

# 《智能检索》实验报告

年级：2014 级

学号：22920142203809

姓名：冯立刚

## 目录

一、	实验内容 .....	2
二、	设计思路 .....	2
三、	具体实现 .....	2
1.	去除停用词 .....	2
2.	建立倒排索引表 .....	3
3.	编写搜索程序 .....	4
四、	系统运行测试 .....	5
1.	启动该系统，系统窗口会显示输入提示，等待用户输入： .....	5
2.	输入测试样例： .....	5
3.	关闭检索系统，在 result.txt 查看检索结果： .....	6
4.	系统鲁棒性测试： .....	7
5.	测试结果分析： .....	7
五、	实验总结 .....	8
六、	附录 .....	8

## 一、 实验内容

基于文本检索技术，编写一个快速文件检索程序，能够实现简单的 Bool 检索，具体要求如下：

1. 对于给定的数据文件进行检索，一共 532 个小文件；
2. 检索结果显示为包含关键词的文档列表的文件；
3. 显示从程序运行到结束所用的时间；
4. 支持多关键词检索（最多 3 个）；

程序的评价指标包括 3 个：

1. 检索时间；
2. 准确率；
3. 召回率；

## 二、 设计思路

1. 建立一张停用词表 stop.txt，去除原始文档集 Init\_C 中的停用词，得到去除停用词的文档集 Pre\_C；
2. 针对文档集 Pre\_C，建立关键词的倒排索引表 dictionary.txt；
3. 编写搜索程序 search.exe，针对用户输入的布尔表达式，检索满足查询表达式的文档，并将文档总数、文档名列表、搜索时间打印到 result.txt；

## 三、 具体实现

### 1. 去除停用词

- a) 从网上找了一个含 1100 多个词的停用词表，然后大致浏览了要检索文档，向停用词表添加停用词到大约 1293 多个词，得到最终的停用词表 stop.txt；
- b) 编写程序，遍历停用词表将其读入二维字符数组中，然后在每个文档中搜索这些停用词，将其删除，去除原始 txt 文档中的停用词，关键代码如下：

```
char stop[1293][30]; //1293个停用词
char buf[50];        //一行 50 个字符
//读入停用词
while (!feof(fstop))
{
    fscanf(fstop, "%s", stop[n]);
    n++;
}
fclose(fstop);
while (!feof(fs))
{
    fscanf(fs, "%s", buf);
    for (i = 0; i < n; i++)
```

```

    {
        if (strcmp(buf, stop[i]) == 0 || strlen(buf) <= 2)
        {
            x = 1;
            break;
        }
    }
    if (x == 1)
    {
        x = 0;
        continue;
    }
    else
        fprintf(fd, "%s\n", buf);
    memset(buf, '\0', sizeof(buf));
}

```

c) 通过上述处理，得到去除停用词的文档集；

## 2. 建立倒排索引表

a) 编写程序, 遍历去除停用词的文档集，将每个词语和每个词语的出现文档列表以“词语-文件 ID 序列”存储在一个 `map<string, vector<int>>` 里面，并且将其打印到 `result.txt` 中，得到倒排索引表 `dictionary.txt`, 重复词语的词汇表 `word.txt`，关键代码如下：

```

map <string, vector<int>> dictionary; //倒排索引表
while (cin >> id >> filepath) //从index中读入文件ID和文件路径名
{
    ifstream fin(filepath.c_str()); //打开文件路径下的文件
    string s;
    while (fin >> s) //一个词语一个词语读入
    {
        dictionary[s].push_back(id); //把当前单词对应的文件名加入到单词
        //对应的id数组中
        vector<int>::iterator itt; //去重
        itt = dictionary[s].end() - 1;
        if (*itt == *(itt-1))
        {
            dictionary[s].pop_back();
        }
    }
}

map<string, vector<int>>::iterator map_it;

```



```

map_it = dictionary.begin();
while (map_it != dictionary.end())//遍历整个索引表输出，因为map的键值
是严格若排序，因此输出的是字典序
{
    string tmp = map_it->first;
    cout << tmp << " ";
    for (int i = 0; i != dictionary[tmp].size(); i++)
        cout << dictionary[tmp][i] << " ";
    cout << endl;
    map_it++;
}

```

### 3. 编写搜索程序

- 程序启动之后，我们将 dictionary 装入 map<key, value>之中，待装入成功后，窗口显示输入提示字样，等待用户输入；
- 该搜索程序支持一个词语、两个词语“或”、两个词语“与”、三个词语“与”的四种布尔检索；

例：

- “嫣然”
- “嫣然 or 回想”
- “嫣然 and 回想”
- “双子座 and 嫣然 and 爱不释手”

- 用户输入后，程序对输入的字串做切分，根据切分词语中是否包含“or”、“and”以及词语的个数，选择进入不同的搜索通道；
- 在每个通道中，先判断这些词语是否存在于倒排索引表中，若存在，则详细查询这些词语的文档出现情况。针对多词的关联搜索，由于我们将词条的文档出现情况存储为向量 Vector 类型，其做交集、并集运算非常方便。得到这些词语的终极文档出现信息后，输出查询结果和查询时间；
- 该系统循环读入用户输入信息，当用户输入非法信息，或者词典中不存在该词语时，会给出“there are no these words!”、“你走~”的提示信息；当用户输入字符‘N’的时候，终止整个检索系统；
- 关键代码如下：

```

//string 字符串按空格切分函数，切分结果存储在 vec 中
void StringSplit(string s, vector<string>& vec)
//两词或搜索
void orsearch(vector<string>& str1, vector<string>& str2)
//两词与搜索
void andsearch(vector<string>& str1, vector<string>& str2)
//三词与搜索
void triandsearch(vector<string>& str1, vector<string>& str2,
vector<string>& str3)
//单个词语查询

```

```

void singleword(vector<string>& str)

    cout << "\n" << "请输入查询词语(多词查询用 "or" 或者 "and" 例："你好
and 同学"、“你好 or 同学”，“输入 ”N“ 结束查询)：" << endl;
    getline(cin, inputt);
    StringSplit(inputt, query); //输入 string 切分
    t_start = clock();
    //含一个词语
    if (query.size()==1)
    //含三个词语
    else if (query.size()==3)
    //含五个词语
    else if (query.size()==5)
    else
    {
        cout << "你走~！" << endl;
    }

```

#### 四、 系统运行测试

测试样例：

- “艾森豪威尔”，“白兰”
- “艾森豪威尔 or 白兰”
- “艾森豪威尔 and 白兰”
- “艾森豪威尔 and 白兰 and 矮人”

1. 启动该系统，系统窗口会显示输入提示，等待用户输入：

2. 输入测试样例：

```
C:\Windows\system32\cmd.exe
请输入查询词语(多词查询用 "or" 或者 "and" 例: "你好 and 同学"、"你好 or 同学", "输入 "N"
结束查询): 艾森豪威尔
7个文件
请输入查询词语(多词查询用 "or" 或者 "and" 例: "你好 and 同学"、"你好 or 同学", "输入 "N"
结束查询): 艾森豪威尔 and 白兰
1个文件
请输入查询词语(多词查询用 "or" 或者 "and" 例: "你好 and 同学"、"你好 or 同学", "输入 "N"
结束查询): 艾森豪威尔 or 白兰
15个文件
请输入查询词语(多词查询用 "or" 或者 "and" 例: "你好 and 同学"、"你好 or 同学", "输入 "N"
结束查询): 艾森豪威尔 or 白兰 or 矮人
1个文件
Output from Build
请输入查询词语(多词查询用 "or" 或者 "and" 例: "你好 and 同学"、"你好 or 同学", "输入 "N"
结束查询):
C:\proj -> E:\VisualStudioProject\search\x64\Release\search.exe
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====

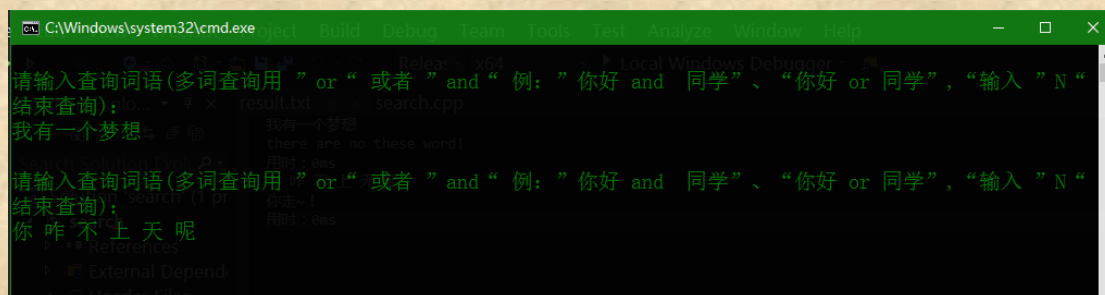
半:
```

3. 关闭检索系统, 在 result.txt 查看检索结果:

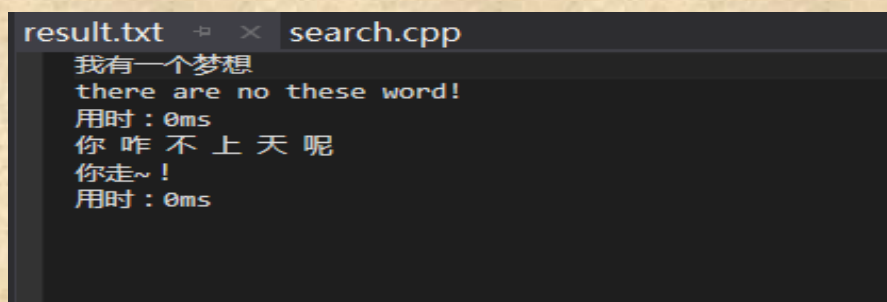
```
艾森豪威尔
data.fz.txt
data.hu.txt
data.zaim.txt
data.zdcy.txt
data.zdnc.txt
data.zenw.txt
data.zfaz.txt
用时: 0ms
艾森豪威尔 and 白兰
data.hu.txt
用时: 1ms
艾森豪威尔 or 白兰
data.fz.txt
data.hu.txt
data.zaim.txt
data.zalm.txt
data.zbcc.txt
data.zchd.txt
data.zcjd.txt
data.zdcy.txt
data.zdkp.txt
data.zdnc.txt
data.zefv.txt
data.zefw.txt
data.zenw.txt
data.zesq.txt
data.zfaz.txt
用时: 1ms
艾森豪威尔 and 白兰 and 矮人
data.hu.txt
用时: 1ms
```



#### 4. 系统鲁棒性测试:



```
C:\Windows\system32\cmd.exe
请输入查询词语(多词查询用 "or" 或者 "and" 例: "你好 and 同学"、"你好 or 同学", "输入 "N" 结束查询):
我有一个梦想
there are no these word!
用时: 0ms
请输入查询词语(多词查询用 "or" 或者 "and" 例: "你好 and 同学"、"你好 or 同学", "输入 "N" 结束查询):
你咋不上天呢
你走~!
用时: 0ms
```



```
result.txt  search.cpp
我有一个梦想
there are no these word!
用时: 0ms
你咋不上天呢
你走~!
用时: 0ms
```

#### 5. 测试结果分析:

i. 测试随机选取了词汇表中的几个单词, 对该检索系统的四种检索式: 单个词语、两个词语“与”, 两个词语“或”, 三个词语“与”进行了简单测试;

ii. 测试结果准确有效, 检索时间显示为 0ms, 1ms, 平均 1ms 左右。在平常使用中也会出现极少数 2ms, 3ms 的检索时间, 尚未出现过 3ms 以上检索时间。此外, 我们发现该检索系统检索时间与硬件平台相关, 比如其在一台游戏本上, 其检索时间几乎全部显示为 0ms;

iii. 该系统相对稳定, 鲁棒性较高, 交互体验良好。当用户启动该系统时, 系统窗口显示输入提示, 提示用户使用方法, 避免用户误用系统。当用户输入的表达式查询没有结果时, 会提示“there are no these words!”。当用户输入非法字符串时, 会提示“你走~!”。而且该系统采取循环输入, 当用户想要退出系统时, 可输入字符“N”退出系统。

综上, 该检索系统功能多样, 检索结果准确有效, 响应迅速且稳定高效, 交互设计优良, 是一个相对不错的检索系统。

## 五、 实验总结

经过三周紧凑的检索技术学习，在掌握基础检索理论的基础上，完成这样一个相对较难的实验，对我的编程能力来说还真是不小的挑战。

编写检索系统的过程中，对总体流程的设计我还是没有多大问题的，但是最难的就是实现上面了。目前，相对而言我最熟悉就是 C 语言了，C++ 仅限于能够应付考试，对大规模数据集处理最擅长的就是遍历了。然后我就兴致勃勃地开始写代码了，代码写好之后运行，接着就是漫长的等待……等待……再等待，剔除停用词两个小时，建立词汇表两天，程序就在哪儿一直跑着，看着都心累。

最后，终于建立了词汇表，内心无比激动。然而，我再也不想要这种龟速了，听说有的同学使用了 C++ 的高级结构非常快，我便索性百度了，只叹相见恨晚。然后现学现用使用了 C++ 的 `map`、`vector`、`string`、等高级数据结构写了搜索程序，感觉 C++ 真是太神奇了！

当我觉得自己的系统已经完善的时候，去找老师检查，结果检索结果不完全。回来后检查，结果发现自己的词典残疾，检索结果怎么会准确。痛定思痛，然后又用 C++ 的高级结构写了建立倒排索引表的程序，建立一个倒排索引表分分钟搞定，C++ 真的太神了！替换新的词典之后，运行我的检索系统，效果甚好，终于完成了！

总之，我觉得《智能检索技术》这门院选修课非常专业，所有的院选修课都非常专业。这次检索系统实验难度中等偏上，设计思路灵活清晰，我从开始到结束，真可谓“无所不用其极”，把文档拆了合、合了拆，程序放到多台电脑上跑，更不用说代码的调试增删了。整个过程我几次心碎，但痛并快乐着，当程序运行通过的那一刻，成就感爆棚啊！

## 六、 附录

源代码：



search.cpp



dictionary.cpp



del\_Stop.cpp



