IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 3, 2019, accepted July 24, 2019, date of publication August 12, 2019, date of current version September 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2934581

QuGAN: Quasi Generative Adversarial Network for Tibetan Question Answering Corpus Generation

YUAN SUN, CHAOFAN CHEN, TIANCI XIA, AND XIAOBING ZHAO

School of Information Engineering, Minzu University of China, Beijing 100081, China
Minority Language Branch, National Language Resource and Monitoring Research Center, Minzu University of China, Beijing 100081, China

Corresponding author: Yuan Sun (tracy.yuan.sun@gmail.com)

This work was supported by the National Natural Science Foundation of China under Grant 61972436, Grant 61501529, and Grant 61331013.

ABSTRACT In recent years, the large-scale open Chinese and English question answering (QA) corpora have provided important support for the application of deep learning in the Chinese and English QA systems. However, for low-resource languages, such as Tibetan, it is difficult to construct satisfactory QA systems, owing to the lack of large-scale Tibetan QA corpora. To solve this problem, this paper proposes a QA corpus generation model, called QuGAN. This model combines Quasi-Recurrent Neural Networks and Reinforcement Learning. The Quasi-Recurrent Neural Networks model is used as a generator for Generative Adversarial Network, which speeds up the generation of text. At the same time, the reward strategy and Monte Carlo search strategy are optimized to effectively update the generator network. Finally, we use the Bidirectional Encoder Representations from Transformers model to correct the generated questions at the grammatical level. The experimental results show that our model can generate a certain amount of effective Tibetan QA corpus, and the BLEU-2 value increases by 13.07% than baseline. Moreover, the speed of the model has been greatly improved.

INDEX TERMS Quasi generative adversarial network, Tibetan, QA corpus generation, reinforcement learning, Monte Carlo search strategy.

I. INTRODUCTION

Faced with the explosive growth of Web content, traditional information retrieval methods based on keywords are failed to meet the needs of users. Users need more efficient and accurate information. In 2011, the University of Washington Professor Oren Etzioni mentioned in Nature that ''web search is on the cusp of a profound change — from simple document retrieval to question answering'' [1].

Recently, the application of deep learning to QA systems has become a hot topic [2]–[13]. However, deep learning algorithms need to be based on a large-scale QA corpus. In Chinese and English QA systems, there are large-scale public QA corpora, such as Google's Natural Questions [14], Facebook's SimpleQuestions [15], Microsoft's WikiQA [16], and Baidu Chinese question and answer dataset - WebQA [17], etc. These corpora provide important support for deep learning applied to Chinese and English QA systems. On the other hand, for low-resource languages, such as Tibetan, it's hard to construct a satisfactory QA system, owing to the lack of large-scale Tibetan QA corpora.

However, comparing with English and Chinese, there are some challenges when generating Tibetan QA corpus. Tibetan is a language with strong grammar rules and has complex syntactic structure. The polysemy of the word appears in Tibetan more frequent, which makes word hard to represent. In addition, Tibetan sentences are often much longer than Chinese and English sentences in the same expression, which makes the model hard to train. Complex syntactic structure makes it not easy to generate coherent and accurate sentences.

In view of these problems, this paper proposes Quasi Generative Adversarial Network (QuGAN) model which can generate a large amount of accurate Tibetan QA corpus. This model combines Quasi-Recurrent Neural Networks (QRNN) [18] and Reinforcement Learning (RL), and optimizes the reward strategy and Monte Carlo search strategy.

The associate editor coordinating the review of this article and approving it for publication was Muhammad Afzal.

The main contributions of this paper are as follows:

(1) To speed up the Tibetan text generation, this paper uses QRNN model as the generator. QRNN model combines the advantages of Long Short Memory Network (LSTM) and Convolutional Neural Network (CNN), which can handle long sequence problems in Tibetan, and the data can be calculated in parallel. Furthermore, to reduce the difference of the probability distribution between the generated data and the real data, this paper uses the Maximum Likelihood Estimation (MLE) to initialize the random questions.

(2) The traditional reinforcement learning uses the Monte Carlo search algorithm to score the entire generated text sequence, so it takes a long time to convergence the Generative Adversarial Networks (GAN). It is no longer to suitable for Tibetan which is often composed of long sentences. Therefore, we optimize the Monte Carlo search algorithm by predicting the scores of next sequences according to generated partial sequences. We also optimize the reward strategy, and effectively update the generator network.

(3) Since the grammatical rules and syntactic structure of Tibetan are complex, this paper uses the Tibetan text corpus to train the Bidirectional Encoder Representations from Transformers (BERT) model, and optimizes the grammatical level of the generated sentence, making the question more accurate and coherent.

II. RELATED WORK

At present, Chinese and English have built a large number of QA corpora. For English corpora, such as SQuAD [19], SimpleQuestion [15] and so on. SQuAD dataset was built by manual extraction of questions from text paragraphs, then given answers. SimpleQuestion is a factual dataset which was generated factual by Freebase [20]. For Chinese corpora, such as WebQA [17], which built by Baidu Knows. InsuranceQA [21] which questions were proposed by users and the high-quality answers were provided by professionals. These corpora are mainly based on manual annotation, and it takes a lot of time and manpower. Therefore, many researchers start to construct virtual question and answer pairs.

One method is template-based method, it uses the knowledge base [22]–[23] or text paragraphs to extract questions and answers through manual extraction, and converts them into natural language questions. Curto *et al*. [24] proposed an automatic question generation algorithm based on template matching. In his work, the existing question and answer pairs are used as the corpus of template extraction, and then the search engine queries the question and answer pairs with the same template to expand corpus and verify the correctness of generated templates. Rashmi and Joshi [25] proposed a method to generate why-questions. They established question and answer pairs by using the causal relations annotated in the Penn Discourse Treebank (PDTB). QALD [26] and FREE917 [27] are based on the knowledge base to generate QA corpus, in which QALD mainly indicates the construction question should start from the category and then cluster the questions. Wang *et al*. [28] used closed fields to generate question and answer pairs. By logically representing the triples in the knowledge base, they transformed the form of closure into questions. But the quality of questions is not guaranteed, the homogeneity of the questions is serious.

Template-based methods cannot avoid to generate homogenous questions. The deep learning method learns information features from the knowledge base or text paragraph through neural networks, and then generates questions. Bauer*et al*. [29] proposed a model for answering these generated questions on the Narrative QA [30] dataset. They found that many sample data cannot be reasoned and given answers through the text information. So they introduced common sense information (in external knowledge base) into the conventional machine reading comprehension model. Rao and Daumé [31] constructed a neural network model for automatically generating questions based on the full information expectation, and they used the QA website to construct a QA dataset which contains triplet information. Serban *et al*. [32] directly used the encoder-decoder model to convert triples into problems. Zhou *et al*. [33] proposed a method for generating problems from texts, which uses features to encode the answer position information to generate questions. These methods require large-scale QA corpora and are supported by external knowledge base. However, they are difficult to be applied directly to low-resource languages, such as Tibetan.

In 2017, considering performance and efficiency, Bradbury *et al*. [18] proposed a hybrid structure QRNN combining the CNN model with the RNN layer, QRNN can parallel calculate based on time step and minibatch like CNN, which ensures high throughput and good length scalability for sequence data. It has the characteristics of past time dependence as well. The work has obvious effects on dealing with sequence data. In 2018, the University of Washington proposed an Embedding from Language Models (ELMo) for deeply extracting semantic features [34]. To solve the problem of lacking the intrinsic connection of continuous text and the ability of language structure expression in word vector, the attention mechanism is used to calculate the relationship between single word and words in the sentence, then it adjusts the weight of the word to obtain the new word expression. The model has a good effect in dealing with the complexity and ambiguity of the word. Google released the BERT model in 2018, they used the Transformer to encoder and got highly performance on natural language processing tasks [35].

In 2014, Goodfellow *et al*. [36] proposed GAN and applied it to computer vision image generation. In 2018, the BigGAN developed by Andrew Brock of Heriot-Watt University and DeepMind team got 166 points on Inception Score (IS) for image generation (233 points for real pictures) [37]. In terms of text processing, the main difficulty of GAN application is that the original GAN is mainly applied to real number space which is continuous data, while the text data is discrete data. On the other hand, the error will be along with the length of the sentence exponential accumulation

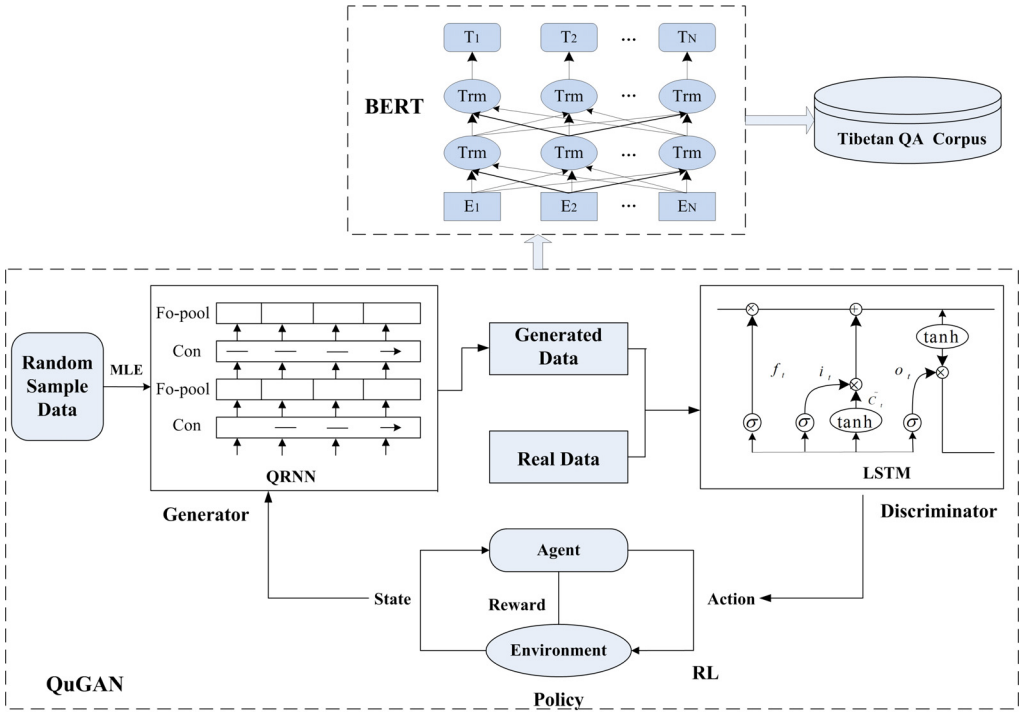

FIGURE 1. The framework of QuGAN model.

when generating text. To solve these problems, in 2016, Zhang *et al*. [38] attempted to apply GAN to the text generation task. They use LSTM as the generator, and CNN as the discriminator. Also, the original sentence and the new sentence which obtained by exchanging the positions of the two words in the sentence are used to train. Yu *et al*. [39] proposed the SeqGAN which uses LSTM as the generator and CNN as the discriminator, and the error as a reward for reinforcement learning. It trains in a feed-forward manner, and updates the generator network with an exploration mode. The BLEU value of the Chinese verse generation reached to 0.7389, and the Obama speech generation reached to 0.427. The above generator can only reward or punish the generated sequence by indirectly passing the feedback the value generated by the discriminator, and cannot directly obtain the information from the sequence. In order to solve this problem, Li *et al*. [40] used the adversarial training method. It uses the seq2seq model as the generator, the discriminator adopts hierarchical decoding, and joint training generator and discriminator by reinforcement learning. To generate an open dialogue, they revised rules in the process of updating the generator. In 2018, Fedus *et al*. [41] proposed the MaskGAN to solve the model collapse and the instability of the training as the sentence length increases in the GAN model. The model uses the actor-critic algorithm in reinforcement learning to train the generator, using the maximum likelihood estimation and stochastic gradient descent to train the discriminator, which guarantees the quality of the generated text. These researches are good explorations of GAN in natural language processing. However, the problems of not being able to effectively pass the gradient to the generation network and generating sentence with grammatical confusion still exist.

III. MODEL ARCHITECTURE

In order to construct the Tibetan QA corpus automatically, this paper proposes the QuGAN model. Firstly, the random text sequences are initialized, and then they are sent to the generator of the QRNN network for automatic text generation. The discriminator uses LSTM model to judge whether the generated text is true. The reinforcement learning is used for feedback, and the optimized Monte Carlo search algorithm is used to accelerate the training of the model. Finally, the BERT model is used to correct the questions at the grammatical level. The framework is shown in Fig 1.

The model consists of five main parts:

(1) Initialization: we randomly generate a certain amount of sample data by using characters in database. To reduce the difference in the probability distribution of the generated data and the real data, we initialize generate random sample data using the Maximum Likelihood Estimation.

(2) Generator G: to accelerate the convergence speed of the GAN model and to support the data parallelizable operation, we use QRNN model as the generator for such strong grammatical rules language-Tibetan, QRNN can save a lot of time.

(3) Discriminator D: the LSTM model is used as the discriminator to judge whether it is real data.

(4) Reward strategy: to make the generated data more realistic, the reinforcement learning is used for feedback. The possible subsequent sequences are obtained by the optimized

Monte Carlo search algorithm, and then we searched for the residual sequence word by word (this process called action). At the same time, aiming at the characteristics of Tibetan language, we optimize the reward strategy to find the best strategy that can get the maximum reward. Then the discriminator scores the generated sequence in whole sentences and feeds the score back to the generator (this process called reward), finally updates the current state of the generator (this process called state).

(5) Grammar optimization: to make the generated questions more coherent, the BERT model is used to modify the Tibetan sentences, because the grammatical rules and syntactic structure of Tibetan are complex.

IV. MODEL DETAILS

A. GENERATOR

Before the model training, we perform MLE on randomly sample data to generate questions more efficiently and define initialized sequence $T = (t_1, t_2, t_3 \cdots t_n)$.The maximum likelihood estimates to derive the maximum probability text sequence $\tilde{T}$ is shown in equation (1).

$$\tilde{T} = arg\max_{T} \prod_{i=1, t_i \epsilon T}^{n} P(t_i) \tag{1}$$

where t_i is a single character or word, T is the entire text sequence as the input of the generator.

We use QRNN model as a generator. QRNN combines LSTM with CNN to handle the problems of long sequence and data cannot be parallelized. The QRNN mainly consists of two components: convolution component and pooling component. It can be broken down into two subcomponents on the network structure. The first layer is a convolutional layer for extracting input features, in which parallel processing can be performed based on the sequence dimension for the processed sequence data. The second layer is the pooling layer, which is used to reduce the number of features, but it is different from the common pooling layer. The common pooling layer adopts the maximum pooling or average pooling, which is a method of combining convolution characteristics on time invariance. But it cannot deal with the large-scale sequence problem. QRNN absorbs the time series processing method in the Recurrent Neural Network (RNN) and adds it to the pooling layer, which effectively solves the time series problem. In QRNN, there are three pooling modes: f-pooling, fo-pooling and ifo-pooling. This paper uses f-pooling.

Convolution component: it is used to extract the input features, as shown in equations (2) - (4).

$$Z = \tanh(W_z * L) \tag{2}$$
$$F = \sigma\left(W_f * L\right) \tag{3}$$
$$O = \sigma\left(W_o * L\right) \tag{4}$$

where W_z,W_f,W_o are the tensors of the real number $R^{k\times d\times m}$, d is the vector dimension, m is the number of channels of the convolution component, and k is the size of sliding window on the sequence dimension. L is the current sequence, Z, F, O are the output of the convolutional layer. For example, if the convolution operation has a window size of two in the sequence dimension, then the above equations can be expressed as equations (5) - (7). l_t stands for the sequence on the current time t, l_{t-1} represents the sequence of the previous moment.

$$z_t = \tanh(W_z^1 l_{t-1} + W_z^2 l_t) \tag{5}$$
$$f_t = \sigma(W_f^1 l_{t-1} + W_f^2 l_t) \tag{6}$$
$$O_t = \sigma(W_o^1 l_{t-1} + W_o^2 l_t) \tag{7}$$

Pooling components: to reduce the extracted features, we use f-pooling, which contains one forget gate. So the current output can be expressed as equation (8).

$$h_t = f_t \odot h_{t-1} + (1 - f_t) \odot z_t \tag{8}$$

where h_{t-1} stands for the output of the upper layer QRNN, f_t, z_t are the output of the convolutional component in the current layer, $\odot$ is the signal of multiplication.

Although, the loop portion of these functions needs to be calculated and parallelized along with the feature dimensions for each time step in a sequence task. In other words, in actual operation, the extra time is negligible when dealing with very long sequences.

B. DISCRIMINATOR

Discriminator D as a classifier in the GAN, this paper uses the fundamental LSTM model as the discriminator to judge if the generated text is real. The discriminator scores the generated sequence in whole sentences and feeds the scores back to the generator. The discriminant function is shown in equation (9).

$$D(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)} \tag{9}$$

where P_{data} is the distribution of the real sample, P_g is the distribution generated by the generator. In general, the equation gives a quantified number for judging that the sequence is a real sample, so it can be seen that the value of discriminator will close to 0.5 when the $P_g(x)$ is closer to P_{data}. In order to get better results, in our experiment, the discriminator is set to calculate the value in multiple times. When the score is close to 0.5 which means the discriminator does not need to be trained.

The discriminator D in the GAN serves as a source of rewards in reinforcement learning. So how to calculate the rewards value is especially important. Suppose we need to generate M sequences of texts length of N, then the reward R_θ for the generated text can be calculated as shown in equations (10) - (11):

$$R_\theta = \frac{1}{M} \sum_{j=1}^{M} D_j(t_{1:i} + t_{i+1:N}) \tag{10}$$

$$t_{i+1:N} = \{t_{i+1}, t_{i+2}, t_{i+3} \cdots t_N\} = MC_\theta(t_{i+1,k}; N - (i+1)) \tag{11}$$

Here, $t_{1:i}$ refers to the partial sequence that has been generated previously, and a possible subsequent sequence $MC_\theta(t_{i+1,k}; N-(i+1))$ is obtained by using the optimized Monte Carlo search algorithm. Although the text generation still looks for the next word with the greatest expected reward word by word, the discriminator D gives the score to the whole generated sequence and feeds the score back to the generator. Before next round of iterative training, the generator continually optimizes itself based on the score returned by the current discriminator, and updates related parameters. As shown in equations (12) - (13).

$$\theta \leftarrow \theta + \eta \nabla R_\theta \tag{12}$$

$$\nabla R_\theta = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{1}{M}\sum_{j=1}^{M}\left[D_j(t_{1:i} + t_{i+1:N})\nabla \log P_\theta(t_{i+1:N}|t_{1:i})\right]\right\} \tag{13}$$

N is the length of each sequence here, η is the weight of the reward, M is the number of sequences, and $P_\theta(t_{i+1:N}|t_{1:i})$ stands for the probability of generating a sequence in the case of a known sequence $t_{1:i}$. It is time to stop updating the new discriminator until the generator gives convincing data.

C. OPTIMIZED REWARD STRATEGY

As mentioned in the previous description, the discriminator D gives a probability score for generating a sample and feeds it back to the generator as a reward. The discriminator D will not give a negative score to penalize the generator, even if a large number of syntax errors in the generated data, because the score is a probability value. In this way, the generator tends to reduce the probability of smaller reward value samples and increases the probability of larger reward value samples. Moreover, due to incomplete and sparse data, this distinction between reward and punishment is unclear which makes the training biased of G. Therefore, this paper subtracts a penalty value from score given by discriminator D for every generated sample, to expand the reward and punishment boundary and speed up the training of the model. Then, the optimized gradient calculation equation of the expected reward bonus (13) can be modified in the equation (14).

$$\nabla R_\theta = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{1}{M}\sum_{j=1}^{M}\left[(D_j(t_{1:i} + t_{i+1:N}) - b)\nabla \log P_\theta\,(t_{i+1:N}|t_{1:i})\right]\right\} \tag{14}$$

D. MONTE CARLO SEARCH OPTIMIZATION

Traditional Monte Carlo search is very time consuming. It needs to generate samples every time. Meanwhile, it calculates the previously generated item when calculating some of the later partial sequence reward estimates, resulting overfit. Therefore, we optimize the Monte Carlo search algorithm and scores the next sequence through the generated partial sequences, so the score of the entire sequence can be quickly obtained. For example, if it needs to calculate the score of $t_{1:i}$, the prefix is fixed by the current generator parameter, and it repeats to generate M possible complete sequences. Then the discriminator calculates the average reward score of the M samples as the simulated reward score $S(t_{1:i})$ of the partial sequence. It can be calculated by equation (15).

$$S(t_{1:i}) = \frac{1}{M}\sum_{j=1}^{M} D_j(t_{1:N}) \tag{15}$$

where M is the number of generated sequences, $t_{1:N}$ is a sequence of length N. In this way, we can calculate the reward of the sequence $t_{i+1:N}$ according to equations (14) -(15), as shown in equation (16).

$$S(t_{i+1:N}) = D_j(t_{1:N}) - S(t_{1:i}) \tag{16}$$

where $D_j(t_{1:N})$ represents the reward given by the discriminator to the sequence $t_{1:N}$. The optimized Monte Carlo search optimization algorithm is used to predict the next sequence of the obtained partial sequence by selecting the subsequent sequence with the maximum probability, as shown in equation (17).

$$MC_\theta\left(t_{i+1,k}; N-(i+1)\right) = \arg\max S(t_{i+1:N}) \tag{17}$$

We directly train a new discriminator D^* that scores some of the generated prefixes based on the original discriminator D. The entire prefix subsequence set of the real sample is recorded as L^+. The entire prefix subsequence set of the generated sample is also recorded as L^-. Then we select one or several subsequences from the two sets respectively and label them as real samples or not, and send them to the discriminator. We repeatedly train the discriminator in this way to increase the ability to score the prefix subsequences. Then we use the discriminator D^* to calculate the score $D_j(t_{1:t})$ of prefix subsequence. Experiments show that this method takes less time than traditional Monte Carlo search, as shown in TABLE 6 in section V.

E. GRAMMAR OPTIMIZATION

To eliminate the grammatical errors of the sentences generated by QuGAN, this paper uses the BERT model to modify and optimize the questions, including two parts: random mask and next sentence prediction.

Random mask: we use 20% probability to replace some words with the special symbol "$*$", and use BERT to predict masked words.

The next sentence prediction: we select 50% sentences from the generated Tibetan questions, and randomly select 50% sentences form corpus, to predict the next sentence.

F. ANSWER MATCHING

We use semantic similarity to match question and answer in corpus. The answer with the highest similarity of the question is the answer to the question.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. DATASET

We have obtained 21,783 Tibetan question and answer pairs from the Tibetan website. In our experiment, we select 17,000 pairs as the training data. The specific data set format is shown in TABLE 1. The UUID is the unique identifier of the question and answer pair.

TABLE 1. Examples in Tibetan QA corpus.

UUID	Questions	Answers
01dc7400-ee34-11e8-bb78-4a00077cdd10	རྣམ་དབྱེ་བརྒྱད་པ་འབོད་སྒྲུབ་ཚིག་ཞིག་རྣམ་པར་དབྱེ་བ་ཡིན་ནམ། (What is the 8th grammar mainly divided into?)	འབོད་སྒྲུབ་མིང་ཚིག་དང་འབོད་དཔེ་ཁྱད་པར་དུ་བྱས་པའི་མིང་གཞིས་རྣམ་པར་དབྱེ། (The eighth grammar is mainly divided into address words and nouns.)
01de5718-ee34-11e8-9734-4a00077cdd10	ཏིག་ཚང་ལ་དབྱེ་བ་གང་འདྲ་ཡོད་དམ། (What are the categories of Thang-ga?)	ཏིག་ཚང་ལ་ཚོན་ཚང་དང་། ནག་ཚང་། མཚལ་ཚང་། སེར་ཚང་བཅས་དབྱེ་བ་བཞི་ཡོད། (Thang-ga is divided into four kinds: Wa-Thang-ga, red Thang-ga, colorless Thang-ga and gold Thang-ga.)

B. DATA PREPROCESSING

Tibetan is an alphabetical language with characters as the basic unit. Characters are directly separated by "་", and a word composed of many characters. Since the accuracy of word segmentation in Tibetan is not high, we segment the text in two modes: word level and character level. For the character level, we divide the text sequence according to the "་" symbol. For the word level, we use the Tibetan words segmentation tool [42], and the format is as follows. For the Tibetan QA corpus, the questions and answers are spilt by the "$" symbol.

རྣམ་དབྱེ་/བརྒྱད་པ་/འབོད་སྒྲུབ་/ཚ་/ཅི་ཞིག་/རྣམ་པ་/ར་/དབྱེ་/བ་/ཡིན་/ནམ་/།/ $ འབོད་/སྒྲུབ་མིང་/ཚམ་/དང་/འབོད་པ་/ཉེ་/ཁྱད་པར་/དུ་/བྱས་/པ་/མིང་/གཞིས་རྣམ་པར་དབྱེ།

In addition, this paper uses the Word2vec tool [43] to represent characters and words.

C. EVALUATION

We refer to the method of machine translation evaluation, and use the machine translation method for Bilingual Evaluation Understand (BLEU) [44] to evaluate.

1) N-gram

BLEU adopts a N-gram matching rule to compare the proportion of the generated sentences similar to the n groups of words in the corpus. The accuracy of the N-gram can be calculated according to the equation (18).

$$P_n = \frac{\sum_i \sum_k \min\left(h_k\left(c_i\right), \max_{j\in m} h_k\left(s_{ij}\right)\right)}{\sum_i \sum_k \min\left(h_k\left(c_i\right)\right)} \tag{18}$$

where $h_k\left(s_{ij}\right)$ is the number of the k^{th} word W_k appeared in the corpus, $h_k\left(s_{ij}\right)$ is the number of W_k appeared in the generated text c_i. Molecular represents the minimum number of occurrences in the resulting sentence and sentence in the corpus.

(2) Penalty factor

Since the matching degree of N-gram may be higher as the length of the sentence becomes shorter, it tends to generate short sentences. In order to avoid this situation, BLEU introduces a length penalty factor (Brevity Penalty) in the final scoring result. The calculation is shown in equation (19).

$$BP = \begin{cases} 1 & if \quad l_c > l_s \\ e^{1-\frac{l_s}{l_c}} & if \quad l_c \le l_s \end{cases} \tag{19}$$

where l_c represents the length of the generated text, l_s represents the effective length of the standard text in the corpus. When there are multiple generated texts, the length closest to the standard text is selected. When the length of the generated text is greater than the length of the standard text, the penalty coefficient is 1, which means no penalty.

(3) BLEU

The accuracy of each N-gram decreases exponentially with the increasing of the order. To balance the statistics of each order, BLEU uses the average value to weight and multiplies by the penalty factor, shown in equation (20).

$$BLUE = BP \times \exp\left(\sum_{n=1}^{2} \frac{1}{2} \log P_n\right) \tag{20}$$

In this paper, we use BLEU-2 to evaluate the model because BELU-2 has low computational complexity and calculates rapidly.

D. PARAMETERS SETTING

Through experiments, we set the parameters of QRNN, LSTM and BERT model, shown in TABLE 2-3.

TABLE 2. Main parameters setting in QRNN.

Parameter	Value
Batch_size	128
Vector_size	64
Conv_size	3
Learning_rate	0.001
Train_iter	1000

TABLE 3. Main parameters setting in LSTM.

Parameter	Value
Batch_size	128
Vector_size	64
Learning_rate	0.01
Epoch	200
η	0.95
b	0.5

TABLE 4. Main parameters setting in BERT model.

Parameter	Value
Hidden_units_num	768
Vector_size	64
Layers_num	12
Attention_heads	12
Vocab_size	4756
Epoch	200

E. EXPERIMENTAL RESULTS OF MODELS

We use BLEU-2 as the evaluation, and conduct the following experiments.

SeqGAN: we use the SeqGAN model [39] as the baseline.

QuGAN: includes QRNN, LSTM, RL with optimization strategy and Monte Carlo optimization.

QuGAN-MCO: removed Monte Carlo optimization (MCO) part from the QuGAN model.

QuGAN-MCO+BERT:add the BERT model.

QuGAN+BERT:includes MC optimization and adds BERT model.

All the experiments consider the character-level and word-level. The experimental results are shown in TABLE 5 and Fig. 2.

TABLE 5. Experimental results of different models.

	BLEU-2			
Model	Character-level	Increase	Word-level	Increase
SeqGAN	0.719	—	0.663	—
QuGAN	0.727	+0.008 ↑	0.672	0.009 ↑
QuGAN-MCO	0.742	+0.023 ↑	0.731	0.068 ↑
QuGAN-MCO+BERT	**0.822**	+0.103 ↑	**0.786**	0.123 ↑
QuGAN+BERT	0.813	+0.094 ↑	0.769	0.106 ↑

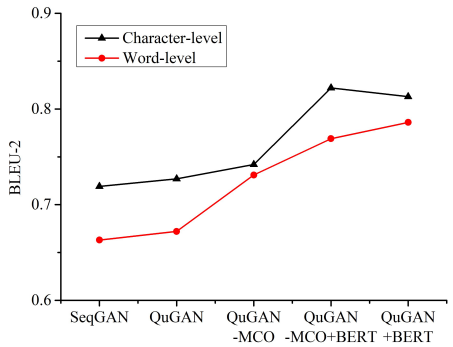

FIGURE 2. The experimental results in different models.

Moreover, we use QRNN and LSTM as generator respectively, and compare the speed of the models. Also,we conduct some experiments on MC optimization, they are shown in TABLE 6 and Fig. 3.

TABLE 6. Time performance comparison.

Model	Time (Hrs /100 sentences)
LSTM+MC	17
LSTM+MCO	11
QRNN+MC	6
QRNN+MCO	4.5

From TABLE 5-6 and Fig.2-3, we can find:

(1) For all the models, the BLEU-2 of word-level is lower than character-level, the average value is 0.04. The main reason is that the accuracy of Tibetan word segmentation is not high, and the wrong results will affect the whole results in the experiments.

(2) Compared with the SeqGAN model, the BLEU-2 value of QuGAN model is superior to the SeqGAN model.

(3) When removing MC optimization, the BLEU-2 increase 0.015 than QuGAN, and increases 0.023 than baseline. But the speed of the model with MC optimization is obviously faster than without MC optimization, which is shown in TABLE 6. LSTM+MCO increases 35.29% than LSTM+MC, and QRNN+MCO increases 25% than QRNN+MC. The main reasons are as follows. Firstly, in the MC optimization process, the grammatical structure of the Tibetan text is not combined, and the search segment selected is randomly selected by the machine every time. Secondly, the simple MC mainly scores the entire sentence generated once, avoiding the overall error caused by the segmentation error. MC optimization are not collected to make the sentence with the highest score. This shows that the optimized MC will lose a little accuracy in the final result but can significantly shorten the model training time.

(4) When adding BERT model, the BLEU-2 values increase 0.094 and 0.103 than baseline. QuGAN-MCO+BERT model has the highest BLEU-2 value 0.822 in the character-level. This proves that language model has a significant effect on Tibetan text generation.

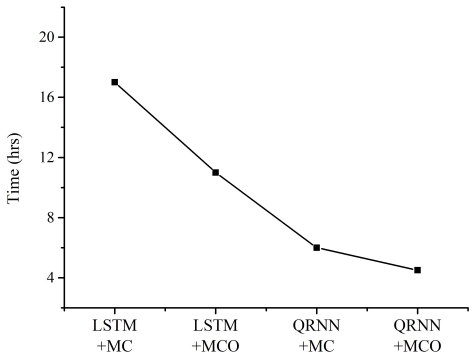

FIGURE 3. Time consuming of different models.

TABLE 6 and Fig. 3 show the speed of using different models. When adding MC optimization, the speed of LSTM+MCO increases 35.29% than LSTM+MC, and QRNN+MCO increases 25% than QRNN+MC.

TABLE 7. Examples of Tibetan QA corpus generated by QuGAN model.

Random initialization	MLE	Generated by Our Model
སྐན་ངག་གནམ་གཤིས་ཚ་ཚང་།	ཅི་ཞིག་ལ་སྐན་ངག་ཞེས། $ སྐན་ཞིང་འཇེབས་པ་ཞིག་ལ་ཞེས།	ཅི་ཞིག་ལ་སྐན་ངག་ཞེས། $ རྩོམ་ཡིག་རིགས་ཤིག
(Poetry, weather, complete.)	(What is poetry? $ Beautiful words, words, sentences.)	(What is poetry? $ A kind of writing.)
དབྱངས་ཞིན་བྱེད་དེག	དཔལ་འབྱོར་དང་དབྱངས་ཞེས། $ སྒྲོན་འཇུག་དང་རྗེས་འཇུག	ཅི་ཞིག་ལ་དབྱངས་ཞེས། $ གནས་དང་ཁྱིད་བའི་རྟགས་ཚམ་འཇུག་སོ་གསལ་ཞིང་། རང་ཉིད་གསལ་ཡེད་རྟགས་དང་ལྡན་པ་ལ་བརྟེན་ནས་མིང་ཚིག་གི་བརྗོད་པའི་སྐྱགས་བོར་འཕྲིན་རྒྱབ་ཞིག་ལ་ཞེས།
		(What is a consonant letter? $ It does not make sense in itself, encountering the base word can be fully issued corresponding tone.)
ཅི་ཞིག་ལ་རྩ་བ་ཞེས།	ཅི་ཞིག་ལ་རྩ་བ་བཞི་ཞེས། $ མ་བྱིན་ལེན། བསྡུན། སྲོག་གཅོད་དེ་བཞི།	ཅི་ཞིག་ལ་རྩ་བ་ཞེས། ཉུང་དངོས་ཀྱི་འབྱུང་རྩ་དང་གཞི་རྩ་ལ་ཞེས།
(What's fundamental?)	(What are the four fundamentals? $ No lying, no harm.)	(What's fundamental? $ The root or basic source or basic of the physical object.)

Meanwhile, the speed of using the QRNN model is much higher than the LSTM model by about 2.83 times. It is proved that MC optimization and QRNN have a certain acceleration effect on the model. It is also seen that the speed of the QRNN model is higher than the MC optimization.

Finally, we produce 3584 question and answer pairs. Examples are shown in TABLE 7.

VI. CONCLUSION

In this paper, we propose the QuGAN model which combines QRNN and RL. And the BERT model, optimized reward strategy and Monte Carlo search strategy are used to improve the performance. However, due to not consider the Tibetan grammar characteristics, there are certain invalid questions.

In the future, we will increase the accuracy of the generated corpus and enrich the information by adding the argument function and Tibetan grammar information.

REFERENCES

[1] O. Etzioni, "Search needs a shake-up," *Nature*, vol. 476, pp. 25–26, Aug. 2011.
[2] W. T. Yih, X. D. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA, 2014, pp. 643–648.
[3] L. Yu, K. M. Hermann, P. Blunsom, and S. Pulman, "Deep learning for answer sentence selection," Dec. 2014, *arXiv:1412.1632*. [Online]. Available: https://arxiv.org/abs/1412.1632
[4] R. Socher, C. C.-Y. Lin, C. D. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, WA, USA, Jul. 2011, pp. 129–136.
[5] G. Alex, "Generating sequences with recurrent neural networks," Aug. 2013, *arXiv:1308.0850*. [Online]. Available: https://arxiv.org/abs/1308.0850
[6] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," *A Field Guide Dyn. Recurrent Netw.*, vol. 6, no. 4, pp. 237–243, 2001.
[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[8] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, Jul. 2015, pp. 2067–2075.
[9] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *Proc. ACL*, Beijing, China, 2015, pp. 707–712.
[10] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proc. ACL*, Vancouver, BC, Canada, 2017, pp. 221–231.
[11] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," Oct. 2016, *arXiv:1606.03391*. [Online]. Available: https://arxiv.org/abs/1606.03391
[12] J. Yin, W. X. Zhao, and X.-M. Li, "Type-aware question answering over knowledge base with attention-based tree-structured neural networks," *J. Comput. Sci. Technol.*, vol. 32, no. 4, pp. 805–813, 2017.
[13] Tim Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiský, and P. Blunsom, "Reasoning about entailment with neural attention," Sep. 2015, *arXiv:1509.06664*. [Online]. Available: https://arxiv.org/abs/1509.06664
[14] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, and K. Toutanova, "Natural questions: A benchmark for question answering research," *Trans. Assoc. Comput. Linguistics*, vol. 7, no. 29, pp. 453–466, 2019.
[15] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," Jul. 2015, *arXiv:1506.02075*. [Online]. Available: https://arxiv.org/abs/1506.02075
[16] Y. Yang, W. Yih, and C. Meek, "WIKIQA: A challenge dataset for open-domain question answering," in *Proc. EMNLP*, Lisbon, Portugal, 2015, pp. 2013–2018.
[17] P. Li, W. Li, Z. He, X. Wang, Y. Cao, J. Zhou, and W. Xu, "Dataset and neural recurrent sequence labeling model for open-domain factoid question answering," Sep. 2016, *arXiv:1607.06275*. [Online]. Available: https://arxiv.org/abs/1607.06275
[18] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," in *Proc. ICLR*, Toulon, France, 2017, pp. 1–20.
[19] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proc. EMNLP*, Austin, TX, USA, 2016, pp. 2383–2392.
[20] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Vancouver, BC, Canada, 2008, pp. 1247–1250.
[21] M. Feng, B. Xiang, M. R. Glass, L. Wang, and B. Zhou, "Applying deep learning to answer selection: A study and an open task," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand. (ASRU)*, Scottsdale, AZ, USA, Dec. 2015, pp. 813–820.
[22] R. Das, M. Zaheer, S. Reddy, and A. McCallum, "Question answering on knowledge bases and text using universal schema and memory networks," in *Proc. ACL*, Vancouver, BC, Canada, 2017, pp. 358–365.
[23] D. Rajarshi, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," Nov. 2017, *arXiv:1711.05851*. [Online]. Available: https://arxiv.org/abs/1711.05851
[24] S. Curto, A. C. Mendes, and L. Coheur, "Question generation based on lexico-syntactic patterns learned from the web," *Dialogue Discourse*, vol. 3, no. 2, pp. 147–175, 2012.

[25] P. Rashmi and A. Joshi, "A discourse-based approach to generating why-questions from texts," in *Proc. Workshop Question Gener. Shared Task Eval. Challenge*, Arlington, VA, USA, 2008, pp. 1–3.

[26] L. Vanessa, C. Unger, P. Cimiano, and E. Motta, "Evaluating question answering over linked data," *J. Web Semantics*, vol. 21, pp. 3–13, Aug. 2013.

[27] Q. Q. Cai and A. Yates, "Large-scale semantic parsing via schema matching and lexicon extension," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguistics*, Sofia, Bulgaria, 2013, pp. 423–433.

[28] T. Wang, X. Yuan, and A. Trischler, "A joint model for question answering and question generation," Jun. 2017, *arXiv:1706.01450*. [Online]. Available: https://arxiv.org/abs/1706.01450

[29] L. Bauer, Y. Wang, and M. Bansal, "Commonsense for generative multi-hop question answering tasks," in *Proc. EMNLP*, Brussels, Belgium, 2018, pp. 4220–4230.

[30] K. Tomáš, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette, "The narrative QA reading comprehension challenge," in *Proc. Trans. Assoc. Comput. Linguistics*, Melbourne, VIC, Australia, 2018, pp. 317–328.

[31] S. Rao and H. Daumé, "Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, Melbourne, VIC, Australia, 2018, pp. 2737–2746.

[32] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio, "Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus," May 2016, *arXiv:1603.06807*. [Online]. Available: https://arxiv.org/abs/1603.06807

[33] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou, "Neural question generation from text: A preliminary study," in *Proc. Nat. CCF Conf. Natural Lang. Process. Chin. Comput.*, Dalian, China, 2017, pp. 662–671.

[34] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proc. NAACL*, New Orleans, LA, USA, 2018, pp. 2227–2237.

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," Oct. 2018, *arXiv:1810.04805*. [Online]. Available: https://arxiv.org/abs/1810.04805

[36] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, New Orleans, LA, USA, Dec. 2014, pp. 2672–2680.

[37] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," Sep. 2018, *arXiv:1809.11096*. [Online]. Available: https://arxiv.org/abs/1809.11096

[38] Y. Z. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," in *Proc. NIPS*, Barcelona, Spain, 2016, pp. 1–6.

[39] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, Feb. 2017, pp. 2852–2858.

[40] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," Sep. 2017, *arXiv:1701.06547*. [Online]. Available: https://arxiv.org/abs/1703.06547

[41] W. Fedus, I. Goodfellow, and A. M. Dai, "MaskGAN: Better text generation via filling in the_____," Mar. 2018. *arXiv:1801.07736*. [Online]. Available: https://arxiv.org/abs/1801.07736

[42] C. J. Long, H. D. Liu, M. H. Nuo, and J. Wu, "Tibetan POS tagging based on syllable tagging," (in Chinese), *J. Chin. Inf. Process.*, vol. 29, no. 5, pp. 211–216, 2015. [Online]. Available: http://jcip.cipsc.org.cn/CN/Y2015/V29/I5/211

[43] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, Lake Tahoe, Spain, Dec. 2013, pp. 3111–3119.

[44] C. Callison-Burch, M. Osborne, and P. Koehn, "Re-evaluation the role of bleu in machine translation research," in *Proc. EACL*, Trento, Italy, 2006, pp. 249–256.

YUAN SUN received the M.S. and Ph.D. degrees from the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, in 2004 and 2007, respectively. She is currently an Associate Professor with the School of Information Engineering, Minzu University of China. She has more than 50 papers published in various journals and international conferences. Her current research interests include natural language processing and knowledge engineering.

CHAOFAN CHEN received the B.E. degree from the Hubei University of Automotive Technology, China, in 2018. He is currently pursuing the M.S. degree with the Minzu University of China. His current research interests include question answering and information extraction.

TIANCI XIA received the B.E. degree from Tianjin Polytechnic University, China, in 2015, and the M.S. degree from the Minzu University of China, Beijing, China, in 2019.

XIAOBING ZHAO received the M.S. degree from Chungwoon University, Hongseong, South Korea, and the Ph.D. degree from Beijing Language and Culture University, Beijing, China, in 2003 and 2007, respectively. She is currently a Full Professor with the Minzu University of China. She has more than 80 papers published in various journals and international conferences. Her current research interest includes natural language processing.