TextCNN

卷积神经网络应用与图像识别有很高的准确率。利用卷积神经网路识别图像一个很关键的地方是输入数据之间的关联性。我们知道，一张图像的局部位置的数据之间是有很强的关联性的，比如一个红色点周围极大的可能也是红色。

本文的工作是实现TextCNN卷积神经网络模型并用于短文本分类。之所以使用卷积神经网络来训练文本是因为它很好地利用了单词的上下文语境，利用了单词之间的关联信息。而传统的朴素贝叶斯算法、LR算法等处理时是假设单词之间是相互独立的，这必然丢失了很多信息。

【数据预处理】

在介绍模型之前，第一步首先是数据预处理。数据预处理是模型训练过程特别关键的一环。神经网络是对数据集敏感的模型，用一个坏的数据集来训练神经网络的话会得到非常差的效果。试想，假如给你一本错误百出的书，你能很顺利并且无误地获得正确的知识吗？因此，在真正开始训练模型之前，需要大量地工作对数据集做一些处理，使得模型能够很好地从训练集中获得正确知识。

首先是降维：

数据集包括文本60000个，每个文本包含200多个单词。简单地使用词袋模型下的tf-idf矩阵和onehot矩阵作为神经网络的输入显然是不现实的，这种情况下每个单词的维度特别大。

这里采用的降维方式是使用python库中word2vec工具处理文本。word2vec是文本处理的降维工具，它是将自然语言中的字词转化为一个向量。相比于onrhot矩阵和tf-idf矩阵，它的维度更低，而且相似单词之间的向量距离一般比较近，人为地增加了一些单词之间的相关度。

其次是去停用词和低高频词：

停用词和高频词一般包好无用的信息去掉它们可以减少无关信息和噪声对训练的影响。低频词有时很好表征文本，但只出现一两次的低频词意义不大，因此过滤掉了出现少于两次的低频词。

最后是特征加强：

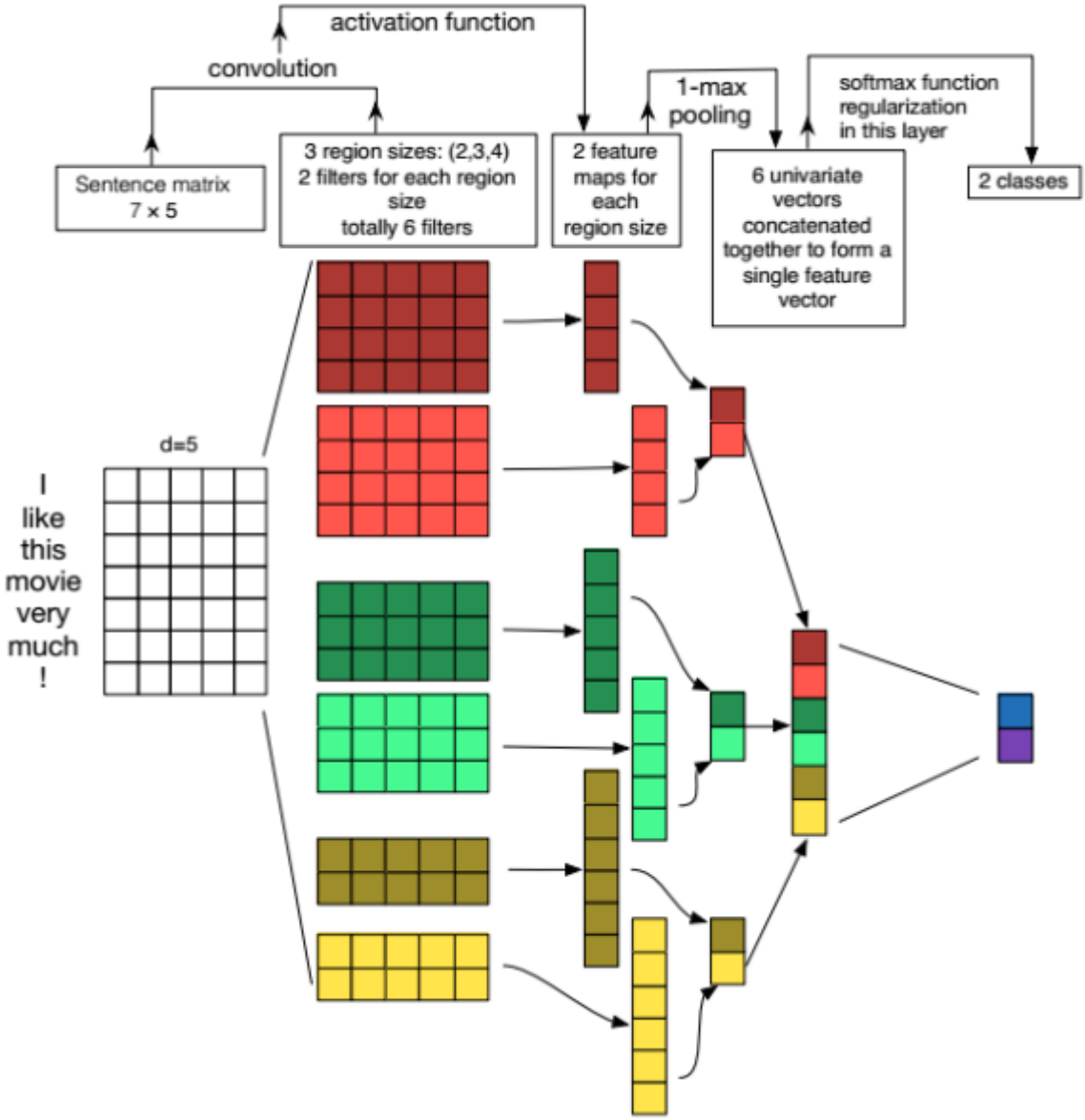
假如给你一张图片让你找出图片中的一只猫来，如果猫在图片中很隐蔽，那么你需要花一些时间才能找出来。但如果我事先将那只猫在图片中圈出来，是不是能一眼就看出猫在哪里？

特征加强实际上做的就是圈猫的过程。这里使用tf-idf矩阵来对word2vec得到的词向量进行加权，而这里的tf-idf代表的是文本中某个词语与某个类别的相关联程度。对于tf-idf的由来，我是用某一类别下单词A的频数作为分子，单词A在训练集的频数作为分母，这样如果单词A在所有类别都均匀分布的化，tf-idf对应的值为1/n(n为类别个数)；如果单词在某个类别出现次数比较多的话，那么在该类别下的tf-idf的值接近为1，意味着该单词与该类别的关联比较大。我们使用所有类别下最大的tf-idf值来作为该单词向量的权重，实际上类似与圈出猫的过程，但我不告诉你那是猫。

【算法原理】

TextCNN模型包括输入矩阵、卷积核、卷积池、池化层和输出神经元。如下图，TextCNN首先用word2vec将单词转化为词向量，借助词向量将一个文本拼接成一个二维的文本矩阵。现在一个文本就转化成了一个跟图像差不多的表现形式。将这个文本矩阵用不同的卷积核对其卷积，经过Relu激活函数后用max-pooling对每一个卷积核提取的文本特征向量进行池化，输出的结果进入池化层。最后池化层的结果经过全连接进入输出神经元，输出神经元使用Softmax分类器分类并输出分类结果。

上述过程有不了解的可以先看下文相关细节的阐述。



下面是上述过程中相关细节的阐述：

TextCNN卷积过程：假设文本矩阵为n x m矩阵，卷积核为k x m矩阵（k<n),卷积开始时，文本矩阵行索引从0开始，划分一个k x m的矩阵与卷积核对应的元素进行点乘并将得到的矩阵求和，将结果保存到卷积层中。然后文本矩阵行索引从1开始划分，重复上述操作最终得到（n-k+1）x 1的卷积层矩阵。

TextCNN的池化过程：每个卷积核得到的矩阵中选择k个值最大的元素，所有卷积核的K个最大值拼接既可以得到池化层。

Softmax分类器：Softmax分类器是logisitc回归分类器的推广，与之区别是Softmax分类器用于多元分类。因此，对于给定的测试输入x，类别数k，我们估计x的每一种分类结果出现的概率用k维向量来表示，最后选取概率最大的值作为类别。

假设函数如下：

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)}=1|x^{(i)};\theta) \\ p(y^{(i)}=2|x^{(i)};\theta) \\ \mathrm{M} \\ p(y^{(i)}=k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \mathrm{M} \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \quad (1)$$

Softmax回归的代价函数为：

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)}=j\} \log \frac{e^{\theta^T j x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

其中：$1\{y^{(i)}=j\}$ 是示性函数，当样本 $x^{(i)}$ 的标签 $y^{(i)}$ 的标签是 j 时取值为1，否则取值为0；对比逻辑回归分类的代价函数，上式同样是求最大似然度。

因此，权重更新梯度为：

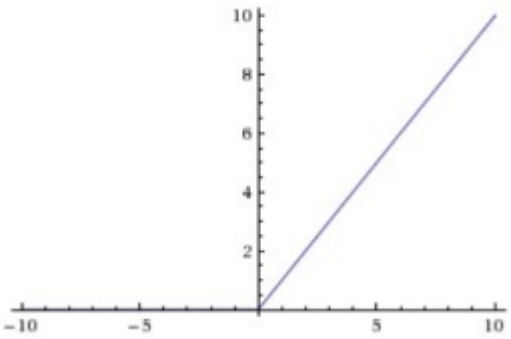
$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}(1\{y^{(i)}=j\} - p(y^{(i)}=j|x^{(i)};\theta)\right] \quad (2)$$

应用Softmax分类器到TextCNN的输出层，正向传播的过程中用（1）式来计算输入样本的属于每个类别的概率，选择概率最大的类别作为输入样本的类别。反向传播过程中应用（2）的权重更新梯度来更新池化层与输出层之间全连接的权重。

Relu激活函数：Relu激活函数是近似生物神经激活的函数，其表达式和图像如下：

表达式： 图像：

$$y = \begin{cases} 0 & (x \le 0) \\ x & (x > 0) \end{cases}$$



选择这个模型有两方面原因：

单侧抑制，只对大于0的输入激活，增加稀疏激活性。从信号方面来看，神经元同时只对输入信号的少部分选择性响应，提取少部分更能表征文本的特征，减少噪音和无关变量的影响，进一步提高学习精度。

,不存在梯度消失，收敛速度快。由于输入的参数比较多，如果sigmoid和tahn等激活函数，实验表明，加权求和后很容易导致输入神经元的值过大而导致梯度消失，是权值更新特别慢。

word2vec：word2vec的基本思想是把自然语言中的每一单词表示成同一纬度的短向量，相较于onehot矩阵，有如下三个特点决定其特别适合TextCNN的输入：

Ø word2vec模型训练得到的词向量维度低，文本数据预处理过程需要对文本特征降维来提高程序运行性能。

Ø word2vec模型在训练过程中，利用上下文单词来训练出一个单词的词向量，因此可以很好的反映词与词之间的相关性。选择word2vec模型训练出的词向量来作为TextCNN的输入，主要是因为CNN卷积通常利用输入之间的相关性来得到文本特征，若单纯使用onehot矩阵作为输入，单词之间独立性很强，很难训练出一个比较好的CNN模型。

Ø word2vec得到的词向量中，意思相近的词语它们的词向量一般距离比较近。由于文本数据没有进过词干提取和词型转变，存在大量意思相近但形式不一样的单词。若使用word2vec，可以比较好得抵消掉上述影响。