同濟大學

TONGJI UNIVERSITY

实验报告

成员	王欣玥（2253300）
学院（系）	国豪书院
专　　业	信息安全
任课教师	程大伟
日　　期	2025 年 3 月 24 日

1、RNN、LSTM、GRU 解释

RNN（循环神经网络）：是一种循环神经网络，用于处理序列数据，具有循环连接，使得它可以在处理序列时保持一种记忆状态。在 RNN 中，每个时间步都有一个隐藏状态，它可以接收当前时间步的输入和上一个时间步的隐藏状态作为输入。隐藏状态的输出不仅取决于当前时间步的输入，还取决于之前所有时间步的输入。

计算公式：设：

x_t：时间步 t 的输入

h_t：时间步 t 的隐藏状态

W_h,W_x,b：网络的参数

f：激活函数

则 RNN 的隐藏状态计算：$h_t=f(W_h h_{t-1}+W_x x_t+b)$，最后，通过一个全连接层计算输出：$yt=W_y h_t+b_y$

LSTM（长短时记忆网络）：是一种改进的循环神经网络架构，旨在解决传统 RNN 中的梯度消失和梯度爆炸问题，以及增强对长期依赖关系的建模能力。LSTM 引入了一个记忆单元，该单元可以存储和访问信息，并通过门控机制来控制信息的流动。LSTM 的关键部分包括输入门、遗忘门、输出门。

输入门：输入门的计算公式有两个, 第一个就是产生输入门门值的公式,意味着输入信息有多少需要进行过滤。输入门的第二个公式是与传统 RNN 的内部结构计算相同。

遗忘门：首先将当前时间步输入 x(t)与上一个时间步隐含状态 h(t-1)拼接, 得到[x(t), h(t-1)], 再通过一个全连接层做变换, 最后通过 sigmoid 函数进行激活得到 f(t),f(t)代表遗忘过去的多少信息。

输出门：输出门部分的公式有两个, 第一个是计算输出门的门值。第二个即是使用这个门值产生隐含状态 h(t), 他将作用在更新后的细胞状态 C(t)上, 并做 tanh 激活,最终得到 h(t)作为下一时间步输入的一部分。输出门是为了产生隐含状态 h(t)。

GRU（门控循环单元）：GRU 是 LSTM 的简化版本，减少了计算复杂度，同时保留了 LSTM 的长时依赖学习能力。GRU 仅使用两个门：更新门（Update Gate）：决定当前时间步的信息如何影响过去的隐藏状态。重置门（Reset Gate）决定当前时间步的信息如何结合过去的隐藏状态。

2、 诗歌生成过程

(1) 读取并解析诗歌数据：读取 poems.txt 诗歌文本文件，提取诗歌正文，并在诗歌前后添加

特殊标记 bos（开始） 和 eos（结束）且过滤过长的诗歌（长度超过 200）。

(2) 词汇表构建：统计字符的出现频率，并按频率排序，构建词汇表。生成字→ID 和 ID→字映射。然后将诗歌转换为数字索引

(3) 构建训练数据集：加载数据集，对其进行批量填充，使所有诗歌长度一致。设置输入和标签：输入 x：诗歌的前 n-1 个字符。目标 y：诗歌的后 n-1 个字符。

(4) 构建诗歌生成模型并训练模型：结构为：词嵌入层：将字符 ID 转换为 64 维向量，RNN 层：使用 128 维隐藏状态学习上下文。全连接层：计算词汇表中所有字符的概率

(5) 诗歌生成：以 begin_word 作为诗歌的起点逐步预测下一个字符，直到达到 max_length 或遇到 eos。

3、生成结果

生成诗歌代码：

```
def generate_poem(begin_word='日', max_length=50):
    if begin_word not in word2id:
        print(f"警告: '{begin_word}' 不在词典中，使用默认起始词 'bos'")
        begin_word = 'bos'

    state = [tf.random.normal(shape=(1, 128), stddev=0.5)]
    cur_token = tf.constant([word2id[begin_word]], dtype=tf.int32)
    generated = [begin_word]

    for _ in range(max_length):
        cur_token, state = model.get_next_token(cur_token, state)
        next_word = id2word[cur_token.numpy()[0]]
        if next_word == 'eos':
            break
        generated.append(next_word)

    return ''.join(generated)


begin_words = ['日', '红', '山', '夜', '湖', '海', '月']
for word in begin_words:
    print(f"以 '{word}' 开头的诗歌: {generate_poem(word)}")
```

结果：

```
以 '日' 开头的诗歌：日暮茫茫茫递寺阳渍，落花花落。
以 '红' 开头的诗歌：红铲，一条红叶满花开。
以 '山' 开头的诗歌：山色，江南春水月，春风落花声。
以 '夜' 开头的诗歌：夜，一年年日不知君，不得人间不可知。
以 '湖' 开头的诗歌：湖上寺中。
以 '海' 开头的诗歌：海畔斜。
以 '月' 开头的诗歌：月上楼花不见，一枝花落不堪归。
```

装

订

线