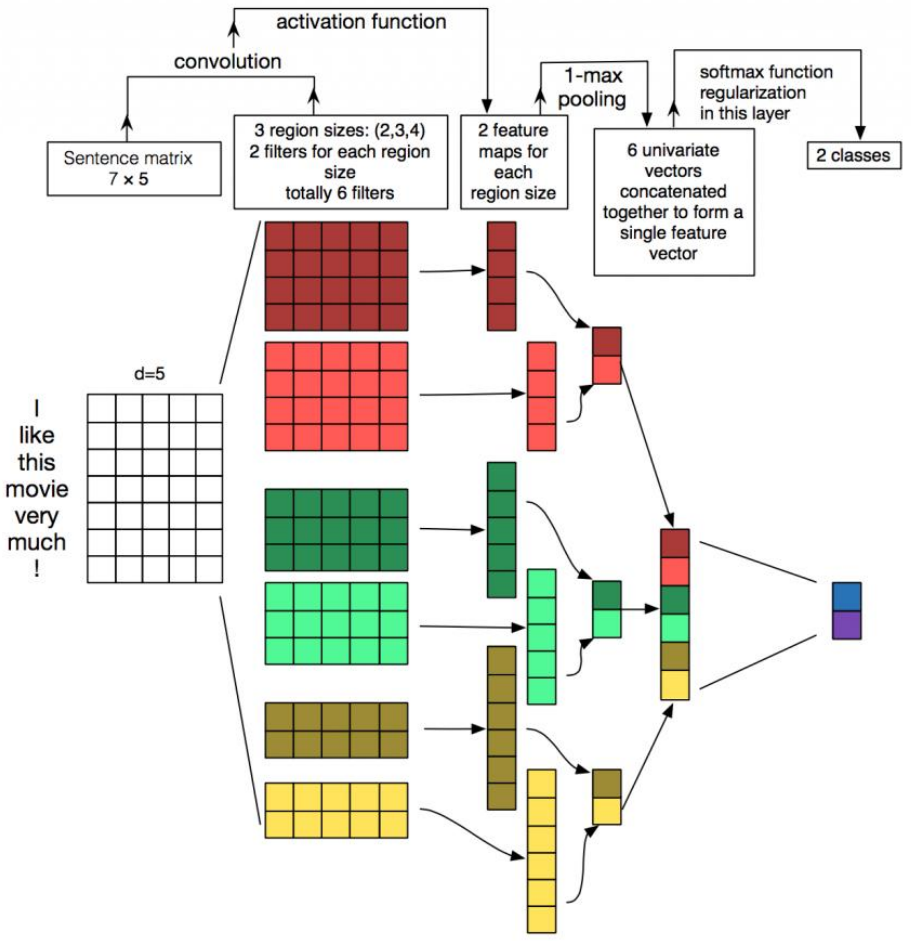
使用 CNN 构建文本语义：
（博客链接：）
英文：
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
中文翻译：
http://www.csdn.net/article/2015-11-11/2826192
（图例说明：）



卷积神经网络作用于 NLP？

NLP 任务的输入不再是像素点了，大多数情况下是以矩阵表示的句子或者文档。矩阵的每一行对应于一个分词元素，一般是一个单词，也可以是一个字符。也就是说每一行是表示一个单词的向量。通常，这些向量都是 word embeddings（一种底维度表示）的形式，如 word2vec 和 GloVe，但是也可以用 one-hot 向量的形式，也即根据词在词表中的索引。若是用 100 维的词向量表示一句 10 个单词的句子，我们将得到一个 10x100 维的矩阵作为输入。这个矩阵相当于是一幅“图像”。

在计算机视觉的例子里，我们的滤波器每次只对图像的一小块区域运算，但在处理自然语言时滤波器通常覆盖上下几行（几个词）。因此，滤波器的宽度也就和输入矩阵的宽度相等了。尽管高度，或者区域大小可以随意调整，但一般滑动窗口的覆盖范围是 2~5 行。综上所述，

处理自然语言的卷积神经网络结构是这样的（花几分钟时间理解这张图片，以及维度是如何变化的。你可以先暂时忽略池化操作，我们在稍后会解释它）：

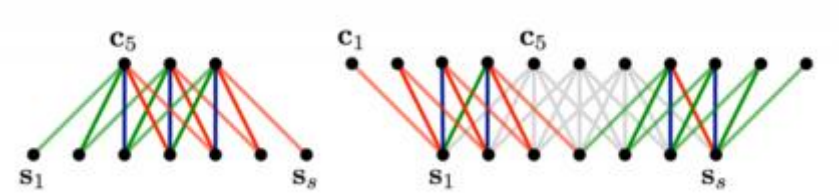
这里我们对滤波器设置了三种尺寸：2、3 和 4 行，每种尺寸各有两种滤波器。每个滤波器对句子矩阵做卷积运算，得到（不同程度的）特征字典。然后对每个特征字典做最大值池化，也就是只记录每个特征字典的最大值。这样，就由六个字典生成了一串单变量特征向量（univariate feature vector），然后这六个特征拼接形成一个特征向量，传给网络的倒数第二层。最后的 softmax 层以这个特征向量作为输入，用其来对句子做分类；我们假设这里是二分类问题，因此得到两个可能的输出状态。

CNN 的超参数

在解释如何将 CNNs 用于 NLP 任务之前，先来看一下构建 CNN 网络时需要面临的几个选择。希望这能帮助你更好地理解相关文献。

窄卷积 vs 宽卷积

在上文中解释卷积运算的时候，我忽略了如何使用滤波器的一个小细节。在矩阵的中部使用 3x3 的滤波器没有问题，在矩阵的边缘该怎么办呢？左上角的元素没有顶部和左侧相邻的元素，该如何滤波呢？解决的办法是采用补零法（zero-padding）。所有落在矩阵范围之外的元素值都默认为 0。这样就可以对输入矩阵的每一个元素做滤波了，输出一个同样大小或是更大的矩阵。补零法又被称为是宽卷积，不使用补零的方法则被称为窄卷积。1D 的例子如图所示：

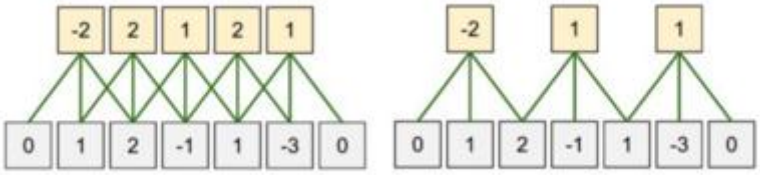


窄卷积 vs 宽卷积。滤波器长度为 5，输入长度为 7。来源：
A Convolutional Neural Network for Modelling Sentences (2014)

当滤波器长度相对输入向量的长度较大时，你会发现宽卷积很有用，或者说很有必要。在上图中，窄卷积输出的长度是 (7-5)+1=3，宽卷积输出的长度是(7+2*4-5)+1=11。一般形式为$n_{out} = (n_{in} + 2 * n_{padding} - n_{filter}) + 1$

步长

卷积运算的另一个超参数是步长，即每一次滤波器平移的距离。上面所有例子中的步长都是1，相邻两个滤波器有重叠。步长越大，则用到的滤波器越少，输出的值也越少。下图来自斯坦福的 cs231 课程网页，分别是步长为 1 和 2 的情况：

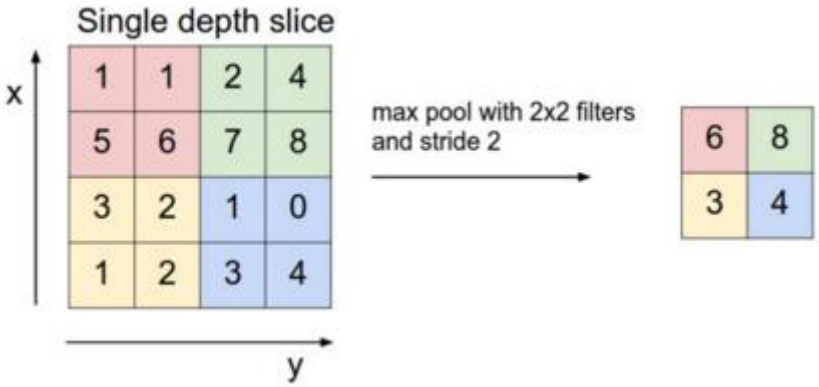


卷积步长。左侧：步长为 1，右侧：步长为 2。来源： http://cs231n.github.io/convolutional-networks/

在文献中我们常常见到的步长是 1，但选择更大的步长会让模型更接近于递归神经网络，其结构就像是一棵树。

池化层

卷积神经网络的一个重要概念就是池化层，一般是在卷积层之后。池化层对输入做降采样。常用的池化做法是对每个滤波器的输出求最大值。我们并不需要对整个矩阵都做池化，可以只对某个窗口区间做池化。例如，下图所示的是 2x2 窗口的最大值池化（在 NLP 里，我们通常对整个输出做池化，每个滤波器只有一个输出值）：



CNN 的最大池化。

来源： http://cs231n.github.io/convolutional-networks/#pool

为什么要池化呢？有许多原因。

池化的特点之一就是它输出一个固定大小的矩阵，这对分类问题很有必要。例如，如果你用了 1000 个滤波器，并对每个输出使用最大池化，那么无论滤波器的尺寸是多大，也无论输入数据的维度如何变化，你都将得到一个 1000 维的输出。这让你可以应用不同长度的句子和不同大小的滤波器，但总是得到一个相同维度的输出结果，传入下一层的分类器。

池化还能降低输出结果的维度，（理想情况下）却能保留显著的特征。你可以认为每个滤波器都是检测一种特定的特征，例如，检测句子是否包含诸如“not amazing”等否定意思。如果这个短语在句子中的某个位置出现，那么对应位置的滤波器的输出值将会非常大，而在其它位置的输出值非常小。通过采用取最大值的方式，能将某个特征是否出现在句子中的信息保留下来，但是无法确定它究竟在句子的哪个位置出现。这个信息出现的位置真的很重要吗？确实是的，它有点类似于一组 n-grams 模型的行为。尽管丢失了关于位置的全局信息（在句子中的大致位置），但是滤波器捕捉到的局部信息却被保留下来了，比如“not amazing”和“amazing not”的意思就大相径庭。

在图像识别领域，池化还能提供平移和旋转不变性。若对某个区域做了池化，即使图像平移/旋转几个像素，得到的输出值也基本一样，因为每次最大值运算得到的结果总是一样的。

通道

我们需要了解的最后一个概念是通道。通道即是输入数据的不同“视角”。比如说，做图像识别时一般会用到 RGB 通道（红绿蓝）。你可以对每个通道做卷积运算，赋予相同或不同的权值。你也同样可以把 NLP 想象成有许多个通道：把不同类的词向量表征（例如 word2vec 和 GloVe）看做是独立的通道，或是把不同语言版本的同一句话看作是一个通道。