Received March 26, 2020, accepted April 30, 2020, date of publication May 11, 2020, date of current version June 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993928

FGGAN: Feature-Guiding Generative Adversarial Networks for Text Generation

YANG YANG, XIAODONG DAN, XUESONG QIU, AND ZHIPENG GAO
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Yang Yang (yyang@bupt.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2103200, in part by the Open Subject Funds of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory under Grant SKX182010049, in part by the Fundamental Research Funds for the Central Universities under Grant 500419319 2019PTB-019, and in part by the Industrial Internet Innovation and Development Project 2018 of China.

ABSTRACT Text generation is a basic work of natural language processing, which plays an important role in dialogue system and intelligent translation. As a kind of deep learning framework, Generative Adversarial Networks (GAN) has been widely used in text generation. In combination with reinforcement learning, GAN uses the output of discriminator as reward signal of reinforcement learning to guide generator training, but the reward signal is a scalar and the guidance is weak. This paper proposes a text generation model named Feature-Guiding Generative Adversarial Networks (FGGAN). To solve the problem of insufficient feedback guidance from the discriminator network, FGGAN uses a feature guidance module to extract text features from the discriminator network, convert them into feature guidance vectors and feed them into the generator network for guidance. In addition, sampling is required to complete the sequence before feeding it into the discriminator to get feedback signal in text generation. However, the randomness and insufficiency of the sampling method lead to poor quality of generated text. This paper formulates text semantic rules to restrict the token of the next time step in the sequence generation process and remove semantically unreasonable tokens to improve the quality of generated text. Finally, text generation experiments are performed on different datasets and the results verify the effectiveness and superiority of FGGAN.

INDEX TERMS Generative adversarial networks, text generation, deep learning, reinforcement learning.

I. INTRODUCTION

Generative Adversarial Networks (GAN) [1] has gradually developed into a hot research field in deep learning since it was proposed. As a generative model, its main application field is image generation, but GAN also has a high research potential in text generation. Poetry writing, dialogue system and intelligent translation are all based on text generation. Although some progress has been made, the quality of generated text is often poor or limited to specific areas and lack of generality.

The combination of GAN and reinforcement learning (RL) for text sequence generation has become one of the hotspots of research. Sequence GAN (SeqGAN) proposed by Yu *et al.* [2] for the first time feeds discriminator output as reward signals of reinforcement learning to generators as decision guidance for generating sequences. GAN combined with reinforcement learning has achieved remarkable results in the field of text generation, but the structure has the following drawbacks:

1) The probability that the discriminator outputs positive and negative samples is used as a reward signal in reinforcement learning. The feedback signal is a scalar and cannot preserve the high-level semantic information of the text. The generator lacks a clear training direction.
2) Sampling is required to complete the sequence and get feedback signal through discriminator in the text sequence generation process. Due to the limitation of sampling times, the sampling process is highly random and inadequate, and may result in semantically unreasonable sequences such as subject repetition and verb deletion.

Based on the current problems of text generation, we propose a text generation algorithm based on the Feature-Guiding Generative Adversarial Networks (FGGAN). The improved algorithm will effectively solve the shortcomings of the existing text generation model. The main contributions of this paper are as follows:

The associate editor coordinating the review of this manuscript and approving it for publication was Nizam Uddin Ahamed.

1) Because the output probability of the discriminator is scalar, the training direction of the generator is not clear. In this paper, a feature guidance module is improved to transform the high-order text feature extracted from the discriminator into the generator network for feedback guidance. The feedback signal is transformed into a guidance vector with more guidance information to improve the generation of sequences by the generator.
2) Because of the randomness and inadequacy in the process of sequence sampling, it is possible to generate non semantic tokens in the process of sequence generation. This paper proposes a method to create a vocabulary mask based on semantic rules, which restricts the tokens generated in the next time step during the sequence generation. Remove the candidate tokens with low correlation with the generated prefix sequence to make the generated sequence is more realistic.

II. RELATED WORK

Text generation is a task of simulating and generating sequence data. Most text generation tasks are based on Recurrent Neural Network (RNN). The Long Short-Term Memory (LSTM) proposed by Hochreiter *et al.* [3] has been widely used. Wen *et al.* [4] used LSTM to build a natural language dialogue system. In 2014, Goodfellow proposed the generative adversarial networks. Compared with the single generation model, GAN has a more significant effect in data generation. GAN uses generator and discriminator to train and optimize in the adversarial way and finally reaches the state of Nash equilibrium to effectively learn the data distribution.

GAN has been successfully applied in the field of images to generate realistic images. Chang *et al.* [5] proposed KGGAN by setting up multiple generators, one of which is responsible for learning the information in a priori knowledge field and directing the learned knowledge to another generator to generate a variety of image data. But the disadvantages are that a priori domain knowledge is needed to assist and the learned diversity is not easily accepted by the discriminator. Lian *et al.* [6] proposed FG-SRGAN for high-resolution image generation, mainly by setting up a guidance module to learn the mapping from low-resolution image to high-resolution image, to improve the quality of the generated image by the generator.

However, due to the discreteness of text data, the original GAN could not optimize the generator parameters based on gradient backpropagation. Martin *et al.* [7] conducted some analysis on the training methods of GAN such as using the Wasserstein divergence instead of the traditional JS divergence to train GAN in the field of text generation. In addition, Che *et al.* [8] proposed the maximum-likelihood augmented discrete GAN (MailGAN) and designed training techniques to directly calculate the difference between the generated data distribution and the real data distribution. Zhang *et al.* [9] completed the training by adjusting the generator to make the generated samples have the same characteristics as the real samples. Su *et al.* [10] have made some achievements in dialogue generation using GAN combined with a hierarchical recurrent encoder-decoder. Fedus *et al.* [11] introduced an actor-critic conditional GAN and produced more realistic text samples.

GAN can be combined with reinforcement learning. The generator can be seen as a decision maker and the network can be optimized using a policy gradient. Reinforcement learning requires reward signals as feedback. At each time step, complete sequences can be sampled using Monte Carlo Tree Search (MC Search) [12] and fed into the discriminator to get reward signal. Yu *et al.* proposed SeqGAN, which is the first time to use discriminator as reward for reinforcement learning. Nie *et al.* [13] proposed Relational GAN (RelGAN), which uses Gumbel-Softmax to train GAN on discrete data and multiple embedded representations in the discriminator to provide a more informative signal. Lin *et al.* [14] proposed RankGAN which replaces the original binary classifier with a sorting model based on cosine similarity to make the feedback of the discriminator more continuous. However, these models still have the disadvantage that the output of the discriminator as a feedback signal is scalar and has weak guidance. In addition, the randomness of the sampling process may cause the network not to learn the implicit semantic information, resulting in the unrealistic generated data. In this paper, FGGAN is proposed for the shortcomings of the existing model and the effectiveness of the improved module is verified by experiments.

III. ALGORITHM

This paper adopts the overall framework of GAN and puts forward improvement aiming at the existing defects. Firstly, the output scalar of the discriminator in the correlation algorithm is used as the feedback signal, which leads to the unclear training direction of the generator. In this paper, a feature guidance module is improved to transform the high-order text feature extracted from the discriminator into the generator network for feedback guidance. In addition, this paper proposes a method to create a vocabulary mask based on semantic rules, which restricts the tokens generated in the next time step during the sequence generation. The improved model is named Feature-guiding GAN, and the overall network structure is shown in Fig. 1.

The overall structure of the GAN is divided into a generator G_θ and a discriminator D_ϕ. The objective of the generator is to find the parameters of the optimal distribution probability of the data. However, the parameter update does not originate from the data sample, but from the back-propagation gradient of the discriminator. The model is trained via the adversarial strategy. Also, the generator and the discriminator are alternately optimized.

The generator on the left activity box in Fig.1 can be regarded as performing a text generation task. The objective of the generator G is to predict the next token based on the

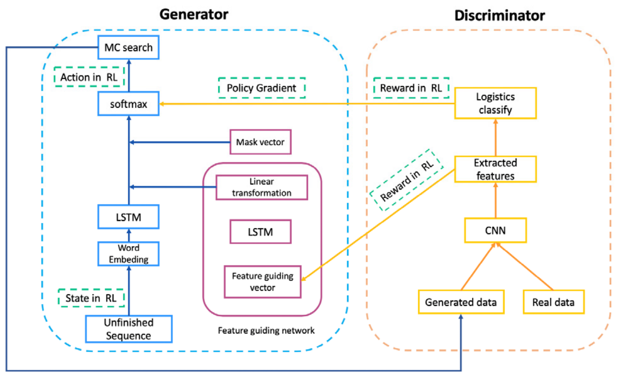


FIGURE 1. **The overall network structure of Feature-guiding GAN**

generated sequence. Assuming a sequence of tokens, where y_{i} represents a token, y represents the vocabulary. Generator is mainly divided into text generation module marked blue in Fig.1 and feature guidance module marked purple in Fig.1, both of which adopt LSTM structure. Combining the improved feature guiding module (Section III-A for details) and the mask vector (Section III-B for details), the generation equations in time step t can be shown in (1) -(3):

$$h_t = R_{\theta_R}(h_{t-1}, x_t) \tag{1}$$

$$P(\cdot|x_1, \ldots, x_t) = z_t(h_t) = \mathrm{softmax}(W(h_t w_t Mask_T) + c) \tag{2}$$

$$y_t \sim p(\cdot|x_1, \ldots, x_t) \tag{3}$$

R_{θ_R} is the set of generator parameters, h_{t-1} is the hidden state of the previous step, x_t represents the input vector of the current time step. The calculated current step hidden state h_t is a y-dimensional vector. h_t combines the guide vector w_t generated by feature guidance module and mask vector $Mask_T$ generated by semantic rule module to pass through a softmax layer, calculates the probability distribution z_t of the tokens and samples a token y_t.

The discriminator on the right activity box in Fig.1 uses the Convolutional Neural Network (CNN). After vectoring the input text, it uses convolutional kernels of different sizes to extract the text features and transforms the extracted features into the generator for more detailed sequence generation guidance. The discriminator is trained by the negative samples generated by the generator and the positive samples in the real data set, which is essentially a binary classification task. For each sample x, the discriminator outputs the probability $D_\phi(x)$ of whether x is the real data and sends it back to the generator as the reward in reinforcement learning. The optimization objective of the discriminator is shown in (4), as follows:

$$\max_{\phi} E_{Y\sim P_{data}}[\log D_\phi(Y)] + E_{Y\sim G_\theta}[\log(1 - D_\phi(Y))] \tag{4}$$

Because of the discretization of the text sequence, the generator cannot solve the gradient of the objective function to D_ϕ. For this problem, the reinforcement learning theory can be combined with the sequence generation process as a series of action selection. The action of each step is to select the next token, i.e. $a_i = y_t$, and the state of each step is a prefix sequence composed of generated tokens, i.e. $s_t = y_1, \ldots, y_{t-1}$. After the last step, the complete sequence is sent to the discriminator and the output probability is used as the reward of the generator. The goal of the generator is to maximize the reward expectation. The optimization objective of the generator is shown in (5), as follows:

$$\max_{\theta} E_{Y_{1:T}\sim G_\theta}[\log D_\phi(Y_{1:T})] \tag{5}$$

where $\log D_\phi(Y_{1:T})$ is the reward of generator The strategy of reinforcement learning is expressed as $G_\theta(a|s) = P(a|s;\theta)$, which specifies the probability of selecting an action under state s. The action reward value function is denoted as $Q^\theta(s, a)$, which indicates the total reward expectation obtained according to the strategy after action a in state s. The reward expectation of the Tth token is shown in (6), as follows:

$$Q^\theta(s = Y_{1:T-1}, a = y_T) = \log D_\phi(Y_{1:T}) \tag{6}$$

The total reward expectation is shown in (7), as follows:

$$E_{Y_{1:T}\sim G_\theta}[Q^\theta(Y_{1:T-1}, y_T)]$$
$$= \sum_{y_1} G_\theta(y_1|s_0) \ldots \sum_{y_T} G_\theta(y_T|Y_{1:T-1}) Q^\theta(Y_{1:T-1}, y_T) \tag{7}$$

The discriminator can only judge the complete sequence, which means that the current generated token cannot be evaluated in the sequence generation. The granularity of evaluating the generation quality is too large when generating a complete sequence, which is equivalent to treating every token in the sequence the same. To obtain better results, the Monte Carlo search method is used to complete the current generated sequence at the intermediate time. In the time step t, the unknown last $T - t$ tokens can be sampled by the Monte Carlo search and a K-time MC search can be shown in (8):

$$\{Y^1_{1:T}, \ldots, Y^K_{1:T}\} = MC(Y_{1:T}; K) \tag{8}$$

The purpose of MC search is to evaluate the corresponding reward with the generated prefix sequence for each new generated token. Random sampling is conducted through the generator to continue to complete the incomplete sequence and the sequence is sent to the discriminator to judge the score. The scores that are obtained via multiple sampling are averaged to obtain the reward expectation of the current token and to reduce the granularity of reward guidance to the level of tokens. The calculation method can be shown in (9):

$$Q^\theta(s = Y_{1:\mathrm{t}-1}, a = y_{\mathrm{t}})$$
$$= \begin{Bmatrix} \frac{1}{K}\sum_{k=1}^{K} \log D_\phi(Y^k_{1:T}),\ Y^{\mathrm{k}}_{1:\mathrm{T}} \in MC(Y_{1:T}; K) & \text{for t<T} \\ \log D_\phi(Y_{1:t}) & \text{for t=T} \end{Bmatrix} \tag{9}$$

where K is the times of sampling when calculating the reward expectation for each time step. We finally set $K = 4$ in the experiment. The increase of sampling times will increase the time complexity of the whole network.

A. FEATURE GUIDING NETWORK

In the process of generating sequence token by token, the generator needs the output of the discriminator as the reward signal of reinforcement learning. However, the comparison algorithm such as SeqGAN directly transmits the probability value as the reward to the generator. The generator only uses this scalar to optimize the network training. The sequence can only be optimized the training direction through a large number of samplings and the limited number of sampling is also very small for a complete spatial distribution. The gradient of the discriminator D_ϕ back to the generator is also very small during training. These problems make the generator unable to optimize the network parameters effectively. In view of the weak guidance of the discriminator feedback signal, we add a feature guidance module in the generator network. The purpose of the feature guidance module is to obtain more abundant text features from the discriminator to guide the generation.

The reward signal that is fed back to the generator by the discriminator is a scalar and does not contain much semantic information. The discriminator is a trained convolutional neural network. The input data to be discriminated is first processed by word embedding layer, then the semantic features of text vectors are extracted by convolution operations of different sizes in convolution layer. Convolutional neural networks typically connect to the pooling layer to reduce the dimension after the convolutional layer. However, the difference between positive and negative samples of the text is often not on the macro level and it is necessary to consider the differences in detail. Therefore, we do not adopt the pooling method. Instead, it is replaced by a convolutional layer with the step size equal to 1. It is subsequently sent to the output layer through the fully connected layer. The output layer uses the sigmoid function. The overall network structure of the discriminator can be shown in (10):

$$D_\phi(x) = sigmoid(\phi_o^T F(x)) \tag{10}$$

where ϕ is the parameter of the discriminator network, ϕ_o represents the parameter of the output layer which is the last layer of the discriminator. F represents all parameters of the network except the output layer and this part can be regarded as a feature extraction network. This output result can be used as the intermediate output to the output layer of the discriminator and can also be sent to the feature guidance module as the input. Since F is the output vector of the feature extraction by the CNN, relative to the scalar of the discriminator output, F contains more detailed semantic information. This vector can represent the mapping of input samples to higher order feature spaces. It can provide more detailed guidance for generator optimization. The feature vector is shown in Fig. 2.

The feature vectors extracted by the convolution layer cannot be directly connected to the input of each time step in the text generation network. The feature vector is extracted for the purpose of discriminator distinguishing positive and negative samples, but each time step of text generation network is to generate residual sequence. Direct access may lead

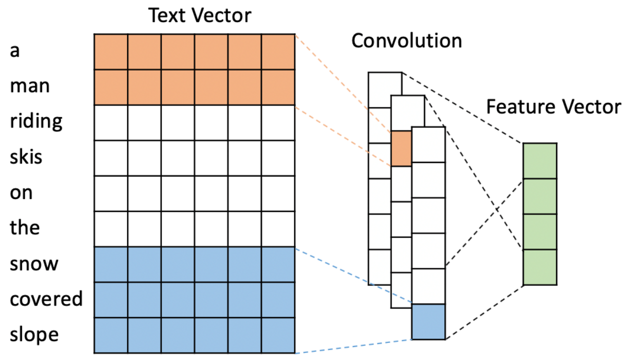


FIGURE 2. The feature vector extracted from the convolutional layer.

to uncoordinated network training, so a conversion module is needed for further processing. In this paper, the network is named the feature-guiding module.

The feature vector passed in by the discriminator is recorded as f_t and continues to use an LSTM network for transformation, keeping synchronization with the text generation module.

The network structure can be shown in (11), as follows:

$$g_t, h_t^C = C_{\theta_C}(f_t, h_{t-1}^C) \tag{11}$$

where C denotes a feature-guiding network, θ_C denotes the parameter set of C, h_t denotes a hidden state vector of the current time step. The feature-guiding vector of the current time step g_t is jointly determined by the feature vector f_t of the current time step and the hidden state vector h_{t-1}^C of the previous time step. To maintain the stability of the network training, g_t can be normalized into a unit vector.

Because the dimension of the feature-guiding vector is inconsistent with the vector that is generated by each time step of the text generation module, linear conversion is also necessary. Considering the stability of the network, the transformation can be combined with the feature vector of the latest k-step. In this paper, $k = 4$ is finally selected. The process can be shown in (12), as follows:

$$w_t = W \sum_{i=1}^{k} g_{t-i} \tag{12}$$

where W represents the dimensional transformation matrix and w_t represents the dimensionally transformed vector. w_t and the vector generated by text generation module jointly determine the generation of the next token. The equation of text generation module is can be shown in (13).

$$h_t^r = R_{\theta_R}(X_t, h_{t-1}^r) \tag{13}$$

where R_{θ_R} represents the text generation module and X_t represents the sequence representation vector of the current time step. One dimension of X_t is vocabulary size and the other dimension is the same as w_t. Through matrix multiplication, we can get a vector of vocabulary size, which represents the probability that each token in vocabulary is selected. Then, the next token of the sequence is selected by the softmax layer until the end. The probability distribution of the next word z_t can be shown in (14), as follows:

$$z_t = \text{softmax}(X_t w_t) \tag{14}$$

B. TEXT SEMANTIC RULES FOR RESTRICTING GENERATION

Sequence generation is the process of token by token generation. The intermediate time sequence is not generated completely, but it still needs to evaluate the reward of the current generated token in this time step. Because the discriminator can only accept the input of the complete sequence and the Monte Carlo sampling can supplement the incomplete sequence, a large number of sampling operations are needed to fill the incomplete sequence in the intermediate time. Then the complete sequence is fed into the discriminator to determine the reward of the current token and the subsequent generation is carried out according to the feedback guidance. Due to the limitation of sampling times, the process of Monte Carlo sampling cannot be fully traversed in the complete vocabulary space, which has a high randomness. There may be generating sequences with unreasonable semantics, such as subject repetition and verb deletion.

This paper proposes a method to create a vocabulary mask based on semantic rules, which restricts the tokens generated in the next time step during the sequence generation. The specific method is to preprocess the real dataset according to the semantic rules to get the corresponding mask vector of each token in the vocabulary. The mask vector represents the relationship between tokens, such as part of speech, similarity. It represents the tokens that should be restricted in the subsequent generation when the token appears in the current step. The dimension of the mask vector is equal to the dimension of the vocabulary. If the parameter value on the dimension is 0, it means that the token corresponding to the dimension has a very low correlation with the generated prefix sequence and subsequent generation should mask the token. The purpose of mask vector is to eliminate the candidate tokens that do not conform to the objective context in the generation, so that the generation process can learn the implicit semantic structure and improve the quality of the final generation sequence.

Assuming that the vocabulary size is n and the sequence length is m, the sampling space is of size n^m. In the current time step, if the prefix sequence has been generated, the sampling space size is still the vocabulary size n. However, there are many alternatives that do not conform to the current semantics. These tokens should be restricted to subsequent generation. Although the text sequence has a complex semantic structure, in the case of the generated prefix sequence, the frequency of occurrence of subsequent tokens has a non-random probability distribution. We define rules in terms of the word similarity. Each token in the text sequence can be encoded into a word vector in the word embedding layer. Word2vec [15] is a related model that is used to generate

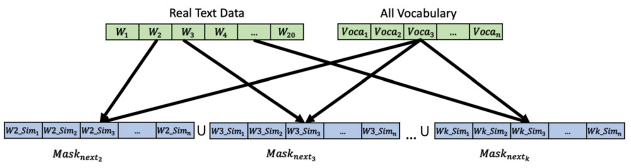

Real Text Data
All Vocabulary
W_1 W_2 W_3 W_4 ... W_{20}
$Voca_1$ $Voca_2$ $Voca_3$ ... $Voca_n$
$W2\_Sim_1$ $W2\_Sim_2$ $W2\_Sim_3$ ... $W2\_Sim_n$ ∪ $W3\_Sim_1$ $W3\_Sim_2$ $W3\_Sim_3$ ... $W3\_Sim_n$ ... ∪ Wk_Sim_1 Wk_Sim_2 Wk_Sim_3 ... Wk_Sim_n
$Mask_{next_2}$
$Mask_{next_3}$
$Mask_{next_k}$

FIGURE 3. The mask vector that corresponds to each token.

the word vector. The model is a shallow two-layer neural network that is used to train to reconstruct the token. After the training has been completed, the word2vec model can be used to map each token to a vector, which can be used to represent the relationships between tokens. This paper uses the cosine distance to characterize the similarity between two tokens. Each token in vocabulary is traversed to find out the tokens with high subsequent relevance in real dataset, so as to restrict in the subsequent generation process. The current token is $word_i$ and the cosine distances are calculated between $word_{next}$, which denotes the k tokens that appear next in the real data sequence, and all tokens in the vocabulary. For every token in the vocabulary, If k-times calculation similarity is less than threshold Th_{sim}, set the value to 0 in the corresponding mask vector, otherwise, set it to 1. Thus, the vector $Mask_i$ that corresponds to each token is obtained. The vector can be shown in Fig. 3 and equations (15)-(15).

$$Wk_{sim_n} = 0 \text{ if } Sim(W_k, Voca_n) < Th_{sim} \text{ else } 1 \tag{15}$$

$$Mask_i = \bigcup_{j=1}^{k} Mask_{next_j} \tag{16}$$

Through the experimental test, the parameter k is set to 5, Th_{sim} is set to 0.6. The vector similarity formula can be shown in (17), as follows:

$$similarity = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{17}$$

where n is the word vector dimension. When generation at each time step, consider the m tokens that are generated by the prefix sequence. The masks are $Mask_i$, $i \in [1, m]$. The final mask is ORed by the mask vectors of the latest m tokens at the current time step, which can be shown in (18).

$$Mask_T = \bigcup_{i=1}^{m} Mask_{T-i} \tag{18}$$

Through the experimental test, the parameter m is set to 4. Finally, $Mask_T$ is used to mask the token selection probability of the current time step. Combined with the mask vector $Mask_T$ and the feature-guiding vector w_t in the feature guidance module, the current time step vector X_t in the text generation module is sent to the softmax layer to determine the next token generation. The probability of the next token can be shown in equation (19) and Fig. 4.

$$z_t = \text{softmax}(X_t w_t Mask_T) \tag{19}$$

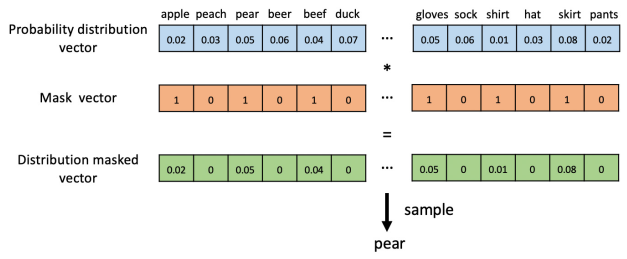


FIGURE 4. Sampling through the mask vector.

C. NETWORK TRAINING

In the early stage of the GAN, the generator will encounter suboptimal model convergence. Because the underlying parameters are too random, it is impossible to generate higher quality text, which causes the feedback to be a large penalty value, which leads to slow convergence of the model. Therefore, the maximum likelihood pre-training method is used to reduce the risk. In the pre-training process, the generator network uses the cross-entropy as the loss function to quickly optimize the generator parameters.

The improved model is still composed of a generator and a discriminator, where in the generator includes a feature-guiding network and a text generation network, both of which are composed of LSTM. The discriminator is composed of a CNN. Except for the last layer of the output layer classification, all layers are extracted as feature extraction network. The text generation network can use supervised pre-training to accelerate the convergence of the model before the whole system begins training to avoid the model collapse.

In the overall training, the method of adversarial training is still adopted. When training the discriminator, the supervised training method is used. The real data are regarded as the positive sample and the sequence that is generated by the generator is regarded as the negative sample. Also, the cross-entropy loss function is utilized. When training the generator, the parameters of the discriminator are fixed and the generator network adopts the token-by-token generation method. The prefix sequence is oversampled to obtain the complete sequence and sent to the discriminator to obtain the reward value. We optimize generator parameters in a gradient descent way to get the highest reward.

IV. EXPERIMENTS

To evaluate the performance of the design, an experiment[1] with three parts was designed: The first part of the experiment uses synthetic data for the text generation test. The training data comes from the well-trained LSTM model G_{oracle}[2] disclosed in the paper of SeqGAN, which randomly initializes the LSTM network parameters and randomly generates sequences with length of 20 as the real dataset. The effect of generator G_θ learning data distribution and generating

[1]the code is available on the https://github.com/danxiaodong/FGGAN
[2]the code is available on the https://github.com/LantaoYu/SeqGAN

TABLE 1. Convolution layer parameters.

(window size, kernel numbers)	(1, 100),(2, 200),(3, 200),(4, 200),(5, 200) (6, 100),(7, 100),(8, 100),(9, 100),(10, 100) (15, 160),(20, 160)

sequence is judged by the negative log-likelihood (NLL) of G_{oracle}. The second part of the experiment uses the COCO dataset [16] as the real text data for the experiment. The COCO dataset is a dataset that the Microsoft team can use for image recognition, which has a manual text description of each image, and it is used as a real dataset. In the third part, the experiment was conducted using a Chinese poetry dataset in which 8,000 Chinese Tang poems were selected as datasets. Each poem contains five or seven Chinese words per sentence. The evaluation of real data experiment is bilingual evaluation understudy (BLEU) [17]. The BLEU score is used to compare and count the number of commonly occurring n-ary words for the quality evaluation of the generated text.

Using the above datasets for comparison experiments, MLE trained LSTM, SeqGAN, RankGAN, RelGAN and FGGAN proposed in this paper are used for text generation comparison tests. The text generation experiments that are conducted in this paper all use free generation by inputting a random character as the starting character. Finally, the quality of the generated text is judged by calculating the corresponding evaluation.

Generator in FGGAN mainly includes text generation module and feature guidance module, both of which adopt LSTM structure. In text generation module, it is specified that the sequence generation length is 20, the dimension of the input embedding layer is 128 and the dimension of the hidden layer is 128. The feature guidance module uses the text feature vector extracted from the discriminator as the input and generates the feature guidance vector with dimension 16. The guidance vector and the output of text generation module are transformed to determine the generation of the next token. The discriminator uses CNN structure for feature extraction and classification output. The dimension of the input embedding layer is 256. The subsequent convolutional layer parameters are shown in the following Table 1 and the convolution step is set to 1. The dimension of text feature vector extracted by convolutional layer is 1720, which is transferred to feature guidance module for processing.

L2 regularization with weight parameter of 0.2 is added on the basis of cross entropy loss function and dropout is set to 0.75 to avoid over fitting. In the adversarial training of generator and discriminator, the ratio of generator and discriminator training times is 1:3, that is, generator training once and discriminator training three times.

In the synthetic data experiment, initialize the LSTM Model G_{oracle} and generate 10000 sentences as the real data. Both FGGAN and comparison algorithm use this dataset for text generation. The NLL function of G_{oracle} is used as the evaluation of data generation effect after each batch of data is

TABLE 2. Experimental results of NLL on synthetic data.

Model	SeqGan	RankGan	RelGAN	FGGAN	Ground Truth
NLL	0.237	0.271	0.323	0.359	0.575

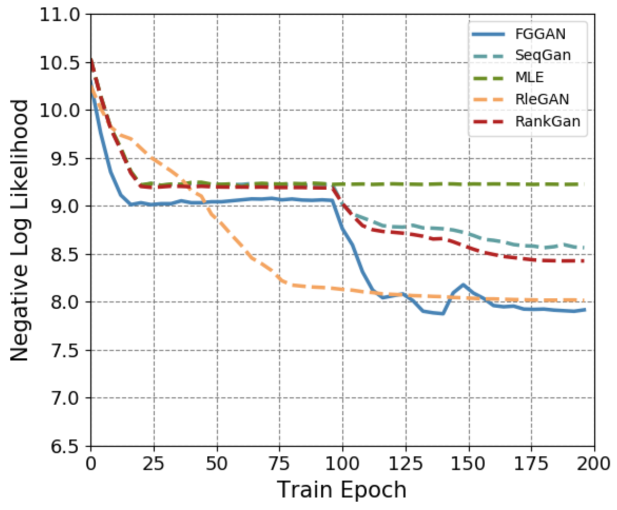


FIGURE 5. Synthetic data experimental training curves.

generated. NLL objectively represents the ability of learning data distribution of the model. The equation is shown in (20), as follows:

$$NLL = E_{Y_{1:T} \sim G_\theta}[\sum_{t=1}^{T} \log G_{oracle}(y_t|Y_{1:t-1})] \quad (20)$$

Each model uses random noise to generate data. The effect comparison is shown in the Fig 5. In the pretraining, the FGGAN model has shown a more advantageous effect on the evaluation compared with other models. After entering the adversarial stage, the FGGAN convergence speed is better than other models and the final NLL value is closer to the real data, as shown in the Table 2.

In the COCO real data experiment, using the image description sequence of dataset as the target of generation. The dataset includes 19,383 words and 198,751 sentences. This experiment preprocesses the dataset by deleting words with frequencies of less than 10 and sentences containing them. The processed dataset includes 5126 words. 45000 sentences were randomly selected as the training set and 5000 sentences were used as the test set. The BLEU scores are presented as following Table 3 and Fig. 6. FGGAN training loss on BLEU is shown in Fig. 7. The results demonstrate that the FGGAN model outperforms the compared models.

In order to verify the enhancement effect of the improved module proposed in this paper, we use the COCO dataset for comparative experiments and the network settings are consistent with the above.The feature guidance moule and the semantic rule module are deleted as the baseline. The

TABLE 3. COCO dataset BLEU score comparison.

	SeqGan	RankGan	RelGAN	FGGAN	Ground Truth
BLEU-2	0.734	0.748	0.765	0.773	0.932
BLEU-3	0.519	0.532	0.548	0.559	0.781
BLEU-4	0.371	0.394	0.406	0.417	0.653
BLEU-5	0.237	0.271	0.323	0.359	0.575

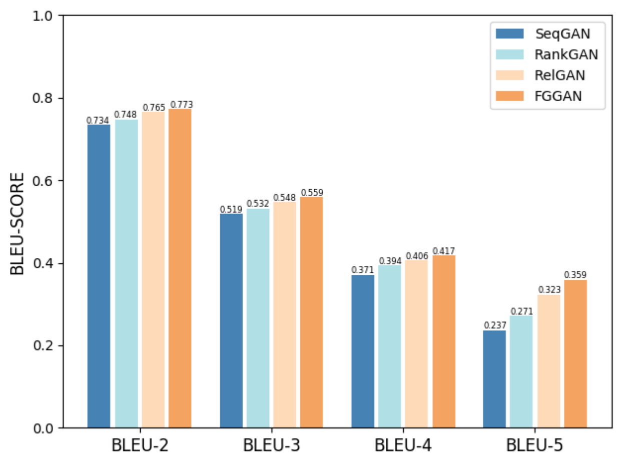


FIGURE 6. BLEU on the COCO dataset.

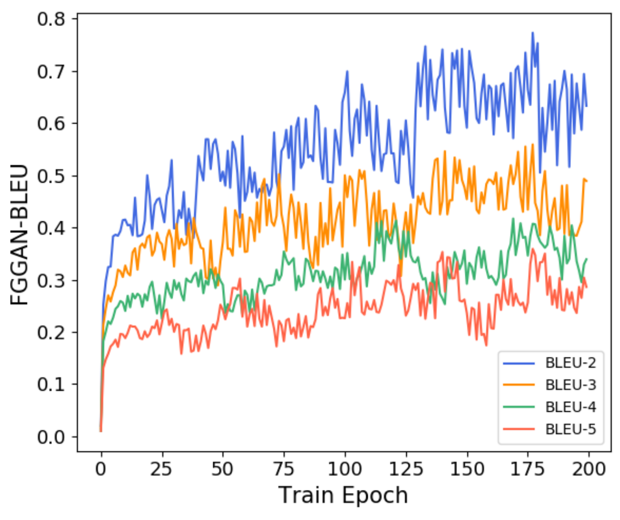


FIGURE 7. FGGAN training loss on BLEU.

effect after adding the improved module is compared and the effect scores of each model are shown in the Fig. 8. It can be seen that the improved module in this paper has improved the effect of the baseline model.

The comparison of real data and generated data on the COCO dataset is shown in table 4.

To evaluate the improved model that is proposed in this paper, text generation of Chinese poetry was also conducted and 8000 Chinese Tang poems were selected as data sets.

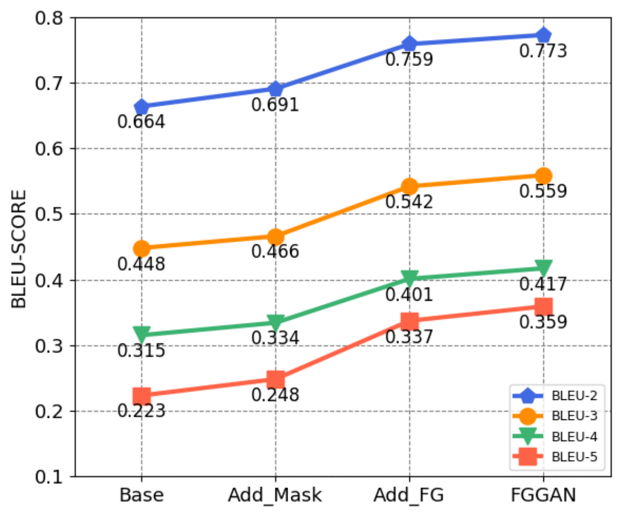


FIGURE 8. The improvement of feature guidance module and semantic rule module.

TABLE 4. Comparison of real data and generated data on the COCO dataset.

Generated Data	A group of people sitting on top of a snow-covered slope. A plate of broccoli and fried egg on a white plate. A baseball player looks at the ball while the pitcher throws the ball A man riding a skateboard in the middle of a street. A person riding skis skiing down a snow-covered mountain. A skateboarder is doing a trick in front of a skateboard. A young man riding on top of a white skateboard. A baseball player is on a baseball field. A baseball player holding a bat on a baseball field. A young boy is holding a baseball bat on a baseball field. A pile of bananas sitting on top of a cutting board. A man is holding up a banana to his nose. A baseball player with a bat on top of his knees. A plate of food with beef, broccoli, and potatoes A man wearing a helmet and riding a skateboard.
Real Data	A person skiing down the snow-covered ground as others look on. A skateboarder performing a trick in the air at the edge of a cliff. A skateboarder goes down a hill with a skateboard. A man doing a trick on a skateboard at a park A person skiing in the snow going down a snow-covered mountain. A man is riding a skateboard on top of a cement block. A man is riding a skateboard in the middle of a cement park. A man riding a skateboard with a skateboard in the street. A plate of food with grilled chicken and broccoli. A group of people riding skis on a snow-covered slope. A man holds his ski poles as he walks down the street.

Each poem contains five or seven Chinese words per sentence. Among them, 4000 poems are used as training sets, 4000 are used as test sets and BLEU-2 is used as the result judgement standard. The scores are shown in Table 5 and Fig. 9. The improved model outperforms other models in simple poetry generation.

TABLE 5. Chinese poetry dataset BLEU score comparison.

Model	SeqGan	RankGan	RelGAN	FGGAN	Ground Truth
BLEU-2	0.624	0.677	0.695	0.713	0.764

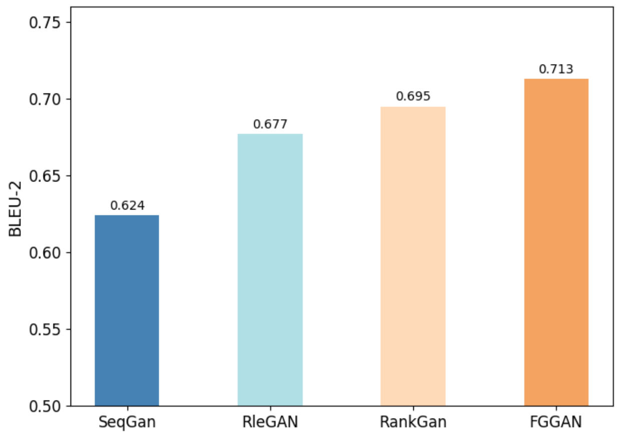


FIGURE 9. BLEU on the Chinese poetry dataset.

V. CONCLUSION

For text sequence generation, this paper proposes an improved framework FGGAN. In order to solve the problem that the feedback signal of the discriminator is not very instructive, this paper proposes a feature guidance module which obtains the text semantic feature feedback to the generator with more guidance. In addition, this paper proposes a method to create a vocabulary mask based on semantic rules which restricts the tokens during the generation to make the sequence more realistic. The superiority of the improved module is evaluated experimentally. In the synthetic experiment, the negative log-likelihood is used for the evaluation. FGGAN proposed in this paper has higher ability of fitting data distribution. In the experiment on real data, BLEU is used for the evaluation. Compared with other models, FGGAN has a higher evaluation score and generates more realistic text data.

VI. DISCUSS

Compared with other comparison algorithms, FGGAN in this paper has some improvements in the relevant dataset, but there are still some problems that need further improvement. Firstly, the feature guidance module extracts the text features from the discriminator and sends them to the text generation module for guidance after transformation. However, because the linear transformation may not be able to adapt to the high-speed changing feature space of the discriminator CNN, the text generation module may not learn advanced semantic features. Further research on extraction and transformation of text feature is needed in the future. Secondly, in the process of using semantic rules to restrict sampling, the mask vector is obtained by preprocessing the dataset according to the semantic rule. However, too complex rule restriction will cause the neural network to mode collapse, resulting in some duplicate text sequences. Subsequent work related to semantic rule optimization needs to be carried out.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, New York, NY, USA, 2014, pp. 2672–2680.

[2] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI*, San Francisco, CA, USA, 2017, pp. 2852–2858.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] T. H. Wen, M. Gasic, and N. Mrksic, "Semantically conditioned LSTM-based natural language generation for spoken dialogue systems," *Comput. Sci.*, vol. 3, no. 17, pp. 144–152, 2015.

[5] C.-H. Chang, C.-H. Yu, S.-Y. Chen, and E. Y. Chang, "KG-GAN: Knowledge-guided generative adversarial networks," 2019, *arXiv:1905.12261*. [Online]. Available: http://arxiv.org/abs/1905.12261

[6] S. Lian, H. Zhou, and Y. Sun, "FG-SRGAN: A feature-guided super-resolution generative adversarial network for unpaired image super-resolution," in *Proc. ISNN*, Moscow, Russia, 2019, pp. 151–161.

[7] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *Proc. ICLR*, Toulon, France, 2017, pp. 124–131.

[8] T. Che, Y. Li, R. Zhang, R. Devon Hjelm, W. Li, Y. Song, and Y. Bengio, "Maximum-likelihood augmented discrete generative adversarial networks," 2017, *arXiv:1702.07983*. [Online]. Available: http://arxiv.org/abs/1702.07983

[9] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," in *Proc. ICML*, Sydney, NSW, Australia, 2017, pp. 4006–4015.

[10] H. Su, X. Shen, P. Hu, W. Li, and Y. Chen, "Dialogue generation with GAN," in *Proc. AAAI*, New Orleans, LA, USA, 2018, pp. 8163–8164.

[11] W. Fedus, I. Goodfellow, and A. M. Dai, "MaskGAN: Better text generation via filling in the," in *Proc. ICLR*, Vancouver, BC, Canada, 2018, pp. 1–17.

[12] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[13] C. Zhang, C. Xiong, and L. Wang, "A research on generative adversarial networks applied to text generation," in *Proc. 14th Int. Conf. Comput. Sci. Edu. (ICCSE)*, New Orleans, LA, USA, Aug. 2019, pp. 1268–1288.

[14] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *Proc. NIPS*, Long Beach, CA, USA, 2017, pp. 3155–3165.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781

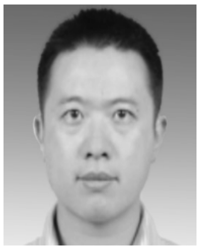
[16] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick, "Microsoft COCO captions: Data collection and evaluation server," 2015, *arXiv:1504.00325*. [Online]. Available: http://arxiv.org/abs/1504.00325

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, Philadelphia, PA, USA, 2002, pp. 311–318.

YANG YANG was born in 1981. She received the Ph.D. degree from the Beijing University of Posts and Telecommunications, in 2011. She is currently an Associate Professor. More than 30 articles have been published in SCI/EI journals and one international standard have been applied. Meanwhile, she is the paper review experts of journals of *Sensors*, the *International Journal of Distributed Sensor Networks*, and ICC conference. Her current research interests are big data analysis and trend depth prediction. She is the Session Chair of CENET2018 International Conference Internet of Things and big data analysis.

XIAODONG DAN was born in 1994. He received the bachelor's degree from Xidian University, in 2017. He is currently pursuing the master's degree in computer science wit the Institute of Network Technology, Beijing University of Posts and Telecommunications. His main research interests include big data analysis and machine learning.

XUESONG QIU was born in 1973. He received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2000. He is currently a Professor and the Ph.D. Supervisor. He has authored about 100 SCI/EI index articles. He presides over a series of key research projects on network and service management, including the projects supported by the National Natural Science Foundation and the National High-Tech Research and Development Program of China. He received 13 national and provincial scientific and technical awards, including the national scientific and technical awards (second-class) twice.



ZHIPENG GAO was born in 1980. He received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2007. He is currently a Professor and the Ph.D. Supervisor. He presides over a series of key research projects on network and service management, including the projects supported by the National Natural Science Foundation and the National High-Tech Research and Development Program of China. He received eight provincial scientific and technical awards.