Advanced Chinese Rap Lyric Generation with Integrated Markov Chain and LSTM Models

Songwei Li
Computer Science, New York University, New York, U.S.A.

Keywords: Chinese Rap Generator, Markov Chain, Long Short-Term Memory, Jieba.

Abstract: This paper aims to innovatively generate Chinese rap lyrics using advanced machine learning technologies, specifically Markov Chains and Long Short-Term Memory (LSTM) models. The project begins with the comprehensive collection and cleaning of Chinese rap lyrics data, covering key steps in data preprocessing, including word segmentation and tagging using Jieba. In the development phase of the two models, I first constructed a Markov Chain model based on enhanced tag analysis for basic lyric generation. Subsequently, I built an LSTM model that predicts the next word in a sequence by learning from sequences of lyrics. For this, I prepared the data by converting lyrics into sequences of tokens and creating corresponding labels for LSTM training. The architecture of the LSTM model was carefully designed to suit the needs of text generation, including embedding and LSTM layers. Additionally, I trained this model, adjusting hyperparameters to achieve optimal performance. In the testing and evaluation phase, I assessed the uniqueness and coherence of the Markov Chain model. For the LSTM model, I used quantitative metrics such as Perplexity or BLEU scores to evaluate the linguistic quality of the generated lyrics, assessing the creativity, thematic consistency, and overall appeal of the LSTM generated lyrics.

1 INTRODUCTION

Language models play a crucial role in the fields of artificial intelligence and natural language processing, particularly in the prediction and generation of text. Traditional language models rely on statistical methods to predict the probability distribution of word sequences, while modern models increasingly utilize deep learning techniques to process and generate language in more complex and efficient ways. Against this backdrop, lyric generation, as a special form of text generation, holds significant importance not only for technological development but also for cultural and artistic expression.

The Markov Chain model, a probabilistic model, typically relies on analyzing and predicting the probability of word sequences for lyric generation. While effective for simple text generation tasks, it may lack in logical coherence and thematic consistency. In contrast, the Long Short-Term Memory (LSTM) model, a type of advanced Recurrent Neural Network (RNN), can learn and process complex language structures and long-term dependencies through its unique gating mechanism, making it more effective in generating deep and creative lyrics. The significance of lyric generation lies not just in technological innovation but also in cultural and artistic aspects. Lyrics generated through machine learning can provide new sources of inspiration for music composition, especially in exploring new themes and styles. This technology can help artists overcome creative barriers and stimulate innovative thinking, thus promoting the development of music and culture. It also provides researchers with a unique perspective to understand and analyze language and its application in music, further fostering interdisciplinary research and collaboration (Whittaker and Thomason 1994, Privato et al 2022 & Ye 2000).

In summary, the application of Markov Chains and LSTM models in lyric generation not only demonstrates the power of deep learning technology in handling complex language tasks but also opens new possibilities for music creation and cultural expression. The development and application of these technologies indicate the future direction of natural language processing and artificial intelligence, while also carving out new realms for artistic creation and

Li, S.
Advanced Chinese Rap Lyric Generation with Integrated Markov Chain and LSTM Models.
DOI: 10.5220/0012842800004547

In *Proceedings of the 1st International Conference on Data Science and Engineering (ICDSE 2024)*, pages 371-376
ISBN: 978-989-758-690-3

cultural research (Nazarko 2021). The main work of this project can be summarized as follows.

(1) Data Collection and Preprocessing: The project team will comprehensively collect data on Chinese rap lyrics and perform necessary cleaning and preprocessing. This includes removing noise data, standardizing text formats, and using tools like Jieba for word segmentation and tagging, ensuring the quality and applicability of the data.

(2) Development and Implementation of Two Models: The core part of the project is the development of two different language models - the Markov Chain model and the Long Short-Term Memory (LSTM) model. For the Markov Chain model, the focus will be on basic lyric generation based on tag analysis; while the LSTM model will concentrate on using deep learning technology to learn and predict complex lyric sequences, generating richer and more coherent lyrics.

(3) Testing, Evaluation, and Iterative Improvement: After the development of the models, comprehensive testing and evaluation of both the Markov Chain model and the LSTM model will be conducted. This includes using quantitative metrics (such as Perplexity or BLEU scores) and qualitative analysis (such as expert reviews) to assess the language quality, creativity, thematic consistency, and overall appeal of the generated lyrics. Based on these evaluation results, further iterative improvements of the models will be made to optimize performance and output quality. Additionally, the integration of the strengths of both models will be explored to achieve more efficient and innovative lyric generation.

2 METHODS

2.1 Markov Chains

Markov Chains are mathematical models used to predict the probability of a system transitioning from one state to another (Ma et al 2021). These models are based on a key assumption known as the "memoryless" property or "Markov property," which posits that the future state of a system depends only on its current state and not on its previous history. In Markov Chains, each possible state has a certain probability of transitioning to another state within the system. These transition probabilities are typically represented using a matrix, known as the transition matrix.

Suppose I have a set of states $S = \{s_1, s_2, \dots, s_n\}$, where each s_i represents a possible state. The transition matrix P of a Markov Chain is an $n \times n$ matrix, where P_{ij} represents the probability of transitioning from state s_i to state s_j. Hence, the transition matrix can be represented as:

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \quad (1)$$

For any i, the transition probabilities satisfy the following condition:

$$\sum_{j=1}^{n} P_{ij} = 1 \quad (2)$$

This means that the total probability of transitioning from any state to any other state in the system must equal 1.

When applying Markov Chains to lyric generation, each state can represent a word or phrase. The transition matrix then defines the probability of moving from one word to another. In this way, the next word can be predicted based on the current word, thereby gradually building the lyrics of a song. The advantage of Markov Chains in this process lies in their simplicity and ability to capture short-term dependencies in a sequence. However, they typically cannot handle long-term dependencies, which can be a limiting factor in complex lyric structures.

2.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a special type of Recurrent Neural Network (RNN) designed to address the difficulties standard RNNs have in handling long-term dependencies (Ma et al 2021). LSTMs introduce unique "gate" structures (including input gates, forget gates, and output gates) that effectively maintain information over sequences, allowing them to capture dependencies over extended periods. Each gate within an LSTM unit has a specific computation method. Here are the fundamental formulas for these gates and the cell state updates.

Forget Gate f_t: Determines what information to discard. It computes this based on a combination of the input x_t(current input) and h_{t-1}(previous hidden state):

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (3)$$

Input Gate i_t and Candidate Values $\widetilde{C_t}$: Decide what new information to store in the cell state. The input gate decides which values will be updated, and

the candidate values are the new information that might be added to the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

$$\widetilde{C_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_C) \quad (5)$$

Cell State Update C_t: The old information C_{t-1} is partly forgotten through the forget gate, and new candidate values are added.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t} \quad (6)$$

Output Gate o_t and the Final Hidden State h_t: The output gate decides what part of the content to output, and the hidden state is based on the updated cell state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t * \tanh(C_t) \quad (8)$$

In these formulas, σ represents the Sigmoid activation function, $tanh$ is the hyperbolic tangent activation function, W and b are the weights and biases learned during training, and $*$ indicates elementwise multiplication.

The application of LSTM in lyric generation effectively handles long-term dependencies, something traditional Markov Chain models cannot achieve. By learning the sequence patterns of lyrics, LSTM can predict the next word or phrase, generating lyrics coherent in theme, style, and emotion. This capability makes LSTM particularly suited for complex and creative text generation tasks, such as lyric writing, where long-term memory of past information is crucial.

3 DATASETS

I will delve into the steps of data preprocessing, a crucial prerequisite for building machine learning models. My first task was data collection, accomplished by cloning a dataset containing Chinese hip-hop lyrics from a GitHub repository (Djwackey 2021). This dataset is organized in JSON files, with each file containing several songs. Then, I preprocess the data using Jieba to tokenize the words and eliminate the punctuation. The purpose of this step was to facilitate the clear identification of song boundaries in subsequent processes. All the formatted lyrics were appended to a text file, thereby creating a large lyric text corpus for model training, with each song's lyrics separated by a 'EOS' marker. These preprocessing steps ensured that the data was clean and structured, providing a well organized foundation for the model to learn from. Subsequent data processing might also include segmentation of vocabulary, removal of stopwords, and normalization to lowercase to further enhance the quality of the dataset.

4 EXPERIMENT

4.1 Markov Chain

In the process of constructing a Markov Chain based model, I begin by reading text data stored in a file: accepting the file, reading the content, and returning it. Subsequently, I utilizes the 'jieba' library for Chinese word segmentation, effectively breaking down continuous Chinese text into individual words. This step is crucial for Chinese data as Chinese text typically lacks the clear word delimiters, such as spaces, found in English. Next, I create a Markov Chain by traversing the list of words generated after segmentation, mapping each word to a list that contains the potential subsequent words. The dictionary thus constructed forms a Markov Chain, defining a series of potential successors for each word, thereby reflecting the probabilistic transitions between words. Then, I can generate new text with the constructed Markov Chain. This function randomly selects a starting word from the chain, then randomly chooses the next word based on the current one, repeating this process until a text of the specified length is produced. This method allows the generated text to appear coherent and logical. Lastly, the code sets a file path and sequentially calls the above-defined functions, starting with reading the text content, proceeding through segmentation and Markov Chain construction, and ultimately generating a text sequence of 50 words in length.

4.2 LSTM

Initially, the script prepares the data by reading a text file and segmenting it into words using 'jieba', a Chinese word segmentation tool. Word segmentation is a crucial step in processing Chinese text data, as, unlike English, which is delimited by spaces, Chinese requires specialized handling to delineate words. After segmentation, a sequence of words from the text data is obtained. Subsequently, a vocabulary is built by aggregating all unique words and creating mappings from words to indices and indices to words, laying the groundwork for the digital encoding of text data. These indices will later be used in an embedding layer, which serves to convert words into a

continuous vector format that the model can process. The words in the text are then converted to corresponding index sequences. On this basis, input sequences and labels required to train the model are generated. Each input sequence consists of a series of consecutive word indices, while the label is the index of the next word in the sequence. These sequences are then converted into PyTorch tensors for model processing.

Once the data is prepared, the script splits the dataset into training and testing sets according to a specified ratio and creates a mechanism for batch processing the data. In the model construction part, the core of the LSTM network includes an embedding layer, an LSTM layer, and a fully connected layer. The embedding layer transforms word indices into dense vector representations, the LSTM layer is responsible for processing sequence data and learning long-term dependencies, and the fully connected layer converts the LSTM layer's output into the final predictive output.

After the model is constructed, hyperparameters are set, as they significantly impact the model's performance. The embedding dimension is set to 128; the number of hidden units in the LSTM layer is set to 256; and the output dimension is set to 10000. The model is then instantiated, using cross entropy loss function and optimizer where learning rate is set to 0.001, and it is ready to be trained. For data loader, the batch size is set to 64.

The training process encompasses several epochs, each involving iterative optimization of the data in the training set. During this process, the model's weights are updated by computing the loss and performing backpropagation. Additionally, the script periodically outputs the loss and perplexity to monitor the model's training progress. After completing an epoch of training, the model is switched to evaluation mode to verify its performance on the test set, with accuracy reported. All code used in Python 3.8 and Pytorch 2.0 (Imambi et al 2021).

4.3 Experiment

4.3.1 Embedding Dimension

This is the dimension of the embedding layer in the model, which determines the size of word vectors. In this experiment, the embedding dimension is set to 128. This means that each vocabulary word is transformed into a 128-dimensional vector. However, I also tried dimensions such as 512, 348, 748, and so on.

4.3.2 Hidden Units in LSTM Layer

This represents the dimension of the hidden states in the LSTM layer. In the code, this value is set to 256. It signifies the number of memory units in the network, and at each time step, the LSTM outputs a 256-dimensional hidden state vector.

4.3.3 Vocabulary Size

The size of the vocabulary, where each dimension corresponds to a specific word. In this project, the output dimension equals vocab_size, which is the total number of words in the vocabulary, and its specific value depends on the size of the vocabulary constructed during the data preprocessing steps (Liu et al 2019). The results of different stages in the training process are shown in Table 1.

Table 1: Perplexity at Different Epochs.

Epoch	Perplexity
1	107.4
2	64.2
3	17.2
4	10.2
5	5.1
6	2.2

4.4 Result

The evaluation of the LSTM model's performance over various training epochs is depicted through the generated text samples and a Table 1 detailing the model's perplexity scores. The text generated after training the LSTM (Sengupta et al 2023 & Yu et al 2019 model showcases a significant improvement in the coherence and thematic consistency of the content. Initially, the sentences might have appeared somewhat disjointed and lacked logical progression. However, as the model progressed through the epochs, there's a discernible enhancement in the structure and flow of the generated text. This indicates that the LSTM model has learned to predict more accurate word sequences after being trained on the dataset. The perplexity scores, which measure how well a probability model predicts a sample, also reflect the model's increasing proficiency. A high perplexity score indicates poorer predictive power,

whereas a lower score signals better predictive capability. From the Table 1, I observe a steep decline in perplexity from 107.4 in the first epoch to 2.2 by the sixth epoch. This dramatic decrease signifies that the model has become significantly better at predicting the next word in a sequence, representing a substantial leap in learning from the data. This improvement in perplexity scores correlates with the qualitative improvements seen in the generated text samples. Initially, the model may produce text with less relevance and randomness, as evidenced by higher perplexity. But as the model trains and the perplexity decreases, the output becomes more coherent and contextually appropriate. This is a typical observation in LSTM models, as they are well-suited to capture and utilize the long-term dependencies within the text data, which is crucial for generating meaningful language sequences. The blue scores of both LAST models and Markov chain models are relatively low, since the models at the stage of generating consecutive words instead of sentences. However, the words in a sentence generated by the LSTM model can be easily put into the same context while the relationship of each word generated by Markov chain model is relatively weak.

In summary, the LSTM model has demonstrated a promising ability to learn from the corpus of Chinese rap lyrics. The generated text samples, although limited, suggest that the model is capturing the nuances of the language and the style of the genre. Meanwhile, the quantitative reduction in perplexity offers a concrete measure of the model's evolving competence. Together, these outcomes underscore the LSTM's potential in natural language generation tasks and its effectiveness in modeling complex language patterns.

5 CONCLUSION

In this paper, I have successfully developed and trained a Long Short-Term Memory (LSTM) network-based language model for generating Chinese rap lyrics that exhibit thematic coherence and logical structure. Evaluating the model's performance across various training epochs, I observed a significant enhancement in its predictive capabilities, evidenced by both the improved quality of generated text samples and a marked reduction in perplexity. The initial generated text may have lacked coherence and logic, but with continued training, the quality of the text substantially improved. Perplexity dropped from 107.4 in the first training epoch to 2.2 by the sixth epoch, indicating a substantial increase in the model's effectiveness in learning from the data. The conclusion drawn is that LSTM models are highly suitable for processing and generating complex language patterns, especially in natural language generation tasks that require an understanding of long-term dependencies. My model demonstrated the potential to capture the unique rhythm and style of Chinese rap lyrics and progressively learned to generate new, creative lyrical content throughout the training process. For future work, I plan to extend and deepen my efforts in several areas:

(1) Model Structure Optimization. Although the current LSTM model has shown promising performance, I believe that deeper neural network architectures or the introduction of more advanced models, such as Transformers or BERT, could further improve the quality of text generation.

(2) Hyperparameter Tuning. I will explore a broader hyperparameter space to find a more optimized model configuration. Additionally, considering the significant impact of different embedding dimensions on model performance, I aim to employ automated hyperparameter search methods, like Bayesian optimization, to determine the optimal settings.

(3) Dataset Expansion. To enhance the model's robustness and generalization ability, I plan to collect and integrate a more diverse set of Chinese rap lyrics data. Moreover, incorporating other forms of Chinese textual data may help the model learn richer language patterns.

(4) Creativity Assessment. I will develop new metrics to quantify the creativity and diversity of the lyrics generated by the model. While current perplexity metrics focus on prediction accuracy, I hope to more comprehensively assess the quality of generated text in the future.

(5) Interactive Generation Tools. Ultimately, I aim to develop an interactive platform that allows users to input specific themes or keywords and have the model generate corresponding lyrics. This will make the model more engaging and practical for real-world applications. By continuing to research and improve, I believe that LSTM models and other deep learning technologies will bring revolutionary progress to the field of natural language processing, particularly in natural language generation, in the future.

REFERENCES

J. Whittaker, M Thomason, A Markov chain model for statistical software testing, *IEEE Transactions on*

Software Engineering, (IEEE Press, NY, 1994), pp. 812-824.

N. Privato, O. Rampado and A. Novello, “A Creative Tool for the Musician Combining LSTM and Markov Chains in Max/MSP”, in *International Conference on Computational Intelligence in Music, Sound, Art and Design*, (Stringer-Verlag, Berlin, 2022), pp. 228-242.

N. Ye, “A markov chain model of temporal behavior for anomaly detection”, in *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, (United States Military Academy, West Point, NY, 2000), 166, pp. 171-174.

K. Nazarko, Practical text generation using GPT-2, LSTM and Markov Chain, Toward data science, (2021).

R. Ma, X. Zheng, P. Wang, The prediction and analysis of COVID-19 epidemic trend by combining LSTM and Markov method, Sci Rep 11, (2021).

Djwackey, Chinese-hiphop-lyrics, https://github.com/djwackey/chinese-hiphop-lyrics.git, (2021).

S. Imambi, K. B. Prakash, and G. R. Kanagachidambaresan, Pytorch, Programming with TensorFlow: Solution for Edge Computing Applications, (Springer Nature Switzerland AG, Cham, 2021), pp. 87–104.

L. Liu, Y, Lin and J. Reid, Improving the Performance of the LSTM and HMM Models via Hybridization, arXiv, (2019).

A. Sengupta, A. Das and S. I. Guler, Hybrid hidden Markov LSTM for short-term traffic flow prediction, arXiv, (2023).

Y. Yu, X. Si, C. Hu and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *in Neural Computation*, (MIT Press, Cambridge, 2019), 31(7), pp. 1235-1270.