作业 4 报告

一、RNN 、LSTM、GRU 模型

RNN（循环神经网络）、LSTM（长短期记忆网络）和 GRU（门控循环单元）都是用于处理序列数据的神经网络结构。序列数据是指那些在不同时间步长上连续产生的数据，如文本、语音、时间序列等。这些模型通过捕捉序列中的时间依赖关系来解决一些复杂的问题。下面我会简单解释这些模型的核心思想。

（一）RNN（循环神经网络）

RNN 的基本思想是使用带有循环的神经网络结构来处理序列数据。在 RNN 中，每个时间步长上的输出不仅取决于当前输入，还取决于上一个时间步长的隐藏状态。这使得 RNN 能够捕捉序列中的时间依赖关系。然而，RNN 在处理长序列时可能会遇到梯度消失或梯度爆炸的问题，导致无法有效捕获长期依赖。

（二）LSTM（长短期记忆网络）

LSTM 是 RNN 的一种变体，它设计了一个更加复杂的内部结构，旨在解决 RNN 中的梯度消失和长期依赖问题。LSTM 通过引入门控机制（输入门、遗忘门和输出门）以及细胞状态来控制信息的流动。遗忘门决定了哪些信息应该被保留，输入门决定了哪些新信息应该被添加到细胞状态中，而输出门则决定了哪些信息应该被输出到下一个时间步长。这种设计使得 LSTM 能够更好地捕获长期依赖关系。

（三）GRU（门控循环单元）

GRU 是另一种 RNN 的变体，它与 LSTM 在思路上有些相似，但结构上更加简化。GRU 同样引入了门控机制，但只有两个门：重置门和更新门。重置门决定了如何将新的输入信息与先前的隐藏状态相结合，而更新门则决定了如何更新隐藏状态。与 LSTM 相比，GRU 的参数更少，计算效率更高，但在某些任务上可能稍逊于 LSTM。

RNN、LSTM 和 GRU 都是用于处理序列数据的神经网络模型，它们通过捕捉序列中的时间依赖关系来解决复杂问题。LSTM 和 GRU 是 RNN 的改进版本，能够更好地处理长期依赖关系。在实际应用中，选择哪种模型取决于具体任务和数据特点。

二、诗歌生成过程

1、数据准备和预处理：在代码中构建了词汇表、创建词嵌入矩阵（word_embedding）以及将诗歌文本转换为数值索引序列。这些步骤是将原始文本数据转换为模型可以处理的格

式。

2、模型定义：代码 rnn.py 定义了 RNN_model 类，它继承自 nn.Module，是 PyTorch 中定义神经网络的基类。

在__init__方法中，模型设置了必要的参数，包括批量大小、词汇表大小、词嵌入维度和 LSTM 的隐藏层维度。

在 RNN_model 类定义了 LSTM 层（self.rnn_lstm），它有两层，并指定了输入和输出的大小；定义了一个全连接层（self.fc），用于将 LSTM 的输出转换为词汇表大小的向量；应用了权重初始化函数 weights_init 来初始化模型的权重；并且定义了 forward 方法，该方法描述了模型的前向传播过程。

3、前向传播过程：在 forward 方法中，首先将输入的句子（数值索引序列）通过词嵌入层转换为词嵌入向量。

将词嵌入向量 reshape 为适合 LSTM 输入的形状（批量大小、序列长度、特征维度），然后初始化 LSTM 的隐藏状态和细胞状态为零，将词嵌入向量输入到 LSTM 中，并得到输出和更新后的隐藏状态，从 LSTM 的输出中取出最后一个时间步的隐藏状态（对于序列预测任务，这通常包含了整个序列的上下文信息），将最后一个隐藏状态通过全连接层，并应用对数 softmax 激活函数，得到每个词汇表中单词的预测概率。

4、训练和生成：代码中还包括了模型的训练过程，包括定义损失函数、优化器，以及通过反向传播更新模型的权重。训练完成后，模型将学习到从输入序列到输出序列的映射关系。

5、诗歌生成：在测试或生成阶段，使用训练好的模型来生成诗歌。生成过程从一个初始的单词或句子开始，然后模型预测下一个最可能的单词。这个预测的单词被添加到序列中，并作为下一个输入提供给模型，如此循环，直到达到预设的诗歌长度或满足其他停止条件。在每个时间步，模型使用 forward 方法预测下一个单词，并可能在 is_test 模式下仅返回最后一个单词的预测概率。

所以，诗歌生成过程涉及数据预处理、模型定义与训练、以及使用训练好的模型进行序列生成和后处理。定义的 RNN_model 类是这一过程的核心部分，它负责学习从输入序列到输出序列的映射，并在生成阶段预测下一个单词。