**基于深度学习的自然语言处理**

**第2次实验报告**

**一、实验名称**

使用深度学习方法训练新闻分类模型

**二、实验日期**

2020.10.09 – 2020.10.23

**三、实验目的**

（1）熟悉深度学习的各种方法

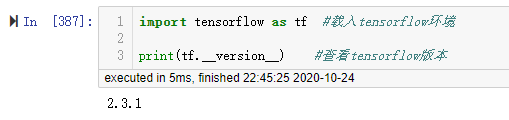
（2）熟悉开源框架（keras, tensorflow）的使用

**四、实验数据**

提供实验的新闻数据

**五、实验步骤及结果分析**

**Tensorflow版本：**

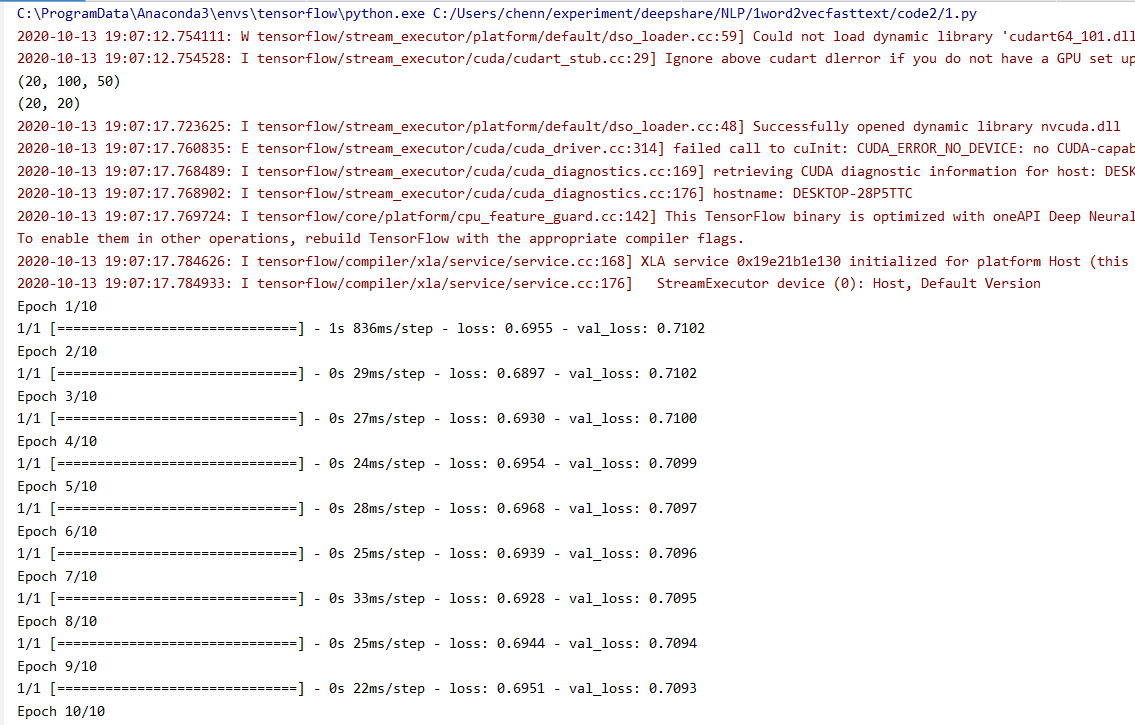


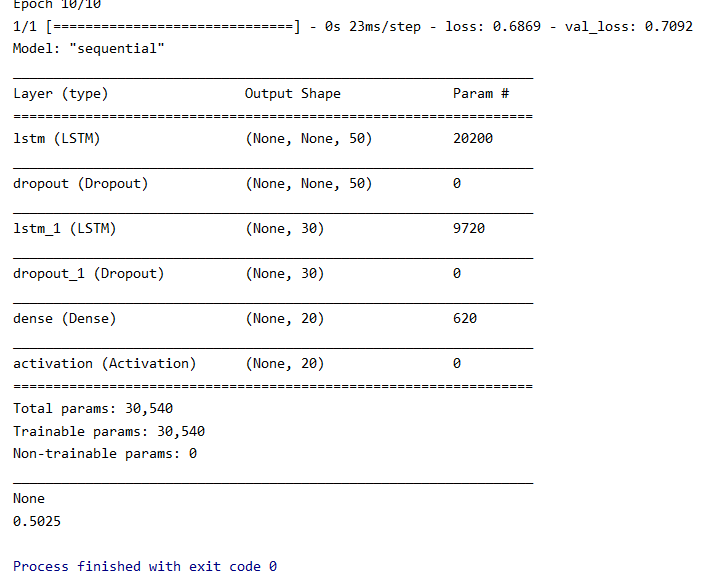
**此版本下不能用GPU（当前为CUDA9）**

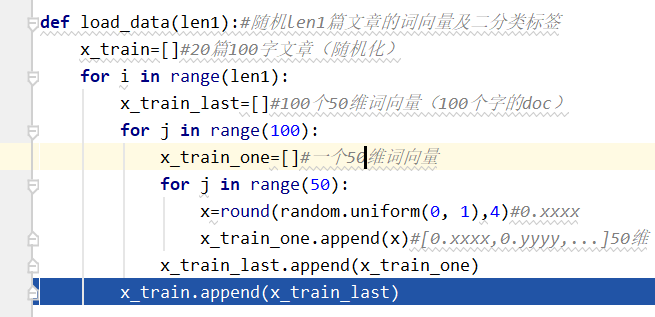
（1）进入百度网盘下载语料content.txt

链接:https://pan.baidu.com/s/1QyVnxUTrJ2Dqj7FIpzuhgg 密码:lekc

在最之前使用助教给的代码（随便造一点词向量训练）







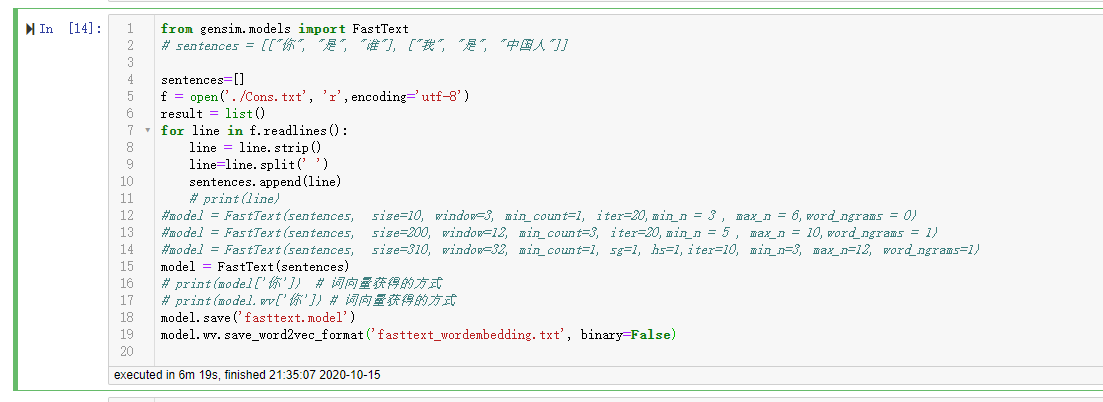
造出来效果正确率50%，还将就。

（2）对语料进行预处理：去掉一些无用的标点符号

对所有文档去除了标点符号，停用词等。

（3）对处理后的语料以及实验1的方法训练这些语料的词向量（Word2vec或者FastText），或者使用开源的词向量BERT等。

将所有词分词并使用FastText训练100维词向量，词向量结果见fasttext\_wordembedding.txt。

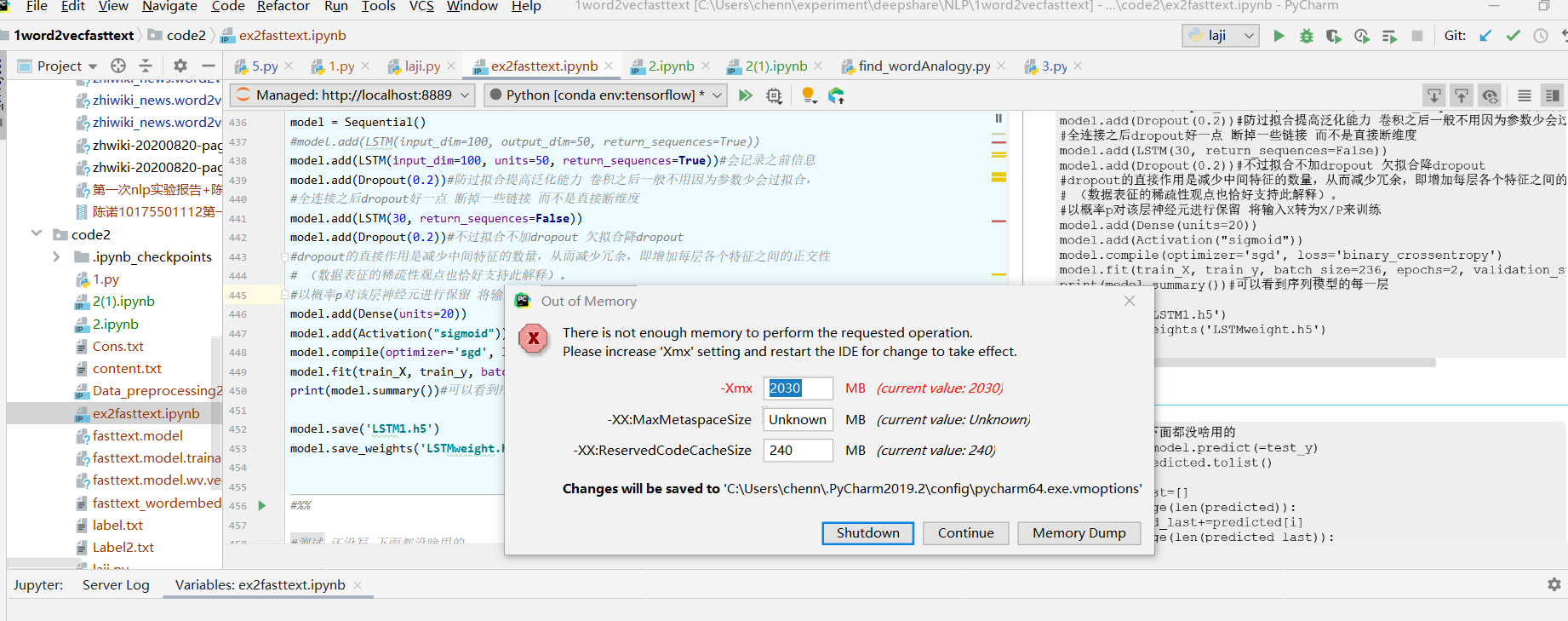


（4）划分训练集和测试集（总共60158条数据，前55000条为训练集，剩下的为测试集，如果一条数据对应的label为空，则不使用这条数据）

最初使用所有数据集（60158数据不变，且文章不截字数）



估算60158\*500(假设一篇文章500字)\*100好像是约16G，总之塞不下内存，所以先试前1001条，结果训练了一天动都没动，甚至连进度条都没有， 所以终止了。选择前1000篇作为文本并每篇文章选前300词训练。（最多一篇文章有一万多词）



（5）使用训练好的语料对每条语料进行表征

表征结果如下



（6）对标签进行预处理，处理成0/1的格式（n-hot的格式[0,1,1,0,0……]）

多标签分类任务对每个类都是一个sigmoid，所以不需要化成概率的形式。

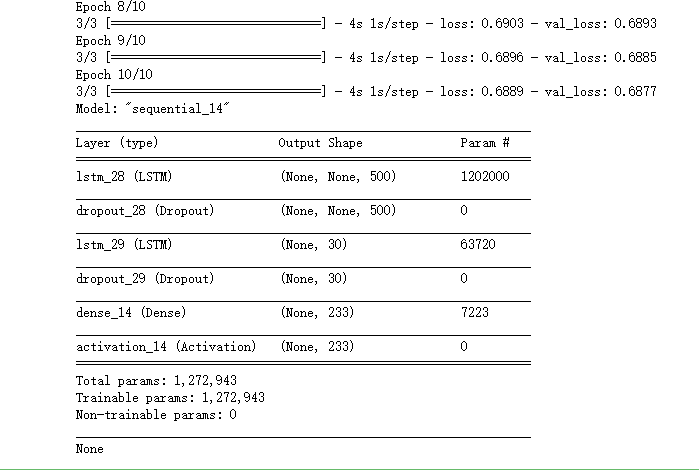
（7）使用开源框架搭建多种不同的模型进行训练（同时使用keras和TensorFlow两种框架完成下面的模型）

**模型三：两层单向LSTM + MLP做分类（给的例子）**

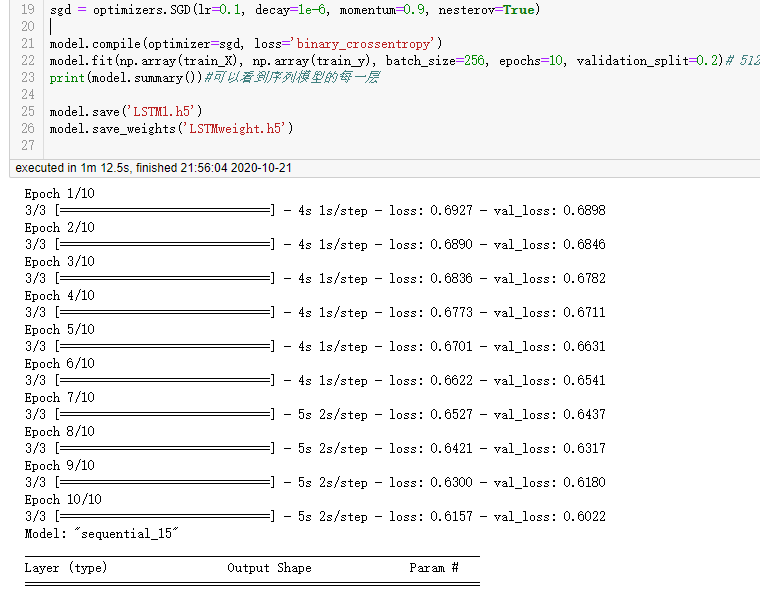
（8）测试训练好的模型（使用我样例上的测试代码）

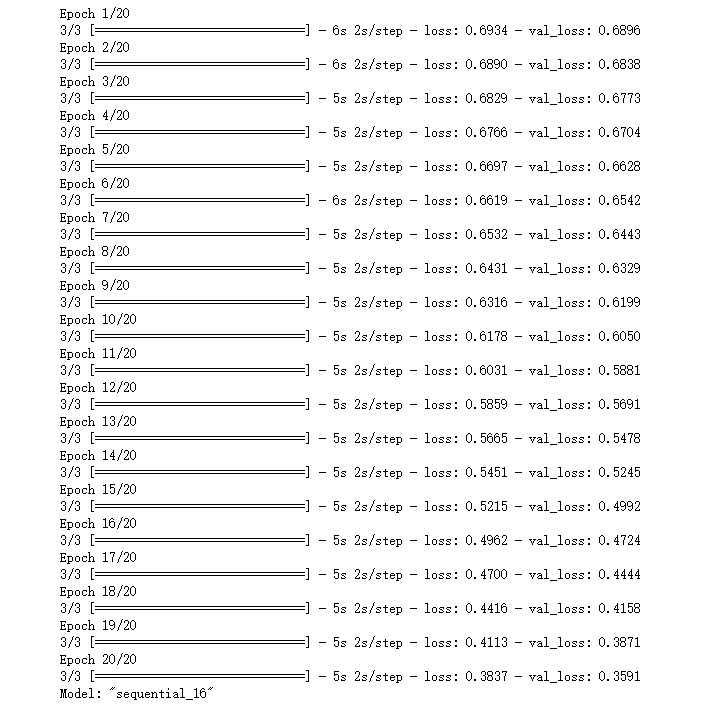
（9）调试模型参数提高测试集的精度



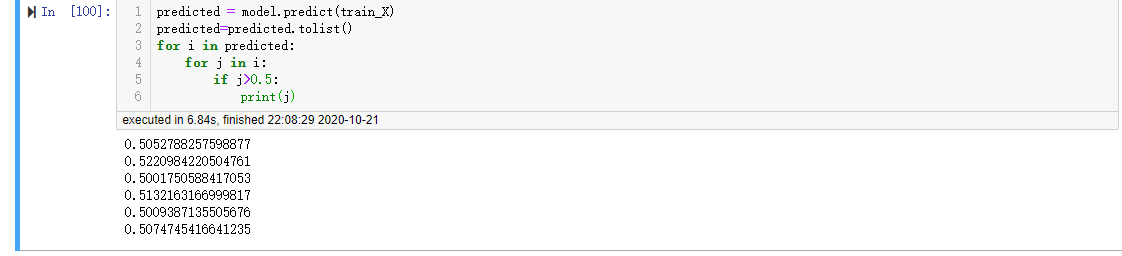


使用默认的sgd可以看到loss降得特别慢，那就把学习率调高点。





可以看到调整学习率以及epoch之后，loss下降地很快（甚至感觉怎么调怎么还能收敛），但是



可以看出由于标签太多，所以预测完全偏向于猜0。而f1之类这种指标，虽然是可以解决标签稀疏的问题，但是他不能用作损失函数，所以似乎只能靠取少一点标签以达到效果。（训练损失越低，最后越倾向于预测全是0）

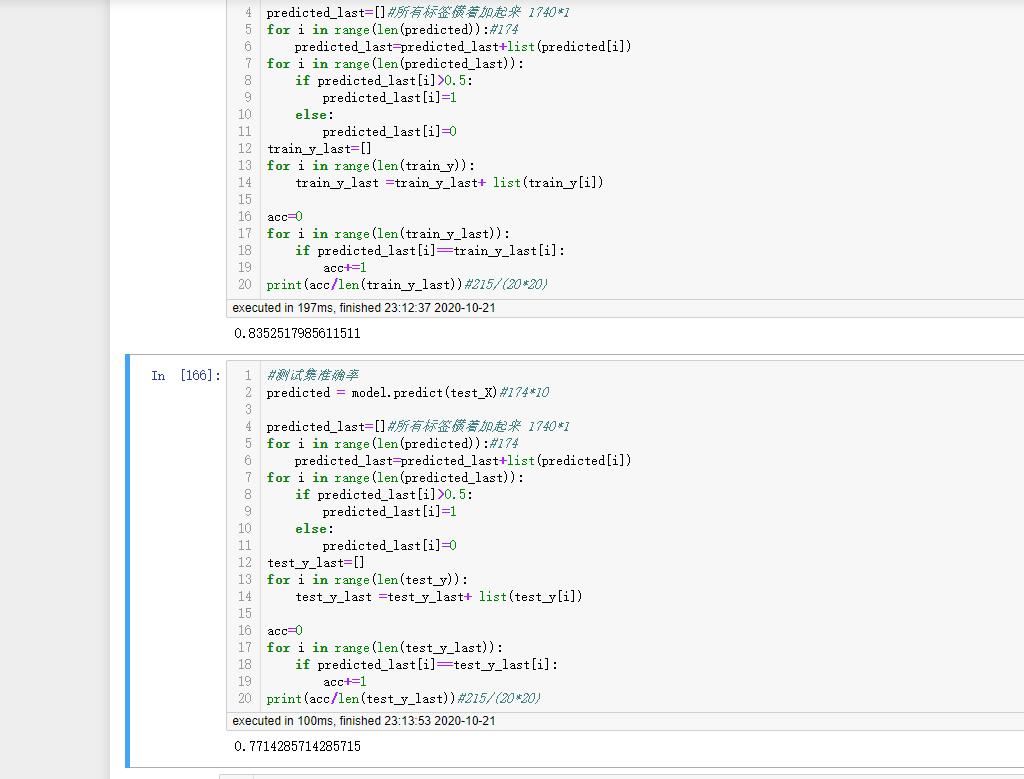
因为不是二分类问题标签不平衡，所以甚至没法过采样。。所以只能砍标签。砍成十分类问题（出现最多的10个标签），并且筛选那些至少在这十个标签中出现2个标签及以上的文章，在这些文章中找出10740篇.（每篇文章截取前100字，不然太大不好算）

同时，使用adam而不是sgd，效果更好。



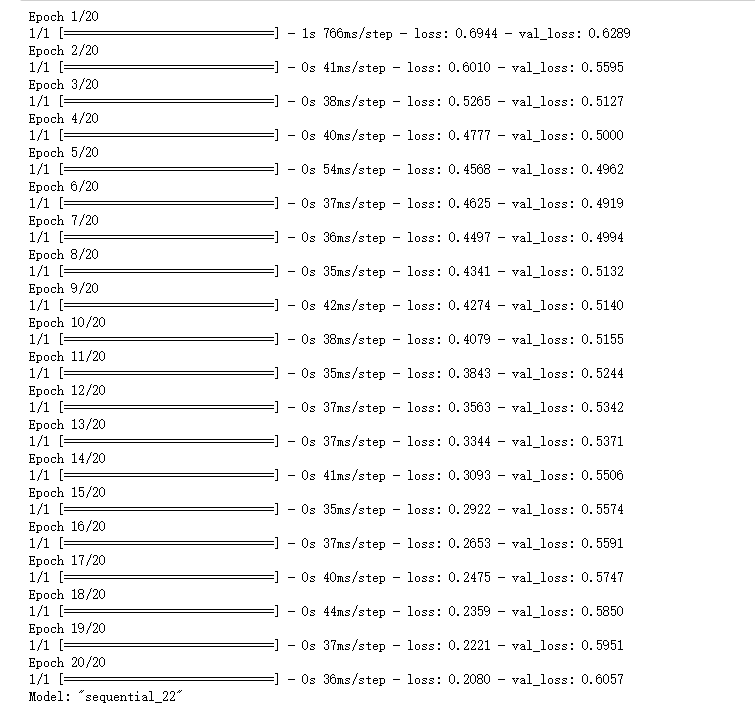
使用adam=optimizers.Adam(lr=0.01, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08)

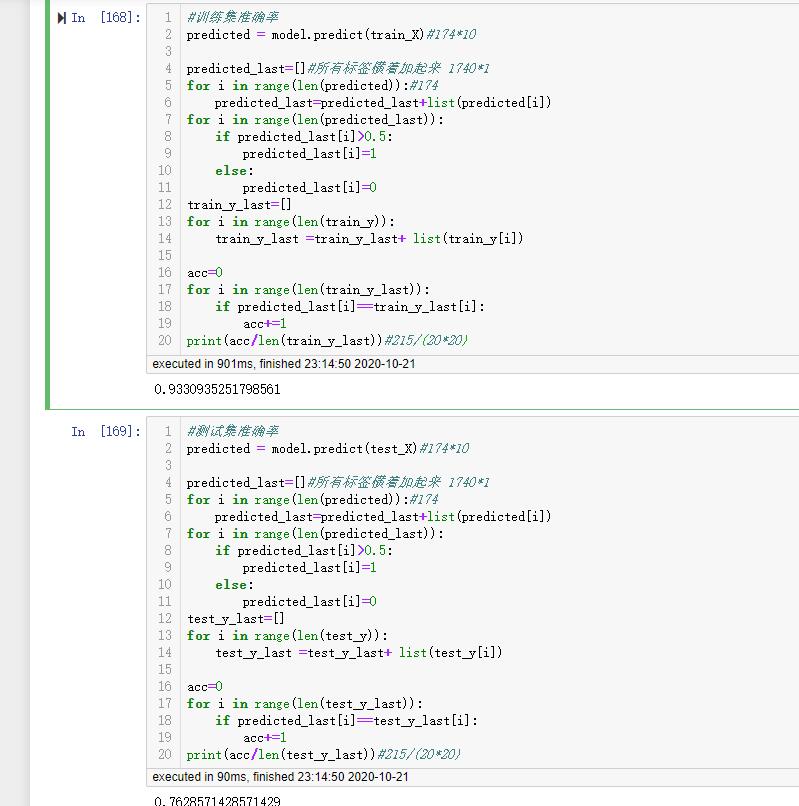
epochs=10

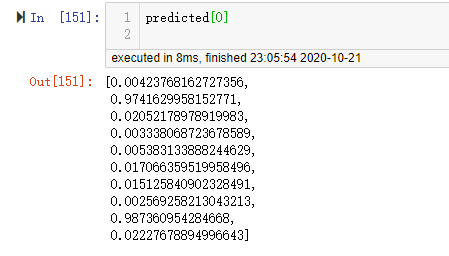
则

训练集0.83，测试集0.77.

尝试epochs=20

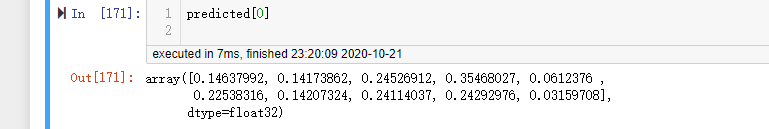




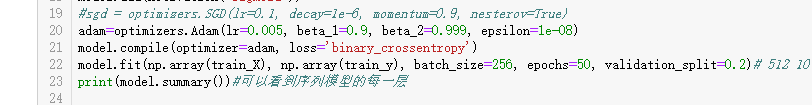


可以看到训练集预测更极端，表现得有些过拟合。所以需要降低学习率及epoch数量防止过拟合。

调整学习率为0.002，epoch为20

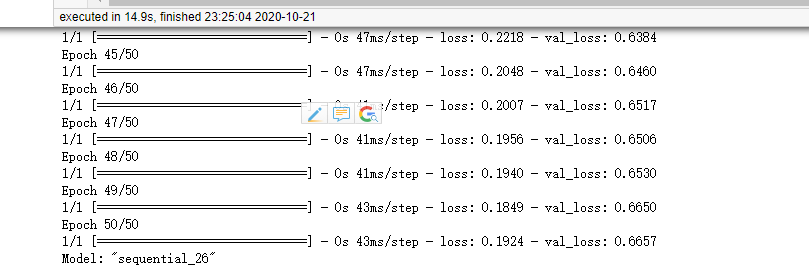


可以看出明显欠拟合。

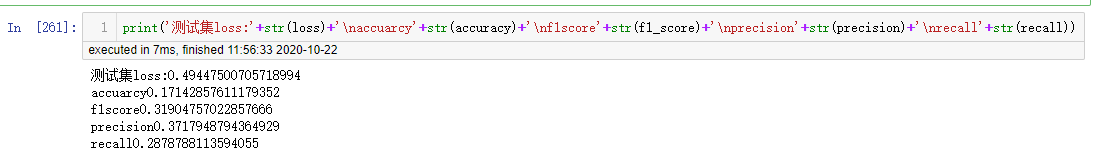




改成0.002，50



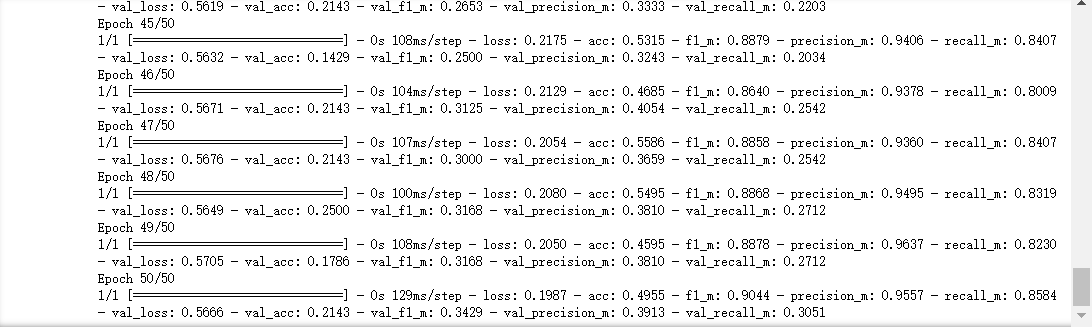




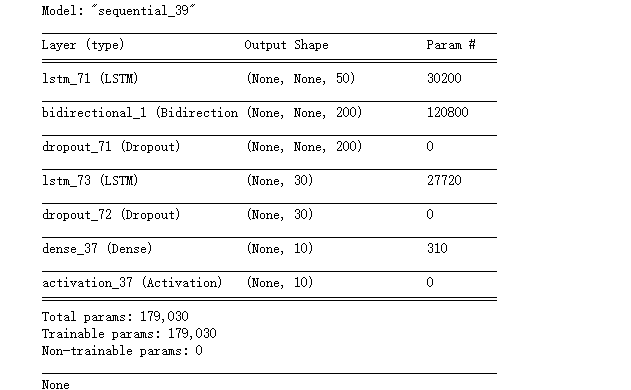
此时测试集准确率达到0.7914，f1达到0.3（标签完全对应才算正确的f1score），表现较好。

**模型四：两层双向LSTM + MLP做分类**

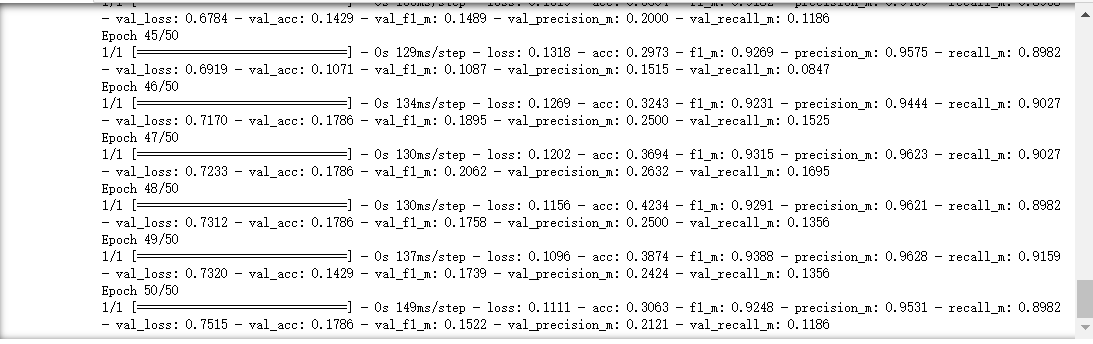
使用一层双向一层单向，效果比单向还差。





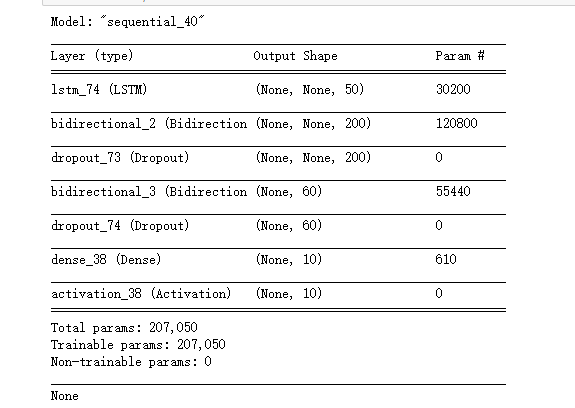


使用两层双向lstm

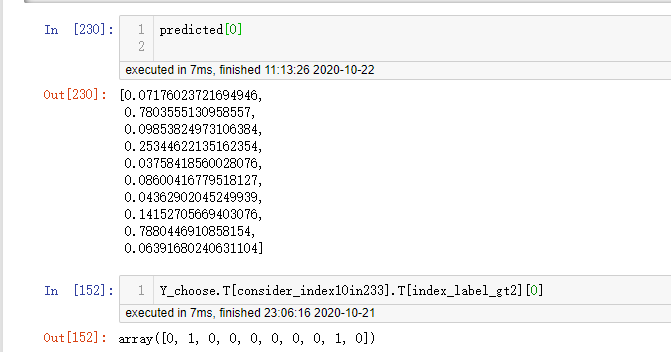




效果有些许提升，故尝试调参。



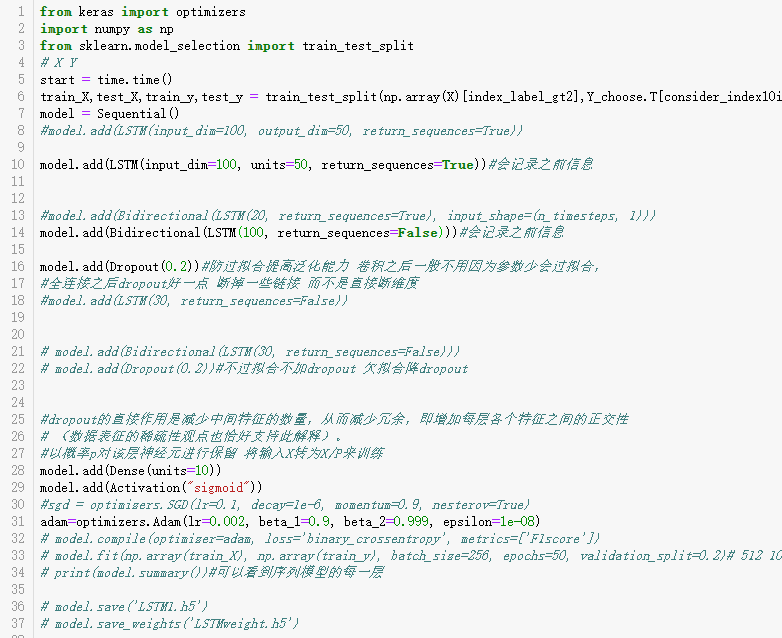
但不论怎么调参，模型还是表现出严重的过拟合情况。下图为第一个文档的真实与预测标签

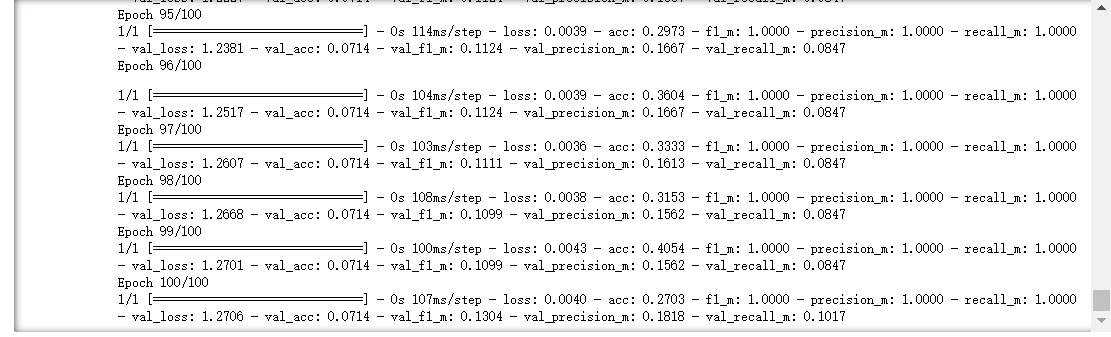


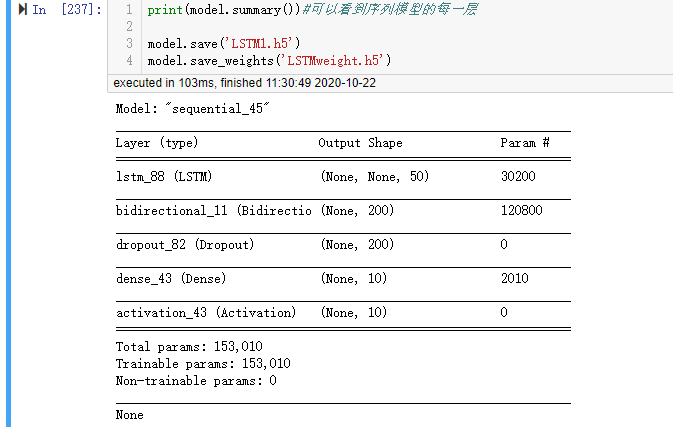


增加epoch后，训练级分数依旧增加迅速，但是测试集准确率几乎没有提高，所以有可能是模型本身的问题。

**模型二：一层双向LSTM + MLP做分类**



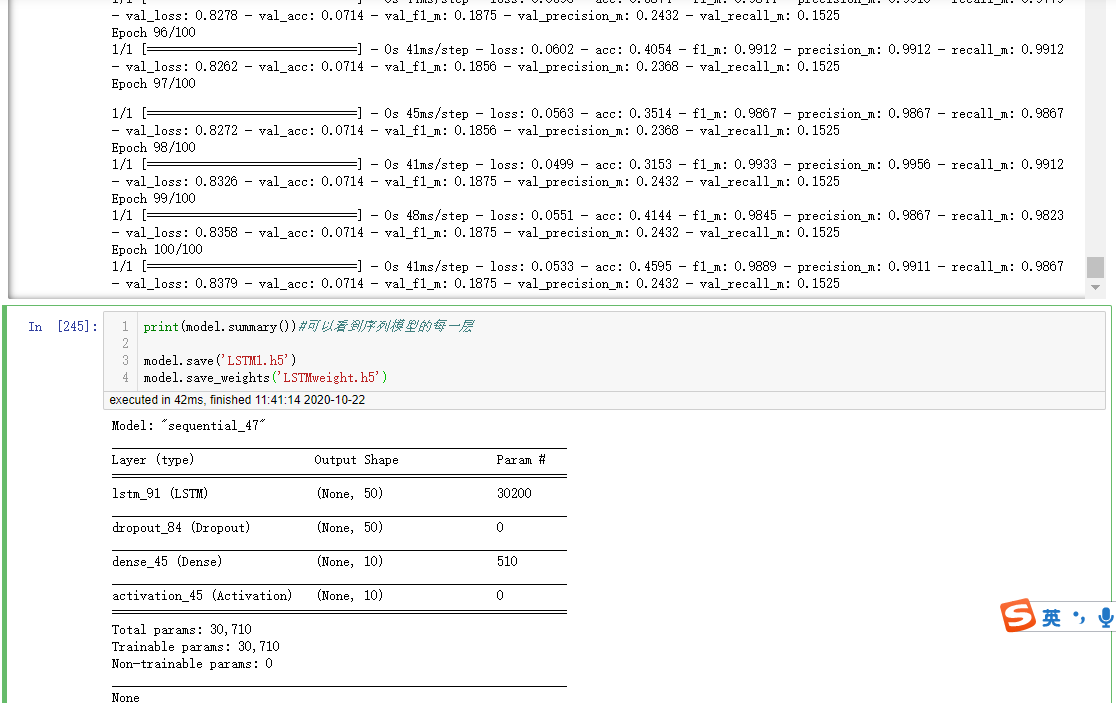


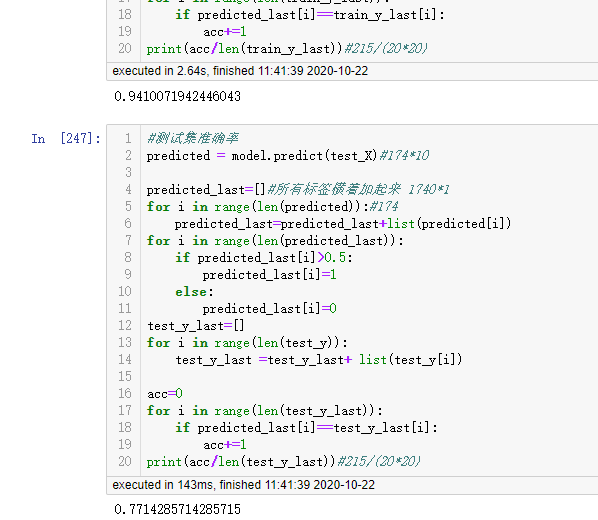


F1score为1了，说明严重过拟合，但是测试集准确率相对也不是很糟。

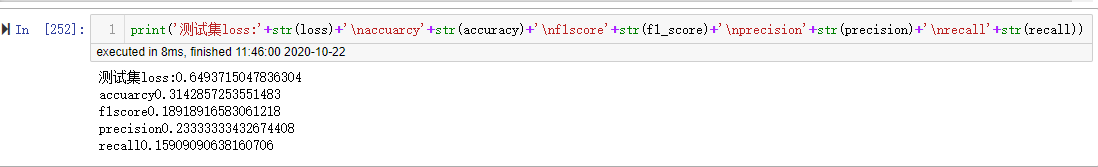


**模型一：一层单向LSTM + MLP做分类**



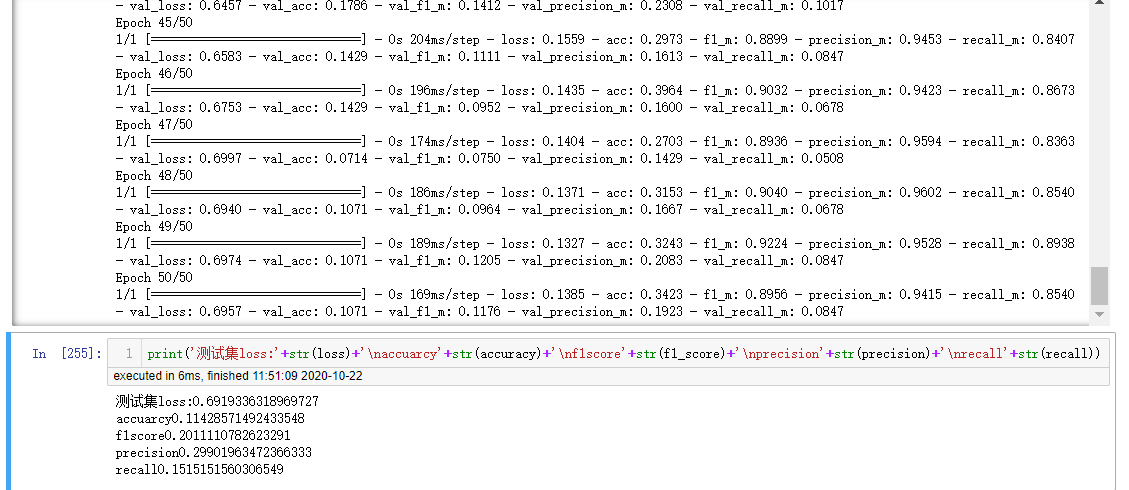


可以看到测试集准确率反而相交双层lstm有提升，这是违反常识的。而且f1score都已经是1，但是val\_acc只有0.07（其实val\_acc没什么用，一位内要所有label完全预测一致才算acc正确），表现出明显地过拟合。然而使用上述准确率作为评价标准，对于label平衡的问题来说可以，但是label不平衡的情况下，应该看如下的测试集f1score的指标。



上图为单层LSTM 的测试集f1score指标。

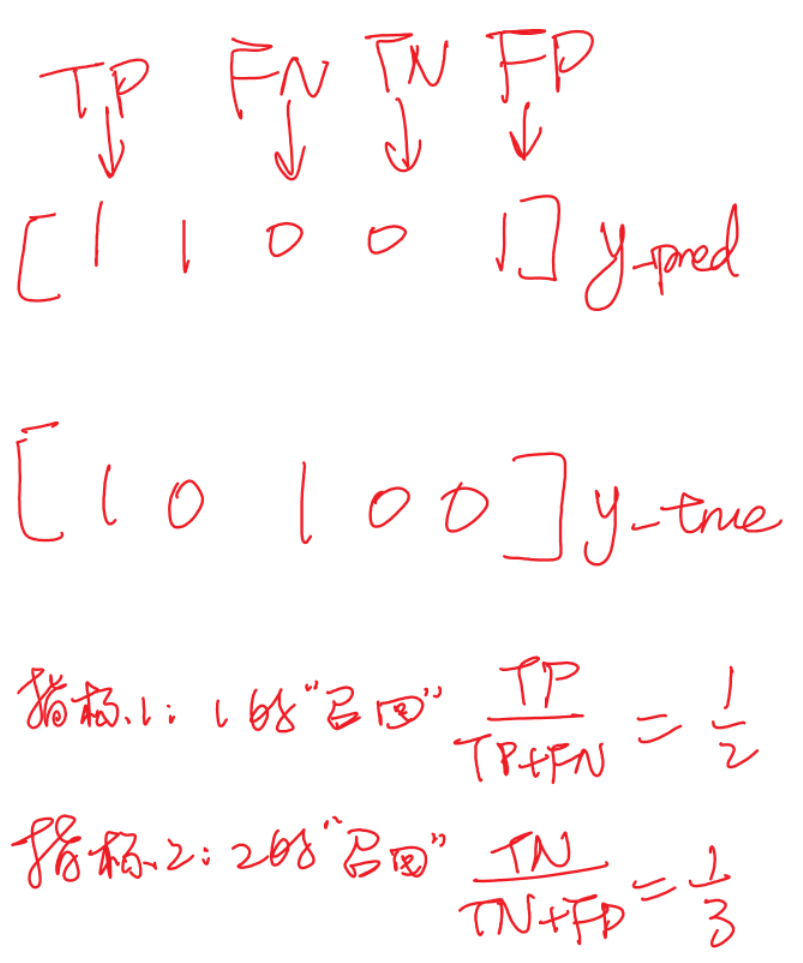
双层双向LSTM 的测试集结果如下。

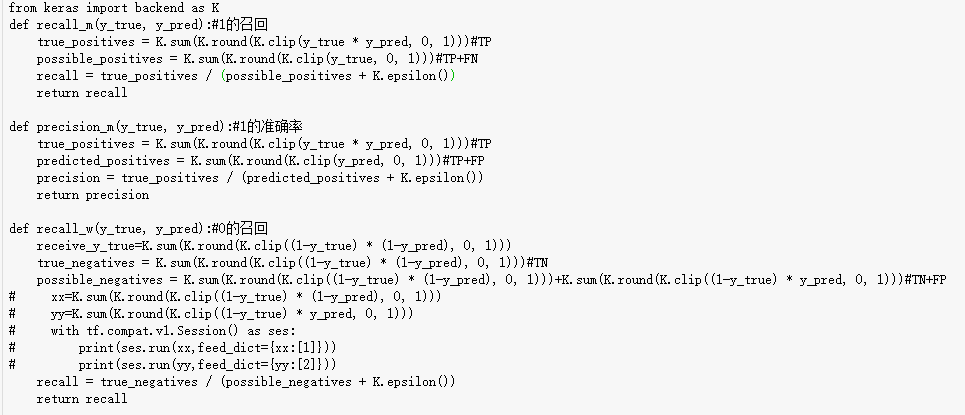


测试集在f1score（标签完全对应才算正确的f1score）约0.20左右。

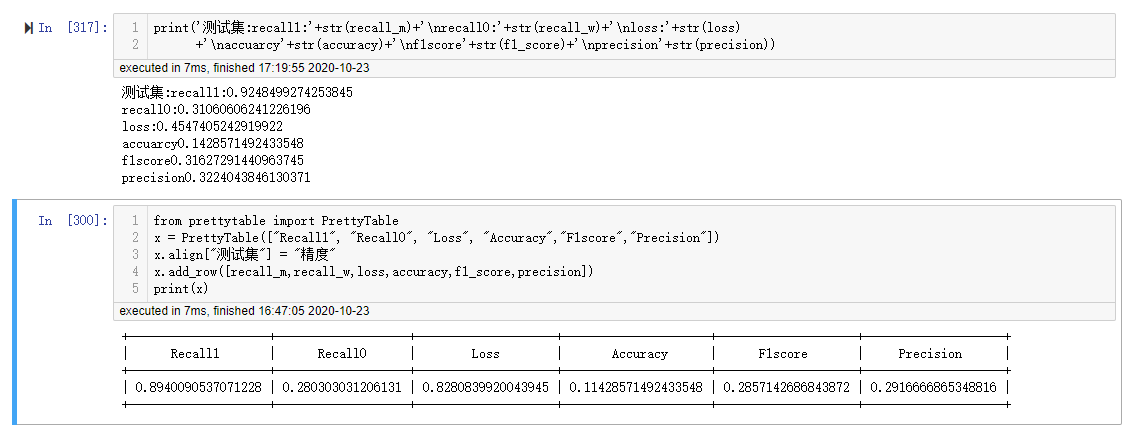
补充： 评价标准

由于是多标签分类问题，所以不能单纯按照测试代码中给出的一样都是1或者0就是对，不一样就不对的方法。而应参照下图所示。（一般工业界用的是top5评价标准，这里使用如下评价标准）

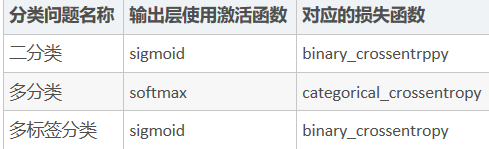


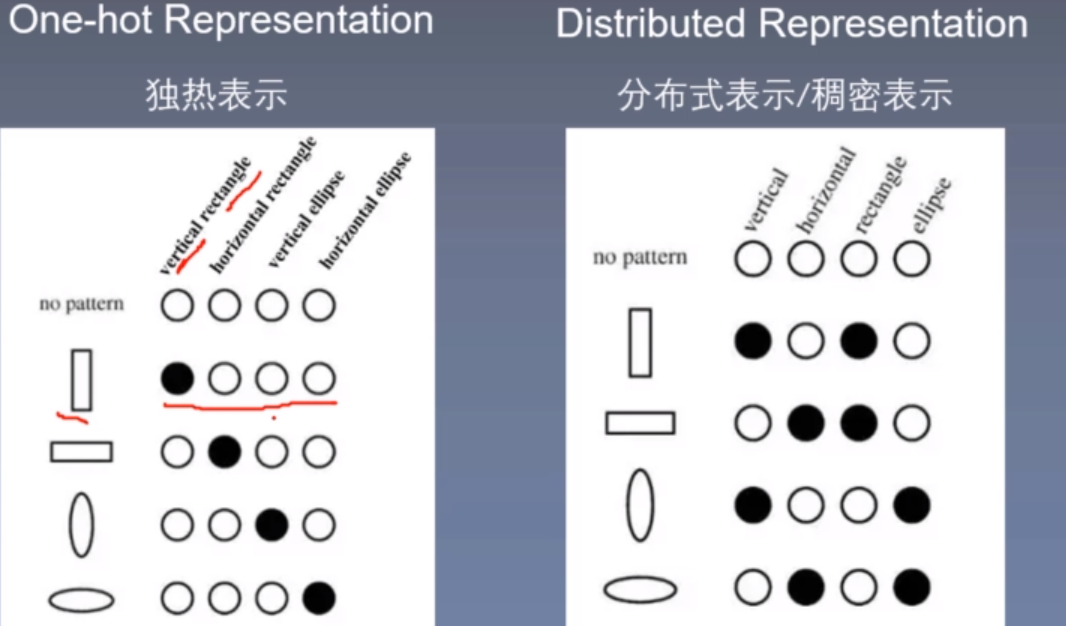
即

在该标准下， 以上最优参数的最高准确率如下：



**六、实验感想**



**对多标签分类问题，每个标签算一个二分类，使用一个sigmoid，最后将问题叠加。若二分类问题遇到标签不平衡，则可以使用过采样等方法，但多标签问题则不可使用该种方法，只能使标签尽量不稀疏（因为类似f1score的评价方法不能作为损失函数，因为不可导）。这样使得我们必须保证数据质量，这也与embedding的思想类似，**

**使用onehot的表示方式（类比若每个文档只有很少label的多标签任务）能表示的信息几乎没有，但若表示向量稠密，则能包含更多的信息。所以在本次实验中只选取10个label，并且只选取在这十个label中至少出现两次以上的文章进行训练与测试（按照0.8：0.2划分训练测试集）。这样避免了模型越收敛约偏向于猜全0。而在embedding的过程中，使用fasttext 对全文进行embedding，这样相交使用已经训练好的词向量的好处是训练结果能包括所有文章内部蕴含的词与词的关联信息。在训练最后LSTM模型的过程中（词向量都放在内存，从文件读边读边训练读一天都读不完），通过观察不同参数训练集和测试集的准确率变化，来调整到最优参数以防止过拟合，并且使用adam而非sgd作为优化器，根据每次迭代的情况优化学习率，以更好地逼近收敛，最终在测试集达到较高的准确率。具体代码解释详见代码注释。**