

班级

学号

姓名

时 间

智能

201900161140

张文浩

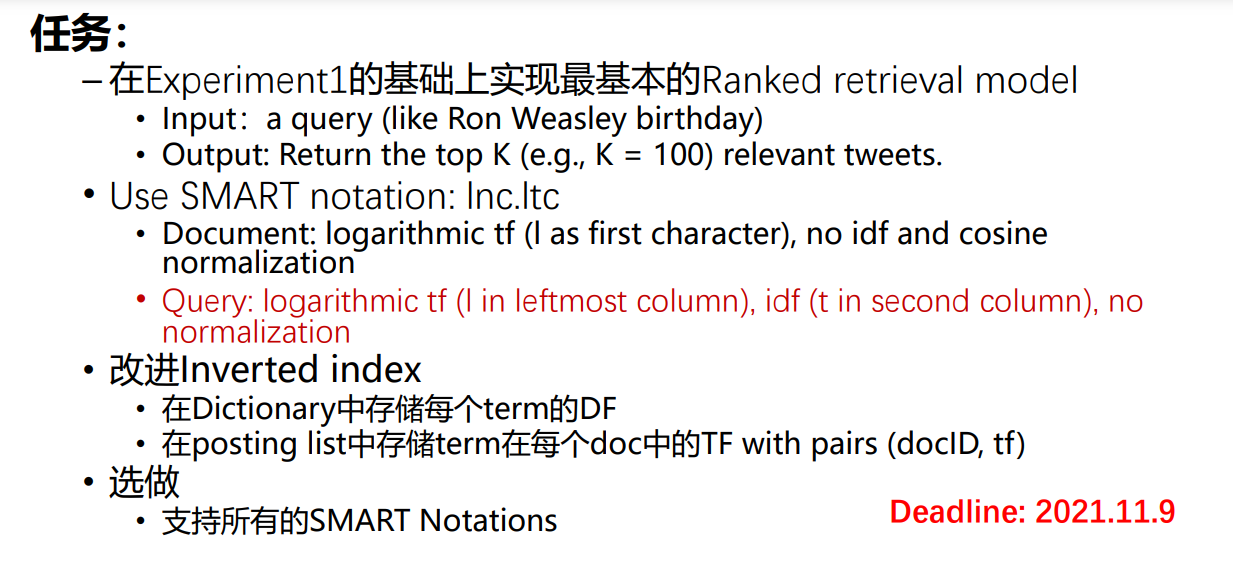
2021年9月29日

实验报告-实验二

信息检索与数据挖掘

实验内容

# Ranked retrieval model

****

# 实验环境：

win10+spyder（anaconda）

实验步骤

我实现了实验选做部分。即支持所有的smart Notations。

**第一步：**

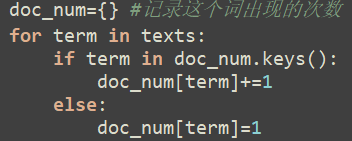
先读取一遍数据集tweets.txt来获得df0数组，这个df0数组的作用是为了后面在计算W的时候获得每个term对应的df用的。



**第二步：**

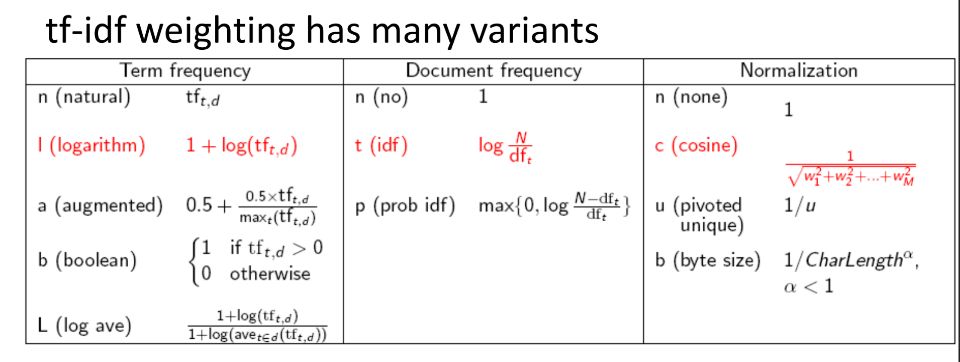
计算Wt\_d。一行一行读入数据，对文本进行预处理操作，与前面相同，这里不做赘述。经过预处理后得到了这一个文本的tweetid和这个文本里包含的单词的集合texts。

计算每个单词出现在当前document的次数

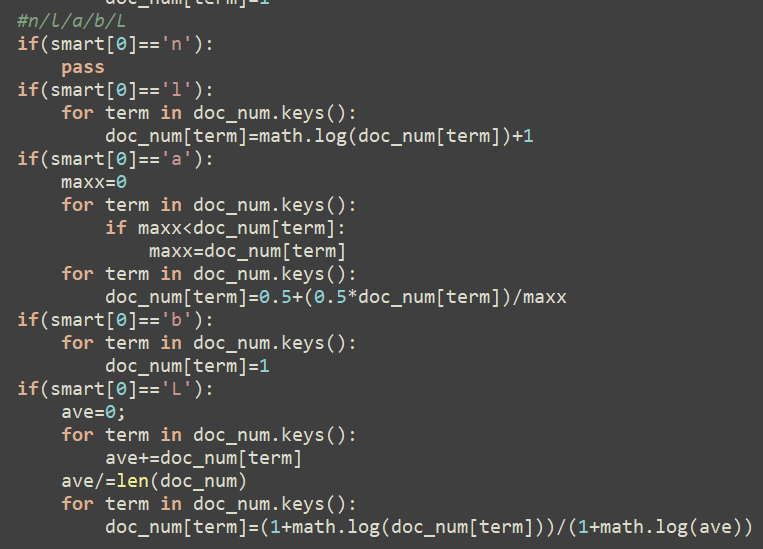


下面可以正式开始计算Wt\_d了

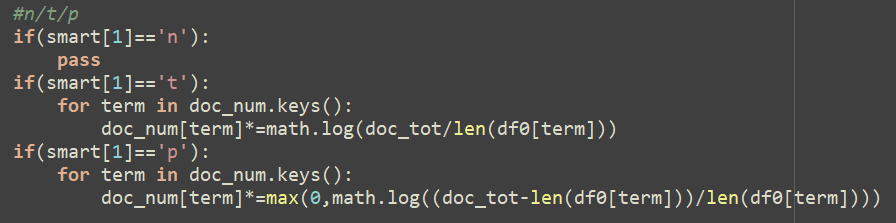
根据输入的smart判断用什么方式计算Wt\_d。参考下方表格



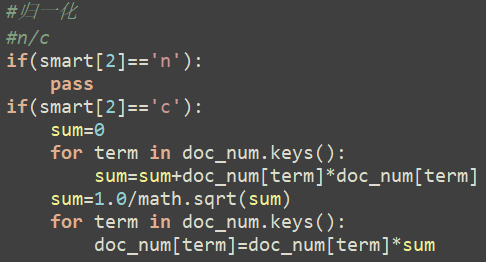
先计算Term frequency部分，如果输入的第1个字符是n就不用操作，直接使用出现次数tf即可；如果第一个字符是l，就是用logarithm的方法……



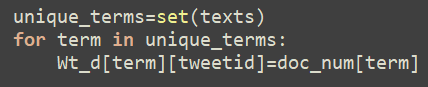
现在第一个term frequency部分计算完毕，下面计算document frequency部分，根据读入的smart字符串的第二个字符判断用什么方法加上document frequency部分的权重。计算方式根据表格填写即可，实现如下：



现在表格前两列都计算完毕，最后判断是否需要归一化，表格中normalization我只实现了前两种，即不归一化和cosine归一化，因为后两种没学过，也没大看懂，就没实现。



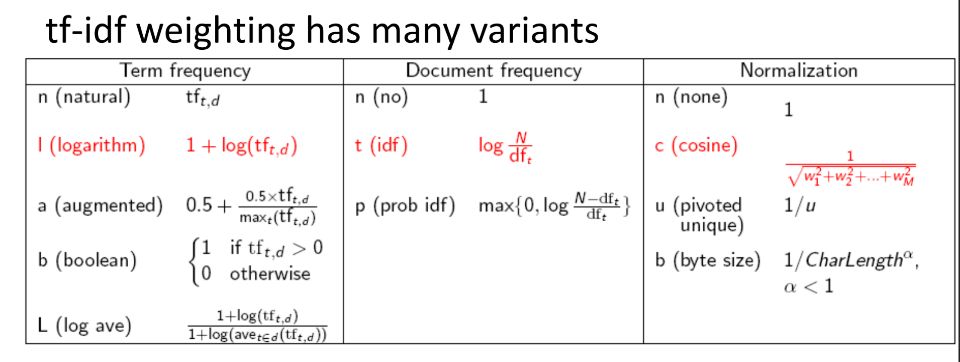
现在当前行的Wt\_d计算完毕，把结果存到Wt\_d字典中即可：



OK，到此为止，我们的Wt\_d就计算完毕了。下面一步可以计算Wt\_q了

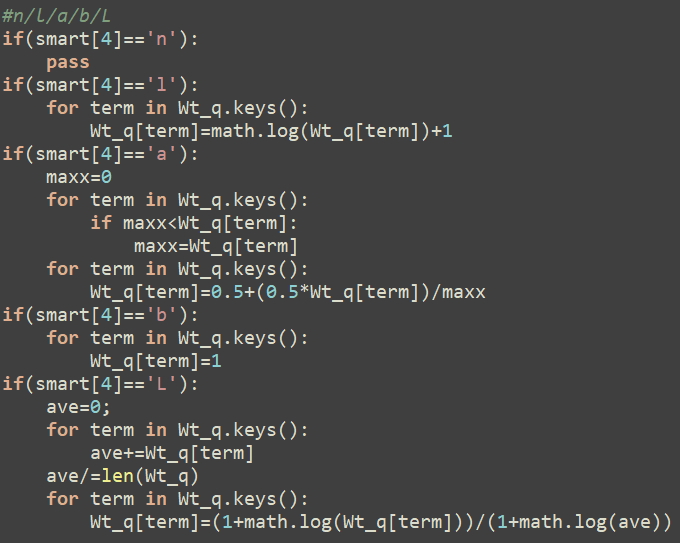
**第三步：**

计算Wt\_q

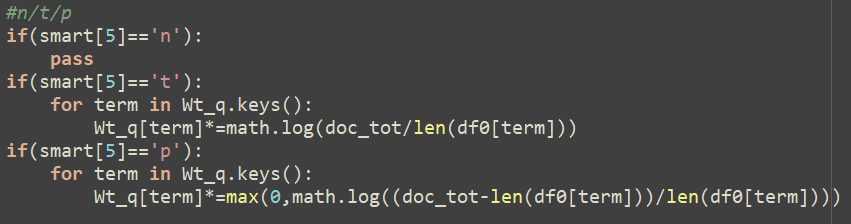


计算方式与Wt\_d完全相同，我再复述一遍。

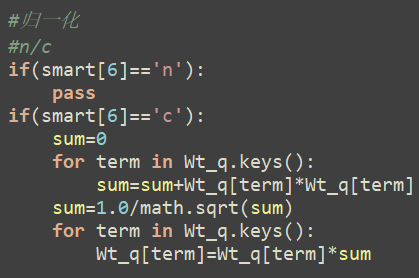
先算出每个term在query中出现的次数即tf，在根据输入的smart字符串的第5个字符判断计算term frequency的方法：



现在第一个term frequency部分计算完毕，下面计算document frequency部分，根据读入的smart字符串的第6个字符判断用什么方法加上document frequency部分的权重。计算方式根据表格填写即可，实现如下：



再根据smart的第7个字符判断是否需要对Wt\_d进行归一化操作。

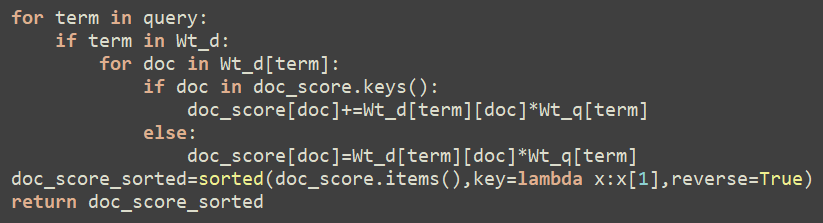


OK，现在Wt\_d和Wt\_q都有了，下一步可以计算score了

**第四步：**

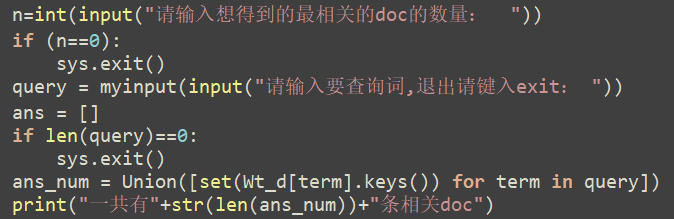
每个doc的score就是每个term的Wt\_d和Wt\_q乘积的和。实现比较简单。

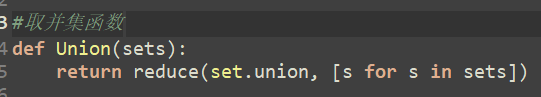
最后根据得分进行排序，方便下一步找到得分最大的k个document



**第五步：**

读取输入的query，进行与文本相同的预处理操作，然后根据Wt\_d查看有多少与这个term相关的document

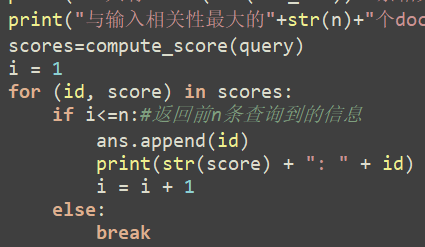


在对得到的相关的tweetid进行了一个取并集的操作，Union函数里面调用了reduce函数

reduce函数的作用和使用方法我放到实验总结里面了。

**第六步：**

调用函数，进行计算，最后输出score得分最高的n个tweetid即可。



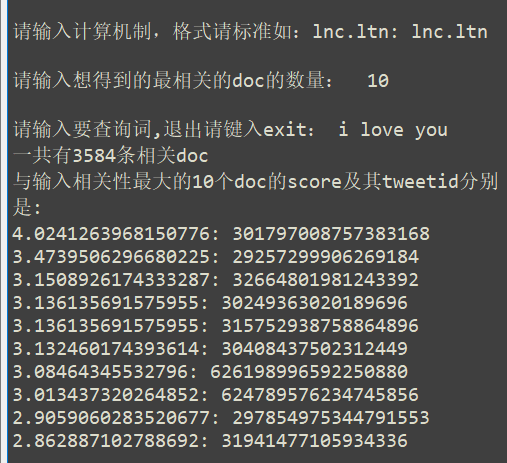
实验结果

测试：

使用计算机制lnc.ltn查询I love you

即document: logarithm tf 、 no idf 、 cosine normalization

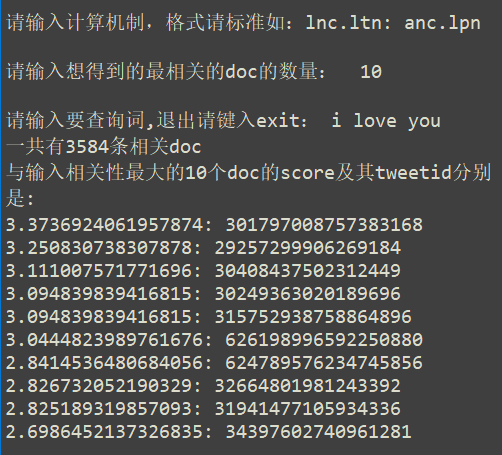
query: logarithm tf 、 idf 、 no cosine normalization



使用计算机制anc.lpn 查询I love you

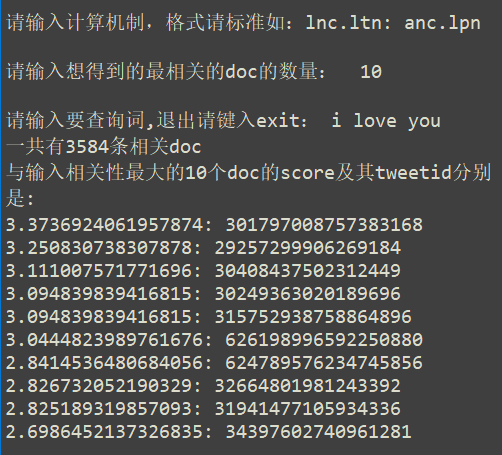
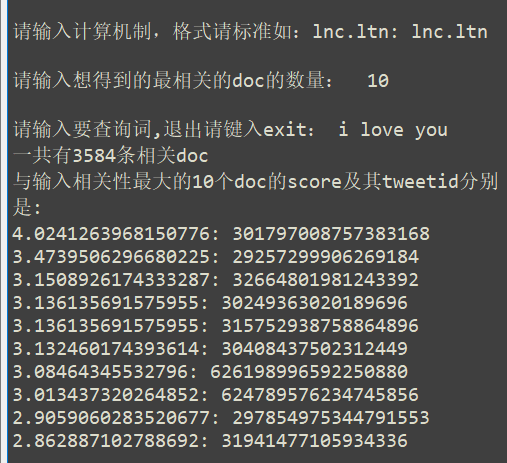
即document： augmented tf 、 no idf 、 cosine normalization

query： logarithm tf 、 prob idf 、 no cosine normalization



结果分析：

我们把使用不同计算机制的两个查询结果放到一起对比一下，看看查询出的得分最高的10个得分最高的document是否相同。



通过对比发现，两个方法得出来的得分最高的两个document虽然得分不同，但是排名相同。后面的排名就有些变化了，左边排名第3的doc到右边排名第8；左边排名第6的doc到右边排名第3；左边排名第7的doc到右边排名第6；左边排名第8的doc到右边排名第7；左边排名第9的doc到右边排名不在前10；左边排名第10的doc到右边排名第9。

根据实验结果我们可以得出结论，不同的计算机制计算出来的score不同，排名基本相同，但差别不大。所以在找相关度最高的document的时候，使用不同的相关度计算方法的结果可能会不同。

实验总结

在本次实验中，我实现了Ranked retrieval model。实践了课堂上学习的计算query查询与doc文本之间相关度的方法，对相关性的相关知识有了更加深刻的理解。同时，在实验过程中我也遇到了一些问题，并在解决问题的过程中学习到了新的知识。

比如在最后获得一共有多少个doc与query有关的数据的时候，用到了取并集的思想，在取并集的函数union中，用到了python库functools中的reduce函数，于是在网上查找了reduce函数的用法，

reduce(function, sequence[, initial]) -> value

reduce函数接受一个function和一串sequence，并返回单一的值，以如下方式计算：

1.初始，function被调用，并传入sequence的前两个items，计算得到result并返回

2.function继续被调用，并传入上一步中的result，和sequence种下一个item，计算得到result并返回。一直重复这个操作，直到sequence都被遍历完，返回最终结果。

我还了解了python中defaultdict和dict的区别

这里的defaultdict(function\_factory)构建的是一个类似dictionary的对象，其中keys的值，自行确定赋值，但是values的类型，是function\_factory的类实例，而且具有默认值。比如defaultdict(int)则创建一个类似dictionary对象，里面任何的values都是int的实例，而且就算是一个不存在的key, d[key] 也有一个默认值，这个默认值是int类型值0.。总之就是如果索引一个不存在的key，dict会报错，而defaultdict不会报错。

我完成了实验的选做部分，实现了所有的smart方法，并且通过比较使用不同的相关度计算方法查询相同的query，对比分析结果，发现不同的计算机制计算出来的score不同，排名基本相同，但差别不大。所以在找相关度最高的document的时候，使用不同的相关度计算方法的结果可能会不同。