Text-to-Text Generative Adversarial Networks

Changliang Li
Institute of Automation
Chinese Academy of Sciences
Beijing, China
changliang.li@ia.ac.cn

Yixin Su
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
yixins1@student.unimelb.edu.au

Wenju Liu
Institute of Automation
Chinese Academy of Sciences
Beijing, China
wenju.liu@ia.ac.cn

***Abstract*—Generative Adversarial Networks (GAN), which are capable of generating realistic synthetic real-valued data, have achieved great progress in machine learning field. However, generator in GAN framework requires being differentiable, which means that the generator cannot produce discrete data, and it poses great challenge for GAN applied in Natural Language Processing (NLP) research. To unlock the potential of GAN in NLP, we develop a novel Text-to-Text Generative Adversarial Networks (TT-GAN), through which we can get generated text based on semantic information translated from source text. We demonstrate that our model can generate not only realistic texts, but also the source text's paraphrase or its semantic summarization. As our best knowledge, it is the first framework capable of generating natural language on semantic level in real sense, and gives a new perspective to apply GAN on NLP research.**

***Index Terms*—Generative Adversarial Networks, Natural Language Processing, Text Judgment, Text Summarization**

I. Introduction

Generative Adversarial Networks (GAN) [1] were recently introduced as a novel way to train a generative model. It has led to significant improvements in machine learning research field, such as image generation [2], video prediction [3] and other domains [4]–[6]. The basic idea of GAN is to pose a discriminator and a generator in a mini-max adversarial game. The discriminator has the task of distinguishing real samples in a dataset from fake samples, while the generator tries to produce fake samples that fool the discriminator.

One requirement imposed on the design of GAN framework is that the generator must be differentiable. Unfortunately, this means that the generator cannot produce discrete output, such as one-hot word or character representations. Removing this limitation is an important research direction that could unlock the potential of GANs on NLP. Solving the problem of generating texts from given texts gains interest in the research community. Though many attempts, such as using the reinforcement algorithm [7], concrete distribution [8] and Gumbel-softmax [9] and so on, have been done, the question is far from being solved.

In this work, we take translating text into its paraphrase or semantic summarization as illustration. For example, conditioned on the movie review "the movie has skilled actor and joyous music", we aim to generate its paraphrase such as "awesome music and best actors" or its semantic summarization "this film is great". In fact, semantic summarization can be viewed as another kind of paraphrase. For easy illustration, in the rest of the paper, we refer both general paraphrase and semantic summarization as paraphrase. To solve this challenging problem, it requires solving two sub-problems: first, learn a text feature representation that captures the important semantic information of source text; and second, use these features to synthesize paraphrase that a human might think it is plausible.

In pursuit of the goal, we propose a novel Text-to-Text Generative Adversarial Networks (TT-GAN) architecture. Our model firstly generates continuous text vector conditioned on source text through neural networks. At next step, we map the continuous text vector to a fixed style matrix. Each row of the output matrix is viewed a discrete token (word or any symbol) represented as word embedding. Then we search the corresponding token from discrete word vector space by semantic computation. The tokens are linked sequentially according to chain rule, and finally are formed to a natural language sentence. As consequence, the result shows that our model can generate realistic sentences using adversarial training. Furthermore, the generated sentence can be the source sentence's paraphrase. The experiment results have shown that our model can handle NLP problems based on GAN architecture.

The rest of our paper is structured as follows: Section II discusses related works, Section III introduces some background knowledge on which our approach based. Section IV gives a detailed description of our model, Section V analyzed experiment results and Section 6 summarizes this work and the future direction.

II. Related Work

GAN has drawn significant attention as a new machine learning model recently [1]. The task of GAN is to produce a generator model by a mini-max game. In a few years, many variations have been invented to improve the original GAN until now, such as f-GAN [10], Energy based GAN [11], info GAN [12], and w-GAN [13], which proves that GAN is a promising model.

Currently, A work that combines deep convolutional neural networks and GAN (DCGAN) [5] has been proposed, which proves that GAN has great potential on image generating area. Inspired by DCGAN, researchers have done much progress on image generating aspect. A model that utilize DCGAN within Laplacian pyramid framework [4] has been proved that

978-1-5090-6014-6/18/$31.00 ©2018 IEEE

it can produce higher quality images than original CDGAN. Meanwhile, DCGAN has been used to generate photo-realistic natural images by mainly updating the loss function [14].

Besides image generating, natural language processing (NLP) is another import task in machine learning area. However, GAN is considered to be difficult to apply on NLP [1] since the update of GAN based on continues space, but languages are discrete.

Though GAN on NLP is tough, there are several attempts on it. An effort that tries to learn words representation from documents [15] has bypass the problem of handling natural language sentences directly. In addition, researchers try to solve the discrete problem by modeling the generator as Reinforcement Learning (RL) called SeqGAN [16]. This model avoids the differentiate problem caused by gradient policy in generator. On the other hand, the discrete problem can be solved by mapping discrete natural languages into continuous vector spaces as well. Combining LSTM and CNN with generated vectors to do text generating [17], researchers successfully generated some realistic sentences. Another work, moreover, established a model that directly researches generating sentences from a continuous space using GAN [18], which is also a progress for GAN on NLP.

Traditional GAN and its applications only receive noise as input and produce output we expected in generator. We cannot decide which class to output if the expected output contains many categories. As solution, Convolutional Generative Adversarial Networks (CGAN) [19], are designed. It adds conditions as part of input both in generator and discriminator. Therefore, GAN can map conditions with expected output, and we are able to generate more accurate output when we feed conditions while testing. [2] is a typical application of CGAN that takes descriptions as conditions and corresponding images as output. Then generator can generate figures according to the descriptive words we input.

In our model, we pushed GAN on NLP further based on previous works. The concept of CGAN is used in our model so that finally it is able generate a judgment sentence or a paraphrase according to the sentence we feed as condition. Meanwhile, the experiment result shows that the generator in our model also learn some syntactic rules while training.

III. Background Knowledge

In this section, we briefly describe some background knowledge.

A. Generative and Discriminative Models

A discriminative model learns a function that maps the input data x_{data} to some desired output class label y_{data}. In probabilistic terms, they directly learn the conditional distribution $p(y_{data}|x_{data})$.

A generative model tries to learn the joint probability of the input data and labels simultaneously. This can be converted to $p(y_{data}|x_{data})$ for classification via Bayes rule, but the generative ability could be used for something else as well, such as creating likely new (x_{data}, y_{data}) samples.

Both types of models are useful, but generative models have one interesting advantage over discriminative models. Generative models have the potential to understand and explain the underlying structure of the input data even when there are no labels. This is very desirable when working on data modelling problems in the real world, as unlabeled data is of course abundant, but getting labelled data is often expensive at best and impractical at worst.

B. Generative Adversarial Networks

The GAN framework establishes two distinct players, a generator *G* and discriminator *D*, and poses the two in an adversarial game. The discriminator model $D(x)$ computes the probability that a point x in data space is a sample from the data distribution (positive samples) that we are trying to model, rather than a sample from our generative model (negative samples). Concurrently, the generator uses a function $G(z)$ that maps samples z from the prior $p(z)$ to the data space. $G(z)$ is trained to maximally confuse the discriminator into believing that samples it generates come from the data distribution. The generator is trained by leveraging the gradient of $D(x)$, and using that to modify its parameters. The solution to this game can be expressed as following [1]:

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

where G and D are generative model and discriminative model respectively, $p_{data}(x)$ is the distribution of data x and p_z is the randomly assigned noise distribution.

The generator G and the discriminator D can be trained in two stages: (a) Train the discriminator to distinguish the true samples from the fake samples generated by the generator. (b) Train the generator so as to fool the discriminator with its generated samples. From a game theory point of view, the convergence of a GAN is reached when the generator and the discriminator reach a Nash equilibrium.

C. Conditional Adversarial Nets

Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y. y could be any kind of auxiliary information, such as class labels or data from other modalities [19].

Generative adversarial nets feeds extra information y into the both the discriminator and generator. As a result, it takes the condition information into account. The generator utilizes both the prior input noise $p_z(z)$ and condition information y as input to learn hidden representation. The adversarial training can produce numerous this hidden representations in flexible way.

In the discriminator x and y are presented as inputs, and then feed-forwarded to a discriminative model (for example neural networks). The objective function of a Generative adversarial nets model is defined as:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x|y)}[\log D(x|y))] + E_{x\sim p_z(z)}[\log(1 - D(G(z|y)))] \quad (2)$$

where D and G are now conditional models that based on condition y, $p_{data}(x|y)$ is the distribution of data x over the condition y.

IV. TEXT-TO-TEXT GAN

In this section, we describe our Text-to-Text Generative Adversarial Network (TT-GAN) in detail and introduce the model training algorithm.

A. Network Architecture

Our approach aims to generate paraphrase text conditioned on given source text. We firstly describe the notation used in this paper, such as input, output and noise space, each with an associated probability distribution:

- R^Z is a noise space used to seed the generative model, where Z is the dimension of the noise input to generator G. Noise data $z \in R^Z$ are sampled from a noise distribution $p_z(z)$.
- R^Y is an embedding space used to condition the generative model on text representation, drawn from the training data, where Y is the dimension of source text vector.
- R^X is the data space which represents an semantic output from the generator or input to the discriminator. X is the dimension of the text vector.

Then we define two functions:

1) $G : (R^Z \times R^Y) \rightarrow R^X$ denotes generator, which accept noise data $z \in R^Z$ along with a source text vector $y \in R^Y$ and produce a generated text vector $x \in R^X$.
2) $D : (R^X \times R^Y) \rightarrow [0, 1]$ denotes discriminator, which accepts a text x and conditioned y. It aims to predict the probability under condition y that x came from the empirical data distribution rather than from the generative model.

Figure 1 gives the overall illustration of our model.

In generator stage, we firstly encode the input text into a fixed-length vector through an encoder. Meanwhile, our model sample noise data from prior $p_z(z) \sim N(0, 1)$. We employ simple Gaussian noise model in this paper.

Then, we merge the text vector y and noise z as the input, which is feed-forwarded to the generator. In this paper, we use a fully-connected neural network as the generator. As a result, we get a generated text vector $\tilde{x}$. Then we transfer $\tilde{x}$ to a matrix M, where each row of which represents a word vector. Next, we mapping the continuous word vectors into discrete texts by comparing the similarity between generated word vectors and the word embedding we used as input. A word vector will be regarded as the word with the most similar corresponding embedding. At last, we employ chain rule to obtain the final generated text.

In discriminator D stage, we examine samples to determine whether they are real or fake. The discriminator learns using traditional supervised learning techniques, dividing inputs into two classes (real or fake). We employ a fully-connected neural network followed by a softmax function to compute the final score from D.

Algorithm 1 Text-to-Text training algorithm

Input: source text set S and real paraphrase text set P
for $s, p \in (S, P)$ **do**
  $Encoder(s) \rightarrow y$, //encode source text into vector
  $Encoder(p) \rightarrow x$, //encode real paraphrase text into vector
  $z \sim N(0, 1)$, //draw noise sample
  $G(z, y) \rightarrow \tilde{x}$, //fake paraphrase text through generator
  $D(x, y) \rightarrow sr$, //the score of source text and its real paraphrase
  $D(\tilde{x}, y) \rightarrow sf$, //the score of source text and its fake paraphrase
  $L_D = \log(sr) + \log(1 - sf)$, // loss function of discriminator
  $D - \alpha\frac{\partial L_D}{\partial D} \rightarrow D$, // update discriminator
  $L_G = \log(sf)$, // loss function of generator
  $G - \alpha\frac{\partial L_G}{\partial G} \rightarrow G$, // update generator
end for

B. Model Training

In our model, the discriminator observes two kinds of inputs: source text with real paraphrase, source text with synthetic paraphrase. While training, each pair of (source text, paraphrase text) is viewed as joint observations. The discriminator is trained to judge whether the paraphrases are real or fake given source text. While the generator receives source text and produce a fake paraphrase text, which will joint with the input source text as a pair and be sent to the discriminator. For each batch of training, both generator model and discriminator model will be trained, but are not synchronous. That is to say, while training one model, the parameters of another model will be fixed. This ensures that the GAN will performs the "mini-max game" correctly. As training goes on, the generator gets better to generate plausible paraphrase texts, which align with the source texts. Likewise, the discriminator must learn to evaluate whether samples from G.

Algorithm 1 gives the training pseudo code. Firstly, we encode the source text s and real paraphrase text p into continuous vectors. Then, we draw noise sample z. After that, we obtain fake text $\tilde{x}$ through generator $G(z, y)$. The discriminator D takes source text with real paraphrase x or fake paraphrase $\tilde{x}$ vectors as inputs, and outputs two scores sr and sf respectively. sr is the score of source text y and its real paraphrase x. sf means the score of source text y and fake paraphrase $\tilde{x}$. Note that we use $\frac{\partial L_D}{\partial D}$ to indicate the gradient of generator's objective with respect to its parameters, and likewise for G. Next, we take a gradient step to update all network parameters.

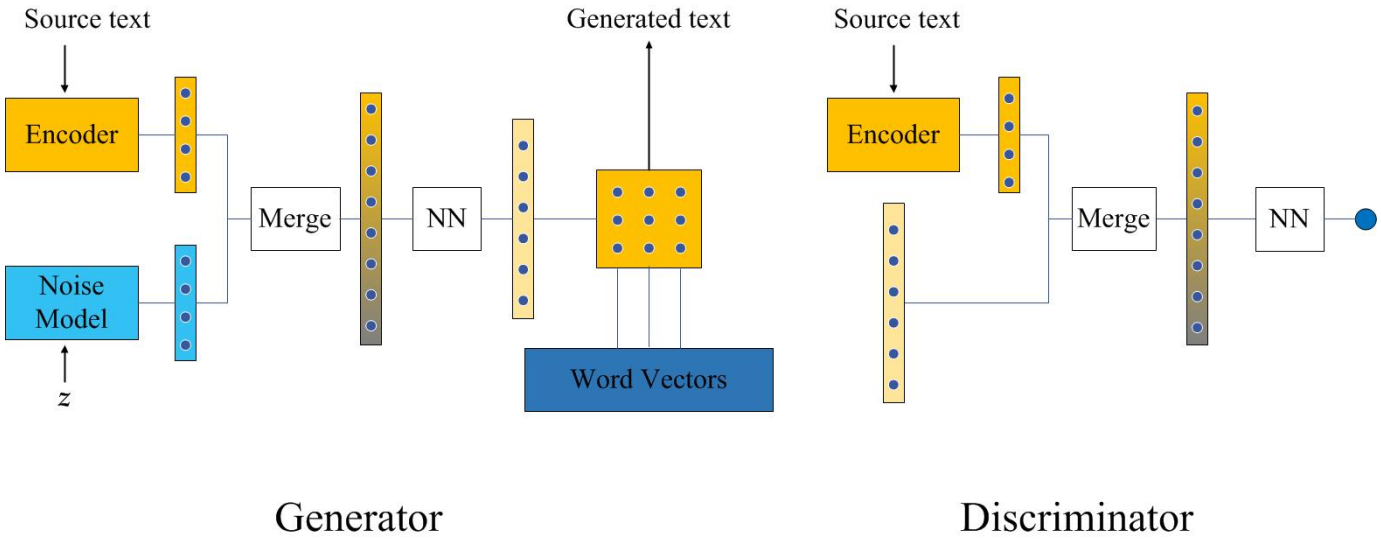

Fig. 1. Text-to-Text Generative Adversarial Networks

V. Experiment

In this section, we firstly describe the experiment data, which is composed of movie reviews (including Chinese and English). Then we conduct experiments on two tasks, generating paraphrases and semantic summarization (simple judgment, positive or negative) given source reviews.

A. Data

In generating semantic summarization task, we investigate the ability of our model in generating simple judgment whether the movie is good or bad. Meanwhile, unlike classification, we use different expressions to represent the judgment, also called semantic summarizations. For instance, to indicate that the movie is good, we can use sentence "this film is awesome" or "how awesome the movie is". Therefore, judgment for reviews are not totally the same. Table I gives some training examples. For better being understood, Chinese reviews are translated into English, as shown in their following brackets.

In the following generating paraphrases task, we still use the same movie reviews, but map to a batch of new judgement. Table II gives some instances in training set.

As shown in table II, "joyous music" indicates that the movie has a good story, and "skilled actors" means that the actors are very professional. This task is more complicated than previous experiments, because there are more flexible sentence expressions. Therefore, we are able to validate whether TT-GAN can generate sentences that are more complicated correctly.

B. Experiment Setting

These reviews are natural language and are unable to put into the model. To transform them into continuous form, we mapped words and symbols into word vectors. For Chinese reviews, we utilized Word2Vec [20] model to generate word vectors. Corpus used is reviews collected form the largest Chinese movie review website www.douban.com. On the other hand, English words representation comes from the existing library of [21]. Each word and symbol is mapped into a 50-dimension vector. Note that all reviews is ended with a '$< /s >$' token. Finally, all words in a review are encoded into a fixed-length vector through an encoder, which will be the input of generator. For simplicity, we concatenated all word embedding together in encoder stage.

In our all experiments, we used the Euclidean distance to measure the similarity between output text vectors and word embedding. That is to say, when we are mapping a continuous output vector into the discrete word, we choose the word that has word embedding with smallest Euclidean distance to the output vector.

TABLE I
Example of semantic summarization dataset

Data set	Source text	paraphrase
English	This movie has motivational content and joyous music	How awesome the movie is
English	The movie has awkward actors with disgusting music	It is a boring movie
Chinese	此片故事混乱,音乐很差"This movie contains tedious plot and disgusting music"	电影太差劲了"It is a bad movie"
Chinese	此片情节丰富紧凑,歌舞非常欢乐"The film has tight plot and joyous music"	电影太差劲了"It is a great movie"

TABLE II
EXAMPLE OF PARAPHRASES DATASET

Data set	Source text	paraphrase
English	This film contains joyous music with skilled actors	Fantastic music with good actors
English	This movie has motivational con-tent and tight plot	Good story and professional director
Chinese	此片音乐令人失望，情节冗长”This film has disgusting music and tedious plot”	糟糕的音乐，差的情节”Awful music with un-promising director”
Chinese	此片情节感人，演员超棒”This film has motivational content with skilled actors”	好故事，好演员”Good story with good actors”

C. Training

With input data generated above, we design four experiments to evaluate our model's ability in generating texts. In the first two experiments, we generate different sentences to represent two semantic summarizations - the movie is good or bad. In other two experiments, we generate paraphrase given movie reviews.

GAN is extremely hard to train. Therefore, we must carefully design the model to guarantee that it works correctly.

Both discriminator and generator are neural networks, which are simple but efficient. In discriminator, gradient descent method is used as optimizer, while in generator, momentum optimizer is preferred. It is because that the function of discriminator is to judge whether a sentence is right or fake giving a review. The gradient direction should always be correct. However, generator may receive inaccurate feedback at first. It happens frequently that generator may updated towards a wrong direction. Meanwhile, generator often gets stuck in a local optimal point. The fake result produced is easily recognized by discriminator, which may crash the whole model. Therefore, momentum optimizer is a good choice to avoid the stuck.

After thousands of epochs training, the model updates correctly and generate relatively good results. Figure 2,3,4 and 5 show that with the training proceeding, generator generates vectors for one training data approaches to the real sentence vector (vectors are reduced to 2-dimensional using PCA method).

Blue dots in figure 2,3,4 and 5 are vectors produced by generator at certain epochs, while the red dot in the end is the vector that generator is expected to approach. From the figure, it is easy to see that at beginning, generator produces vectors that are far from real vector (red point). It tries different vectors to find the rule so that in first several thousands of epochs, points are dispersive. After that, generator gradually finds the rule and begins to approach real vector. Besides, it is interesting that there are some vibrates before reaching around

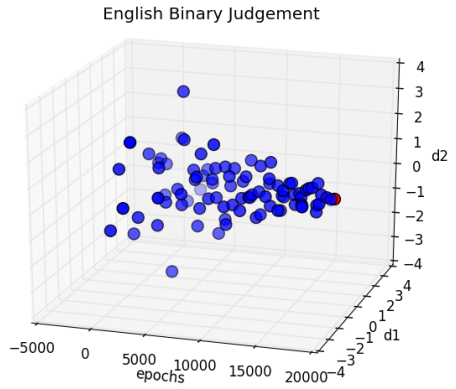

Fig. 2. English Binary Judgment

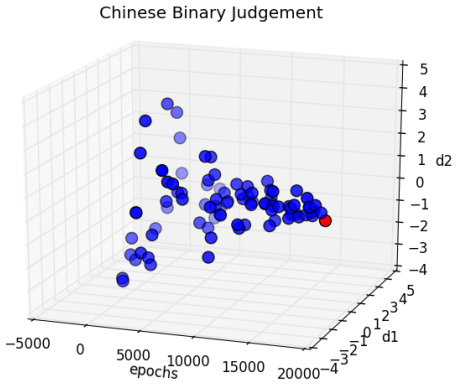

Fig. 3. Chinese Binary Judgment

true distribution. That may be the result of using momentum optimizer that will avoid generating the same result for all inputs and crashing in a wrong area.

After full training, we can run our model on test data to say whether it can generate meaningful output.

D. Test Result

While testing, we input reviews that have never appear in training datasets, in order to see whether the model meets our requirement.

1) Semantic Summarizations (Simple Judgment): Table III shows the experiment result on generating semantic summarizations task. The result shows that our model can judge the review correctly. Meanwhile, there are probabilities that reviews in training data can generate judgment in different expressions.

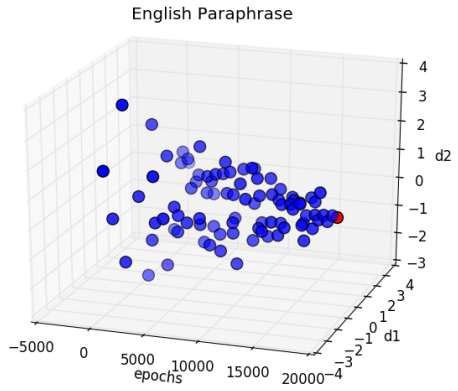

Fig. 4. English Paraphrase

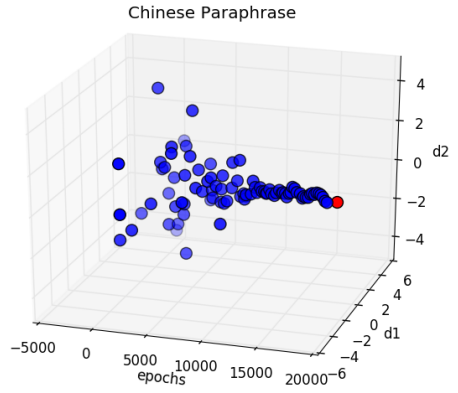

Fig. 5. Chinese Paraphrase

This illustrates that TT-GAN is able to learn useful semantic information as well as to generate plausible semantic summarizations.

2) Paraphrase: Since TT-GAN has presented the potential of generating semantic summarization, Table IV shows the experiment result on generating paraphrases task. In order to better illustrate the difference between generating paraphrase and semantic summarizations, we use the same source data as shown in table III.

The result above demonstrates that the model is capable of discovering the correspondence between source reviews and paraphrase. That is to say, the model can catch useful information in a sentence and reveals as corresponding paraphrase.

For instance, we test sentence "this movie has tight plot and skilled actors", which information combination has never appeared in training dataset. As consequence, the model output paraphrase of the review, "professional director with good actors". This shows that the information in the review has been correctly detected and plausible paraphrase is generated.

In addition, we use ROUGE [22] to numerically evaluate the test results. In this paper, the two tasks, semantic summarizations and Paraphrase, are evaluated using ROUGE-N metrics respectively. N refers to the n-grams used in evaluation and we choose N as 1, 2 and 3.

TABLE III
SIMPLE JUDGMENT (TRAINING INPUT IS THE INPUT REVIEWS, GENERATOR OUTPUT IS THE SENTENCE THAT GENERATOR ACTUALLY OUTPUT)

Data set	Source text	Generated text
English	This movie has tight plot and skilled actors	How awesome the movie is
English	This movie has tedious plot and awkward actors	This is a bad movie
Chinese	此片歌舞非常欢乐，内容感人"The movie contains joyous music and motivational content"	电影特别好看"This is a fantastic movie"
Chinese	此片音乐一般，情节冗长"This film has disgusting music and tedious plot"	电影很差"It is a boring movie"

TABLE IV
PARAPHRASE (TRAINING INPUT IS THE INPUT REVIEWS, GENERATOR OUTPUT IS THE SENTENCE THAT GENERATOR ACTUALLY OUTPUT)

Data set	Source text	Generated text
English	This movie has tight plot and skilled actors	Professional director with good actors
English	This movie has tedious plot and awkward actors	Unpromising director with boring actors
Chinese	此片歌舞非常欢乐，内容感人"The movie contains joyous music and motivational content"	很棒的音乐，很好的故事"Awesome music and good story"
Chinese	此片音乐一般，情节冗长"This film has disgusting music and tedious plot"	音乐糟糕，故事不好"Awful music and bad story"

TABLE V
ROUGE EVALUATION RESULTS ON TWO TASKS IN ENGLISH AND CHINESE

Datasets and Tasks	Metrics		
	ROUGE-1	ROUGE-2	ROUGE-3
English Judgment	0.802	0.605	0.397
English Paraphrase	0.657	0.342	0.133
Chinese Judgment	0.743	0.615	0.343
Chinese Paraphrase	0.693	0.515	0.233

Table V gives the result of the numerical evaluation. The results are promising, which proved the ability of TT-GAN on fulfilling two tasks in both English and Chinese environments. One interesting point is that in the same dataset, judgment tasks receives better ROUGE value than paraphrase tasks on all three ROUGE metrics. It may because that the output of judgment tasks usually have the fixed sentence patterns and are shorter than paraphrase outputs, which makes the generative distribution become easy to learn. However, from table III and table IV, it is clear to see that even there are some mismatch between references and output sentences (which may pull down the ROUGE value), the general meaning are concordant.

In summary, the results of the experiments have proved the potential of TT-GAN on discovering information of expression and generating plausible judgment and paraphrases.

VI. Conclusion and Future Work

In this work, to unlock the potential of GAN in NLP, we proposed a novel Text-to-Text Generative Adversarial model. Four experiments are designed on movie reviews dataset. We demonstrated that our model can synthesize many plausible realistic texts of a given source text. Furthermore, our work has shown the potential for generating paraphrase and semantic summarization of reviews. We will further scale up the model to flexible types of text, and explore more interesting and useful applications.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, and M. Mirza, "Generative adversarial nets," *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[2] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[3] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *arXiv preprint arXiv:1511.05440*, 2015.

[4] E. Denton, S. Chintala, and S. Chintala, "Deep generative image models using a laplacian pyramid of adversarial networks," *Advances in neural information processing systems*, pp. 1486–1494, 2015.

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *arXiv preprint arXiv:1511.06434*, 2015.

[6] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, pp. 2226–2234, 2016.

[7] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3–4, pp. 229–256, 1992.

[8] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *arXiv preprint arXiv:1611.00712*, 2016.

[9] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *arXiv preprint arXiv:1611.01144*, 2016.

[10] B. C. S. Nowozin and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," *Advances in neural information processing systems*, 2016.

[11] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," in *arXiv preprint arXiv:1609.03126*, 2016.

[12] Y. D. X. Chen, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in Neural Information Processing Systems*, 2016.

[13] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," in *arXiv preprint arXiv:1701.07875*, 2017.

[14] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, and et al, "Photo-realistic single image super-resolution using a generative adversarial network," in *arXiv preprint arXiv:1609.04802*, 2016.

[15] J. Glover, "Modeling documents with generative adversarial networks," in *arXiv preprint arXiv:1612.09122*, 2016.

[16] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: sequence generative adversarial nets with policy gradient," in *arXiv preprint arXiv:1609.05473*, 2016.

[17] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," 2016.

[18] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *arXiv preprint arXiv:1511.06349*, 2015.

[19] M. Mirza and S. Osindero, "Conditional generative adversarial nets," in *arXiv preprint arXiv:1411.1784*, 2014.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *arXiv preprint arXiv:1301.3781*, 2013.

[21] D. Tang, B. Qin, and T. Liu, "Aspect level sentiment classification with deep memory network," in *arXiv preprint arXiv:1605.08900*, 2016.

[22] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," *Text Summarization Branches Out*, 2004.