# 深度学习与自然语言处理第四次作业

李家哲 ZY2203105
lijiazhebuaa@buaa.edu.cn

## 作业要求

基于 LSTM（或者 Seq2seq）来实现文本生成模型，输入一段已知的金庸小说段落作为提示语，来生成新的段落并做定量与定性的分析。（训练语料可以只选择上面的语料库，也建议增加更多的语料作为训练数据）。

## 主要方法

### M1: LSTM 模型

LSTM（长短期记忆网络）是 RNN 的一种变体，目的是为了解决长期记忆问题，其结构如下图所示。

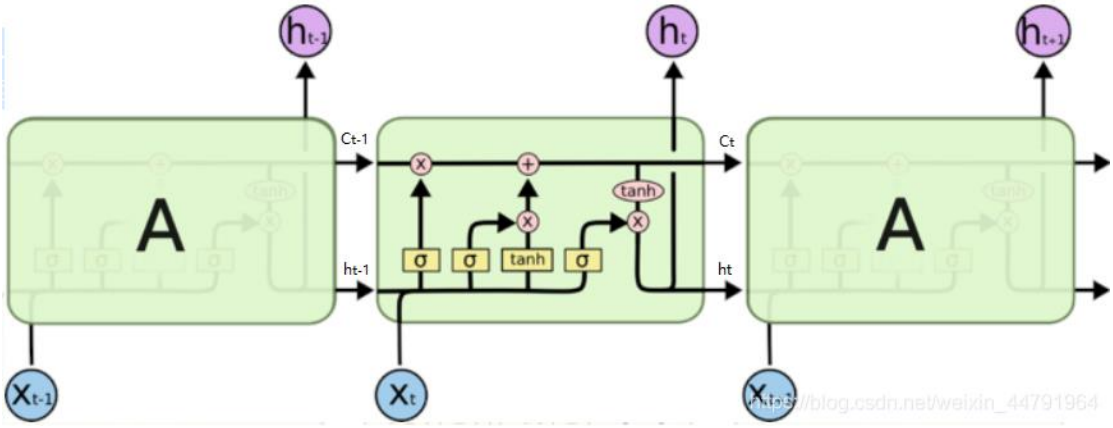



图 1　LSTM 结构

我们可以看出，在 n 时刻，LSTM 的输入有三个:

- 当前时刻网络的输入值 $X_t$；
- 上一时刻 LSTM 的输出值 $h_{t-1}$；
- 上一时刻的单元状态 $c_{t-1}$。

LSTM 的输出有两个:

- 当前时刻 LSTM 输出值 $h_t$；

- 当前时刻的单元状态 $c_t$ 。

LSTM 用两个门来控制单元状态 $c_n$ 的内容: 、

- 遗忘门（forget gate），它决定了上一时刻的单元状态 $c_{n-1}$ 有多少保留到当前时刻；
- 输入门（input gate），它决定了当前时刻网络的输入 $c'_n$ 有多少保存到新的单元状态 $c_n$ 中。

LSTM 用一个门来控制当前输出值 $h_n$ 的内容:

- 输出门（output gate），它利用当前时刻单元状态 $c_n$ 对 $h_n$ 的输出进行控制。

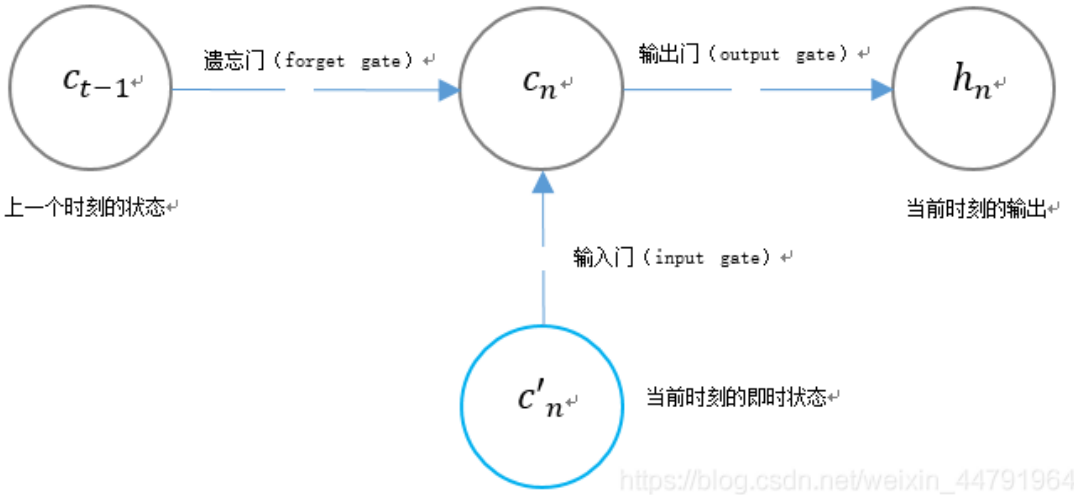



图 2　LSTM 门结构

**（a）遗忘门**

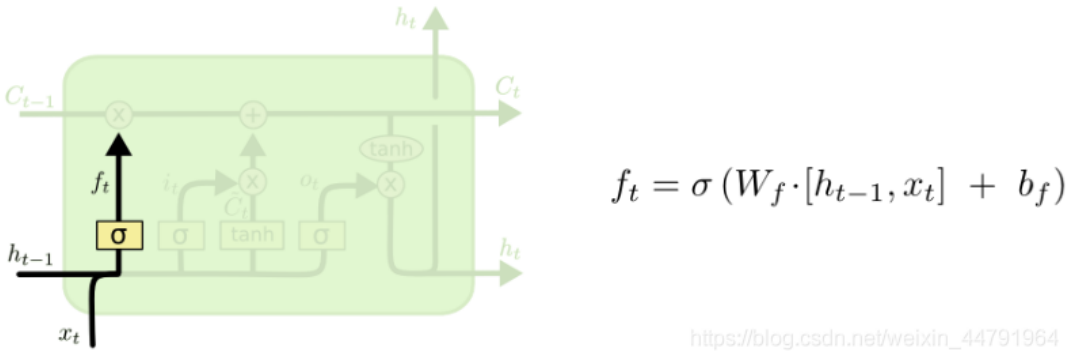



图 3　遗忘门

遗忘门这里需要结合 $h_{t-1}$ 和 $X_t$ 来决定上一时刻的单元状态 $c_{t-1}$ 有多少保留到当前时刻；

由图我们可以得到，我们在这一环节需要计算一个参数 $f_t$。

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

其中需要训练的参数分别是$W_f$和$b_f$。

在定义 LSTM 的时候我们会使用到一个参数叫做 units，其实就是神经元的个数，也就是 LSTM 的输出——$h_t$ 的维度。

所以：

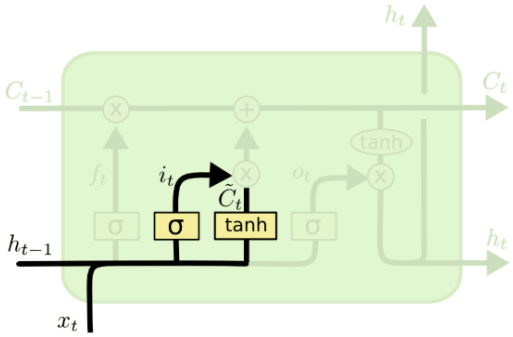
$$W_f\text{的参数量} = (x_{\text{dim}} + h_{\text{dim}}) * h_{\text{dim}}$$

$$b_f\text{的参数量} = h_{\text{dim}}$$

遗忘门的总参数量为：

$$\text{总参数量} = ((\mathrm{x}_{\text{dim}} + \mathrm{h}_{\text{dim}}) * \mathrm{h}_{\text{dim}} + \mathrm{h}_{\text{dim}})$$

**（b）输入门**




$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



图 4　输入门

输入门这里需要结合 $h_{t-1}$ 和 $X_t$ 来决定当前时刻网络的输入 $c_n^{'}$ 有多少保存到单元状态 $c_n$ 中。由图我们可以得到，我们在这一环节需要计算两个参数，分别是 $i_t$。

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

和 $c_t^{'}$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

里面需要训练的参数分别是$W_i$、$b_i$、$W_C$和$b_C$。

$$W_i\text{的参数量} = (x_{\text{dim}} + h_{\text{dim}}) * h_{\text{dim}}$$

$$b_i\text{的参数量} = h_{\text{dim}}$$

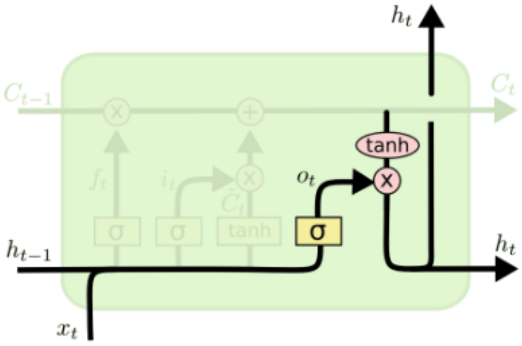
$$W_c\text{的参数量} = (x_{\text{dim}} + h_{\text{dim}}) * h_{\text{dim}}$$

$$b_c\text{的参数量} = h_{\text{dim}}$$

输入门的总参数量为：

$$\text{总参数量} = 2*((\text{x}_{\text{dim}} + \text{h}_{\text{dim}})*\text{h}_{\text{dim}} + \text{h}_{\text{dim}})$$

**（c）输出门**




$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



图 5　输出门

输出门利用当前时刻单元状态 $c_n$ 对 $h_n$ 的输出进行控制；

由图我们可以得到，我们在这一环节需要计一个参数 $o_t$ 。

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

里面需要训练的参数分别是 $W_o$ 和 $b_o$ 。

$$W_o\text{的参数量} = (x_{\text{dim}} + h_{\text{dim}}) * h_{\text{dim}}$$

$$b_o\text{的参数量} = h_{\text{dim}}$$

输出门的总参数量为：

$$\text{总参数量} = ((\text{x}_{\text{dim}} + \text{h}_{\text{dim}})*\text{h}_{\text{dim}} + \text{h}_{\text{dim}})$$

**（d）全部参数量**

所有的门总参数量为：

$$\text{总参数量} = 4*((\text{x}_{\text{dim}} + \text{h}_{\text{dim}})*\text{h}_{\text{dim}} + \text{h}_{\text{dim}})$$

## M2：算法流程

step1：读取样本数据

选择金庸先生 16 本中长篇小说中的任意一本作为数据集（暂不支持选择全部的小说），对数据集进行预处理：1.删除开头的无关信息；2.删除文中的隐藏符号（换行符、分页符等）。

注意：与前几次实验不同，本次实验需要保留中文的标点符号，如“”，。？！等，且不需要删除停词，这样可以使得训练出的语句可以正常断句。使用 jieba 以词为单位进行分词，将分词结果返回作为训练数据。

```
    def data_processing(file_path):
    """
    获取文件信息并预处理
    :param file_path: 文件名对应路径
    :return data_out: 字符串形式的语料库
    :return words_out: 分词
    """
    # 读取小说
    delete_symbol = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@★、…【】《》
\'[\\]^_`{|}~「」『』（）]+'
    with open(file_path, 'r', encoding='ANSI') as f:
        data_out = f.read()
        data_out = data_out.replace('本书来自 www.cr173.com 免费 txt 小说下
载站\n 更多更新免费电子书请关注 www.cr173.com', '')
        data_out = re.sub(delete_symbol, '', data_out)
        data_out = data_out.replace('\n', '')
        data_out = data_out.replace('\u3000', '')
        data_out = data_out.replace(' ', '')
        f.close()
    # 以词为单位进行分词
    words_out = list(jieba.cut(data_out))
    return data_out, words_out
```

step2：训练 LSTM 模型并进行预测

将分词结果作为数据集，首先需要进行数据转换，将数据处理成 LSTM 模型要求的格式。去除数据集中重复的单词，并创建单词到索引的映射，将中文单词映射为模型能处理的数字形式，并统计数据集中的字符数和词汇量。

对整体的数据集，以 3 个词为间隔，分解成若干个长度为 60 词的段落，作为模型的输入；以段落对应的下一个词为模型的输出。输入数据需转换为 LSTM 需要的格式，即一个三维数组，第一个维度为样本数量，第二个维度为样本的时间步长，第三个维度为样本序列对应的单元数(特征数)。输出数据需转换为 one-hot 形式，在本例中即一个与映射同维度的 01 一维数组，只有输出单词对应的索引处为 1，其余部分均为 0。

然后，创建 LSTM 模型。首先添加一个 embedding 层进行文本向量化，然后在添加一个 LSTM 层，维度选为 256，最后添加一个 Dense 层。定义模型的检查点，当学习率低于某个值后，认为模型不再改善并中断训练。利用处理完成的数据进行模型训练。

当训练结束后，利用给定的文本，将其转为 LSTM 标准格式，输入到模型中，预测出下一个词，再将文本整体后移一个词后再次输入到模型中进行预测，以此循环，得到往后预测 100 或 200 个词的结果。

```
    def lstm(words_in):
    """
    获取文件信息并预处理
    :param words_in: 数据集
    """
    # 创建字符与数字的映射
    chars = list(set(words_in))
    char_to_int = dict((c, i) for i, c in enumerate(chars))

    # 统计数据集中的字符数和词汇量
    n_chars = len(words_in)
    n_vocab = len(chars)

    # 设定 LSTM 模型的参数
    seq_length = 60
    step = 3
    X_data = []
    y_data = []
    for i in range(0, n_chars - seq_length, step):
        seq_in = words_in[i:i + seq_length]
        seq_out = words_in[i + seq_length]
        X_data.append([char_to_int[char] for char in seq_in])
        y_data.append(char_to_int[seq_out])
    n_patterns = len(X_data)
    # 将输入数据转换为 LSTM 模型需要的格式
    X = np.reshape(X_data, (n_patterns, seq_length, 1))
    y = []
    for i in y_data:
        temp = [0] * n_vocab
        temp[i] = 1
        y.append(temp)
    y = np.array(y)
    # 创建 LSTM 模型
    model = Sequential()
    model.add(Embedding(y.shape[1], 256))
    # model.add(Dropout(0.2))
    model.add(LSTM(256))
    model.add(Dense(y.shape[1], activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam')
    # 定义模型的检查点
    # filepath = "weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
    # checkpoint = ModelCheckpoint(filepath, monitor='loss',
verbose=1, save_best_only=True, mode='min')
```

```
    # callbacks_list = [checkpoint]
    callbacks_list = [
        keras.callbacks.ModelCheckpoint(  # 在每轮完成后保存权重
            filepath='text_gen.h5',
            monitor='loss',
            save_best_only=True,
        ),
        keras.callbacks.ReduceLROnPlateau(  # 不再改善时降低学习率
            monitor='loss',
            factor=0.5,
            patience=1,
        ),
        keras.callbacks.EarlyStopping(  # 不再改善时中断训练
            monitor='loss',
            patience=3,
        ),
    ]

    # 训练 LSTM 模型
    model.fit(X, y, epochs=200, callbacks=callbacks_list)

    # 使用训练好的模型生成新的文本
    num = 200
    text = '这少女这两下轻轻巧巧的刺出，戳腕伤目，行若无事，不知如何，那吴国剑士竟是避让不过。余下七名吴士大吃一惊，一名身材魁梧的吴士提起长剑，剑尖也往少女左眼刺去。剑招嗤嗤有声，足见这一剑劲力十足。'
    # text = '两名剑士各自倒转剑尖，右手握剑柄，左手搭于右手手背，躬身行礼。两人身子尚未站直，突然间白光闪动，跟着铮的一声响，双剑相交，两人各退一步。旁观众人都是"咦"的一声轻呼。'
    text_cut = list(jieba.cut(text))[:seq_length]
    pattern = [char_to_int[char] for char in text_cut]
    print("Seed:")
    print(''.join(text_cut))
    for i in range(num):
        x = np.reshape(pattern, (1, len(pattern), 1))
        prediction = model.predict(x, verbose=0)
        index = np.argmax(prediction)
        result = chars[index]
        print(result, end='')
        pattern.append(index)
        pattern = pattern[1:len(pattern)]
```

# 实验结果

由于受到电脑性能的限制，本实验选取 16 本小说中篇幅最短的一篇《越女剑》进行实验。此外，由于电脑性能限制，为了平衡训练时间，参数调整范围有限，得到的结果可能不甚理想，本实验仅为验证演示。

程序中设定训练次数 epochs=200，由于存在检查点，当模型不再改善时会中断训练，某次实验中在 “Epoch 110/200” 处中断训练，此时学习率降为 “lr: 4.7684e-10” (详细训练过程见 result.txt )。

模型训练完成后，给定两段开头，利用模型进行预测求解，得到的结果如下:

**（a）给定开头 1 往后预测 100 个词**

程序跑两次，得到两个结果如下所示

---

**结果 1：**

Seed:

两名剑士各自倒转剑尖，右手握剑柄，左手搭于右手手背，躬身行礼。两人身子尚未站直，突然间白光闪动，跟着铮的一声响，双剑相交，两人各退一步。旁观众人都是“咦”的一声轻呼。

*三剑二青衣剑士竟不挡，身子给是越国宣到欺得手指？”范蠡叫道“姑娘，原来我是四个的草地。”勾践道“打早已兜，直削！”勾践道“风打断比剑，只除给我们伍子胥说“伍子胥公鸡。两人一商量，少说都有有的名字声响。眼前吴士，不见这场极厚阴谋，一双一阵剑，又狞笑迷惑上乘，事剑士面前，直如以高兴羞惭之中之中，江边攻吴此则只*

**结果 2：**

Seed:

两名剑士各自倒转剑尖，右手握剑柄，左手搭于右手手背，躬身行礼。两人身子尚未站直，突然间白光闪动，跟着铮的一声响，双剑相交，两人各退一步。旁观众人都是“咦”的一声轻呼。

*”八名剑士微微大笑，我要他们伍子胥羊儿，轻纱了伍子胥剑。他兵败这人终究的长蛇的长蛇的女郎影子。只须青衣剑士长剑，一个“他为声响，且的丽影，过。阿青是竹棒也觉她！他抬头这人之劲，随即以往劲气。眼前吴士又了吴国邻近来斤工布，不料吴士将这这本事的站着几名少女，听到吴纯钧的名字影子我邻近小腹。从少女见文种咩*

---

**（b）给定开头 2 往后预测 200 个词**

---

Seed:

这少女这两下轻轻巧巧的刺出，戳腕伤目，行若无事，不知如何，那吴国剑士竟是避让不过。余下七名吴士大吃一惊，一名身材魁梧的吴士提起长剑，剑尖也往少女左眼刺去。剑招嗤嗤有声，足见这一剑劲力十足。

*他抬头映面，听到的一声，又利剑和吴国和吴国吴国，谨供，谨供。另外勾践到得“你师父我的羊儿，只除师哥！”范蠡又道“姑娘，咱们快快剑却对手一百只和这勾践便便便便便便便去了他为吴国，不料他手中说在吴国，未必呼呼数十丈。但众更是将适才便跟她神情半天的乡下姑娘，每天温软了你又我大王，我时时还要还要他很你不许腻。那九术一声，惊恐越国找来和这少女落地。吴士吴士四名这少女吴士的好，心下中。后来勾践道“打大夫，原来不是到伍子胥说，那他的心千口。他抬头映面，退开，江边勾践在街边。第三天。七名吴士*

*轻轻了道“此剑姑娘，原来小人胡子，我要“你师父我？”范蠡轻轻道“你打断应声，像“阿青姑娘，教你到我*

---

从上述结果可以看出，模型所生成的文本有着鲜明的金庸先生武侠小说的语言特色，但是从逻辑上来看有所欠缺，所生成的文本逻辑不通，无法进行正常阅读，且会出现连续的词语如 *“和吴国和吴国吴国，谨供，谨供”*、*“这勾践便便便便便便便去了”* 等，并且生成文本似乎与所给定开头段落不太相符。

为了改善预测效果，可以采取增加 LSTM 神经元数量、增加隐藏层数、增加段落的时间步长、减少段落的词间距等方法，或者采用更加有效的时间序列预测模型。由于设备限制，本实验不在深入。

# 结论

本实验利用金庸先生的小说作为数据集，进行了文本生成模型的研究，采用了被广泛使用的 LSTM 长短时记忆网络进行训练。实验中仅采用了最基本的网络模型，未加改进，其训练速度较长，得到的结果也差强人意。在今后的研究任务中可以探索更为先进的神经网络模型来进一步优化。

# 参考文献

[1] Bubbliiiing，神经网络学习小记录 36——Keras 实现 LSTM 与 LSTM 参数量详解，https://blog.csdn.net/weixin_44791964/article/details/103896884

[2] 小乖乖的臭坏坏，理解 LSTM 中的输入和输出形状 tf.keras.layers.LSTM（以及对于 return_sequences 的解释），https://blog.csdn.net/weixin_42887138/article/details/122323092