Neurocomputing 321 (2018) 146–155

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Generative adversarial training for neural machine translation

Zhen Yang[a,b,*], Wei Chen[b], Feng Wang[b], Bo Xu[b]

[a] University of Chinese Academy of Sciences, China
[b] Institute of Automation, Chinese Academy of Sciences, No. 95 Zhongguancun East Road, Beijing 100190, PR China

ARTICLE INFO

Article history:
Received 12 September 2017
Revised 7 July 2018
Accepted 10 September 2018
Available online 20 September 2018

Communicated by Dr. Tie-Yan Liu

Keywords:
Neural machine translation
Multi generative adversarial net
Human-like translation

ABSTRACT

Neural machine translation (NMT) is typically optimized to generate sentences which cover n-grams with ground target as much as possible. However, it is widely acknowledged that n-gram precisions, the manually designed approximate loss function, may mislead the model to generate suboptimal translations. To solve this problem, we train the NMT model to generate human-like translations directly by using the generative adversarial net, which has achieved great success in computer vision. In this paper, we build a conditional sequence generative adversarial net (CSGAN-NMT) which comprises of two adversarial sub models, a generative model (generator) which translates the source sentence into the target sentence as the traditional NMT models do and a discriminative model (discriminator) which discriminates the machine-translated target sentence from the human-translated one. The two sub models play a minimax game and achieve a win-win situation when reaching a Nash Equilibrium. As a variant of the single generator-discriminator model, the multi-CSGAN-NMT which contains multiple discriminators and generators, is also proposed. In the multi-CSGAN-NMT model, each generator is viewed as an agent which can interact with others and even transfer messages. Experiments show that the proposed CSGAN-NMT model obtains substantial improvements than the strong baseline and the improvement of the multi-CSGAN-NMT model is more remarkable.

1. Introduction

Machine translation [1,2] is a challenging task in traditional NLP area, which aims to translate one source-language sentence into the corresponding target-language sentence automatically. Recently, with the rapid development of deep neural networks, the neural machine translation [3–9] which leverages a single neural network directly to transform the source sentence into the target sentence, has obtained state-of-the-art performance for several language pairs [10–12]. Contrary to the traditional statistical machine translation (SMT) [13–15] which consists of many small subcomponents tuned separately, this end-to-end NMT model only consists of two sub recurrent neural nets. The encoder network reads and encodes the source sentence into the context vector representation; and the decoder network generates the target sentence word by word based on the context vector. To dynamically generate a context vector for a target word being generated, the attention mechanism which enables the model to focus on the relevant words in the source-side sentence, is usually deployed. Typically, the NMT model is optimized to directly maximize the likelihood of the training data. Specifically, at each decoding step, the NMT model is optimized to maximize the likelihood estimation of the ground word (MLE) at the current step. Ranzato et al. [8] indicate that the MLE loss function is only defined at the word level instead of the sentence level. Hence the NMT model may generate the best candidate word for the current time step yet a bad component of the whole sentence in the long run. Shen et al. [16] solve this problem by introducing the minimum risk training from SMT. They incorporate the sentence-level BLEU [17] into the loss function so that the NMT model can be optimized to generate sentences with high BLEU points directly. Since the BLEU point is computed as the geometric mean of the modified n-gram precisions [18], we conclude that almost all of the prior NMT models are trained to predict sentences which cover n-grams with the ground target sentence as much as possible (MLE can be viewed as training the NMT to cover more 1-gram with the target sentence).

However, it is widely acknowledged that higher n-gram precisions don't ensure a better sentence [19,20]. Additionally, the manually defined loss function is unable to cover all crucial aspects and the NMT model may be trained to deviate from the data distribution and generate suboptimal sentences. Intuitively, the model should be trained to directly generate a human-like translation instead of covering the human designed approximation features, i.e., n-gram precisions. From the Turing test perspective, the NMT

* Corresponding author at: Institute of Automation, Chinese Academy of Sciences, No. 95 Zhongguancun East Road, Beijing 100190, PR China.
E-mail addresses: yangzhen2014@ia.ac.cn, yangzhen.wuda@whu.edu.cn (Z. Yang).

https://doi.org/10.1016/j.neucom.2018.09.006
0925-2312/

model should be trained to generate the sentence which is indistinguishable from the human-generated one. Based on the analysis above, we propose that a good training objective for NMT includes: (1) no manually defined approximation feature is used to guide the NMT model; (2) the NMT model should be directly exposed to the true data distribution. Specifically, the model should be trained to directly output the translation indistinguishable from human-generated translations and if one poor sentence is generated, the model should be penalized with how far the poor sentence is from the human-generated one.

Borrowing the idea of generative adversarial training in computer vision [21–23], we build a conditional sequence generative adversarial net which implements the training objective mentioned above. In the proposed CSGAN-NMT, we jointly train two models, a generator (implemented as the traditional NMT model) which generates the target-language sentence based on the input source-language sentence, and a discriminator which conditioned on the source-language sentence, predicts the probability of the target-language sentence being a human-generated one. During the training process, the generator aims to fool the discriminator into believing that its output is a human-generated sentence, and the discriminator makes effort not to be fooled by improving its ability to distinguish the machine-generated sentence from the human-generated one. This single generator-discriminator training achieves a win-win situation when the generator and discriminator reaches a Nash Equilibrium [24]. As a variant, we also build the multi-CSGAN model for NMT (referred to as multi-CSGAN-NMT), which incorporates multiple generators and discriminators. The generators in the multi-CSGAN-NMT are viewed as agents and they not only learn from the data but also interact among themselves. The interaction among the generators include conceding and competing. Additionally, the generator is also enabled to pass messages to others. Experiments show that the interaction and message passing among the generators are beneficial to the translation performance of each generator.

In summary, we mainly make the following contributions:

- We introduce the generative adversarial training into NMT, which trains the NMT model to generate sentences which are indistinguishable from the human-generated sentences. We build a conditional generative adversarial net which can be applied to any end-to-end NMT systems. We do not assume the specific architecture of the NMT model. As a variant, we also build the multi-CSGAN model for NMT which enables the generators not only learn from large quantities of data but also learn from each other.
- The extensive experiments on Chinese-to-English and English-to-German translation tasks show that the proposed CSGAN-NMT significantly outperforms the strong attention-based NMT model, which serves as the baseline. We present detailed, quantitative results to demonstrate the effectiveness of the proposed CSGAN-NMT. Additionally, experiments show that the multi-CSGAN-NMT model achieves more substantial improvements.
- We report our specific training strategies for the proposed CSGAN-NMT. This provides a new reliable route for applying generative adversarial nets into other NLP tasks.

2. Related work

2.1. Neural machine translation

This subsection briefly describes the attention-based NMT model which simultaneously conducts dynamic alignment and generation of the target sentence. The NMT model produces the translation sentence by generating one target word at every time step. Given an input sequence $X = (x_1, \ldots, x_{T_x})$ and previous translated words $(y_1, \ldots, y_{i-1})$, the probability of next word y_i is:

$$p(y_i|y_1, \ldots, y_{i-1}, X) = g(y_{i-1}, s_i, c_i) \tag{1}$$

where s_i is an decoder hidden state for time step i, which is computed as:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{2}$$

Here f and g are nonlinear transform functions, which can be implemented as long short term memory network (LSTM) [30] or gated recurrent unit (GRU) [31], and c_i is a distinct context vector at time step i, which is calculated as a weighted sum of the input annotations h_j:

$$c_i = \sum_{j=1}^{T_x} a_{i,j} h_j \tag{3}$$

where h_j is the annotation of x_j calculated by a bidirectional RNN. The weight a_{ij} for h_j is calculated as:

$$a_{i,j} = \frac{exp(e_{ij})}{\sum_{t=1}^{T_x} exp(e_{i,t})} \tag{4}$$

where

$$e_{i,j} = v_a tanh(W s_{i-1} + U h_j) \tag{5}$$

where v_a is the weight vector, W and U are the weight matrixes. All of the parameters in the NMT model, represented as θ, are optimized to maximize the following conditional log-likelihood of the M sentence aligned bilingual samples:

$$\ell(\theta) = \frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{T_y} \log p(y_i^m | y_{<i}^m, X^m, \theta) \tag{6}$$

2.2. Generative adversarial net

Generative adversarial network in which a generative model is trained to generate outputs to fool the discriminator, has enjoyed great success in computer vision and has been widely applied to image generation [21]. The conditional generative adversarial nets apply an extension of generative adversarial network to a conditional setting, which enables the networks to condition on some arbitrary external data.

However, to the best of our knowledge, this idea has not been applied in traditional NLP tasks with comparable success and few quantitative experimental results has been reported. Some recent works have begun to apply the generative adversarial training into the NLP area: Chen et al. [25] apply the idea of generative adversarial training to sentiment analysis and Zhang et al. [26] use the idea to domain adaptation tasks. For sequence generation problem, Yu et al. [27] leverage policy gradient reinforcement learning to back-propagate the reward from the discriminator, showing presentable results for poem generation, speech language generation and music generation. Similarly, Zhang et al. [28] generate the text from random noise via adversarial training. Lin et al. [32] propose a novel generative adversarial network for generating high-quality language descriptions. Rather than training the discriminator to learn and assign absolute binary prediction, they train a discriminator to analyze and rank a collection of human-written and machine-written sentences. To provide intermediate information about text structure during the generative process, Guo et al. [33] propose a new framework to allow the discriminative net to leak its own high-level extracted features to the generative net to further help the guidance. Che et al. [34] propose maximum-likelihood augmented discrete generative adversarial nets. They derive a novel and low-variance objective using the discriminator's output that corresponds to the log-likelihood. In parallel to our

work, Li et al. [29] propose a similar conditional sequence generative adversarial training for dialogue generation. They use a hierarchical LSTM architecture for the discriminator.

The work of Wu et al. [35] is the most related work to ours, and they propose a similar approach for applying GAN to NMT. However, they only focus on the scenario where the GAN only consists of one generator and one discriminator. In this paper, we take the first step to investigate the more ambitious scenario where the GAN contains multiple generators and discriminators. The generators and discriminators are viewed as agents, which can concede, compete and pass message with each other. Furthermore, we present detailed training strategies for the proposed model and extensive quantitative results are reported.

3. The CSGAN-NMT

In this section, we describe in detail the CSGAN-NMT that consists of a generator G which generates the target-language sentence based on the source-language sentence and a discriminator D which distinguishes the machine-generated sentence from the human-generated one. The sentence generation process is viewed as a sequence of actions that are taken according to a policy regulated by the generator. In this work, we take the policy gradient training strategies which are same as Yu et al. [27].

3.1. Generator

Resembling the traditional NMT model, the generator G generates the target-language sentence conditioned on the input source-language sentence. It defines the policy that generates the target sentence y given the source sentence x. The generator takes exactly the same architecture with the traditional NMT model. Note that we do not assume the specific architecture of the generator. Here, we adopt the strong attention-based NMT model which is implemented as the open-source system dl4mt,[1] as the generator. The dl4mt system typically consists of an encoder which is implemented as one-layer bidirectional GRU, and a decoder which is implemented as one-layer conditional GRU.

3.2. Discriminator

Recently, the deep discriminative models such as the CNN and RNN have shown a high performance in complicated sequence classification tasks. In this paper, we implement the discriminator based on CNN.

Since sentences generated by the generator have a variable length, the CNN padding is used to transform the sentence to a sequence with the fixed length T which is the maximum length for the input sequence set by the user beforehand. Given the source-language sequence $x_1, \ldots, x_T$ and target-language sequence $y_1, \ldots, y_T$, we build the source matrix $X_{1:T}$ and target matrix $Y_{1:T}$ respectively as:

$$X_{1:T} = x_1; x_2; \ldots; x_T \tag{7}$$

and

$$Y_{1:T} = y_1; y_2; \ldots; y_T \tag{8}$$

where x_t, $y_t \in R^d$ is the d-dimensional word embedding and the semicolon is the concatenation operator. For the source matrix $X_{1:T}$, a kernel $w_j \in R^{l \times d}$ applies a convolutional operation to a window size of l words to produce a series of feature maps:

$$k_{ji} = \rho(BN(w_j \otimes X_{i:i+l-1} + b)) \tag{9}$$

where $\otimes$ operator is the summation of element-wise production and b is a bias term. ρ is a non-linear activation function which is implemented as ReLU in this paper. Note that the batch normalization [36] which accelerates the training significantly, is applied to the input of the activation function (represented as BN in Eq. (9)). To get the final feature with respect to kernel w_j, a max-over-time pooling operation is leveraged over the feature maps:

$$\widetilde{k}_j = max\{k_{j1}, \ldots, k_{jT-l+1}\} \tag{10}$$

We use various numbers of kernels with different window sizes to extract different features, which are finally concatenated to form the source-language sentence representation k_x. Identically, the target-language sentence representation k_y can be extracted from the target matrix $Y_{1:T}$. Finally, given the source-language sentence, the probability that the target-language sentence is being human-generated can be computed as:

$$p = \sigma(V[k_x; k_y]) \tag{11}$$

where V is the transform matrix which transforms the concatenation of k_x and k_y into a 2-dimension embedding and σ is the logistic function. The proposed discriminator is depicted as Fig. 1.

3.3. Policy gradient training

Following Yu et al. [27], the objective of the generator G is defined as to generate a sequence from the start state to maximize its expected end reward. Formally, the objective function is computed as:

$$J(\theta) = \sum_{Y_{1:T-1}} G_\theta(Y_{1:T}|X) \cdot R_D^{G_\theta}((Y_{1:T-1}, X), y_T) \tag{12}$$

where $Y_{1:T} = y_1, \ldots, y_T$ indicates the generated target sequence, θ represents the parameters of G, $R_D^{G_\theta}$ is the action-value function of a target-language sentence given the source sentence X, i.e. the expected accumulative reward starting from the state $(Y_{1:T-1}, X)$, taking action y_T, and following the policy G_θ. To estimate the action-value function, we consider the estimated probability of being human-generated by the discriminator D as the reward:

$$R_D^{G_\theta}((Y_{1:T-1}, X), y_T) = D(X, Y_{1:T}) - b(X, Y_{1:T}) \tag{13}$$

where $b(X, Y)$ denotes the baseline value to reduce the variance of the reward. Practically, we take $b(X, Y)$ as a constant, 0.5 for simplicity.[2] The question is that, given the source sequence, the discriminator D only provides a reward value for a finished target sequence. If $Y_{1:T}$ is not a finished target sequence, the value of $D(X, Y_{1:T})$ makes no sense. Therefore, we cannot get the action-value for an intermediate state directly. To evaluate the action-value for an intermediate state, the Monte Carlo search under the policy of G is applied to sample the unknown tokens. Each search ends until the end of sentence token is sampled or the sampled sentence reaches the maximum length. To obtain more stable reward and reduce the variance, we represent an N-time Monte Carlo search as:

$$\{Y_{1:T_1}^1, \ldots, Y_{1:T_N}^N\} = MC^{G_\theta}((Y_{1:t}, X), N) \tag{14}$$

where T_i represents the length of the sentence sampled by the ith Monte Carlo search. $(Y_{1:t}, X) = (y_1, \ldots, y_t, X)$ is the current state and $Y_{t+1:T_N}^N$ is sampled based on the policy G. The discriminator provides N rewards for the sampled N sentences respectively. The final reward for the intermediate state is calculated as the average of the N rewards. Hence, for the target sentence with the length T, we compute the reward for y_t in the sentence level as:

$$R_D^{G_\theta}((Y_{1:t-1}, X), y_t) = \tag{15}$$

[1] https://github.com/nyu-dl/dl4mt-tutorial

[2] We have tested different settings for the baseline value $b(x, y)$ such as 0.4, 0.5, 0.6 and 0.8, and we find that it shows little effect on the translation performance.

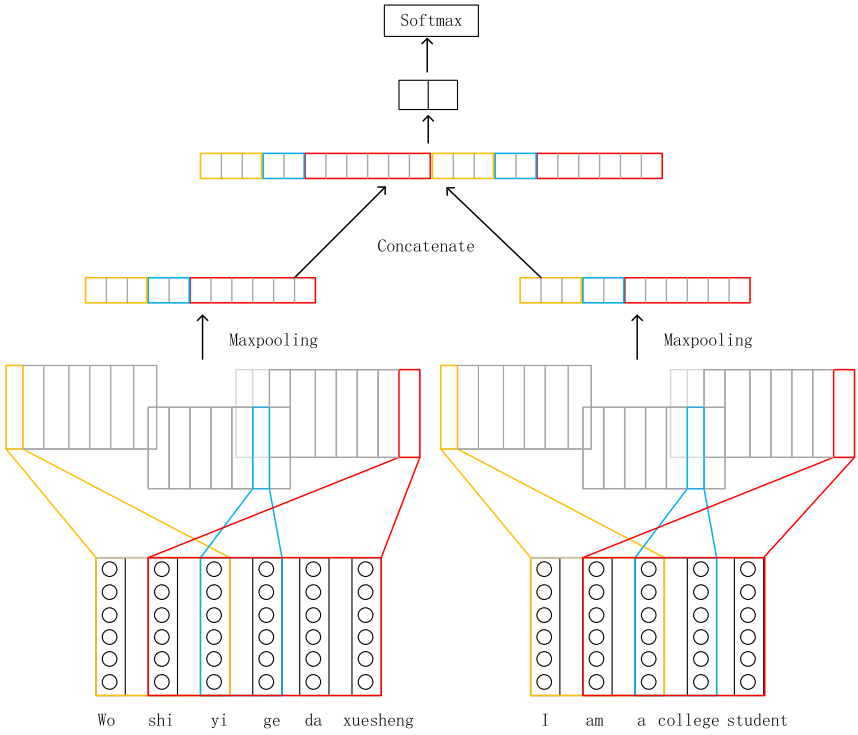


Fig. 1. The CNN-based architecture for the discriminator. The source sentence is a word sequence "wo shi yi ge da xuesheng" where the Chinese word "xuesheng" consists of two Chinese characters, i.e., "xue" and "sheng". The target sentence is "I am a college student".

$$\begin{cases} \frac{1}{N}\sum_{n=1}^{N} D(X, Y_{1:T_n}^n) - b(X, Y_{1:T_n}^n), Y_{1:T_n}^n \in MC^{G_\theta}((Y_{1:t}, X), N) & t < T \\ D(X, Y_{1:t}) - b(X, Y_{1:t}) & t = T \end{cases}$$

Using the discriminator as a reward function can further improve the generator iteratively by dynamically updating the discriminator. Once we get more realistic generated sequences, we re-train the discriminator as:

$$\min -\mathbb{E}_{X,Y\in P_{data}}[\log D(X,Y)] - \mathbb{E}_{X,Y\in G}[\log(1 - D(X,Y))] \tag{16}$$

After updating the discriminator, we are ready to re-train the generator. The gradient of the objective function $J(\theta)$ w.r.t the generator's parameter θ is calculated as:

$$\begin{aligned} \nabla J(\theta) &= \frac{1}{T}\sum_{t=1}^{T}\sum_{y_t} R_D^{G_\theta}((Y_{1:t-1}, X), y_t) \cdot \nabla_\theta(G_\theta(y_t|Y_{1:t-1}, X)) \\ &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{y_t\in G_\theta}[R_D^{G_\theta}((Y_{1:t-1}, X), y_t) \cdot \nabla_\theta \log p(y_t|Y_{1:t-1}, X)] \end{aligned} \tag{17}$$

3.4. Training strategies

It is hard to train the generative adversarial networks since the generator and discriminator need to be carefully synchronized. To make this work easier to reproduce, this paper gives detailed strategies for training the CSGAN-NMT model.

Firstly, we use the maximum likelihood estimation to pre-train the generator on the parallel training set until the best translation performance is achieved.

Then, generate the machine-generated sentences by using the generator to decode the training data. We simply use the greedy sampling method instead of the beam search method for decoding. Hence, it is very fast to decode all of the training set.

Next, pre-train the discriminator on the combination of the true parallel data and the machine-generated data until the classification accuracy of the discriminator ξ achieves at an appropriate value.[3]

Finally, we jointly train the generator and discriminator. The generator is trained with the policy gradient training method. We randomly sample a batch of source sentences from the parallel training set as the training examples for the generator. Note that the target sentences are useless when the generator is undergoing the policy gradient training. However, in our practice, we find that updating the generator only with the simple policy gradient training leads to unstable training. The translation performance drops sharply after a few updating. We conjecture that this is because the generator can only indirectly access to the golden target sentence through the reward passed back from the discriminator, and this reward is used only to promote or discourage the machine-generated sentences. To alleviate this issue, we adopt the teacher forcing approach which is similar to Lamb et al. [37] and Li et al. [29]. We directly make the discriminator to automatically assign a reward of 1 to the golden target-language sentence and the generator uses this reward to update itself on the true parallel examples. We run the teacher forcing training for one time once the generator is updated by the policy gradient training. After the generator gets updated, we use the new stronger generator to generate η more realistic sentences, which are then used to train the discriminator. Following Arjovsky et al. [38], we clamp the weights of the discriminator to a fixed box ($[-\epsilon, \epsilon]$) after each gradient update. We perform one optimization step for the discriminator for each step of the generator.

In our practice, we set ξ as 0.82, η as 5000, ϵ as 1 and the N for Monte Carlo search as 20. We apply the Adam optimization method, with the initial learning rate of 0.001, for pre-training the generator and discriminator. During the process of generative adversarial training, the RMSProp optimization method with the

[3] The ξ has great effect on the training process, which will be discussed in our experiments.

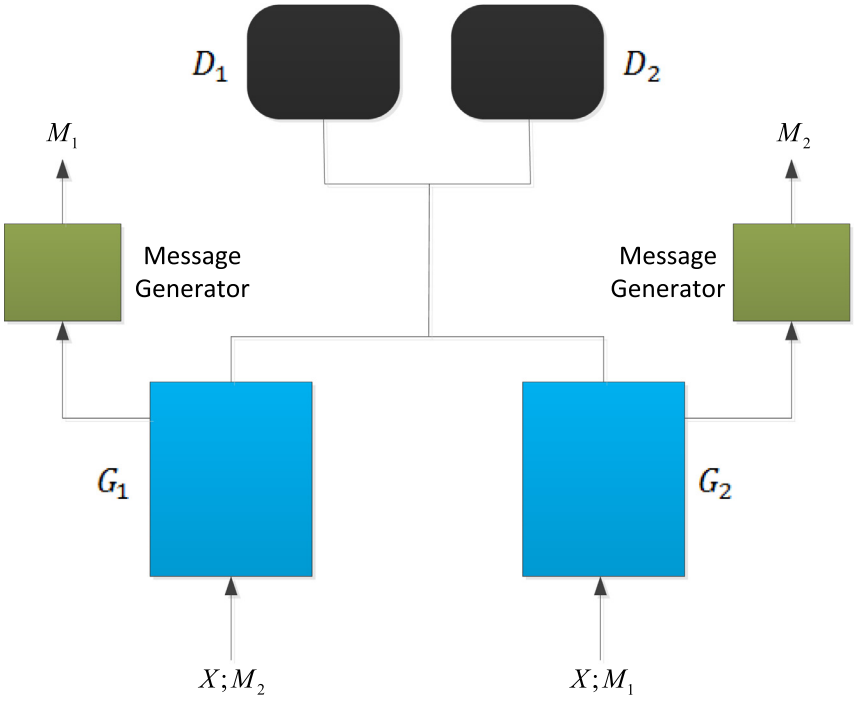


Fig. 2. The multi-CSGAN-NMT model which contains two generators G_1 and G_2, two discriminators D_1 and D_2, and two message generators. The message M_1 is passed and fed into G_2. Similarly, the message M_2 is passed and fed into G_1.

initial learning rate of 0.0001 is utilized for the generator and discriminator.

4. The multi-CSGAN-NMT

In this section, we describe the proposed multi-CSGAN-NMT model. We consider the setting where there are two generators and two discriminators. Each generator is viewed as an agent which can interact with others by competing and conceding. Each discriminator gives its own discrimination independently which is expected to be complementary to the other's. With the ensemble of predictions from the two discriminators, generators are able to receive more accurate rewards.

4.1. Model architecture

Fig. 2 illustrates the architecture of the proposed multi-CSGAN-NMT model which contains two generators and two discriminators. Each generator receives two rewards from D_1 and D_2 respectively and the reward for G_1 and G_2 is computed as:

$$R_D^{G_{1\theta}} = \alpha R_{D_1}^{G_{1\theta}} + (1-\alpha) R_{D_2}^{G_{1\theta}} \tag{18}$$

$$R_D^{G_{2\theta'}} = \alpha R_{D_1}^{G_{2\theta'}} + (1-\alpha) R_{D_2}^{G_{2\theta'}} \tag{19}$$

where α[4] is a hyper-parameter which can be set by the user beforehand, θ represents the parameters of G_1, θ' indicates the parameters of G_2. In the proposed multi-CSGAN-NMT model, each discriminator is trained such that both the generations by G_1 and G_2 are labeled to be as machine-translated sentences. Therefore, the objective functions for D_1 and D_2 can be computed respectively as:

$$\min -\mathbb{E}_{X,Y\in P_{data}}[\log D_1(X,Y)] \quad -\mathbb{E}_{X,Y\in G_1}[\log(1-D_1(X,Y))] \quad -\mathbb{E}_{X,Y\in G_2}[\log(1-D_1(X,Y))] \tag{20}$$

$$\min -\mathbb{E}_{X,Y\in P_{data}}[\log D_2(X,Y)] \quad -\mathbb{E}_{X,Y\in G_1}[\log(1-D_2(X,Y))] \quad -\mathbb{E}_{X,Y\in G_2}[\log(1-D_2(X,Y))] \tag{21}$$

where $X, Y \in P_{data}$ means the training examples from the true training data, $X, Y \in P_{G_1}$ means the training examples generated by G_1, and $X, Y \in P_{G_2}$ means training examples generated by G_2.

4.2. Competing and conceding

In the proposed multi-CSGAN-NMT model, each generator interacts with the other at every update. Specifically, the generator may compete with the other so that it gets better rewards for its generation from the discriminators. Based on this principle, we calculate the competing rewards for the two generators as:

$$R_D^{G_{1\theta}} = R_D^{G_{1\theta}} - max\left(\left(R_D^{G_{2\theta'}} - R_D^{G_{1\theta}}\right), 0\right) \tag{22}$$

$$R_D^{G_{2\theta'}} = R_D^{G_{2\theta'}} - max\left(\left(R_D^{G_{1\theta}} - R_D^{G_{2\theta'}}\right), 0\right) \tag{23}$$

The maximization objective for G_1 pushes it to get better rewards from the discriminators and vice versa for G_2. The two generators can also guide each other in order to get better scores for its generations from the discriminators. Hence we also introduce the conceding rewards for the generators, which are computed as:

$$R_D^{G_{1\theta}} = 1 - R_D^{G_{1\theta}} + max\left(\left(R_D^{G_{2\theta'}} - R_D^{G_{1\theta}}\right), 0\right) \tag{24}$$

$$R_D^{G_{2\theta'}} = 1 - R_D^{G_{2\theta'}} + max\left(\left(R_D^{G_{1\theta}} - R_D^{G_{2\theta'}}\right), 0\right) \tag{25}$$

The maximization objective for G_1 pushes G_2 to get better rewards from the discriminators and vice versa for G_2.

[4] We tested different values for α such as 0.4, 0.5, 0.6 and 0.8, and we find it shows little effect on the translation performance. We set it as 0.5 in our experiments

The competing and conceding objectives enable the two generators to communicate with each other implicitly. Other than learning from the training data, the generators are also able to learn from each other.

4.3. Message passing

While the generators can interact with each other through competing or conceding, they cannot communicate and share intelligence directly. The generator is unable to approach the hidden statues of the other generator, which is expected to be helpful for the coordination between generators. To address this limitation, we enable the generator to generate its message with the message generator module and pass its message to the other. The message generator modules are based on the principle that the messages passed between the two generators will make the generators explore different subspaces of sentence embedding space.

With the message passing, each generator in the proposed multi-CSGAN-NMT model generates the sentence conditioned on the input source-side sentence and the message that it receives from the other generator. The two generators share the message generator module which is implemented as a simple forward neural network layer. Specifically, the message $M_1 = (m_{11}, \ldots, m_{1T_x})$ and $M_2 = (m_{21}, \ldots, m_{2T_x})$ are the sequences with the same length of the input sequence X. The m_{ij} is computed as:

$$m_{ij} = W_{mg} h_{ij} \tag{26}$$

where h_{ij} is the h_j of the generator G_i (see Eq. (3)). The W_{mg} is the weight matrix of the message generator module which is shared by the two generators. With the message from the other generator, we compute the context vector c_i by replacing Eq. (3) with:

$$c_i = \sum_{j=1}^{T_x} a_{i,j} \widetilde{h}_j \tag{27}$$

where $\widetilde{h}_j$ is the new hidden vectors which summarizes the original hidden vectors and the message it receives. For G_1, the $\widetilde{h}_j$ is computed as:

$$\widetilde{h}_j = h_j + m_{2j} \tag{28}$$

And for $\widetilde{h}_j$ in generator G_2, we have:

$$\widetilde{h}_j = h_j + m_{1j} \tag{29}$$

Note that we still use the original hidden vector h_j to compute the weight a_{ij} with Eq. (4) and Eq. (5) for the generators.

4.4. Training details

The training strategies for the multi-CSGAN-NMT model is almost the same with the CSGAN-NMT. However, there are two special issues for training the multi-CSGAN-NMT model: (1) we pre-train the generator G_1 and G_2 with different parameter initialization methods so that they can achieve different initial conditions for the proposed multi-CSGAN-NMT model. With the same motivation, the parameters of D_1 and D_2 are also initialized with different methods. (2) During the teacher forcing training for the generators, we set the message from the other generator to be zero. This is to alleviate the inconsistency between the training and testing since the generator is tested independently without message from the other generator during inference. An overview of training the multi-CSGAN-NMT model is shown in Fig. 3.

Pre-train G_1, G_2, D_1, D_2
For number of training iterations **do**
  For i=1, D-steps **do**
    Sample （X,Y） from real data
    If message Passing
      Compute M_1, M_2
    Else
      M_1=M_2 = **0**
    Sample $\widehat{Y}_1 \sim G_1(\cdot|X, M_2)$
    Sample $\widehat{Y}_2 \sim G_2(\cdot|X, M_1)$
    Update D_1 using (X, Y) as positive data, (X, $\widehat{Y}_1$) and (X, $\widehat{Y}_2$) as negative data
    Update D_2 using (X, Y) as positive data, (X, $\widehat{Y}_1$) and (X, $\widehat{Y}_2$) as negative data
  For i=1, G-steps **do**
    Sample （X,Y） from real data
    If message Passing
      Compute M_1, M_2
    Else
      M_1=M_2 = **0**
    Sample $\widehat{Y}_1 \sim G_1(\cdot|X, M_2)$
    Sample $\widehat{Y}_2 \sim G_2(\cdot|X, M_1)$
    flag =randint(2)
    If flag ==0 // do competing
      Compute competing reward r_1 for (X, $\widehat{Y}_1$) using D_1, D_2
      Compute competing reward r_2 for (X, $\widehat{Y}_2$) using D_1, D_2
    Else // do conceding
      Compute conceding reward r_1 for (X, $\widehat{Y}_1$) using D_1, D_2
      Compute conceding reward r_2 for (X, $\widehat{Y}_2$) using D_1, D_2
    Update G_1 on (X, M_2, $\widehat{Y}_1$) using reward r_1
    Update G_2 on (X, M_1, $\widehat{Y}_2$) using reward r_2
    Teacher-Forcing:
      M_1=M_2 = **0** **Update** G_1 on （X, M_2, Y）, update G_2 on （X, M_1, Y）

Fig. 3. An overview for training the multi-CSGAN-NMT model. The reward r can be competing reward or conceding reward randomly. The training iteration is the maximum epoch for training the model. D-steps and G-steps are both set to 1. We have tested different settings for D-steps and G-steps, and we obtain almost the same results. For speeding the training, we set them both as 1.

5. Experimental setup

We mainly evaluated our approaches on the widely used NIST Chinese–English translation task. In order to show the effectiveness of our approaches, we also provide results on the English–German translation task.

5.1. Data sets

Chinese–English LDC. For Chinese–English, our training data consists of 1.25M sentence pairs randomly extracted from LDC corpora.[5] We choose the NIST02 as the development set. For testing, we use NIST03, NIST04 and NIST05 data sets. We apply word-level translation and the Chinese sentences are segmented beforehand. To speed up the training procedure, the sentences of length over 50 words are removed. We limit the vocabulary in both Chinese and English to the most 30K words and the out-of-vocabulary words are replaced with UNK.

WMT'14 English–German. For English–German, we use the WMT 2014 training corpus that contains 4.5M sentence pairs.[6] We report results on newstest2014. The newstest2013 is used as validation. As vocabulary, we use 30K sub-word based on byte-pair encoding [39]. The sentences of length over 50 words are removed.

5.2. Model parameters and evaluation

We use 512 hidden units for both encoders and decoders, unless otherwise stated. All embeddings also have dimensionality 512. The other hyper-parameters are set according to the

[5] LDC2002L27, LDC2002T01, LDC2002E18, LDC2003E07, LDC2004T08, LDC2004E12, LDC2005T10.

[6] http://nlp.stanford.edu/projects/nmt

Table 1
BLEU score on Chinese–English translation tasks compared to strong previous works.

Models	NIST02	NIST03	NIST04	NIST05	Ave
Bahdanau et al. [3]	36.07	32.13	34.02	30.30	32.15
Shen et al. [16]	36.64	32.55	34.61	31.04	32.73
CSGAN-NMT	37.41	33.42	34.85	31.80	33.35

Table 2
The average BLEU score on the test sets for the Chinese–English translation task. G_1 and G_2 are the two generators. The CSGAN-NMT means that the two generators are trained independently with the CSGAN-NMT model. The multi-CSGAN-NMT indicates that the two generators are trained jointly in the multi-CSGAN-NMT model. We also report the results of the ensemble prediction.

Models	G_1	G_2	Ensemble
Bahdanau et al. [3]	32.15	30.01	34.23
CSGAN-NMT	32.73	30.92	35.64
multi-CSGAN-NMT	33.05	30.16	36.02
multi-CSGAN-NMT+messagePassing	33.20	30.92	36.21

Table 3
BLEU scores on English–German translation tasks. We compare the proposed CSGAN-NMT model with the strong baseline.

Models	newstest2013	newstest2014
Bahdanau et al. [3]	20.7	20.4
Shen et al. [16]	20.9	20.8
CSGAN-NMT	21.5	21.3

Table 4
The BLEU score on the newstest2014 for the English–German translation task. G_1 and G_2 are the two generators. The CSGAN-NMT means that the two generators are trained independently with the CSGAN-NMT model. The multi-CSGAN-NMT indicates that the two generators are trained jointly in the multi-CSGAN-NMT model. We also report the results of the ensemble prediction.

Models	G_1	G_2	Ensemble
Bahdanau et al. [3]	20.4	19.3	21.8
CSGAN-NMT	21.3	19.9	22.5
multi-CSGAN-NMT	21.7	20.1	22.7
multi-CSGAN-NMT+messagePassing	21.9	20.2	22.8

Section 3.4. The evaluation metric is BLEU [18]. For Chinese–English, we apply case-insensitive BLEU which is widely used in Chinese–English translation tasks. For English–German, we evaluate the translation performance with the script *multi-belu.pl*.[7] Translations are generated by a beam search and the log-likelihood scores are not normalized by sentence length. We use a beam width of 10 in all the experiments. Dropout is not used in our experiments. All models are implemented in TensorFlow [40] and trained on up to four K80 GPUs synchronously in a multi-GPU setup on a single machine.

6. Results

In this section, we detail our experimental results. The generator of the proposed model is implemented the same with the open-source NMT system dl4mt, which implements an attentional encoder-decoder architecture similar to Bahdanau et al. [3]. The system has been used to build top-performing submissions to shared translation tasks at WMT and IWSLT [41]. Therefore, the generator is a strong baseline in our experiments.

6.1. Results on Chinese–English translation

For Chinese–English translation, the model is optimized with the mini-batch of 64 examples and it takes about 25 h to pre-train the generator. We need 15 more hours to train the CSGAN-NMT model and 25 more hours to train the multi-CSGAN-NMT model.

We compare the proposed CSGAN-NMT model to the following work: Bahdanau et al. [3], i.e., the implementation of the generator, is the traditional attentional encoder-decoder architecture and the Shen et al. [16] introduces the minimum risk training for the attention-based NMT model. The results (Table 1) show that the proposed CSGAN-NMT model leads to improvement up to +1.20 BLEU points averagely compared to Bahdanau et al. [3] and our model outperforms Shen et al. [16] by +0.62 BLEU points on average.

Table 2 shows the average BLEU score over the three test sets for the multi-CSGAN-NMT model. The two generators are pre-trained until convergence and they reach different initial conditions, i.e., 32.15 BLEU points for G_1 and 30.01 BLEU points for G_2 (line 2 in Table 2). To show the effectiveness of the multi-CSGAN-NMT model, we compare the translation performances of the two generators in two different conditions: The two generators are trained independently with the CSGAN-NMT (line 3 in Table 2); The two generators are trained jointly in the multi-CSGAN-NMT (line 4 in Table 2). In the multi-CSGAN-NMT model, as our focus is on obtaining a single effective network, we only care for the translation performance of G_1 and we select models according to the behavior of G_1 on the development set.[8] The results show that the multi-CSGAN-NMT model outperforms the CSGAN-NMT model by +0.32 BLEU points, which indicates that the generator benefits from interacting with others. With the message passing, the multi-CSGAN-NMT model is able to achieve another +0.15 BLEU points improvement. We also report the translation performance of the ensemble of the two generators. We can find that while the translation performance of the G_2 in the multi-CSGAN-NMT is worse than the translation performance of its counterpart in the CSGAN-NMT, the ensemble of the two generators in the multi-CSGAN-NMT model still outperforms the ensemble of the two counterparts trained independently with the CSGAN-NMT model by +0.36 BLEU points. From the results we can see that the interaction between the two generators is beneficial to the ensemble prediction. We leave the explanation for this phenomenon as the future work.

6.2. Results on English–German translation

For the English–German translation task, the model is optimized with the mini-batch of 40 examples and it takes about 2 days to pre-train the generator. It takes one more day to train the CSGAN-NMT model and 40 more hours to train the multi-CSGAN-NMT model.

Table 3 shows that the proposed CSGAN-NMT outperforms [3] by 0.9 BLEU points on newstest2014. And compared to [16], our model also achieves +0.5 BLEU points improvement. We report the translation performance of the multi-CSGAN-NMT model in Table 4. The two generators achieve the BLEU score at 20.4 and 19.3, respectively, after the pre-training finished. When the two generators are trained with the CSGAN-NMT model independently, they achieve the BLEU score at 21.3 and 19.9, which outperform the baseline by +0.9 and +0.6 BLEU points. When trained jointly with the multi-CSGAN-NMT model, they can beat the CSGAN-NMT with the margin of +0.3 and +0.2 BLEU points. With the message passing, they can achieve another +0.2 and +0.1 BLEU points

[7] https://github.com/moses-smt/mosesdecoder/blob/617e8c8/scripts/generic/multi-bleu.perl;mteval-v13a.pl

[8] As we select models on the behavior of G_1, we can understand why G_2 in mutli-CSGAN-NMT performs worse than it in CSGAN-NMT.

Table 5
The human evaluations on Chinese–English and English–German translation tasks.

Translation	CSGAN-win	CSGAN-lose	Tie
Chinese–English	0.71	0.13	0.16
English–German	0.64	0.15	0.21

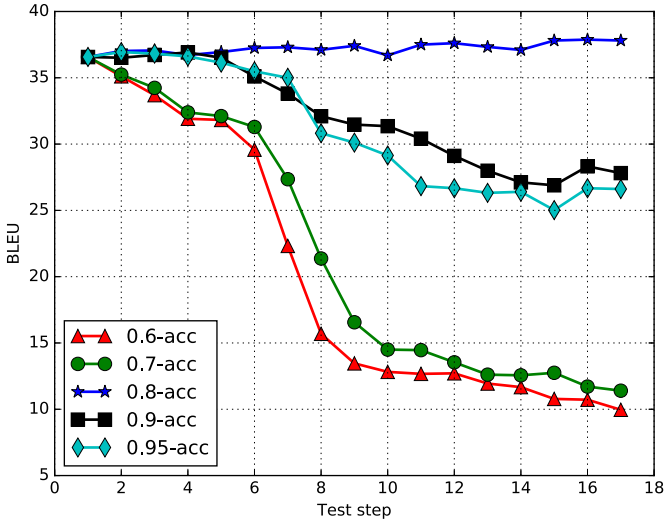


Fig. 4. BLEU score on the development set for the CSGAN-NMT where the discriminators have different initial accuracy. "0.6-acc" means the initial accuracy is 0.6. We report the results on the Chinese–English translation tasks.

improvements. To show the effectiveness of the proposed model, we also report the translation performance of the ensemble.

6.3. Human evaluation

We conduct human evaluations on Chinese–English and English–German translation tasks. For each translation task, we evaluate a random sample of 200 items from the test set [29]. We compare the baseline model, i.e., Bahdanau et al. [3], and the proposed CSGAN-NMT. We present both an source sentence and its corresponding generated sentences to 3 judges which are professional human translators, and ask them to decide which of the two output is better. Ties are allowed. The results are reported in Table 5. We observe a significant quality improvement on both Chinese–English and English–German translation tasks from the proposed CSGAN-NMT model.

7. Analysis

We analyze the effectiveness of the proposed CSGAN-NMT model on Chinese–English translation tasks by testing the effect of the hyper-parameters and the sub modules.

7.1. Initial accuracy of the discriminator

The initial accuracy ξ of the discriminator which can be viewed as a hyper-parameter, can be controlled carefully during the process of pre-training. A natural question is that when shall we end the pre-training. Do we need to pre-train the discriminator until that its accuracy reaches as high as possible? To answer this question, we test the impact of the initial accuracy of the discriminator. We pre-train five discriminators which have the accuracy as 0.6, 0.7, 0.8, 0.9 and 0.95, respectively. With the five discriminators, we train five different CSGAN-NMT models and test their translation performances on the development set at regular intervals. Fig. 4 reports the result and we can find that the initial accuracy of the discriminator shows great impacts on the translation performance of the proposed model. From Fig. 4, we show that the initial accuracy of the discriminator needs to be set carefully and no matter it is set too high (0.9 and 0.95) or too low (0.6 and 0.7), the CSGAN-NMT performs badly. This suggests that it is important for the generator and discriminator to keep a balance relationship at the beginning of the generative adversarial training. If the discriminator is too strong (with high accuracy), the generator is always penalized for its bad predictions and gets no idea about how to give right predictions. Hence, the generator is discouraged all the time and the performance gets worse and worse. On the other hand, if the discriminator is too weak (with low accuracy), the discriminator is unable to give right guidance for the generator, i.e. the gradient direction for updating the generator is random. Empirically, we pre-train the discriminator until its accuracy reaches 0.82.

Table 6
The translation performance of the CSGAN-NMT model with different N for Monte Carlo search. "–" means that the proposed model shows no improvement than the baseline model or it cannot be trained stably. With N set as 0, the model is the baseline model which is not trained by the CSGAN-NMT.

N	NIST02	NIST03	NIST04	NIST05
0	36.07	32.13	34.02	30.30
5	–	–	–	–
10	–	–	–	–
15	37.08	32.69	34.57	31.15
20	37.41	33.42	34.85	31.80
25	37.38	33.44	35.21	31.61
30	37.40	33.51	35.14	31.60

7.2. Sample times for Monte Carlo search

We are also curious about how the sample times N for Monte Carlo search affects the translation performance. Intuitively, if N is set as a small number, the intermediate reward computed as Eq. (15) may be incorrect and if otherwise, the computation shall be very time consuming. There is a trade-off between the accuracy and computation complexity here. We investigate this problem on the Chinese–English translation tasks. Table 6 presents the translation performance of the CSGAN-NMT on the test sets when the N are set from 5 to 30 with interval 5. From Table 6, the proposed CSGAN-NMT model achieves no improvement than the baseline when N are set less than 15 and we don't report the BLEU score on the table. As a matter of fact, the translation performance of the CSGAN-NMT gets worse and worse and the BLEU score reaches near zero in the end. With N set as 30, we get little improvement than the model with N set as 20. However, the training time has exceeded our expectation. Therefore, we set N as 20 in our experiments.

8. Conclusions and future work

In this work, we propose the CSGAN-NMT which leverages the generative adversarial training to improve the neural machine translation. As a variant, we also propose the multi-CSGAN-NMT model which contains multiple generators and discriminators. In the multi-CSGAN-NMT, each generator is viewed as an agent and has the ability to interact with others by the competing objective function and conceding objective function. Apart from this, the generators can share messages with each other directly. Experimental results show that our proposed CSGAN-NMT model can significantly outperform the strong previous works and the multi-CSGAN-NMT model even achieves more improvements. Additionally, we provide detailed training strategies for the CSGAN-NMT

model since GAN is widely criticized for its notorious instability.

In the future, we plan to test our method in other NLP tasks, like dialogue system and question answering. We notice that it is somewhat time-consuming to train the proposed model since the generator and discriminator need to be pre-trained. It is very interesting to modify the model so that the pre-training process can be dispensed. Since the higher BLEU point does not ensure a better sentence, another interesting direction is to apply the discriminator to measure the translation performance more equally. We also believe that it deserves much effort to reduce the hyper-parameters of the proposed model in the future work.

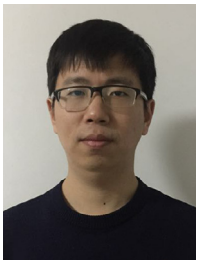
Acknowledgment

This work is supported by National Program on Key Basic Research Project of China (973 Program) (Grant No. 2013CB329302). We would like to thank Xu Shuang for her preparing data used in this work. Additionally, we also want to thank Chen Zhineng for his invaluable discussions on this work.

References

[1] J.A. Silvestre-Cerd, J. Andrs-Ferrer, C. Jorge, Explicit length modelling for statistical machine translation, Pattern Recognit. 45 (2012) 3183–3192.
[2] P. Gupta, M.R. Costa-juss, P. Rosso, E.B. Rafael, A deep source-context feature for lexical selection in statistical machine translation, Pattern Recognit. Lett. 75 (2016) 24–29.
[3] D. Bahdanau, K. Cho, B. Yoshua, Neural Machine Translation by Jointly Learning to Align and Translate, in: Proceedings of the International Conference on Learning Representations (ICLR), 2015.
[4] K. Cho, B.V. Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, B. Yoshua, Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.
[5] H. Choi, K. Cho, B. Yoshua, Context-dependent word representation for neural machine translation, Comput. Speech Lang. 45 (2017) 149–160.
[6] O. Firat, K. Cho, B. Sankaran, F.T.Y. Vural, Y. Bengio, Multi-way, multilingual neural machine translation, Comput. Speech Lang. 45 (2017) 236–252.
[7] N. Kalchbrenner, B. Phil, Recurrent continuous translation models, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2013, pp. 1700–1709.
[8] M. Ranzato, S. Chopra, M. Auli, W. Zaremba, Sequence Level Training with Recurrent Neural Networks, 2015, arXiv preprint:1511.06732.
[9] I. Sutskever, O. Vinyals, V.V.L. Quoc, Sequence to sequence learning with neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.
[10] M. Johnson, M. Schuster, Q.V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al., Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation, Transactions of the Association of Computational Linguistics 5 (1) (2017) 339–351.
[11] J. Bradbury, S. Richard, MetaMind neural machine translation system for WMT 2016., in: Proceedings of the First Conference on Machine Translation, Association for Computational Linguistics, Berlin, Germany, 2016.
[12] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation, 2016, arXiv preprint:1609.08144.
[13] V. Alabau, A. Sanchis, F. Casacuberta, Improving on-line handwritten recognition in interactive machine translation, Pattern Recognit. 47 (3) (2014) 1217–1228.
[14] P. Martnez-Gmez, G. Sanchis-Trilles, F. Casacuberta, Online adaptation strategies for statistical machine translation in post-editing scenarios, Pattern Recognit. 45 (9) (2012) 3193–3203.
[15] M.S. Maucec, Z. Kacic, D. Verdonik, Statistical machine translation of subtitles for highly inflected language pair, Pattern Recognit. Lett. 46 (2014) 96–103.
[16] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, Y. Liu, Minimum Risk Training for Neural Machine Translation, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), 1, 2015, pp. 1683–1692.
[17] B. Chen, C. Cherry, A systematic comparison of smoothing techniques for sentence-level BLEU, in: Proceedings of the 2014 Association for Computational Linguistics (ACL), 2014, p. 362.
[18] K. Papineni, S. Roukos, T. Ward, W.J. Zhu, BLEU: a method for automatic evaluation of machine translation, in: Proceedings of the 2002 Association for Computational Linguistics, 2002, pp. 311–318.
[19] C. Callison-Burch, O. Miles, Re-Evaluating the Role of BLEU in Machine Translation Research, in: 11th Conference of the European Chapter of the Association for Computational Linguistics, 2006.
[20] N. Chatterjee, A. Johnson, M. Krishna, Some improvements over the BLEU metric for measuring translation quality for hindi, in: Proceedings of the International Conference on Computing: Theory and Applications, ICCTA'07, IEEE, 2007, pp. 485–490.
[21] E.L. Denton, S. Chintala, R. Fergus, et al., Deep generative image models using an Laplacian pyramid of adversarial networks, in: Proceedings of the 2015 Advances in Neural Information Processing Systems, 2015, pp. 1486–1494.
[22] A. Ghosh, V. Kulharia, V. Namboodiri, Message Passing Multi-Agent GANs, 2016,
[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 2014 Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[24] J. Zhao, M. Mathieu, Y. LeCun, Energy-Based Generative Adversarial Network, 2016, arXiv preprint:1609.03126.
[25] X. Chen, Y. Sun, B. Athiwaratkun, C. Cardie, K. Weinberger, Adversarial Deep Averaging Networks for Cross-Lingual Sentiment Classification, 2016, arXiv preprint:1606.01614.
[26] Y. Zhang, R. Barzilay, T. Jaakkola, Aspect-Augmented Adversarial Networks for Domain Adaptation, Transactions of the Association of Computational Linguistics 5 (1) (2017) 515–528.
[27] L. Yu, W. Zhang, J. Wang, Y. Yong, SeqGAN: sequence generative adversarial nets with policy gradient, in: Proceedings of the 2016 Association for the Advancement of Artificial Intelligence, 2016.
[28] Y. Zhang, Z. Gan, L. Carin, Generating text via adversarial training, in: Proceedings of the 2016 Conference on Neural Information Processing Systems (NIPS), 2016.
[29] J. Li, W. Monroe, T. Shi, A. Ritter, D. Jurafsky, Adversarial Learning for Neural Dialogue Generation, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing(EMNLP), 2017, pp. 2157–2169.
[30] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.
[31] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical Evaluation of gated Recurrent Neural Networks on Sequence Modeling, in: NIPS 2014 Workshop on Deep Learning, December 2014.
[32] K. Lin, D. Li, X. He, Z. Zhang, M.T. Sun, Adversarial ranking for language generation, in: Proceedings of the 2017 Advances in Neural Information Processing Systems, 2017.
[33] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, J. Wang, Long Text Generation via Adversarial Training with Leaked Information, 2017, arXiv preprint:1709.08624.
[34] T. Che, Y. Li, R. Zhang, W.L. Hjelm, Y. Song, Y. Bengio, Maximum-Likelihood Augmented Discrete Generative Adversarial Networks, 2017, arXiv preprint:1702.07983.
[35] L. Wu, Y. Xia, L. Zhao, F. Tian, T. Qin, J. Lai, T.Y. Liu, Adversarial Neural Machine Translation, 2017, arXiv preprint:1704.06933.
[36] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network training by Reducing Internal Covariate Shift, 2015, arXiv preprint:1502.03167.
[37] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, Y. Bengio, Professor forcing: a new algorithm for training recurrent networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 4601–4609.
[38] M. Arjovsky, S. Chintala, B. Léon, Wasserstein Gan, 2017, arXiv preprint:1701.07875.
[39] R. Sennrich, B. Haddow, B. Alexandra, Neural Machine Translation of Rare Words with Subword Units, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (2016).
[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, Tensorflow: Large-Scale Machine Learning on Heterogeneous Systems, in: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation, 2015, pp. 265–283.
[41] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A.V.M. Barone, J. Mokry, Nematus: A Toolkit for Neural Machine Translation, in: Software Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics(EACL), 2017.

Zhen Yang is a Ph.D. candidate in the Institute of Automation of Chinese Academy Sciences (CASIA), Beijing, China. His research interests include deep learning, machine translation and generative adversarial net.



Wei Chen received B.S. degree in Harbin Institute of Technology, Ph.D. degree in Institute of Automation, Chinese Academy of Science, and now is an assistant professor at Institute of Automation, Chinese Academy of Science. His research interests include NLP, ASR and MT.

Feng Wang received B.S. degree in the Hunan Agricultural University and Master degree in the North University of China. Now he is an assistant professor at Institute of Automation of Chinese Academy Sciences. His interests include NLP and ASR.

Bo Xu is a Professor at the Institute of Automation of Chinese Academy of Sciences, Beijing, China. He received his B.S. degree in Zhejiang University and his Ph.D. degree in Institute of Automation of Chinese Academy of Sciences. His interests include NLP, ASR and human-like intelligence.