**RNN唐诗生成任务报告**

2252334 孙毓涵

1. **RNN,LSTM,GRU模型**
2. **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：

ht=tanh(Wxhxt+Whhht−1+bh)

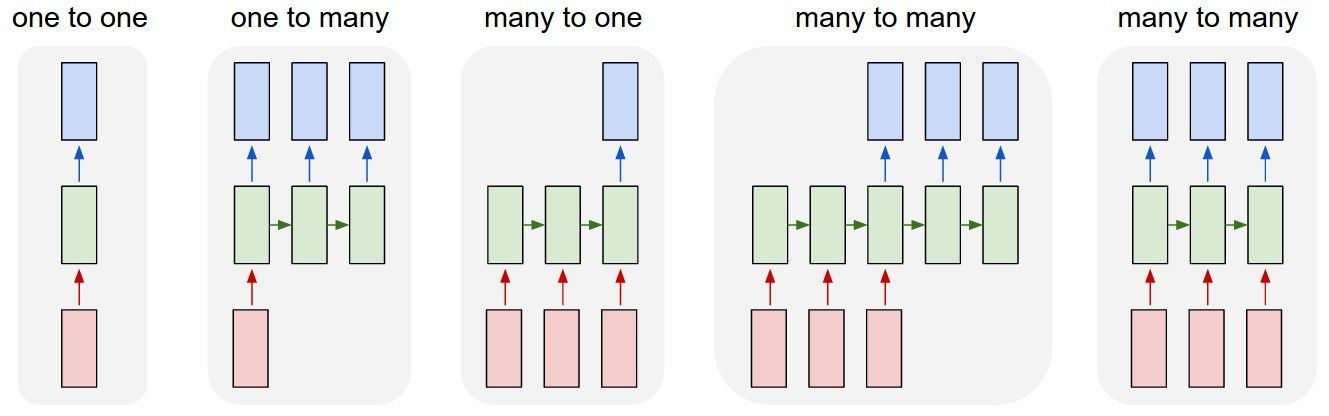
yt=Whyht+by

**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。



1. **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。

ft=σ(Wf⋅[ht−1,xt]+bf)

* **输入门**：决定更新哪些新信息。

it=σ(Wi⋅[ht−1,xt]+bi)

C~t=tanh(WC⋅[ht−1,xt]+bC)

* **细胞状态**：长期记忆的存储与更新。

Ct=ft⊙Ct−1+it⊙C~t

* **输出门**：控制当前隐藏状态的输出。

ot=σ(Wo⋅[ht−1,xt]+bo)

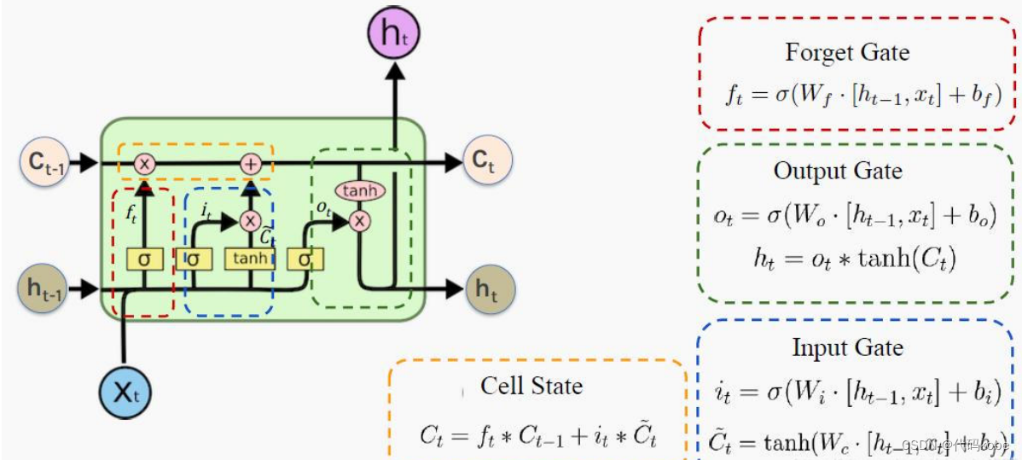
ht=ot⊙tanh(Ct)

**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。



1. **GRU**

**核心思想**：简化LSTM，合并部分门控结构，保持性能。  
**关键结构**：

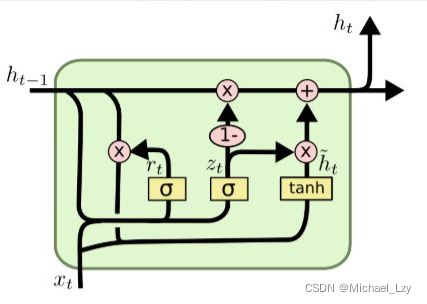
* **重置门**：控制历史信息的忽略程度。  
  rt=σ(Wr⋅[ht−1,xt]+br)
* **更新门**：平衡新旧信息的比例。  
  zt=σ(Wz⋅[ht−1,xt]+bz)
* **候选隐藏状态**：结合当前输入和部分历史信息。  
  h~t=tanh(W⋅[rt⊙ht−1,xt]+b)
* **最终隐藏状态**：  
  ht=(1−zt)⊙ht−1+zt⊙h~t

**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。



1. **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**

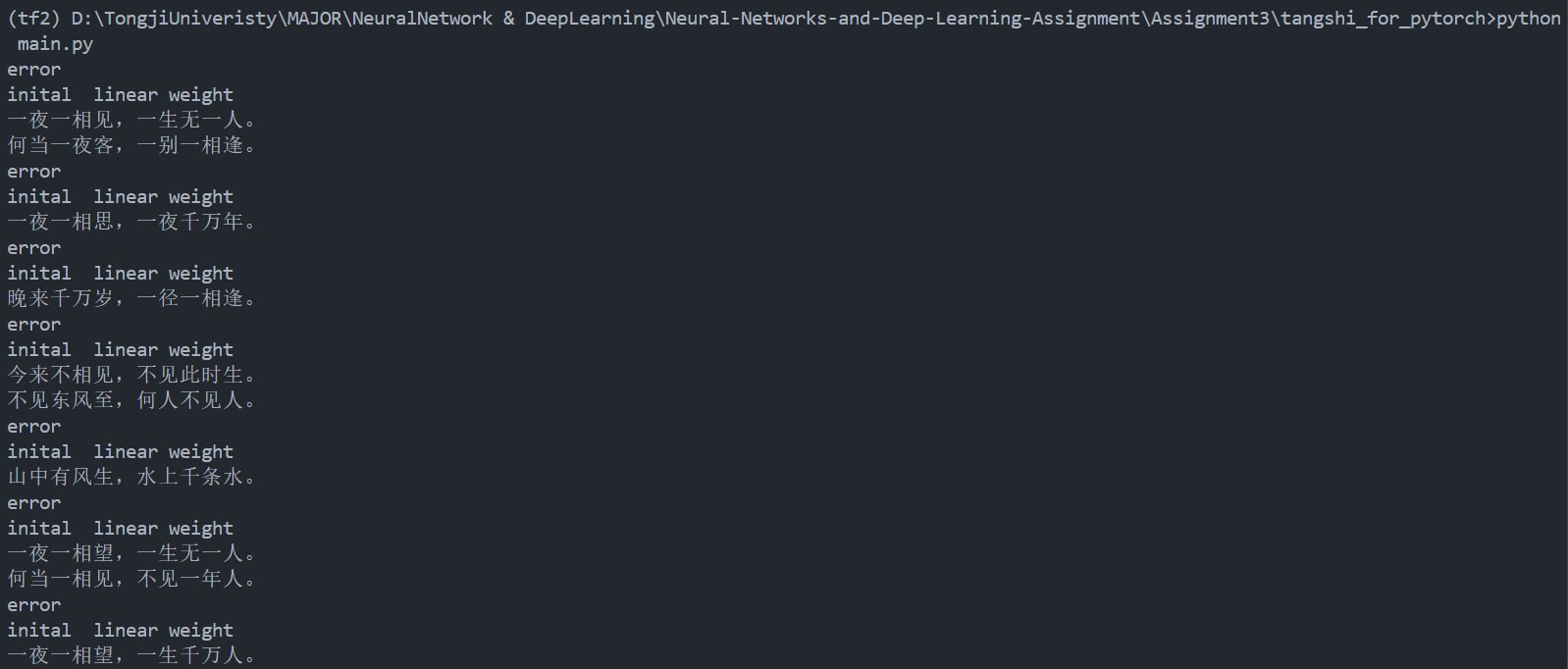
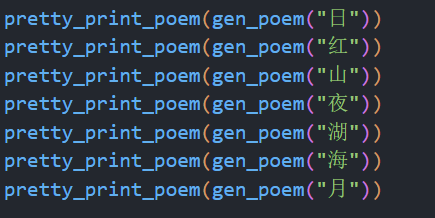
* **数据分批次**：
  + 将 poems\_vector 分成多个批次（batch\_size），每个批次包含输入数据 x\_batches 和目标数据 y\_batches。
* **损失函数**：
  + 使用负对数似然损失（NLLLoss）计算模型输出与目标字符之间的误差。
* **优化器**：
  + 使用 RMSprop 优化器更新模型参数。
* **训练过程**：
  + 遍历所有批次，计算损失并反向传播更新模型参数。
  + 每隔一定批次保存模型参数到文件（如poem\_generator\_rnn）。

**（4）诗歌生成**

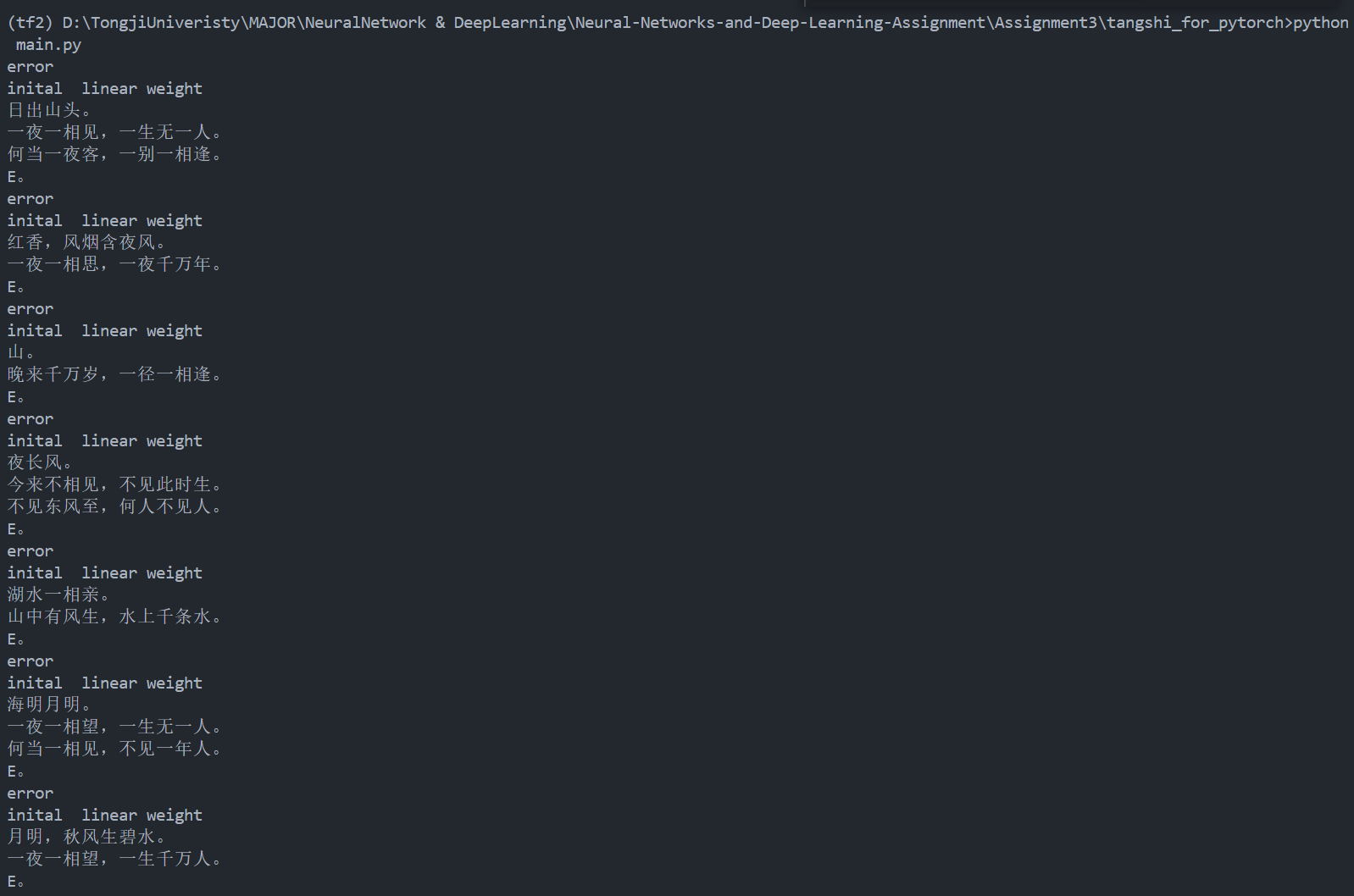
* **加载模型**：
  + 从保存的文件中加载训练好的模型参数。
* **生成过程**：
  + 给定一个起始字符（如 "日"），将其转换为整数索引并输入模型。
  + 模型预测下一个字符的概率分布，选择概率最高的字符作为输出。
  + 将预测的字符追加到当前诗歌中，并作为下一个输入。
  + 重复上述过程，直到生成结束标记E或达到最大长度。
* **结果输出**：
  + 将生成的字符序列转换为诗歌文本，并格式化输出。

1. **诗歌生成**

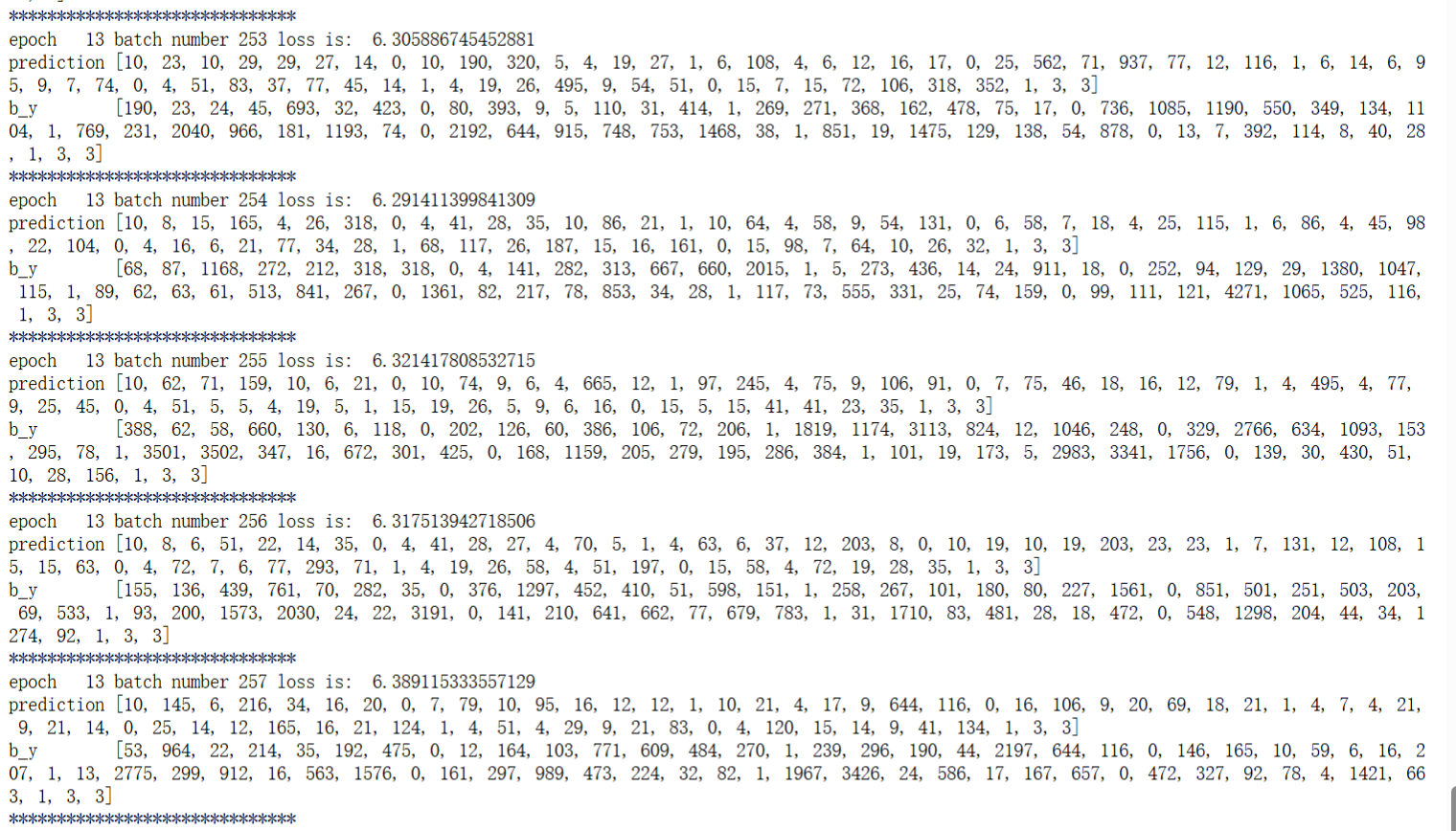
生成诗歌 开头词汇是 “ 日 、 红 、 山 、 夜 、 湖、 海 、 月 ”。



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：



**训练过程截图：**



1. **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。