文本分类问题不需要ResNet？小夕解析DPCNN设计原理（下）

原创 夕小瑶 夕小瑶的卖萌屋 2018-04-07

来自专辑

卖萌屋@自然语言处理

> 哎呀呀，说好的不拖稿的又拖了两天T_T，小夕过一阵子分享给你们这两天的开心事哦。后台催稿调参系列的小伙伴们不要急，下一篇就是第二篇调参文啦。

好啦，接着上一篇文章，直接搬来DPCNN、ShallowCNN、ResNet的对比图。

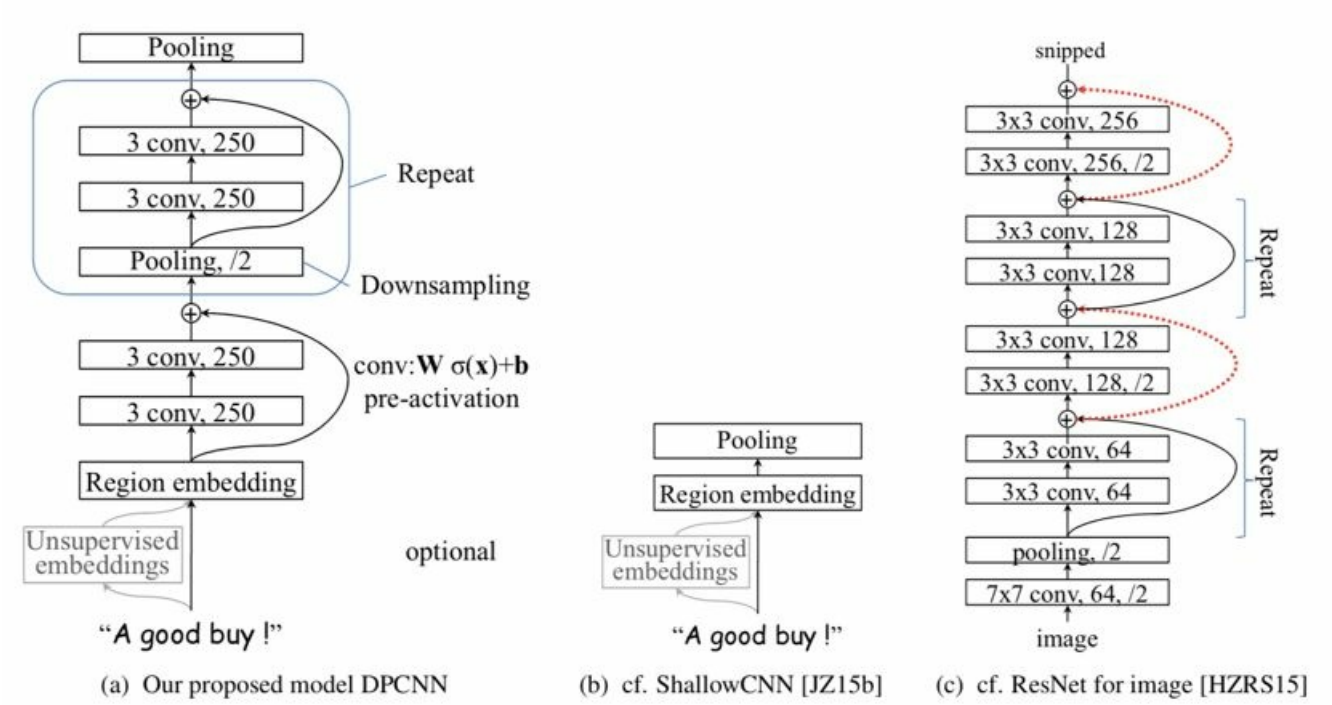


Figure 1: (a) Our proposed model DPCNN. (b,c) Previous models for comparison. ⊕ indicates addition. The dotted red shortcuts in (c) perform dimension matching. DPCNN is dimension-matching free.

从图中的a和c的对比可以看出，DPCNN与ResNet差异还是蛮大的。同时DPCNN的底层貌似保持了跟TextCNN一样的结构，这里作者将TextCNN的包含多尺寸卷积滤波器的卷积层的卷积结果称之为Region embedding，意思就是对一个文本区域/片段（比如3gram）进行一组卷积操作后生成的embedding。

对一个3gram进行卷积操作时可以有两种选择，一种是保留词序，也就是设置一组size=3*D的二维卷积核对3gram进行卷积(其中D是word embedding维度)； 还有一种是不保留词序（即使用词袋模型），即首先对3gram中的3个词的embedding取均值得到一个size=D的向量，然后设置一组size=D的一维卷积核对该3gram进行卷积。显然TextCNN里使用的是保留词序的做法，而DPCNN使用的是词袋模型的做法，DPCNN作者argue前者做法更容易造成过拟合，后者的性能却跟前者差不多(其实这个跟DAN网络(Deep averaging networks)中argue的原理和结论差不多，有兴趣的可以下拉到下一部分的知乎传送门中了解一下）。

产生region embedding后，按照经典的TextCNN的做法的话，就是从每个特征 图中挑选出最有代表性的特征，也就

是直接应用全局最大池化层（max-over-time-pooling layer），这样就生成了这段文本的特征向量（假如卷积滤波器的size有3，4，5这三种，每种size包含100个卷积核，那么当然就会产生3*100幅特征图，然后将max-over-time-pooling操作应用到每个特征图上，于是文本的特征向量即3*100=300维）。

但是显然TextCNN这样做会有很严重的问题诶，这样做的意义本质上与词袋模型（含ngram）+weighting+NB/MaxEnt/SVM的经典文本分类模型没本质区别，只不过one-hot表示到word embedding表示的转变避免了词袋模型遭遇的数据稀疏问题罢了。可以说，TextCNN本质上收益于词向量的引入带来的“近义词有相近向量表示”的bonus，同时TextCNN恰好可以较好的利用词向量中的知识（近义关系）罢了。这意味着，经典模型里难以学习的远距离信息（如12gram）在TextCNN中依然难以学习。**那么这些长距离复杂模式如何让网络学习到呢？**

显然，要么加深全连接层，要么加深卷积层。加深哪个更好呢？小夕埋下了一个伏笔哦，答案就在小夕这个知乎回答里：

传送门： https://www.zhihu.com/question/270245936

在得到Region embedding后，为了避免后续想象太抽象，我们不妨还是把Region embedding看成word embedding，假想为交给网络后面的就是word embedding序列哦。

首先交代一下卷积的一个基本概念——等长卷积。我们在文本分类里最常用的可能是窄卷积，输入序列长度为seq_len，卷积核大小为n的话，窄卷积后的输出序列的长度就是seq_len-n+1。而等长卷积顾名思义就是输出序列的长度等于输入序列长度seq_len。没有想像出来的同学自行Google一下哦，就不展开讲啦。

那么对文本，或者说对word embedding序列进行等长卷积的意义是什么呢？

既然输入输出序列的位置数一样多，我们将输入输出序列的第n个embedding称为第n个词位，那么这时size为n的卷积核产生的等长卷积的意义就很明显了，那就是**将输入序列的每个词位及其左右((n-1)/2)个词的上下文信息压缩为该词位的embedding**，也就是说，产生了每个词位的被上下文信息修饰过的更高level更加准确的语义。

好，回到DPCNN上来。我们想要克服TextCNN的缺点，捕获长距离模式，显然就要用到深层CNN啦。那么直接等长卷积堆等长卷积可不可以呢？

显然这样会让每个词位包含进去越来越多，越来越长的上下文信息，但是这样效率也太低了喂，显然会让网络层数变得非常非常非常深，这样笨拙的操作怎么能有呢哼。不过，既然等长卷积堆等长卷积会让每个词位的embedding描述语义描述的更加丰富准确，那么**当然我们可以适当的堆两层来提高词位embedding的表示的丰富性**。

所以region embedding层（这里假想为word embedding层，对应序列为 “小娟 姐姐 带来 的 抹茶 青团 好 好吃 哦”）之上就可以如图2这样设计啦：

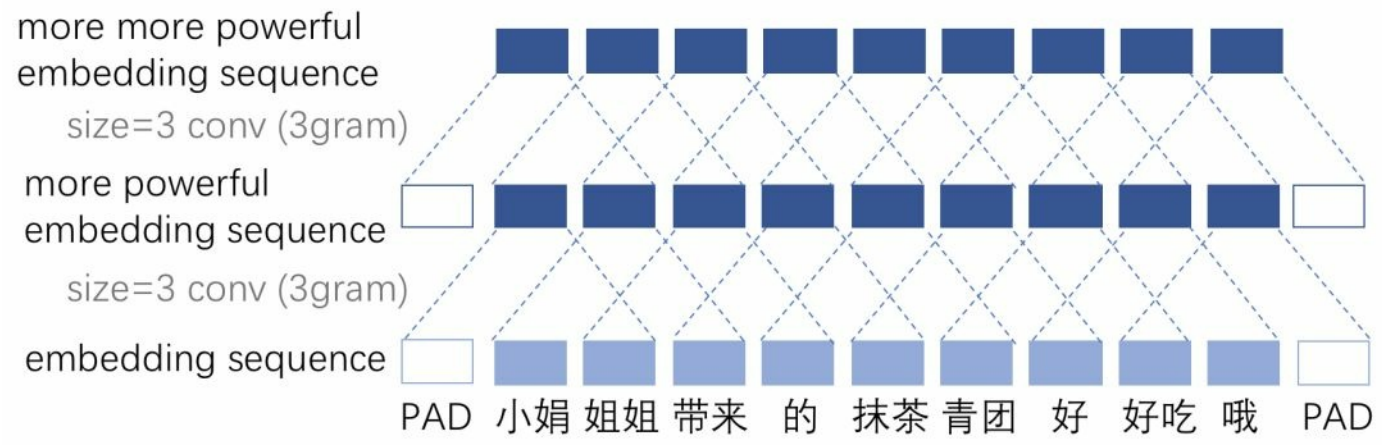


图2 (转载该图请后台告知小夕噢)

在表示好每个词位的语义后，其实很多邻接词或者邻接ngram的词义是可以合并的，例如“小娟 姐姐 人 不要 太好”中的“不要”和“太好”虽然语义本来离得很远，但是作为邻接词“不要太好”出现时其语义基本等价为“很好”，这样完全可以把“不要”和“太好”的语义进行合并哇。同时，合并的过程完全可以在原始的embedding space中进行的，毕竟原文中直接把“不要太好”合并为“很好”是很可以的哇，**完全没有必要动整个语义空间**。

而实际上，相比图像中这种从“点、线、弧”这种low-level特征到“眼睛、鼻子、嘴”这种high-level特征的明显层次性的特征区分，文本中的特征进阶明显要扁平的多，即从单词（1gram）到短语再到3gram、4gram的升级，其实很大程度上均满足“语义取代”的特性。而图像中就很难发生这种”语义取代“现象（例如“鼻子”的语义可以被”弧线“的语义取代嘛？）。

因此（划重点），DPCNN与ResNet很大一个不同就是，在DPCNN中固定死了feature map的数量，也就是固定住了embedding space的维度（为了方便理解，以下简称语义空间），使得网络有可能让整个邻接词（邻接ngram）的合并操作在原始空间或者与原始空间相似的空间中进行（当然，网络在实际中会不会这样做是不一定的哦，只是提供了这么一种条件）。也就是说，整个网络虽然形状上来看是深层的，但是从语义空间上来看完全可以是扁平的。而ResNet则是不断的改变语义空间，使得图像的语义随着网络层的加深也不断的跳向更高level的语义空间。

好啦，所以提供了这么好的合并条件后，我们就可以用pooling layer进行合并啦。每经过一个size=2, stride=2（大小为2，步长为2）的池化层（以下简称1/2池化层），序列的长度就被压缩成了原来的一半（请自行脑补）。这样同样是size=3的卷积核，每经过一个1/2池化层后，其能感知到的文本片段就比之前长了一倍。

例如之前是只能感知3个词位长度的信息，经过1/2池化层后就能感知6个词位长度的信息啦，这时把1/2池化层和size=3的卷积层组合起来如图3所示。

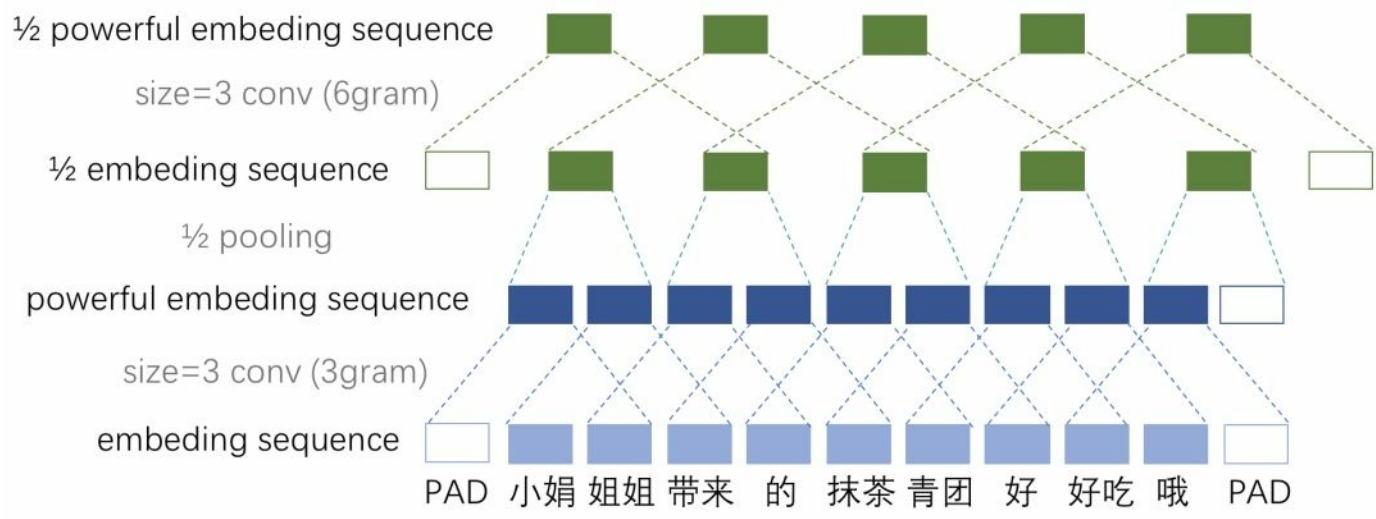


图3 (转载该图请后台告知小夕噢)

好啦，看似问题都解决了，目标成功达成。剩下的我们就只需要重复的进行等长卷积+等长卷积+1/2池化就可以啦，也就是重复如图4的Block：

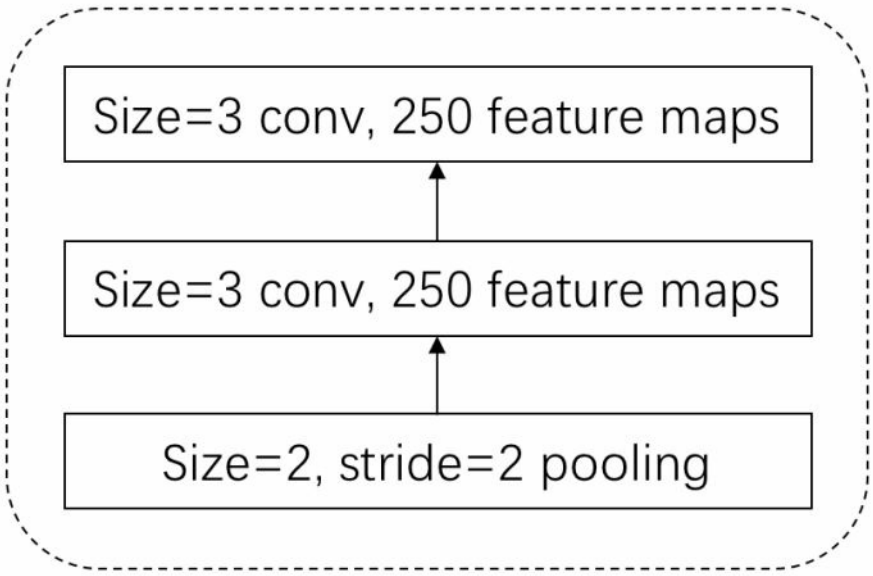


图4

但是！如果问题真的这么简单的话，深度学习就一下子少了超级多的难点了。

首先，由于我们在初始化深度CNN时，往往各层权重都是初始化为一个很小的值，这就导致最开始的网络中，后续几乎每层的输入都是接近0，这时网络的输出自然是没意义的，而这些小权重同时也阻碍了梯度的传播，使得网络的初始训练阶段往往要迭代好久才能启动。

同时，就算网络启动完成，由于深度网络中仿射矩阵（每两层间的连接边）近似连乘，训练过程中网络也非常容易发生梯度爆炸或弥散问题（虽然由于非共享权重，深度CNN网络比RNN网络要好点）。

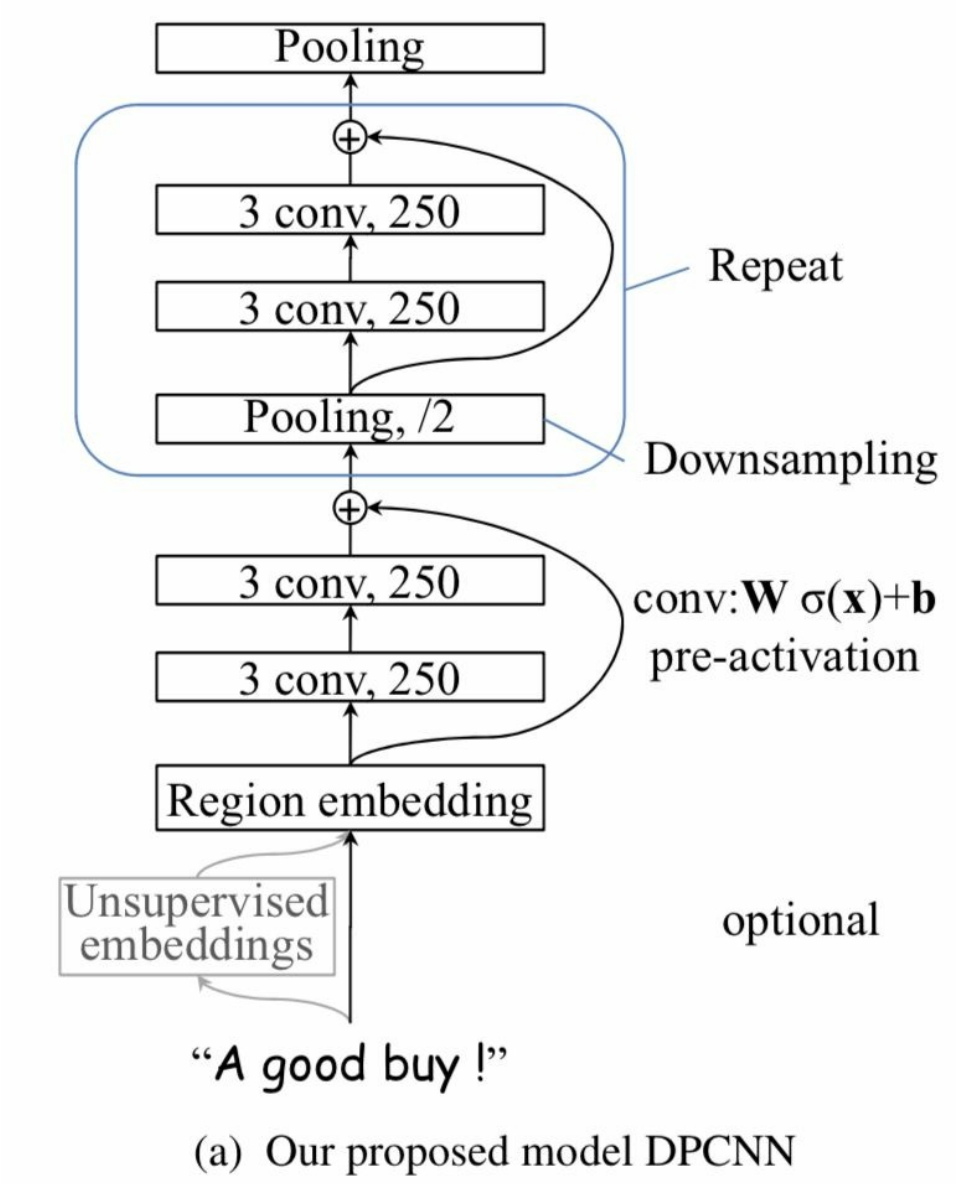
当然，上述这两点问题本质就是梯度弥散问题。那么如何解决深度CNN网络的梯度弥散问题呢？当然是膜一下何恺明大神，然后把ResNet，DenseNet的精华拿来用啦～DPCNN里用的是ResNet的方案。

ResNet中提出的shortcut connection (skip-connection)就是一种非常简单、合理、有效的解决方案。看着图4想一下，**既然每个block的输入在初始阶段容易是0而无法激活，那么直接用一条线把region embedding层连接到每个block的输入乃至最终的池化层/输出层不就可以啦！**

想象一下，这时的shortcut connection由于连接到了各个block的输入（当然为了匹配输入维度，要事先经过对应次数的1/2池化操作），这时就相当于一个短路连接，即region embedding直接短路连接到了最终的池化层或输出层。等等，这时的DPCNN不就退化成了TextCNN嘛。深度网络不好训练，就一层的TextCNN可是异常容易训练的。这样模型的起步阶段就是从TextCNN起步了，自然不会遭遇前面说的深度CNN网络的冷启动问题了。

同样的道理，有了shortcut后，梯度就可以忽略卷积层权重的削弱，从shortcut一路无损的传递到各个block手里，直至网络前端，从而极大的缓解了梯度消失问题。

所以DPCNN里的Block里加上了shortcut connection后，就完美多啦。即设计出了如下最终版的网络形态：

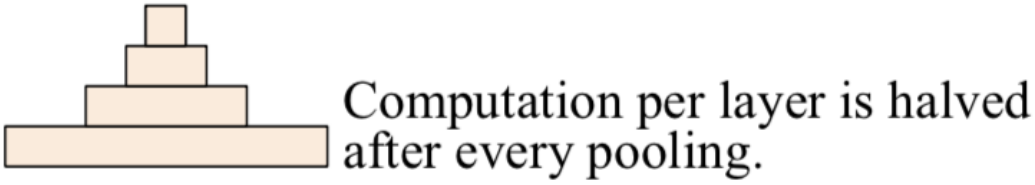


(a) Our proposed model DPCNN

最后点一下题目，由于前面所述的1/2池化层的存在，文本序列的长度会随着block数量的增加呈指数级减少，即

$$num\_blocks = log_2 seq\_len$$

这导致序列长度随着网络加深呈现金字塔形状：



最最后，我猜你们肯定会好奇文章里的小娟姐姐是谁，以后告诉你们咯(￣▽￣)