**文本分类实验报告**

1. 实验描述

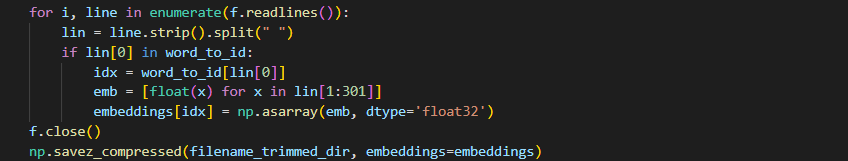
根据训练集train.txt和搜狗字向量，训练TextCNN模型，对test.txt中的文本进行预测。

本次实验的代码总共存放在main.py,model.py,utils.py,train\_eval.py四个文件中，其中所有参数均由model.py文件下的config类统一管理。

1. **实验过程**

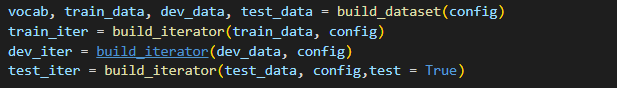
由于语料较小，考虑到准确率，本次实验选择从网上下搜狗新闻已经预先训练好的字向量，所有训练和测试以字为标准。

首先训练模型前运行python utils.py，从train.txt从抽出所有字加上UNK和PAD组成字表，去除所有空格和数字，但保留可以辅助感情表达的符号，随后从搜狗新闻中训练好的字向量中找到字表中每个字对应的值（方便梯度计算，长度设为300），组成train.txt自己的字向量文件，存储在./Mydata/data/embedding\_SougouNews中。

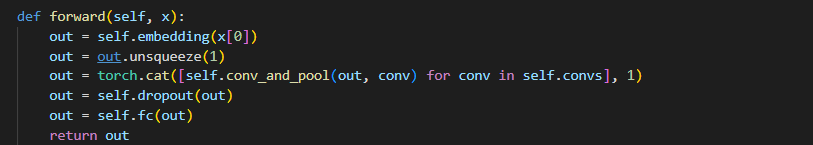


考虑到模型的泛化性能和防止过拟合，从训练集中取后五百条作为验证集，同时设置神经元随机失活的概率为0.5。

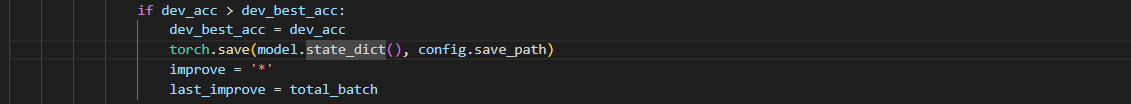
在训练模型之前，对数据进行处理，首先构建训练集的字表，将训练语料中的每个字提取出来，并从已经训练好的字向量文件中找到自己的字向量，重新构建整个句子的字向量，同时，为方便后续处理，对句子短填长切（所填字符均记为PAD），保证每个样例对应句子的长度都为64。其中对于验证集和测试集的数据进行处理时出现的不存在于训练集中的字，一律作为UNK处理。随后分别构建相应迭代器方便后续处理。



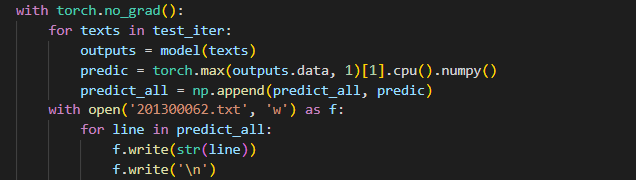
模型采用CNN模型，分四种大小（1,2,3,4,5），长度超过5的字组合基本可以拆分为上述四种。每种一百个卷积核，在样本embedding为字向量后，进行卷积并拼接，最大池化之后通过全连接层计算其被分到每一类的概率，随后与样本标签计算交叉熵，作为loss交给优化器进行梯度下降优化。



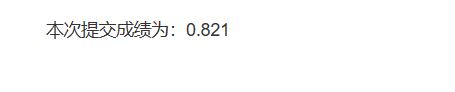
为了防止过拟合，每训练过100bacth进行一次验证集预测，若正确率取得新高，则储存模型参数，同时，为了防止训练时间过长，若超过1000batch验证集性能都无法取得更高，则默认为模型已经开始过拟合，结束训练。



训练时读取模型参数，并将结果写入201300062.txt之中。



最终测试结果如下：



1. 结论与感想

词向量作为CNN文本分类的核心，在训练中的作用举足轻重，如果包含一个所有字符的字向量表，将显著提高分类准确率，如，若在本实验中使用所下载的全部搜狗新闻的字向量，那么那些在test中而不在train中的字也有自己的准确字向量表示，可以通过与train中已有字向量近似的方式，使test的预测更加准确，事实上，使用全部字向量的准确率相比于本次结果提升了0.03。