LSTM Encoder-Decoder with Adversarial Network for Text Generation from Keyword

Dongju Park and Chang Wook Ahn(✉)

School of Electrical Engineering and Computer Science,
Gwangju Institute of Science and Technology (GIST), Cheomdangwagi-Ro,
Buk-Gu, Gwangju 61005, Republic of Korea
cwan@gist.ac.kr

Abstract. Natural Language Generation (NLG), one of the areas of Natural Language Processing (NLP), is a difficult task, but it is also important because it applies to our lives. So far, there have been various approaches to text generation, but in recent years, approaches using artificial neural networks have been used extensively. We propose a model for generating sentences from keywords using Generative Adversarial Network (GAN) composed of a generator and a discriminator among these artificial neural networks. Specifically, the generator uses the Long Short-Term Memory (LSTM) Encoder-Decoder structure, and the discriminator uses the bi-directional LSTM with self-attention. Also, the keyword for input to the encoder of the generator is input together with two words similar to oneself. This method contributes to the creation of sentences containing words that have similar meanings to the keyword. In addition, the number of unique sentences generated increases and diversity can be increased. We evaluate our model with BLEU Score and loss value. As a result, we can see that our model improves the performance compared to the baseline model without an adversarial network.

Keywords: Text generation · Generative Adversarial Network
Natural Language Processing

1 Introduction

Automatically generating natural and contextual sentences is difficult, but it is important in Natural Language Processing (NLP) and Artificial Intelligence. For instance, Natural Language Generation (NLG) can be used when writing poems, novels, and letters. It also plays a significant role in NLP fields such as dialogue system [1], image captioning [2] and machine translation [3].

Recently, in the field of machine learning, Recurrent Neural Network Language Model (RNNLM) [4] is used for NLG work, and it is conceptually simple. However, RNN has a vanishing and exploding gradient issue that causes long-term dependency problems. The emergence of language model using Long Short-Term Memory (LSTM) [5], a variant of RNN, alleviates this problem. Nevertheless, these RNN variation based Language Models still have problems such as the *exposure bias* [6] due to the discrepancy between training and reasoning processes. The difference is that in the learning process, the next word is predicted by looking at the previous ground truth word, but in the inference process, it is generated by the previously deduced word.

Generative Adversarial Network (GAN) [7] is used as a solution to mitigate the problems mentioned above. In general, the GAN consists of two artificial neural network models that compete against each other in the learning process. One network is *Discriminator* and it is aimed to distinguish whether the given data is real data or synthetic data. The other is *Generator* that synthesizes fine quality generating data to fool the discriminator. During learning, the generator creates synthesized data similar to real data from the latent variable and uses it as training data of the discriminator. The discriminator uses real data and synthesized data from the generator to train and then uses the gradient of the learning loss as a guide to update the parameters of the generator. The GAN was first introduced to deal with the problems of continuous data such as image synthesis and computer vision task, and has been very successful. For example, DCGAN [8] incorporating the convolution layer enables the vector arithmetic operation performed by the generator learned in DCGAN, and it stabilizes the learning in most situations. In addition, StarGAN [9] is a single model that enables image-to-image translation for multiple domains.

On the other hand, text data is, unlike image data, discrete so that it is difficult to transfer a gradient from the discriminator to the generator. A recent study was able to deal with discrete data with SeqGAN [10] using reinforcement learning to resolve the problem. Thereafter, various models for discrete data emerged, including RankGAN [11] for learning by using the relative ranking scores of data and LeakGAN [12] for long text generation.

In this paper, we propose a model for generating text based on a given word. Our work contributes to widening the diversity of text generated. In order to increase the diversity of the text generated from a single word, this model finds a word near to a given word and learns it together. In addition, this method generates a sentence containing words obtained from the main word but not the main word among the input words. The model consists of SeqGAN based LSTM Encoder-Decoder and we use GAN together to improve learning efficiency.

2 LSTM Encoder-Decoder with Adversarial Network

The basic structure of the proposed model consists of a generator that generates realistic text based on a given word, and a discriminator that can distinguish between generated and real data.

2.1 Word Embedding

Text data such as $(w_1, \cdots, w_T)$ consisting of T words are divided into words and then converted into real number vectors by skip-gram [13] method, learned by neural networks, a kind of Word2Vec. These words are embedded in the user-defined N dimensions, and the similarity between words can be measured by the distances among the words. In the proposed model, skip-gram is used to select two words with high similarity to a given word.

2.2 Generator

We choose to use the RNN Encoder-Decoder [14] architecture as the generator to generate text with the given word as conditional information. Also, our generator uses LSTM instead of RNN for both encoder and decoder. The encoder reads the input words, such as a sequence of vectors $(x_1, \cdots, x_T)$, to form a context vector c by the last hidden state of the LSTM. The decoder is trained and inferenced with the context vector c given by the encoder and all the formerly words $(y_1, \cdots, y_{T'})$ it to predict the next word. That is, it can be expressed as follows:

$$p(y_1, \cdots, y_{T'} | x_1, \cdots, x_T) = \prod_{t=1}^{T'} p(y_t | c, y_1, \cdots, y_{t-1}) \tag{1}$$

In the case of the input data of the encoder, one word given for learning and reasoning, two words similar to the word are added, and a total of three words are used as the input data of the encoder. The decoder starts with the last hidden state of the encoder and <START_TOKEN> . If the sentence consists of 20 or fewer words, it is filled with <PAD> (see Fig. 1).

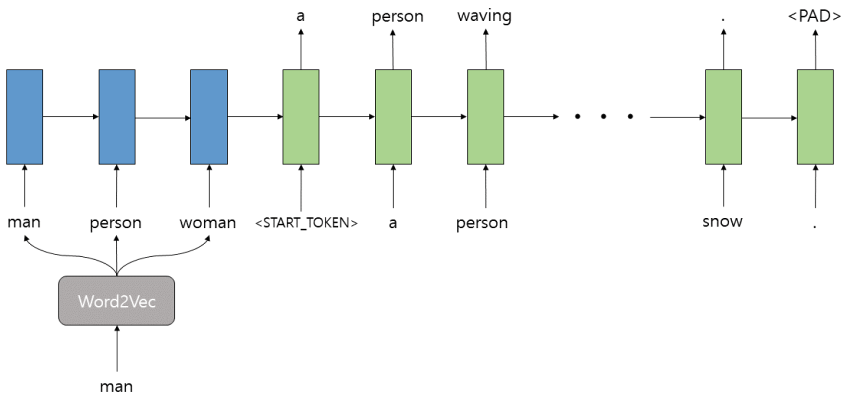

Fig. 1. The illustration of our generator model. The blue rectangles represent the LSTM layer of the encoder and the green are the LSTM layer of the decoder. <man, person, woman> are input data, and <a, person, waving, …, snow,.> are output data. (Color figure online)

2.3 Discriminator

The discriminator is an adversarial network of RNN Encoder-Decoder (generator). We use the bi-directional LSTM using self-attention [15] with one forward layer and one backward layer instead of the Convolutional Neural Network (CNN) used for text classification [16] in SeqGAN. In this model, bi-directional LSTM, hidden states H to be used in self-attention is obtained by concatenating hidden states from forward LSTM layers and backward LSTM layers. After, H is multiplied by the annotation matrix A, and a new sentence embedding matrix M is constructed as

$$M = AH. \tag{2}$$

Subsequently, it passes through fully-connected layers to determine whether it is real or fake text (see Fig. 2)

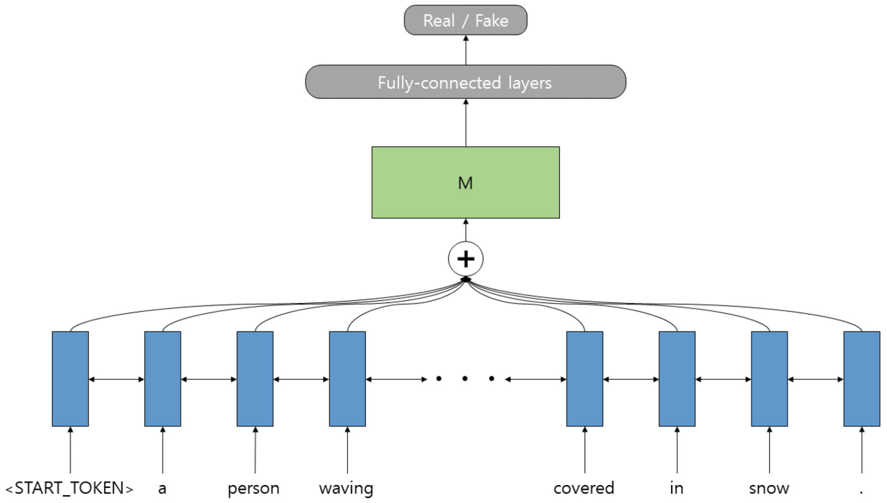

Fig. 2. The illustration of our discriminator model. The blue rectangles represent the hidden layers of the bidirectional LSTM. Circle with + denotes annotation matrix, and the green square is a sentence embedding matrix. In addition, the softmax function layer is used in the final layer to distinguish between real and fake texts. (Color figure online)

2.4 Policy Gradient Training

Discrete data such as text cannot be learned by the generator directly from the discriminator's gradient. Thus, most models that deal with discrete data in GAN use the REINFORCE algorithm as the policy gradient [17]. The training method of the proposed model is similar to the learning method of SeqGAN which learns by REINFORCE algorithm that includes Monte Carlo Tree Search with the encoder part added.

The objective of the generator model is to generate text that maximizes reward. The objective function of the policy gradient is as follows:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum\nolimits_{s \in S} d(s) \sum\nolimits_{a \in A} \pi_\theta(s,a) Q(s,a) \tag{3}$$

r is the reward, s is the state, $d(s)$ is the probability of reaching the state, a is the action, π is the policy, Q is the action-state value, and θ is policy parameter. The gradient of Eq. (3) can be defined as:

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum\nolimits_{s\in S} d(s) \sum\nolimits_{a\in A} \pi_\theta(s,a) \nabla log\pi_\theta Q(s,a) \\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)Q(s,a)]\end{aligned} \tag{4}$$

We can use this gradient to update the parameters of the generator as follows:

$$\theta \leftarrow \theta + \alpha_k \nabla_\theta J(\theta) \tag{5}$$

where α_k is the learning rate at k-step.

3 Experiments

3.1 Datasets

We use COCO Dataset [18] as data for learning and evaluating the proposed model. The datasets are made up of image and text pairs of data for image captioning. In these data pairs, 20000 texts consisting of 15 to 20 words are set as training sets and 3000 as evaluation sets. Also, we set a keyword for each sentence. For convenience, we choose the first noun in the sentence and pair each sentence. In total 7428 words are used for training and evaluation, and a word dictionary is created after embedding in a vector space using a skip-gram.

3.2 Evaluation Measure

BLEU Score [19] is used to evaluate our model and Baseline model as LSTM Encoder-Decoder using only Maximum Likelihood Estimation (MLE). The BLEU Score is originally designed for machine translation, but we use it to compare the similarity of the generated sentence with the ground-truth sentence. The BLEU score calculation method is as follows:

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{\left(1-\frac{r}{c}\right)} & \text{if } c \leq r \end{cases} \tag{6}$$

Then,

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum\nolimits_{n=1}^{N} w_n log p_n\right) \tag{7}$$

The BP in Eq. (6) is used as a Brevity Penalty to eliminate the case where the higher the sentence length is, the higher the score is. r is the length of the sentence in the dataset, and c is the length of the generated sentence by the model. Equation (7) denotes the final BLEU score, where w_n is the weight for n-grams and p_n represents the precision for the corresponding gram.

We calculate the BLEU Score from 2-gram to 4-gram based on the evaluation dataset

3.3 Training Setting

First, we embed all the words in the dataset into the vector space using the skip-gram model. They are used not only as word embedding vectors for the generator and the discriminator but also for finding words similar to the keyword. Then, pre-train the generator and discriminator before implementing the adversarial training. When learning the generator, for convenience, select the first noun in a given sentence and designate it as a keyword, and use Word2Vec to get the two words that are used as input to the encoder along with the keyword. The target data is a sentence extracted from the keyword. In the case of the discriminator, it learns to classify by using the generator-synthesized sentence and ground-truth data. Sometimes the generator generates new data for the discriminator learning. Thereafter, adversarial training is carried out with two pre-training models.

4 Experimental Results

Our experimental results are divided into three. The first is the number of unique sentences. Second, we compare the loss of our model trained with GAN and the baseline model learned only with MLE. Finally, the sentences generated from the two models used above are evaluated based on the BLEU Score.

4.1 Number of Unique Sentences

The result of comparing the number of unique sentences according to the number of input words, using the second and third words obtained from the keyword together with the keyword makes more unique sentences than learning a single word. In addition, the use of three words with shuffle is better than its exclusion. In other words, this method can be said that the diversity of generated sentences is widened as the result of comparing the number of unique sentences among the generated sentences.

Furthermore, when shuffling three words for learning and reasoning, those words appear similar in generated sentences, and the sentences that contain no words are also slightly reduced with this method. In contrast, when learning with one word and with three words, the first word is overwhelming, and the second and third words extracted from the first word using Word2Vec are rarely shown. (We do not include it if it comes with the first word). Tables 1 and 2 show the experimental results.

Table 1. The number of unique sentences among the total 20000 sentences generated by the number of input words and the rate of increase in the number of the unique sentences compared to one word.

Number of input words	Number of unique sentences	Rate of increase
One word	15385	–
Three words	16418	6.71%
Three words (Shuffle)	**17713**	**15.13%**

Table 2. Percentage of words in sentences generated by the number of words entered

Number of input words	First word (keyword)	Second word	Third word	None
One word	85.24%	0.22%	0.12%	14.42%
Three words	84.85%	0.41%	0.27%	14.47%
Three words (Shuffle)	**31.07%**	**28.70%**	**26.54%**	**13.69%**

4.2 Learning Curves

We compare the baseline model that we learn using only our model and MLE with Negative Log-Likelihood (NLL). The baseline model learns 250 epochs only with MLE, and our model pre-training 150 epochs before adversarial training. As a result, the two models are similar to each other up to 150 epochs, but after that, we confirm that our model shows the lower losses than the baseline model (see Fig. 3).

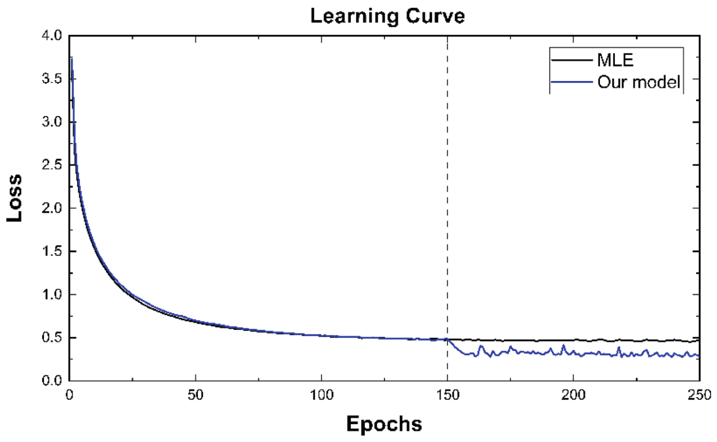

Fig. 3. The learning curve of the two models. The vertical dash line is the end of the pre-training of the proposed model.

4.3 BLEU Score

We measure the BLEU Score (2-gram, 3-gram, and 4-gram) using the test data. The measurement results show that the model using the GAN has the higher score. Table 3 shows the results.

Table 3. The BLEU Score of two models

Method	BLEU-2	BLEU-3	BLEU-4
MLE	0.7958	0.6120	0.4359
Our model	**0.8182**	**0.6255**	**0.4378**

In addition, Table 4 shows the result of the generated text according to the input words. A bold word is a word contained in the generated sentence.

Table 4. Example of the generated text according to the input words.

Input word	Generated text
People/ **men**/women	- Two **men** playing a video game in a park bench with a book to it
Restroom/urinals/ **toilet**	- A **toilet** and sink next to each other by the small window in the bathroom
Birthday/ **cake**/chocolate	- A **cake** with a side of a cake lit candles while sliced leaves on a cake
Fruits/ vegetables/stir	- Two **fruits**, one of them on a plate with a knife and fork
Cheese/pizza/ **broccoli**	- **Broccoli**, tomato, cucumber, onion and meat in a plastic container on top of an office
Doubles/apartment/ **streets**	- Two red **streets** sign stand on a wall, a lamp, has a lamp base
Racket/ball/**tennis**	- A **tennis** player wearing a blue outfit stretches up high with his **racket** on a tennis court
Business/ **balcony**/towels	- A **balcony** with chairs and a table is seen through a glass door

5 Conclusion

In this paper, we introduced an LSTM Encoder-Decoder model with the adversarial network for text generation from the keyword. Experimental results show that our model using GAN has the higher BLEU score and the lower loss than the MLE model alone. We also found that when we find words similar to keywords, shuffle them with keywords, and use them as input data, we increase the variety of sentences generated. Furthermore, by using this method, not only the sentences containing the keyword but also the sentences containing the similar words obtained from the keyword are generated. For future work, we plan to expand the model so that the relationship between the input words can be entered together, or the model can be inferred directly so that it will work well to generate various sentences we want.

Acknowledgements. This work was supported by Global University Project (GUP) grant funded by the GIST in 2018. Also, this work was supported by the NRF funded by MEST of Korea (No. 2015R1D1A1A02062017).

References

1. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A.C., Pineau, J.: Building end-to-end dialogue systems using generative hierarchical neural network models. In: AAAI, pp. 3776–3784 (2016)
2. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: International Conference on Machine Learning, pp. 2048–2057 (2015)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
4. Mikolov, T., Karafiát, M., Burget, L., Černocký, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association (2010)
5. Sundermeyer, M., Schlüter, R., Ney, H.: LSTM neural networks for language modeling. In: Thirteenth annual Conference of the International Speech Communication Association (2012)
6. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. In: Advances in Neural Information Processing Systems, pp. 1171–1179 (2015)
7. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
8. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
9. Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., Choo, J.: StarGAN: unified generative adversarial networks for multi-domain image-to-image translation. arXiv preprint arXiv:1711.09020 (2017)
10. Yu, L., Zhang, W., Wang, J., Yu, Y.: SeqGAN: sequence generative adversarial nets with policy gradient. In: AAAI, pp. 2852–2858 (2015)
11. Lin, K., Li, D., He, X., Zhang, Z., Sun, M.-T.: Adversarial ranking for language generation. In: Advances in Neural Information Processing Systems, pp. 3155–3165 (2017)
12. Guo, J., Lu, S., Cai, H., Zhang, W., Yu, Y., Wang, J.: Long text generation via adversarial training with leaked information. arXiv preprint arXiv:1709.08624 (2017)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
14. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
15. Lin, Z., et al.: A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017)
16. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
17. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**, 229–256 (1992)
18. Chen, X., et al.: Microsoft COCO captions: data collection and evaluation server. arXiv preprint arXiv:1504.00325 (2015)
19. Papineni, K., Roukos, S., Ward, T., Zhu, W.-J.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 311–318. Association for Computational Linguistics (2002)