Experiment-1

Inverted index and BooleanRetrieval Model

一、实验要求

1.构建invertedindex:

构建倒排索引是基于tweets数据集的。

我们首先观察该数据集，如下：


```
{"userName": "Mariah Peoples", "clusterNo": 82, "text": "House may kill
Arizona-style immigration bill, Rep. Rick Rand says: The House is unlikely to
pass the \"Ari... http://tinyurl.com/4jrjcdz", "timeStr": "Sun Jan 23 00:02:37
+0000 2011", "tweetId": "28965792812892160", "errorCode": "200",
"textCleaned": " ", "relevance": 2}
{"userName": "servando", "clusterNo": 35, "text": "Mourners recall Sarge
Shriver's charity, idealism \n    (AP): AP - R. Sargent Shriver was always an
optimist, pio... http://bit.ly/gqMcdG", "timeStr": "Sun Jan 23 00:07:48 +0000
2011", "tweetId": "28967095878287360", "errorCode": "200", "textCleaned": " ",
"relevance": 1}
{"userName": "Heide Eversoll", "clusterNo": 60, "text": "Bass Fishing
Techniques: 2 Fantastic Tips To Improve Your Casting Skills", "timeStr": "Sun
Jan 23 00:10:05 +0000 2011", "tweetId": "28967672074993664", "errorCode":
"200", "textCleaned": " ", "relevance": 2}
{"userName": "Ailsa Hung", "clusterNo": 97, "text": "#Financial Aid | Proper
Method Of Getting Financial Aid For Education http://ping.fm/BK0R3
#applying-for-financial-aid financial-aid-essay #", "timeStr": "Sun Jan 23
00:11:03 +0000 2011", "tweetId": "28967914417688576", "errorCode": "200",
"textCleaned": " ", "relevance": 2}
```


我们可以看到，tweets数据集中包含了许多信息，包括username，clusterNo，text，timestr,tweetid，errorcode，textcleaned，relevance。其中，我们需要提取的信息是text部分。

–1. 实现Boolean Retrieval Model，使用TREC 2014 test topics进行测试；

–2. Boolean Retrieval Model：

2.输入输出要求：

• **Input**： a query (like Ron and Weasley)

• **Output**: print the qualified tweets.

• **Query**：支持and, or ,not；查询优化可以选做；

3.注意：

•**预处理**：对于tweets与queries使用相同的预处理；

二、实验步骤

1.文本预处理

首先，我们定义一个函数，对每一行文本进行处理：对该行文本进行处理。

1.首先，将文本的每一个字符转换为小写，然后使用index函数，查找字符串中“text”字符和“timestr”字符的位置，从而定位出text文本在字符串中的位置。保存到新的字符串中。

2.其次，我们使用文本处理工具TextBlob对文本进行处理：
(1) 使用singularize函数对词汇进行单数化（复数转化为单数）

(2) 遍历每一个词汇，将动词转化为动词原形。

(3) 返回词汇列表。

2.创建倒排索引记录表

(1) 首先，逐行读取tweet文件，遍历每一行，并将每一行的每一个词汇存入一个集合中，即实现集合中词汇的去重；遍历集合中的每一个term，如果该term在全局字典中存在对应的keys，则对以该terms为keys的元组存在时，那么append该term存在的行号，(此处以行号作为每一个term的标识)，如果不存在，则创建对应的元组。

(2)记录文本对应的行数。并输出提示信息。

注意：postings词典 是在全局声明的。

3.创建基本的查询函数

(1) 对两个词项的and操作：

1)传入参数有两个，即要执行and操作的两个term。创建一个空的列表ans。

2)如果两个term都不在字典中，那么直接放回ans。

3)否则，获取以term1和term2为键值的列表，分别设定两个迭代器对列表进行遍历，如下：

```
        len1=len(postings[term1])
        len2=len(postings[term2])
        p1=0
        p2=0
        while p1<len1 and p2<len2:
            if postings[term1][p1] == postings[term2][p2]:
                ans.append(postings[term1][p1])
                p1+= 1
                p2+= 1
            elif postings[term1][p1] < postings[term2][p2]:
                p1 += 1
            else:
                p2+= 1
```

最终返回记录得到的ans列表。

(2) 对两个词项的or操作：

```
ans=[]
   if term1 not in postings and term2 not in postings:
       ans=[]#都不在为空
   elif term1 in postings and term2 not in postings:
       ans=postings[term1]
   elif term2 in postings and term1 not in postings:
       ans=postings[term2]
   else:
       ans=postings[term1]
       for item in postings[term2]:
           if item not in ans:
               ans.append(item)
```

(3) 对两个词项的not操作：

1)首先，创建一个全局列表，该列表是一个和文本数目相同的从1开始的列表。

2)然后，设计单独的函数，实现对单个词项的not操作，返回一个列表。如下：

```
    ans=[]
    if term not in postings:
        return ans
    # print("not")
    # print(all)
    # print("not")
    for item in all:
        if item not in postings[term]:
            ans.append(item)
    return ans
```

3)在单个此项取not的基础上，设计对两个词项之间not操作的函数。因为我们的not事实上指的是and not 的缩写，所以，此时，对应的处理和and操作类似，即相当于两个列表的合并。代码如下所示：

```
    if term1 not in postings or term2 not in postings:
        return ans
    else:
        ans1=postings[term1]
        ans2=signal_not(term2)
        len1=len(ans1)
        len2=len(ans2)
        i=0
        j=0
        while i<len1 and j < len2:
            if ans1[i] == ans2[j]:
                ans.append(ans1[i])
                i=i+1
                j=j+1
            elif ans1[i]<ans2[j]:
                i=i+1
            else:
                j=j+1
    return ans
```

4.输入文本处理及main函数书写

1) 对于输入文本，对term的处理方式同对文本的处理方式。

2) 对于词项的数目小于等于3时，做如下处理：

```
if terms==[]:
        print("your input is empty")
    if len(terms)==1:

        print(postings[terms[0]])
    elif len(terms)==2:
        print("sorry ,your input format is wrong!")
    #简单查询
    elif len(terms)==3:
        search_three_tuple(terms)

#词项数目等于3
def search_three_tuple(terms):
    global postings

    if len(terms)==3:
        answer = []
        # A and B
        if terms[1] == "and":
            answer = merge_and2(terms[0], terms[2])
            print(answer)
        # A or B
        elif terms[1] == "or":
            answer = merge_or2(terms[0], terms[2])
            print(answer)
        # A and (not) B   为方便处理，此处省略  and
        elif terms[1] == "not":
            answer = twice_not(terms[0], terms[2])
            print(answer)
        # 输入的三个词格式不对
        else:
            print("sorry ,your input format is wrong!")

```

5.改进

如果输入的文本包含了括号或者输入的查询大于3，那么仅仅通过以上函数就无法实现了，因此，我对输入的查询大于3的情况作了如下改进。

1) 此时，对输入文本的处理略有不同，应该保留括号等元素(之前的处理直接将特殊符号去除了)此处使用nltk中的分词工具进行处理，其他地方基本一致。

2) 如果查询中不包含括号，那么对应的操作就应该是基于两个term操作的基础之上的，对两个term进行and or not操作得到一个列表，对得到的列表和第三个至多个元素进行操作，即为列表的交并补集操作。如下：

```
elif '(' not in split_list and ')' not in split_list:
        for i in range(len(split_list)):
            #print(ans)

            if i%2!=0:

```

```
                continue
            #print(i)
            if i == 0:
                if split_list[i + 1] == 'and':
                    ans = merge_and2(split_list[i], split_list[i + 2])
                elif split_list[i + 1] == "or":
                    ans = merge_or2(split_list[i], split_list[i + 2])
                elif split_list[i + 1] == "not":
                    ans = twice_not(split_list[i], split_list[i + 2])
                #i+=2
            elif i>=len(split_list)-1:
                break
            else:
                if split_list[i + 2] not in postings:
                    ans=ans
                else:
                    if split_list[i + 1] == 'and':
                        #and 对应列表的交集
                        #print(postings[split_list[i + 2]])

                        #print(set(postings[split_list[i + 2]])&set(ans))
                        ans = sorted(list(set(ans) &
set(postings[split_list[i + 2]])))
                    elif split_list[i + 1] == "or":
                        #or对应列表相加
                        ans = ans + postings[split_list[i + 2]]
                        ans = sorted(list(set(ans)))
                    elif split_list[i + 1] == "not":
                        temp=[]
                        for i in all:
                            if i not in postings[split_list[i + 2]]:
                                temp.append(i)
                        ans = sorted(list(set(ans) & set(temp)))
                #i+=2

        return ans
```

3) 如果输入查询中包含了括号，暂时不做处理。（还未实现）

三、实验结果

1.单词项测试

```
预处理完成
文本数量共计： 30548 项
倒排索引记录表建立完成
Tips:
Model 1: you can input only one term;
Model 2: your input can include 'and','or' and 'not',each operator is a binary operator;

Please input your query:is
[1, 6, 8, 13, 21, 24, 25, 31, 36, 38, 43, 51, 56, 59, 63, 68, 77, 87, 97, 102, 112, 116, 117, 120, 123, 126, 132, 152,
 160, 163, 167, 172, 173, 174, 180, 190, 192, 193, 202, 205, 216, 217, 218, 221, 228, 231, 236, 237, 238, 246, 253, 255,
 259, 264, 274, 275, 276, 292, 295, 297, 303, 304, 305, 307, 314, 321, 322, 326, 330, 332, 333, 334, 338, 344, 351, 359,
 361, 365, 369, 371, 390, 397, 399, 415, 420, 421, 422, 440, 443, 450, 455, 457, 463, 466, 468, 473, 474, 477, 481, 487,
 495, 496, 506, 507, 509, 510, 513, 514, 517, 519, 520, 521, 525, 529, 539, 550, 566, 576, 578, 579, 580, 581, 590, 600,
 606, 611, 614, 616, 618, 621, 630, 631, 634, 637, 643, 662, 665, 666, 667, 676, 696, 699, 708, 710, 716, 719, 721, 729,
 730, 731, 734, 738, 747, 748, 757, 763, 770, 778, 785, 789, 792, 798, 801, 811, 834, 837, 839, 842, 853, 859, 861, 865
```

2.两个词项执行and操作

```
Please input your query:is and always
[332, 3579, 3583, 3996, 5497, 7115, 7771, 8135, 8369, 10032, 11806, 13649, 14208, 14214, 16015, 18362, 19047, 20744,
 21992, 22097, 22859, 23550, 24591, 24828, 27136, 27476, 28501, 28845, 29921]
```

3.两个词项执行or操作

```
Please input your query:sleep or always
[260, 511, 983, 1229, 5094, 6702, 8861, 10758, 11858, 11860, 12357, 15103, 16919, 16970, 17013, 17152, 17186, 17327,
 17332, 17413, 17582, 17602, 18483, 19044, 19050, 19051, 19052, 19239, 20885, 21844, 24987, 25851, 25912, 26032, 26188,
 26189, 26236, 27641, 2, 45, 83, 115, 332, 754, 1402, 1986, 2264, 3579, 3583, 3996, 4087, 4734, 4762, 4956, 5153, 5497,
 5641, 6860, 6967, 7021, 7115, 7771, 8135, 8313, 8369, 8488, 8729, 10032, 10923, 11018, 11806, 12977, 13649, 14208,
 14214, 14994, 15017, 15514, 16015, 16218, 18362, 18988, 19047, 20744, 21148, 21399, 21433, 21539, 21719, 21992, 22097,
 22210, 22739, 22859, 23147, 23533, 23550, 24591, 24747, 24828, 24838, 26772, 27136, 27476, 28501, 28503, 28845, 29921,
 30131, 30165, 30172]
```

4.两个词项执行not操作(and not)

```
Please input your query:always not is
[2, 45, 83, 115, 754, 1402, 1986, 2264, 4087, 4734, 4762, 4956, 5153, 5641, 6860, 6967, 7021, 8313, 8488, 8729, 10923,
 11018, 12977, 14994, 15017, 15514, 16218, 18988, 21148, 21399, 21433, 21539, 21719, 22210, 22739, 23147, 23533, 24747,
 24838, 26772, 28503, 30131, 30165, 30172]
```

5.多个词项执行and，or，not操作

```
Please input your query:is and always or simple
[574, 977, 1224, 1245, 2767, 3830, 4426, 4712, 5101, 5187, 5242, 7137, 7174, 16478, 20828, 24157, 25488, 25604, 27597]
Please input your query:is and simple not always
[977, 1245, 4426, 5101, 5242, 7137, 20828, 24157, 27597]
```

四、实验改进与不足

(1) 实验中没有将倒排索引记录表存储在文件中，每次动态建立倒排索引记录表；

(2) 实验中已经实现对于多个词项的布尔查询，但是还未实现对于含括号时的文本处理与检索分析；

(3)对于tweet数据集只保留了text信息，同时，对每一行文本使用集合set处理，丢失了词频的信息。