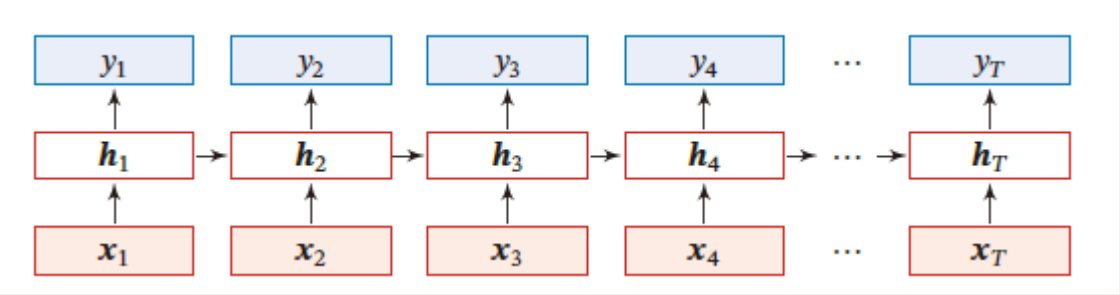
诗歌生成模型报告（作业三）

2252699 王鉴戈

1. RNN、LSTM 和 GRU 模型介绍

循环神经网络（RNN）是一种适用于处理序列数据的神经网络结构。其核心思想是维护一个隐藏状态（hidden state），用于存储历史信息，并通过递归方式更新。

1.1 RNN（循环神经网络）



RNN 通过一个循环结构处理序列数据，每个时间步的隐藏状态由前一个时间步的隐藏状态和当前输入决定。其数学表达式如下：

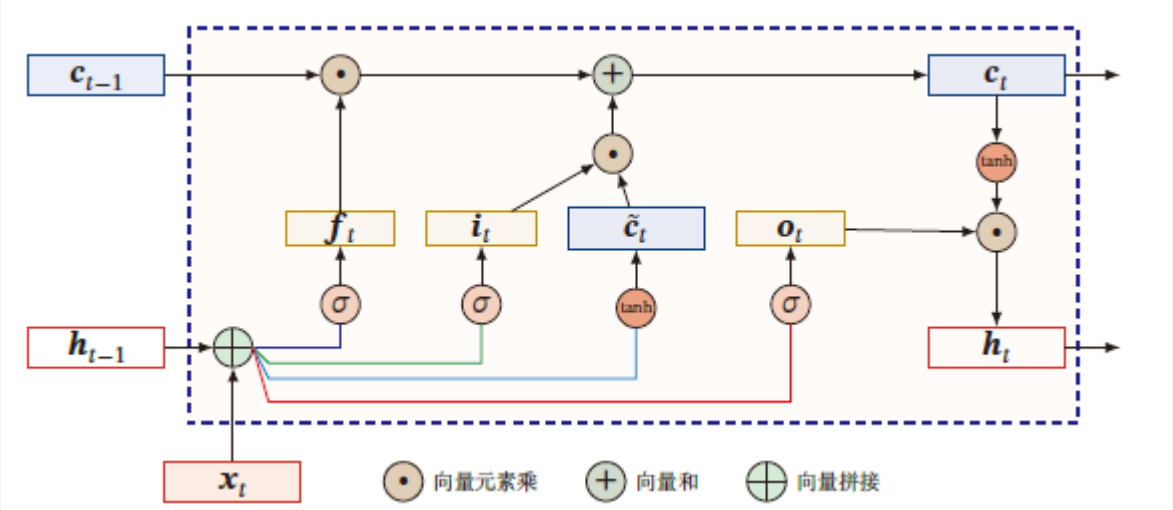
$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$
$$y_t = V h_t$$

其中：

- h_t 表示当前时间步的隐藏状态，
- x_t 是当前时间步的输入，
- W_h, W_x, b, V 是模型的参数，
- f 通常为非线性激活函数（如 tanh 或 ReLU）。

RNN 存在梯度消失问题，使得长期依赖信息难以学习。

1.2 LSTM（长短时记忆网络）



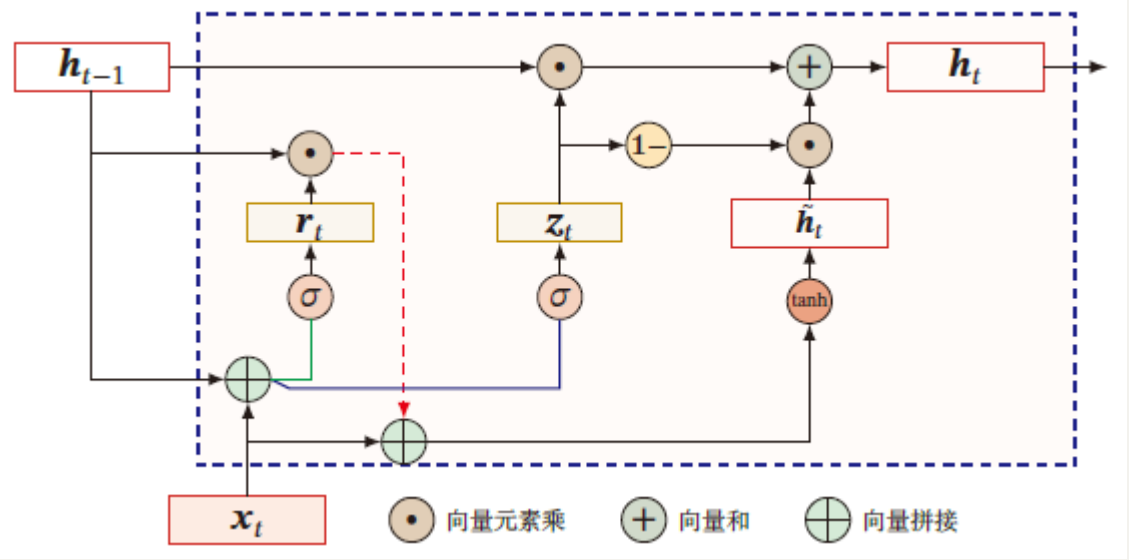
LSTM 通过引入门控机制（遗忘门、输入门、输出门）解决了 RNN 的梯度消失问题。

LSTM 的核心计算如下：

- 遗忘门：$f_t = \sigma(W_f[h_{t-1}, x_t]^T + b_f)$
- 输入门：$i_t = \sigma(W_i[h_{t-1}, x_t]^T + b_i)$
- 输出门：$o_t = \sigma(W_o[h_{t-1}, x_t]^T + b_o)$
- 候选记忆：$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t]^T + b_C)$
- 记忆单元更新：$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
- 隐藏状态更新：$h_t = o_t * \tanh(C_t)$

LSTM 适用于长序列的建模，能有效捕捉长期依赖关系。

1.3 GRU（门控循环单元）



GRU 是 LSTM 的简化版本，它合并了遗忘门和输入门，使计算更加高效。

GRU 计算方式如下：

- 更新门：$z_t = \sigma(W_z[h_{t-1}, x_t]^T + b_z)$
- 重置门：$r_t = \sigma(W_r[h_{t-1}, x_t]^T + b_r)$
- 候选隐藏状态：$\tilde{h}_t = \tanh(W_h[r_t * h_{t-1}, x_t]^T + b_h)$
- 计算当前隐藏状态：$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

GRU 计算量更小，适用于高效建模。

---

2. 诗歌生成过程

本项目的诗歌生成基于 RNN 模型，主要包括以下几个步骤：

2.1 TensorFlow版本：

2.1.1 数据处理

1. 读取 `poems.txt` 诗歌数据，每行为一首诗。
2. 为每首诗添加特殊起始符 `bos` 和结束符 `eos`，以便模型学习文本的边界。

3. 统计所有字符的出现频率，建立词汇表，将每个字符映射为索引，构建 `word2id`（字符到索引的映射）和 `id2word`（索引到字符的映射）。
4. 使用 `TensorFlow Dataset` 进行批处理，数据会被填充（padding）到相同长度。

2.1.2 模型构建

模型 `myRNNModel` 结构如下：

- **词嵌入层（Embedding）**：将字符索引转换为向量，向量维度为 64。
- **RNN 层（SimpleRNN）**：包含 128 个单元，处理序列数据，输出隐藏状态。
- **全连接层（Dense）**：输出词汇表大小的预测概率。

2.1.3 训练过程

1. 计算损失：
    - 使用 `sparse_softmax_cross_entropy_with_logits` 计算交叉熵损失。
    - 目标序列为输入序列的后移版本（即输入 `x` 生成目标 `y`）。
2. 优化器：
    - 采用 `Adam` 优化器，学习率设为 0.0005。
    - 通过 `GradientTape` 计算梯度并更新模型参数。
3. 训练：
    - 每个批次进行前向传播计算损失，共训练10个批次。
    - 反向传播计算梯度，更新模型权重。
    - 每隔 500 步打印一次损失值。

2.1.4 诗歌生成

1. 输入起始词（如“日”）：
    - 使用 `word2id` 将起始字符转换为索引。
    - 设定 RNN 的初始隐藏状态。
2. 逐步预测下一个字符：
    - 将前一个预测结果作为下一个时间步的输入。

- 通过 `argmax` 选取概率最高的词。
- 直到生成结束符 `eos` 或达到最大长度。

3. 转换为文本输出：
   - 使用 `id2word` 将索引转换回字符。
   - 拼接成完整诗句。

2.2 PyTorch版本

2.2.1 数据处理

1. 数据读取
   - 诗歌数据存储于 `poems.txt` 文件中，每行为一首诗。
   - 读取诗歌数据时，为每首诗添加特殊起始符 `start_token='G'` 和结束符 `end_token='E'`，确保模型能够识别文本边界。
   - 过滤特殊字符，如 `_ ( ) 《 》 [ ]`，去除过短（少于 5 字）或过长（超过 80 字）的诗歌。
2. 构建词汇表
   - 统计所有字符的出现频率，按照词频排序。
   - 构建 `word2id`（字符到索引的映射）和 `id2word`（索引到字符的映射），用于将文本转换为数字序列。
   - 诗歌数据转换为索引列表 `poems_vector`，用于训练。
3. 生成批量数据
   - 训练时使用 **batch_size=64**。
   - `generate_batch(batch_size, poems_vector, word_to_int)` 方法用于切分数据，每批次数据划分为 `x`（输入序列）和 `y`（目标序列）。
   - 目标序列 `y` 是输入序列 `x` 向右偏移一个时间步后的版本。

2.2.2 模型构建

本项目的 PyTorch 模型 `RNN_model` 主要包含以下结构：

1. 词嵌入层（**Embedding**）：

- `word_embedding` 类用于将字符索引转换为嵌入向量。
- 嵌入维度设为 `embedding_dim=100`。

2. **LSTM 层**：
   - `RNN_model` 采用 **2 层 LSTM**，隐藏层维度 `lstm_hidden_dim=128`。
   - 设定 `batch_first=True`，确保输入形状为 `(batch_size, seq_length, embedding_dim)`。
   - 初始化 **隐藏状态** `**h0**` **和 细胞状态** `**c0**` **为 0**。
3. **全连接层（Linear）**：
   - `fc` 负责将 LSTM 输出转换为词汇表大小的概率分布。
   - 使用 **ReLU 激活函数** 进行非线性变换。
   - 最终输出使用 **LogSoftmax** 计算类别概率。

2.2.3 训练过程

1. **损失函数**
   - 采用 **NLLLoss（负对数似然损失）**，适用于 LogSoftmax 输出。
2. **优化器**
   - 使用 **RMSprop** 进行梯度更新，学习率设为 `lr=0.01`。
   - 训练时使用 `clip_grad_norm` 限制梯度范数，避免梯度爆炸。
3. **训练循环（30 轮）**
   - 读取 `poems_vector` 并生成 `batches_inputs, batches_outputs`。
   - 遍历 `batch_size=100` 的数据，进行前向传播计算损失。
   - 反向传播计算梯度，更新模型权重。
   - 每 20 批次保存一次模型。

训练过程：

```
PS F:\Deep_Learning_Exercise\chap6_RNN\tangshi_for_pytorch> python main.py
finish  loadding data
inital linear weight
F:\Deep_Learning_Exercise\chap6_RNN\tangshi_for_pytorch\rnn.py:71: UserWarning: Implicit dimension choice for log_softmax has b
  out = self.softmax(out)
prediction [4357, 2064, 4357, 4103, 3236, 3236, 2869]
b_y        [28, 546, 104, 718, 1, 3, 3]
******************************
epoch   0 batch number 0 loss is:  8.718670845031738
main.py:165: UserWarning: torch.nn.utils.clip_grad_norm is now deprecated in favor of torch.nn.utils.clip_grad_norm_.
  torch.nn.utils.clip_grad_norm(rnn_model.parameters(), 1)
finish  save model
prediction [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
b_y        [267, 2888, 267, 2045, 0, 26, 1067, 26, 349, 1, 3, 3]
******************************
epoch   0 batch number 1 loss is:  7.481380939483643
prediction [117, 1, 2506, 32, 1, 4, 5116, 581, 3, 3, 2506, 72, 2506, 581]
b_y        [617, 564, 80, 601, 132, 0, 9, 184, 2007, 373, 305, 1, 3, 3]
******************************
epoch   0 batch number 2 loss is:  8.793425559997559
prediction [68, 0, 71, 71, 71, 71, 71, 71, 71, 71, 71, 138, 138, 1121, 0, 3, 3, 3]
b_y        [125, 9, 3862, 695, 16, 1721, 1721, 0, 373, 591, 246, 695, 900, 80, 431, 1, 3, 3]
******************************
epoch   0 batch number 3 loss is:  7.977275371551514
prediction [0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
b_y        [1539, 1058, 3996, 64, 0, 793, 977, 550, 381, 1, 3867, 3285, 972, 1287, 0, 1541, 1541, 1779, 1779, 1, 3, 3]
******************************
epoch   0 batch number 4 loss is:  9.17796516418457
prediction [18, 36, 36, 4, 9, 4, 51, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 4]
b_y        [194, 348, 66, 89, 73, 0, 574, 21, 855, 101, 1013, 1, 293, 162, 1283, 117, 383, 0, 25, 74, 4, 253, 611, 1, 3, 3]
******************************
epoch   0 batch number 5 loss is:  7.763591766357422
prediction [7, 7, 17, 7, 7, 0, 0, 63, 17, 7, 7, 7, 7, 7, 7, 17, 7, 7, 7, 7, 7, 7, 7, 7, 7]
b_y        [349, 351, 456, 220, 850, 0, 304, 146, 195, 350, 576, 1, 264, 178, 35, 323, 189, 0, 243, 12, 273, 61, 116, 1, 3, 3]
******************************
epoch   0 batch number 6 loss is:  7.795065879821777
prediction [8, 8, 1, 42, 0, 0, 0, 33, 1, 0, 0, 0, 0, 6, 6, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 6]
b_y        [18, 517, 56, 17, 191, 0, 1552, 1231, 88, 298, 61, 1, 57, 252, 311, 129, 259, 0, 397, 402, 412, 178, 347, 1, 3, 3]
******************************
epoch   0 batch number 7 loss is:  7.099910736083984
prediction [1, 1, 1, 1, 1, 1, 4, 1, 14, 1, 1, 1, 1, 1, 1, 14, 1, 1, 1, 3, 14, 1, 1, 1, 1, 1]
b_y        [3049, 18, 738, 137, 60, 0, 60, 127, 375, 36, 8, 1, 742, 20, 236, 117, 30, 0, 331, 108, 71, 21, 91, 1, 3, 3]
******************************
epoch   0 batch number 8 loss is:  6.97847318649292
prediction [133, 3, 133, 133, 133, 0, 133, 3, 133, 133, 133, 8, 3, 3, 3, 3, 133, 8, 3, 3, 3, 3, 133, 8, 3, 3]
b_y        [216, 16, 296, 57, 12, 0, 12, 88, 84, 4, 612, 1, 67, 41, 1838, 1198, 5, 0, 37, 95, 28, 26, 244, 1, 3, 3]
******************************
```

```
epoch   29 batch number 340 loss is:  6.15299129486084
finish  save model
prediction [10, 16, 46, 46, 84, 0, 9, 7, 4, 140, 107, 1, 10, 6, 5, 290, 43, 0, 7, 85, 17, 20, 113, 1, 110, 19, 9, 20, 469, 0, 4, 172, 47, 4, 151, 1, 15, 71, 4, 36, 290, 0, 4, 186, 77, 83, 71,
 1, 39, 8, 9, 9, 64, 0, 46, 23, 43, 54, 55, 1, 3, 6, 4, 416, 508, 0, 4, 39, 4, 5, 379, 1, 3, 3]
b_y        [1859, 1859, 179, 783, 783, 0, 84, 102, 240, 257, 616, 1, 183, 3536, 142, 256, 43, 0, 6, 614, 1169, 67, 123, 1, 951, 31, 19, 121, 64, 0, 4, 26, 27, 1018, 151, 1, 514, 1202, 672, 80
4, 1605, 0, 141, 316, 77, 107, 71, 1, 39, 29, 67, 9, 73, 0, 97, 168, 5, 977, 35, 1, 40, 440, 467, 225, 328, 0, 14, 403, 1164, 1112, 3196, 1, 3, 3]
******************************
epoch   29 batch number 341 loss is:  6.012487888336182
prediction [10, 51, 4, 51, 43, 0, 4, 124, 4, 172, 113, 1, 82, 30, 4, 9, 528, 0, 39, 5, 221, 4, 171, 1, 4, 215, 47, 10, 18, 0, 22, 119, 19, 138, 46, 1, 7, 47, 83, 18, 335, 0, 7, 165, 34, 40, 2
07, 1, 4, 34, 4, 34, 98, 0, 4, 288, 4, 22, 215, 1, 3, 104, 9, 51, 24, 0, 4, 172, 19, 162, 32, 1, 3, 3]
b_y        [1450, 384, 4, 51, 286, 0, 941, 82, 4, 51, 151, 1, 27, 489, 905, 110, 528, 0, 731, 2338, 221, 54, 35, 1, 1156, 281, 10, 216, 2526, 0, 6, 21, 337, 284, 46, 1, 299, 722, 214, 16, 202
, 0, 245, 29, 154, 178, 220, 1, 28, 1698, 2234, 77, 503, 0, 258, 3287, 703, 1900, 2258, 1, 504, 104, 4, 51, 1428, 0, 103, 1334, 22, 95, 171, 1, 3, 3]
******************************
epoch   29 batch number 342 loss is:  6.134788513183594
prediction [10, 318, 12, 74, 85, 0, 10, 186, 10, 12, 13, 1, 4, 41, 80, 283, 45, 0, 4, 109, 36, 508, 14, 1, 7, 119, 6, 17, 177, 0, 7, 177, 17, 15, 40, 1, 10, 74, 47, 12, 165, 0, 6, 12, 77, 207
, 133, 1, 10, 15, 9, 4, 100, 0, 7, 29, 82, 7, 46, 1, 4, 19, 84, 21, 18, 0, 9, 7, 4, 4, 5, 1, 3, 3]
b_y        [318, 318, 201, 74, 85, 0, 375, 186, 261, 60, 13, 1, 110, 1295, 115, 283, 28, 0, 622, 713, 1187, 770, 471, 1, 8, 196, 200, 54, 707, 0, 7, 344, 99, 123, 714, 1, 1307, 137, 277, 452,
 107, 0, 371, 690, 987, 347, 76, 1, 111, 38, 105, 2450, 393, 0, 99, 609, 1182, 2989, 121, 1, 101, 324, 214, 692, 789, 0, 46, 187, 61, 2528, 5, 1, 3, 3]
******************************
epoch   29 batch number 343 loss is:  6.215723514556885
prediction [10, 83, 9, 56, 48, 0, 12, 115, 77, 82, 204, 1, 7, 7, 79, 210, 8, 0, 82, 237, 4, 375, 13, 1, 82, 95, 9, 119, 8, 0, 119, 111, 77, 186, 14, 1, 7, 283, 138, 82, 186, 0, 7, 290, 17, 13
2, 343, 1, 4, 51, 9, 9, 153, 0, 89, 40, 77, 18, 5, 1, 15, 119, 47, 15, 51, 0, 15, 528, 43, 26, 305, 1, 3, 3]
b_y        [1279, 1279, 1310, 154, 72, 0, 206, 206, 43, 293, 1497, 1, 2085, 3029, 452, 209, 8, 0, 60, 237, 92, 546, 13, 1, 1219, 983, 7, 14, 177, 0, 220, 75, 964, 145, 76, 1, 137, 78, 444, 30
9, 245, 0, 178, 1762, 392, 87, 535, 1, 678, 357, 28, 260, 114, 0, 857, 11, 99, 1110, 121, 1, 6, 1523, 2101, 4, 51, 0, 35, 1775, 644, 922, 1979, 1, 3, 3]
******************************
epoch   29 batch number 344 loss is:  6.174333572387695
prediction [10, 7, 4, 44, 63, 0, 10, 112, 4, 56, 131, 1, 10, 17, 6, 15, 29, 0, 84, 21, 16, 145, 56, 1, 7, 105, 138, 4, 117, 0, 6, 45, 41, 138, 40, 1, 182, 48, 12, 9, 422, 0, 13, 217, 19, 19,
28, 1, 10, 32, 47, 4, 42, 0, 6, 18, 34, 9, 131, 1, 39, 17, 9, 23, 43, 0, 33, 185, 4, 39, 21, 1, 3, 3]
b_y        [196, 330, 69, 44, 70, 0, 581, 451, 18, 714, 457, 1, 8, 169, 5, 24, 73, 0, 6, 40, 100, 54, 626, 1, 185, 11, 32, 1148, 1148, 0, 249, 16, 63, 236, 236, 1, 118, 48, 168, 450, 378, 0,
6, 21, 101, 289, 615, 1, 22, 50, 259, 232, 252, 0, 33, 364, 117, 93, 594, 1, 233, 161, 88, 201, 943, 0, 86, 30, 43, 39, 632, 1, 3, 3]
******************************
epoch   29 batch number 345 loss is:  6.193174839019775
prediction [10, 18, 13, 7, 13, 0, 39, 8, 43, 22, 7, 1, 10, 6, 117, 4, 95, 0, 36, 272, 36, 5, 32, 1, 8, 18, 19, 6, 16, 0, 22, 32, 106, 106, 12, 1, 6, 16, 4, 103, 16, 0, 10, 165, 9, 17, 177, 1,
 10, 441, 32, 190, 8, 0, 4, 43, 36, 22, 6, 1, 3, 45, 55, 55, 64, 0, 42, 0, 10, 164, 138, 17, 4, 5, 5, 1, 3, 3]
b_y        [457, 152, 475, 147, 70, 0, 223, 8, 37, 313, 1117, 1, 302, 457, 231, 652, 1824, 0, 222, 704, 70, 147, 5, 1, 457, 48, 84, 88, 16, 0, 201, 72, 1766, 372, 1209, 1, 88, 16, 112, 594, 9
3, 0, 12, 57, 56, 23, 79, 1, 149, 278, 80, 309, 2542, 0, 28, 175, 254, 217, 1113, 1, 271, 488, 450, 135, 43, 15, 42, 0, 12, 65, 564, 475, 540, 157, 29, 1, 3, 3]
******************************
epoch   29 batch number 346 loss is:  6.281210422515869
prediction [10, 23, 9, 6, 91, 108, 0, 0, 10, 178, 4, 85, 10, 178, 79, 1, 4, 23, 13, 77, 79, 140, 140, 0, 10, 368, 9, 292, 4, 44, 23, 1, 10, 45, 4, 119, 140, 172, 51, 0, 83, 131, 94, 77, 4, 47
, 146, 1, 10, 5, 4, 341, 9, 22, 469, 0, 4, 117, 94, 82, 4, 10, 178, 1, 10, 39, 4, 151, 9, 108, 469, 0, 4, 62, 5, 117, 10, 6, 207, 1, 3, 3]
b_y        [188, 682, 804, 242, 143, 242, 70, 0, 10, 242, 297, 41, 10, 242, 100, 1, 342, 71, 191, 6, 240, 60, 154, 0, 555, 85, 1335, 1335, 69, 97, 107, 1, 139, 284, 325, 196, 4, 149, 95, 0, 3
6, 11, 99, 73, 144, 37, 381, 1, 38, 5, 112, 163, 1572, 241, 867, 0, 182, 7, 1076, 1076, 243, 134, 434, 1, 441, 70, 296, 457, 869, 108, 20, 0, 729, 131, 907, 117, 66, 6, 539, 1, 3, 3]
******************************
```

2.2.4 诗歌生成

1. 输入起始词（如“日”）
    - `gen_poem(begin_word)` 以 `begin_word` 作为起始字符。
    - 将字符转换为索引，并输入到 LSTM 模型。
2. 逐步预测下一个字符
    - 模型基于当前输入预测下一个字符的概率分布。
    - 采用 **argmax 选择概率最高的词**，并将其添加到诗歌序列中。
    - 直到生成结束符 `E` 或达到最大长度（30 字）。
3. 转换为文本输出
    - `to_word(predict, vocabs)` 将索引转换回字符。
    - `pretty_print_poem(poem)` 负责格式化诗歌，按 **整句**（逗号、句号）进行换行。
4. 示例
    - 调用 `gen_poem()` 生成以 日、红、山、夜、湖、海、月 开头的诗句。

---

3. 生成诗歌示例

使用 `日、红、山、夜、湖、海、月` 作为开头，生成诗歌如下：

3.1 TensorFlow版本

3.2 PyTorch版本

```
日日重吟，风光出处时。山光开处望，山色入长泉。已得长如此，何人
山光开处望，山色入长泉。
inital linear weight
红露中，风光夜夜空。玉阶金翠水，风月入清风。不是天台上，何人不
玉阶金翠水，风月入清风。
inital linear weight
山上花香。玉鞭不相见，不得不能同。何事无人迹，何人不得无。E
玉鞭不相见，不得不能同。
何事无人迹，何人不得无。
inital linear weight
夜吟，风清不同。不得已，终日中。明月，，歌和。日，歌清永，天下
inital linear weight
湖水头草，红霞不见春。花开不见水，一日在东风。今日无人迹，无心
花开不见水，一日在东风。
inital linear weight
海皆见人，不得在何时。谁能此时心，不是天下人。E
谁能此时心，不是天下人。
inital linear weight
月飞春，风起月中春。风清风，香。风光，，歌舞，风，风，风，风，
风光，，歌舞，风，风，风，风，。
inital linear weight
君断望，水木夜中来。一夜秋风起，秋光入夜长。何人得无事，不见一
一夜秋风起，秋光入夜长。
```