信息检索与数据挖掘实验报告

基本信息:

姓名: 逯沐一

班级: 2018级数据班

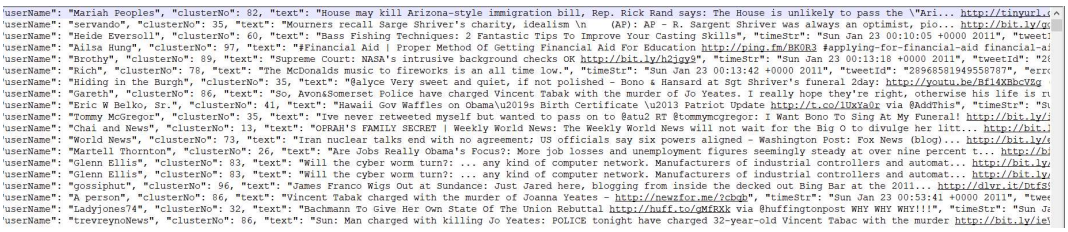
学号: 201800130020

实验内容: Ranked retrieval model

实验任务:

- 实现最基本的Ranked retrieval model
- Use SMART notation: lnc.ltn
- 改进Inverted index(加入DF与tf)
- 支持所有的SMART Notations(选做)

实验数据:(部分数据截图数据)



实验分析:

首先我们应该根据原数据整理出倒排索引，我们可以延续上一个实验以json的方式读取数据并进行预处理，但此时预处理需要增加项因为我们postingList的每一项都是(term,tf)的格式因此我们构建了一个新的类Term作为postingList的每一项的数据类型,我们只要遍历一遍文档就可以构建出倒排索引，这里仍然选择一个字典列表这种数据结构。

之后我们以lnc.ltc为例解释一下smart notation的工作原理，smart notation可以帮助我们给query和document的关系打分主要'.'之前的表示对文档的处理，后面的表示对查询的处理，第一个字母表示对tf的处理l表示log处理（1+log（tf) 如果tf为0则直接返回0），通过这

个操作可以的到tf-wght矩阵（tf的权重矩阵），第二个字母表示对df的处理n表示none(直接返回1), 而t则表示返回idf(log(N/tf))，这个处理后的结果就是idf,通过刚刚的tf-wght向量和idf向量我们通过对应项相乘可以的到初始权重矩阵（weight)，最后一个字母表示正则化的处理(n表示不做任何处理直接返回1,c表示cosine(权重向量平方和的倒数 如[1,2,3]就是1/(1*1+2*2+3*3))), 这里的正则化与前面的不太相同正则化仅仅返回一个值需要用这个值与权重向量的每一项相乘才可以得出最后的权重矩阵。最后我们将query的正则后的权重矩阵与文档正则后的权重矩阵进行点积就可以的到最后的打分(score)。

对于全部的smart notation我们可以通过查表得到对应的处理函数。

实验结果与分析:

实验结果如图(不填写smart notation按照默认的 lnc.ltn处理 这里输出了前10个)，后两幅图是我们使用测试数据的posting_list我们可以发现分数最高的是他们两个共有的可见我们的推荐是正确的 同时我们可以注意到后面的非零项score对应的id都是这两个单词的posting里的，但是我们同时注意到第二名居然不是剩下单词中出现次数最多的，实际上这也是正常的因为我们做了idf的处理因此会淡化某一个单词作为专业词 在某一个文档中反复出现对实验结果的影响。

```
Please input the query:ari rand

Please input the smart notation:
qualified tweets:
{'docId': '28965792812892160', 'score': 2.1912387236098616}
{'docId': '30933944996200448', 'score': 1.4227382630191494}
{'docId': '302231773549563904', 'score': 1.1820172492452092}
{'docId': '304176819094032385', 'score': 1.1559369375953736}
{'docId': '30826402152452097', 'score': 1.1020483197370465}
{'docId': '306546013664063488', 'score': 1.0507618444102387}
{'docId': '624439968421642240', 'score': 1.0455919047992899}
{'docId': '306923983373209601', 'score': 1.0060278736344104}
{'docId': '301835705401872384', 'score': 1.0027725943144015}
{'docId': '625935829207126017', 'score': 1.0019653818027248}
```

```
ari : (28965792812892160,1),(624439968421642240,1),(625935829207126017,1),
rand : (28965792812892160,1),(30826402152452097,1),(30933944996200448,1),(301835705401872384,1),(302231773549563904,1),(304176819094032385,2),
       (306546013664063488,1),(306668004991721472,1),(306794245132533760,1),(306923983373209601,1),
```

收获与反思:

这次实验在代码上的感觉是不算很复杂的，结合上一次的代码可以很快的写出这次的大体框架，因此难点在于对smart notation的理解，一开始的时候在很多地方上（函数处理 以及提供什么样的数据）有很多的 疑惑，但通过指导书上的实例和跟同学的交流逐渐的掌握了smart notation的精髓所在，最后也是比较顺利的完成了整个实验，并且可以支持多种smart notation的处理，总的来说这次的实验令我收获了很多。