textsum\_model.py

**def document\_reader(self):**

1. **self.input\_x :** [batch\_size, max\_num\_sequence, sentence\_length]

batch\_size篇文档、一篇文档的句子的最大数量、一个句子的单词的最大数量

这个矩阵中存储都是一篇文档的每个单词的id，而不是单词的embedding。

1. **self.embedded\_words :** [max\_num\_sequence, sentence\_length, embed\_size]

一篇文档的句子的最大数量、一个句子的单词的最大数量、word embedding的维度

这个变量相比self.input\_x进一步细分到了一篇文档。这个变量中存储的是一篇文档的每个单词的word embedding。

1. **self.embedded\_words\_expanded** : [max\_num\_sequence, sentence\_length, embed\_size, 1]

一篇文档的句子的最大数量、一个句子的单词的最大数量、word embedding的维度、最后一个维度是1的原因是因为**self.embedded\_words\_expanded**是作为卷积函数**tf.nn.conv2d的input参数，卷积函数的input参数要求是4维，最后一个维度是in\_channels，对于一张图片来说就是通道数，对于文本来说那就是1个维度了。**

1. **filter：**[filter\_size, self.embed\_size, 1, self.feature\_map[i]]

filter的宽、高、输入通道数、输出通道数

1. **conv:** [max\_num\_sequence, sequence\_length-filter\_size+1, 1, num\_filters]

conv=tf.nn.conv2d(conv\_s, filter, strides=[1,1,1,1], padding="VALID",name="conv")

一篇文档的句子的最大数量、feature map的高度、宽度、输出通道数。

注意：第三个维度是1是因为**self.embedded\_words\_expanded**的第三个维度和filter的第二个维度是相等的，做卷积之后结果为1，第四个维度num\_filters和**filter**中的最后一个维度是相等的。

**Conv**就相当于是存储量max\_num\_sequence个句子的feature maps，一个句子有num\_filters个feature map，feature map的维度是[sequence\_length-filter\_size+1, 1]

1. **pooled:** [max\_num\_sequence, 1, 1, num\_filters]

pooled=tf.nn.max\_pool(h, ksize=[1,self.sequence\_length-filter\_size+1,1,1], strides=[1,1,1,1], padding='VALID',name="pool")

pooled代码如图所示，其中输入变量h的维度和conv的维度是一样的，所以就是在conv的第二个维度上面做pooling操作。而且pooling核的大小是self.sequence\_length-filter\_size+1，故pooling之后的结果是其他维度不变，第二个维度变为1。就相当于是取conv的第二维中最大的值。

**Pooled**相当于一个句子的embedding的维度是num\_filtes

1. **pooled\_temp:** [max\_num\_sequence, num\_filters \* class\_filters]

**pooled\_temp**是把pooled四维矩阵转化为2维矩阵。一个句子的embedding也就是num\_filters\*class\_filters。

num\_filters：是一个大小为3的卷积核的out\_channels的个数，比如一个大小为3的卷积核的out\_channels可能为[20,20,30,40,50,70,70]其中之一**。**

Class\_filters：是卷积核的个数，比如论文中用了[1,2,3,4,5,6,7]7个卷积核。

1. **cnn\_outputs:** [batch\_size, max\_num\_sequence, num\_filters \* class\_filters]

**cnn\_outputs**中存储的就是就是batch\_size篇文章的sentence embedding。

1. **lstm\_outputs:** [batch\_size, max\_time, hidden\_size]

lstm\_outputs, cell\_state = tf.nn.dynamic\_rnn(lstm\_cell, cnn\_outputs, dtype = tf.float32)

其中hidden\_size=num\_filters\*class\_filters，这是作者定义的。但其实不相等也可以。其中hidden\_size表示LSTM中隐藏层神经元的个数，max\_time与cnn\_outputs中的max\_num\_sequence是相等的（这时max\_time理解为一篇文章的最大句子数，每一个时间步的输入都是一个sentence embedding）。**lstm\_outputs**中存储的是batch\_size篇文章的embedding。

1. **cell\_state:** [batch\_size, hidden\_size]

**cell\_state**中存储的是outputs中最后一个时间步的ct和ht，ct和ht的维度都是[batch\_size, hidden\_size]

**def sentence\_extractor(self):**

1. h\_t: decoder LSTM output
2. At: encoder LSTM output (document level)
3. St: encoder CNN output (sentence level)
4. probability value: [p\_t = activation(MLP(h\_t:At)) for h\_t in h\_t\_steps ]