# Controllable text generation based on varied frameworks - rhyming lyrics generation technology

*Zixiang* Li[1], *Qinfeng* Wu[2*], and *Longjie* Zhong[3]

[1]School of Computer Science and Technology, Guangdong University of Technology, Guangdong, 510000, China
[2]School of Computer Science, Wuhan University, Hubei, 430072, China
[3]School of Architecture, South China University of Technology, Guangdong, 510006, China

**Abstract.** Controllable Text Generation, as a cutting-edge technology in Natural Language Processing (NLP), has significantly improved the quality of text generation. Users can customize the generated content by setting specific attributes, formats, and emotional characteristics, thereby achieving the goal of conserving resources. However, despite notable progress in this field, several challenges remain, such as limited text diversity under multiple conditions and information disconnection during long-text generation. In light of this, this paper focuses on controllable text generation technology within a Chinese context, particularly emphasizing the key element of rhyming. The aim is to investigate an effective method for generating rhyming lyrics and poetry. By comparing the text generation performance under Recurrent Neural Networks (RNN), Bidirectional Recurrent Neural Networks (Bi-RNN), and Transformer frameworks, and evaluating the results using n-grams metrics, this study attempts to reveal which architecture is better suited for handling the specific controllable generation requirement of rhyming. This provides theoretical support and technical guidance for automatically creating Chinese poetry and lyrics.

## 1 Introduction

As artificial intelligence and deep learning continue to evolve, natural language processing (NLP) has become a central focus in research. By transforming human language into computer-recognizable data types and using neural network computing, researchers perform sentence emotion analysis, semantic understanding, text classification, and text generation. Therefore, the technology of controllable text generation has developed rapidly. By controlling the attributes, formats, and emotional characteristics of the generated text, the same data can accomplish various tasks through controllable generation to enhance the quality of text and reduce the cost of training.

For controllable text generation, there are some technologies related to deep learning currently in the academic field. The development process, including the Jordan network, was

* Corresponding author: 2023302111273@whu.edu.cn

proposed by Jordan et al. [1] in 1986, and the simplified Recurrent Neural Networks (RNN) model was further developed in 1990; in 2017, Vaswani et al. proposed the Transformer model [2], which gradually became popular and mainstream in the whole field of natural language understanding; in 2018, Li et al. proposed a delete-retrieval-generation framework to carry out the text emotion transfer task [3]. They first use the N meta-model method based on statistical principles to remove the emotional factors in the text and get the retained text. Then, they retrieve and replace similar sentences in the corpus with target emotion. The prompt learning (Prompt Learning) [4], proposed in the GPT-3 model in 2020, refers to a certain template. Process the input text information, recombine the input into a form that takes more advantage of the pre-trained language model processing, and generate the output from the language model. The prefix tuning method first proposed by Li et al. in 2021 [5], demonstrated powerful effects on the text generation task. Unlike cue learning methods, designers no longer construct models through natural language instead, the model is trained to get a prompt message, and get rid of the prior knowledge of the designer's language.

Yet, despite the continuous emergence of controllable text generation technologies in recent years, there have always been some challenges and problems. These include poor text control results. When multiple control conditions or more fine-grained control conditions occur, the variety in text generation is constrained and the generated text can not meet the control conditions well, text drift condition is serious, and content jump between the long distance first text and the second text is huge and lacks semantic connection, the resulting text content is repeated, the scarcity of sources of training data, and the huge amount of controllable parameters generated using the pre-trained model, the model training process is slow and difficult to converge and other phenomena [6].

Based on the above problems, relying on the controllable text generation technology, targeting the unique control conditions in the Chinese language field, this paper try to establish and explore an effective controllable text generation method to solve the problems of non-rhyming and smooth text generation in the field of lyrics and poetry. Rhyme can often make poetry work with a stronger rhythm and format neat. For rhyme the control conditions, this paper is based on the classic RNN, Bidirectional RNN, and Transformer model, by comparing a variety of models under the rhyme control of text generated results, to analyze and explore different neural network framework for rhyme the implementation effect, the controllable conditions in the field of Chinese poetry text generation, provide certain reference value and research direction. The remainder of this paper is organized as follows. Section 2 shows the basic model of the method used in this paper and the model information of the method used in this paper. Section 3 introduces the experimental results and conclusions, and Section 4 summarizes the full text.

## 2 Figures and tables models and methods

### 2.1 Basic models

RNN is a type of neural network that can process sequential data, the emergence of RNN solved the traditional neural network in processing sequence information limitations, to capture the long-distance dependence of the lyrics and the text information. This study adopts the two-way circulation neural network (Bi-RNN) as one of the basic models. Compared with traditional unidirectional RNN, Bi-RNN can more effectively capture the sequence information [7] at different positions in the input sequence by processing both forward-backward (forward RNN) and backward-forward (reverse RNN). In this regard, at each time step, at t, given a small batch number of input data. A forward RNN updates the current time step’s hidden state based on the input at the current time step t and the previous hidden state

(t-1). For a backward RNN, at each time step t, the hidden states are processed in the opposite direction. In bidirectional RNNs, the hidden states from both directions are combined at each time step t to form the final hidden state, respectively, as in equations (1), (2), (3), and (4) as shown. Among them, $W_{xh}^{(f)}$ is the weight matrix from the input to the hidden layer, $W_{hh}^{(f)}$ is the weight matrix from the hidden layer to the hidden layer, $b_h^{(f)}$ is the bias term, $\tanh$ is the activation function, $W_{ho}$ is the weight matrix from the hidden layer to the output layer, $b_o$ is the bias term of the output layer, $\mathrm{softmax}$ is the log-probability function, $\leftarrow$ denotes backpropagation, and $\rightarrow$ denotes forward propagation.

$$\overrightarrow{H}_t = \tanh\left(X_t W_{xh}^{(f)} + \overrightarrow{H}_{t-1} W_{hh}^{(f)} + b_h^{(f)}\right) \quad (1)$$

$$\overleftarrow{H}_t = \tanh\left(X_t W_{xh}^{(f)} + \overleftarrow{H}_{t-1} W_{hh}^{(b)} + b_h^{(b)}\right) \quad (2)$$

$$h_t = \left[\overrightarrow{H}_t, \overleftarrow{H}_t\right] \quad (3)$$

$$O_t = \mathrm{softmax}(W_{ho} h_t + b_o) \quad (4)$$

The Transformer language model processes text by inputting it into both the encoder and the decoder, relying on the attention mechanism to capture the characteristic information within the text. It allows the decoder to perform decoding operations on the output hidden layer $h_i$ of the encoder, ultimately achieving the prediction of the text output, specific as shown in Equation (5), Q, K, and V respectively by words embedded vector sequence through fully connected neural network layer calculation, $d_k$ is a single head contained in the long attention of vector dimension.

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (5)$$

## 2.2 Methods

This study explores two different lyric generation models, one based on RNN and Bi-RNN, combined with the Luong Attention mechanism, using Word2Vec for word vector encoding; the other is based on Transformer architecture and using enhanced inverse attention models. The RNN model aims to guide the generation of target lyrics through the semantic features and attention mechanism of the source language, consisting of the source language encoder, attention mechanism module, and decoder. The source language encoder uses long short-term memory network (LSTM) units to encode the input lyric sequence, The attention mechanism module provides a context vector during the decoding phase, enabling the model to better focus on critical information segments during generation. In the model based on Transformer, this paper constructs a network framework matching encoder and decoder, realizes text generation with the help of a reverse attention computer system, and explores the difference [8] between the realization effects of different frameworks on rhyme control.

### *2.2.1 The RNN and Bi-RNN models based on the Luong Attention*

To dynamically focus on various parts of the source sequence and enhance the coherence and relevance of the generated lyrics. This paper combines RNN with the Luong Attention mechanism to allow the decoder to rely on a fixed context vector when generating each output symbol but to dynamically focus from all the encoders to the most relevant information according to the current decoding state.

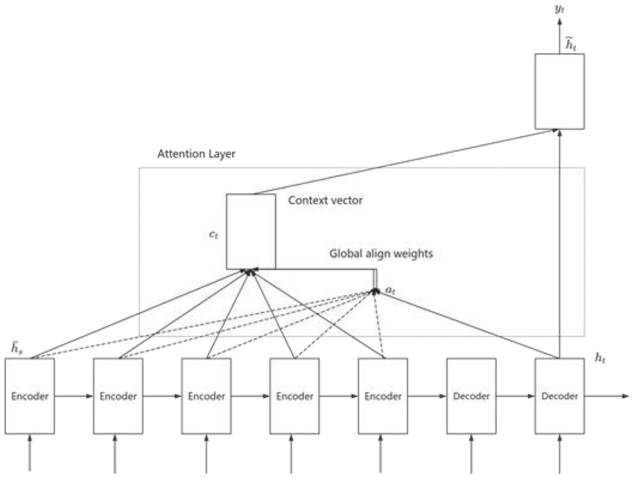


**Fig. 1.** Luong Attention mechanism (Photo/Picture credit: Original).

As shown in Figure 1, $h_t$ represents the current state of the decoder, while $\widetilde{h_s}$ stands for the state of the encoder at any position. The state $h_t$ has different attention scores concerning different $\overline{h_s}$ states from the encoder; relevant states receive higher scores, whereas irrelevant ones get lower scores. These attention scores are denoted as $a_t$. The attention scores are then multiplied with their corresponding encoder states and summed up to produce a new weighted context vector $c_t$, which is used together with $h_t$ to decode the output $\widetilde{h_t}$.

In practical terms, first, semantic vectors are generated as shown in Equations (6), (7), (8), and (9). Then, the hidden state is passed forward and predictions are made as indicated in Equations (10) and (11). Next, the initial hidden layer $s_t$ is computed, followed by the calculation of the hidden state of the attention layer denoted as $\widetilde{s_t}$. Finally, this is fed into the softmax layer to output the prediction distribution. Here, $c_t$ represents the generated semantic vector, $\alpha_{ti}$ denotes the computation of attention scores, $s_t$ refers to the initial hidden state, $y_{t-1}$ is the output from the previous step, and $W_a$ is the attention weight matrix, while $h_i$ represents the states of the input end encoder. The process involves computing the initial hidden layer $s_t$, subsequently calculating the hidden state $\widetilde{s_t}$ of the attention layer, and finally feeding it into the softmax layer to output the prediction distribution.

$$c_t = \sum_{i=1}^{T} \alpha_{ti} h_i \tag{6}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{T} \exp(e_{tk})} \tag{7}$$

$$s_t = \tanh(W[s_{t-1}, y_{t-1}]) \tag{8}$$

$$e_{ti} = s_t^T W_a h_i \tag{9}$$

$$\overline{s_t} = \tanh(W_c[s_t, c_t]) \tag{10}$$

$$o_t = \text{softmax}(V\widetilde{s_t}) \tag{11}$$

### *2.2.2 The Transformer model based on the inverse attention*

In contrast with the aforementioned RNN framework, this article utilizes a classic Transformer framework, implementing an inverse attention mechanism to achieve the goal of text generation with rhyme control. In traditional language sequences, rhyming words

typically appear at the end of sentences. This positioning poses a challenge when the decoder uses a unidirectional autoregressive attention mechanism to generate predictive text, as words preceding the ending will not contain the rhyming information. As a result, changing the sentence's rhyme will not alter the preceding text, leading to a disconnect between the rhyme and the semantic content of the sentence.

To address this issue, the inverse attention Transformer model proposed in this article reverses the order of the language sequence. It transforms the original sequence $[\langle \mathrm{BOS} \rangle, \langle \mathrm{word1} \rangle, \langle \mathrm{word2} \rangle \ldots \langle \mathrm{word_n} \rangle, \langle \mathrm{EOS} \rangle]$ by swapping the head and tail to become $[\langle \mathrm{EOS} \rangle, \langle \mathrm{word_n} \rangle, \langle \mathrm{word_{n-1}} \rangle \ldots \langle \mathrm{word_1} \rangle, \langle \mathrm{BOS} \rangle]$, while keeping the rhyming word $\langle \mathrm{word_n} \rangle$ fixed, and sequentially generating $\langle \mathrm{word_{n-1}} \rangle ... \langle \mathrm{word_1} \rangle$ thereafter. This approach fosters a connection between the preceding text and the rhyme, thus resolving the problem of rhymes being disconnected from the semantic content of the text [9].

## 3 Experiments and analysis

### 3.1 Dataset

To verify the Chinese lyric generation method based on rhyme control proposed in this paper, this paper carries out the relevant experimental verification on the Chinese lyric data set. The dataset required for the experiment came from the Chinese Lyrics database, which covers the vast majority of Chinese singers' songs before 2019. The database contains the works of 4,019 singers, of which 1,086 have more than 20 songs and 233 have more than 100 songs, with a total of 102,197 songs. The songs are recorded in a JSON file, and each record of the lyric data includes three fields: name (name of a song), singer (name of a singer), and lyric (lyrics).

During the data preprocessing stage, the collected lyric data undergoes text cleaning and format standardization to ensure data quality. Following this, the preprocessed dataset is split into a training set comprising 70% of the data and a test set comprising 30%. During the model training process, the prepared lyric data is fed into the Encoder and Decoder parts of the model. The model uses cross-entropy loss to evaluate the difference between the generated and the target words, optimizing accordingly.

### 3.2 Evaluation indicators

The n-gram model is commonly used as a statistical language model for evaluating language probabilities and generating text, it is based on the assumption that the occurrence of a word depends only on the previous n-1 words [10]. The n-grams model has the following features and steps. The first is the choice of n. The value of n determines the length of the context considered in the n-grams model. The common ones include unigram (1-gram, single word), bigram (2-gram, two consecutive words), trigram (3-gram, three consecutive words), etc. The second is to build the n-grams model, which is to count the occurrence frequency of each n-grams sequence for a given corpus. For example, if use bigram (n=2), for the generated sentence: "The sky is blue and so on misty rain". The resulting bigrams are ("day", "green"), ("green", "color"), ("color", "etc"), ("etc", "smoke"), ("smoke", "rain"), and then count the number of times each bigram appears in the whole training data set. The probabilities of each n-gram were then calculated. Probability can be obtained by the total number of times of the n-grams except for all the n-grams of the previous word (or the previous n-1 word). Let $\mathrm{Count}(w_{i-n+1:i})$ represent the frequency of occurrence of a sequence of n consecutive words $w_{i-n+1:i}$ within the corpus, and let $\mathrm{Count}(w_{i-n+1:i-1})$ represent the frequency of occurrence

of another sequence of n consecutive words $w_{i-n+1:i-1}$ within the same corpus. The process for calculating the probability of a sentence $w_{1:N}$ appearing is given by Equation (12).

$$P_{n-grams}(w_{1:N}) = \prod_{i=n}^{N} \frac{Count(w_{i-n+1:i})}{Count(w_{i-n+1:i-1})} \tag{12}$$

In addition, smoothing techniques are used. To address the sparsity and zero-probability issues in generated lyrics, additive smoothing (also known as Laplace smoothing) is applied to adjust probability estimates, ensuring that even if certain n-grams do not present in the training data, they will still have a non-zero probability. Let $Count(ng)$ be the count of n-grams ng occurring in the training set, $Total$ be the overall number of all n-grams in the training set, $smoothing\_factor$ be a small fraction used for smoothing, and $Vocab\_Size$ be the number of all possible preceding words. The calculation process is shown in equation (13).

$$P(ng) = \frac{Count(ng) + smoothing_{factor}}{Total + Smoothing_{factor} \times Vocab_{Size}} \tag{13}$$

The Mean coverage rate (Average Coverage) was calculated. Average Coverage refers to the proportion of n-grams in the generated sentences appearing in the training data. It measures how much of the n-grams in the generated sentence was ever present in the training data. It is calculated for each single generated sentence, how many of all n-grams were present in the training data, and then the proportion of these n-grams. Assuming that have a set of generated sentences $S = \{s_1, s_2, \ldots, s_m\}$, for each sentence $s_i$, and first calculate the set $N = \{n_{i1}, n_{i2}, \ldots, n_{ik}\}$ of n-grams for all of them. Then, check how many of these n-grams are present in the training data. Let R be the set of all n-grams in the training data, then for each sentence $s_i$ the proportion R of n-grams is shown in Equation (14). Next, for all the generated sentences, the average of their coverage was calculated.

$$C(s_i) = \frac{|N_i \cap R|}{|N_i|} \tag{14}$$

### 3.3 Design of the experimental parameters

The truncated gradient value of the RNN model is set to 5 to ensure that the model does not cause gradient explosion due to the large gradient, the word embedding dimension is set to 200 to capture rich semantic information, and the network output dimension is set to 10000 to match the size of the vocabulary and ensure that the model can generate every word in the vocabulary. The hidden layer size of the encoder and decoder is set to 200, to maintain the balance between the capacity of the model and computing efficiency, the learning rate decay rate is 0.98, decay learning rate after every 2 epochs, to enable the model to gradually adjust the weight during training. Each sequence is fixed at a length of 32, which helps to control the scale of the input data and improve training efficiency. The model uses 2 layers in the encoder and 2 layers in the decoder, with each hidden layer containing 200 nodes. In addition, a Dropout value of 0.1 was set to avoid overfitting.

The other parameters of the Transformer model are consistent with the original Transformer base model, using the structure of the 6-layer encoder and decoder. The word embedding vector dimension is 512, the feedforward full connection intermediate vector dimension is 2048, and the head number of long-head attention is 8. By dividing lyrics into many batches of the first sentence and the next sentence, the first sentence is entered into the

encoder, the next sentence is entered into the decoder, and the softmax function is used to complete the probability prediction of the generated text at the output end.

### 3.4 Experimental results

In order to verify the model's effectiveness and the quality of the generated output, this study comprehensively evaluated the RNN and Bi-RNN based on the Luong Attention mechanism Transformer model, using n-grams as the main evaluation index. The experimental results are illustrated in Table 1.

**Table 1.** The coverage results.

| | n-grams (Average Coverage) | |
|---|---|---|
| | n=2 | n=3 |
| RNN | 0.69 | 0.28 |
| Bi-RNN | 0.70 | 0.30 |
| Transformer | 0.64 | 0.26 |

Table 1 shows the coverage profile of each model. With n=2 and n=3, the Bi-RNN model achieves 0.70 coverage and 0.30, respectively, indicating that Bi-RNN contains more phrases in the reference corpus in the generated sentences, Bi-RNN can cover more targets or samples with strong predictive power. As n increases to 3, the coverage of the RNN model changes to 0.28, and Bi-RNN still maintains the highest coverage of 0.30, compared with Transformer performing slightly behind the remaining models, with lower coverage at these two n-grams, which may mean that the model has some challenges in capturing the context of multiple consecutive words.

## 4 Conclusion

This paper mainly studies the controllable text generation technology of Chinese poems and lyrics based on rhyme control conditions, aiming to solve the problems of generated text without rhyme and smooth meaning. This paper proposes a text generation method with rhyme as the control condition, which can ensure the consistency of the words in the sentence in the generation process, thus enhancing the rhythm and format of the generated text. Based on the classic RNN, two-way RNN, and Transformer models, this paper introduces additional mechanisms to better control the rhyme characteristics of the generated output and improve the ability of the generated text while maintaining diversity and consistency while satisfying the rhyme conditions. Finally, this paper verifies the experiment on the data set generated by Chinese lyrics, and the results can improve the rhyming effect of the generated text to a certain extent and provide valuable reference and technical support for the generation of poetry and lyrics in the field of Chinese language.

## Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

## References

1. M.-I. Jordan, Attractor dynamics and parallelism in a connectionist sequential machine, in Proceedings of the Annual Meeting of the Cognitive Science Society, 8 (1986)

2. A. Vaswani, N. Shazeer, N. Parmar, et al., Attention is all you need. Adv. Neural Inf. Process. Syst. 30 (2017)
3. J. Li, R. Jia, H. He, et al., Delete, retrieve, generate: a simple approach to sentiment and style transfer. arXiv preprint arXiv:1804.06437 (2018)
4. T.-B. Brown, Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
5. X.-L. Li, P. Liang, Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190 (2021)
6. Z.-P. Guo, Research on controllable text generation algorithm. Master Thesis, Chongqing University of Technology (2023)
7. H.-R. Cao, P.-X. Xu, J.-J. Chen, et al., Domestic pig price prediction and experiments based on Bi-RNN-LSTM model. Mech. Eng. Technol. **52**(10), 260-263+289 (2023)
8. X. Li, Research and application of poetry generation model based on Transformer. Master Thesis, East China Normal University (2023)
9. A.-V. Glazkova, D.A. Morozov, Applying transformer-based text summarization for keyphrase generation. Lobachevskii J. Math. **44**(1), 123-136 (2023)
10. S. Damian, H. Calvo, A. Gelbukh, Fake News detection using n-grams for PAN@ CLEF competition. J. Intell. Fuzzy Syst. **42**(5), 4633-4640 (2022)