

Homework 3 Inverted index and Boolean Retrieval Model 实验报告

实验内容：

1. 在 tweets 数据集上构建 inverted index
2. 实现 Boolean Retrieval Model，使用 TREC 2014 test topics 进行测试

Boolean Retrieval Model:

- Input: a query (like Ron Weasley birthday)
 - Output: print a list of top k relevant tweets.
 - 支持 and, or, not
3. 查询优化

实验步骤：

1. 在 tweets 数据集上构建 inverted index

首先用 `json.load()` 方法读 json，获取每个 tweets 的文本，然后用 nltk 的 `word_tokenize` 进行分词。之后计算 inverted index，具体我是先用 set 统计每个词出现的文章，然后最后转 list 然后 sort，就得到了每个词出现在的文章的 id 的序列，而且已去重且有序。

```
def calc_inverted_index(articles):  
    # 计算inverted-index  
    inverted_index1 = {}  
    inverted_index2 = {}  
    for i in range(len(articles)):  
        for word in articles[i]:  
            if word not in inverted_index1.keys():  
                inverted_index1[word] = set()  
                inverted_index1[word].add(i)  
    for word in inverted_index1.keys():  
        inverted_index2[word] = list(inverted_index1[word])  
        inverted_index2[word].sort()  
    return inverted_index2
```

2. 实现 Boolean Retrieval Model

用 &, |, ! 来表示 and, or, not，按 not>and>or 的运算顺序。
and 实现两个序列的交：

```
def intersect(l1,l2):
    # 计算两个列表l1和l2的交集
    i = 0
    j = 0
    ans = []
    while i < len(l1) and j < len(l2):
        if l1[i] < l2[j]:
            i += 1
        elif l1[i] > l2[j]:
            j += 1
        else:
            ans.append(l1[i])
            i += 1
            j += 1
    return ans
```

or 实现两个序列的并:

```
def union(l1,l2):
    # 计算两个列表l1和l2的并集
    i = 0
    j = 0
    ans = []
    while i < len(l1) or j < len(l2):
        if i == len(l1):
            ans.append(l2[j])
            j += 1
        elif j == len(l2):
            ans.append(l1[i])
            i += 1
        elif l1[i] < l2[j]:
            ans.append(l1[i])
            i += 1
        elif l1[i] > l2[j]:
            ans.append(l2[j])
            j += 1
        else:
            ans.append(l1[i])
            i += 1
            j += 1
    return ans
```

not 实现一个序列的补:

```
def notto(L,num):
    l.append(num)
    ans = []
    j = 0
    for i in l:
        while j < i:
            ans.append(j)
            j += 1
        j += 1
    return ans
```

对布尔查询进行计算:

```
def calc(str,inverted_index,num):
    # 根据查询str和inverted-index返回查找结果
    arr = str.split('|')
    for i in range(len(arr)):
        arr[i] = arr[i].split('&')
        for j in range(len(arr[i])):
            arr[i][j] = arr[i][j].strip()
            if arr[i][j][0] == '!':
                arr[i][j] = arr[i][j][1:-1].strip()
                print(arr[i][j])
            if arr[i][j] in inverted_index.keys():
                arr[i][j] = notto(inverted_index[arr[i][j]],num)
            else:
                arr[i][j] = list(range(num))
        else:
            if arr[i][j] in inverted_index.keys():
                arr[i][j] = inverted_index[arr[i][j]]
            else:
                arr[i][j] = []
        arr[i] = and_all(arr[i])
    return or_all(arr)
```

3. 查询优化

在 and_all 和 or_all 方法里, 用一个优先队列维护未被处理的列表, 按长度从小到大, 每次取出两个最小的进行合并, 使得整体的合并更加快速。