Improving Mongolian-Chinese Neural Machine Translation with Morphological Noise

Yatu JI* **Hongxu HOU*†** **Nier WU*** **Junjie CHEN***
jiyatu0@126.com cshhx@imu.edu.cn wunier04@126.comchenjj@imau.edu.cn
*Computer Science dept, Inner Mongolia University / Hohhot, China

Abstract

For the translation of agglutinative language such as typical Mongolian, unknown (UNK) words not only come from the quite restricted vocabulary, but also mostly from misunderstanding of the translation model to the morphological changes. In this study, we introduce a new adversarial training model to alleviate the UNK problem in Mongolian→Chinese machine translation. The training process can be described as three adversarial sub models (generator, value screener and discriminator), playing a win−win game. In this game, the added screener plays the role of emphasizing that the discriminator pays attention to the added Mongolian morphological noise[1] in the form of pseudo-data and improving the training efficiency. The experimental results show that the newly emerged Mongolian→Chinese task is state-of-the-art. Under this premise, the training time is greatly shortened.

1 Introduction

The dominant neural machine translation (NMT) (Sutskever et al., 2014) models are based on recurrent (RNN, (Mikolov et al., 2011)), convolutional neural networks (CNN, (Gehring et al., 2017)) or entirely eliminates recurrent connections and relies instead on a repeated attention mechanism (Transformer, (Vaswani et al., 2017)) which are achieved by an attention mechanism (Bahdanau et al., 2014). A considerable weakness in these NMT systems is their inability to correctly translate very rare words: end-to-end NMTs tend to have relatively small vocabularies with a single $< unk >$ symbol that represents every possible out-of-vocabulary (OOV) word. The problem is more prominent in agglutinative language tasks, because the varied morphology brings great confusion to model decoding. The change of suffix and component case[2] in Mongolian largely deceives the translation model directly resulting in a large amount of OOV during decoding. This OOV is then crudely considered the same as an $< unk >$ symbol.

Generally, there are three ways to solve this problem. A usual practice is to speed up training (Morin and Bengio, 2005; Jean et al., 2015; Mnih et al., 2013), these approaches can maintain a very large vocabulary. However, it works well when there are only a few unknown words in the target sentence. These approaches have been observed that the translation performance degrades rapidly as the number of unknown words increases. Another aspect is the information in context (Luong et al., 2015; Hermann et al., 2015; Gulcehre et al., 2016), they motivate their work from a psychological evidence that humans naturally have a tendency to point towards objects in the context. The last aspect is the input/output change, these approaches change to a smaller resolution, such as characters (Graves, 2013a) and bytecodes (Sennrich et al., 2016). However, it is worth thinking that the training process usually becomes much harder because of the length of sequences considerable increases.

For NLP tasks, generative adversarial network (GAN) is immature. Some studies, such as (Chen et al., 2016; Zhang et al., 2017), used GAN for semantic analysis and domain adaptation. (Yu et al., 2016; Zhen et al., 2018; Wu et al., 2018) successfully applied GAN to sequence generation tasks. (Zhang Y, 2017) propose matching the high-dimensional latent feature distributions of re-

[1]These morphological noises exist in most agglutinative languages in the form of appended stem, which are used to determine the presentation or tense of words. Some typical noises, such as suffixes and cases in Mongolian, Korean and Japanese.

[2]Mongolian-case is a special suffix used to determine its relationship to other words in a sentence.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 123–129
Florence, Italy, July 28 - August 2, 2019.

al and synthetic sentences, via a kernelized discrepancy metric. This eases adversarial training by alleviating the mode-collapsing problem.

In the present study, GAN is used for UNK problem. The motivation for this is GAN′s advantage in approaching real data effectively based on noise in a game training. To obtain generalizable adversarial training, we propose a noise-added strategy to add noise samples into the training set in the form of pseudo data. The noise is the main cause of UNK, such as the segmentation of suffixes and the handling of case components in Mongolian. A representative example is used to illustrate the decoding search process of Mongolian sentences in adversarial training (Fig. 1). During decoding, decoder usually can not solve the problem of morphological variability of words (caused by morphological noise) through vocabulary, which leads to OOV. Therefore, we introduce GAN mod-

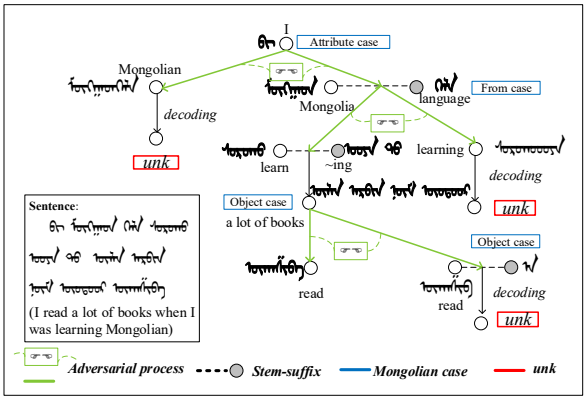

Figure 1: Given a sentence, Mongolian words face different suffix and case noises in each decoding process of the adversarial training, which are the main reasons for $< unk >$. For instance, the verb '*(learn)*' and '*(read)*' need to add the verb-suffixes and tense-cases in order to associate with nouns in Mongolian. Conversely, (*"learning"*) and (*read+' '*), which are confused by suffixes and cases, do not appear in the vocabulary. This will directly cause <*unk*> appear in the decoding process. The proposed model aims to improve the generalization ability of noise through adversarial training.

el with a value screener (VS-GAN), a generalization of GAN, which makes the adversarial training specific to the noise. The model also improves the efficiency of GAN training by value iteration network (VIN) (Tamar et al., 2016) and addresses the problem of optimal parameter updating in Reinforcement Learning(RL) training. These are our two contributions. The third contribution is a thorough empirical evaluation on four different noises. We compare several strong baselines, including MIXER (Ranzato et al., 2015), Transformer (Vaswani et al., 2017), and BR-CSGAN (Zhen et al., 2018). The experimental results show that VS-GAN achieves much better time efficiency and the newly emerged state-of-the-art result on Mongolian-Chinese MT.

2 GAN with the Value Screener

In this section, we describe the architecture of VS-GAN in detail. VS-GAN consists of the following components: generator G, value screener, and discriminator D. Given the source language sequence $\{x_1, ..., x_{N_x}\}$ with length N, G aims to generate sentences $\left\{y_1^{'}, ..., y_{N_y}^{'}\right\}$, which are indistinguishable by D. D attempts to discriminate between $\left\{y_1^{'}, ..., y_{N_y}^{'}\right\}$ and human translated ones $\{y_1, ..., y_{N_y}\}$. The value screener uses the reward information generated by G to convert the decoding cost into a simple value, and determines whether the predictions of current state need to be passed to D.

2.1 Generator G

The selection of G is individualized and targeted. In this work, we focus on long short term memory (LSTM(Graves, 2013b)) with attention mechanism and Transformer (Vaswani et al., 2017). The temporal structure of LSTM enables it to capture dependency semantics in agglutinative language. Transformer has refreshed state-of-the-art performance on several languages pairs. For the necessary policy optimization in GAN training, we focus our problem on the RL framework (Mnih et al., 2013). The approach can solve the long-term reward problem because a standard model for sequential decision making and planning is the markov decision process (MDP) (Dayan and Abbott, 2003) in RL training. G can be viewed as an agent which interacts with the external environment (the words and the context vector at every timestep). The parameters of agent define a policy θ, whose execution results in the agent is selecting an action $a \epsilon A$. In NMT, an action represents the prediction of the next word $y_t^{'}$ in the sequence at t-th timestep. After taking an action, the agent will update its internal state $s \epsilon S$ (i.e., the hidden units). RL will observe a reward $R(s, a)$ once the end of a sequence (or the maximum sequence length) is reached. We can choose any reward function,

and in this case, we choose BLEU because it is the metric we used at the test time.

2.2 Value Screener

So far, the constructed G is still confused by noise because the effect of noise has not been fully utilized due to the lack of attention from D. To solve this problem, we add a VIN implemented value screener between G and D to enhance the generalization ability of G to the noise. In VIN, the $< unk >$ symbol corresponds to a low training reward, whereas the low training reward corresponds to a low value. This is what the screener wants to emphasize.

To achieve VIN, we introduce an interpretation of an approximate VI algorithm as a particular form of a standard CNN. Specifically, VI in this form, which makes learning the MDP (R., 1957; Bertsekas., 2012) parameters and reward function natural by backpropagation through the network. We can train the entire policy end-to-end on the basis of its simplification by backpropagation. For the training process, each iteration of VI algorithm can be seen as passing the previous value of V_{t-1} and reward R by a convolution layer and max-pooling layer. In this analogy, the active function in the convolution layer corresponds to the Q function. We can formulate the value iteration as:

$$y_t = max_a Q(s,a)$$
$$= max\left[R(s,a) + \sum_{t=1}^{N} P(s|s_{t-1},a)V_{t-1}\right] \quad (1)$$

where *Q(s,a)* indicates the value of action a under state s at t-th timestep, the reward $R(s,a)$ and discounted transition probabilities $P(s|s_{t-1},a)$ are obtained from G which mentioned in Section 2.1. N denotes the length of the sequence. Thus, the value of sequence V_n will be produced by applying the convolution layer recurrently several times according to the length of the sentence, and for a batch, n is valued between 1 and $batchsize$ of training. The optimal value $V_{update} = Average(V_1, ..., V_{batchsize})$ is the average long-term return possible from a state. The value of current predictions represents the cost of decoding at current state. We select the value of optimal pre-training model as the initial $V^*$and compare it with V_{update}. Subsequently, we observe the decoding effect of the current batch; thus, we can decide the necessity of taking the negative example as an input of D. The conditions of screening are as follows:

$$\begin{cases} direct\ input\ to\ D & if\ V_{update} < V^* \\ screening\ and\ V^* = V_{update} & if\ V_{update} > V^* \end{cases} \quad (2)$$

Since VIN is simply a form of CNN, once a VIN design is selected, implementing the screener is straightforward. The networks in the experiments all require only several lines of Tensor code.

2.3 Discriminator D

We implement D on the basis of CNN. The reason for this is that CNN has advantages in dealing with variable length sequences. The CNN padding is used to transform the sentences to sequences with fixed length. A source matrix $X_{1:N}$ and a target matrix $Y_{1:N}$ are created to represent $\{x_1, ..., x_{N_x}\}$ and $\left\{y_1^{'}, ..., y_{N_y}^{'}\right\}$. We concatenate every k dimensional word embedding into the final matrix $x_{1:N}$ and $y_{1:N}$ respectively. A kernel $w_j \epsilon R_{l\times k}$ applies a convolutional operation to a window size of l words to produce a series of feature maps:

$$c_{ji} = a(w_j \oplus X_{i:i+l-1} + b), \quad (3)$$

where b is a bias term and $\oplus$ is the summation of element production. We use $Relu$ as the function to implement the nonlinear activation function a. Then a max-pooling operation is leveraged over the feature maps:

$$c_{j\sim max} = max(c_{j_1}, ..., c_{j_{N-l+1}}). \quad (4)$$

For different window sizes, we set the corresponding kernel to extract the valid features, and then we concatenate them to form the source sentence representation c_x for D. And the target sentence representation c_y can be extracted from the target matrix $Y_{1:N}$. Then given the source sentence, the probability that the target sentence is being real can be computed as:

$$p_D = sigmoid(T[c_x; c_y]), \quad (5)$$

where T is the transform matrix which transforms the concatenation of c_x and c_y into a 2-dimension embedding. We can get the final probability if we use the matrix of 2-dimensional mapping as the input of the sigmoid function.

2.4 Training Process

We present a standard VS-GAN training process in the form of data flow directions (Fig. 2):

• *Pre-training G with RL algorithm.* Note that we pre-train G to ensure that an optimal parameter is directly involved in training, and provides a

good search space for beam search.

• *Observe the reinforcement reward.* Once the end of sentence (or the maximum sequence length) is reached, a cumulative reward matrix R is generated. The observed reward can measure the cumulative value of agent (G) in the prediction process (action) of a set of sequences.

• *Value screening.* The reward R is fed into a convolutional layer and a linear activation function. This layer corresponds to a particular action Q. The next-iteration value is then stacked with the reward and fed back into the convolutional layer N times, where N depends on the length of the sequence. Subsequently, a long-term value V_{update} is generated by decoding a sentence. The batch is screened by the set conditions, as shown in Eq.(2).

• *Stay awake.* D is dedicated to differentiating the screened negative result with the human-translated sentences, which provide the probability p_D.

• *Adversarial game.* When a win-win situation is achieved, adversarial training will converge to an optimal state. That is, G can generate confusing negative samples, and D has an efficient discrimination ability for negative and human translations. Thus, the training objective is as follows:

$$J_\theta = E_{(x,y)}[logp_D(x,y)] + E_{(x,y')}[log(1-p_D(x,y'))], \quad (6)$$

where (x, y) is the ground truth sentence pair, (x, y') is the sampled translation pair, as positive and negative training data respectively. $p_D(.,.)$ represents a probability which mentioned in D about the similarity. J_θ can be regard as a game process between maximum and minimum expectations. That is, the maximum expectation for the generation G, and the minimum expectation for D.

A common shortcoming of adversarial training in NLP applications is that it is non-trivial to design the training process, i.e., texts (Huszr, 2015). Given that the discretely sampled y' makes it difficult to back-propagate the *error signals* from D to G directly, making J_θ nondifferentiable w.r.t. $G's$ model parameters θ. To solve this problem, the Monte Carlo search under the policy of G is applied to sample the unknown tokens for the estimation of the $signals$. The objective of training G can be described as minimizing the following loss:

$$Loss = E_{(x,y')}[log(1 - p_D(x, y'))]. \quad (7)$$

We use $log(1 - p_D(x, y')$ as a Monte-Carlo estimation of the $signals$. By simple derivation, we can get the corresponding gradient of θ:

$$\frac{\partial_{Loss}}{\partial_{\theta\sim G}} = E_{y'}[log(1 - p_D(x, y'))\frac{\partial}{\partial_{\theta\sim G}} logG(y'|x)], \quad (8)$$

where $\frac{\partial}{\partial\theta\sim G} logG(y'|x)$ represents the gradients specified with parameters of the translation model based on RL. Therefore, the gradient update of parameters can be described as:

$$\theta_{\sim G} \leftarrow \theta_{\sim G} + l\frac{\partial}{\partial_{\theta\sim G}}, \quad (9)$$

where *l* is the learning rate, and we back propagate the gradient along negative direction. Note that we have not observed a high variance is accompanied by such a computation.

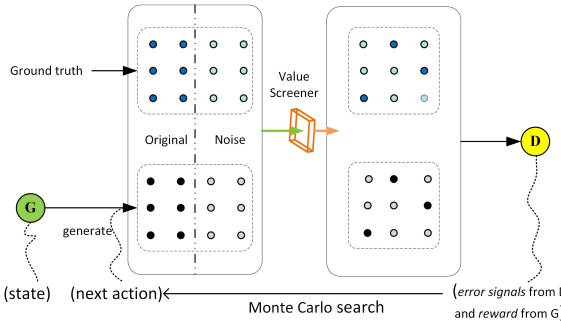

Figure 2: Presentation of VS-GAN model, in which different colors represent each component in VS-GAN.

3 Experiment and Analysis

3.1 Dataset and Noise Addition

We verify the effectiveness of our model on a language pair where one of the languages involved is agglutinative: Mongolian-Chinese(M-C). We use the data from CLDC and CWMT2017 evaluation campaign. To avoid allocating excessive training time on long sentences, all sentence pairs longer than 50 words either on the source or target side are discarded. Finally, by adding noise, we divide the training data of Mongolian into five categories[3]:{Original, BPE[4], Original&Suffixes, Original&Case, Original&Suffixes&Case}. For the target Chinese besides BPE processing, we adopt character granularity to provide a smaller unit corresponding to the morphological noise. Some effective work on morphological segmentation (Ataman D, 2017; ThuyLinh Nguyen, 2010) can be ap-

[3](Original&*) represents a mixed sample of the original data and the * segmentation of the original data.

[4]Note that BPE can only represent an open vocabulary through the variable-length character sequence, it is *insensitive* to morphological noise.

plied to agglutinative language. However, in order to be more specific and accurately, we perform independent-developed Mongolian segmenter. The final training corpus consists of about 230K original sentences (including 1000 validation and 1000 test) and corresponding pseudo-data sentences. We tried several num-operands of BPE[5] on the data set, and the final selection is: Mongolian: 35,000, Chinese: 15,000.

3.2 Experimental Setup

We select three strong baselines. Transformer presents an outstanding approach to most MT tasks. MIXER addresses exposure bias problem in traditional NMT well through RL, and BR-CSGAN is among the best endeavors to introduce the generative adversarial training into NMT.

The screening conditions mentioned in Section 2 enable the model to be trained efficiently. One problem is that under such conditions, V will gradually increase. Therefore, in the screening process, one situation should be considered, e.g., in $batch_1$ $\{V_1 = 1, ..., V_n = 10\}$, $V_{update} = 5.5$, in $batch_2$ $\{V_1 = 4, ..., V_n = 6\}$, $V_{update} = 5$. We have observed that $batch_1$ has worse sentences worth noting by D. However, because of the higher average value, the $batch_1$ will be screened out. In fact, we insist that such an operation is still reasonable, because the higher value batches occur only at the end of the training, and the n-gram natural of BLEU calculation indicates that the $batch_2$ needs more attention.

For the LSTM and MIXER, we set the dimension of word embedding as 512 and dropout rate as 0.1/0.1/0.3. We use a beam search with a beam size of 4 and length penalty of 0.6. For the Transformer, the Transformer_base configuration in (Vaswani et al., 2017) is an effective experience setting for our experiments. We set the G to generate 500 negative examples per iteration, and the number for Monte Carlo search is set as 20.

3.3 Main Results and Analysis

We mainly analyze the experimental results in three aspects: BLEU evaluation, the number of $< unk >$ symbols in the translations, and the time efficiency of model training.

• **BLEU** We use BLEU (Papineni et al., 2002) score as an evaluation metric to measure the similarity degree between the generation and the human translation.

For G, we select the model with 50 epochs of pre-training as the initial state, and 80 adversarial training epochs is used to joint train G and D. The results (Table 1) show that the GAN-based model is obviously superior to baseline systems in any kind of noisy corpus, and VS-GAN performs better than each baseline with average 2-3 BLEU. For the same model, the added noise provides the excellent generalization ability in testing, a notable result shows that VS-GAN improves 3.8 BLEU on the basis of the original corpus by adding both two kinds of noise. We notice that in the training of adding case noise only, the effect of VS-GAN is not outstanding. The reason for this is that the individual Mongolian-case is not obviously 'helpful' to the production of $< unk >$ symbol in Mongolian, so the screener is insensitive to it.

• **UNK** We count the number of $< unk >$ symbols in each system with 50 epochs of training to translate the source sentences (Fig. 3). For BR-CSGAN and VS-GAN, we directly count the number of $< unk >$ symbols in the negative example.

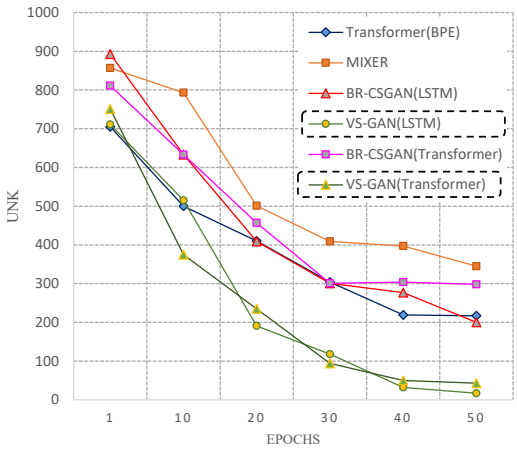

Figure 3: Number of $< unk >$ symbols in the translations of different models in each epoch.

In comparison with Transformer, MIXER optimizes BLEU through RL training, which can directly enhance the BLEU score of the translation. However, in terms of UNK, it is inefficient. The optimal initial state cannot be effectively maintained in the rest of the training (orange lines). We can see that the change of BLEU coincides with the change of UNK number in combination with Table 1 and Fig. 3. Furthermore, we note that UNK not only affects the accuracy of source word decoding, but also affects the semantic prediction of

[5] https://github.com/rsennrich/subword-nmt

Table 1: Training time consuming and BLEU score of systems under different noise modes. We stop the pre-training of G (including Transformer and LSTM) until the validation accuracy achieves at δ which is set to 0.6 in BR-CSGAN and VS-GAN. For the pre-training of D, we consider the threshold of classification accuracy and set it to 0.7.

	Original	BPE	Original&suffixes	Original&Case	Original&Suffixes&Case
MIXER(Ranzato et al., 2015)	29.7	26	30.4	28.6	31.3
BR-CSGAN (G:LSTM)(Zhen et al., 2018)	29.9|(15+17)h	30.8|(21+30)h	31.7|(22+25)h	31.1|(15+19)h	32.3|(27+32)h
VS-GAN (G:LSTM)	29.6|(15+11)h	31.5|(21+18)h	32.5|(22+19)h	30.8|(15+15)h	⋆***35.4***|(27+21)h
Transformer(Vaswani et al., 2017)	30.5	31.5	30.2	29.8	30.5
BR-CSGAN (G:Transformer)(Zhen et al., 2018)	27.4|(22+29)h	28|(27+34)h	28.9|(23+25)h	28.1|(22+30)h	32.1|(38+36)h
VS-GAN (G:Transformer)	28.8|(22+18)h	31.1|(27+26)h	32|(23+18)h	29.4|(22+18)h	***33.2***|(38+22)h

the entire sentence in translation.

• **Training Efficiency** In terms of training efficiency, we compared the two GAN-based models by counting the time of pre-training and adversarial training(italics in Table 1), (e.g., 15 + 17 indicates 15 h of pre-training and 17 h of adversarial training). Reinforcement pre-training is the same for BR-CSGAN and VS-GAN. In adversarial training, VS-GAN has a remarkable time reduction in each noise training strategy. This result depends on the screener for negative generations, so that D can regulate G, following UNK directly. Such combination of structures can converge to an optimal state rapidly. From the results in Table1, in the case of the two GANs the training time for the LSTM is shorter than for the Transformer. We attribute this to two reasons: *i*) the time consumed by LSTM is mainly used to explore long-distance dependencies in sequences. However, most of our corpus consists of short sentences ($<$*50* words), which bridges the gap between LSTM and Transformer and even exceeds Transformer (when it achieves the same accuracy of validation set). *ii*) in fact, according to our extensive experimental results on Mongolian-based NMT(including Mongolian-Chinese and Mongolian-Cyrillic Mongolian), Transformer usually converges slower than LSTM when the corpus size exceeds *0.2M*.

4 Conclusion

We propose a GAN model with an additional VIN approximation of value screener to solve the UNK problem in Mongolian→Chinese MT, which is caused by the change of suffixes or component cases in Mongolian and the limited vocabulary. In our experiment, we adopt the pretreatment method on the basis of noise addition to enhance the generalization ability of the model for UNK problem. Experimental results show that our approach surpasses the state-of-the-art results in a variety of noise-based training strategies and significantly saves training time. In future research, we will focus more on the combination of GAN and language features to enhance other agglutinative language NMT tasks, such as the guidance of syntax tree for GAN training. On the contrary, it is also a worthwhile attempt to modify the grammar tree constructed by adversarial training.

5 Acknowledgments

This study is supported by the Natural Science Foundation of Inner Mongolia (No.2018MS06005), and Mongolian Language Information Special Support Project of Inner Mongolia (No.MW-2018-MGYWXXH-302).

References

Turchi M et al. Ataman D, Negri M. 2017. Linguistically motivated vocabulary reduction for neural machine translation from turkish to english. *The Prague Bulletin of Mathematical Linguistics*, 108(1):331–342.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv*, 1409(0473).

D Bertsekas. 2012. Dynamic programming and optimal control. *Athena Scientific*, (04).

Xilun Chen, Yu Sun, and et al. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. In *Association for Computational Linguistics (ACL)*, pages 557–570.

P. Dayan and L F Abbott. 2003. Theoretical neuroscience: Computational and mathematical modelling of neural systems. *Journal of Cognitive Neuroscience*, 15(1):154–155.

Jonas Gehring, Michael Auli, David Grangier, and et al. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning(ICML)*, pages 1243–1252.

Alex Graves. 2013a. Generating sequences with recurrent neural networks. *arXiv preprint arXiv*, 1308(0850).

Alex Graves. 2013b. Generating sequences with recurrent neural networks. *arXiv preprint arXiv*, 1308(0850).

Caglar Gulcehre, Sungjin Ahn, and et al. 2016. Pointing the unknown words. In *Association for computational linguistics (ACL)*, pages 140–149.

Karl Moritz Hermann, Tom Kocisk, and et al. 2015. Teaching machines to read and comprehend. In *Neural Information Processing Systems (NIPS)*, pages 1693–1701.

Ferenc Huszr. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *Computer Science*, 1511(05101).

Sbastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *International joint conference on natural language processing (IJCNLP)*, pages 1–10.

Minh-Thang Luong, Ilya Sutskever, and et al. 2015. Addressing the rare word problem in neural machine translation. In *International joint conference on natural language processing (IJCNLP)*, pages 11–19.

Tom Mikolov, Stefan Kombrink, Luk Burget, and et al. 2011. Extensions of recurrent neural network language model. In *International conference on acoustics, speech, and signal processing*, pages 5528–5531.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, and et al. 2013. Playing atari with deep reinforcement learning. *arXiv*, 1312(5602).

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *International conference on artificial intelligence and statistics (Aistats)*, volume 5, pages 246–252.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics(ACL)*, pages 311–318.

Bellman R. 1957. Terminal control, time lags, and dynamic programming. *National Academy of Sciences of the United States of America*, 43(10):927–930.

Marc'Aurelio Ranzato, Sumit Chopra, and et al. 2015. Sequence level training with recurrent neural networks. *arXiv*, 1511(06732).

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Association for computational linguistics (ACL)*, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Conference and Workshop on Neural Information Processing Systems (NIPS)*, pages 3104–3112.

A Tamar, Y Wu, G Thomas, and et al. 2016. Value iteration networks. In *Neural Information Processing Systems (NIPS)*, pages 2154–2162.

Noah A.Smith. ThuyLinh Nguyen, Stephan Vogel. 2010. Nonparametric word segmentation for machine translation. In *Proceedings of the International Conference on Computational Linguistics, (COLING10)*, pages 815–823.

Ashish Vaswani, Noam Shazeer, and et al. 2017. Attention is all you need. In *Conference and Workshop on Neural Information Processing Systems (NIPS)*, pages 5998–6008.

L Wu, Y Xia, L Zhao, and et al. 2018. Adversarial neural machine translation. In *Asian Conference on Machine Learning (ACML)*, pages 374–385.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. In *The Association for the Advancement of Artificial Intelligence (AAAI)*, pages 2852–2858.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics*, 5(1):515–528.

Fan K et al. Zhang Y, Gan Z. 2017. Adversarial feature matching for text generation. In *Proceedings of the International conference on machine learning, (ICML17)*, pages 4006–4015.

Yang Zhen, Chen Wei, Wang Feng, and Xu Bo. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 1346–1355.