实验说明

实验环境：python3.6

使用到的模块：

**from** word\_similarity **import** WordSimilarity2010  
**import** jieba  
**from** scipy.linalg **import** norm  
**from** sklearn.feature\_extraction.text **import** CountVectorizer, TfidfVectorizer  
**import** numpy **as** np

**import** jieba.analyse  
**import** re

1.原始数据xml文件：（中文语料）

2.经过xml处理.py处理成txt文件：



上图表示解析测试数据，file1用于存储中文测试数据，file2用于存储中文测试数据结果（即xml中的label）。Parser.parse()传入的文件为待处理的文件名。

经过处理：

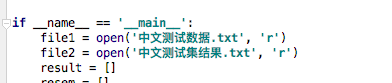
左图为分别存储训练集的句子对和标签结果

左图分别存储测试集的句子对和标签

3.关于模型建立部分

<第二阶段>

基于TF和TFIDF的句子余弦相似度计算(第二阶段(TF+TFIDF.py)))



file1是句子对的文本，file2是实际结果的文本。



在此处调用不同的函数：

cosine\_similarity\_tf(s1, s2):*#计算两个句子的TF余弦相似度*

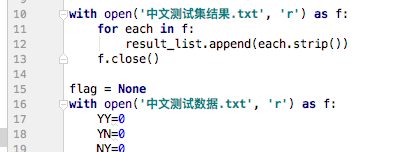
cosine\_similarity\_tfidf(s1, s2):*#计算两个句子的TFIDF余弦相似度*

通过不同方法计算句子相似度，预测结果将存入resem[]数组中，实际结果存在result[]中。程序最后会根据这两个数组计算F值。当前阈值t设定在0.5，由于正确率较低，调参价值太低，所以接下来的程序对结果进行了改进。



<第三阶段>

基于结巴的关键词权值计算(第三阶段(基于结巴的关键词提取).py)



两部分分别读取测试数据和测试集实际结果，由于计算两句子相同关键词的权值的标准差并根据标准差对句子进行相似度预测。所以这个阈值需要调整，也就是程序中的one。



阈值确定：

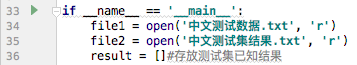
通过程序效率较快，所以采用循环的方式分别传入one值，多次调整后将one的上限设定在15，每次增加0.5，从而寻找最高正确率和最大F值的one值。





<第四阶段>

基于同义词林和TF余弦值计算的句子相似度计算(第四阶段(基于同义词+TF余弦).py)



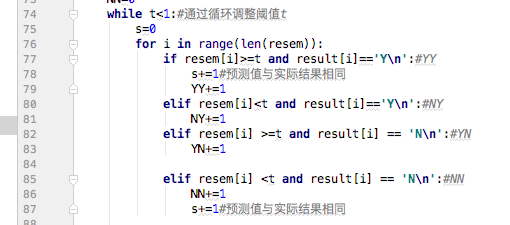
与前几个阶段相同，打开目标文件并存储。

阈值的确定：



此处阈值设定在了0.9，这个阈值表示在对比两个词语的相似度时是否划为同义或者近义的分界线。经过循环确定。







此处阈值t，经过while循环确定，用于确定最终两句子语义是否相同的分界线。步长为0.02，上限为1，初始值应当大于0，因为从0开始的话，NN=0，在计算F值时会出现分母为0的情况。每次循环都分别计算测试集准确率和F值。

4.其他文件说明

（1）同义词林.txt 存储同义词林，其中文本存储的格式为“词语 词语意义个数 词语编码”

（2）同义词编码.txt 存储词林所有的编码方式

（3）基于同义词林的词语相似度比较(自己实现).py 根据同义词林的编码规则，自己实现的词语间相似度对比。（由于这样需要每次都遍历txt，导致效率较低，所以后来改用Word\_Similarity模块）