同济大学

《神经网络与深度学习》实验报告

2253099 生芮瑞

一、实验内容

利用循环神经网络（RNN），实现唐诗生成任务。

二、实验要求

- 补全程序，主要是前面的 3 个空和生成诗歌的一段代码。
- 解释一下 RNN，LSTM，GRU 模型
- 叙述诗歌生成的过程。
- 生成诗歌。开头词汇是“ 日、红、山、夜、湖、海、月”。

三、实验过程

1、补全程序

2、解释模型

RNN（循环神经网络）是一种用于处理序列数据的神经网络模型，它具有记忆先前输入信息的能力。它可以记住每一时刻的信息，每一时刻的隐藏层不仅由该时刻的输入层决定，还由上一时刻的隐藏层决定，从而可以捕捉到时间序列中的依赖关系。利用了 RNN 的这种能力，深度学习模型在解决语音识别、语言模型、机器翻译以及时序分析等 NLP 领域的问题时有所突破。然而，传统的 RNN 存在梯度消失或梯度爆炸的问题，限制了其对长序列的建模能力。如果一个句子很长，到句子末尾时，它将记不住这个句子的开头的详细内容。

为了解决 RNN 的长期依赖问题，LSTM（长短期记忆网络）和 GRU（门控循环单元）被提出。它们都是 RNN 的变体，在 RNN 的 Memory Cell 基础上引入门控机制来控制信息的流动，即对信息进行挑选，从而更好地捕捉长期依赖关系。

LSTM 包含三个门：Input Gate（输入门）、Forget Gate（遗忘门）和 Output Gate（输出门），以及一个 Memory Cell 来存储信息。信息先经过输入门，看是否有信息输入，再判断遗忘门是否选择遗忘 Memory Cell 里的信息，最后再经过输出门，判断是否将这一时刻的信息进行输出。这些门控制信息的流动，使得 LSTM 可以选择性地记住或遗忘信息，从而更好地处理长序列数据。

GRU 与 LSTM 类似，但简化了架构，只包含 Update Gate（更新门）和 Reset Gate（重置门）。重置门决定了如何将新的输入信息与前面的记忆相结合，更新门定义了前面记忆保存到当前时间步的量。GRU 通过这两个门控制信息的更新和重置，减少了参数数量，同时在一定程度上保持了模型的表现。

总的来说，LSTM 和 GRU 是相对于传统 RNN 更为先进的模型，能够更好地处理长序列数据并捕捉长期依赖关系。

3、生成过程

- **数据预处理**

从诗歌数据集 poems.txt 中以“：”为分隔读取数据，同时为数据加入特殊标记 bos（开始）和 eos（结束），帮助模型识别。然后

统计每个字符出现的次数，并对计数结果进行排序得到一个有序元组。然后对元组进行解压，得到 words 列表代表所有字符。该字符的行号即为该字符索引。再通过函数 dict()得到字符与行号对应的索引字典 word2id，以及 id2word。最后根据索引字典将字符数据集映射成索引数据集。

- **建立 RNN 模型**

定义一个继承 keras.Model 的 RNN 模型，包括一个 Embedding 层、一个 SimpleRNNCell 层、一个 RNN 层和一个全连接 Dense 层。通过各层的转换后由输入和当前状态得到输出。

- **训练模型**

从预处理后的数据中得到训练集，然后选择合适的损失函数和优化器，通过参数的不断迭代更新降低损失函数值，提高模型预测的准确率。

- **文本生成**

定义函数，根据指定的句首字让模型预测后一个字，逐字预测，以 eos 为结束标志，生成诗句。

4、生成结果

模型根据指定的句首字生成的诗句如下：

```
generated_poem = gen_sentence(start_words=['日'])
print(generated_poem)
generated_poem = gen_sentence(start_words=['红'])
print(generated_poem)
generated_poem = gen_sentence(start_words=['山'])
print(generated_poem)
generated_poem = gen_sentence(start_words=['夜'])
print(generated_poem)
generated_poem = gen_sentence(start_words=['湖'])
print(generated_poem)
generated_poem = gen_sentence(start_words=['海'])
print(generated_poem)
generated_poem = gen_sentence(start_words=['月'])
print(generated_poem)
```

```
日暮阳山日不见人间，不得无人不可知。
红窦日逢一里来，不得无人不可知。
山上不知无处处，不知何处不相思。
夜落月，山水上云中。
湖上山山不可知。
海阳人不知何处，一年何处不相思。
月，一枝风落花中。
```

由于一次性生成容易导致诗句重复度较高，且会出现单字成句的情况，因此尝试逐句生成，得到如下结果：

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['日'])
print(generated_poem)
```

```
日日，山风不可知。
```

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['日'])
print(generated_poem)
```

```
日暮秋风，一片花声满水中。
```

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['红'])
print(generated_poem)
```

```
红粉叶斜，月落花，一月无人。
```

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['山'])
print(generated_poem)
```

山畔月来无处处，一人不见一人间。

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['夜'])
print(generated_poem)
```

夜暮，一枝花落月中风。

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['湖'])
print(generated_poem)
```

湖上月无人，一年无事在，何处不相思。

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['海'])
print(generated_poem)
```

海畔千里月。风风吹雨落，一月无人不可知。

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['月'])
print(generated_poem)
```

月暮花花不得，何人不得无人事，不得无人不得人。

```
# 打印生成的诗歌
generated_poem = gen_sentence(start_words=['日'])
print(generated_poem)
```

日暮秋风，一片花声满水中。

四、实验总结

通过本次的实验，我更加深刻地体会到了循环神经网络 RNN 在语言处理等需要结合语境（结合时序关系）的问题中具有很大的优势，能够得到更加理想准确的结果，也在调整训练次数的过程中感受到了不同训练次数下模型最终预测效果的巨大差异。同时我还学到了一些机器学习中对自然语言数据预处理的方法，如统计数量、建立索引、添加特殊标记等。除此之外，之前几次作业中学习的内容都相对比较简单，机器能够较快得到理想结果，但这次是对于诗句的处理，难度大大提升，需要的训练量也大大提升（每次都要跑一个多小时）。