基于循环神经网络的诗歌生成模型分析报告

徐志铭　2251804

1　循环神经网络系列模型分析

1.1　循环神经网络 RNN

循环神经网络（RNN）是为处理序列数据而设计的神经网络。与前馈神经网络不同，RNN 拥有内部的记忆状态，允许信息在网络中持续存在，使其能够捕捉序列中的时间依赖性。RNN 的核心思想是利用一个循环更新的隐藏状态 h_t 来集成序列的历史信息。其基本结构展示在图 1中。

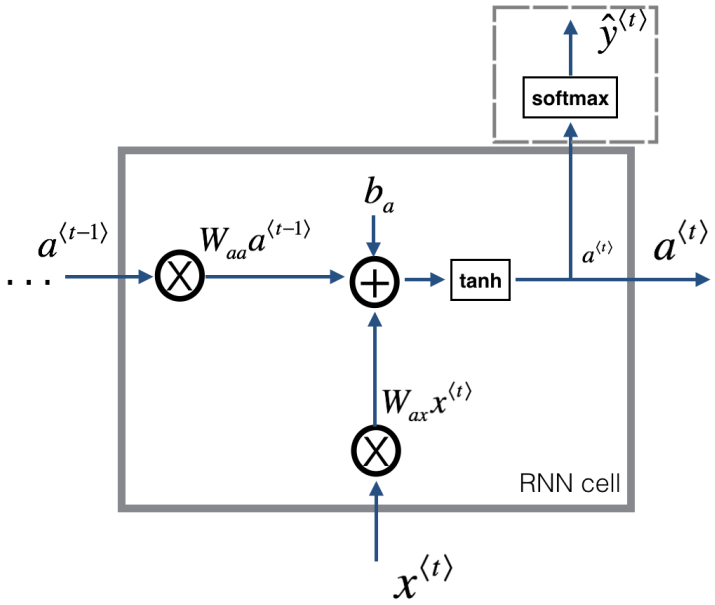


图 1: RNN 单元示意图

1.1.1　前向传播公式

在时间步 t，RNN 的隐藏状态 h_t 由当前输入 x_t 和前一时间步的隐藏状态 h_{t-1} 计算得出。输出 y_t 则由当前的隐藏状态 h_t 生成。

- **隐藏状态更新:**

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

其中，W_{hh} 是隐藏层到隐藏层的权重矩阵，W_{xh} 是输入层到隐藏层的权重矩阵，b_h 是隐藏层的偏置项。tanh 是双曲正切激活函数。

- **输出计算:**

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

其中，W_{hy} 是隐藏层到输出层的权重矩阵，b_y 是输出层的偏置项。

1.2 长短期记忆网络 LSTM

标准 RNN 在处理长序列时会面临梯度消失或梯度爆炸问题，难以学习长距离依赖。LSTM 通过引入一个更复杂的单元结构来解决此问题，该结构包含一个细胞状态（cell state）和三个门（输入门、遗忘门、输出门），其基本结构展示在图 2中。

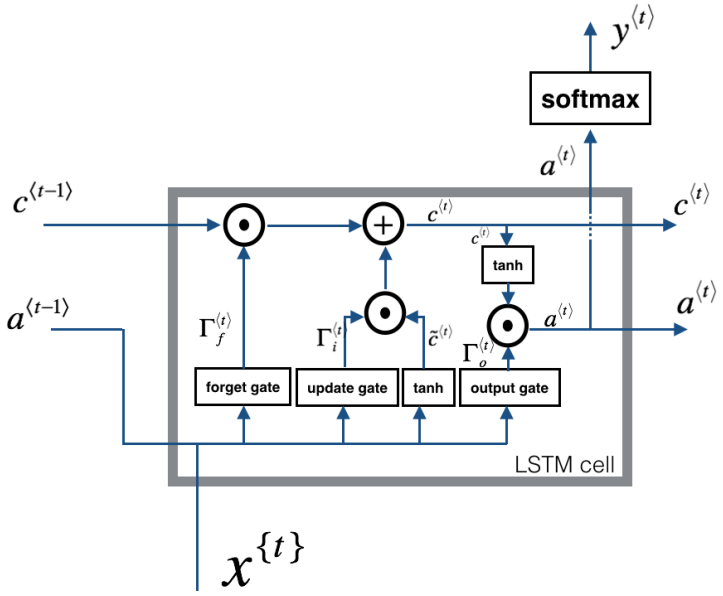


图 2: LSTM 单元示意图

1.2.1 前向传播公式

LSTM 的核心是细胞状态 c_t，门的结构允许网络有选择性地向细胞状态中添加或移除信息，从定性层面上支持了信息的长距离传递。

- **遗忘门 (Forget Gate):** 决定从前一细胞状态 c_{t-1} 中丢弃多少信息。

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

- **输入门 (Input Gate):** 决定哪些新信息将被存入细胞状态。

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

- **候选细胞状态 (Candidate Cell State):** 创建可能被添加的新信息。

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

- **细胞状态更新 (Cell State Update):** 结合遗忘和输入信息，更新细胞状态。

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

- **输出门 (Output Gate):** 决定细胞状态的哪些部分将被输出。

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

- **隐藏状态更新 (Hidden State Update):**

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

其中 $\odot$ 表示 Hadamard 积。

1.3 门控循环单元 GRU

GRU 是 LSTM 的一种简化变体，它将遗忘门和输入门合并为一个 “更新门”，并融合了细胞状态和隐藏状态。这使得 GRU 的结构更简单，参数更少，但在相同任务中能达到和 LSTM 较为相近的效果。

1.3.1 前向传播公式

GRU 包含两个门：重置门和更新门。

- **重置门 (Reset Gate):** 决定在计算候选隐藏状态时，多大程度上忽略前一隐藏状态。

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

- **更新门 (Update Gate):** 决定新旧隐藏状态的组合比例。

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

- **候选隐藏状态 (Candidate Hidden State):**

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

- **隐藏状态更新 (Hidden State Update):**

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

1.4 模型演变过程

RNN、LSTM 和 GRU 的演变是为解决序列建模中长程依赖问题而驱动的。

1. **从 RNN 到 LSTM:** 原始 RNN 因梯度消失/爆炸问题难以学习长距离依赖。LSTM 通过引入门控机制和独立的细胞状态，有效地控制信息流，极大地缓解了梯度消失问题，从而能够捕捉长程依赖关系。

2. **从 LSTM 到 GRU:** GRU 是 LSTM 的简化版本。它将 LSTM 的三个门简化为两个，合并了细胞状态和隐藏状态，减少了模型参数和计算复杂度。在许多任务中，GRU 能以更快的训练速度达到与 LSTM 相媲美的性能。

2 诗歌生成过程分析

本节叙述诗歌生成过程中的核心步骤。

2.1 数据预处理

1. **加载与清洗:** 从文本文件 (`poems.txt`) 中加载诗歌。脚本将一条数据分割为标题和内容，并移除内容中的空格。同时，过滤掉包含特殊符号或长度不合规（过长或过短）的诗句。

2. **添加标记:** 在每首诗歌的开头和结尾分别添加起始符 G 和结束符 E，这有助于模型学习诗歌的边界。

3. **构建词典与向量化:** 统计所有汉字的出现频率，构建一个从汉字到整数索引的映射。随后，将每首诗歌转换为一个整数序列（类似于大模型中的 `tokenizer`）。

2.2 构建训练批次

对于每个输入序列 `x`，生成一个对应的目标序列 `y`，其是 `x` 向左平移一个时间步的结果。模型的目标是根据当前及之前的字符序列，预测下一个字符。

2.3 模型构建与训练

1. **模型架构:** 模型 (`rnn.RNN_model`) 主要由一个词嵌入层和一个 LSTM 层构成。词嵌入层将输入的整数索引转换为低维稠密的词向量，LSTM 层负责捕捉序列中的时序依赖关系。

2. **训练过程:**

- **序列填充:** 由于批次内的诗歌长度不一，训练时使用 `pad_sequence` 函数对序列进行填充，使其长度一致（对原始的训练脚本进行了改进，原来的训练逻辑 `batch_size` 为 1，训练效率较低）。
- **损失函数:** 采用负对数似然损失 (`torch.nn.NLLLoss`)，并通过设置 `ignore_index` 来忽略填充部分对损失的贡献。
- **优化:** 训练循环中，模型进行前向传播得到预测，计算损失后进行反向传播，并使用 Adam 优化器更新模型参数。

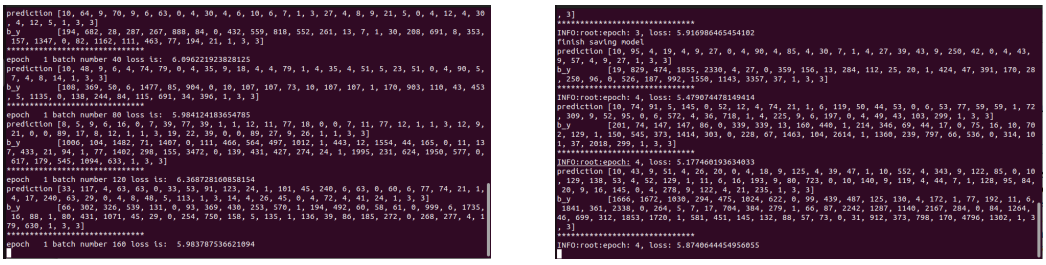
3. **模型保存:** 脚本支持从检查点恢复训练，并在训练过程中定期保存模型状态。

2.4 诗歌生成 (推理)

1. **生成过程:** `gen_poem` 函数执行生成任务。它以一个用户指定的汉字作为初始输入。
2. **循环预测:** 函数进入一个循环。在每一步，它将当前已生成的序列作为输入，预测下一个最可能的汉字)，然后将该汉字追加到序列末尾。
3. **终止条件:** 当模型预测出结束符 'E' 或生成长度达到上限时，循环终止。最终对生成的诗歌进行格式化输出。

3 训练结果

给出训练初期和后期的结果截图在图 3中。最后利用 "日、红、山、夜、湖、海、月" 分别作为开头词汇生成的诗歌结果展示在图 4中。



(a) (b)

图 3: 训练过程截图。(a) 训练初期；(b) 训练后期

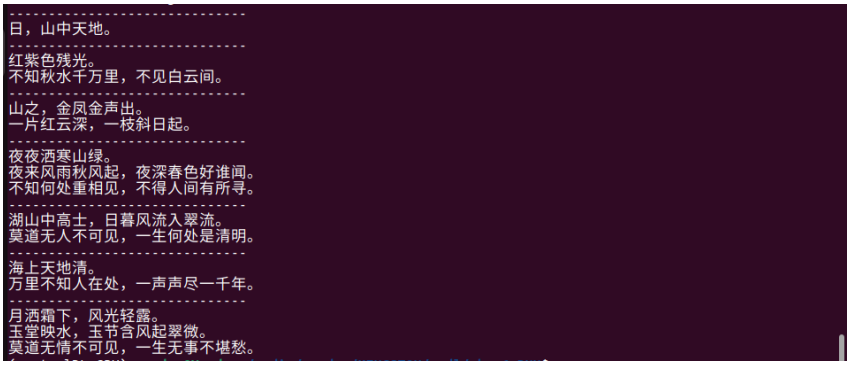


图 4: 指定开头词诗歌生成结果