循环神经网络(RNN)

RNN、LSTM、GRU 模型解释

1. **RNN（循环神经网络）**
   - **原理**：通过隐藏状态传递序列信息，每个时间步接收当前输入和前一隐藏状态，输出当前隐藏状态。
   - **缺点**：梯度消失/爆炸问题严重，难以捕捉长期依赖。
2. **LSTM（长短期记忆网络）**
   - **改进**：引入输入门、遗忘门、输出门，控制信息流动。
   - **优点**：通过门控机制缓解梯度消失，能有效学习长期依赖。
3. **GRU（门控循环单元）**
   - **改进**：合并 LSTM 的输入门和遗忘门为“更新门”，简化结构。
   - **优点**：参数更少，训练更快，性能接近 LSTM。

诗歌生成过程

1. **数据处理**
   - 从文本文件加载诗歌，添加 `bos`（开始）和 `eos`（结束）标记。
   - 构建词汇表，将字符映射为索引（ID）。
   - 将诗歌转换为索引序列，并进行填充（Padding）和分批（Batching）。
2. **模型构建**
   - **嵌入层**：将字符索引转换为稠密向量。
   - **RNN层**（如 `SimpleRNN/LSTM/GRU`）：处理序列数据，捕捉上下文信息。
   - **全连接层**：输出每个时间步的字符概率分布。
3. **训练**
   - 使用交叉熵损失函数，通过反向传播优化模型参数。
   - 每个批次输入为 `x[:, :-1]`，标签为 `x[:, 1:]`，实现自回归训练。
4. **生成**
   - 从起始词（如“日”）开始，逐步预测下一个字符。
   - 使用贪心搜索（取概率最高的字符）或采样策略生成文本，直到遇到 `eos` 或达到最大长度。

生成诗歌截图

- **一**：

```python
def gen_sentence():
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]
    cur_token = tf.constant([word2id['bos']], dtype=tf.int32)
    collect = []
    for _ in range(50):
        cur_token, state = model.get_next_token(cur_token, state)
        collect.append(cur_token.numpy()[0])
    return [id2word[t] for t in collect]
print(''.join(gen_sentence()))
```

✓ 0.0s Python

```
一片花声满，风吹落日深。eos生无限路，不得是君心。eos有无人事，何人不可怜。eos生无限处，不得是君心。eos有
```

- **日**:

```python
def gen_sentence(start_words="一"):
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]

    start_tokens = [word2id['bos']]
    for char in start_words:
        if char in word2id:
            start_tokens.append(word2id[char])
        else:
            start_tokens.append(word2id['UNK'])

    for token in start_tokens[1:]:
        cur_token = tf.constant([token], dtype=tf.int32)
        _, state = model.get_next_token(cur_token, state)

    collect = start_tokens.copy()
    for _ in range(50 - len(start_words)):
        cur_token = tf.constant([collect[-1]], dtype=tf.int32)
        next_token, state = model.get_next_token(cur_token, state)
        collect.append(next_token.numpy()[0])

    poem = [id2word[t] for t in collect if id2word[t] not in ['bos', 'eos']]
    return ''.join(poem)
print(gen_sentence(start_words="日"))
```

✓ 0.0s Python

日来风吹落，春风不可怜。人无限路，不得是君心。有无人事，何人不可怜。生无限处，不得是君心。有

- **山**:

```python
def gen_sentence(start_words="一"):
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]

    start_tokens = [word2id['bos']]
    for char in start_words:
        if char in word2id:
            start_tokens.append(word2id[char])
        else:
            start_tokens.append(word2id['UNK'])

    for token in start_tokens[1:]:
        cur_token = tf.constant([token], dtype=tf.int32)
        _, state = model.get_next_token(cur_token, state)

    collect = start_tokens.copy()
    for _ in range(50 - len(start_words)):
        cur_token = tf.constant([collect[-1]], dtype=tf.int32)
        next_token, state = model.get_next_token(cur_token, state)
        collect.append(next_token.numpy()[0])

    poem = [id2word[t] for t in collect if id2word[t] not in ['bos', 'eos']]
    return ''.join(poem)
print(gen_sentence(start_words="山"))
```

执行上方所有的单元格

✓ 0.0s Python

山上，一片花声不可知。有不知何处处，不知何处到人间。来不得无人事，不得无人不得归。道不知何处处

- **红**:

```python
def gen_sentence(start_words="一"):
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]

    start_tokens = [word2id['bos']]
    for char in start_words:
        if char in word2id:
            start_tokens.append(word2id[char])
        else:
            start_tokens.append(word2id['UNK'])

    for token in start_tokens[1:]:
        cur_token = tf.constant([token], dtype=tf.int32)
        _, state = model.get_next_token(cur_token, state)

    collect = start_tokens.copy()
    for _ in range(50 - len(start_words)):
        cur_token = tf.constant([collect[-1]], dtype=tf.int32)
        next_token, state = model.get_next_token(cur_token, state)
        collect.append(next_token.numpy()[0])

    poem = [id2word[t] for t in collect if id2word[t] not in ['bos', 'eos']]
    return ''.join(poem)
print(gen_sentence(start_words="红"))
```

✓ 0.0s Python

红叶满枝枝。声不得无人事，不得无人不得归。道不知何处处，不知何处到人间。来不得无人事，不得无人

- **夜**:

```
def gen_sentence(start_words="一"):
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]

    start_tokens = [word2id['bos']]
    for char in start_words:
        if char in word2id:
            start_tokens.append(word2id[char])
        else:
            start_tokens.append(word2id['UNK'])

    for token in start_tokens[1:]:
        cur_token = tf.constant([token], dtype=tf.int32)
        _, state = model.get_next_token(cur_token, state)

    collect = start_tokens.copy()
    for _ in range(50 - len(start_words)):
        cur_token = tf.constant([collect[-1]], dtype=tf.int32)
        next_token, state = model.get_next_token(cur_token, state)
        collect.append(next_token.numpy()[0])

    poem = [id2word[t] for t in collect if id2word[t] not in ['bos', 'eos']]
    return ''.join(poem)
print(gen_sentence(start_words="夜"))
```

✓ 0.0s Python

夜，月上春。生不可见，不知此中心。生不可见，不得一年年。有无人事，何人不可怜。生无限处，不得

- **湖**:

```
def gen_sentence(start_words="一"):
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]

    start_tokens = [word2id['bos']]
    for char in start_words:
        if char in word2id:
            start_tokens.append(word2id[char])
        else:
            start_tokens.append(word2id['UNK'])

    for token in start_tokens[1:]:
        cur_token = tf.constant([token], dtype=tf.int32)
        _, state = model.get_next_token(cur_token, state)

    collect = start_tokens.copy()
    for _ in range(50 - len(start_words)):
        cur_token = tf.constant([collect[-1]], dtype=tf.int32)
        next_token, state = model.get_next_token(cur_token, state)
        collect.append(next_token.numpy()[0])

    poem = [id2word[t] for t in collect if id2word[t] not in ['bos', 'eos']]
    return ''.join(poem)
print(gen_sentence(start_words="湖"))
```

✓ 0.0s Python

湖，一片花声不可知。道不知何处处，不知何处到人间。来不得无人事，不得无人不得归。道不知何处处，

- **海**:

```
def gen_sentence(start_words="一"):
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]

    start_tokens = [word2id['bos']]
    for char in start_words:
        if char in word2id:
            start_tokens.append(word2id[char])
        else:
            start_tokens.append(word2id['UNK'])

    for token in start_tokens[1:]:
        cur_token = tf.constant([token], dtype=tf.int32)
        _, state = model.get_next_token(cur_token, state)

    collect = start_tokens.copy()
    for _ in range(50 - len(start_words)):
        cur_token = tf.constant([collect[-1]], dtype=tf.int32)
        next_token, state = model.get_next_token(cur_token, state)
        collect.append(next_token.numpy()[0])

    poem = [id2word[t] for t in collect if id2word[t] not in ['bos', 'eos']]
    return ''.join(poem)
print(gen_sentence(start_words="海"))
```

✓ 0.0s Python

海，风风吹落月。然不可见，不得无人间。有不可见，不知何处归。然不可见，不得一年年。有无人事，

- **月**：

```python
def gen_sentence(start_words="一"):
    state = [tf.random.normal(shape=(1, 128), stddev=0.5), tf.random.normal(shape=(1, 128), stddev=0.5)]

    start_tokens = [word2id['bos']]
    for char in start_words:
        if char in word2id:
            start_tokens.append(word2id[char])
        else:
            start_tokens.append(word2id['UNK'])

    for token in start_tokens[1:]:
        cur_token = tf.constant([token], dtype=tf.int32)
        _, state = model.get_next_token(cur_token, state)

    collect = start_tokens.copy()
    for _ in range(50 - len(start_words)):
        cur_token = tf.constant([collect[-1]], dtype=tf.int32)
        next_token, state = model.get_next_token(cur_token, state)
        collect.append(next_token.numpy()[0])

    poem = [id2word[t] for t in collect if id2word[t] not in ['bos', 'eos']]
    return ''.join(poem)
print(gen_sentence(start_words="月"))
```

```
0.0s  Python
月。然不可怜，不得无人间。有不可见，不知何处归。然不可见，不得一年年。有无人事，何人不可怜。
```

实验总结

1. **训练效果**
   - 损失从初始值 8.821 下降至 5.196，模型逐渐收敛。
   - 生成诗歌的连贯性随训练轮次增加而提升。
2. **问题分析**
   - 简单 RNN 存在梯度消失，生成诗歌较短且重复。
   - 可替换为 LSTM 或 GRU 以提升长文本生成能力。
3. **改进方向**
   - 使用更大的数据集和更深层的网络。
   - 引入注意力机制或预训练模型。
4. **结论**
   RNN 在诗歌生成任务中能学习基本模式，但更复杂的模型（如 `LSTM`）和策略（如 `Beam Search`）可显著提升生成质量。