TD-GAN for Chatbot Text Generation

Yu Huang[1], **Wentai Zhang**[1], **Yunwen Zhou**[2]
[1]Department of Mechanical Engineering [2]Department of Electrical and Computer Engineering
Carnegie Mellon University, PA
`yuh1, wentaiz, yunwenz@andrew.cmu.edu`

1 Introduction

A dialog system is a computer system aims to converse with a human. There are two major structures: retrieval-based model and generative model. Retrieval-based model selects answers from fixed database. In other word, it is grammatical errors and improper wordings free while it is not able to generate contents itself or generate unseen conversational style. Our goal is to train a generative model which generates proper responses with various sentence lengths based on all previous contexts. During pretrain, we constructed a encoder-decoder generator pretrained with seq2seq learning process(1), along with WGAN(2) pretained critic; in adversarial training, we used policy gradient method (3) to collect gradient information from scores for updating generator parameters. During testing, we found our model shows the ability to use diverse words from the vocabulary set. However, due to the limited layer size, limited number of hidden units, and short training time (around 2 days on one gtx1080 gpu), our model is not able to made reasonable response to all input sentences.

2 Related work

Trials of applying reinforcement learning with handcrafted reward has been explored by Jiwei Li et al. (4) and more flexible reward design could boost performance. Seq2seq model(1) is a basic Retrieval-based model but still strong in performance. Ilya Sutskever et al. proposed this model to introduce drastic parameter reduction on simple retrieval-based model.

Ian J. Goodfellow et al. introduced GAN (5), a new learning structure for deep generative model. However, GAN is often used in image generation, and our task is text generation. One major obstacle for sentence generation is that it is impracticable to use GAN to do direct gradient passing from discriminator to generation, because words in sentence generated by deep network are time dependant discrete tokens.

Lantao Yu et al.(6) addressed this issue by applying policy gradient method in reinforcement learning to his SeqGAN model, so that information passing from discriminator to generator is feasible. However, the seqGAN model requires tremendous training time.

Jiwei Li et al.(7) introduced another approach to solve the infeasible training problem. Their adversarial evaluation network directly trains a discriminator capable of discriminating partially completed sentences. However, authors admit that "Compared with the Monte Carlo search model, this strategy is significantly more time-effective, but comes with the weakness that the discriminator becomes less accurate after partially decoded sequences are added in as training examples. We find that the MC model performs better when training time is less of an issue."

To improve the traditional GAN, Martin Arjovsky et al.(2) designed a Wasserstein GAN (WGAN) to keep sample diversity and solves collapse mode problem and to introduce a loss function which quantitatively scores the generator.

When designing our model, we followed the basic framework of the Seq2seq model and made 2 main adjustments. First, our discriminator assigns scores to generated samples instead of probability or other modified training process. Second, we estimate the score or reward at max cut length λ instead of after sentence termination. Note that when $\lambda = 1$, our frame work is similar to adversarial model (7) and when λ is maximum sentence length, our model will be similar to seqGAN(6). In conclusion, we introduced these extensions to compromise between generation performance and training time consumption. We will discuss the detailed model architecture and training algorithm in next section.

3 Proposed method

3.1 Model Architecture

We designed our model based on the SeqGAN(6), the overall architecture is denoted as figure 1.

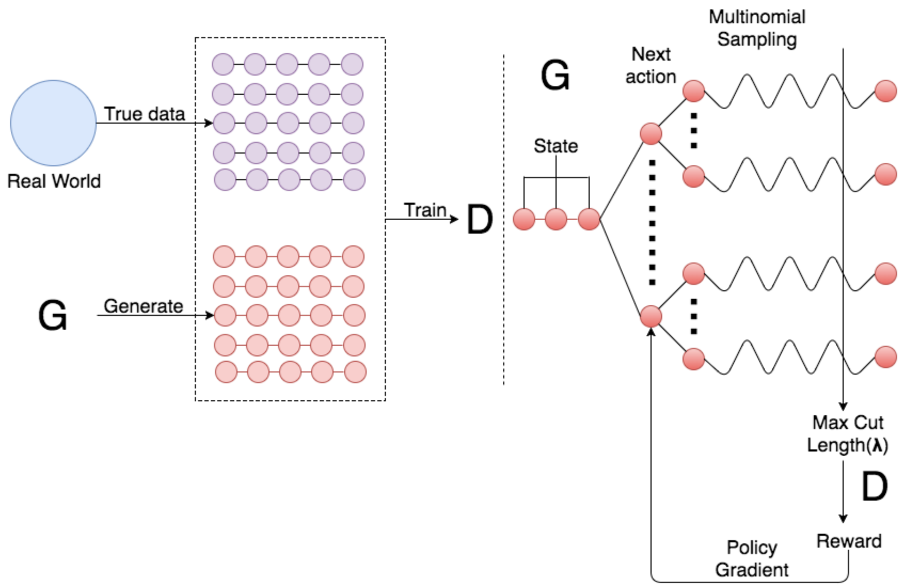


Figure 1: Model architecture of TD-GAN. The left part shows that D is trained by the true data and the generated data from G. The right part shows that the sentence is generated by G via multinomial sampling. Then G is updated by the rewards from D.

First, given true data, real text from Twitter Corpus, the generator G with weight parameter θ produces a sequence $\hat{Y}_{1:T} = (y_1, y_2, ..., y_T), y \in \{Y\}_V$ where $\{Y\}_V$ is our vocabulary set and T, V are maximum sentence length and vocabulary size respectively.

We interpret this problem based on reinforcement learning. In time step t, the current produced tokens are $\hat{Y}_{1:t}$, in which $\hat{Y}_{1:(t-1)} = (y_1, ..., y_{t-1})$ is previous state and the action is to select a current token $\hat{Y}_t$. Thus the policy model(generator) $G_\theta(y_t|\hat{Y}_{1:(t-1)})$ is stochastic, whereas the state transition is deterministic after an token has been chosen, i.e. current state $s_t = \hat{Y}_{1:t}$ is determined by the previous state s_{t-1} and current $\hat{Y}_t$ sampled from generator. Once the sentence completed to a certain length λ, it should be fed into discriminator D to obtain a score $D_\phi(\hat{Y}_{1:\lambda})$. Then the gain for a given time step is estimated by λ-step temporal difference estimation R. Sutton et al. (3):

$$\begin{aligned} Gain(\hat{Y}_{1:t}) = \gamma \mathbb{E}_{\hat{Y}_{(t+1):max(t+\lambda,T)} \sim G(\hat{Y}_{1:(t+1)})}[D_\phi(concatenate(\hat{Y}_{1:(t+1)}, \hat{Y}_{(t+1):max(t+\lambda,T)}))] \\ + \gamma^2 \mathbb{E}_{\hat{Y}_{(t+2):max(t+\lambda,T)} \sim G(\hat{Y}_{1:(t+2)})}[D_\phi(concatenate(\hat{Y}_{1:(t+1)}, \hat{Y}_{(t+1):max(t+\lambda,T)}))] \\ + ... + \gamma^\lambda D_\phi(\hat{Y}_{1:max(t+\lambda,T)}) \end{aligned} \tag{1}$$

where γ is the discount rate, and $\mathbb{E}$ is the score expectation. Then we replace the expectation by doing multiple times of sentence sampling and calculate the average score:

$$\mathbb{E}_{Y_{(t+1):max(\lambda,T)} \sim G(Y_{1:(t+1)})}[D_\phi(concatenate(\hat{Y}_{1:(t+1)}, \hat{Y}_{(t+1):max(t+\lambda,T)}))] = \sum_i^S D_\phi(\hat{Y}_t^{(i)}) \tag{2}$$

where $concatenate(\hat{Y}_{1:t}, \hat{Y}_{t:(t+\lambda)})$ denotes completed sentence by given time step t up till time step $t+\lambda$ which exploited the multinomial sampling method in figure 1. We define $\hat{Y}_t^{(i)}$ as completed sentences from t to $max(\lambda, T)$. In addition, when $D_\phi(\hat{Y}_{1:\lambda})$ is calculating a score, it actually uses the embedded sentences, $score = D_\phi(embedding(\hat{Y}_{1:\lambda}))$, but we omitted the embedding look-up step for notation simplification in all equations in this paper. $Gain(\hat{Y}_{1:t})$ is parameters used to update policy gradients in TD-GAN update algorithm. As shown in figure 1, we also train a discriminator D with weight parameter ϕ for updating the generator G_θ. Different from seqGAN (6), our model uses WGAN(2)(8) instead of traditional GAN framework(5)(Goodfellow et al., 2014). More specifically, vanilla GAN was proven to be unstable during training process and its modified target function has no theoretical ground according to I. Goodfellow (5), however, Arjovsky claims that WGAN has more stable training process and the output score can be positive or negative which resembles reward in reinforcement learning setup and we will use score or reward interchangeably in the rest of the paper. Another advantage of our modification is that temporal difference estimation does not require complete the sentence to the maximum length which save generator evaluation time. Therefore, we consider that this extension will be helpful in (1)saving computational time and (2)stabilizing training process.

As mentioned, $D_\phi(\hat{Y}_{1:T})$ is a score indicating the goodness of a generated sequence $\hat{Y}_{1:T}$. In other words, the training objective for discriminator D_ϕ is to score real data high and score "false data" low. This can be implemented by minimizing the following loss (8)(I. Gulrajani et al., 2017):

$$\mathcal{L}_D = D_\phi(\hat{Y}_{1:\lambda}) - D_\phi(Y_{1:\lambda}) + \alpha(||\nabla_\phi D_\phi(\tilde{Y})||_2 - 1)^2 \tag{3}$$

Such discriminator loss with gradient penalty was first introduced as blended images in (8)(I. Gulrajani et al., 2017). In detail, blended images is the linear combination of fake data and real data. Originally, I. Gulrajani designed such algorithm for images, we introduce the same idea to text by doing linear combination with coefficient $\epsilon \in [0, 1]$ of sentence embedding:

$$embedding(\tilde{Y}) = \epsilon \times embedding(\hat{Y}_{1:\lambda}) + (1-\epsilon) \times embedding(Y_{1:\lambda}) \tag{4}$$

then the weights update equation for discriminator is:

$$\phi_{s+1} = \phi_s + \alpha_d \nabla_\phi \mathcal{L}_D \tag{5}$$

To quickly sum up, as our model architecture illustrated in Figure 1, the discriminative model D_ϕ is trained by providing positive examples from the real sequence data and negative examples from the synthetic sequences generated from the generative model G_θ. At the same time, the generative model G_θ is updated by employing policy gradient and temporal difference learning on the basis of the estimated gain received from the discriminative model D_ϕ. The reward is estimated by the score that discriminative model D_ϕ gives.

3.2 Training Process

Next, we will briefly describe our methodology in pretrain. Our generator is based on LSTM-cell encoder-decoder model. During encoding, encoder takes in the question embedding and finalize its hidden state. Then decoder's hidden state is initialized by encoder final state and then start generating answers. During pretrain, we apply greedy decoding to generator, which suggests taking the word with the maximum probability of decoder output and then feed it into decoder to make the next prediction. As vanilla seq2seq model, the training objective is minimizing the cross entropy loss for each time step with respect to target sequences (1) (I. Sutskever). Specifically, for each given real response sentence batch $\{Y_i\}_B$ with batch size B, the objective is:

$$min_\theta - \sum_{i=1}^{B} \log(G_\theta(Y_{t,i}|Y_{1:(t-1),i}))/B \tag{6}$$

After generator and discriminator are pretrained, we start adversarial training. For each training epoch, we sample a batch with N pairs of questions and answers from training set. Then N answering sentences $\{\hat{Y}_{1:T}^{(i)}\}_N$ are generated by feeding questioning sentences into encoder in generator and applying multinomial sampling to decoder. Then we apply 1 to estimate $Gain(y_t|\hat{Y}_{1:(t-1)}) = Gain(\hat{Y}_{1:t})$. Finally the policy gradient update equation mentioned before for learning step s is:

$$\theta_{s+1} = \theta_s + \alpha_g Gain(y_t|\hat{Y}_{1:(t-1)}) \bigtriangledown_\theta \log(G(y_t|\hat{Y}_{1:(t-1)};\theta)) \tag{7}$$

where α_g is the learning rate for generator.

We summarized the above into algorithm 1. Note that hyperparamters are still tentative, we will report those in final submission.

Algorithm 2 TD-GAN Training Algotihm

```
Require: initialized generator $G(s;\theta)$ and discriminator $D(s;\phi)$; $N_G$ generator pretrain epoch; $N_D$ discriminator pretrain epoch; $N_A$ adverserial training epoch; $M_A$ number of samples generated in each adversarial training epoch.
 1: generator pretrain
 2: for $i = 1$ to $N_G$ do
 3:     seq2seq pretrain minimize sequence to sequence loss 7.
 4: end for
 5: discriminator pretrain
 6: for $i = 1$ to $N_D$ do
 7:     sample { $Y^{(i)}\}_{i=1}^{M}$ from real data set $\{Y\}$ and $\{\hat{Y}^{(i)}\}_{i=1}^{M}$ from generated data set $\{\hat{Y}\}$
 8:     pretrain discriminator minimize equation 3
 9: end for
10: Adversarial Training
11: for $i = 1$ to $N_A$ do
12:     sample { $Y^{(i)}\}_{i=1}^{M}$ from real data set $\{\hat{Y}^{(i)}\}_{i=1}^{M}$ from generated data
13:     for $j = 1$ to $M_A$ do
14:         compute $Gain(Y_t|Y_{1:(t-1)})$ using equation 2 and 1
15:     end for
16:     update $\theta$ in $G(s,\theta)$ using equation 7
17:     update $\phi$ $D(s,\phi)$ using equation 5
18: end for
```

3.3 Limitations

We trained word embedding on Word2Vec model. This embedding is not constructed specifically on our data set, which means some words may have different context. Another important limitation is that our discriminator is only designated to make a judgment on whether the sentence come from the real data set. However, if we really want the chatbot to be realistic, the discriminator should be able to score the relevance but this is not implementable on our algorithm. Therefore, our generator may be trained to generate meaningful sentences but might not be highly relevant to questions.

4 Data Description

We use Twitter Corpus Dataset(`https://github.com/Marsan-Ma/chat_corpus`) to train our Chatbot Text Generator. The dataset collects over 750,000 lines of tweets and comments. The text file saves original tweets in odd rows and the corresponding retweets in the even rows. For example, a typical pair of tweet and retweet looks like:

"worth checking out?"

"yes! i'm not sure if/when video will be available online. but it was incredibly powerful."

One interesting mark about this dataset is that it contains emojis:

"happy birthday gab ❤️ pe u have the best day, keep killin it love u always ❤️"

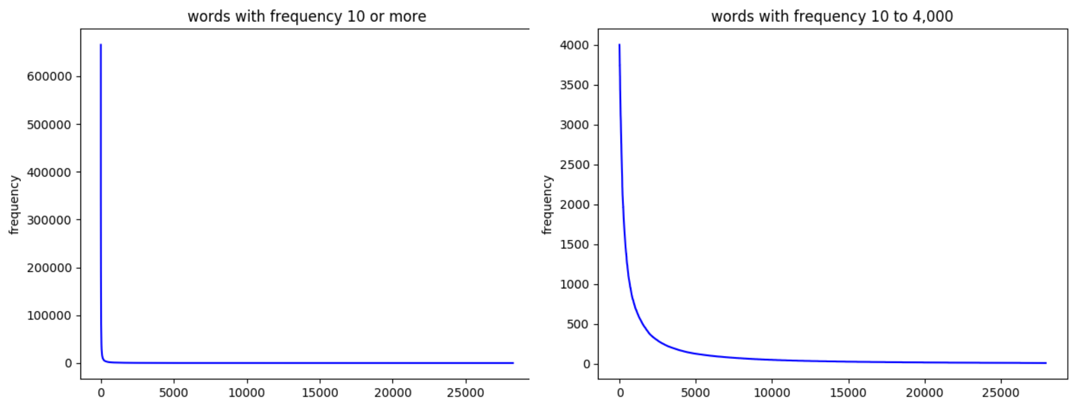
"thanks girly love you lots ❤️❤️❤️"

Before starting training, we split the dataset into three parts: 1,000 pair of tweets (2,000 sentences) as test dataset, 3,000 pair of tweets (6,000 sentences) as validation dataset, and the rest as training dataset.

5 Results

5.1 Pre-processing

During the pre-processing, we first pad a space character between words, punctuation and emojis. While defining a vocabulary set, we find the text corpus includes around 230,000 unique words and emojis. Then we filtered out tokens with frequency less than 10. Fig.2a shows frequency distribution of all words in vabulary set V and Fig.2b shows that there are around 5,500 most common words. After that we use google word2vec model(9) to achieve the pretrained word embedding weights. Finally we set maximum sentence length to 10. More specifically, we only keep first 10 tokens if the original sentence is longer than 10 and we pad the sentence with END tags if the original sentence is shorter than 10. After pre-process, our training data includes words, emotion words, like :), and emojis, such as 😋.

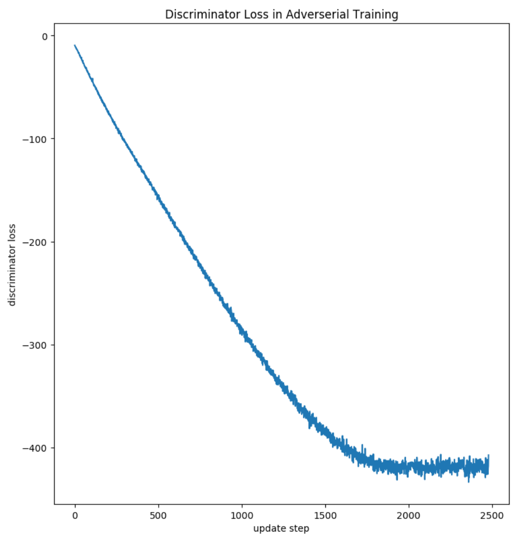


(a) Vocabulary Distribution (b) Partial Vocabulary Distribution

Figure 2: Vocabulary

5.2 Experiment Results

We implemented the learning system using TensorFlow. A quick summary of our implementation is as follows: the encoder and decoders are all made up of 2 layers of LSTM cells with hidden size 128. Discriminator is made up of 1 layer of LSTM cell and the final state of this cell is projected to a score and the discounted rate $\gamma = 0.9$. We used AdamOptimizer with learning rate 0.001, $\beta_1 = 0$ $\beta_2 = 0.9$. The sequence to sequence loss during the pretraining is stablized at around 5.1, we only listed the loss during adversarial training in figure 3 and figure 4. In addition, we will provide some generated sentences in the evaluation part.



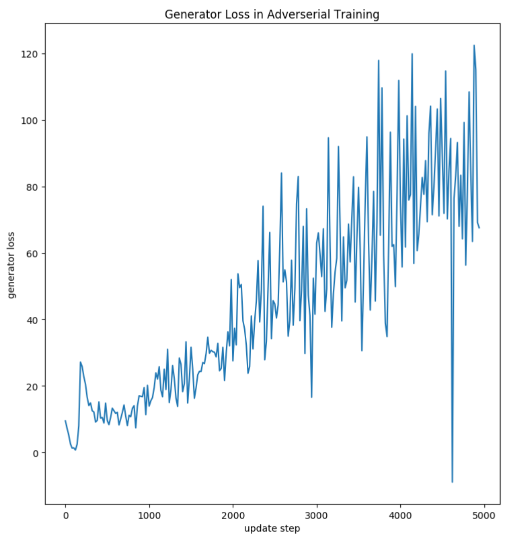


Figure 3: Loss record during training process for discriminator and generator. Discriminator loss is stabilized at -450 and generator is provided with comparatively bounded gradient

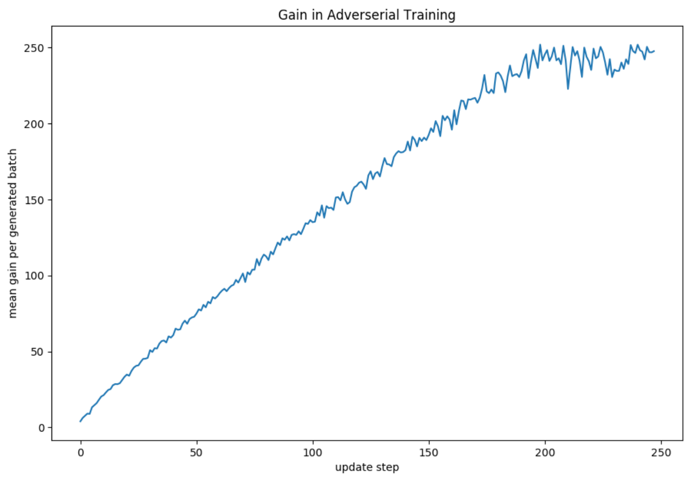


Figure 4: Gain value log obtained from adversarial learning process. Based on our method, this value should directly reflect the learning process of generator. At the end of training, gain value is stabilized around 250, which suggests generator has reached its maximum fitting capacity

6 Evaluation

We use a Seq2Seq chatbot text generation model "ChatterBot"(10) as our baseline. It is a typical retrieval-based model which can generate responses based on collections of known conversations.

6.1 BLEU

As shown in table 1, we use BLEU-1(11)(Papineni et al. in 2002) to quantitatively evaluate our result from test dataset. Using this metric, the performance of our test results is:

Chatbot Model	BLEU-1 Score
Seq2Seq	0.0812
TD-GAN	0.014

We can observe that both models have a very low score. Although BLEU is frequently used as a quantitative evaluation on generated text, it is originally designed to evaluate the quality of machine-translated text. More specifically, BLEU quantifies the correspondence between a machine's output and the ground truth. Therefore it is not able to exactly evaluate the quality of chatbot generated response. For example:

input	"How long have u been doing it for?"
ground truth	"since september."
reponse 1 (Seq2Seq)	"maybe in my class last semester lmao."
reponse 2 (TD-GAN)	"violent the sky."

We determine that response 1 is a reasonable reply and response 2 is confusing. However, both texts have bleu-1 score 0 since none word from the generated candidate is found in the reference.

6.2 Human Evaluation

Alternatively, we use human evaluation metric to quantify chatbot generated text. Nicole M. R. et al. proposed 3 quality attributes(12), efficiency, effectiveness, and satisfaction, to evaluate quality of chatbots and intelligent conversational agents. We designed 2 questions based on efficiency and effectiveness for a simple human evaluation. Then we ask three anonymous users to give scores in a 0-to-10 rating scale to all questions. The test results of both models are shown in table 2:

quality attribute	question	Seq2Seq score	TD-GAN score
EFFICIENCY	Is the response robust in grammar?	7.81	6.74
EFFECTIVENESS	Does the response follow previous topics?	4.37	4.45

7 Discussion and Analysis

Based on gain obtained by each generated batch, we could infer that the generator has converged and it has reached its maximum capacity. Most responses to short questions from TD-GAN are short but are still understandable in a sense. This suggested that our model are only capable of remembering and analyzing features within a small time window.

In addition, based on human evaluation, we find that the Seq2Seq retrieval model outperforms TD-GAN generative model in efficiency, which means Seq2Seq model has less grammatical errors and is more fluent in sentence. This makes sense because the responses of seq2seq model are selected from its collections of known conversations. On the other hand, although with more grammatical errors, TD-GAN receives higher score in effectiveness part, which means responses of TD-GAN fit more in the conversational context.

Detailed case study is provided in this section, where there are both successful response and nonsensical responses.

7.1 Case Analysis

Grounded on all demonstrated cases, our model has a small probability to generate grammatically erroneous sentences. And as discussed in the limitation, our chatbot‘s responses are not significantly relevant to questions. Also, we observed TD-GAN generated several repetitive word instead of a sequence of "END" when the sentence should be closed. This might be caused by the discriminator forcing generator to fill more words into a sentence.

7.1.1 TD-GAN is not able to learn emoji.

Input	"Jorah and Sandor used dragonglass daggers in that episode."
Ground Truth	"Sandor killed some with a big ass maul I thought."
Seq2Seq Response	for real , come by if you can 😀.
TD-GAN Response	"cold."

7.1.2 TD-GAN performs better in unseen conversation situation.

Input	"Hey. I'm a hockey player. How long have you played for?"
Ground Truth	"13 years. I played in high school and have Played In a recreational league since I got out."
Seq2Seq Response	"internet friends are the best friends."
TD-GAN Response	"hello."

7.1.3 TD-GAN suffers grammatical errors.

Input	"Ugh, don't you just hate shopping?"
Ground Truth	"Gosh I do hate shopping. Yes alcohol and video games are the best lol!"
Seq2Seq Response	"cutler is the true definition of mediocre."
TD-GAN Response	"he hate me tag."

8 Future Plan

For now, we only implemented 2 hidden layers(128 nodes in each) in the LSTM cells. But according to the network architechture in Jiwei's paper(7), there are 2048 hidden nodes in each lstm hidden layer. Also in Ilya's paper(1), in the seq2seq model, there are 4 hidden layers(1024 hidden nodes in each). Comparing to these models, we have relatively small number of hidden nodes and layers, which is one of our misunderstandings of the hyperparameters at the very beginning. In the future, our model can be modified to have more hidden layers and hidden nodes to enhance the performance.

References

[1] Ilya Sutskever, Oriol Vinyals, Quoe V. Le, Sequence to Sequence Learning with Neural Networks

[2] Martin Arjovsky, Soumith Chintala, Léon Bottou.(2017).Wasserstein GAN. arXiv:1701.07875

[3] Richard S. Sutton, Andrew G. Barto.(1998) Reinforcement Learning : An Introduction , MIT Press .

[4] Jiwei Li, Will Monroe, Alan Ritter.(2016).Deep Reinforcement Learning for Dialogue Generation. arXiv:1606.01541

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio.(2014) Generative Adversarial Networks. arXiv:1406.2661

[6] Lantao Yu, Weinan Zhang, Jun Wang, Yong Yu.(2017).SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. arXiv:1609.05473

[7] Jiwei Li, Will Monroe, Tianlin Shi.(2017).Adversarial Learning for Neural Dialogue Generation. arXiv:1701.06547

[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville.(2017) Improved Training of Wasserstein GANs. arXiv:1704.00028v2

[9] Google word2vec. https://code.google.com/archive/p/word2vec/

[10] Chatbot baseline model. https://github.com/gunthercox/ChatterBot

[11] Kishore Papineni, Salim Roukos, Todd Ward.(2002).BLEU: a Method for Automatic Evaluation of Machine Translation.

[12] Nicole Radziwill and Morgan Benton.(2017). Evaluating Quality of Chatbots and Intelligent Conversational Agents. arXiv:1704.04579