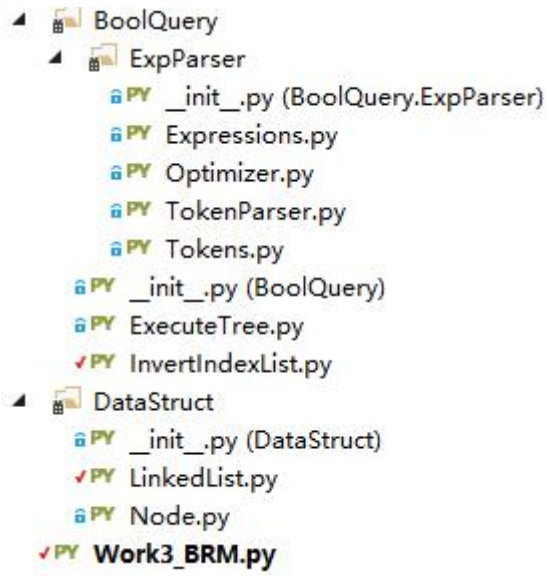
山东大学 计算机科学与技术 学院

信息检索与数据挖掘 课程实验报告

学号：201600130032	姓名： 陆禹承	班级： 智能 16
实验题目：布尔查询		
实验内容： 对输入文件进行处理，建立 Inverted Index List 用户输入查询语句，实现简单的查询功能		
实验过程中遇到和解决的问题： 一、基本程序结构 这次代码量偏多，尝试了 python 的模块和包。  布尔查询 BoolQuery 包： 包括一个输入分析 ExpParser 包 TokenParser 词法分析 Expressions 语法分析 ExecuteTree 用于执行语句		

InvertedIndexList 存储单词-文档序号

数据结构包：

包含一个链表和节点（链表节点和树节点）

Work3_BRM 用于用户交互，基本数据读写

二、存储结构

因为 python 没有链表，所以自己模拟了一个链表出来，这个链表类包含这些功能：

按照大小顺序插入，

基本运算（交、并、补）

因为都是有序链表，这些基本运算都是基于归并排序的原理上实现的。

单词表由 python 自带的字典实现，<单词，序号>

Inverted Index List 实现也很简单，就是对每个单词建立一个有序链表即可。

三、查询语句分析

首先定义语法：（A 和 B 为表达式，优先级高的先计算）

与运算（A 和 B 都要存在，优先级 2）：A&B

或运算（A 和 B 存在一个即可，优先级 1）：A|B

取反运算（不存在 A，优先级 3）：!A

允许使用括号改变优先级次序，比如 A & (B|C)&D

单词之间的与运算可以省略掉&，比如 that & me 简写为 that me

但是表达式之间的&不能省略。

（这个在词法分析的时候自动添加）

然后分析用户输入的指令，解析成 token

然后语法分析，这里不直接计算，而是生成一个计算树

这个计算树每个节点可以有多个孩子，

对于与（或）运算，多个孩子表示这多个孩子一起进行与（或）运算

非运算只有一个孩子

单词节点没有孩子

分析完毕后，将树放入优化器，简化：

多次 NOT 会自动抵消

将 AND 和 OR 运算能合并到一个节点的都合并起来

公共子表达式删除（没有实现）

实现时使用的是改进的逆波兰算法，可以很快解析出整个表达式

解析过程中附带优化（多个连续 AND 操作会合并），方便编写程序。

四、执行

递归计算节点，（这里计算结果都是一个文档列表）

单词节点直接向上返回值

NOT 节点直接将孩子的值取反并向上返回值

AND 和 OR 节点比较复杂：

首先先计算每个孩子节点，获得每个孩子的列表长度

优先计算 2 个最低长度的列表，反复直到只剩下一个列表（计算完毕）

计算时为了方便比较长度，用小根堆来实现

AND 运算如果遇到空列表，那么终止运算，直接返回空列表。

五、数据输入处理

因为数据全部都是 JSON，因此使用 JSON 库来解析文档

解析后的文档去除表情、网址、邮箱等信息，留下的单词全部小写。

然后建立起单词表和 Inverted Index

为了方便调试，生成结果会暂存。需要时会加载上次运算结果以节省时间。

六、结果

常规查询

```
Input command:
i do not like
----expression tree-----
  ||Tokens.ExpAnd4
    ||like
    ||not
    ||do
    ||i
=====result=====
Total:  [ 1]
[ 18064 ]
@LookingForKay I do not like buying clothes online because I could get the wrong size or something.
===============
```

比较复杂的查询

```
==============
Input command:
i me mine | he him himself| she her herself
----expression tree-----
  ||Tokens.ExpOr3
    ||Tokens.ExpAnd3
      ||himself
      ||him
      ||he
    ||Tokens.ExpAnd3
      ||mine
      ||me
      ||i
    ||Tokens.ExpAnd3
      ||herself
      ||her
      ||she
=====result=====
Total:  [ 1]
[ 6463 ]
Me too@thomaslynn: I'll be sporting mine will you? RT @chelsea_mac1: Wednesday, February 23 is Anti-Bullying Day http://
www.pinkshirtday.ca/
==============
```

非常简单的查询

```
================
Input command:
a
----expression tree-----
  ||a
=====result=====
Total:  [ 6397]
print all tweets? Y/n
n
================
```

极其复杂的查询（有半秒的停顿）

七、一些工程上的问题

Inverted Index 的序列化是手动存储的，因为直接序列化会导致 python 递归栈爆炸，调高递归栈限制则会使解释器爆炸。(可能是 node 链表递归的结果)

八、可能的优化

后期生成 linked list 非常缓慢，可以考虑分割成多个部分最后一起归并

对于 a AND NOT b 运算，可以直接运算而不需要 2 次运算

布尔表达式只是做了个简单的优化，可以使用 Quine-McCluskey 或其他启发式算法来优化布尔表达式。

多个 AND 或者 OR 同时计算可以大幅度提高速度

结论分析与体会：

经过本次实验，对布尔查询有了较深的理解