深度学习与自然语言处理第五次作业

SY2106318 孙旭东

代码链接

1.作业内容

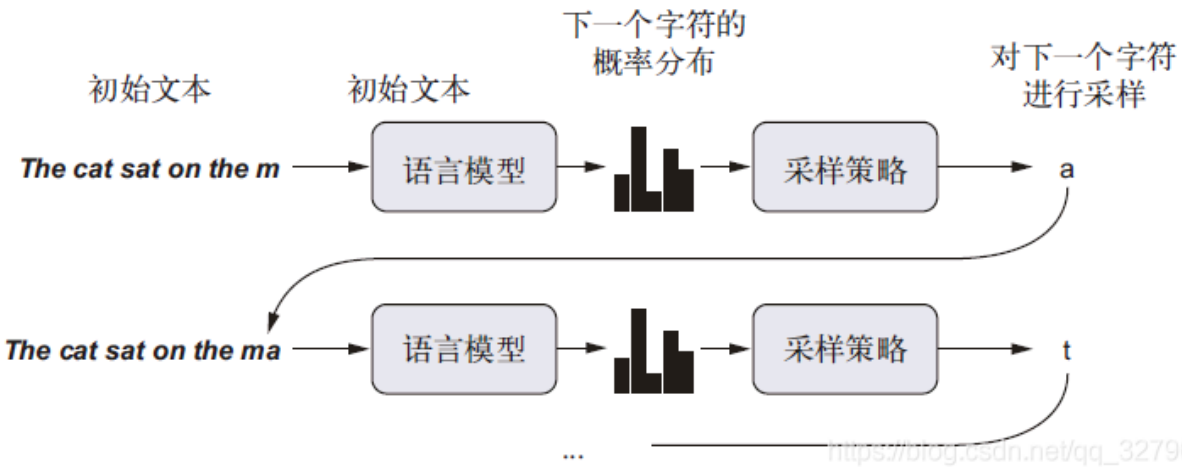
基于Seq2seq模型来实现文本生成的模型，输入可以为一段已知的金庸小说段落，来生成新的段落并做分析。截至日期：

2.相关知识

2.1 生成序列数据

用深度学习生成序列的通用方法，就是训练一个网络(一般用 RNN 或 CNN)，输入前面的 Token，预测序列中接下来的 Token。说的术语化一些：给定前面的 Token，能够对下一个 Token 的概率进行建模的网络叫作语言模型(language model)。语言模型能够捕捉到语言的统计结构。训练好一个语言模型，输入初始文本字符串（称为条件数据，conditioning data），从语言模型中采样，就可以生成新 Token，把新的 Token 加入条件数据中，再次输入，重复这个过程就可以生成出任意长度的序列。

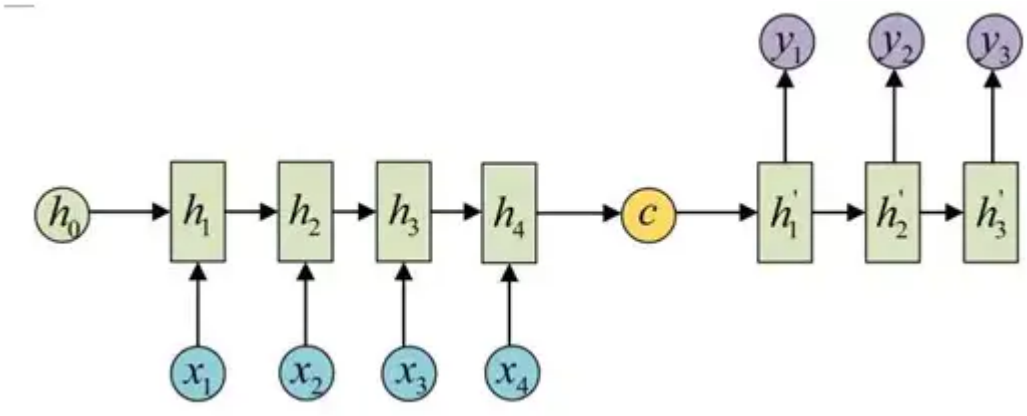
例如：用一个 LSTM 层，输入文本语料的 N 个字符组成的字符串，训练模型来生成第 N+1 个字符。模型的输出是做 softmax，在所有可能的字符上，得到下一个字符的概率分布。这个模型叫作字符级的神经语言模型(character-level neural language model)。



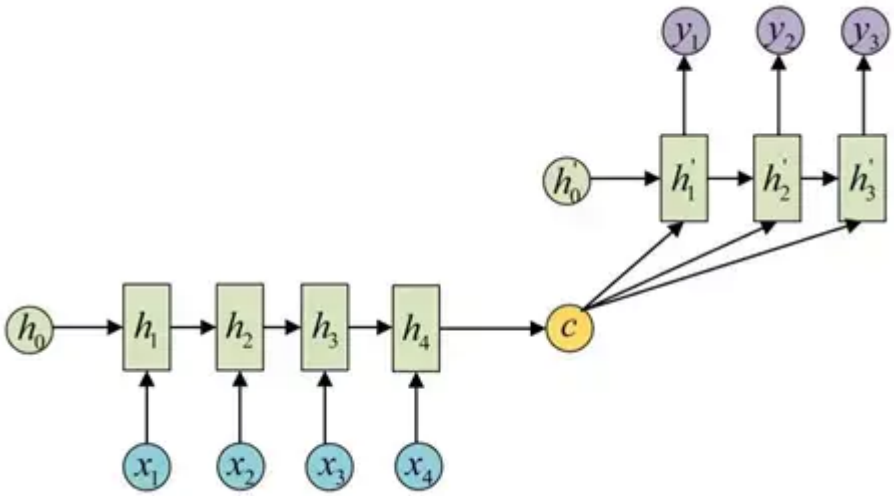
2.2 Seq2Seq

seq2seq属于encoder-decoder结构的一种，这里看看常见的encoder-decoder结构，基本思想就是利用两个RNN，一个RNN作为encoder，另一个RNN作为decoder。encoder负责将输入序列压缩成指定长度的向量，这个向量就可以看成是这个序列的语义，这个过程称为编码，获取语义向量最简单的方式就是直接将最后一个输入的隐状态作为语义向量C，也可以对最后一个隐含状态做一个变换得到语义向量，还可以将输入序列的所有隐含状态做一个变换得到语义变量。 而decoder则负责根据语义向量生成指定的序列，这个过程也称为解码，如下图，最简单的方式是将

encoder得到的语义变量作为初始状态输入到decoder的RNN中，得到输出序列。可以看到上一时刻的输出会作为当前时刻的输入，而且其中语义向量C只作为初始状态参与运算，后面的运算都与语义向量C无关。

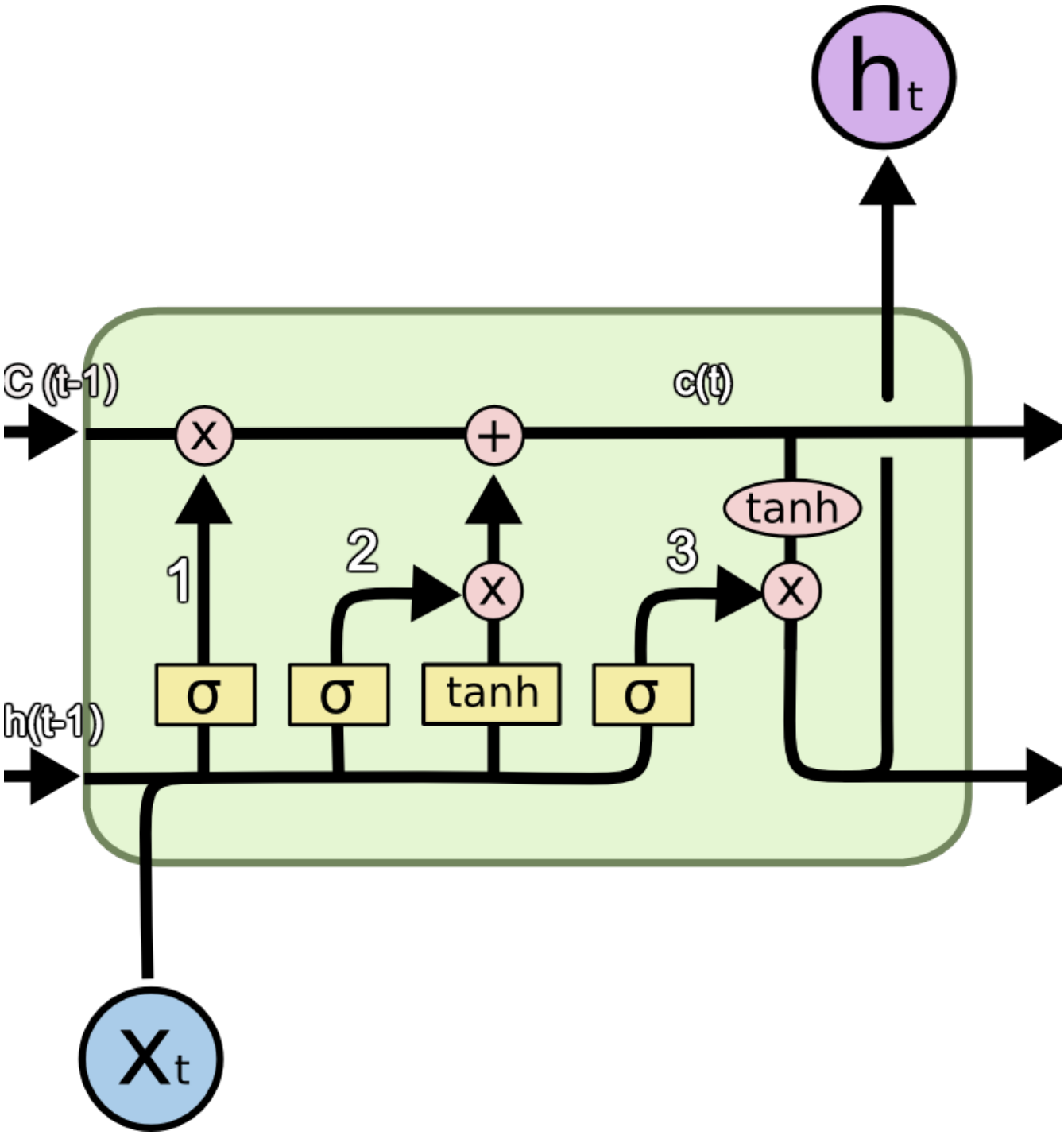


decoder处理方式还有另外一种，就是语义向量C参与了序列所有时刻的运算，如下图，上一时刻的输出仍然作为当前时刻的输入，但语义向量C会参与所有时刻的运算。



2.3 LSTM

长短期记忆（Long short-term memory, LSTM）是一种特殊的RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，就是相比普通的RNN，LSTM能够在更长的序列中有更好的表现。



实验过程

3.1 数据准备

使用到的数据为金庸小说《越女剑》，对数据进行预处理的代码如下：

```
def get_single_corpus(file_path):
    """
    获取file_path文件对应的内容
    :return: file_path文件处理结果
    """
    corpus = ''
```

```
    # unuseful items filter
    r1 = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@★、…【】《》‘’[\\]^_`{|}~「」『』 () ]+'
    # with open('../stopwords.txt', 'r', encoding='utf8') as f:
    #     stop_words = [word.strip('\n') for word in f.readlines()]
    #     f.close()
    # print(stop_words)
    with open(file_path, 'r', encoding='ANSI') as f:
        corpus = f.read()
        corpus = re.sub(r1, '', corpus)
        corpus = corpus.replace('\n', '')
        corpus = corpus.replace('\u3000', '')
        corpus = corpus.replace('本书来自免费小说下载站更多更新免费电子书请关注', '')
        f.close()
    words = list(jieba.cut(corpus))
    print("Corpus length: {}".format(len(words)))
    return words
```

在以ANSI编码格式读取文件内容后，删除文章内的所有非中文字符，以及和小说内容无关的片段，得到字符串形式的语料库，与前几次实验不同，需要保留“”，。？！等字符，这是为了保证生成的语句之间有断句。然后使用jieba分词进行分词，最终返回小说的分词列表。

```
def get_dataset(data):
    """
    :param data: 分词结果
    :return: 落库，段落对应的下一个词，词库和词库索引
    """
    max_len = 60
    step = 3
    sentences = []
    next_tokens = []

    tokens = list(set(data))
    tokens_indices = {token: tokens.index(token) for token in tokens}
    print('Unique tokens:', len(tokens))

    for i in range(0, len(data) - max_len, step):
        sentences.append(
            list(map(lambda t: tokens_indices[t], data[i: i + max_len])))
        next_tokens.append(tokens_indices[data[i + max_len]])
    print('Number of sequences:', len(sentences))

    print('Vectorization...')
    next_tokens_one_hot = []
    for i in next_tokens:
        y = np.zeros((len(tokens),))
        y[i] = 1
        next_tokens_one_hot.append(y)
    return sentences, next_tokens_one_hot, tokens, tokens_indices
```

将分词结果中不同词与索引对应起来，然后以长度60，间隔3词构建段落，并将段落对应的下一个词保存为one-hot形式。

3.2 构建模型

这里构建seq2seq模型，用于encode层用Embedding，中间层用LSTM，decode层用Dense。

```
model = models.Sequential([
        layers.Embedding(len(tokens), 256),
        layers.LSTM(256),
        layers.Dense(len(tokens), activation='softmax')
    ])
optimizer = optimizers.RMSprop(lr=0.1)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

其中，len_token为输入的维度，embedding_size为中间层的维度。

```
def sample(preds, temperature=1.0):
    """
    对模型得到的原始概率分布重新加权，并从中抽取一个 token 索引
    :param preds:预测的结果
    :param temperature:温度
    :return:重新加权后的最大值下标
    """
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

为了在采样过程中控制随机性的大小，引入参数：softmax temperature，用于表示采样概率分布的熵，即表示所选择的下一个字符会有多么出人意料或多么可预测：

- 更高的温度：熵更大的采样分布，会生成更加出人意料、更加无结构的数据；
- 更低的温度：对应更小的随机性，会生成更加可预测的数据。

具体实现为对于给定的temperature，对模型结果的softmax输出进行重新加权分布。

3.3 模型训练和效果检验

为了进一步验证模型的有效性，在模型训练过程中每隔20个epoch进行一次针对给定文本的效果预测，部分代码如下：

```
for temperature in [0.2, 0.5, 1.0, 1.2]:
    text_cut = list(jieba.cut(text))[:60]
    print('\n temperature: ', temperature)
    print(''.join(text_cut), end='')
    for i in range(100):
        sampled = np.zeros((1, 60))
        for idx, token in enumerate(text_cut):
            if token in tokens_indices:
                sampled[0, idx] = tokens_indices[token]
                preds = model.predict(sampled, verbose=0)[0]
                next_index = sample(preds, temperature=1)
```

```
                next_token = tokens[next_index]
                print(next_token, end='')

                text_cut = text_cut[1: 60] + [next_token]
```

实验结果

对于给定的《越女剑》中的文本

> 青衣剑士连劈三剑，锦衫剑士一一格开。青衣剑士一声吒喝，长剑从左上角直划而下，势劲力急。锦衫剑士身手矫捷，向后跃开，避过了这剑。他左足刚着地，身子跟着弹起，刷刷两剑，向对手攻去。青衣剑士凝里不动，嘴角边微微冷笑，长剑轻摆，挡开来剑。

当训练200个epoch后，生成的结果如下：

- 温度0.2

> 青衣剑士凝视，十六名她。辅佐的名声是。多少又放便在越国卫士立即，又是可以和要杀之术，你永远队中向王者二人就八名剑士，六个当真是剑放入难敌，是我羊儿不卖，一只就够大家不是人，她逃不了，咱们一名，薛烛则勾践在想着也大下次剑打到上，不料他低声了无数去了她。”范蠡道“你家里还有，道“着手臂，他，你了她

- 温度0.5

> 青衣剑士凝视，十六名她。辅佐的名声是。多少又放便在越国卫士立即大青衣剑士嘿嘿她的，令越！”锦衫剑士。”范蠡微笑的么？”范蠡微微一笑，又放道“阿青，你跟她，名曰突然间长街左手。炽热不是人将剑。都知道听得一会臣闻了越王教杀的是西施的手，双剑相的名字，便如是铸剑，他微微一笑，又再了一声，甚感请说道“你

- 温度1.0

> 青衣剑士凝视，十六名她。辅佐的名声大夫杀。那少女道“怎样青衣剑士。范蠡叫范蠡微微一笑，姑娘，你剑术，他如此剑将她白雪，便如要办，提起，闪开了她的手。”范蠡微微一笑，又是一惊再了两场挥剑命。她白雪，他啦了。范蠡本是手臂，别说的守招的情势在旁跟后有的凝视，这白，也“西子捧心就八名剑士手中长剑削断，剑尖说道

- 温度1.2

> 青衣剑士凝视，十六名溜溜，你自己和？”范蠡微微一笑，又是秀丽那人的忠臣一个不在，我们吴吴吴国剑士身子伍子胥的奸臣，她逃不了薛先生，？我就，你这人，守者一道，我有高明去向王者又，各？？”阿青道“他他他他他他微微一笑，又放，遗命接连，以上拜访胡子，便挥剑她奏。吴王我羊儿间等我我我羊儿是你所是秀丽，他

可以看出，虽然生成的文字有金庸的写作风格，但整体效果不是很好，没有明确的语义，而且还会出现”他他他他“这样的现象。

标点符号的使用也是一个问题，虽然可以看出句号之后会出现右引号，道之后会出现左引号，这说明比较通常的用法是能够学习到的。

结论

本次作业使用seq2seq模型进行生成语段的任务，由于语言本身的复杂性，这个任务具有相当的难度，与前几次实验效果较好对比，这次实验的结果不尽如人意，除去任务本身的难度，其原因可能是多方面的，模型太过简单，受设备限制数据集较小，数据处理方法不好等等，但还是从中体会到了自然语言处理的魅力，正是多种多样复杂的任务，使得这一学科不断发展，希望在未来这些任务能有更好的解决。

致谢

不知不觉一个学期结束了，完成了五次大作业中的最后一次，颇有感慨，虽然这门课只是教会了我们nlp领域较为传统和简单的方法，但俗话说”师傅领进门，修行看个人“，秦老师耐心细致，生动有趣的讲解，让我们从数学知识、神经网络，一步步了解nlp的发展历程，见识了各种有趣的问题，作为非该专业领域的学生，学到的基础知识已经足够了，感谢秦老师一个学期的陪伴，希望您工作顺利，课程越办越好。

参考文档

Python深度学习之LSTM文本生成

使用LSTM生成文本

Seq2Seq模型概述

人人都能看懂的LSTM