1. RNN ，LSTM，GRU模型介绍

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模型。它们在自然语言处理（NLP）、时间序列分析、语音识别等任务中得到了广泛的应用。

**RNN（循环神经网络）：** RNN 是一种基本的序列模型，它具有反馈连接，允许信息在网络内部传递。在 RNN 中，每个时间步的输入不仅取决于当前的输入，还取决于上一个时间步的输出，形成了一种循环的结构。然而，标准的 RNN 存在梯度消失或梯度爆炸的问题，这使得它在处理长序列时表现不佳。

**LSTM（长短期记忆网络）：** LSTM 是为了解决 RNN 中长期依赖问题而提出的一种特殊结构。它通过增加门控单元来控制信息的流动，包括遗忘门、输入门和输出门。这些门控制着信息流中的信息是否被遗忘、输入以及输出，从而在长序列中更好地捕捉到长期依赖关系。LSTM 具有记忆单元，可以选择性地记忆或遗忘信息，因此能够更好地处理长期的时间依赖关系。

**GRU（门控循环单元）：** GRU 是另一种解决 RNN 中长期依赖问题的结构，与 LSTM 类似，但更加简化。GRU 合并了遗忘门和输入门，以及更新门和重置门，减少了参数数量，降低了计算成本。虽然相对于 LSTM 来说，GRU 在一些任务上的表现略逊一筹，但它在实践中常常被用作一种更轻量级的替代方案。

总的来说，LSTM 和 GRU 是 RNN 的改进版本，旨在克服 RNN 在处理长序列时的限制。它们通过引入门控机制，使得模型可以更好地捕捉到长期依赖关系，从而在各种序列数据的任务中取得更好的性能表现。

1. 诗歌生成过程

该诗歌生成是基于一个循环神经网络（RNN）的模型，通过大量的诗歌数据集训练该模型，从首个词语开始，不断循环调用该模型生成下一个词语直到遇到终止条件，以此来达到生成诗歌的效果。以下是诗歌生成的大概过程：

**1.数据预处理：**

提供了两个数据预处理函数 process\_poems1 和 process\_poems2，它们从给定的文件中读取诗歌数据，并将其转换成数字化的形式，即将每个字转换为对应的索引。同时构建了词汇表和词到索引的映射关系。

**2.生成批次数据：**

generate\_batch 函数用于生成训练时所需的批次数据，将诗歌数据划分成多个批次，每个批次包含多个样本。同时生成对应的输入和输出数据，其中输入数据是诗歌的前一部分，输出数据是对应输入的后续部分。

**3.模型定义与训练：**

定义了 RNN 模型 RNN\_model，其中包含了词嵌入层和 LSTM 层，用于学习诗歌的语义特征和上下文信息。

使用负对数似然损失函数作为损失函数，优化器采用 RMSprop。

使用循环迭代对数据进行训练，每个 epoch 中遍历所有批次数据，计算损失并进行梯度更新。

**4.诗歌生成：**

gen\_poem 函数用于生成诗歌，接受一个起始字作为输入，然后利用训练好的 RNN 模型不断预测下一个字，直到生成结束标志。

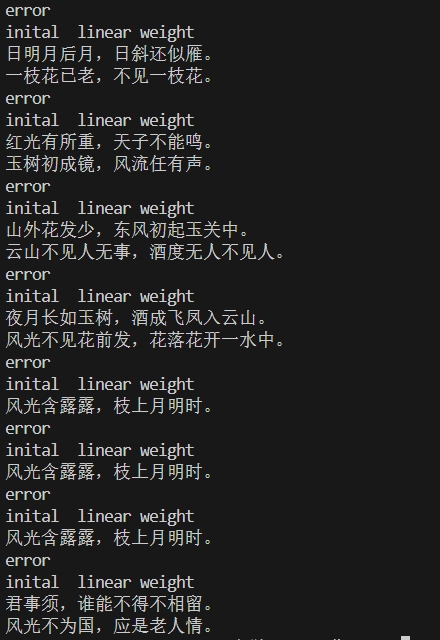
在生成过程中，每次预测都将当前已生成的诗句作为输入，得到下一个字的预测结果，并将其添加到诗句中，直到达到一定长度或者生成结束标志。

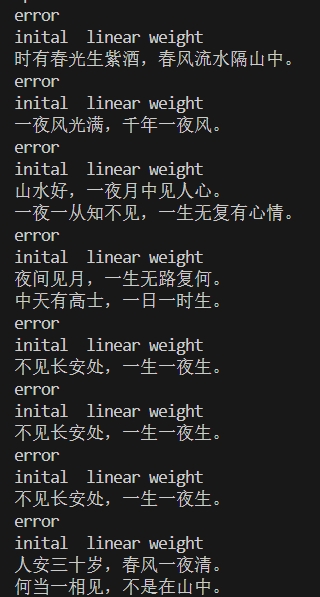
**5.结果展示：**

最后通过 pretty\_print\_poem 函数对生成的诗歌进行格式化打印，使得输出更具可读性。

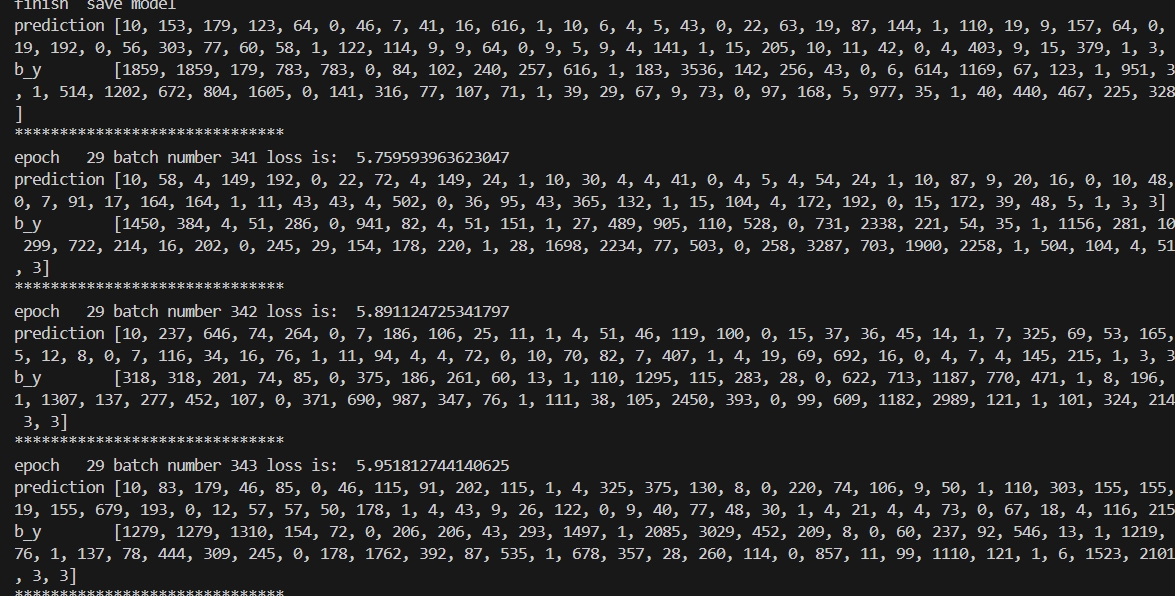
三、生成诗歌

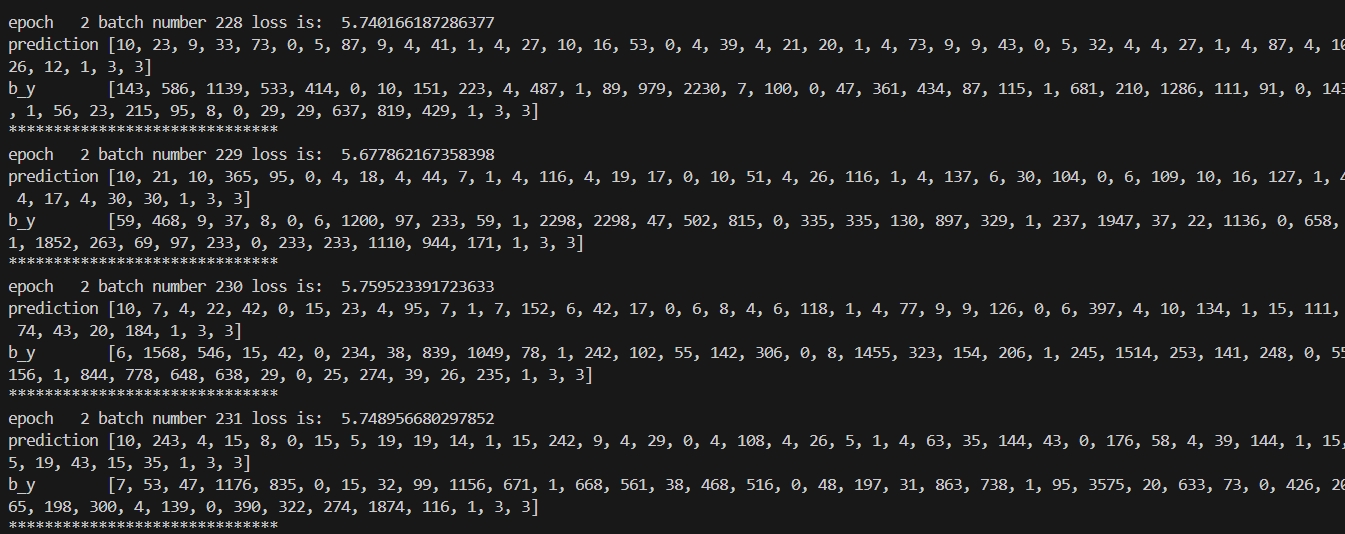
最后，通过调用pretty\_print\_poem()以及gen\_poem()方法生成诗歌。





以下是使用pytorch训练模型时的截图：





1. 实验总结

通过完成诗歌生成的实验，我对深度学习模型在自然语言处理任务中的应用有了更深入的理解。以下是我对这次实验的总结：

**1.理论知识应用：**

在这次实验中，我将课堂学习到的循环神经网络（RNN）模型应用到了诗歌生成任务中。通过理解RNN模型的原理和工作机制，我能够更好地设计和构建用于生成诗歌的模型。

**2.数据预处理：**

在实验中，我学会了如何对诗歌数据进行预处理，包括读取数据、清洗数据、将诗歌转换为模型可接受的数字表示等步骤。这些预处理步骤对于模型的训练和生成过程至关重要。

**3.进一步改进：**

尽管实验取得了一定的成果，但我也意识到诗歌生成模型仍然存在改进的空间。未来，我可以尝试引入注意力机制、增加训练数据的多样性、调整模型结构等方法，以进一步提高生成诗歌的质量和多样性。

通过这次实验，我不仅加深了对深度学习模型的理解，还提升了在自然语言处理领域应用模型的能力。我相信这些经验对我的学习和研究都将有所帮助，也为我未来在人工智能领域的探索打下了坚实的基础。