **for** k times **do**:
7:       Sample $\{x^{(i)}\}_{i=1}^m \sim P_x$ and Sample $\{z^{(i)}\}_{i=1}^m \sim N(0, I)$.
8:       Compute generated text $\{x_{gen}^{(i)}\}_{i=1}^m \sim G(z)$
9:       Backpropagate discriminator loss $L_{discriminator}(w)$.
10:      Update with $w \leftarrow Adam(L_{discriminator}(w), \alpha, \beta_1, \beta_2)$.
     end for
     Train the generator:
11:    Sample $\{x^{(i)}\}_{i=1}^m \sim P_x$ and Sample $\{z^{(i)}\}_{i=1}^m \sim N(0, I)$.
12:    Compute generated text $\{x_{gen}^{(i)}\}_{i=1}^m \sim G(z)$
13:    Backpropagate generator loss $L_{Gen}(\theta)$.
14:    Update with $\theta \leftarrow Adam(L_{Gen}(\theta), \alpha, \beta_1, \beta_2)$.
    end for

---

5 Experiments

5.1 Dataset and Experimental Setup

We carried out our experiments on two different datasets: Google 1 billion benchmark language modeling data[1] and the Stanford Natural Language Inference (SNLI) corpus[2]. Our text generation is performed at character level with a sentence

---

[1]http://www.statmt.org/lm-benchmark/
[2]https://nlp.stanford.edu/projects/snli/

length of 32. For the Google dataset, we used the first 1 million sentences and extract the most frequent 100 characters to build our vocabulary. For the SNLI dataset, we used the entire preprocessed training data [3], which contains 714667 sentences in total and the built vocabulary has 86 characters. We train the AE using one layer with 512 LSTM cells [10] for both the encoder and the decoder. We train the autoencoder using Adam optimizer with learning rate 0.001, $\beta_1=$ 0.9, and $\beta_2=$ 0.9. For decoding, the output from the previous time step is used as the input to the next time step. The hidden code c is also used as an additional input at each time step of decoding. The greedy search approach is applied to get the best output [13]. We keep the same CNN-based generator and discriminator with residual blocks as in [7]. The discriminator is trained for 5 times for 1 GAN generator iteration. We train the generator and the discriminator using Adam optimizer with learning rate 0.0001, $\beta_1=$ 0.5, and $\beta_2=$ 0.9.

We use the *BLEU-N* score to evaluate our techniques. *BLEU-N* score is calculated according to the following equation [16,3,18]:

$$BLEU\text{-}N = BP \cdot \exp(\sum_{n=1}^{N} w_n \log(p_n)) \tag{5}$$

where p_n is the probability of n-gram and $w_n = \frac{1}{n}$. We calculate BLEU-n scores for n-grams without a brevity penalty [29]. We train all the models for 200000 iterations and the results with the best *BLEU-N* scores in the generated texts are reported. To calculate the *BLEU-N* scores, we generate ten batches of sentences as candidate texts, i.e. 640 sentences (32-character sentences) and use the entire test set as reference texts.

5.2   Experimental Results

The results of the experiments are depicted in Table 1 and 2. As seen in these tables, the proposed TextKD-GAN approach yields significant improvements in terms of *BLEU-2, BLEU-3* and *BLEU-4* scores over the IWGAN [7], and the ARAE [13] approaches. Therefore, softened smooth output of the decoder can be more useful to learn better discriminator than the traditional one-hot representation. Moreover, we can see the lower *BLEU*-scores and less improvement for the Google dataset compared to the SNLI dataset. The reason might be the sentences in the Google dataset are more diverse and complicated. Finally, note that the text-based one-hot discrimination in IWGAN and our proposed method are better than the traditional code-based ARAE technique [13].

Some examples of generated text from the SNLI experiment are listed in Table 3. As seen, the generated text by the proposed TextKD-GAN approach is more meaningful and contains more correct words compared to that of IWGAN [7].

We also provide the training curves of Jensen-Shannon distances (JSD) between the n-grams of the generated sentences and that of the training (real) ones

---

[3]https://github.com/aboev/arae-tf/tree/master/data_snli

Table 1: Results of the *BLEU-N* scores using 1 million sentences from 1 billion Google dataset

Model	BLEU-2	BLEU-3	BLEU-4
IWGAN	0.50	0.27	0.11
ARAE	0.13	0.02	0.00
TextKD-GAN	**0.51**	**0.29**	**0.13**

Table 2: Results of the *BLEU-N* scores using SNLI dataset

Model	BLEU-2	BLEU-3	BLEU-4
IWGAN	0.57	0.44	0.30
ARAE	0.37	0.27	0.17
TextKD-GAN	**0.62**	**0.50**	**0.38**

Table 3: Example generated sentences with model trained using SNLI dataset

IWGAN	TextKD-GAN
The people are laying in angold	Two people are standing on the s
A man is walting on the beach	A woman is standing on a bench .
A man is looking af tre walk aud	People have a ride with the comp
A man standing on the beach	A woman is sleeping at the brick
The man is standing is standing	Four people eating food .
A man is looking af tre walk aud	The dog is in the main near the
The man is in a party .	A black man is going to down the
Two members are walking in a hal	These people are looking at the
A boy is playing sitting .	the people are running at some l

in Figure 5. The distances are derived from SNLI experiments and calculated as in [7]. That is by calculating the log-probabilities of the n-grams of the generated and the real sentences. As depicted in the figure, the TextKD-GAN approach further minimizes the JSD compared to the literature methods [7,13]. In conclusion, our approach learns a more powerful discriminator, which in turn generates the data distribution close to the real data distribution.

5.3   Discussion

The results of our experiment shows the superiority of our TextKD-GAN method over other conventional GAN-based techniques. We compared our technique with those GAN-based generators which does not need pre-training. This explains why we have not included the RL-based techniques in the results. We showed the power of the continuous smooth representations over the well-known tricks to work around the discontinuity of text for GANs. Using AEs in TextKD-GAN adds another important dimension to our technique which is the latent space, which
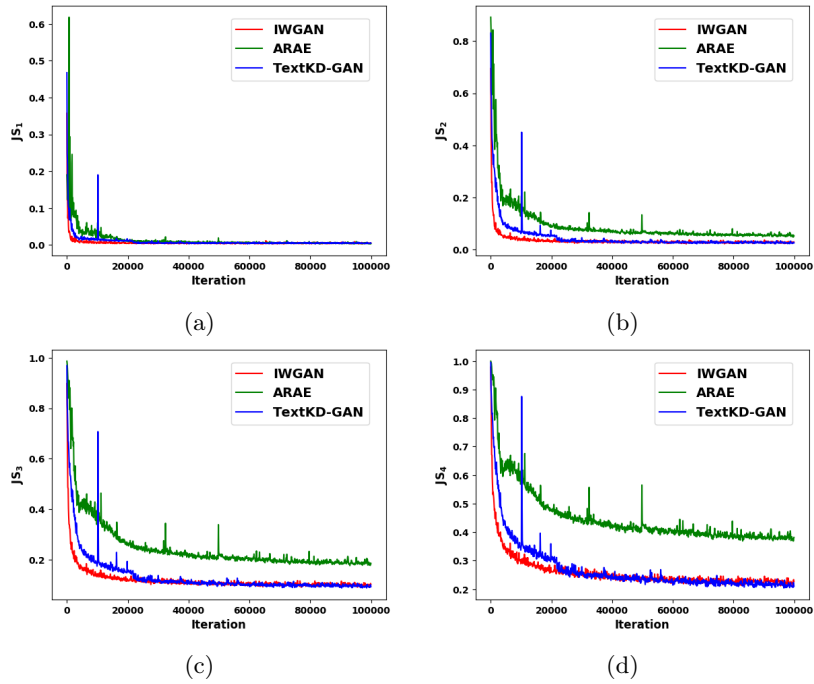
Fig. 5: Jensen-Shannon distance (JSD) between the generated and training sentences n-grams derived from SNLI experiments. a) js_1, b) js_2, c) js_3, and d) js_4 represent the JSD for 1, 2, 3, and 4-grams respectively

can be modeled and exploited as a separate signal for discriminating the generated text from the real data. It is worth mentioning that our observations during the experiments show training text-based generators is much easier than training the code-based techniques such as ARAE. Moreover, we observed that the gradient penalty term plays a significant part in terms of reducing the mode-collapse from the generated text of GAN. Furthermore, in this work, we focused on character-based techniques; however, TextKD-GAN is applicable to the word-based settings as well. Bear in mind that pure GAN-based text generation techniques are still in a newborn stage and they are not very powerful in terms of learning semantics of complex datasets and large sentences. This might be because of lack of capacity of capturing the long-term information using CNN networks. To address this problem, RL can be employed to empower these pure GAN-based techniques such as TextKD-GAN as a next step .

6   Conclusion and Future Work

In this work, we introduced TextKD-GAN as a new solution using knowledge distillation for the main bottleneck of using GAN for generating text, which is

the discontinuity of text. Our solution is based on an AE (Teacher) to derive a continuous smooth representation of the real text. This smooth representation is distilled to the GAN discriminator instead of the conventional one-hot representation. We demonstrated the rationale behind this approach, which is to make the discrimination task of the discriminator between the real and generated texts more difficult and consequently providing a richer signal to the generator. At the time of training, the TextKD-GAN generator (Student) would try to learn the manifold of the smooth representation, which can later on be mapped to the real data distribution by applying the argmax operator. We evaluated TextKD-GAN over two benchmark datasets using the *BLEU-N* scores, JSD measures, and quality of the output generated text. The results showed that the proposed TextKD-GAN approach outperforms the traditional GAN-based text generation methods which does not need pre-training such as IWGAN and ARAE. Finally, We summarize our plan for future work in the following:

1. We evaluated TextKD-GAN in a character-based level. However, the performance of our approach in word-based level needs to be investigated.
2. Current TextKD-GAN is implemented with a CNN-based generator. We might be able to improve TextKD-GAN by using RNN-based generators.
3. TextKD-GAN is a core technique for text generation and similar to other pure GAN-based techniques, it is not very powerful in generating long sentences. RL can be used as a tool to accommodate this weakness.

References

1. Belghazi, M.I., Rajeswar, S., Mastropietro, O., Rostamzadeh, N., Mitrovic, J., Courville, A.: Hierarchical adversarially learned inference. arXiv preprint arXiv:1802.01071 (2018)
2. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning. pp. 41–48. ACM (2009)
3. Cer, D., Manning, C.D., Jurafsky, D.: The best lexical metric for phrase-based statistical mt system optimization. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 555–563. Association for Computational Linguistics (2010)
4. Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference. arXiv preprint arXiv:1606.00704 (2016)
5. Fedus, W., Goodfellow, I., Dai, A.M.: Maskgan: Better text generation via filling in the _ . arXiv preprint arXiv:1801.07736 (2018)
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
7. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028 (2017)
8. Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J.: Long text generation via adversarial training with leaked information. arXiv preprint arXiv:1709.08624 (2017)

9. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation, 9(8):1735–1780 (1997)
11. J, W.R., David, Z.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation, 1(2):270–280 (1989)
12. Kim, Y., Rush, A.M.: Sequence-level knowledge distillation. In: EMNLP. pp. 1317–1327 (2016)
13. Kim, Y., Zhang, K., Rush, A.M., LeCun, Y., et al.: Adversarially regularized autoencoders for generating discrete structures. arXiv preprint arXiv:1706.04223 (2017)
14. Kusner, M.J., Hernández-Lobato, J.M.: Gans for sequences of discrete elements with the gumbel-softmax distribution. arXiv preprint arXiv:1611.04051 (2016)
15. Li, J., Monroe, W., Shi, T., Ritter, A., Jurafsky, D.: Adversarial learning for neural dialogue generation. arXiv preprint arXiv:1701.06547 (2017)
16. Liu, C.W., Lowe, R., Serban, I.V., Noseworthy, M., Charlin, L., Pineau, J.: How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. arXiv preprint arXiv:1603.08023 (2016)
17. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. arXiv preprint arXiv:1511.05644 (2015)
18. Papineni, K., Roukos, S., Ward, T., Zhu, W.: Bleu: a method for automatic evaluation of machine translation. In: ACL. pp. 311–318 (2002)
19. Press, O., Bar, A., Bogin, B., Berant, J., Wolf, L.: Language generation with recurrent generative adversarial networks without pre-training. arXiv preprint arXiv:1706.01399 (2017)
20. Rajeswar, S., Subramanian, S., Dutil, F., Pal, C., Courville, A.: Adversarial generation of natural language. arXiv preprint arXiv:1705.10929 (2017)
21. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. In: ICLR (2015)
22. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems. pp. 2234–2242 (2016)
23. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: NIPS. pp. 1057–1063 (1999)
24. Wu, L., Xia, Y., Zhao, L., Tian, F., Qin, T., Lai, J., Liu, T.Y.: Adversarial neural machine translation. arXiv preprint arXiv:1704.06933 (2017)
25. Yang, Z., Chen, W., Wang, F., Xu, B.: Improving neural machine translation with conditional sequence generative adversarial nets. arXiv preprint arXiv:1703.04887 (2017)
26. Yang, Z., Hu, J., Salakhutdinov, R., Cohen, W.W.: Semi-supervised qa with generative domain-adaptive nets. arXiv preprint arXiv:1702.02206 (2017)
27. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: AAAI. pp. 2852–2858 (2017)
28. Zhang, Y., Gan, Z., Fan, K., Chen, Z., Henao, R., Shen, D., Carin, L.: Adversarial feature matching for text generation. arXiv preprint arXiv:1706.03850 (2017)
29. Zhu, Y., Lu, S., Zheng, L., Jiaxian, G., Weinan, Z., Jun, W., Yong, Y.: Texygen: A benchmarking platform for text generation models. arXiv preprint arXiv:1802.01886. (2018)