Generating Tweets using Generative Adversarial Networks

Ruowen Wang
SCPD
ruowen19@stanford.edu

Abstract

In this work, we explore the possibility and challenges of applying Generative Adversarial Networks (GAN) to the task of generating tweets with a specific language style to simulate tweets from a real Twitter account. We use President Donald Trump's Tweets as a case study to train and evaluate whether a GAN-based text generation model can learn the language styles and generate realistic tweets. We also compare the GAN model with a basic RNN model as the baseline in terms of training difficulty and text generation quality. The preliminary results show that it is harder to train a GAN model than the baseline RNN model. The problems of training a GAN model, such as mode collapse, are observed. With fine-tuned hyperparameters, the generated texts from a GAN model are slightly better than the baseline RNN model. However, current results indicate that more efforts are needed on choosing better hyperparameters, finding better cost functions and designing better training strategy when applying GAN to text generation.

1 Introduction

Text generation is an active research topic in Deep Learning and Natural Language Processing (NLP) [12]. It can be applied to many tasks including machine translation, article summarization, image captioning, questions and answering, etc. Generally, the goal of text generation is to generate natural language texts that resemble real-world training texts in similar linguistic styles and contexts. Various generative models have been studied in this domain. Recurrent Neural Networks (RNN) is commonly used in text generation [13, 7]. By feeding words one by one in a time series, a trained RNN can recursively predict the next most likely word. Sequence (and more complicated Variational [1]) autoencoders are utilized to capture high-level features of a complete sentence for better text generation. However, as pointed out by Bowman et al. [1], although cherry-picked results from previous work look promising, it is often found that the generated texts are grammatically or semantically incorrect. As researchers continuously propose new models and techniques, the text generation problem deserves more exploration and may benefit from new advances.

In this work, we investigate whether Generative Adversarial Networks (GAN) [6] can be applied or built on top of existing models to further improve the quality of text generation, in terms of mimicking more similar linguistic styles and achieving better grammatical correctness. Previously, GAN has shown great promise in generating realistic images [3]. With two adversaries, a generator and a discriminator, playing against each other, GAN pushes both sides to the possibly optimal stage so that the generator can generate realistic but fake samples that the discriminator cannot distinguish from the real training samples. One previous challenge of applying GAN to discrete sequence such as text or music, is that due to the discrete tokens, it is difficult to propagate the gradient updates from the discriminator to the generator. Researchers proposed to use Reinforcement Learning [18] to bypass the differential problem by using policy gradient updates (e.g., SeqGAN [21], MaskGAN [5]). Yet, other challenges in GAN such as mode collapse, non-convergence and unbalance between generator and discriminator may still cause problems and require more investigation.

As a case study, we explore a specific text generation task, *using GAN to generate Trump-style tweets*. As President Donald Trump is famous for using Twitter to promote political opinions and policy with a distinctive language style, the text corpus of Trump's tweets [19] becomes a valuable dataset for the NLP research, since it has a considerable data amount, 140 (or 280) length limits, and relatively simple and political-focused semantics. Previous CS230 project has explored using basic Recurrent Neural Network (RNN) or LSTM to generate "Presidential"-style tweets [17]. We take a step further to investigate if GAN can be applied as a new learning model to this problem and discuss new challenges we encounter during the model training.

After an one and a half month work and experiment, our preliminary results show that compared to a basic RNN model, it is practically harder to train a GAN model. We have tried and revised a few open source implementations of GAN-based models. Several typical issues in a GAN training are observed, including mode collapse, vanish gradient and unbalance. In other rounds with previously fine-tuned hyperparameters, we observe slightly better text generation of GAN models than basic RNN model, in terms of grammar and semantics. However, current results indicate that given the same training dataset, GAN models do not exceed basic RNN models, specially given higher training difficulty than basic models. This indicates that more efforts on GAN models are needed to further refine the hyperparameters, cost functions and training strategy, to truly utilize the potentials of GAN models. In the following sections, we will explain the details of dataset, baseline RNN and GAN models, the hyperparameters we choose and the text generation result.

2 Dataset

We use a dataset of 10-year tweets from @realDonaldTrump (excluding retweets) from 01/01/2009 to 02/20/2019, collected by [19]. In total, there are 14,686 original tweets published by @realDonaldTrump. We sanitize the dataset by removing http URL links, unifying single/double quotes from Unicode to ASCII. Currently, we keep @ mentions and # hashtags there as these are popular tokens in tweets, differentiated from ordinary text corpus. We hope the model can learn how to use these tokens in the text context to generate more realistic tweets. The dataset is preprocessed based on word level and an embedding is trained to include special tokens of names and hashtags.

Given the size of the above dataset, one concern is that it might not be large enough and may cause overfitting problem. Other datasets can be considered include general tweets from various twitter accounts (to pretrain general tweet styles and format), Trump's speech transcript [4] (to pretrain or fine-tune the embedding and RNN weights of Trump's language style). However, given the time constraint, currently we only use the above 14,686 tweets as the dataset for training and comparing both baseline RNN models and GAN models.

3 Baseline RNN Model

The baseline model[1] is a plain 3-layer LSTM network. As shown in Figure 1, the input is a batch of sentences of maximum length 15 word-level tokens, fed into a 100-dimension embedding layer, then followed with three 128-unit LSTM layers, and finally fully connected with softmax to predict the next probable token. RMSprop is used to minimize the categorical cross entropy loss.

After trained with 20 epochs, Figure 2 shows the loss and accuracy of the training and validation. After Epoch 3, the validation curve deviates from the training curve, indicating a possibly overfitting to the training set. However, since we are training a generative model, the categorical cross entropy, or negative log likelihood (NLL) is just one metric of the performance of a generative model. Especially in text generation, a temperature is typically used to adjust the diversity or novelty of the generated token. As pointed out in [21],in addition to NLL, a human study can be used as an oracle to judge the quality of generated text.

As an example, Listing 1 shows the top 3 generated Trump-style tweets, based on prediction softmax probability with three different temperature settings after 20 epochs training. It is surprising to see

[1] https://github.com/wangruowen/textgenrnn

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input (InputLayer)           (None, 15)                0         
_________________________________________________________________
embedding (Embedding)        (None, 15, 100)           1000200   
_________________________________________________________________
rnn_1 (CuDNNLSTM)            (None, 15, 128)           117760    
_________________________________________________________________
rnn_2 (CuDNNLSTM)            (None, 15, 128)           132096    
_________________________________________________________________
rnn_3 (CuDNNLSTM)            (None, 128)               132096    
_________________________________________________________________
output (Dense)               (None, 10002)             1290258   
=================================================================
Total params: 2,672,410
Trainable params: 2,672,410
Non-trainable params: 0
```

Figure 1: 3-layer LSTM baseline model

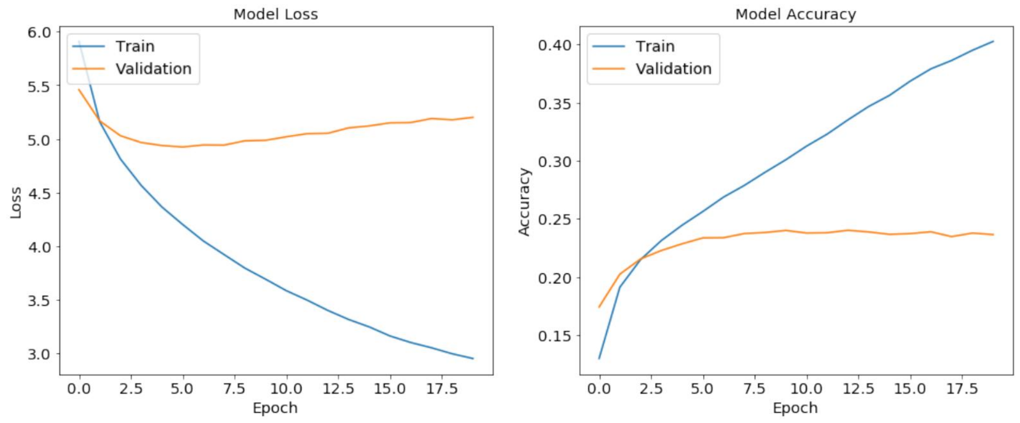

Figure 2: Baseline Model Loss and Accuracy

that just a plain 3-layer network can already pick up quite good Trump-style tweet language. Note that there is no pre-training. The network is trained from scratch for both embedding and LSTM weights. One reason of this result is that the training dataset does provide short and repetitive patterns for the model to capture, which is indeed Trump's tweet style, since the President often talks about certain political subjects repeatedly (e.g., Democrats, Hillary, etc.) and keeps posting similar tweets (e.g., there are more than 100 tweets of "i will be interviewed on ..." in the dataset). This shows that the dataset is not diversified enough, which is more likely to cause overfitting.

```
####################
Temperature: 0.2
####################
the democrats are so smart that most fake news media likes to do to do . " — @jack _ welch

thank you to all of the incredible volunteers and friends in the united states . i will be on
     the great state of wisconsin . we will never forget !

i will be interviewed on @foxandfriends at 7 : 00 a . m .

####################
Temperature: 0.5
####################
i will be interviewed on @foxandfriends at 7 : 00 a . m . enjoy !

. . . . . . . . . . .

i ’ m in the swamp —— — the best thing is the best !

####################
Temperature: 1.0
####################
secretary is not finding going on from mar — the person is one of candidates for clinton war ?
```

```
unbelievable crowd from brussels . hope he wins for ok in the middle east and jobs in that
     danger . everybody will be announcing @foxnews there will get but strong people . give
     dana and being gone for her problems and we will vote !

word took major two years to trump ' s old commander and have been more up . it was out of
     strong as no reporting . so dishonest !
```

Listing 1: Baseline model generated tweets after 20 epochs training (Not cherry picked).

4 GAN Model

Generative Adversarial Networks for text generation is still an active topic in NLP research, due to the difficulty of passing gradients to the generator given discrete tokens. Different architectures and approaches have been proposed recently, including using the Gumbel Softmax trick [11], MLE pre-training with reinforcement learning, such as SeqGAN [21, 2, 10, 15]. Without pre-training, it becomes even more difficult and needs additional technique to help train GAN, such as using curriculum learning [14] or Wasserstein GAN[8]. In the following subsections, we will describe a few aforementioned work we have tried and revised.

4.1 SeqGAN Keras Implementation

SeqGAN [21] treats the text generation as a decision making process, using already generated tokens as the current state to determine the next token to be generated. Then the discriminator's classification probability is used as a reward to train the generator via policy gradient. For the purpose of fast prototyping, we tried one Keras-based SeqGAN implementation[2] [20] by designing the generator to be the same as the baseline RNN model (the original paper used just one layer LSTM), and the discriminator to also be the same except replacing the baseline RNN's softmax with sigmoid to classify real or generated texts. We use the following cost function with Adam to minimize it (We negate the original cost function in the SeqGAN paper to change the optimization from maximize to minimize).

$$J(\theta) = -\sum_{y_1 \in Y} G_\theta(y_1|s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1) \tag{1}$$

Due to space constraint, please refer [21] for more detailed definitions and annotations. The generator is pre-trained with 20 epochs using the Trump tweets dataset. And the discriminator is pre-trained with one epoch. Then the whole model is trained with 20 epochs. In each epoch, we choose two sets of hyperparameters, one is that the generator is trained 5 times and then the discriminator is trained once. The other is both the generator and the discriminator are trained once.

4.2 Mode Collapse

Unfortunately, during training, the above implementation and hyperparameters quickly raise an issue that seems to be mode collapse as shown in Listing 2. Mode collapse [16] means the generated tokens are collapsed to one or a few tokens, in this case, "great", which is one of the most populuar token in the Trump tweet dataset. However, at this stage, this is not fully understood and still under investigation.

```
<BOS> the the new office . congress are a great great great great great great great
<BOS> . i will be a great great great great great great great great great great
<BOS> . i will be a great great great again ! <EOS> <PAD> <PAD> <PAD> <PAD>
```

Listing 2: Possible mode collapse issue appears during SeqGAN Keras training

4.3 Texygen Implementation

As GAN requires careful design and implementation, to avoid potential implementation or compatibility issues and isolate different issues for easy analysis (i.e. determine whether the training difficulty is caused by dataset input or implementation bugs or wrong cost functions), we also trained and evaluated our tweet generation task based on different GAN implementations with already fine-tuned

[2]https://github.com/wangruowen/SeqGAN

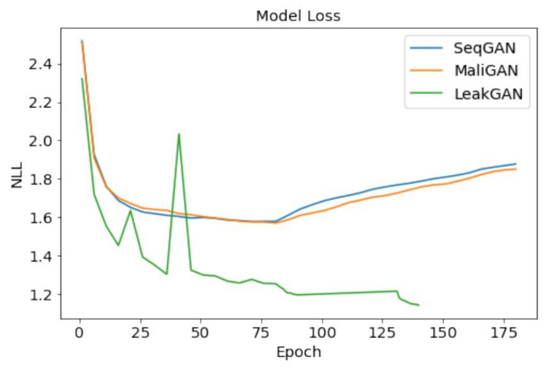

Figure 3: Validation NLL Losses of Three GAN models using the same Trump Tweets training set

hyperparameters for cross reference and comparison. One framework is called Texygen [22], which provides several GAN-based text generation implementations[3].

Here, we mainly focus on three GAN-based text generation models implemented in Texygen: SeqGAN [21], MaliGAN [2] and LeakGAN [9]. Apart from the aforementioned SeqGAN, MaliGAN improves SeqGAN by stablizing the training and mitigating the gradient saturating problem. LeakGAN is a further improvement that allows the discriminator to leak its own high-level extracted features to the generator to further help the guidance of the next token to be generated.

Figure 3 shows the validation NLL of the three models. We can see that both SeqGAN and MaliGAN start overfitting after 75 epochs, while LeakGAN can achieve lower NLL with some oscillation. Due to the time limit, currently only the overall losses are analyzed here. More detailed analysis and explanation on the losses with different hyperparameter settings are needed in the future work.

In addition, to further evaluate the quality of the generated tweets, we manually go through 100 tweets generated by each trained model respectively, assuming human as the oracle to judge how many of them are grammatically correct and furthermore, semantically making sense. Table 1 shows the number of generated tweets by each model. These tweets are manually checked and classified into four categories in terms of language quality. SeqGAN and MaliGAN are at the same level. It is still difficult for them to generate grammar-correct texts, let alone semantically correct ones. LeakGAN is much better compared to the previous two. Around 80 out of 100 tweets have at least partially correct grammar. There are even 20 tweets that look quite realistic. It also took much longer time to train LeakGAN than the other two.

Table 1: Number of generated tweets in different qualities by three GAN models

Models	Make no sense	Grammar partially correct	Grammar mostly correct	Make sense semantically
SeqGAN	35	46	12	7
MaliGAN	44	32	16	8
LeakGAN	18	32	30	20

5 Conclusion & Future Work

In this work, we explore applying Generative Adversarial Networks to generate tweets, using Trump Tweets as the training set. The results show that it is more difficult to train a GAN model than a basic RNN model. We have observed issues like mode collapse. We also tried fine-tuned GAN models and found different GANs have different performance in terms of text generation quality. In the future work, more efforts are needed to address training difficulty and design better cost functions. Larger dataset is also needed to further improve the diversity of the generated text and reduce the overfitting issue. Please refer to the github links for more details.

[3]https://github.com/wangruowen/Texygen

References

[1] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. *arXiv preprint arXiv:1511.06349*, 2015.

[2] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

[3] E. L. Denton, S. Chintala, R. Fergus, et al. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *NIPS*, pages 1486–1494, 2015.

[4] Factbase. Donald Trump Speech and Interview Transcripts. https://factba.se/transcripts, 2019. [Online; accessed Jan-22-2019].

[5] W. Fedus, I. Goodfellow, and A. M. Dai. MaskGAN: Better Text Generation via Filling in the _. *arXiv preprint arXiv:1801.07736*, 2018.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[7] A. Graves. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[9] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] R. D. Hjelm, A. P. Jacob, T. Che, A. Trischler, K. Cho, and Y. Bengio. Boundary-seeking generative adversarial networks. *arXiv preprint arXiv:1702.08431*, 2017.

[11] M. J. Kusner and J. M. Hernández-Lobato. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

[12] K. McKeown. *Text Generation*. Cambridge University Press, 1992.

[13] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[14] O. Press, A. Bar, B. Bogin, J. Berant, and L. Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*, 2017.

[15] S. Rajeswar, S. Subramanian, F. Dutil, C. Pal, and A. Courville. Adversarial generation of natural language. *arXiv preprint arXiv:1705.10929*, 2017.

[16] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[17] S. Scott and N. Srivatsa. Make AI Great Again: Generating Tweets in 'Presidential' Style Using Recurrent Neural Networks. http://cs230.stanford.edu/projects_spring_2018/reports/8284387.pdf, 2018.

[18] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[19] D. J. Trump. Trump Twitter Archive. http://www.trumptwitterarchive.com/archive, 2019. [Online; accessed Jan-22-2019].

[20] tyo yo. SeqGAN with keras. https://github.com/tyo-yo/SeqGAN, 2018. [Online; accessed Jan-22-2019].

[21] L. Yu, W. Zhang, J. Wang, and Y. Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*, pages 2852–2858, 2017.

[22] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Texygen: a benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100. ACM, 2018.