# 使用 RNN 生成唐诗报告

2254275 陶鸿周

## 一、RNN ，LSTM，GRU 模型

### 1. RNN

#### 1.1 简介

RNN（Recurrent Neural Network）是一种专门处理序列数据（如时间序列、文本、语音）的神经网络。其核心特点是循环结构，允许信息在时间步之间传递，能够捕捉序列中的短期依赖关系。但传统 RNN 存在梯度消失/爆炸问题，难以处理长序列依赖。

#### 1.2 主要组成部分

- 输入层（Input）：接收当前时间步的输入数据（如词向量）。
- 隐藏层（Hidden State）：存储历史信息，通过循环连接传递到下一时间步。
- 输出层（Output）：基于隐藏状态生成当前时间步的预测结果。
- 循环权重矩阵：控制隐藏状态在不同时间步之间的信息传递。

#### 1.3 工作流程

- 初始化：隐藏状态 $h_0$ 通常初始化为零向量。
- 循环计算：对于每个时间步 t，输入 $x_t$ 与上一隐藏状态 $h_{t-1}$ 结合，通过激活函数（如 tanh）生成新隐藏状态：$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$
- 输出 $y_t$ 由隐藏状态映射得到：$y_t = w_{hy}h_t + b_y$
- 序列处理：重复上述步骤直至序列结束。

### 2. LSTM

#### 2.1 简介

LSTM（Long Short-Term Memory）是 RNN 的改进版本，通过门控机制（遗忘门、输入门、输出门）解决长序列依赖问题。核心创新是引入细胞状态（Cell State），可长期保存关键信息。

2.2 主要组成部分

- 细胞状态（$C_t$）：贯穿整个序列的“记忆通道”，受门控机制保护。
- 遗忘门（Forget Gate）：决定从细胞状态中丢弃哪些信息。
- 输入门（Input Gate）：控制新信息写入细胞状态的比例。
- 输出门（Output Gate）：基于细胞状态生成当前隐藏状态。

2.3 工作流程

- 遗忘门：计算丢弃信息的比例：$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
- 输入门：
  - 生成候选信息：$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
  - 更新比例：$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
  - 更新细胞状态：$C_t = f_t \odot \mathrm{C}_{\mathrm{t\text{-}1}} + i_t \odot \tilde{\mathrm{C}}_t$
- 输出门：
  - 生成隐藏状态：$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), h_t = o_t \odot \tanh(\mathrm{C_t})$

3. GRU

3.1 简介

GRU（Gated Recurrent Unit）是 LSTM 的简化版本，通过合并细胞状态和隐藏状态，减少参数数量。包含更新门（Update Gate）和重置门（Reset Gate），在保持性能的同时提高计算效率。

3.2 主要组成部分

更新门（$z_t$）：控制历史信息与新信息的融合比例。

重置门（$r_t$）：决定忽略多少历史信息。

候选隐藏状态（$\tilde{h}_t$）：基于当前输入和重置门生成临时状态。

3.3 工作流程

• 门控计算：

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

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

• 候选隐藏状态：

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

• 状态更新：

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

## 二、诗歌生成的过程

1. 数据预处理部分：这部分用到了函数 process_poems1。

- process_poems1：从指定的文件（这里是./poems.txt）中加载诗歌数据，对其做一些预处理，包括：
  - 若出现"_"，"("等无用字符或内容长度不在规定范围内，则跳过；
  - 符合要求的数据则加上起始符和结束符后加入数据集。
  - 统计所有字符的词频，建立 word_int_map（字符到索引的映射）和反向词汇表 words（索引到字符的映射）。
  - 将每首诗歌转换为索引序列 poems_vector（如 [G，日，落, E] → [0, 10, 25, 1]）

2. 模型训练部分：这部分用到了函数 run_training。

- run_training：实现了模型的训练过程，主要包含的步骤如下：
  - 使用函数 process_poems1 加载并预处理训练数据。
  - 使用 word_embedding 将字符索引映射为词向量
  - 定义双层 LSTM 模型 RNN_model，输入为词向量，输出为下一个字符的概率分布。
  - 通过 generate_batch 将 poems_vector 切分为输入 x_data 和目标 y_data（y_data 是 x_data 右移一位的结果）。
  - 对每个批次，计算模型预测值与真实值的交叉熵损失
  - 反向传播优化参数，使用梯度裁剪防止梯度爆炸。

- 每 20 个批次保存一次模型参数（poem_generator_rnn）。

3. 生成诗歌：这部分用到了函数 gen_poem。

- gen_poem：从起始字开始，逐字生成诗歌。
  - 加载预训练的 RNN_model 和词汇表。
  - 将起始字转换为索引序列（如 "日" → [0, 10]，包含起始符 G）。
  - 将当前序列输入模型，模型输出下一个字符的概率分布。
  - 将预测得出的字加入诗歌，循环以上过程直到生成结束符或者超过 30 个字符。

4. 输出诗歌：这部分用到了函数 to_word 和 pretty_print_poem

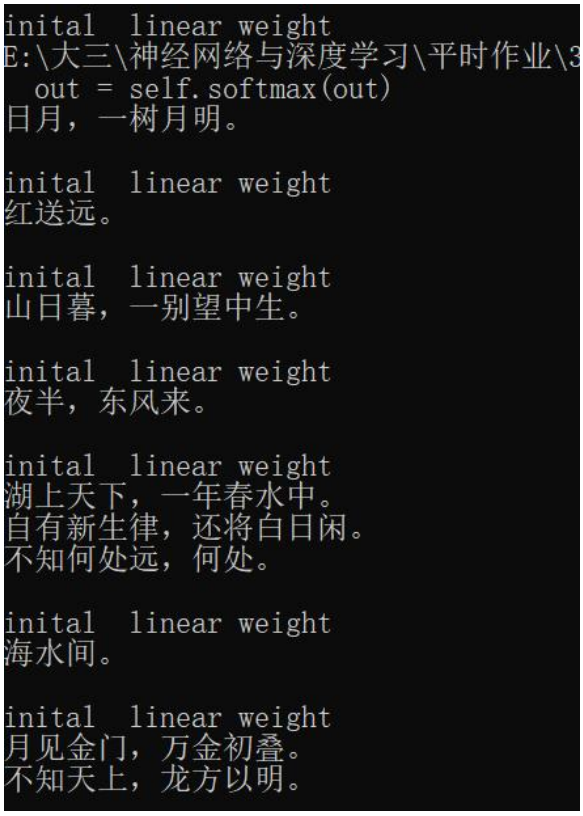
- to_word：将模型输出的概率分布映射回字符，选择最大概率对应的字符。若索引超出词汇表，则取最后一个字符（兜底逻辑）。
- pretty_print_poem：按句号 。 分句，确保诗句工整。

## 三、截图展示

1. 训练过程

```
prediction [3236, 2826, 3868, 3868, 2057, 2057, 2057]
b_y        [28, 546, 104, 718, 1, 3, 3]
********************************
epoch   0 batch number 0 loss is:  8.718445777893066
E:\大三\神经网络与深度学习\平时作业\3\main.py:164: UserWarning: torch.nn.util
  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:  7.454458713531494
prediction [2064, 3, 3, 3, 5676, 69, 4029, 186, 374, 275, 374, 157, 1726, 157
b_y        [617, 564, 80, 601, 132, 0, 9, 184, 2007, 373, 305, 1, 3, 3]
********************************
epoch   0 batch number 2 loss is:  8.592375755310059
prediction [157, 67, 67, 67, 67, 67, 67, 67, 67, 67, 67, 67, 157, 157, 15
b_y        [125, 9, 3862, 695, 16, 1721, 1721, 0, 373, 591, 246, 695, 900, 80
********************************
epoch   0 batch number 3 loss is:  8.167795181274414
prediction [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
b_y        [1539, 1058, 3996, 64, 0, 793, 977, 550, 381, 1, 3867, 3285, 972,
********************************
epoch   0 batch number 4 loss is:  7.189705848693848
prediction [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
b_y        [194, 348, 66, 89, 73, 0, 574, 21, 855, 101, 1013, 1, 293, 162, 12
********************************
epoch   0 batch number 5 loss is:  7.740779399871826
prediction [99, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3
b_y        [349, 351, 456, 220, 850, 0, 304, 146, 195, 350, 576, 1, 264, 178,
********************************
```

2. 生成结果


inital  linear weight
E:\大三\神经网络与深度学习\平时作业\3
out = self.softmax(out)
日月，一树月明。
inital  linear weight
红送远。
inital  linear weight
山日暮，一别望中生。
inital  linear weight
夜半，东风来。
inital  linear weight
湖上天下，一年春水中。
自有新生律，还将白日闲。
不知何处远，何处。
inital  linear weight
海水间。
inital  linear weight
月见金门，万金初叠。
不知天上，龙方以明。