Chinese Poetry Generation based on Long Short-Term Memory Model, Hidden Markov Model and Maximum Entropy Model

Junkai Li
2023316022

Ziqiang Zhang
2023210911

Tianyu Zhang
2023316085

Abstract

This project aims to implement Chinese poetry generation using custom Bidirectional Long Short-Term Memory (LSTM) Model, Hidden Markov Model (HMM) and Maximum Entropy Model. We first preprocess the Chinese Poetry Dataset to segment the poetry sentences and convert them into a format suitable for input to the model. In Bidirectional LSTM Model, we implemented the model from scratch and set up a two-layer bidirectional LSTM network to train on the preprocessed dataset. Experimental results show that the model effectively captures the semantics and structure of the poetry, producing high-quality generated lines. In HMM model, we implement both Baum-Welch and classical Viterbi algorithms to train the model and generate poetry. Experiments show that our model can generate poetry with high BERT similarity scores, which means the model can successfully learn the poetry semantics. In Maximum Entropy Model, we leverage the principle of maximum entropy to make predictions that are as uniform as possible given the known constraints. Experiments show that this model could produce the next poetry line with the same length and similar semantics and structures.

1 Introduction

With the development of natural language processing technologies, poetry generation, as a special text generation task, has gained widespread attention. Chinese poetry, as an important part of traditional Chinese culture, possesses unique literary value and artistic charm. This project aims to use modern deep learning techniques, HMM and Maximum Entropy Model to achieve automatic poetry generation, thereby promoting the inheritance and development of traditional culture.

2 Method

2.1 Bidirectional LSTM Model

One method of this project uses a custom bidirectional LSTM model. The model mainly consists of the following parts:

- **Data Preprocessing**: Segment the poetry sentences from the dataset and convert them into a format suitable for input to the neural network.
- **Custom LSTM Unit**: Implement a basic LSTM unit to capture the temporal dependencies in the input sequences.
- **Custom Bidirectional LSTM**: Based on the basic LSTM unit, implement a bidirectional LSTM to better capture the contextual information of the sentences.
- **Model Training**: Train the model using the backpropagation algorithm so that it can generate the next line based on the input line.

Preprint. Under review.

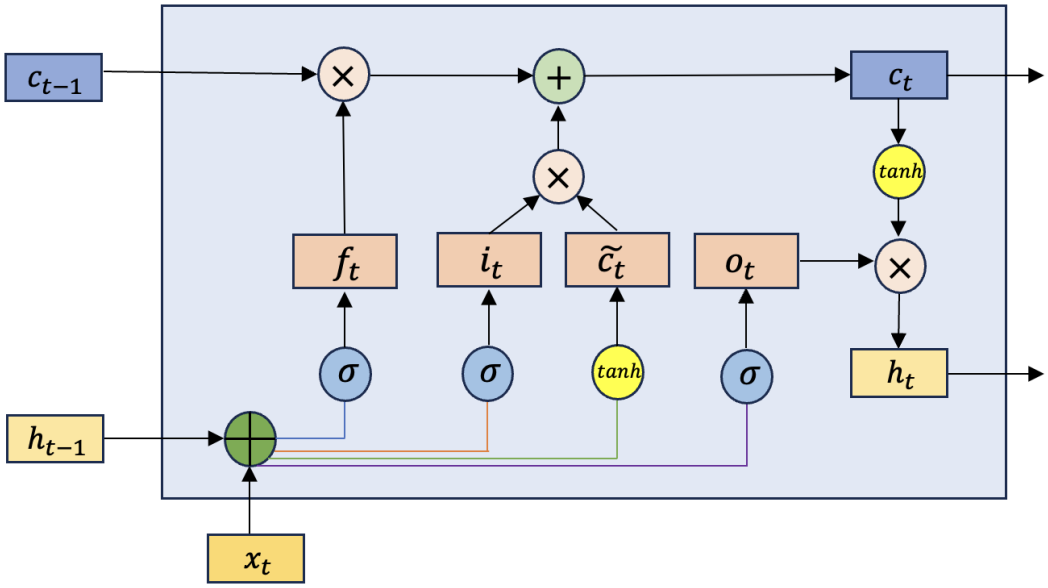

Figure 1: LSTM Unit

2.1.1 Data Preprocessing

First, we preprocess the Chinese Poetry Dataset by segmenting each line of poetry and converting the words into corresponding word indices. Specifically, we extract the upper and lower lines of ancient poems from the dataset and construct the upper and lower lines of ancient poems into input-output pairs. Then, we use the Tokenizer object to convert the input-output pairs into sequences of numbers, where each number corresponds to a Chinese character. We also pad the input and output sequences to ensure they have consistent length.

2.1.2 Custom LSTM Unit

The LSTM unit Hochreiter and Schmidhuber [1997] introduces multiple gate mechanisms to control the flow of information, effectively addressing the long-term dependency problem in traditional Recurrent Neural Networks (RNN). As shown in Figure 1, an LSTM unit contains three main gates: the forget gate, the input gate, and the output gate, along with a cell state.

The forget gate decides which information in the cell state should be discarded.

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

where σ is the sigmoid activation function, and W_f and b_f are the weight matrix and bias vector of the forget gate.

The input gate controls which new information is stored in the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3}$$

The cell state is updated by combining the previous cell state c_{t-1} and the new information.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{4}$$

The output gate controls the information output from the cell state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

2.1.3 Custom Bidirectional LSTM

The bidirectional LSTM includes two LSTM units, one processing the sequence forward and the other processing it backward. By concatenating the outputs of the two LSTMs, we can capture the contextual information of the sequence. This is particularly useful in natural language processing, where the meaning of words or characters often depends on both preceding and following context.

Specifically, the advantages of Bidirectional LSTM include:

- **Richer Contextual Information**: By considering both forward and backward information, Bidirectional LSTM better understands the context. This is crucial for language generation tasks, where the meaning of each word may be influenced by the surrounding words.
- **Improved Accuracy**: Due to the ability to utilize more complete contextual information, Bidirectional LSTM shows higher accuracy and better performance in many tasks.
- **Handling Long-Distance Dependencies**: Bidirectional LSTM effectively captures and utilizes long-distance dependencies, which is essential when dealing with long texts or time sequences.

2.1.4 Model Training

We train the model using the cross-entropy loss function and the Adam optimizer. During training, the input sequences are the upper lines of the poetry, and the target sequences are the corresponding lower lines.

2.2 HMM

The second part of this project applies HMM, a statistical machine learning model used to represent systems that are Markov processes with hidden states. This section mainly outlines the following parts:

- **Problem Definition**: We define the poetry generation problem in the HMM scenario. Specifically, we define the hidden states and observable events of HMM in poetry generation.
- **Training Phase**: This subsection describes how we use the pre-processed data to train the model and get the best HMM arguments.
- **Generating Phase**: This subsection describes how the model generates the next line of poetry, given the previous observations.

2.2.1 Problem Definition

The poetry generation problem asks the model to generate the poetry's next line with the previous line provided. In HMM, there are two important components: observations and hidden states. Clearly, there are connections between the poetry and HMM. So we define the problem as: *Given a poetry line as the observation sequence, the HMM needs to find the best hidden state sequence, which is the next poetry line.* From the above definition, the HMM has the following properties:

- The upper and lower lines of the pre-processed data from §2.1.1 define the input observation set $\mathcal{O}$ and hidden state set $\mathcal{S}$.
- Its initial hidden state probability vector π has dimension N, where N is the total number of unique characters in $\mathcal{S}$.
- Its transitional hidden state probability matrix A and observation emission matrix B has dimension $N * N$ and $N * M$ correspondingly, where M is the total number of unique characters in $\mathcal{O}$.

2.2.2 Training Phase

The HMM model training phase consists of two parts: Parameter Statistical Initialization and Parameter Fine-Tuning.

Parameter Statistical Initialization. As we already have input set $\mathcal{S}$ and $\mathcal{O}$, we can statistically build HMM arguments π, A and B. For example, π corresponds to the statistical counting ratio of all the first characters of hidden state sequences among $\mathcal{S}$.

Parameter Fine-Tuning. We can directly use the arguments from the first step, as it has learned from all the training data. But we can still fine-tune the HMM with Baum-Welch algorithm shown in Wikipedia [2024a]. The Baum-Welch algorithm is a specific instance of the Expectation-Maximization (EM) algorithm tailored for HMMs (HMMs). It has the following advantages: First, it is specifically designed for HMMs, which allows it to take advantage of the structure and properties of HMMs. Second, it leverages the forward-backward algorithm to efficiently compute the required probabilities. It has a time complexity of $O(N^2T)$, where T is the length of the observation sequence. In Baum-Welch algorithm, HMM arguments π, A and B are updated in each training round as the following:

$$\pi_i = \gamma_1(i) \tag{7}$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{8}$$

$$b_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot \mathbb{I}(O_t = k)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{9}$$

Here, i, j represents different hidden states, and k means the output observation state. $\gamma_t(i)$ denotes the probability of being in state i at time t. $\xi_t(i,j)$ denotes the probability of transitioning from state i to state j at time t.

2.2.3 Generating Phase

Our HMM utilizes the classical Viterbi algorithm shown in Wikipedia [2024b]. Given an observation sequence with length T, the Viterbi algorithm finds the most probable sequence of hidden states using dynamic programming. First, the Viterbi algorithm tries to record the state that maximizes the state probability for each time (Equation 10, 11).

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \le t \le T, \quad 1 \le j \le N \tag{10}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], \quad 2 \le t \le T, \quad 1 \le j \le N \tag{11}$$

After obtaining the last hidden state that maximizes the probability, Viterbi applies backtracking (Equation 12) and generates the hidden state sequence.

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1 \tag{12}$$

2.3 Maximum Entropy Model

This section outlines the implementation and application of a Maximum Entropy model for generating ancient Chinese poetry. The model is a probabilistic model widely used for various natural language processing tasks due to its flexibility and capacity to capture complex relationships in data De Martino and De Martino [2018]. The key components of our approach include:

- **Data Preprocessing**: Preparing the dataset by transforming poetry lines into a suitable format for model training.
- **Model Implementation**: Developing the model to predict the next character based on the input sequence.
- **Model Training**: Training the model using a large corpus of poetry to optimize the parameters.
- **Prediction and Generation**: Using the trained model to generate new lines of poetry.

2.3.1 Data Preprocessing

Data preprocessing involves converting the raw poetry data into input-output pairs suitable for model training. The dataset consists of lines from ancient Chinese poems, segmented into sequences of characters. Specifically, we process each line to form input sequences and corresponding target sequences.

To ensure uniformity, each input and target sequence is padded to a fixed length. A 'Tokenizer' object is used to convert these sequences into numerical indices representing each character. This transformation allows the model to process the textual data efficiently.

2.3.2 Model Implementation

The core of our approach is the implementation of the Maximum Entropy model. The model leverages the principle of maximum entropy to make predictions that are as uniform as possible given the known constraints. This is achieved through the following steps:

- **Parameter Initialization**: Randomly initializing the model parameters w, which are stored as a dictionary.
- **Feature Mapping**: Constructing feature mappings based on the input data to estimate the empirical distributions $\hat{P}(X)$ and $\hat{P}(X, Y)$.
- **Expectation Calculation**: Computing the expected values of the feature functions under the empirical distribution $\hat{P}(X, Y)$ and the model distribution $P(X|Y)$.
- **Parameter Update**: Iteratively updating the model parameters using gradient ascent to maximize the log-likelihood of the training data.

Mathematical Formulation:

The model computes the conditional probability $P(Y|X)$ as:

$$P(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_i w_i f_i(X, Y)\right)$$

where $Z(X)$ is the normalization factor, w_i are the model parameters, and $f_i(X, Y)$ are the feature functions.

The parameters are updated using the following rule:

$$w_i \leftarrow w_i + \eta \log\left(\frac{\hat{E}[f_i]}{E[f_i]}\right)$$

where η is the learning rate, $\hat{E}[f_i]$ is the empirical expectation, and $E[f_i]$ is the model expectation.

2.3.3 Model Training

Training the model involves optimizing the model parameters using the training dataset. We use the empirical distributions derived from the data to iteratively adjust the parameters such that the model's predictions align with the observed data. The training process includes:

- **Building Mappings**: Calculating the empirical distributions $\hat{P}(X)$ and $\hat{P}(X, Y)$ from the training data.
- **Expectation Calculation**: For each training iteration, computing the expected values of the feature functions under the current model.
- **Parameter Update**: Adjusting the parameters using the computed expectations to improve the model's fit to the data.

The training process continues until the parameters converge or the maximum number of iterations is reached.

2.3.4 Prediction and Generation

Once the model is trained, it can be used to generate new lines of poetry. The prediction process involves:

- **Input Reconstruction**: Reconstructing the input sequences by renaming the variables to differentiate them.
- **Probability Calculation**: Computing the conditional probabilities $P(Y|X)$ for each possible output based on the input sequence.
- **Output Generation**: Selecting the output with the highest probability as the predicted next character or word.

This approach allows the model to generate coherent and contextually appropriate lines of poetry, maintaining the stylistic elements of ancient Chinese poetry. The generated lines can then be evaluated for their quality and fluency.

The complete implementation and training of the Maximum Entropy model for poetry generation are provided in the accompanying code. This includes data preprocessing, model training, and generation functions, ensuring a comprehensive pipeline for ancient Chinese poetry generation.

3 Experiments

3.1 Bidirectional LSTM Evaluation

3.1.1 Setup

We used a Chinese dataset containing 10 thousands of poems for training. The dataset was divided into training and validation sets. Through 250 epochs of training, the model's accuracy on the training and validation set gradually improved, and it eventually produced high-quality generated lines.

3.1.2 Result

The accuracy rate on the training set reached 99.72%, and the accuracy rate on the validation set reached 78.48%. The model can output the corresponding lower sentence for the upper sentence it has seen, and can output the lower sentence with parallelism and rhyme for the upper sentence it has not seen. This proves that the model has indeed learned the information such as parallelism and rhyme of the upper and lower sentences of ancient poems, and the effect is good.

3.2 HMM Evaluation

3.2.1 Setup

We extracted all the couplets from 50,000 Tang dynasty poems, resulting in approximately 110,000 corresponding pairs of lines. These sequence pairs are used as the training data set. Despite the large number of training pairs, the training data for HMM is limited and sparse, as the total number of unique Chinese characters is large. We use several ways to modify the training process.

- As the probability matrix dimensions are too large, we do not apply the parameter fine-tuning process mentioned in §2.2.2.
- To minimize the negative effect of some frequently-shown words (such as "ming yue", "you you", etc.), we adopt the approach of taking the logarithm of the final statistical result in parameter initialization.

We use another 9300 observation sequences extracted from another 2,000 Tang dynasty poems as the testing data set. To verify the quality of our model, we apply BERT similarity score metric from Zhang et al. [2019], which shows the semantic similarity of the generated poetry and the original poetry lines. If the BERT similarity is close to 1, that means our model can generate poetry lines with high quality.

3.2.2 Result

The HMM model extracts 8800 unique Chinese characters from the training data. So the total number of model arguments is 154,888,800. We compute the average BERT similarity of all testing observation sequences, it shows that the result is 0.80, which means the HMM can learn the poetry semantics and structures effectively.

4 Conclusion

This project achieved automatic Chinese poetry generation using a custom Bidirectional LSTM Model, HMM and Maximum Entropy Model. Experimental results show that both Bidirectional LSTM Model and HMM model effectively capture the semantics and structure of the poetry, producing high-quality generated lines. In future work, we will further optimize the model architecture to improve the quality and diversity of generated poetry.

References

Andrea De Martino and Daniele De Martino. An introduction to the maximum entropy approach and its application to inference problems in biology. *Heliyon*, 4(4):e00596, 2018. ISSN 2405-8440. doi: https://doi.org/10.1016/j.heliyon.2018.e00596. URL `https://www.sciencedirect.com/science/article/pii/S2405844018301695`.

S Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997.

Wikipedia. Baum–welch algorithm. `https://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm`, 2024a. [Online; accessed 9-June-2024].

Wikipedia. Viterbi algorithm. `https://en.wikipedia.org/wiki/Viterbi_algorithm`, 2024b. [Online; accessed 9-June-2024].

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *ICLR 2020 Conference*, 2019.