Neural Models Comparison in Natural Language Generation

Lixue Zhang, Wen Cui, Sha Tong, Ran Xu, Yifeng Liu
University of California, Santa Cruz
{lzhang75,wcui7,stong4,rxu3,yliu263}@ucsc.edu

Abstract

In the process of learning Machine Learning, we realized that natural language processing is a difficult task in Machine Learning in general due to the nature complexity of text process. Recurrent Neural Networks (RNNs) is current state-of-art model for language generation, whereas images can now be well generated by generative adversarial networks (GANs) method. Under this status quo, We would like to implement the two methods to extract the characteristics and generate language. We used the scripts of The Big Bang Theory since every character in the comedy has their own characteristics. So we expect that the two models will be able to extract their unique characteristics through training enough dataset and then generate new languages that may further generate a new episode. So far, due to the limit of time, we have trained Sheldon's lines, and RNN has achieve a good result, however, GAN model has only trained 100 iterations, we are expecting better result for a larger iteration.

1 Introduction

In this project, we explored different models for natural language generation. We firstly implemented current state-of-art model for language generation which is Recurrent Neural Networks (RNNs). In the meanwhile, we also experimented Generative Adversarial Networks (GANs) which recently achieved breakthrough in image generation. Then we compared their performance in the aspect of the quality of the text generated.

2 Related work

Karpathy in 2015 worked on generation of Shakespearean text[5]. Stanley discussed the word level and character level language models for Shakespearean sonnet generation [7]. Ofir Press has done language generation based on GANs model by training 1 Billion Word Language Model[6] . In his model, after 5000 iterations, the generation cost starts to lower.

3 Data

We will use the TV scripts from The Big Bang Theory as a dataset. We have scraped 10 seasons (220 episodes) scripts from this fan blog https://bigbangtrans.wordpress.com/. The dialogues distribution is listed in Table 1.

Character	Sheldon	Leonard	Penny	Howard	Raj	Amy	Bernadette	Rest
Dialogues	10949	9251	7276	5552	4181	3079	2437	8123

Table 1: The number of dialogues of major character in the whole dataset

Due to the limit time of the project, we only used the scripts from Sheldon in both RNNs and GANs models.

4 Methods and Results

4.1 RNN

Recurrent Neural Networks (RNNs) have achieved many success in natural language processing. It is inherently sequential, such as human speech, reading novels, magazine and papers. It makes use of sequential information. Every output is effected by the prior output. Recurrent Neural Networks are a family of neural networks designed specifically for sequential data processing. Therefore, it is suitable for text understanding, representation, and generation.

4.1.1 Implementation

We can now start fomalizing our ideas. Consider a sentence S composed by T words:

$$S = (w_1, w_2, ..., w_T)$$

At the same time, each symbol W_i is from a vocabulary V which contains all the possible words

$$V = V_1, V_2, ..., V_{|V|}$$

If we want to compute the probability of a sentence, we can use the chain rule to get

$$P(S) = p(w_1, w_2, ..., w_t) = p(w_1)p(w_2|w_1)p(w_3|w_2, w_1)...p(w_t|w_{T-1}, W_{T-2}, ..., W_1)$$

The loss for a given sentence is the negative log probability of the model assigns to the correct output

$$L(x) = -\sum_t logp_model(w_t = x_{t+1}) = -\sum_t logo_t[x_{t+1}]$$

We generated our own Big Bang Theory scripts using RNNs. There are basically four steps to generate text. Complete the iterations, which is decided by num_epochs = 150. Implement iteration output into next iteration, trained the model. Wait for the train loss converges. Implement the model to generate new transcript. The RNNs was built by implementing the following functions:

- get_inputs()
- get_init_cell(batch_size, rnn_size)
- get_embed(input_data, vocab_size, embed_dim)
- build_rnn(cell, inputs)
- build_rnn(cell, rnn_size, input_data, vocab_size)
- get_batches(int_text, batch_size, seq_length)

We have tried both word-based LSTM and char-based LSTM.

The parameters tuning for word-based LSTM are as following:
Input rnn size: 512
Layer: 2
Basic cell: basic lstm cell
embedding: trained 512 x 14837
Number of epoch: 150 completed

The parameters tuning for char-based LSTM are as following:
Total chars: 72
lstm: 2 layers, 128 input
Epoch: 10
Sequence Length: 9

Word-based RNN displays higher accuracy and lower computational cost than char-based model. With the increase of sequence length, predication of new sentence would be much more accurate for character based solution. While the training time is unaffordable. Since the importance here is to implement word-based solution, we just trained 10 iterations for char-based solution to show the general idea while it indeed failed to converge to some point with limit training epochs. But still it could show some drawbacks with char-based solution:

(1) It's hard for char-based solution to capture long-short term memory. Sentence struct depends on words expressions only, with shorter sequence length, char-based solution could follow words constructions while failed to follow sentence structure; With really longer sequence length, like 100 characters, it could take advantage of sentence structure while always failed to generate correct words since char far always from this could also contribute a lot to its generation.

(2) Time complexity of char-based solution is unaffordable. A basic training of word-based solution could be several days. But char-based solution might spend weeks on computer with CPU only.

4.1.2 Results

Word-base model results

We trained the neural network model word-based LSTM on the preprocessed data. It took two days to get a good loss.

$$
\begin{array}{llllll}
Epoch & 0 & Batch & 00/57 & trainLoss = 9.603 \\
Epoch & 1 & Batch & 43/57 & trainLoss = 1.557 \\
Epoch & 3 & Batch & 29/57 & trainLoss = 1.620 \\
Epoch & 5 & Batch & 15/57 & trainLoss = 1.528 \\
Epoch & 7 & Batch & 01/57 & trainLoss = 1.357 \\
 & & & \bullet & \\
 & & & \bullet & \\
 & & & \bullet & \\
Epoch & 143 & Batch & 49/57 & trainLoss = 0.055 \\
Epoch & 145 & Batch & 35/57 & trainLoss = 0.060 \\
Epoch & 147 & Batch & 21/57 & trainLoss = 0.050 \\
Epoch & 149 & Batch & 07/57 & trainLoss = 0.053
\end{array}
$$

Loss value converges from 100 interations.
Example of word-based output:

sheldon: i have twizzlers instead of compromise,

sheldon: oh, yes. amy's taking this conversation from

sheldon: oh it's no. i see that were up

sheldon: oh, yes. i'm in tea with support.

sheldon: oh, i wake.

sheldon: because carefully came i'd have ever as it's

sheldon: oh, let me actually, i thought he would

sheldon: it's a negative. the day parallel out from the office.

sheldon: really? okay?

sheldon: leonard, i

sheldon: hmm. i thought we've made up in a suspicion suspension account.

sheldon: sorry?

sheldon: why many of last man, i have

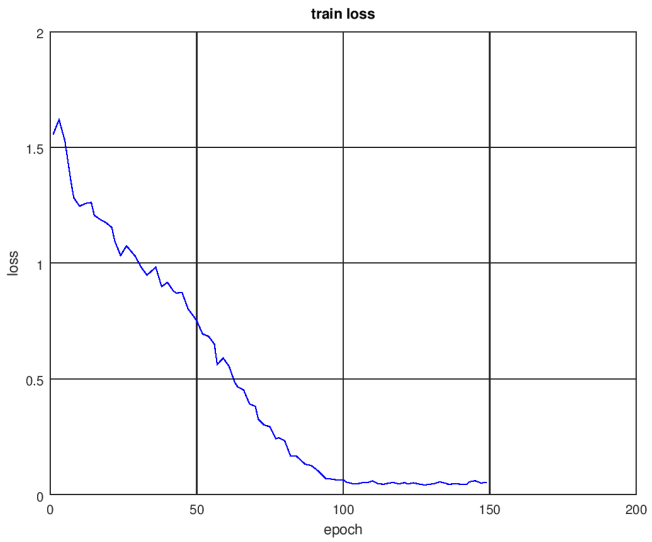


Figure 1: Loss-Epoch

sheldon: it has been, based on his nose

sheldon: do you have those well i would.

sheldon: amy

sheldon: oh, listen to them. now howard

sheldon: oh! out of a course of ...

Example of char-based output:

sheldon: hello, the tinet an accidence that substage of start the scientist? stay.

sheldon: why not entitled new of science.

sheldon: i don't take you hear it.

sheldon: i can't be an astroied the what sastent the startic new night.

sheldon: oh, there's in the on start the exploto becusely something the she a think the in a tending start th

sheldon: it's an explodicted hat to space to be more part to go back to the pretty.

sheldon: i'm sorry, if you didn't want to pretent of your conite science for me point sure i be leonard of th...

4.2 Wasserstein GAN

Generative adversarial networks (GANs) [3] have achieved great success at generating realistic and sharp looking images. However, they still remain remarkably difficult to train a traditional GANs. [1] Wasserstein GAN [2] however utilizes a different objective function with nicer properties.

4.2.1 Different Distance

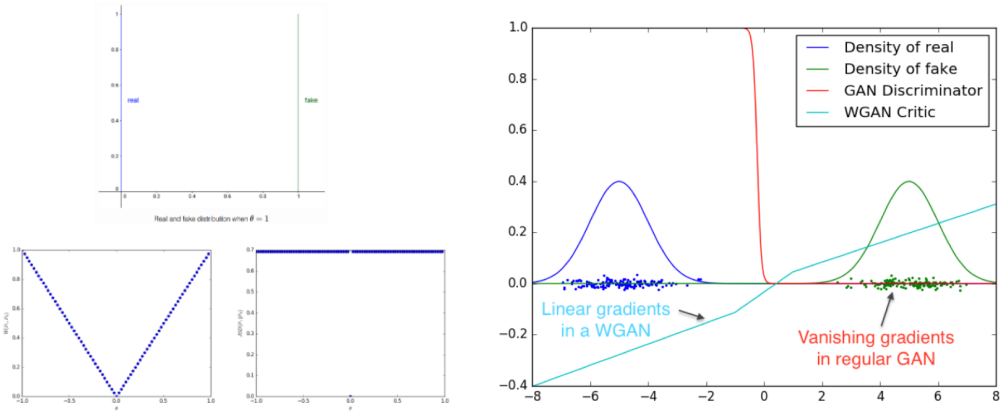
The original GANs are trained to minimize the distance between the generated and true data distributions, denote as $\mathbb{P}_g, \mathbb{P}_r$ respectively. Jensen-Shannon divergence was used as this distance metric in Equation 1,

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \parallel \mathbb{P}_m) + KL(\mathbb{P}_g \parallel \mathbb{P}_m) \tag{1}$$

where $\mathbb{P}_m$ is the mixture $(\mathbb{P}_r + \mathbb{P}_g)/2$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$. Whereas the WGAN utilizes the Earth-Mover (EM) distance or Wasserstein-1 as its distance metric shown in Equation 2.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{2}$$

Learning parallel lines. Let $Z \sim U[0, 1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$(a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta = (\theta, z)$ with θ a single real parameter. The different loss can be seen in Figure 2. And The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.



(a) Learning parallel lines by the EM distance(left) or the JS divergence(right)

(b) Learning Gaussian distribution by the EM distance(left) or the JS divergence(right).

Figure 2: JS divergence and EM distance comparison

4.2.2 Training WGAN

We use off-the-shelf tool [6] and modify to fit in our dataset. Language is learned at character level. Both Discriminator (D) and Generator (G) are using one layer GRU cells with hyperparameters listed in Table 2.
The training loss of G and D are shown in Appendix A. As we can see training of sequence length of 1, the model converges quickly and gets to its optimal after 100 iterations. Whereas the other pictures don't show clear evidence that the model gets to optimal. Which essentially tells us we should increase the iterations per sequence. However this is not applicable due the long training time. The training ran on a local machine with specifications listed below.

```
RAM: 32G DDR3, CPU: i7-3770k 4 cores 8 threads, Graphic card: AMD Radeon HD7700
```

The training time is therefore plotted in Figure 3. As we can see the training time grows exponentially. And for this experiment setup the model was running for 7 days to finish the training.

Hyperparameters	Values
BATCH SIZE	64
SEQUENCE LENGTH	79
GENERATOR ITERATIONS	50
ITERATIONS PER SEQUENCE LENGTH	100
G STATE SIZE	512
D STATE SIZE	512

Table 2: Experiment setup

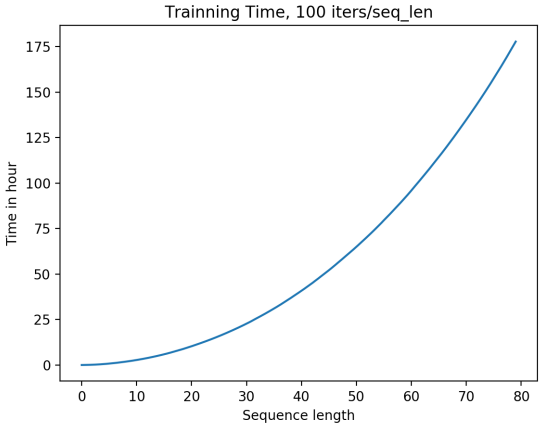


Figure 3: Training time of WGAN

4.2.3 Language Generation

After training WGAN model for 100 iterations for 79 sequence length, we can use saved model to generate text and a sample output of 79 sequence length is listed below.

```
You know un she coldienss to don't slick. And right the perpoct a seeppacterfic
And root has snided in interppctuse is interpacs in inting to don't slick. And r
Yeard, Leonarderferppict a seeppactly coome right the perpoct a seeppacterfic sh
Wh, Leonarderferppict a seeppactly coome right the perpoct a seeppacterfic she c
I seed to paction s in in in in in interppctuse in inting the perpopterfact.
```

5 Comparison and Evaluation

5.1 Comparison

Based on our result word-based RNN, char-based RNN, as well as char-based WGAN models, we can see that WGAN takes much more time for training than RNN model. And also char-based model will give us more spell errors than word-based model.

5.2 Evaluation

A collection of functions that measure the readability of a given body of text. Most of the metrics estimate the grade level required to comprehend a given block of text. Here is a list of common readability metrics: ARI, FleschReadingEase, FleschKincaidGradeLevel, GunningFogIndex, SMOGIndex, ColemanLiauIndex. [4] These metrics are related to the number of complex words, the number of syllables in sentences an so on. Here we use NTLK python library to write code for these metrics. The final result is an average grade level based on several metrics outputs. For char-based RNN training, the generation result is approximately equal to a reading level of 8th grade U.S. education. For word-based RNN training, the result is about a 5th grade reading level. For char-based GAN training, the reading level needed is 13th grade. All these outputs are in the normal range of readability indexes. The evaluation test is a standard measurement to prove that these generations give readable results, which is fairly good.

6 Discussion

LSTM is current state-of-the-art approach for language generation, it is efficient compared to GAN in terms of running time. We also get a good result using RNN model. GAN has achieved state-of-the-art results in image generation. We expect that GAN performs better because when training the data, generate cost is taken into account, however, GAN computational cost is too high. Because of the

time limits, we only trained GAN model for 100 iterations for 79 sequence length, we can see that the results are already reasonable. From figure 3, we can see that the training time increase exponentially respect to the linear increase of sequence length. If we can train the GAN model with more iterations with the data for a longer sequence length, we are expecting to see better results of GAN.

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

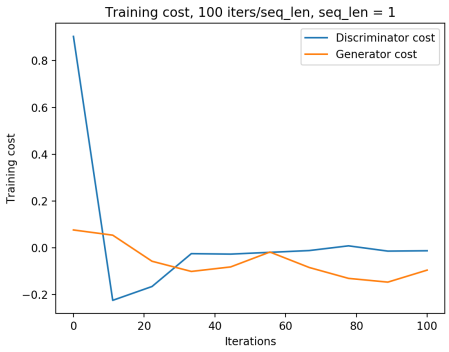
[4] Thomas Jakobsen and Thomas Skardal. Readability index. *Agder University*, 2007.

[5] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy blog*, 2015.

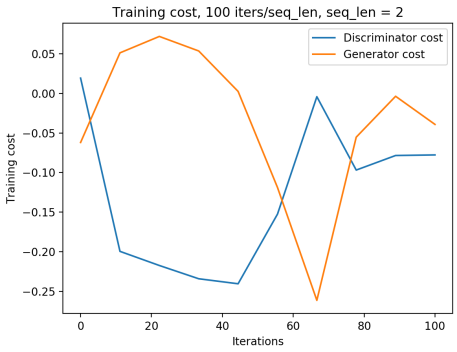
[6] Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. Language generation with recurrent generative adversarial networks without pre-training. *arXiv preprint arXiv:1706.01399*, 2017.

[7] Stanley Xie, Ruchir Rastogi, and Max Chang. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation.

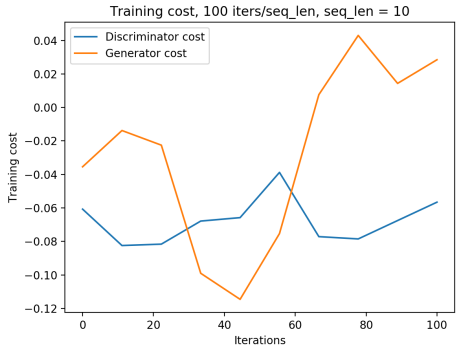
Appendix A Training loss of WGAN



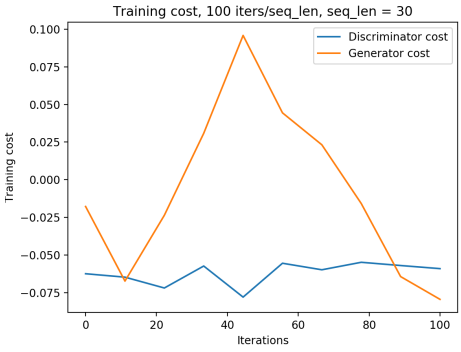
(a) Training Loss of sequence length = 1



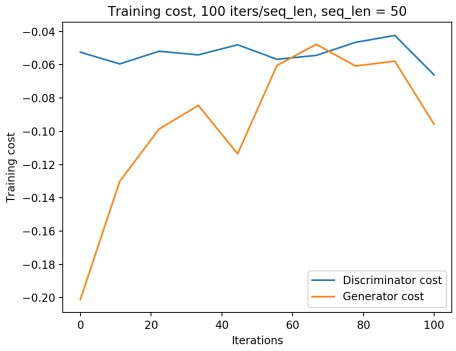
(b) Training Loss of sequence length = 2



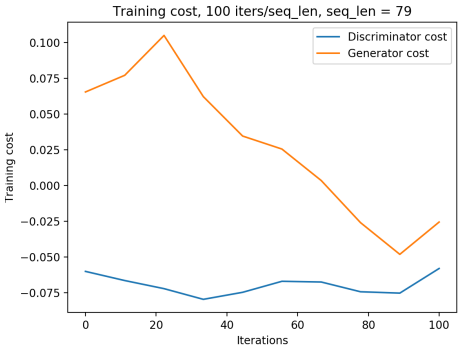
(c) Training Loss of sequence length = 10



(d) Training Loss of sequence length = 30



(e) Training Loss of sequence length = 50



(f) Training Loss of sequence length = 79