一、问题描述

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。

二、实验原理

1. Seq2Seq 模型

Seq2Seq 全称为：sequence to sequence，是 2014 年被提出来的一种 Encoder-Decoder 结构。其中 Encoder 是一个 RNN 结构（LSTM、GRU、RNN 等）。

Seq2Seq 的主要思想是输入一个序列，通过 encoder 编码成一个语义向量 c（context），然后 decoder 成输出序列。这个结构重要的地方在于输入序列和输出序列的长度是可变的。

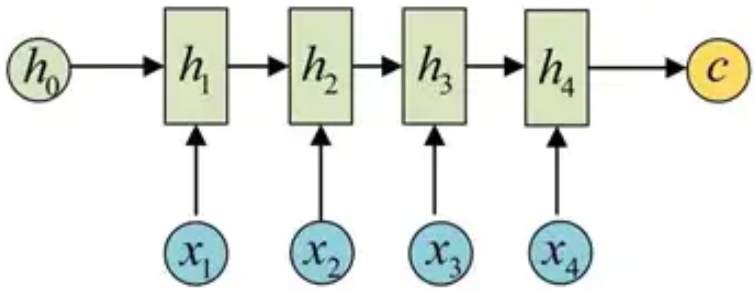
Seq2Seq 模型是输出的长度不确定时采用的模型，这种情况一般是在机器翻译的任务中出现，将一句中文翻译成英文，那么这句英文的长度有可能会比中文短，也有可能会比中文长，所以输出的长度就不确定了。如下图所，输入的中文长度为 4，输出的英文长度为 2。

在网络结构中，输入一个中文序列，然后输出它对应的中文翻译，输出的部分的结果预测后面，根据上面的例子，也就是先输出“machine”，将“machine”作为下一次的输入，接着输出“learning”，这样就能输出任意长的序列。

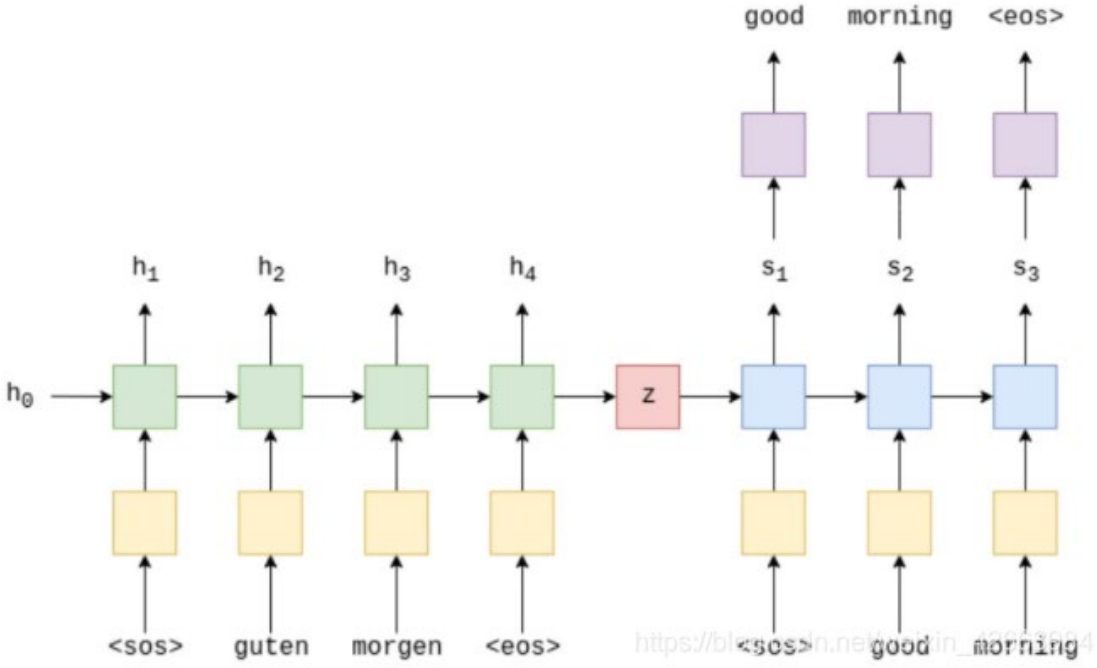
机器翻译、人机对话、聊天机器人等等，这些都是应用在当今社会都或多或少的运用到了我们这里所说的 Seq2Seq。

1.1 Seq2Seq 结构

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，如下图，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。



首先将源语句输入至 encoder 编码为一个向量，我们称为上下文向量，它可以视为整个输入句子的抽象表示。然后，该向量由第二个 LSTM 解码，该 LSTM 通过一次生成一个单词来学习输出目标语句。下面给出文本翻译的例子。



源语句被输入至 embedding 层（黄色），然后被输入编码器（绿色），我们还分别将序列的开始和序列的结束标记附加到句子的开始和结尾，sos 为 start of sentence，eos 为 end of sentence。在每一个时间步，我们输入给 encoder 当前的单词以及上一个时间步的隐藏状态h_{t-1}，encoder 吐出新的h_t，这个 tensor 可以视为目前为止的句子的抽象表示。这个 RNN 可以表示为一个方程：

$$h_t = EncoderRNN(emb(x_t), h_{t-1})$$

这里的 RNN 可以是 LSTM 或 GRU 或任何 RNN 的变体。在最后一个时间步，我们将h_t赋给 z，作为 decoder 的输入。

在每个时间步，解码器 RNN（蓝色）的输入是当前单词的嵌入，以及上一个时间步的隐藏状态，其中初始解码器隐藏状态就是上下文向量，即初始解码器

隐藏状态是最终编码器隐藏状态。因此，方程：

$$s_t = DecoderRNN(emb(y_t), x_{t-1})$$

然后在每一个时间步，我们将s_t输入给线形层（紫色），得到$\hat{y}_t$，即$\hat{y}_t = f(s_t)$而后用$\hat{y}_t$与y进行交叉熵计算，得到损失，并优化参数。

1.2 Encoder(编码器)

在前向计算中，我们传入源语句，并使用嵌入层将其转换为密集向量，然后应用 dropout。然后将这些嵌入传递到 RNN。当我们将整个序列传递给 RNN 时，它将为我们自动对整个序列进行隐藏状态的递归计算。请注意，我们没有将初始的隐藏状态或单元状态传递给 RNN。

1.3 Decoder(解码器)

解码器执行解码的单个步骤，即，每个时间步输出单个 token。第一层将从上一时间步中接收到一个隐藏的单元格状态，并将其与当前嵌入的 token 一起通过 LSTM 馈送，以产生一个新的隐藏的单元格状态。后续层将使用下一层的隐藏状态，以及其图层中先前的隐藏状态和单元格状态。

Decoder 的参数与 encoder 类似，但要注意它们的 hid_dim 要相同，否则矩阵无法运算。

1.4 模型应用领域

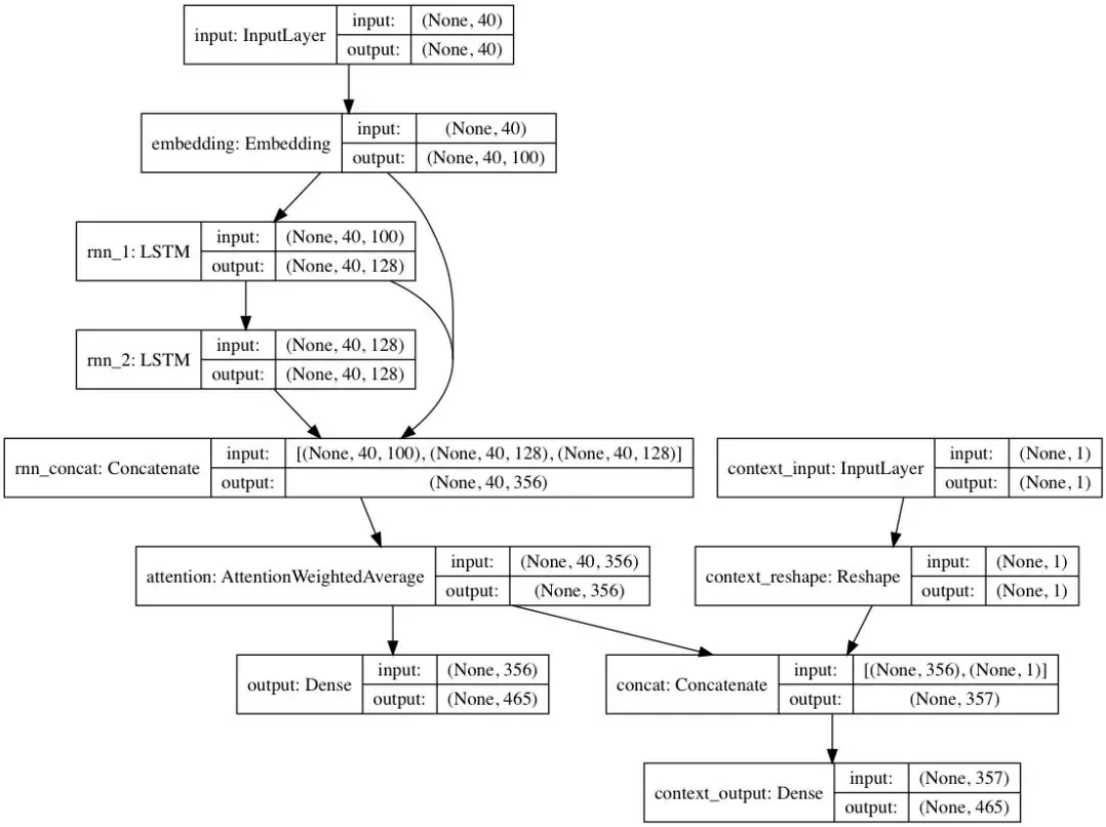
首先作为为机器翻译问题为出发点提出来的 Seq2Seq 模型，机器翻译的准确率因为该模型的提出而有了较大的提升。

作为 Seq2Seq 模型研发团队，Google Brain 团队在 2014 年的文章的应用案例中对 LSTM 的隐藏结点做了主成分分析，如下图所示，从图中可以看出，模型中的语境向量很明显包涵了输入序列的语言意义，后沟将不同次序所产的不同意思的语句划分开，这对于提升机器翻译的准确率很有帮助。其次 Seq2Seq 模型因为突破了传统的固定大小输入问题框架，因而除了翻译场景，还被用于智能对话与问答的实现以及微博的自动回复，2015 年华为团队，通过 Seq2Seq 为基础设计的模型实现了计算机对微博的自动回复，并通过模型间的对比得到了一系列有意思的结果。如下图，post 为微博主发的文，其余四列为不同模型对该条微博做出的回复。

ID	Post	NRM-glo	NRM-loc	NRM-hyb	Rtr.-based
1	每年元旦都来高烧。 High fever attacks me every New Year's Day	多休息，增强免疫力！ Rest more to boost your immune system.	每年元旦都去医院看病 Go to see a doctor every New Year's Day	祝早日康复，身体健康！ Get well soon and stay healthy!	高烧… High fever…
2	又老了一岁，感谢我们组，我很高兴 I gain one more year. Grateful to my group, so happy	生日快乐！永远支持你！ Happy birthday! Will stand by you forever!	生日快乐！永远年轻漂亮！ Happy birthday! Wish you always stay young and pretty!	老了老了，岁月不饶人啊！ Getting old now. Time has no mercy.	我只是老了一岁 I just gain only one more year.
3	吃素第一天，坚持住，崔朵拉。 First day of being a vegetarian. Hold on, **CuiDuoLa**	必须来支持！加油！ I come to offer my support. Keep it up!	坚持住，坚持下去。 Hold on, keep it up.	每天都是素食主义者 I have been being a vegetarian everyday	坚持住！ Hold on!
4	这种天气真应该带个帅锅出去踏青赏花 We should go out with some cute guys to enjoy a great outing in such a nice weather.	这是哪儿啊？真美！ Where is it? It is so beautiful!	阳光明媚，心情舒畅。 Such a nice sunny day! I am in a great mood.	这种天气真不是盖的。 It is indeed a very nice weather.	文山啊出去踏青寻找灵感哈哈 **WenShan**, let's go out to get some inspiration. Ha! Ha!

2. Textgenrnn

文本生成是一件很神奇的自然语言处理任务，深度学习给文本生成带来的全新的技术途径，是一种不可思议又高效的方式。textgenrnn 就是采用 RNN 的方式来实现文本生成的一个简洁高效的库，代码量非常少，又非常易于理解。其架构是采用了 LSTM+Attention 的方式来实现。如下图所示：



textgenrnn 接受最多 40 个字符的输入，首先每个字符转换为 100 维的词(char)

向量，并将这些向量输入到一个包含 128 个神经元的长短期记忆（LSTM）循环层中。其次，这些输出被传输至另一个包含 128 个神经元的 LSTM 中。以上所有三层都被输入到一个注意力层中，用来给最重要的时序特征赋权，并且将它们取平均（由于嵌入层和第一个 LSTM 层是通过跳跃连接与注意力层相连的，因此模型的更新可以更容易地向后传播并且防止梯度消失）。该输出被映射到最多 394 个不同字符的概率分布上，这些字符是序列中的下一个字符，包括大写字母、小写字母、标点符号和表情。而且关键是上述的参数都可以设置。

三、实验过程与结果分析

1. 实验过程

安装 TensorFlow，调用 textgenrnn 完成实验。数据集选用三体部分章节。

2. 实验结果与分析

在 textgenrnn 的参数中，可见有个 temperatures，它可以用来代表生成文本的温度。（从结果来看，似乎可以认定为文本带的感情色彩强烈与否，其中 0.2 一般为偏负面，0.5 代表偏中性，1.0 代表相对正能量一些。）50 个 epochs 后的训练结果如下：

```
5730/5730 [==============================] - 73s 13ms/step - loss: 0.1062
####################
Temperature: 0.2
####################
```

“我说过，我和你们没关系了，今后请不要跟踪我！”

“你到底知道些什么？告诉我！”

“你到底知道些什么？告诉我！”

```
####################
Temperature: 0.5
####################
```

“他们好像也做过一些调查。”

“你们……”汪淼想知道杨冬生活中的一切，但又不知该如何问。

“我说过，我和你们没关系了，今后请不要跟踪我！”

```
####################
Temperature: 1.0
####################
```

“刚才您说的那些，与军方有什么关系？”

汪淼旁边的，那是一位著名学者的不专家，对汪淼说出了他，有些相机之中的一行。汪淼以前从未达到过来，在想象过的将军。作为某种超出人类理解力的力量代言的女人，冷酷地封死了汪淼的一切出路。

“不吱声没人拿你当哑巴！”旁边一位警官探过身去对大史低声说，后者拿起桌上的茶杯，看到里面的烟头后，“咚”的一声又放下了。

可见整体训练效果较好。但 temperature 较低时，生成的文本较为简单、机械，存在重复；temperature 较高时，生成的文本较混乱、复杂，部分地方会有违和感；temperature 居中时呈现的效果最好，因此该函数中 temperature 默认值也设置为 0.5。

四、参考资料

[1] 非常简单操作的 Tensorflow 安装（适用于最新版）

https://www.bilibili.com/read/cv16248473

[2] 【解决】ImportError: cannot import name ‘multi_gpu_model’ from ‘tensorflow.keras.utils’

https://blog.csdn.net/zoey_peak/article/details/122494982

[3] 从 keras.backend.tensorflow_backend 导入 set_session

https://www.pythonheidong.com/blog/article/506442/0491a78d43077dcd3ef8/

[4] 仅用四行代码实现 RNN 文本生成模型

https://blog.csdn.net/weixin_40746796/article/details/91410913

[5] Easily train your own text-generating neural network of any size and complexity on any text dataset with a few lines of code.

https://github.com/minimaxir/textgenrnn

[6] textgenrnn 文本生成实战

https://blog.csdn.net/sparkexpert/article/details/80136357