**文本分类**

卢立仁

**摘要：**本文将会介绍使用不同的深度学习模型对yelp数据集进行文本分类的过程和相关原理，以及作者的思考。

**关键词：**深度学习、nlp、文本分类、cnn、lstm

1. **文本预处理**
   1. **任务简介及相关说明**

任务简介：本任务是独自完成与文本分类相关的深度学习模型的构建，并使其预测准确度达到最好。

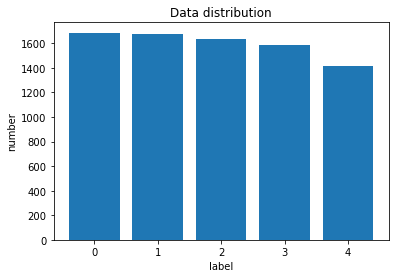
数据集：本文中的数据集为来自yelp（大型评价网站），内容为消费者对商家的评价内容，共有5种类别

训练平台及框架：本任务均在Google colab及九天·毕昇两个平台上进行，使用的框架为Tensorflow

* 1. **数据分析**

从数量上看，训练集共有8001个样本，测试集共有20001个样本，因此对模型进行预训练时需要把测试集的语料也包含进去。

训练集的类别数量如下图：



从图上可看出，训练集的各类别比例均衡，不存在类别不平衡问题，整体质量有保障。

从文本内容上看，训练集中的部分样本内容存在如下图中的特殊符号、URL等不必要的内容，这在文本预处理阶段应该多加留意。



* 1. **文本预处理**
     1. 整体流程



* + 1. Word embedding

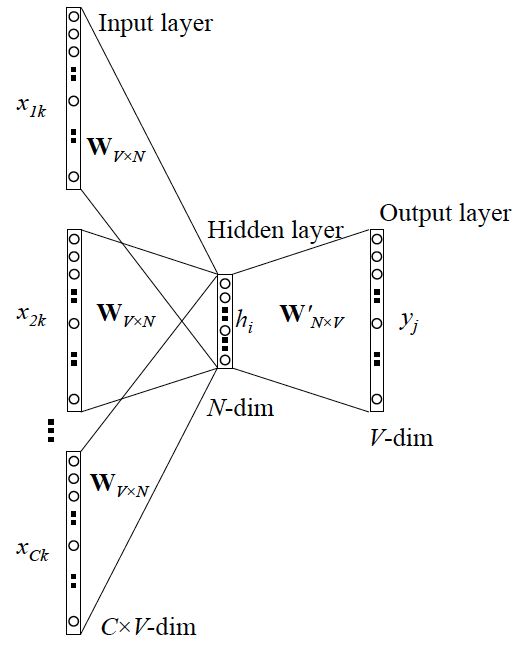
模型选择：word2vector模型中的CBOW模型

模型原理：

当我们构建好词表后，每个词都有对应的编号，我们当然可以用one-hot编码形式对每个词进行编号，然后用每个词的one-hot向量表示这个词，并输入到到模型中计算。但这种做法存在不少缺陷。比如，若词表非常大时，多个one-hot向量耗费的空间资源非常大，且one-hot向量无法表达词与词之间的关系，这个时候就需要用到词嵌入模型来避免上述的问题了。

该模型本质为一个神经网络，网络内有包含词典中每个词的词向量所组成的矩阵，我们将特定词的one-hot向量与矩阵相乘即可得到该词的词向量。我们的目的是通过训练这个神经网络，使得矩阵中的每个词向量都能正确代表每个词。

CBOW模型结构如下图所示：



其中，表示输入到网络中的词的个数，表示每个词向量的维度，而便是我们要得到的词向量矩阵。

CBOW模型的原理为通过上下文的词语预测中心词，其输出层的激活函数为softmax。其工作流程如下图所示：

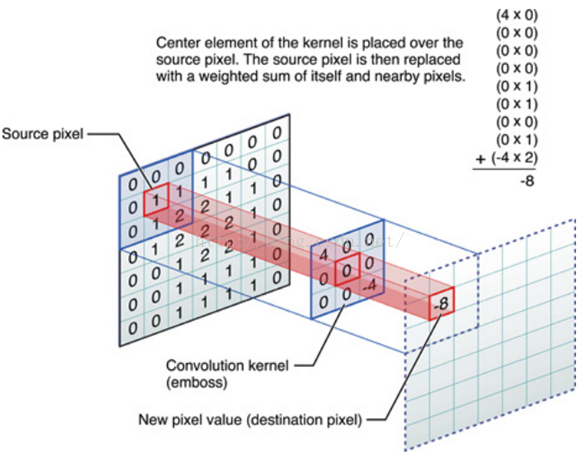


1. **训练**
   1. **模型种类**
      1. CNN

原理：

一个结构最简单的CNN神经网络包括卷积层、池化层、Flatten层及全连接层。

卷积层：从图像说起，首先机器需要在图中选取一定数量的特征，这些特征矩阵也被称为卷积核（filter）。然后，机器在图像上的起始位置开始选择一个与卷积核大小一样的“窗口”，并对窗口与卷积核进行卷积运算，即先对“窗口”和卷积核两个矩阵进行哈达马积（对应位置相乘），形成一个大小与二者一致的新矩阵，再把新矩阵的所有元素相加取平均形成一个新的值，最后把这个值填入到一个新的矩阵中，这个矩阵称为“特征图”。卷积运算的意义：代表了所选取的“窗口”与卷积核内容的匹配程度。大致过程如下图所示：



从卷积运算的步骤不难看出，图像边缘的元素的利用次数较低，从而容易丢失原图的特征。因此，我们往往需要进行padding操作，即往图像矩阵的周边补0，这样能尽量保证特征图与原图大小保持一致。

卷积层的超参数：卷积核的大小（）、步长、padding的层数



卷积层的激活函数为relu，即，其意义为将特征图中与卷积核相关性差的元素（即值<0的元素）全部设为0，使后面的运算更简便。

池化层：本任务选择用max polling算法，即在特征图中选择指定大小的“窗口”，将窗口内的最大值保留下来，该窗口可以像卷积核一样在特征图中不断滑动，由此可以形成一个新的矩阵，使模型的参数变少，运算更快。其工作流程如下图所示：



Flatten层：假设我们在卷积层选用了多个卷积核，则我们能得到多个特征图，从而我们在将参数输入全连接层前就可能有维度大于2维的张量。因此，我们需要将这些张量进行扁平化操作，即将张量变成一个一维的向量，向量的参数数目等于张量各维度数值的乘积。其过程如下图所示：

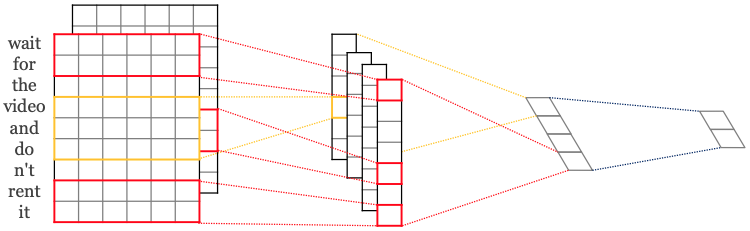


全连接层：在文本分类任务（多分类任务）中，全连接层的激活函数为softmax。

综上，CNN神经网络的流程可如下图所示：



特别地，当CNN用于文本分类时，其卷积核的宽度必须与词向量的维度相同，其大致流程如下图所示：

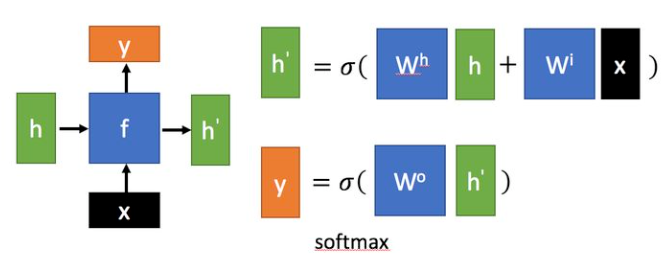


* + 1. LSTM

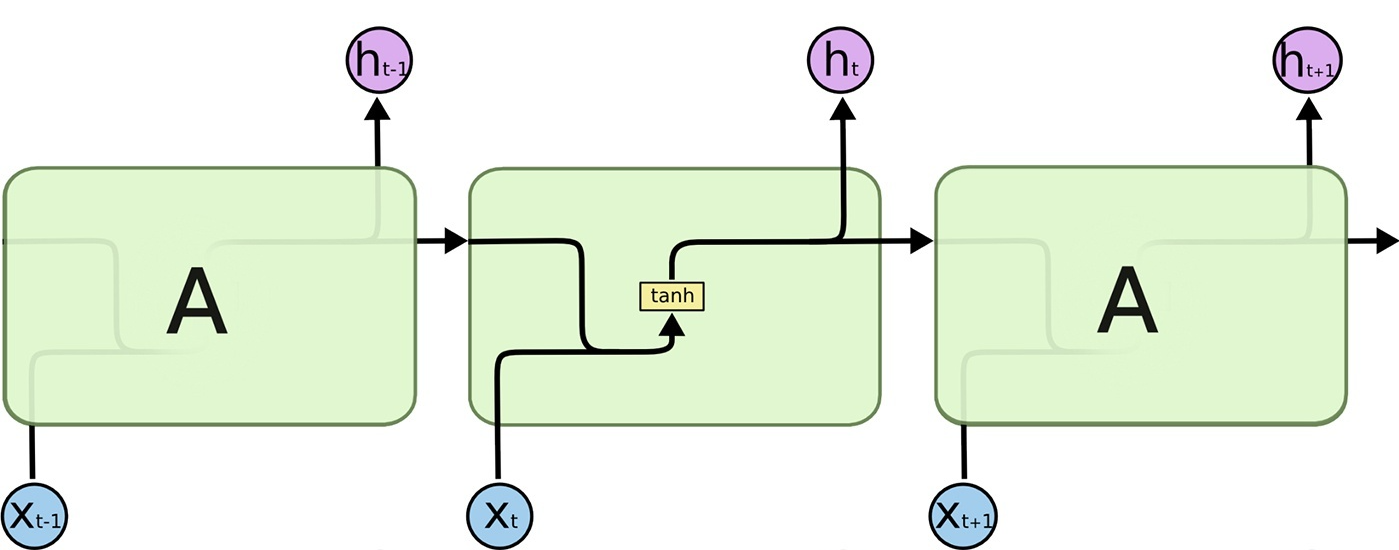
原理：

LSTM是RNN中的一种模型，其改变的主要是隐藏层的结构，因此，我们首先讲讲RNN的原理。假设我们的任务是根据一段话来预测其中个别词的类别，我们当然可以用最普通的全连接网络，将一个词输入网络，得到一个词的类别。但是因为同一个词在不同的语境中所处的类别有可能不同，所以普通的全连接网络很难做到这一点。因此，我们需要一个有“记忆力”的网络，即能通过上下文（至少是上文）的内容使特定词的预测准确度达到最高。那么，RNN网络就具有这种特性。

RNN网络通过每一个“时间片”（句子中一个单词输入到RNN网络称为一个时间片）得到的结果影响下一个时间片模型对结果的预测来达到目的，而每一个时间片所用到的隐藏层网络全部一致，即权重与偏置是共享的。RNN网络的一个时间片的工作流程如下图所示：



如图，RNN网络之所以拥有“记忆性”，是因为的存在，每个单词的输入都会与上一个时间片的一起构成新的,然后影响到预测值的输出。隐藏层的激活函数一般为tanh，输出层的激活函数根据任务而定（本任务为softmax）。多个时间片的工作流程如下图所示：

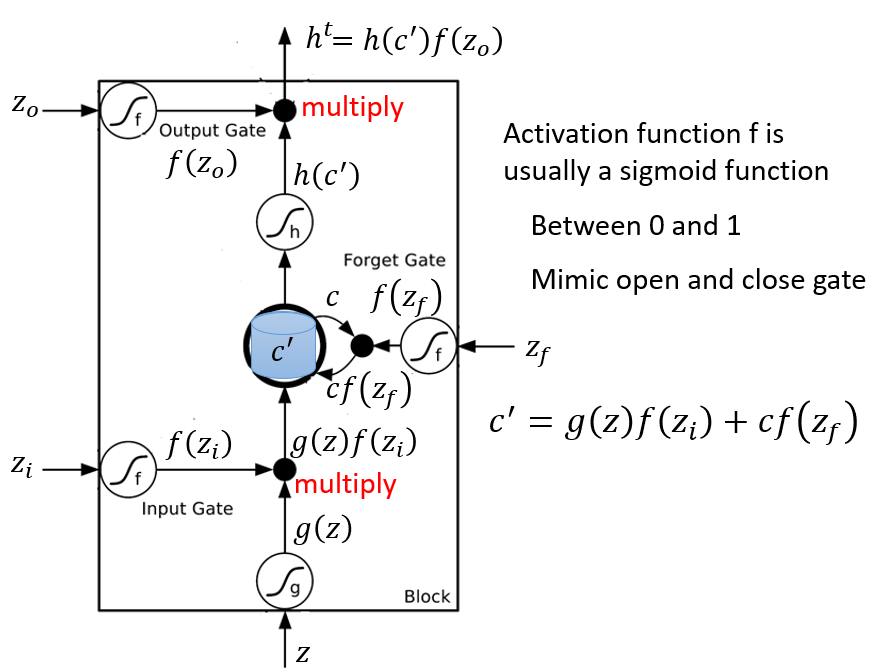
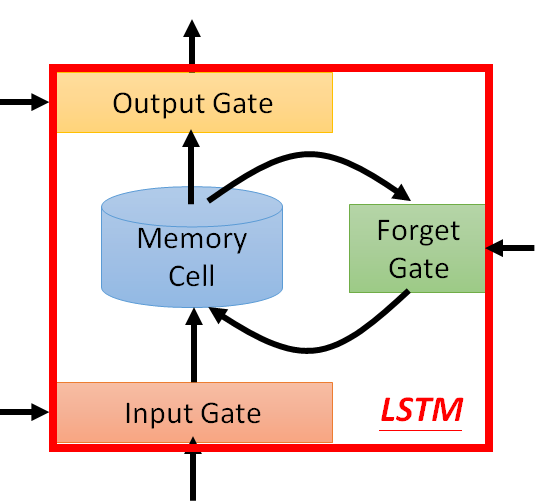


RNN网络存在的问题：由于每个时间片都共享权重，则网络很容易出现梯度爆炸或梯度下降的问题。举个简单的例子，假设每个时间片中，则



如图，在求偏导的过程中，出现了权重连乘的情况，并且随着时间片的延长，连乘的情况还会越来越多。再简化一点，当我们假设激活函数不存在，即整个隐藏层都是普通的线性函数时，在前向传播和反向传播过程中都会出现权重连乘的情况，这时权重的设置会变得十分麻烦，因为设置得小一点容易造成梯度消失，反之容易造成梯度爆炸。那么这时候就需要对RNN模型中的隐藏层做一点改变，LSTM模型就是改变方法中的一种。

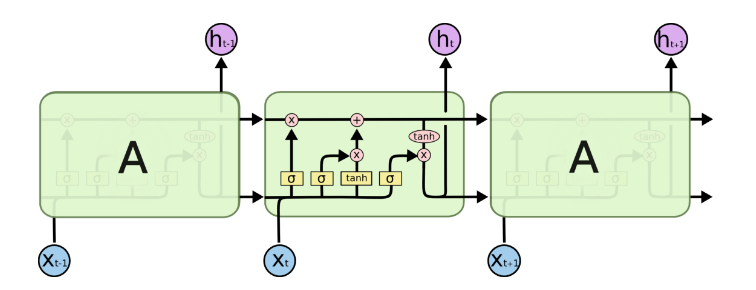
LSTM模型原理：与RNN不同的是，LSTM模型的隐藏层中有输入门、遗忘门与输出门三个结构，并且记忆单元的更新方法也不同，其大致结构及计算过程如下图所示：



从图中不难看出，LSTM模型的记忆单元更新方式采用的是加法，与RNN模型中的不同，其每次加上的都不会使它的值过小，从而解决了梯度消失的问题。另外，当前时间片的也会影响下一时间片各个门的输入值，其影响的方法为与下一时间片的词向量拼接，具体如下图所示：



综上，LSTM的整体工作流程如下图所示：



* 1. **模型训练**

训练流程如下图所示：



训练方法：总的来说是从简单到复杂，即先从最结构简单的模型开始，随着训练的不断进行而增加层数，参数也一样。每次训练完后都记录一下结果以方便下一轮训练对参数进行有针对性的改动。另外，每次改动的参数尽可能少，以便于与上一次训练作对比。

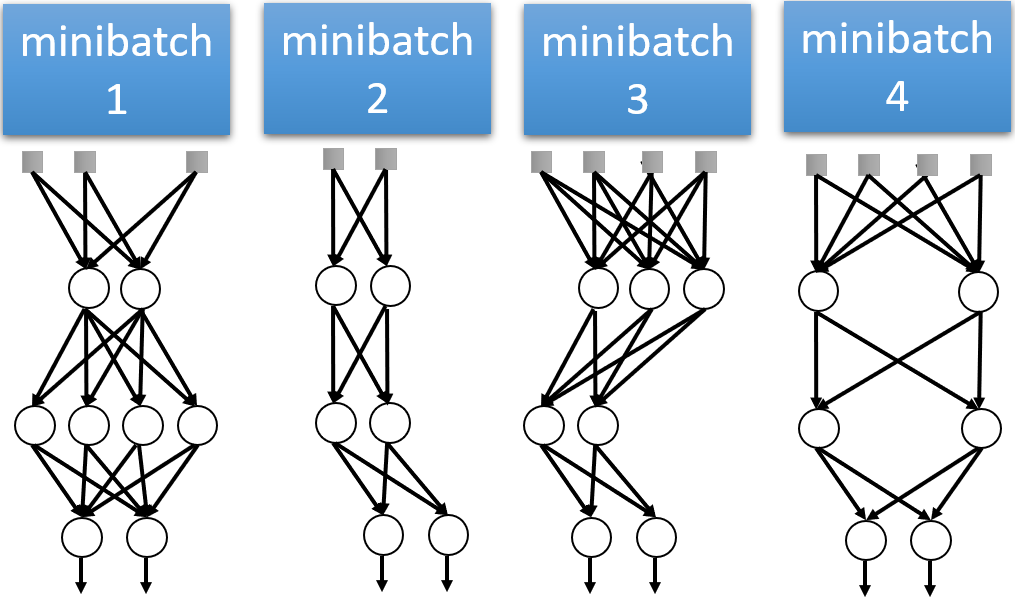
训练结果的记录如下表所示：



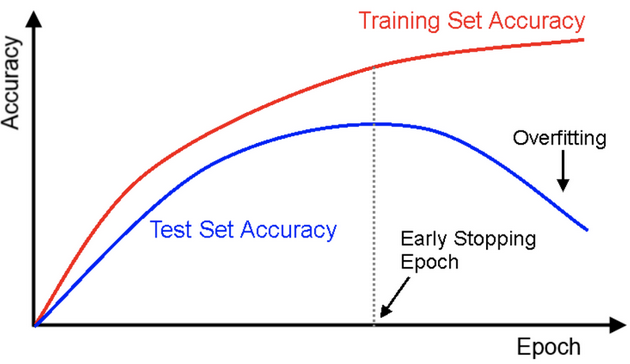
* 1. **防止过拟合**

本文采用的防止过拟合方法主要有两种，即Dropout和Early stopping。

Dropout原理：在训练时，模型会在每个mini-batch开始迭代时根据设置的Dropout rate（比如0.5）屏蔽掉网络中的一些单元（若rate=0.5，则每个单元都有0.5的几率被屏蔽掉），使模型变得更简单，训练速度更快。这种做法相当于用了多个不同的模型进行训练，得出的结果取平均值，从而能降低用单一模型进行训练时产生过拟合的风险。（注意：模型在测试时是不用Dropout的，且各参数要乘以(1-rate)）Dropout原理如下图所示：

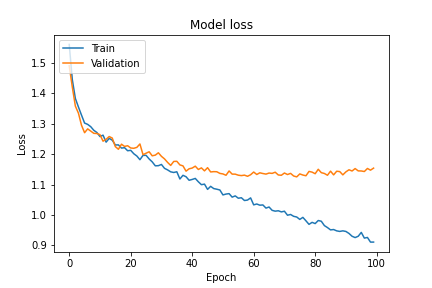
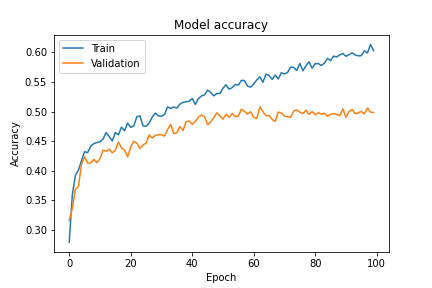


Early stopping原理：随着迭代次数的增加，验证集的精度可能会出现停滞现象，这时可能发生了过拟合，因此无论再怎么迭代，验证集的精度也不可能上升，甚至可能下降。Early stopping的作用是在模型接近过拟合时保留验证集指标最好的时间点的参数并记录其指标值，若在之后的数次训练中验证集的指标都无法超过记录，则训练停止，模型的参数采用的是指标最好时的。技巧：首先设置适合的迭代次数让模型接近过拟合的状态，然后在迭代结束之后引入Early stopping，再设置迭代次数开始新的训练。Early stopping原理如下图所示：



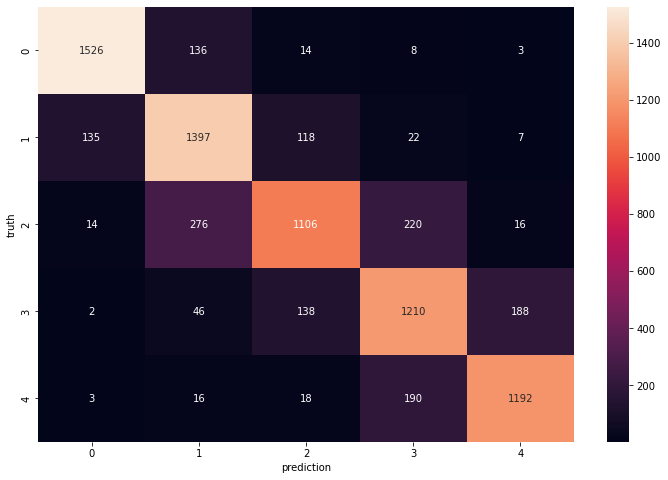
1. **模型评估**
   1. **评估方法**

通过准确率和损失值的变化趋势评估：



从图中不难看出，模型在前60次迭代的拟合效果不差，但在这之后验证集的准确度就没怎么提高了，而且损失值有所增加，属于过拟合现象。

通过混淆矩阵评估：



对角线处的值为预测正确的样本数。从图中不难看出，本模型对于多分类任务的各个类别的预测正确率比较均衡，对类别为2的样本的预测错误数量较多。

* 1. **评估结果**

从参数上看，lstm模型的参数明显比cnn要少，因此调参的时间相对较短、资源耗费相对较少；

从训练时间上看，lstm的训练时间明显比cnn要长，且迭代次数明显比cnn多；

从准确率上看，lstm的平均准确率比cnn高，但二者差距不大。

下图分别为cnn与lstm的参数



各模型的准确度对比：

**Table 1** The accuracy of different models

|  |  |  |
| --- | --- | --- |
| Model | Corpus | Average accuracy |
| CNN | train | 0.451 |
| CNN | train + test | 0.486020313 |
| LSTM | train + test | 0.501283920473522 |
| Bidirectional LSTM | train + test | 0.47651467770338 |

从上表可以看出，并不是越复杂的模型准确率越高，反而最简单的模型效果更好。

1. **总结**

这次考核最使我头疼的是入门的阶段，因为有太多没有见过的名词与概念，所以我在头一个多星期内连词表与词嵌入所形成的词向量之间的关系都没搞明白，这使我的心态变得很差，进度也越来越慢了。但还好我没有彻底放弃，并及时调整了心态，还改变了学习的策略，即不管细枝末节的东西，先把模型的整套流程学习一遍，这样能使我对相关概念和步骤有更加全面的认识。当我走出了阴影后，第一个模型也就顺利完成了，但问题就是准确度不够高。起初，我以为是超参数设置得不好，但当我调了几十次模型的准确度还是没有明显提高时，我意识到可能是文本预处理的工作没有做好。于是，我回头看了一下前面的工作，发现，原来我没有用到测试集的语料。因此，当我用了测试集的语料之后，模型的准确度果然有了很大的提高，这令我意识到回顾也是很重要的。现在的问题是准确率不能突破0.52，我认为还是文本预处理的问题。不管怎么说，这次考核使我学到了很多，更锻炼了我的心态，是一次非常不错的体验。