RNN 唐诗生成任务报告

2252334 孙毓涵

1. RNN,LSTM,GRU 模型

（1） RNN

核心思想：通过循环结构处理序列数据，保留历史信息。

结构：

- 每个时间步接受当前输入和上一时刻的隐藏状态，输出当前隐藏状态和预测结果。

  公式：

$$h_t=\tanh(W_{xh}x_t+W_{hh}h_{t-1}+b_h)$$

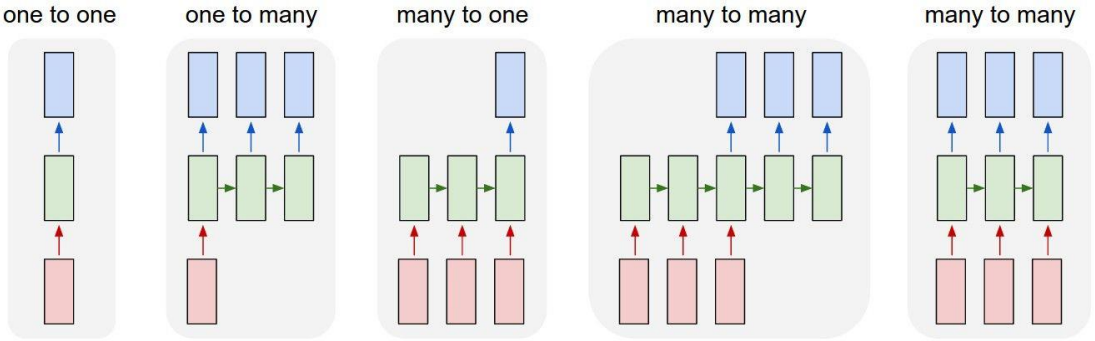
$$y_t=W_{hy}h_t+b_y$$

优点：

- 能够处理变长序列，捕捉时间依赖性。

缺点：

- **梯度消失/爆炸**：长序列训练时，梯度可能指数级衰减或增长。
- **短期记忆**：难以学习远距离依赖关系。



（2） LSTM

核心思想：引入门控机制，选择性保留或遗忘信息。

关键结构：

- **遗忘门**：决定丢弃多少旧信息。

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

- **输入门**：决定更新哪些新信息。

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

$$\tilde{C}_t=\tanh(W_C\cdot[h_{t-1},x_t]+b_C)$$

- **细胞状态**：长期记忆的存储与更新。

$$C_t=f_t\odot C_{t-1}+i_t\odot\tilde{C}_t$$

- **输出门**：控制当前隐藏状态的输出。

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

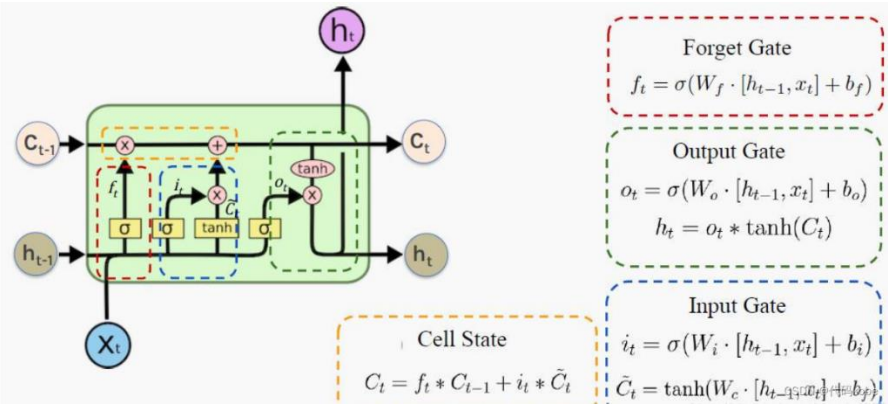
$$h_t=o_t\odot\tanh(C_t)$$

优点：

- 解决梯度消失问题，擅长学习长距离依赖。
- 适用于复杂序列建模（如机器翻译、语音识别）。

缺点：

- 参数较多，计算成本高。



（3） GRU

核心思想：简化 LSTM，合并部分门控结构，保持性能。

关键结构：

- **重置门**：控制历史信息的忽略程度。

  $$r_t=\sigma(W_r\cdot[h_{t-1},x_t]+b_r)$$

- **更新门**：平衡新旧信息的比例。

  $$z_t=\sigma(W_z\cdot[h_{t-1},x_t]+b_z)$$

- **候选隐藏状态**：结合当前输入和部分历史信息。

  $$\tilde{h}_t=\tanh(W\cdot[r_t\odot h_{t-1},x_t]+b)$$

- **最终隐藏状态**：

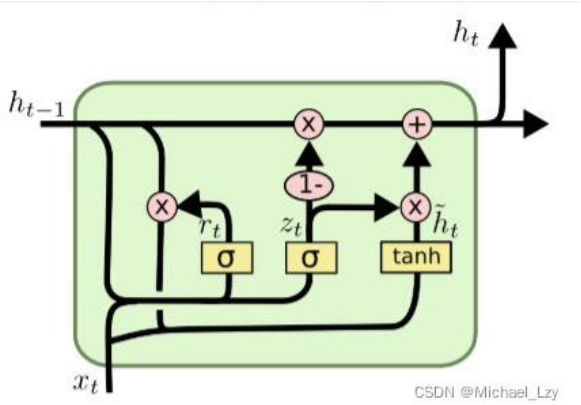
  $$h_t=(1-z_t)\odot h_{t-1}+z_t\odot\tilde{h}_t$$

优点：

- 参数比 LSTM 少，训练更快。
- 在多数任务中表现接近 LSTM。

缺点：

- 极长序列中可能略逊于 LSTM。



2. 诗歌生成过程

诗歌生成的过程主要基于循环神经网络（RNN）和长短期记忆网络（LSTM），以下是详细的步骤描述：

（1）数据预处理

- **输入数据**：从文本文件（如 poems.txt）中读取诗歌数据。
- **清洗数据**：
  - 去除空格、标点符号等无关字符。
  - 过滤掉包含特殊符号（如 _、(、《 等）的诗歌。
  - 限制诗歌长度在 5 到 80 个字符之间。
- **添加标记**：在每首诗歌的开头和结尾分别添加起始标记 G 和结束标记 E。
- **构建词汇表**：
  - 统计所有字符的出现频率，生成词汇表。
  - 将每个字符映射为一个唯一的整数索引，形成 word_int_map。
- **向量化**：将每首诗歌转换为整数序列（poems_vector），便于模型处理。

（2）模型构建

- **词嵌入层**：
  - 使用 word_embedding 类将字符索引映射为固定维度的词向量。

- **LSTM 层：**
  - 使用两层 LSTM 网络，输入为词向量，输出为隐藏状态。
  - LSTM 的隐藏状态维度为 lstm_hidden_dim。
- **全连接层：**
  - 将 LSTM 的输出通过全连接层映射到词汇表大小的向量。
- **激活函数：**
  - 使用 LogSoftmax 作为输出层的激活函数，生成每个字符的概率分布。

（3）模型训练

- **数据分批次：**
  - 将 poems_vector 分成多个批次（batch_size），每个批次包含输入数据 x_batches 和目标数据 y_batches。
- **损失函数：**
  - 使用负对数似然损失（NLLLoss）计算模型输出与目标字符之间的误差。
- **优化器：**
  - 使用 RMSprop 优化器更新模型参数。
- **训练过程：**
  - 遍历所有批次，计算损失并反向传播更新模型参数。
  - 每隔一定批次保存模型参数到文件（如 poem_generator_rnn）。

（4）诗歌生成

- **加载模型：**
  - 从保存的文件中加载训练好的模型参数。
- **生成过程：**
  - 给定一个起始字符（如 "日"），将其转换为整数索引并输入模型。
  - 模型预测下一个字符的概率分布，选择概率最高的字符作为输出。
  - 将预测的字符追加到当前诗歌中，并作为下一个输入。
  - 重复上述过程，直到生成结束标记 E 或达到最大长度。
- **结果输出：**
  - 将生成的字符序列转换为诗歌文本，并格式化输出。

3. 诗歌生成

生成诗歌 开头词汇是 “ 日 、 红 、 山 、 夜 、 湖、 海 、 月 ”。

```
pretty_print_poem(gen_poem("日"))
pretty_print_poem(gen_poem("红"))
pretty_print_poem(gen_poem("山"))
pretty_print_poem(gen_poem("夜"))
pretty_print_poem(gen_poem("湖"))
pretty_print_poem(gen_poem("海"))
pretty_print_poem(gen_poem("月"))
```

```
(tf2) D:\TongjiUniveristy\MAJOR\NeuralNetwork & DeepLearning\Neural-Networks-and-Deep-Learning-Assignment\Assignment3\tangshi_for_pytorch>python
 main.py
error
inital  linear weight
一夜一相见，一生无一人。
何当一夜客，一别一相逢。
error
inital  linear weight
一夜一相思，一夜千万年。
error
inital  linear weight
晚来千万岁，一径一相逢。
error
inital  linear weight
今来不相见，不见此时生。
不见东风至，何人不见人。
error
inital  linear weight
山中有风生，水上千条水。
error
inital  linear weight
一夜一相望，一生无一人。
何当一相见，不见一年人。
error
inital  linear weight
一夜一相望，一生千万人。
```

原因：在 def pretty_print_poem(poem)打印函数中，只选择打印长度大于 10 的句子，以特定汉字开头的句子被忽略。如果不要长度大于 10 的条件：

```
(tf2) D:\TongjiUniveristy\MAJOR\NeuralNetwork & DeepLearning\Neural-Networks-and-Deep-Learning-Assignment\Assignment3\tangshi_for_pytorch>python
 main.py
error
inital  linear weight
日出山头。
一夜一相见，一生无一人。
何当一夜客，一别一相逢。
E。
error
inital  linear weight
红香，风潮含夜风。
一夜一相思，一夜千万年。
E。
error
inital  linear weight
山。
晚来千万岁，一径一相逢。
E。
error
inital  linear weight
夜长风。
今来不相见，不见此时生。
不见东风至，何人不见人。
E。
error
inital  linear weight
湖水一相亲。
山中有风生，水上千条水。
E。
error
inital  linear weight
海明月明。
一夜一相望，一生无一人。
何当一相见，不见一年人。
E。
error
inital  linear weight
月明，秋风生碧水。
一夜一相望，一生千万人。
E。
```

训练过程截图：

```
******************************
epoch   13 batch number 253 loss is:  6.305886745452881
prediction [10, 23, 10, 29, 29, 27, 14, 0, 10, 190, 320, 5, 4, 19, 27, 1, 6, 108, 4, 6, 12, 16, 17, 0, 25, 562, 71, 937, 77, 12, 116, 1, 6, 14, 6, 9
5, 9, 7, 74, 0, 4, 51, 83, 37, 77, 45, 14, 1, 4, 19, 26, 495, 9, 54, 51, 0, 15, 7, 15, 72, 106, 318, 352, 1, 3, 3]
b_y        [190, 23, 24, 45, 693, 32, 423, 0, 80, 393, 9, 5, 110, 31, 414, 1, 269, 271, 368, 162, 478, 75, 17, 0, 736, 1085, 1190, 550, 349, 134, 11
04, 1, 769, 231, 2040, 966, 181, 1193, 74, 0, 2192, 644, 915, 748, 753, 1468, 38, 1, 851, 19, 1475, 129, 138, 54, 878, 0, 13, 7, 392, 114, 8, 40, 28
, 1, 3, 3]
******************************
epoch   13 batch number 254 loss is:  6.291411399841309
prediction [10, 8, 15, 165, 4, 26, 318, 0, 4, 41, 28, 35, 10, 86, 21, 1, 10, 64, 4, 58, 9, 54, 131, 0, 6, 58, 7, 18, 4, 25, 115, 1, 6, 86, 4, 45, 98
, 22, 104, 0, 4, 16, 6, 21, 77, 34, 28, 1, 68, 117, 26, 187, 15, 16, 161, 0, 15, 98, 7, 64, 10, 26, 32, 1, 3, 3]
b_y        [68, 87, 1168, 272, 212, 318, 318, 0, 4, 141, 282, 313, 667, 660, 2015, 1, 5, 273, 436, 14, 24, 911, 18, 0, 252, 94, 129, 29, 1380, 1047,
 115, 1, 89, 62, 63, 61, 513, 841, 267, 0, 1361, 82, 217, 78, 853, 34, 28, 1, 117, 73, 555, 331, 25, 74, 159, 0, 99, 111, 121, 4271, 1065, 525, 116,
 1, 3, 3]
******************************
epoch   13 batch number 255 loss is:  6.321417808532715
prediction [10, 62, 71, 159, 10, 6, 21, 0, 10, 74, 9, 6, 4, 665, 12, 1, 97, 245, 4, 75, 9, 106, 91, 0, 7, 75, 46, 18, 16, 12, 79, 1, 4, 495, 4, 77,
9, 25, 45, 0, 4, 51, 5, 5, 4, 19, 5, 1, 15, 19, 26, 5, 9, 6, 16, 0, 15, 5, 15, 41, 41, 23, 35, 1, 3, 3]
b_y        [388, 62, 58, 660, 130, 6, 118, 0, 202, 126, 60, 386, 106, 72, 206, 1, 1819, 1174, 3113, 824, 12, 1046, 248, 0, 329, 2766, 634, 1093, 153
, 295, 78, 1, 3501, 3502, 347, 16, 672, 301, 425, 0, 168, 1159, 205, 279, 195, 286, 384, 1, 101, 19, 173, 5, 2983, 3341, 1756, 0, 139, 30, 430, 51,
10, 28, 156, 1, 3, 3]
******************************
epoch   13 batch number 256 loss is:  6.317513942718506
prediction [10, 8, 6, 51, 22, 14, 35, 0, 4, 41, 28, 27, 4, 70, 5, 1, 4, 63, 6, 37, 12, 203, 8, 0, 10, 19, 10, 19, 203, 23, 23, 1, 7, 131, 12, 108, 1
5, 15, 63, 0, 4, 72, 7, 6, 77, 293, 71, 1, 4, 19, 26, 58, 4, 51, 197, 0, 15, 58, 4, 72, 19, 28, 35, 1, 3, 3]
b_y        [155, 136, 439, 761, 70, 282, 35, 0, 376, 1297, 452, 410, 51, 598, 151, 1, 258, 267, 101, 180, 80, 227, 1561, 0, 851, 501, 251, 503, 203,
 69, 533, 1, 93, 200, 1573, 2030, 24, 22, 3191, 0, 141, 210, 641, 662, 77, 679, 783, 1, 31, 1710, 83, 481, 28, 18, 472, 0, 548, 1298, 204, 44, 34, 1
274, 92, 1, 3, 3]
******************************
epoch   13 batch number 257 loss is:  6.389115333557129
prediction [10, 145, 6, 216, 34, 16, 20, 0, 7, 79, 10, 95, 16, 12, 12, 1, 10, 21, 4, 17, 9, 644, 116, 0, 16, 106, 9, 20, 69, 18, 21, 1, 4, 7, 4, 21,
 9, 21, 14, 0, 25, 14, 12, 165, 16, 21, 124, 1, 4, 51, 4, 29, 9, 21, 83, 0, 4, 120, 15, 14, 9, 41, 134, 1, 3, 3]
b_y        [53, 964, 22, 214, 35, 192, 475, 0, 12, 164, 103, 771, 609, 484, 270, 1, 239, 296, 190, 44, 2197, 644, 116, 0, 146, 165, 10, 59, 6, 16, 2
07, 1, 13, 2775, 299, 912, 16, 563, 1576, 0, 161, 297, 989, 473, 224, 32, 82, 1, 1967, 3426, 24, 586, 17, 167, 657, 0, 472, 327, 92, 78, 4, 1421, 66
3, 1, 3, 3]
******************************
```

4. 试验总结

本试验基于 LSTM 构建诗歌生成模型，通过清洗 poems.txt 数据并构建字符索引映射，设计双层 LSTM 网络结合词嵌入层进行训练。针对模型加载时的设备报错问题，采用 map_location=torch.device('cpu')解决；修复 LogSoftmax 维度设置后，生成诗歌虽符合五言/七言格式，但仍存在语义跳跃及重复问题（如“一别一相逢”“不见此时生”），主要受限于训练数据规模小、模型容量不足及贪心搜索策略。后续可通过扩充数据集、引入注意力机制及温度采样优化生成多样性和逻辑连贯性。