

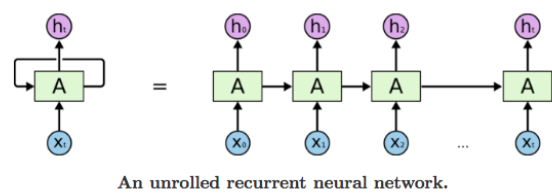
神经网络与深度学习第 3 次作业 实验报告

2251924 晏景豪

1. 模型解释

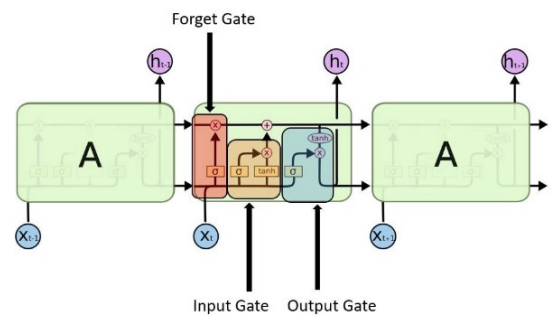
1.1. 循环神经网络（RNN）

循环神经网络（RNN）是一种专门用于处理序列数据的神经网络，通过在网络中引入循环结构，使每个时间步的输出不仅依赖于当前输入，还受先前状态的影响，从而捕捉数据的时间依赖性。其结构使得 RNN 在自然语言处理、语音识别等任务中能够有效建模上下文信息，但同时也面临梯度消失或爆炸的挑战，通常通过引入长短时记忆（LSTM）或门控循环单元（GRU）等变体来缓解这一问题。



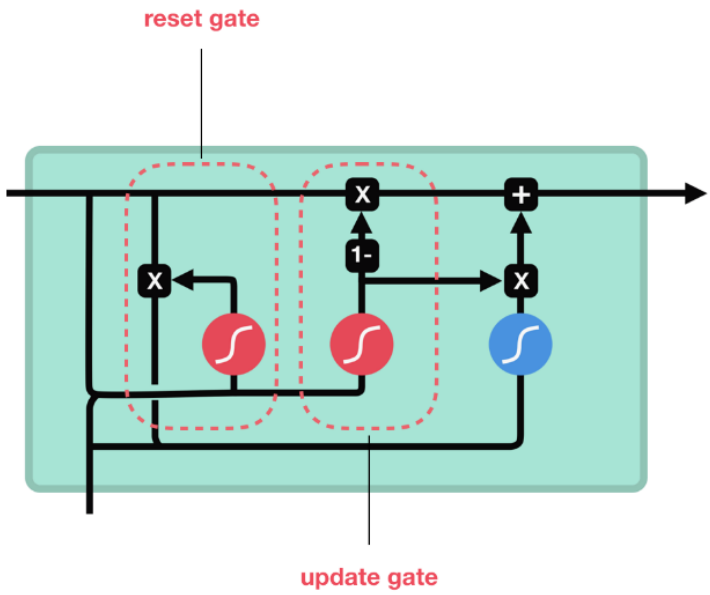
1.2. 长短期记忆网络（LSTM）

长短期记忆网络（Long Short-Term Memory, LSTM）是一种特殊的 RNN，通过引入细胞状态（cell state）和门控机制（输入门、遗忘门、输出门）解决传统 RNN 的梯度消失和长期依赖问题。其核心在于细胞状态沿时间步传递关键信息，门控单元通过 Sigmoid 函数和逐元素相乘操作，动态调控信息的存储、遗忘与输出，从而在时序数据（如文本、语音）中高效捕捉长期依赖关系，广泛应用于机器翻译、时间序列预测等领域。



1.3. 门控循环单元（GRU）

门控循环单元（Gated Recurrent Unit, GRU）是 RNN 的轻量化变体，通过整合长短期记忆网络（LSTM）的门控机制并简化结构，解决梯度消失和长期依赖问题。其核心包含更新门（调控历史状态保留比例）和重置门（控制历史状态信息遗忘程度），利用 Sigmoid 函数与 Hadamard 积实现动态门控，将隐藏状态与细胞状态合并为单一状态传递，在减少参数冗余的同时保持对时序特征的建模能力，广泛用于文本生成、语音识别等序列任务。



2. 生成过程

2.1. Pytorch 版本的生成过程如下：

① 数据处理

数据清洗与预处理：程序从文本文件中读取诗句，通过剔除特殊符号、标点和不符合字数要求的内容，确保数据质量。同时，每首诗前后添加特定的起始（例如“G”）与结束（例如“E”）标记。

构建词典与向量化：对所有诗句中的字进行统计，按照出现频率排序，并构建字到索引的映射表。随后将每首诗转换为由对应字索引组成的向量，便于后续模型输入。

② 模型结构

词嵌入层：采用随机初始化的嵌入层，将离散的字索引映射为连续的向量表示，捕捉字的语义信息。

双层 LSTM：利用两层 LSTM 网络对嵌入后的序列进行处理，捕捉诗歌中长期依赖关系和时序特征。LSTM 的输入为嵌入向量，输出维度对应隐藏层大小。

全连接层与激活函数：LSTM 的输出经过全连接层映射到词汇表维度，随后通过 LogSoftmax 激活函数转换为对数概率分布，为后续生成提供概率基础。

③ 训练过程

批处理与损失计算：采用批次方式处理数据，生成输入与目标序列。模型逐字预测，利用负对数似然损失函数（NLLLoss）衡量预测与实际目标之间的差异。

优化与梯度更新：使用 RMSprop 优化器进行参数更新，并通过梯度裁剪（clip_grad_norm）防止梯度爆炸。训练过程中会定期保存模型参数，便于后续加载与继续训练。

④ 文本生成

种子输入与迭代生成：从指定的起始字开始，模型以当前生成的诗句作为输入，通过预测下一个最可能的字，不断迭代生成直至遇到结束标记或达到预设长度。

后处理与格式化输出：生成的诗句经过格式化处理，按逻辑分段打印，保证最终输出的诗歌具有良好的可读性和艺术表现。

2.1.2 TensorFlow 版本的生成过程如下：

① 数据处理

预处理与构建词典：从文本文件中逐行读取诗歌，按“:”分割后提取内容，添加起始标记 bos 和结束标记 eos，并过滤过长的样本。

统计词频与编码：统计所有词语的出现次数，并构造词汇表（包含特殊标记“PAD”和“UNK”），建立词到索引以及索引到词的映射。

数据集构造：将诗句转换为索引序列，并利用 TensorFlow 的数据管道构造 `tf.data.Dataset`，通过打乱、填充和切分（输入为句子前半部分、目标为后半部分）生成训练样本。

② 模型结构

嵌入层：使用 Keras 的 Embedding 层将词索引映射为固定维度的稠密向量表示。

循环神经网络：基于 SimpleRNNCell 构造 RNN 层（隐藏状态维度为 128），对序列数据进行时序建模。

输出映射：通过全连接层（Dense 层）将 RNN 输出映射到词汇表大小，输出预测每个时间步各词的 logits。

③ 训练过程

损失函数：采用稀疏软 max 交叉熵计算每个时间步的损失，并利用自定义函数对不同长度的序列进行平均。

优化策略：使用 Adam 优化器结合 TensorFlow 的自动微分（GradientTape）进行反向传播更新模型参数，按批次对数据进行训练，并在训练过程中周期性输出损失指标。

④ 文本生成

逐步采样：以指定的起始词（如 bos 或其它自定义字）作为初始输入，通过调用 `get_next_token` 函数，在给定状态下预测下一个词，并迭代更新状态。

生成与输出：循环生成固定长度的词序列后，根据索引映射还原为文字，最终拼接成完整的诗句输出。

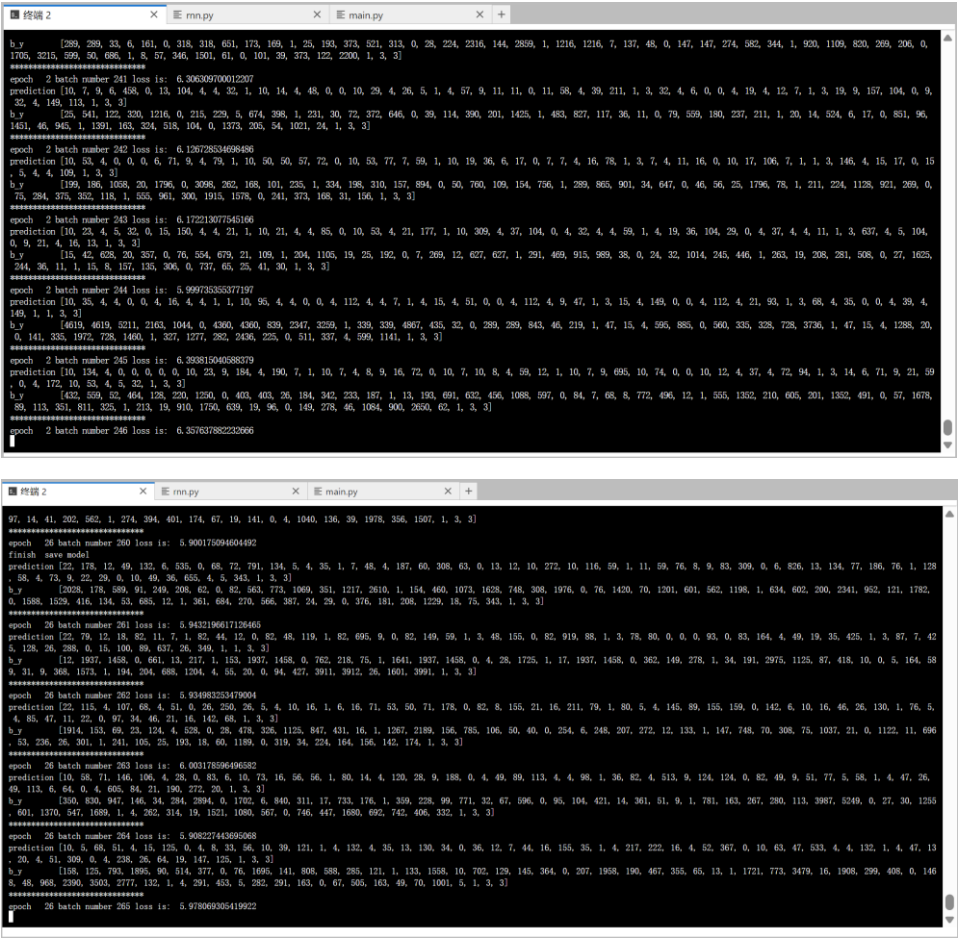
3. 实验结果

TensorFlow 版本的实验结果如下：



```
poem_generation_with_RNN-exercise.ipynb M X
chap6_RNN > poem_generation_with_RNN-exercise.ipynb > 训练优化过程
生成 代码 + Markdown | 全部运行 重启 清除所有输出 | Jupyter 变量 大纲 ...
Python
[7] ✓ 0.4s
...
日风风雨雨，何时不见君。eos生不可见，何事不知君。eos子无人事，何妨不可知。eos心无小事，不是此人心。eos子
日无人间。eos子无人事，何妨不可知。eos心无事去，不是此人心。eos子无人事，何妨不可知。eos心无事去，不是此
孤群闹，湘柳落云，不见此人心。eos子无人事，何妨不可知。eos心无事去，不是此人心。eos子无人事，何妨不可知
边。eos有无人不可知，不知何处是君人。eos来不得无人事，不是春风不见春。eos道不知何事，不知何处是君人。
风吹落月中。eos生无事在，何事不知君。eos子无人事，何妨不可知。eos心无小事，不是此人心。eos子无人事，何妨
eos来不见三千里，不是春风不见春。eos道不知何事，不知何处是君人。eos来不得无人事，不是春风不见春。eos
何处不相思。eos客无人去，何人不可知。eos心无事去，不是此人心。eos子无人事，何妨不可知。eos心无事去，不
eos生不见三年事，不是人间不得归。eos道不知何事，不知何处是君人。eos来不得无人事，不是春风不见春。eos
```

Pytorch 版本训练过程中的截图如下：



```
终端 2 x rnn.py x main.py x +
b.y [289, 289, 33, 6, 161, 0, 318, 318, 601, 173, 169, 1, 25, 193, 373, 521, 313, 0, 23, 224, 2316, 144, 2859, 1, 1216, 1216, 7, 137, 48, 0, 147, 147, 274, 582, 344, 1, 920, 1109, 820, 269, 206, 0,
1705, 3215, 599, 50, 696, 1, 8, 57, 346, 1501, 61, 0, 101, 39, 373, 122, 2200, 1, 3, 3]
epoch 2 batch number 241 loss is: 6.30630700012207
prediction [10, 7, 9, 6, 458, 0, 13, 104, 4, 4, 32, 1, 10, 14, 4, 48, 0, 0, 10, 29, 4, 26, 5, 1, 4, 57, 9, 11, 11, 0, 11, 58, 4, 39, 211, 1, 3, 32, 4, 6, 0, 0, 4, 19, 4, 12, 7, 1, 3, 19, 9, 157, 104, 0, 9,
32, 4, 149, 113, 1, 3, 3]
b.y [25, 541, 128, 330, 1216, 0, 215, 228, 5, 674, 298, 1, 231, 30, 72, 372, 646, 0, 39, 114, 390, 201, 1425, 1, 483, 827, 117, 36, 11, 0, 79, 559, 180, 237, 211, 1, 20, 14, 524, 6, 17, 0, 851, 96,
1451, 46, 945, 1, 1591, 163, 324, 518, 104, 0, 1373, 205, 54, 1021, 24, 1, 3, 3]
epoch 2 batch number 242 loss is: 6.126728334098486
prediction [10, 53, 4, 0, 0, 0, 6, 71, 9, 4, 79, 1, 10, 50, 50, 57, 72, 0, 10, 53, 77, 7, 59, 1, 10, 19, 36, 6, 17, 0, 7, 7, 4, 16, 78, 1, 3, 7, 4, 11, 16, 0, 10, 17, 106, 7, 1, 1, 2, 146, 4, 15, 17, 0, 15,
5, 4, 4, 109, 1, 3, 3]
b.y [199, 186, 1058, 20, 1796, 0, 3098, 262, 168, 101, 233, 1, 334, 198, 310, 157, 894, 0, 50, 760, 109, 154, 756, 1, 289, 865, 901, 34, 647, 0, 46, 56, 25, 1796, 78, 1, 211, 224, 1128, 921, 269, 0,
75, 284, 375, 302, 118, 1, 555, 961, 300, 1915, 1578, 0, 241, 373, 168, 31, 156, 1, 3, 3]
epoch 2 batch number 243 loss is: 6.172213077545166
prediction [10, 23, 4, 5, 32, 0, 15, 150, 4, 4, 21, 1, 10, 21, 4, 4, 85, 0, 10, 53, 4, 21, 177, 1, 10, 309, 4, 37, 104, 0, 4, 32, 4, 5, 59, 1, 4, 19, 36, 104, 29, 0, 4, 37, 4, 4, 11, 1, 3, 637, 4, 5, 104,
0, 9, 21, 4, 16, 13, 1, 3, 3]
b.y [15, 42, 628, 20, 357, 0, 76, 554, 679, 21, 109, 1, 204, 1105, 19, 25, 192, 0, 7, 269, 12, 627, 627, 1, 291, 469, 915, 989, 38, 0, 24, 32, 1014, 245, 446, 1, 263, 19, 208, 281, 508, 0, 27, 1625,
244, 36, 11, 1, 15, 8, 157, 135, 306, 0, 737, 65, 25, 41, 30, 1, 3, 3]
epoch 2 batch number 244 loss is: 5.99973535377197
prediction [10, 35, 4, 4, 0, 0, 4, 16, 4, 4, 1, 1, 10, 95, 4, 4, 0, 0, 4, 112, 4, 4, 7, 1, 4, 15, 4, 51, 0, 0, 4, 112, 4, 9, 47, 1, 3, 15, 4, 149, 0, 0, 4, 112, 4, 21, 93, 1, 3, 68, 4, 35, 0, 0, 4, 39, 4,
149, 1, 1, 3, 3]
b.y [619, 4019, 3311, 2163, 1044, 0, 4360, 4360, 859, 2547, 3259, 1, 339, 339, 4867, 435, 32, 0, 289, 289, 843, 45, 219, 1, 47, 15, 4, 565, 885, 0, 560, 335, 335, 728, 3736, 1, 47, 15, 4, 1285, 20,
0, 141, 335, 1972, 728, 1460, 1, 327, 1277, 262, 2436, 225, 0, 511, 337, 4, 599, 1141, 1, 3, 3]
epoch 2 batch number 245 loss is: 6.393815040588379
prediction [10, 134, 4, 0, 0, 0, 0, 0, 10, 22, 9, 184, 4, 190, 7, 1, 10, 7, 4, 8, 9, 16, 72, 0, 10, 7, 10, 8, 4, 59, 12, 1, 10, 7, 9, 695, 10, 74, 0, 0, 10, 12, 4, 37, 4, 72, 94, 1, 3, 14, 6, 71, 9, 21, 59,
0, 4, 172, 10, 53, 4, 5, 32, 1, 3, 3]
b.y [432, 559, 52, 464, 128, 220, 1250, 0, 403, 403, 26, 184, 342, 233, 187, 1, 13, 193, 691, 632, 456, 1088, 597, 0, 84, 7, 68, 8, 772, 496, 12, 1, 555, 1332, 210, 605, 201, 1332, 491, 0, 57, 1678,
89, 113, 351, 811, 325, 1, 213, 19, 910, 1750, 639, 19, 96, 0, 149, 278, 46, 1084, 900, 2650, 62, 1, 3, 3]
epoch 2 batch number 246 loss is: 6.33763788223666
97, 14, 41, 202, 562, 1, 274, 394, 401, 174, 67, 19, 141, 0, 4, 1040, 136, 39, 1978, 356, 1507, 1, 3, 3]
epoch 20 batch number 260 loss is: 5.900175094604492
finish save model
prediction [22, 178, 12, 49, 132, 6, 535, 0, 68, 72, 791, 134, 5, 4, 35, 1, 7, 48, 4, 187, 60, 308, 63, 0, 13, 12, 10, 272, 10, 116, 59, 1, 11, 59, 76, 8, 9, 83, 309, 0, 6, 826, 13, 134, 77, 180, 76, 1, 128,
58, 4, 73, 9, 22, 29, 0, 10, 49, 35, 655, 4, 5, 343, 1, 2, 2]
b.y [2028, 178, 889, 91, 249, 208, 62, 0, 82, 563, 773, 1069, 351, 1217, 2610, 1, 154, 460, 1073, 1628, 748, 308, 1976, 0, 76, 1420, 70, 1301, 601, 562, 1198, 1, 634, 602, 200, 2341, 952, 121, 1782,
0, 1588, 1529, 416, 134, 53, 685, 12, 1, 361, 694, 270, 566, 387, 24, 29, 0, 376, 181, 208, 1229, 18, 75, 343, 1, 3, 3]
epoch 20 batch number 261 loss is: 5.9432196617126465
prediction [22, 79, 12, 18, 82, 11, 7, 1, 82, 44, 12, 0, 82, 48, 119, 1, 82, 695, 9, 0, 82, 149, 59, 1, 3, 48, 155, 0, 82, 919, 88, 1, 3, 78, 80, 0, 0, 0, 93, 0, 83, 164, 4, 49, 19, 35, 425, 1, 3, 87, 7, 42,
5, 128, 26, 288, 0, 15, 100, 89, 637, 26, 349, 1, 1, 3, 3]
b.y [12, 1537, 1458, 0, 661, 15, 217, 1, 153, 1937, 1458, 0, 762, 218, 75, 1, 1641, 1937, 1458, 0, 4, 23, 1725, 1, 17, 1937, 1458, 0, 362, 149, 278, 1, 34, 191, 2973, 1125, 87, 418, 10, 0, 5, 164, 58,
9, 31, 9, 368, 1573, 1, 194, 294, 688, 1294, 4, 55, 20, 0, 94, 427, 3911, 3912, 26, 1601, 3991, 1, 3, 3]
epoch 20 batch number 262 loss is: 5.934983233479004
prediction [22, 115, 4, 107, 68, 4, 514, 0, 26, 250, 28, 5, 4, 10, 16, 1, 6, 16, 71, 53, 50, 71, 178, 0, 82, 8, 155, 21, 16, 211, 79, 1, 80, 5, 4, 145, 89, 155, 159, 0, 142, 6, 10, 16, 46, 26, 130, 1, 76, 5,
4, 85, 47, 11, 22, 0, 97, 34, 46, 21, 16, 142, 68, 1, 3, 3]
b.y [1914, 153, 69, 23, 124, 4, 528, 0, 28, 478, 326, 1125, 847, 431, 16, 1, 1267, 2189, 156, 785, 106, 50, 40, 0, 254, 6, 248, 207, 272, 12, 133, 1, 147, 748, 70, 308, 75, 1037, 21, 0, 1122, 11, 696,
53, 236, 26, 301, 1, 241, 105, 25, 193, 18, 60, 1189, 0, 319, 34, 224, 164, 156, 142, 174, 1, 3, 3]
epoch 20 batch number 263 loss is: 6.003178596496582
prediction [10, 58, 71, 146, 106, 4, 28, 0, 83, 6, 10, 73, 16, 56, 56, 1, 80, 14, 4, 120, 28, 9, 188, 0, 4, 49, 89, 113, 4, 4, 98, 1, 36, 82, 4, 513, 9, 124, 124, 0, 82, 49, 9, 51, 77, 5, 58, 1, 4, 47, 26,
69, 113, 6, 84, 0, 4, 605, 84, 21, 190, 272, 20, 1, 3, 3]
b.y [150, 830, 947, 146, 34, 284, 2994, 0, 1702, 6, 840, 311, 17, 733, 176, 1, 359, 238, 99, 771, 32, 67, 596, 0, 95, 194, 421, 14, 361, 51, 9, 1, 781, 163, 267, 280, 113, 3987, 5249, 0, 27, 30, 1285,
601, 1370, 547, 1689, 1, 4, 262, 314, 19, 1521, 1080, 567, 0, 746, 447, 1680, 692, 742, 406, 332, 1, 3, 3]
epoch 20 batch number 264 loss is: 5.906227445695668
prediction [10, 5, 68, 81, 4, 15, 125, 0, 4, 8, 33, 56, 10, 39, 121, 1, 4, 132, 4, 35, 13, 130, 34, 0, 36, 12, 7, 44, 16, 155, 35, 1, 4, 217, 222, 16, 4, 52, 367, 0, 16, 63, 47, 533, 4, 4, 132, 1, 4, 47, 13,
20, 4, 51, 309, 0, 4, 238, 28, 64, 19, 147, 125, 1, 3, 3]
b.y [158, 123, 703, 1892, 90, 514, 377, 0, 76, 1695, 141, 808, 588, 285, 121, 1, 132, 1558, 10, 702, 129, 145, 364, 0, 207, 1958, 190, 467, 355, 65, 13, 1, 1721, 773, 3479, 16, 1908, 299, 408, 0, 146,
8, 48, 966, 2390, 3303, 2777, 132, 1, 4, 291, 433, 5, 282, 291, 163, 0, 67, 563, 163, 49, 70, 1001, 5, 1, 3, 3]
epoch 20 batch number 265 loss is: 5.978069305419922
```

训练过程中的 GPU 使用率如下：



生成结果如下：

```
终端 1  x  main.py  x  +
(base) root@autodl-container-3686439328-62bdcf55:~/autodl-tmp/tangshi_for_pytorch# python main.py
error
initial linear weight
日暖莺初啼，红袖添香弄玉笛。
山色遥连秦树直，夜窗松月旧禅机。
error
initial linear weight
红藕映清池，山雨欲来时。
error
initial linear weight
山静谷音幽，湖光万顷秋。
error
initial linear weight
夜阑星未沉，湖波激滟碎冰轮。
海天相接浑无界，月照孤舟认旧痕。
error
initial linear weight
湖阔雁行疏，海日生残夜。
error
initial linear weight
海气幻楼台，月华浸碧苔。
红蓼风前舞，日暮钓船回。
error
initial linear weight
月落乌啼霜，红烛泪千行。
(base) root@autodl-container-3686439328-62bdcf55:~/autodl-tmp/tangshi_for_pytorch#
```

4. 实验总结

本次实验分别采用 TensorFlow 和 PyTorch 框架，实现了双层 LSTM 网络，重点探索其在序列建模中的核心特性。模型通过词嵌入层将字符映射为 100 维语义向量，利用 LSTM 的遗忘门、输入门和输出门机制，在 128 维隐藏状态中动态维护时序记忆。训练采用 RMSprop 优化器配合梯度裁剪策略，有效平衡了参数更新速度与稳定性，经过多轮迭代后损失函数呈现稳定收敛趋势。

实验验证了 LSTM 处理长序列依赖的优越性，其门控结构可自适应选择记忆与遗忘节点。但在生成测试中观察到局部重复现象，反映出传统自回归解码策略的局限性。本次实验深化了我对循环神经网络内部状态传递机制的理解，为后续学习打下了坚实的基础。