**RNN实验报告**

tensorflow和pytorch版均实现，诗歌生成过程、截图根据pytorch版描述

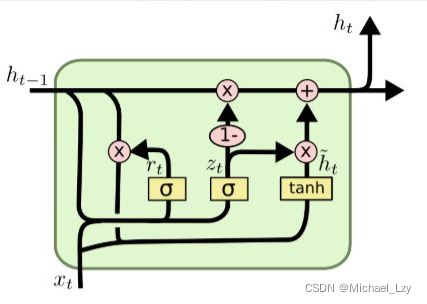
**1 RNN、LSTM、GRU**

1. RNN：RNN是一种处理序列数据的神经网络，每一时刻的输入不仅依赖于当前的输入，还依赖于上一时刻的隐藏状态。RNN的核心公式为：

其中 是当前时刻的隐藏状态， 是当前时刻的输入，是上一时刻的隐藏状态， 是激活函数（如），均为可学习参数。

1. LSTM：RNN的一种改进版本，解决了传统RNN在长序列中容易出现的梯度消失和梯度爆炸问题，以及RNN长期记忆遗忘的问题。它通过引入门控机制（输入门、遗忘门、输出门）以及细胞状态（cell state），能够更好地捕捉长期依赖关系。模型包括：遗忘门，控制丢弃多少旧信息；输入门，控制添加多少新信息；输出门，控制输出的隐藏状态。
2. GRU：LSTM的简化版本，同样用于解决长序列中的梯度消失、长期遗忘的问题。它将LSTM的三个门（输入门、遗忘门、输出门）合并为两个门（更新门和重置门），从而减少了参数数量。更新门：决定保留多少旧状态。重置门：决定忽略多少过去的信息。核心公式为：

其中 是更新门， 是重置门，为激活函数，为当前时刻的候选隐藏状态。

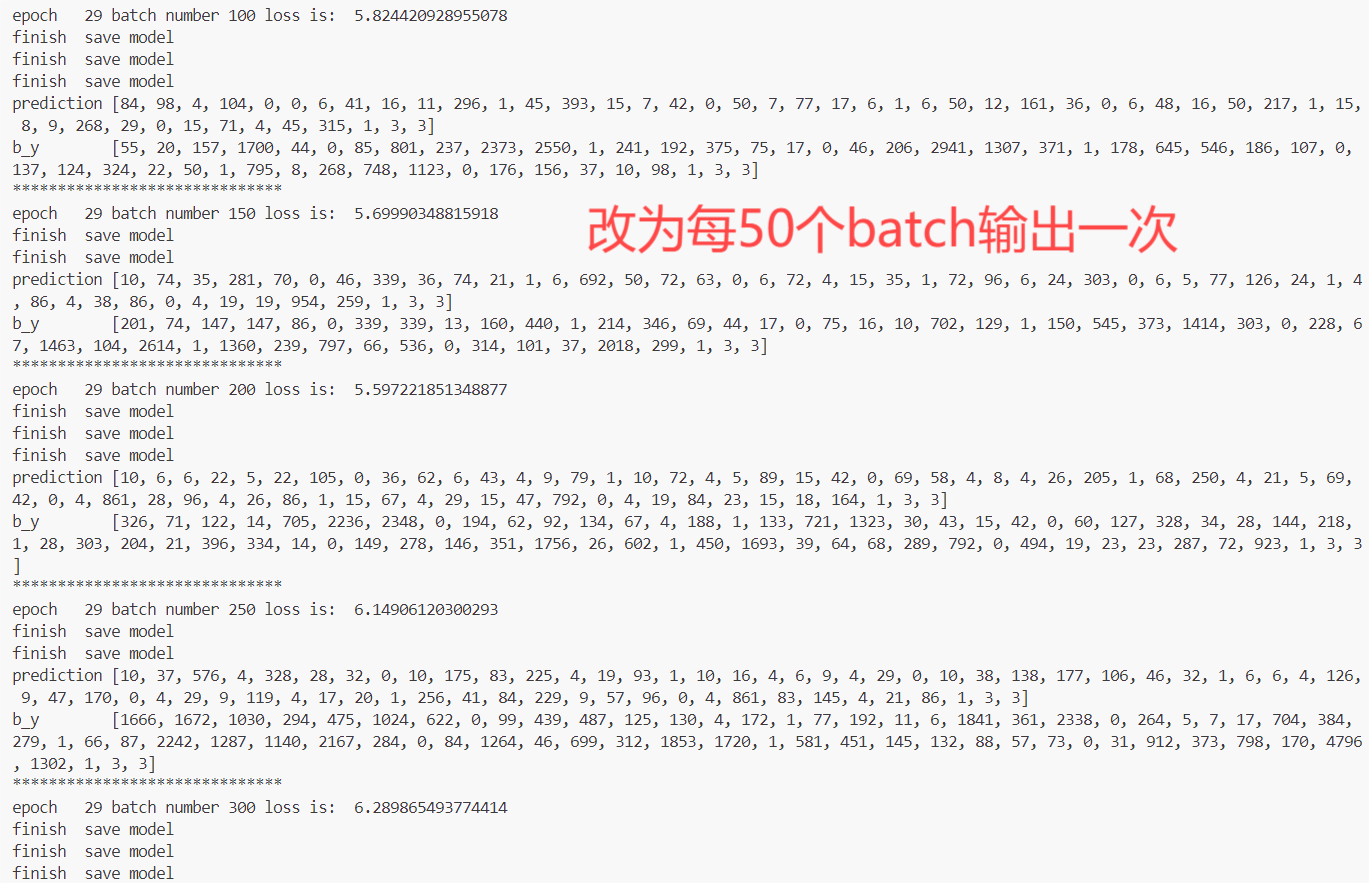


**2 诗歌生成过程解释**

1. 数据预处理：采用process\_poems1或process\_poems2函数完成，读取指定fille\_name文件，去除特殊字符如“\_”，确保诗歌内容在合理长度（5-80），诗歌开头加上‘G’，结尾加上‘E’，按照频率排序，构建字到索引的映射（word\_int\_map），将每首诗改为字索引序列（poems\_vector）。
2. 生成数据：使用generate\_batch将诗歌数据划分为批量，包括输入序列和目标序列，其中目标序列是输入序列向后平移一个位置的结果。例如：[6,2,7,8,4]->[2,7,8,4,4]
3. 模型定义：词嵌入（word\_embedding）：将字索引转换为特定维度的向量；LSTM层（self.rnn\_lstm）；全连接层（self.fc），将LSTM的输出映射回词汇表大小；softmax层，生成每个字的概率分布。
4. 模型训练：使用RMSProp优化器，负对数似然损失（NLLLoss），在30个epoch中，每进入一次epoch，生成一次batchs数据，对于每一个batch，计算损失并反向传播更新权重，每隔一定batch步数打印结果、保存模型状态。
5. 生成诗歌：加载训练好的模型权重，指定初始字begin\_word，将输入转化为torch，逐字生成word并加入到poem后面，如果生成终止字符‘E’或长度大于30，则退出并返回。

**3 截图**

训练：因输出次数过多，改为每50个batch输出一次



生成诗歌：

在源代码中增加了“海”、“月”为开头的输出。注意到要求以“山”、“海”为开头的诗歌不包含开头字，并且生成的诗歌缺少第一句，目前不清楚原因。



**4 总结**

了解了对文本数据的基本处理过程：将字转化为索引，标签为每个字的下一个字；熟悉了word2vec的词嵌入方法，以及LSTM、GRU的基本架构和模型调用方法；熟悉了先训练，保存模型，再加载模型进行推理的训练-测试流程。