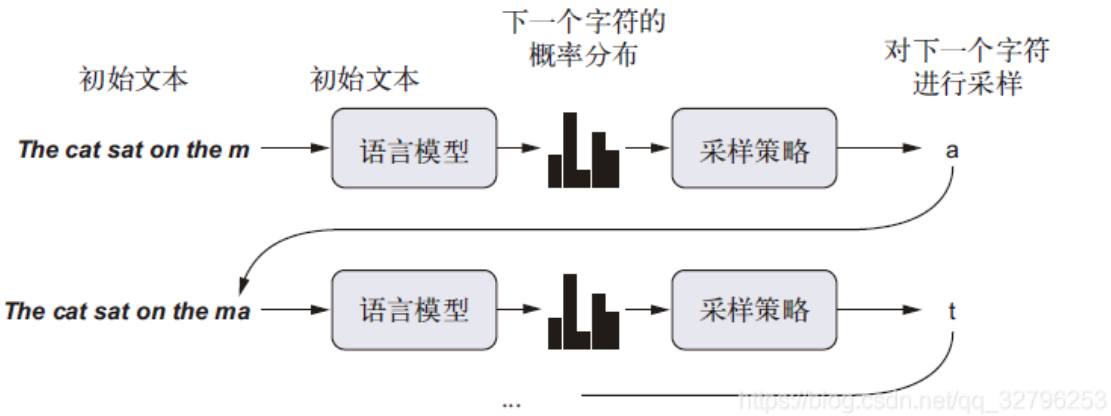
DLNLP_Homework4 王彪-ZY2203114

作业四：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。

1. 序列生成

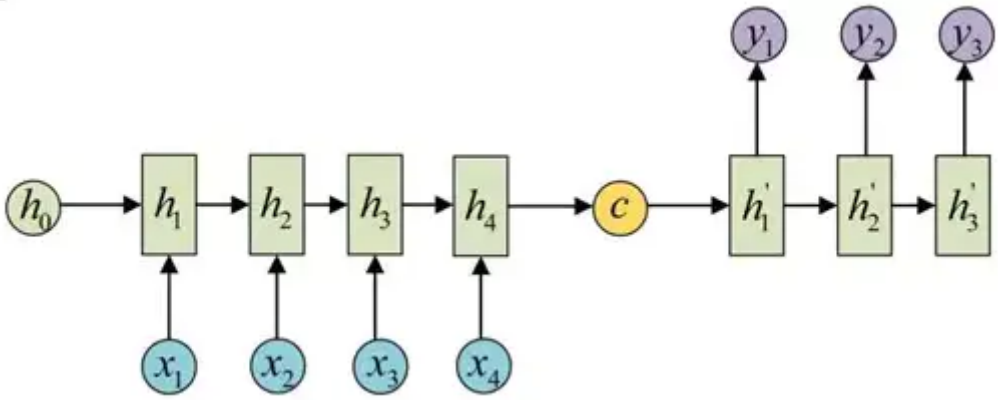
基于深度学习生成序列的通用方法，一般是训练一个循环神经网络（RNN），输入前序的token，预测序列中接下来的token。也就是，给定前序的token，能够对下一个token的概率进行建模的网络叫做语言模型。语言模型能够捕捉到语言的统计结构，当训练好一个语言模型后，输入初始的文本字符串（称为条件数据），从语言模型中采样，就可以生成新token，把新的token加入条件数据中，再次输入，重复这个过程就可以生成任意长度的序列。

例如：使用一个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处理，在所有可能的字符上，得到下一个字符的概率分布。这个模型叫做字符级的神经语言模型。

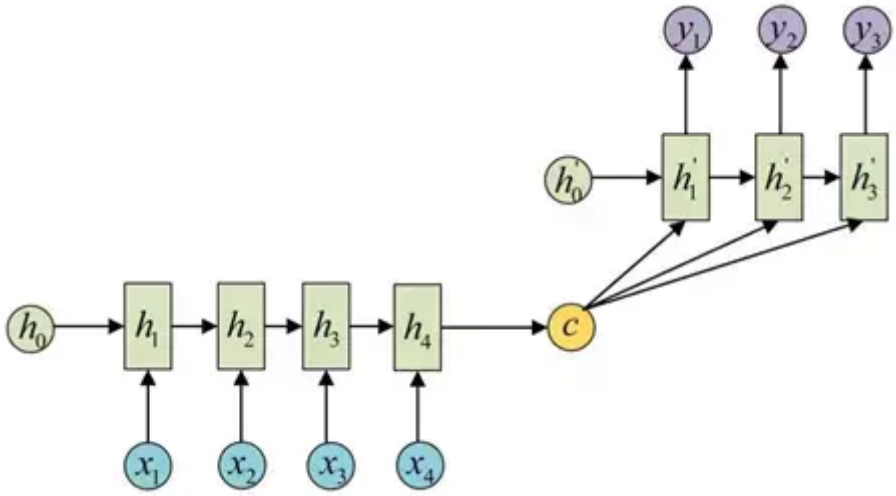


2. Seq2Seq

seq2seq属于encoder-decoder结构的一种，这里常常见的encoder-decoder结构，基本思想就是利用两个RNN，其中一个作为编码器，另一个用来作为解码器。encoder负责将输入序列压缩维指定长度的向量，这个向量就是这个序列的语义信息，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态作为语义向量，也可以对最后一个隐态做一个变换得到语义向量，还可以将输入状态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。

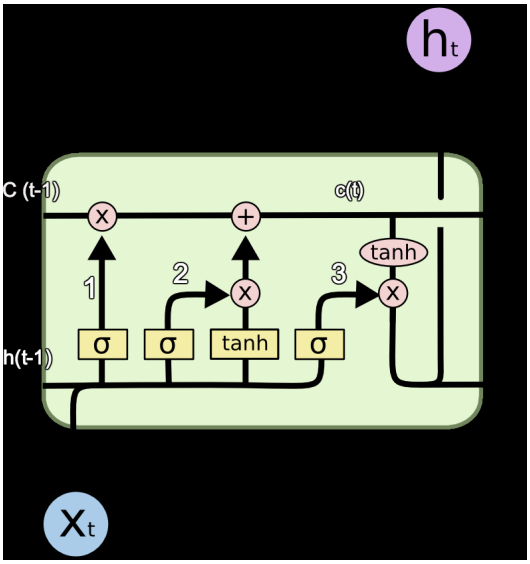


decoder处理方式还有另外一种，就是语义向量C参与了序列所有时刻的运算，如下图，上一时刻的输出仍然作为当前时刻的输入，但语义向量C会参与所有时刻的运算。



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。



4. 实验

4.1 准备数据

我只使用《鸳鸯刀》这本小说训练，首先进行预处理：

```
def get_single_corpus(file_path):
    rr = u'[a-zA-Z0-9’!"#$%&\'()*+,-./:: ;<=>?@★、…【】《》‘’[\\]^_`{|}~「」『』（）]+'  # # 保留“”，。？！等字符
    with open(file_path, 'r', encoding='ANSI') as f:
        corpus = f.read()
        corpus = re.sub(rr, '', 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("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《鸳鸯刀》小说的分词列表。

```
def get_dataset(data):  # data为分词结果
    max_len = 50
    step = 3
    sentences = []
    next_tokens = []

    tokens = list(set(data))
    tokens_indices = {token: tokens.index(token) for token in tokens}
    print('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50，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。

4.2 模型构建

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')
    ])
```

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。

```
def sample(preds, temperature=1.0):  # 预测的结果、温度
    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)
```

4.3 训练与测试

```
def train_and_test(x, y, tokens, tokens_indices, epochs=100):
    x = np.asarray(x)
    y = np.asarray(y)
    dataset = tf.data.Dataset.from_tensor_slices((x, y))
    dataset = dataset.shuffle(buffer_size=4096)
    dataset = dataset.batch(128)
    dataset = dataset.prefetch(buffer_size=tf.data.experimental.AUTOTUNE)

    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)  # 交叉熵损失

    for e in range(epochs):

        model.fit(dataset, epochs=1, callbacks=callbacks_list)
        text = '他夫妻俩越争越大声。萧中慧再也忍耐不住，「啊」的一声，掩面奔出。萧中慧心中茫然一片，只觉眼前黑蒙蒙的，了无生趣。'
        print(text, end='')
        if e % 10 == 0:
            for temperature in [0.2, 0.5, 1.0]:
                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]
```

每10个epoch对给定文本测试一次，分0.2，0.5，1.0三个温度进行测试。

5. 结果分析

《鸳鸯刀》文本：

```
他夫妻俩越争越大声。萧中慧再也忍耐不住，「啊」的一声，掩面奔出。萧中慧心中茫然一片，只觉眼前黑蒙蒙的，了无生趣。
```

训练20个epoch后，loss~=3，结果如下：

- 0.2

```
还是之还是铁鞭铁鞭铁鞭铁鞭这。对是？周威信，早已假向外两个今朝手中？也只是手中道？迳是路是谁人路今朝还两个假能今朝眼见只是周威信铁鞭这是？周威信。这是。还是的眼见铁鞭这是周威信。铁鞭这是？铁鞭眼见铁鞭铁鞭铁鞭铁鞭铁鞭铁鞭是铁鞭这是英雄？铁鞭路假今朝的还是眼见铁鞭这是周威信不谁谁今朝周威信。铁鞭这是这。还是
```

- 0.5

```
是周威信。铁鞭铁鞭铁鞭铁鞭这是今朝。还是眼见周威信。袁冠南这是是是是是是是是是是是是是周威信不今朝眼见周威信。铁鞭铁鞭铁鞭铁鞭。铁鞭铁鞭铁鞭铁鞭这是今朝。还是袁冠南江湖拼命铁鞭铁鞭铁鞭铁鞭铁鞭铁鞭铁鞭铁鞭铁鞭铁鞭这是？这是今朝在拼命铁鞭。人是这今朝？一齐气鼓鼓一齐铁鞭铁鞭铁鞭铁鞭铁鞭铁鞭这是？，是在背上周威信也英雄颈是区区口中
```

- 1.0

```
铁鞭铁鞭这。正是铁鞭铁鞭铁鞭这是今朝。。还是袁冠南铁鞭铁鞭这是是英雄这路还是手中今朝手中的铁鞭铁鞭这是正是正是铁鞭铁鞭铁鞭这是今朝的？早已不两个谁谁谁谁今朝正是铁鞭江湖铁鞭这。还是今朝周威信的。还是今朝左腿铁鞭这是周威信？。人今朝？是我们在背上假警卫铁鞭这是今朝周威信的？。还是今朝左腿铁鞭这是周威信。铁鞭铁鞭铁鞭这是
```

生成的文字略微有金庸先生的写作风格，但整体效果很差，没有明确的语义，而且还会出现”是是是是是是是是是是是是是“这样的现象。标点符号的使用也是一个问题，可以看出“？”之后会出现“，”与“。”，但是比较通常的标点符号用法是能够学习到的。

6. Huggingface Transformers

既然我们的代码效果很差，就想试一下基于transformer的编解码架构的大规模训练的文本生成模型效果如何，我们选择在huggingface开源的预训练模型（支持中文），且不在金庸小说中再次训练微调，此模型的网址如下：

IDEA-CCNL/Wenzhong2.0-GPT2-3.5B-chinese · Hugging Face

```
IDEA-CCNL/Wenzhong2.0-GPT2-3.5B-chinese
```

既可以在网页端在线测试（比较慢），也可以在IDE上下载预训练模型进行测试（模型较大）。

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel
tokenizer = GPT2Tokenizer.from_pretrained('IDEA-CCNL/Wenzhong2.0-GPT2-3.5B-chinese')
model = GPT2LMHeadModel.from_pretrained('IDEA-CCNL/Wenzhong2.0-GPT2-3.5B-chinese')
text = "他夫妻俩越争越大声。萧中慧再也忍耐不住，「啊」的一声，掩面奔出。萧中慧心中茫然一片，只觉眼前黑蒙蒙的，了无生趣。"
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input)
from transformers import pipeline, set_seed
set_seed(55)
generator = pipeline('text-generation', model='IDEA-CCNL/Wenzhong2.0-GPT2-3.5B-chinese')
generator("他夫妻俩越争越大声。萧中慧再也忍耐不住，「啊」的一声，掩面奔出。萧中慧心中茫然一片，只觉眼前黑蒙蒙的，了无生趣。", max_length=30, num_return_sequences=1)
```

对于同样的文本，我们可以看到在此预训练模型上生成的语句具有鲜明的语义信息，标点符号使用正确，甚至学习到了「啊」这种标点语气词的用法。

```
他夫妻俩越争越大声。萧中慧再也忍耐不住，「啊」的一声，掩面奔出。萧中慧心中茫然一片，只觉眼前黑蒙蒙的，了无生趣。她翻个身，拉起裙子就往窗户里爬去。 「爸爸...」萧中慧想道：「我都还要跑回去...要是哥哥在旁边，肯定会加以制止的...」不料，竟是一个细微之处，一瞬间，一个白影消失在自己身后，没有留下任何痕迹，显得宛如梦境一般。她心头轰轰作响。 杜庭延再也抑制不
```