

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。

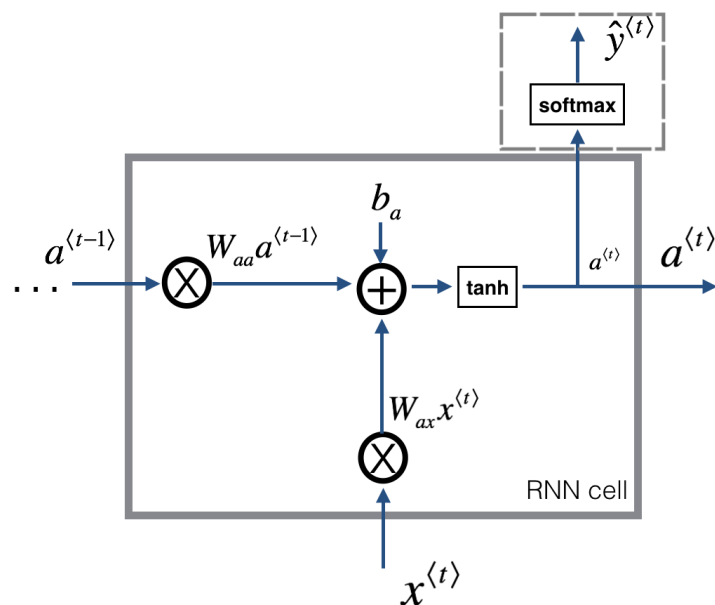


图 1: RNN 单元示意图

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:

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

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。

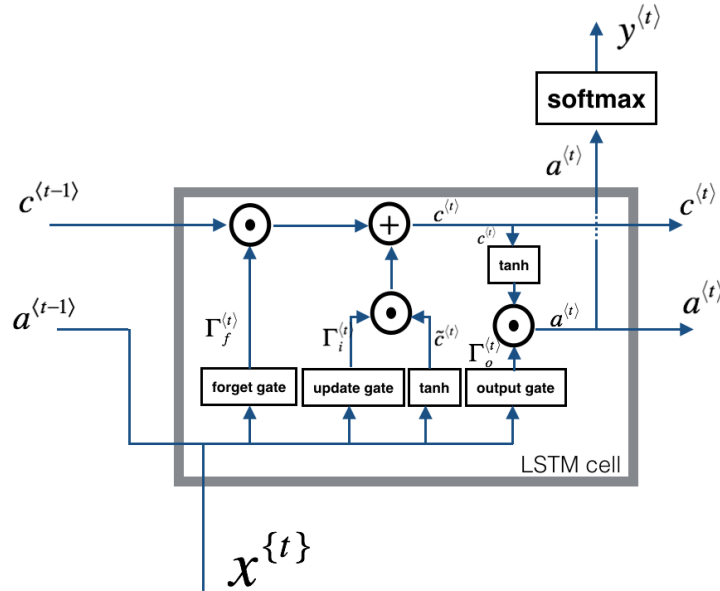


图 2: LSTM 单元示意图

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。

```
prediction [10, 64, 9, 70, 9, 0, 63, 0, 4, 30, 4, 0, 10, 0, 7, 1, 3, 27, 4, 0, 9, 21, 5, 0, 4, 12, 4, 30
, 4, 12, 5, 1, 3, 3]
b.y [194, 682, 28, 287, 287, 888, 884, 0, 432, 559, 818, 552, 261, 13, 7, 1, 30, 208, 691, 0, 353,
127, 1347, 0, 62, 1152, 111, 403, 77, 104, 21, 1, 3, 3]
*****
epoch 1 batch number 40 loss ts: 6.09021923828125
prediction [10, 48, 9, 6, 4, 74, 79, 0, 4, 35, 9, 18, 4, 4, 79, 1, 4, 35, 4, 51, 5, 23, 51, 0, 4, 90, 5,
7, 4, 0, 14, 1, 3, 3]
b.y [180, 360, 50, 6, 1477, 85, 904, 0, 10, 107, 107, 73, 10, 107, 107, 1, 170, 903, 110, 43, 453
, 5, 1135, 0, 138, 244, 84, 115, 691, 34, 396, 1, 3, 3]
*****
epoch 1 batch number 80 loss ts: 5.984124183654785
prediction [0, 9, 0, 16, 0, 7, 39, 77, 29, 1, 4, 12, 11, 77, 18, 0, 0, 7, 11, 77, 12, 1, 1, 3, 12, 9,
21, 0, 0, 89, 17, 8, 12, 1, 1, 3, 19, 22, 39, 0, 6, 89, 27, 9, 26, 1, 1, 3, 3]
b.y [1000, 104, 1482, 71, 1487, 0, 111, 468, 564, 407, 1012, 1, 443, 12, 1254, 44, 165, 0, 11, 43
7, 433, 21, 94, 1, 77, 1482, 208, 155, 3072, 0, 130, 431, 427, 274, 24, 1, 1595, 131, 624, 1050, 577, 0,
617, 179, 545, 1094, 633, 1, 3, 3]
*****
epoch 1 batch number 120 loss ts: 6.368728108981514
prediction [23, 117, 4, 63, 69, 0, 39, 59, 91, 223, 24, 1, 101, 45, 240, 6, 63, 0, 69, 6, 77, 74, 21, 1,
4, 17, 240, 63, 25, 0, 4, 0, 48, 5, 113, 1, 3, 14, 4, 26, 45, 0, 4, 72, 4, 41, 24, 1, 3, 3]
b.y [66, 302, 326, 539, 131, 0, 93, 369, 430, 253, 570, 1, 104, 492, 68, 50, 61, 0, 999, 0, 1735,
16, 88, 1, 89, 431, 1071, 45, 79, 0, 254, 959, 150, 5, 135, 1, 130, 38, 80, 185, 272, 0, 208, 277, 4, 1
79, 638, 1, 3, 3]
*****
epoch 1 batch number 160 loss ts: 5.983787536621894
```

(a)

```
, 3]
*****
INFO:root:epoch: 3, loss: 5.916986465454102
finish saving model
prediction [10, 95, 4, 19, 4, 9, 27, 0, 4, 90, 4, 85, 4, 30, 7, 1, 4, 27, 39, 43, 9, 250, 42, 0, 4, 43,
9, 27, 4, 9, 27, 1, 3, 3]
b.y [19, 829, 474, 1855, 2330, 4, 27, 0, 359, 156, 13, 284, 112, 25, 20, 1, 424, 47, 391, 170, 28
, 250, 90, 0, 520, 107, 992, 1550, 1143, 3357, 37, 1, 3, 3]
*****
epoch 4, loss: 5.479074478149414
prediction [10, 34, 31, 5, 145, 0, 52, 10, 4, 74, 21, 1, 6, 119, 50, 44, 53, 0, 6, 53, 77, 59, 59, 1, 72
, 309, 9, 52, 95, 0, 6, 572, 4, 36, 710, 1, 4, 225, 9, 6, 197, 0, 4, 49, 43, 103, 299, 1, 3, 3]
b.y [261, 74, 442, 247, 865, 0, 339, 359, 13, 158, 440, 1, 224, 340, 69, 44, 17, 0, 75, 16, 10, 70
2, 129, 1, 150, 545, 375, 1414, 303, 0, 228, 67, 1463, 104, 2014, 1, 1360, 239, 757, 46, 536, 0, 310, 10
1, 37, 2018, 299, 1, 3, 3]
*****
epoch 4, loss: 5.177460193634033
prediction [10, 43, 9, 51, 4, 66, 20, 0, 4, 18, 9, 125, 4, 39, 47, 1, 10, 552, 4, 343, 9, 122, 85, 0, 10
129, 138, 53, 4, 52, 129, 1, 11, 6, 10, 193, 9, 80, 723, 0, 10, 140, 9, 119, 4, 44, 7, 1, 128, 95, 84,
28, 9, 16, 145, 9, 2, 878, 9, 122, 4, 21, 225, 1, 3, 3]
b.y [1860, 1672, 1930, 204, 475, 1804, 622, 4, 99, 439, 487, 125, 130, 4, 172, 1, 77, 192, 11, 6,
1841, 361, 2338, 0, 264, 5, 7, 17, 704, 384, 279, 1, 66, 87, 2242, 1287, 1140, 2107, 284, 0, 84, 1264,
46, 697, 12, 1853, 1720, 1, 581, 451, 145, 132, 88, 57, 75, 0, 31, 212, 377, 704, 470, 470, 1301, 1,
, 3]
*****
INFO:root:epoch: 4, loss: 5.874064445495055
```

(b)

图 3: 训练过程截图。(a) 训练初期; (b) 训练后期

日，山中天地。
红紫色残光。
不知秋水千万里，不见白云间。

山之，金凤金声出。
一片红云深，一枝斜日起。

夜夜洒寒山绿。
夜来风雨秋风起，夜深春色好谁闻。
不知何处重相见，不得人间有所寻。

湖山中高士，日暮风流入翠流。
莫道无人不可见，一生何处是清明。

海上天地清。
万里不知人在处，一声声尽一千年。

月洒霜下，风光轻露。
玉堂映水，玉节含风起翠微。
莫道无情不可见，一生无事不堪愁。

图 4: 指定开头词诗歌生成结果