Twin-GAN for Neural Machine Translation

Jiaxu Zhao[1], Li Huang[1], Ruixuan Sun[2], Liao Bing[3] and Hong Qu[1]
[1]*University of Electronic Science and Technology of China, Chengdu 610054, China*
[2]*Yelp Inc., U.S.A.*
[2]*Chengdu Dajiangtong Technology Co., Ltd, China*

Keywords: Neural Machine Translation, Exposure Bias, GAN.

Abstract: In recent years, Neural Machine Translation (NMT) has achieved great success, but we can not ignore two important problems. One is the exposure bias caused by the different strategies between training and inference, and the other is that the NMT model generates the best candidate word for the current step yet a bad element of the whole sentence. The popular methods to solve these two problems are *Schedule Sampling* and Generative Adversarial Networks (GANs) respectively, and both achieved some success. In this paper, we proposed a more precise approach called "similarity selection" combining a new GAN structure called twin-GAN to solve the above two problems. There are two generators and two discriminators in the twin-GAN. One generator uses the "similarity selection" and the other one uses the same way as inference (simulate the inference process). One discriminator guides generators at the sentence level, and the other discriminator forces these two generators to have similar distributions. Moreover, we performed a lot of experiments on the IWSLT 2014 German$\rightarrow$English (De$\rightarrow$En) and the WMT'17 Chinese$\rightarrow$English (Zh$\rightarrow$En) and the result shows that we improved the performance compared to some other strong baseline models which based on recurrent architecture.

1 INTRODUCTION

Thanks to recent advances in deep learning, many approaches have emerged to implement Neural Machine Translation (NMT). Because the translation task is a sequential problem, most of those approaches are based on Seq2Seq models (Kalchbrenner and Blunsom, 2013) (Sutskever et al., 2014) (Cho et al., 2014). The Seq2Seq models generally consist of two sub neural networks called encoder and decoder. The encoder is to extract a representation of the input source sequence; the decoder is to transfer the representation to the target sequence. Generally, the decoder's input is a target-side sequence during training (teacher forcing mode) and the input is generated by itself during the inference phase (recurrent mode). This strategy brings a discrepancy called *exposure bias* (Ranzato et al., 2015) causing a gap between the training and the inference. That is, the words accepted by the decoder in training is the ground-truth sentence, but in testing, it accepts the output of its previous unit as the input of this unit. The inconsistency between these two settings will lead to error accumulation at test time. *Schedule Sampling* (Bengio et al., 2015) is one of the most common solutions to this problem. This method is only used in the training and it works like this: in the early stages of training, the method mainly uses the target-side sequence as the decoder's input, which can quickly guide the model from a randomly initialized state to a reasonable state. As the training progresses, the method will gradually use the generated words as the decoder's input. The probability of using the generated word as input in *Schedule Sampling* changes with the training time, which is reasonable. But we think that the probability should change with the performance of the model, that is, the better the model performs, the higher the probability should be. In this paper, we use the similarity of generated words and labels to indicate the performance of the model, which we call "similarity selection". Like *Schedule Sampling*, it still does not completely solve the problem of inconsistent data distribution during training and generation (Huszár, 2015). Inspired by (Lamb et al., 2016), we introduced Generative Adversarial Networks (GANs)(Goodfellow et al., 2014) (Goodfellow, 2016) in combination with "similarity selection" to solve the problem. GAN is a deep learning model and one of the most popular methods for unsupervised learning on complex distributions in recent years. It consists of two modules (Gen-

Zhao, J., Huang, L., Sun, R., Bing, L. and Qu, H.
Twin-GAN for Neural Machine Translation.
DOI: 10.5220/0010217300870096
In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021) - Volume 2*, pages 87-96
ISBN: 978-989-758-484-8

erative Network and Discriminative Network) and it produces fairly good output through the mutual game learning of the two modules. Since GAN has achieved remarkable success in image generation since it was proposed, some works (Lamb et al., 2016) (Zhang et al., 2016) introduced GAN to NLP. In this paper, we design two generators, one using "similarity selection" to generate sequences, the other using recurrent mode to generate sequences. And we use a discriminator to make the latter generator close to the forward one to solve the problem of inconsistent data distribution.

Most of the NMT models are learned only under the guidance of the ground-truth translations. Models are generally optimized by maximizing the likelihood estimation of the target-side sequence at each step at training time. This optimization strategy just guarantees that the model can generate the best word at every single step but ignore the coherence and naturalness of the whole sentence. To address this issue, there are some works (Shen et al., 2015) (Bahdanau et al., 2016) trying to optimize the model in sentence level not just in word level. Those works aim to avoid the limitation of maximum likelihood estimation by directly optimizing the evaluation (BLEU) (Papineni et al., 2002) and the results show that they have improved the quality of translation a little. Some researchers also use the idea of GAN to solve the problem: the discriminator learns to distinguish whether a given sentence is a ground-truth sentence or not, which can be regarded as a guide of the generator in sentence level. The generator can be the ordinary neural machine translation networks, which learns by generating sentences more similar to ground-truth translations. However, applying the traditional GANs directly to sequence generation like neural machine translation is not appropriate. Traditional GANs generating an image is a real-valued data mapping transformation, which is a continuous process. Therefore, the discriminator's gradient can be propagated back to the generator. But for NMT, in the process of decoding and generating text, the model generation is actually the process of selecting words in the vocabulary, which is a discrete and non-differential process (Huszár, 2015). Therefore, the gradient of the discriminator can not be propagated back to the generator, and the parameters of generator can not be updated (Goodfellow, 2016). To address the issue and guarantee those two sub-modules in GANs are effectively optimized when the data is discrete, (Kusner and Hernández-Lobato, 2016) proposed the Gumbel-softmax distribution, which is a continuous approximation to a polynomial distribution based on the $softmax$ function. Inspired by (Yang et al., 2017) (Wu et al., 2017), in this work, we use the idea of the policy gradient method (Williams, 1992), widely used in reinforcement learning (Sutton et al., 1998), to make the gradient of the discriminator can be propagated back to the generator. Therefore, we propose another discriminator that identifies the coherence and naturalness of the generated sentences and feeds the value back to the generator using the recurrent mode through the policy gradient.

Our work provides the following contributions:

- We propose a novel generative adversarial network model for NMT. As our best knowledge, this is the first work that leverages two generators and two discriminators in GAN. One generator generates sentences with "similarity selection" and the other generates with ordinary recurrent mode as in the test. One discriminator closes the distance between the two generators, the other guides the generator to generate more natural and fluent sentences in sentence level.
- We design a more precise method named "similarity selection" as opposed to the "Scheduled Sampling". We believe that selecting a word as input should depend on the quality of the generated data rather than the training time.
- We carried out extensive experiments on German→English and Chinese→English translation tasks. We compared some strong basic recurrent structure models with twin-GAN and the result of the twin-GAN significantly outperforms the RNNsearch (Bahdanau et al., 2015) by +4.20 BLEU points on IWSLT 2014 De-En data set. Comparing with the related model, our model further achieved a significant improvement by about +1.09 BLEU points.

2 RELATED WORK

2.1 Neural Machine Translation

Neural Machine Translation (NMT) is a Machine Translation method that applies neural networks to predict the likelihood of a sequence of words. Since Sequence to Sequence (Seq2Seq) architecture (Kalchbrenner and Blunsom, 2013) (Sutskever et al., 2014) (Cho et al., 2014) was proposed, the most popular structures of NMT are based on it, such as (Wu et al., 2016) (Tan et al., 2019) (Artetxe et al., 2017) (Lample et al., 2017) (Lample et al., 2018) (He et al., 2018) (Song et al., 2018). The architecture of the Seq2Seq comprises of two processes, the encoder process and decoder process. And these two parts are very flexible. Researchers started with RNN in these two processes and later gradually introduced Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) and Gate Recurrent Unit (GRU) (Cho et al., 2014) to in place of RNN units. Convolutional Neural Network (CNN) is the most commonly used neural network for analyzing visual images. As witnessed the great success of CNN in computer vision, some researchers introduced it to NLP. And some works (Lamb and Xie, 2016) (Kalchbrenner et al., 2016)(Gehring et al., 2017) used CNN for machine translation with success. The two processes in Seq2Seq are relatively independent and not closely related. To this end, (Luong et al., 2015) proposed the attention mechanism, which greatly improved the performance of Seq2Seq through making these two parts more closely. (Vaswani et al., 2017) proposed a fully-connected attention-based model called transformer, which discards the CNN and RNN. And transformer became popular in natural language processing applications, notably machine translation and language modeling. Since the Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) was proposed, it was widely used in unsupervised learning in the field of computer vision and had a good performance. Therefore, some researchers (Yu et al., 2017) (Lamb et al., 2016) (Goodfellow, 2016) (Zhang et al., 2016) introduced GAN into NLP. For NMT, (Yang et al., 2017) and (Wu et al., 2017) achieved competitive performance through GAN.

2.2 Exposure Bias

As described above, the Seq2Seq model (Sutskever et al., 2014) has been popular in Neural Machine Translation recently. The decoder of Seq2Seq requires the target sequence as it's input at training time, but the output of the previous time step is used to instead at test time. The difference between training and inference causes the *exposure bias* (Ranzato et al., 2015). (Venkatraman et al., 2015) proposed a method to alleviate the impact of errors in the current time step on posterior steps by reusing the training data which serves as a "demonstrator" by providing corrections for the errors. Based on this method, (Bengio et al., 2015) proposed another one called "Schedule Sampling". That is, the input used in the training phase uses the ground-truth with a probability of p, and uses the output of the previous word with a probability of 1-p. And this probability p increases as training progress. And (Zhang et al., 2019) has a more precise formula to get the probability p. Unlike those works that use a simple function to calculate a probability, we use the similarity of label and output to guide the probability p.

Another direction of trying to solve *exposure bias* is to borrow the idea of GANs, like (Lamb et al., 2016), (Yang et al., 2017) and (Wu et al., 2017). However, they still use "teacher forcing" mode (Williams and Zipser, 1989) in the generator at training time. Unlike them, we use the "similarity selection" to replace the "teacher forcing" mode. Furthermore, We also have another generator, which generates sentences in the same way as in the test. And we use a discriminator to force the distribution of the latter generator to be close to the previous one. In the final test, we only use the latter generator, whose data distribution is the same for testing and training.

2.3 Sentence Level Strategy

To make the generated sequence closest to the ground-truth sequence, maximizing the likelihood estimation (MLE) of the ground-truth sequence at each time step is the most commonly used optimization method. This method optimizes model in word level because that MLE forces model to generate a word that corresponded to the target word at the current step. In this work, we use a discriminator that is similar to (Yang et al., 2017) and (Wu et al., 2017), which optimizes generator in sentence level by discriminating the sentence is generated by machine or written by humans. Therefore, in order to "cheat" the discriminator, the generator will generate more natural and coherent sentences like those written by humans. But there is a limitation when generating sequences of discrete tokens because it's difficult to pass the gradient update from the discriminator to the generator. Same as (Yu et al., 2017), we borrowed a idea of reinforcement learning called policy gradient to make it differentiable in this discrete process. We use the result of the discriminator as a reward and feed it back to the generator.

The remainder of this paper is organized as follows. Section 3 provides the background of Neural Machine Translation, including its commonly used architecture and the application of GAN in NMT. Our proposed twin-GAN is introduced in Section 4. We discuss our experiments in Section 5, including data sets, specific model settings and results analysis. Section 6 concludes the paper.

3 BACKGROUND

3.1 Encoder-Decoder with Attention

The translation task is to transfer one sequence into another sequence. In Neural Machine Translation, Encoder-Decoder is the most popular and empirically effective structure. Encoder and decoder can be any model like RNN based and CNN based. Assume the source sequence and target sequence are X = $\{x_1,x_2, \dots ,x_n\}$ and Y = $\{y_1,y_2, \dots ,y_m\}$ respectively. Note the E = $\{e_1,e_2, \dots ,e_n\}$ is employed to represent the embedding matrix of the source sentence X. When y_i,i$\in$ 1,2...,m is generated, the model assigns attention to different words in source sequence for the i-th decode step. According to different attention, E is encoded as intermediate context representation c_i,i$\in$ 1,2...,m that varies based on the current word generated. And then input the combination of context vector c_i and previous output into the decoder. Finally, the decoder outputs y_i,i$\in$ 1,2...,m.

3.2 Generative Adversarial Network in NMT

(Goodfellow et al., 2014) proposed Generative Adversarial Networks (GANs) and it got great success in computer vision. The GANs is a new framework that optimizes the generative model through the adversarial process. The idea of GANs is a two-player game that is played between a generative model and a discriminative model. The training strategies of its two sub-models are as follows: the training process of the generator G is to maximize the probability of the discriminator making mistakes, that is, the discriminator mistakenly believes that the data are real samples rather than the fake samples generated by the generator. The training process of the discriminator D is to maximize the probability of making a correct judgment on the input data. Generally, the discriminator is a classification model or scoring model like multilayer perceptron (MLP). GANs are first applied to generate realistic pictures with random noise. It works well in generating images since the image is continuous data, and small changes can be reflected on the pixels. But if the generated data is based on discrete tokens, the information given by D is often meaningless. Small changes to tokens are invalid due to there may not be corresponding tokens in the vocabulary space.

In neural machine translation, sentences are composed of a series of discrete tokens. Because the generator G needs to be trained using the gradient obtained from the discriminator D, and both G and D need to be completely differentiable. Problems arise when there are discrete variables, where Back Propagation (BP) (Rumelhart et al., 1986) cannot provide the training gradient form D for G. There are two main approaches to solve this problem, Gumbel-softmax (Kusner and Hernández-Lobato, 2016) and policy gradient (PG) method (Williams, 1992). In this paper, we use the latter which is a common optimization method in reinforcement learning. Different from original back-propagation, PG does not back propagate by error but directly carries out back propagation by selecting a behavior through observation information. It does not have errors, but uses rewards to directly enhance and diminish the likelihood of a chosen action, Therefore, good actions increase the probability of being selected next time, while bad actions decrease the probability of being selected next time.

(Wu et al., 2017) is a recent work that applied GANs and the idea of policy gradient to NMT, which replaces the previous method. It improves the translation quality of model by minimizing the difference between human translation and the translation given by the NMT model. This work proposed a Convolutional Neural Network that can accurately capture high-level abstract correspondence of concatenation of source sequence and target sequence. First, they join the output sequence $\tilde{Y}$ of the NMT model or the human translation output sequence Y with the source language sentence X to form a two-dimensional matrix and then measure the similarity between Y (or $\tilde{Y}$) and X through the convolution neural network. The output label is 0(irrelevant) or 1(relevant). Specifically, they join the word embedding corresponding to the i-th word x_i in the source sequence and the j-th word y_j in the target sequence follow $z_{i,j} = [x_i^T, y_j^T]^T$ to gain a three-dimensional tensor representation Z. Then through the convolution layers and max-pooling layers to get more abstract feature expressions. The obtained feature expressions are then fed to a multilayer perceptron, and the *sigmoid* activation function of the last layer outputs a 2-class probability distribution. And generative network update parameters by the policy gradient. The loss of the discriminative network is used as a reward to the generative network, which makes the gradient of discriminator can propagate to the generator.

4 MODEL

Part of our model is based on (Wu et al., 2017), and we have another generator and discriminator in our model. We use the “similarity selection” we designed to train *generator*$_1$ and use the policy gradient to train *generator*$_2$. Two discriminators are called "sentence discriminator" and "hidden state discriminator", which are represented by D_s and D_h respectively. D_s is used to distinguishing whether the sentence is generated by our generators or written by human. And its result is used as a reward to *generator*$_2$; and D_h is used to reduce the difference between *generator*$_1$ and *generator*$_2$. The framework of the model is shown in the Figure.1. We introduce our model in three parts:(1) "similarity selection", (2) Generators and (3) Discriminators.

4.1 Similarity Selection

The Encoder-Decoder framework has achieved great success and has been widely used in many natural language processing tasks. And Recurrent Neural Network (Mikolov et al., 2010) is the most commonly used model in the two sub-modules of Encoder-Decoder when it comes to machine translation task. In the structure of RNN, dealing with the timing problem like sentences is to recurrent input and output one by one. Assume X = $\{x_0, x_1, \ldots, x_n\}$ are the input units and Y = $\{y_0, y_1, \ldots, y_m\}$ is the label. RNNs also contain hidden units that marked as $\{s_0, s_1, \ldots, s_n\}$ and express them as vector form S. s_t is the state at the t-th time step of the hidden layer, which is the memory unit of the network. s_t is calculated based on the current input x_t and the previous state of the hidden layer s_{t-1}.

$$s_t = f(U * x_t + W * s_{t-1}) \tag{1}$$

in which f is a non-linear activation function like ReLU. U and W are model's weights. When calculating s_0, that is, the hidden state of the first word, s_0 can be 0 vector. The output process of RNNs as blew:

$$o_t = V * s_t \tag{2}$$

where o_t is the output at time step t, that is, the vector representation of the next word. And V is the learnable weights. In the decoding process, these hidden units complete the most important work but the way to get s_t will cause *exposure bias* (Ranzato et al., 2015). To address it, we proposed a new and effective method called "similarity selection". As the name suggests, it uses the previous output or the label as input based on the similarity between the output and the label. When the output of the previous time step and label have high similarity, we use the output of previous step as the current input, otherwise, we use the current label to input. Inspired by (Li and Han, 2013), (Muflikhah and Baharudin, 2009) and (Nguyen and Bai, 2010), the similarity of these two vectors is calculated by Eq.3:

$$\begin{aligned} similarity(A,B) &= \frac{A \cdot B}{\| A \| \| B \|} \\ &= \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \end{aligned} \tag{3}$$

where A and B represent two word embeddings, and n is the dimension of the word embedding. The similarity of the two vectors ranges from 1 to -1, and the angle between them increases as the similarity decreases. The smaller the value is from 1, the closer these two vectors are. When the similarity is 0, the two vectors are vertical. When the similarity is less than 0, it means that the vectors are more dissimilar. So we treat the vectors with a similarity of 0 to -1 as the vectors with a similarity of 0. In this work, We first calculate the similarity of the output of the previous time step and the label of the current time step. Then we input the similarity into ReLU function and get a result from 0 to 1. And then we take this result as the probability of using the output instead of the label. It means that the more similar these two word vectors are, the more likely we are to use the output from the previous step to input.

4.2 Generators

We have two generators named *generator*$_1$ and *generator*$_2$ in our work. *generator*$_1$ uses "similarity selection" and *generator*$_2$ uses policy gradient. These two generators are both based on the encoder-decoder framework and share parameters. We use Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) in encoder and decoder:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5}$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{6}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{8}$$

$$h_t = o_t * tanh(C_t) \tag{9}$$

where W_i, W_f, W_c, W_o and b_i, b_f, b_c, b_o are matrices of learnable weights and biases. Eq.4 is input gate; Eq.5 and Eq.6 is forget gate; Eq.7 is output gate; Eq.8 and Eq.9 are represent long term memory and short term memory respectively. h_t is the hidden state of the network at time step t and x_t is the input unit at the current step t. Different from other works, here x_t is the current label or the previous step output selected by the "similarity selection" in *generator*$_1$. And we

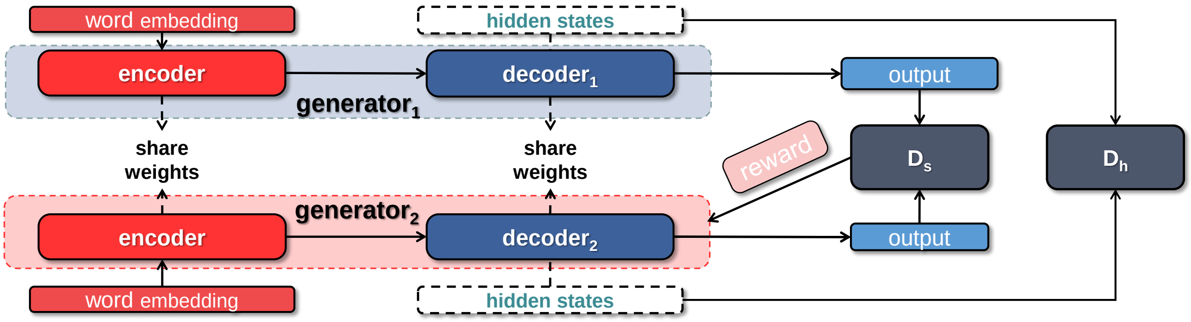

Figure 1: Model framework: The top of the framework is $generator_1$, and the bottom is $generator_2$. The D_h represents the "hidden state discriminator", whose input is the hidden states of the generator's decoder. The D_s represents the "sentence discriminator". The generator's outputs are input to D_s and its output is fed back to $generator_2$ as a reward.

also add attention mechanism (Bahdanau et al., 2015) into the connection between encoder and decoder:

$$c_t = \sum_{i=1}^{|X|} \alpha_{it} h_i \tag{10}$$

$$\alpha_{it} = \frac{exp(F(s_i, h_{t-1}))}{\sum_{j=1}^{|X|} exp(F(s_i, h_{t-1}))} \tag{11}$$

$$F(s_i, h_{t-1}) = h_{t-1} \cdot s_i \tag{12}$$

where |X| represents the length of the source sentence. c_t is the result of the attention mechanism for decoder at time step t. h_i and s_i represent the i-th hidden state of encoder and decoder respectively. We use Eq.12 (dot-multiply) to calculate the attention score. Eq.11 is the $softmax$ function and it obtains α_{it}, which is the final attention score of the encoder hidden states for the t-th time step decoder. The iterative optimization scheme of $generator_1$ is:

$$minimizing : -\sum_{j=1}^{|Y|} logP_j[y_j] \tag{13}$$

where | Y | indicates the length of the ground-truth sentence, P_j refers to the predicted probability distribution at the j-th step, y_j is the token generated by $generator_1$ at the j-th step.

In $generator_2$, we train it in the same way as the inference (recurrent mode), that is, we use the previous step output as the current input to train it recurrently. This strategy during training makes it more like an inference process to achieve the purpose of alleviating the discrepancy between training and inference. We apply the idea of policy gradient to $generator_2$. We use the result of the D_s as the reward to this generator. With the notations for $generator_2$ model $G(\theta_2)$ and sentence discriminator model D_s, the iterative optimization scheme of $generator_2$ is:

$$minimizing : -E_{X \sim P_{data}(X), \tilde{Y} \sim G(\theta_2, X)} log(1 - D_s(X, \tilde{Y})) \tag{14}$$

where X is the data sampled from source sentences, θ_2 is parameters of $generator_2$ and $\tilde{Y}$ is the target sentences generated by $generator_2$.

We use a discriminator D_h (introduced in the Section 4.3) to force $generator_2$ to close $generator_1$, and finally make their data distribution close. At the end, we use $generator_2$ to test.

4.3 Discriminators

In this paper, we proposed two discriminators in one GAN innovatively. And one is to discriminate between the results of the generators and ground-truth translations, and the other is to discriminate against the data distributions of these two generators.

One discriminative network, called D_s for short, is inspired by (Wu et al., 2017). We use it to directly reduce the difference between sentences generated by generators and written by human to train these two generators at the same time. And we use the results of D_s as a reward feed to $generator_2$. Therefore, $generator_2$ is updated by its own error and the reward received from D_s. We first concatenate the source sentence (X) and synthetic sentence ($\tilde{Y}$) generated by our generators or the target ground-truth sentence (Y) to form a matrix $\tilde{Z}$ or Z, which also helps to measure the translative matching degree of source sentence and target sentence. Then the $\tilde{Z}$ and Z are fed to D_s and outputs a probability that $\tilde{Z}$ and Z is from ground-truth data. The structure of D_s is based on CNNs which ignores some overly detailed features of input sequences to capture the overall features and the relevance of these two sequences.

For the other discriminator, called D_h for short, it is used to make the model distributions of these two generators more similar through discriminating the hidden states of them. It is based on a bidirectional recurrent neural network (Bi-RNN) that consists of two RNNs. One runs forward in time on the input hidden states sequence, and the other runs backward in time with the same input sequence. The hidden states of the two RNNs are connected at each time step to form a new hidden state which is continuing to feed the next layers. The output of Bi-RNN is fed to the Multilayer Perceptron (MLP) and the output layer consists of an affine transformation and a *sigmoid*. Finally, D_h outputs a result indicating the probability that the input hidden state sequence is the hidden states of $generator_1$. We re-train the D_h as:

$$\begin{aligned} minimizing: & -E_{X\sim P_{data}(X),h_1\sim G(\theta_1,X)} log(D_h(h_1)) \\ & -E_{X\sim P_{data}(X),h_2\sim G(\theta_2,X)} log(1-D_h(h_2)) \end{aligned} \tag{15}$$

in which h_1 and h_2 are the hidden states of $generator_1$ and $generator_2$ respectively.

5 EXPERIMENTS

5.1 Dataset

We carry out experiments on the IWSLT 2014 (Cettolo et al., 2014) German→English (De→En) and the WMT'17 Chinese→English (Zh→En) respectively.

IWSLT 2014: This data set is bilingual corpus pairs of German and English, consisting of a training set, a validation set and a test set, with 153k, 7k and 6.5k pairs of sentences respectively. And we set the maximum sentence length to 50. Our dictionary keeps the first 22,822 high-frequency English words and 32,009 German words, and the remaining words not in the vocabulary are replaced by "<UNK>".

WMT'17: In this dataset, in order to obtain Chinese word segmentation, we use jieba[1] to process Chinese corpus. We use the *newsdev2017* as the dev set and *newstest2017* as the test set. And we use byte pair encoding (BPE) (Sennrich et al., 2015) approach to preprocess the Chinese and English corpus, generating Chinese and English vocabularies with 37,516 and 25,524 types, respectively.

5.2 System

For the whole model, the dimension of the word embedding is set as 512; We initialized the learning rate of both discriminators and generators at 0.001. Since discriminators are simpler and easier to train than generators, we trained them at different frequencies. We trained discriminators only once every ten times the generators are trained. The source code is available at GitHub[2].

Generators: both two generators share the same encoder-decoder structure as described in Section 4.2, and these two generators share weights. The details of generators are as follow: 2-layers LSTM networks act as encoder and decoder employ 2-layers LSTM and an attention. We set the hidden units for both encoders and decoders as 1000. We initialized the parameters of LSTM networks following the Uniform distribution $\mathcal{U}$(-1, 1). The attention mechanism we used in the decoder is the same as (Bahdanau et al., 2015). And we also applied dropout(=0.3) for training the generators. During testing, we used a beam search with a beam size of 5 and the length penalty is not applied. In the forward process, we used the "similarity selection" method and the inference way in $generator_1$ and $generator_2$ respectively to train them. And we set fifty percent probability to train $generator_1$, and another fifty percent probability to train $generator_2$.

Discriminators: for "sentence discriminator" (D_s), we used two CNN layers and each layer is followed by a max-pooling layer, with 3×3 convolution kernel size and 2×2 pooling window size. And then fed the output of the CNN layer to a MLP network which has two full connection layers and each layer with a ReLU activation function. Finally, we mapped it into a score through a *sigmoid* function. For hidden state discriminator(D_h), it is based on a bidirectional recurrent neural network, which consists of two RNNs (two GRU networks (Cho et al., 2014)), The hidden states of these two RNNs are fed to an MLP network. That MLP has three layers, each consist of an affine transformation and a ReLU activation function. Finally, the output layer comprises an affine transformation and a *sigmoid*.

5.3 Results

The translation results of our experiments are evaluated by BLEU (Papineni et al., 2002) score[3]. As shown in table Table 1 which compares the results of our twin-GAN model with some strong basic models on the IWSLT 2014 De→En data set. (Bahdanau et al., 2016) uses an approach to train model to generate sequences through using actor-critic approach from reinforcement learning (RL) to replace the universal log-likelihood training approach. (Bahdanau et al., 2015) allows a basic encoder–decoder model to

[1]https://github.com/fxsjy/jieba

[2]https://github.com/zhao1402072392/twin-GAN

[3]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

automatically search for parts of a source sequence related to the predicted target word to improve the performance of it. And we carried out experiments with our twin-GAN model at different learning rates and the different ratios of times of training generators and discriminators. And the result shows that this BLUE score of this work (with learning rate=1e-3, update G/10 times update D) on the IWSLT 2014 De$\rightarrow$En data set is 1.09 points higher than that of the related model (Wu et al., 2017).

Table 1: Comparisons of different competitive NMT systems on IWSLT 2014 De$\rightarrow$En data set.

Systems		BLEU
CNN + actor-critic		22.45[a]
RNN Search		23.87[b]
Adversarial NMT		26.98[c]
twin-GAN	update/5 times, lr=1e-3	27.73
	update/10 times, lr= 5e-4	27.81
	update/10 times lr=1e-3	**28.07**[†]

[a] reported in (Bahdanau et al., 2016)
[b] reported in (Wiseman and Rush, 2016)
[c] reported in (Wu et al., 2017)
[†] the best performance of our twin-GAN model

We also train twin-GAN with other strategies in *generator*$_1$, instead of "similarity selection", we use "teacher forcing" and schedule sampling respectively, in order to verify that it was important to use this more reasonable method "similarity selection" on the twin-GAN model.

Table 2: Comparisons of different training strategies on twin-GAN based on IWSLT 2014 De$\rightarrow$En data set.

Strategies	BLEU
twin-GAN with "teacher forcing"	27.10
twin-GAN with schedule sampling	27.74
twin-GAN with "similarity selection"	**28.07**[†]

[†] the best performance of twin-GAN model with different strategies in *generator*$_1$

In addition, we evaluate different models, based on WMT'17 Zh$\rightarrow$En corpus. The results are shown in Table 3, where "RNN Enc-Dec Att" represents the encoder and decoder of the model are the same RNN architecture(2 layers LSTM network here) with an attention mechanism. We also compared the result of our work with Adversarial NMT (Wu et al., 2017). The result shows that our method improved the performance of those recurrent architecture model.

Table 3: Comparisons of different competitive NMT systems on WMT'17 Zh$\rightarrow$En data set.

Systems		BLEU
RNN Enc-Dec Attn		18.06
Adversarial NMT		19.90
twin-GAN	update/5 times, lr=1e-3	20.86
	update/10 times, lr= 5e-4	20.32
	update/10 times lr=1e-3	**21.06**[†]

[†] the best performance of our twin-GAN model

6 CONCLUSIONS

The end-to-end model is very popular and effective in Neural Machine Translation. But this process of generating translations word by word is different during training and inference, which can cause *exposure bias*. Many researchers tried to use Generative Adversarial Networks to address this problem and they indeed had some success, but there is still room for improvement. We introduce the first GAN with two generators and two discriminators for NMT. We not only make the training phase and test phase similar in data distribution, but also make them similar in model distribution to solve the *exposure bias* problem through one of the discriminators. In addition, the proposed "similarity selection" method in *generator*$_1$ can more accurately make the data distributions of the model similar at training and test. The results clearly show that after using this new model, it does improve the performance of the basic recurrent model. The twin-GAN also has enhanced the Adversarial NMT model after adding an extra "hidden state discriminator" and applying "similarity selection" method, achieving the competitive results on recurrent models.

ACKNOWLEDGEMENTS

This work was supported by National Key R&D Program of China under Grant 2018AAA0100202, and in part by the National Science Foundation of China under Grant 61976043.

REFERENCES

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2017). Unsupervised neural machine translation. *CoRR*.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2016). An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *ICLR*.

Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., and Federico, M. (2014). Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Empirical Methods in Natural Language Processing (EMNLP)*.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Goodfellow, I. (2016). Generative adversarial networks. *Conference and Workshop on Neural Information Processing Systems(NIPS)*, page 2672–2680.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

He, T., Tan, X., Xia, Y., He, D., Qin, T., Chen, Z., and Liu, T.-Y. (2018). Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*, pages 7944–7954.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huszár, F. (2015). How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. v. d., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

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

Lamb, A. and Xie, M. (2016). Convolutional encoders for neural machine translation. *WEB download*.

Lamb, A. M., Goyal, A. G. A. P., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Lample, G., Conneau, A., Denoyer, L., and Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *CoRR*.

Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. *In EMNLP, pp. 5039–5049*.

Li, B. and Han, L. (2013). Distance weighted cosine similarity measure for text classification. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 611–618. Springer.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Muflikhah, L. and Baharudin, B. (2009). Document clustering using concept space and cosine similarity measurement. In *2009 International Conference on Computer Technology and Development*, volume 1, pages 58–62. IEEE.

Nguyen, H. V. and Bai, L. (2010). Cosine similarity metric learning for face verification. In *Asian conference on computer vision*, pages 709–720. Springer.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2015). Minimum risk training for neural machine translation. *The Association for Computational Linguistics(ACL)*.

Song, K., Tan, X., He, D., Lu, J., Qin, T., and Liu, T.-Y. (2018). Double path networks for sequence to sequence learning. *In Proceedings of the 27th International Conference on Computational Linguistics, pp. 3064–3074*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Tan, X., Ren, Y., He, D., Qin, T., Zhao, Z., and Liu, T.-Y. (2019). Multilingual neural machine translation with knowledge distillation. *ICLR*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Venkatraman, A., Hebert, M., and Bagnell, J. A. (2015). Improving multi-step prediction of learned time series models. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Wiseman, S. and Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.

Wu, L., Xia, Y., Zhao, L., Tian, F., Qin, T., Lai, J., and Liu, T.-Y. (2017). Adversarial neural machine translation. *arXiv preprint arXiv:1704.06933*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR, abs/1609.08144*.

Yang, Z., Chen, W., Wang, F., and Xu, B. (2017). Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Zhang, W., Feng, Y., Meng, F., You, D., and Liu, Q. (2019). Bridging the gap between training and inference for neural machine translation. *The Association for Computational Linguistics(ACL)*.

Zhang, Y., Gan, Z., and Carin, L. (2016). Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, volume 21.