

# 信息检索实验报告

## ——倒排索引

实验编号	一
实验名称	倒排索引
班级	软件学院 14 级二班
学号-姓名	刘嘉洋-1412620
实验日期	2016 年 9 月 23 日
至	2016 年 10 月 13 日

评分教师		实验报告成绩	
评分日期	年	月	日

## 一、实验目的：

- 了解信息检索原理的基本思路
- 掌握倒排索引的基础构建过程

## 二、实验环境

Ubuntu v16.04

Qt Creator 4.0.2

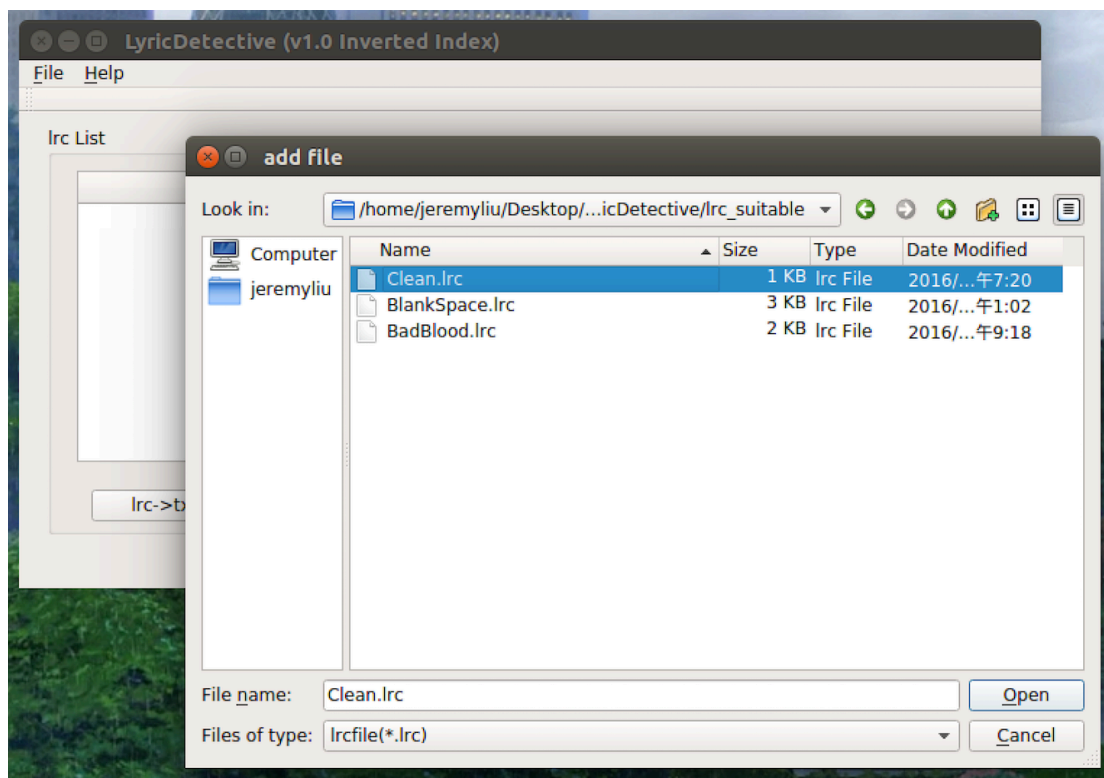
## 三、实验要求

- 编写程序实现为给定目录下 lrc(歌词)文件建立倒排索引文件
- 要求过滤掉 lrc 文件中的标识和时间标签

## 四、软件操作展示



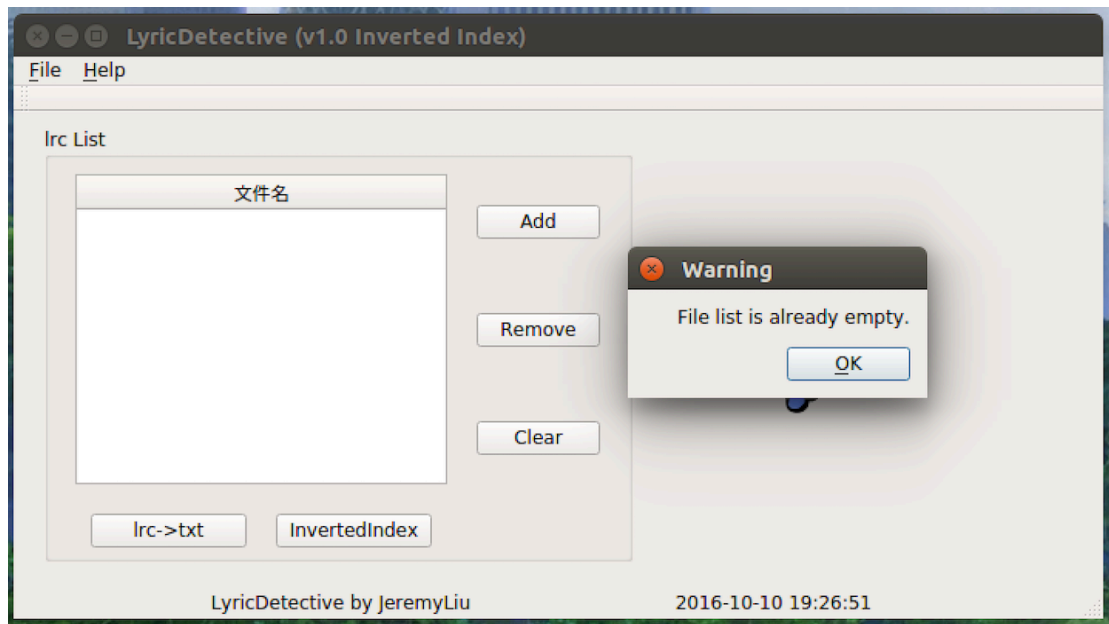
[图 1] LyricDetective“歌词探长”界面



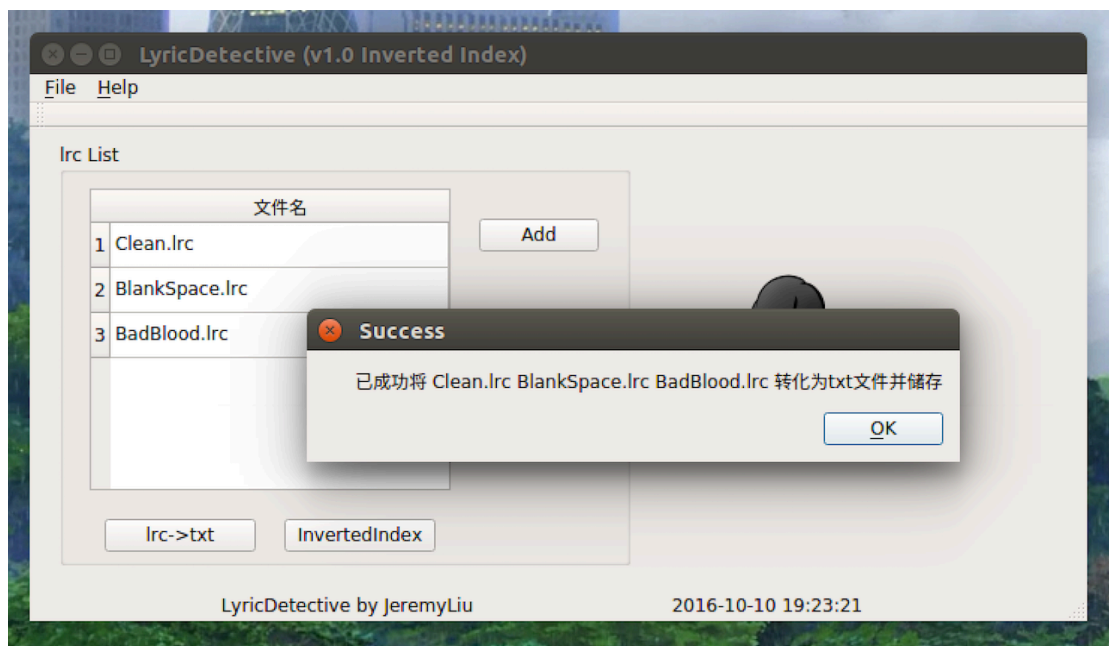
[图 2] 点击“ADD”按钮或通过 File 菜单中的“Add .lrc”显示添加歌词文件界面



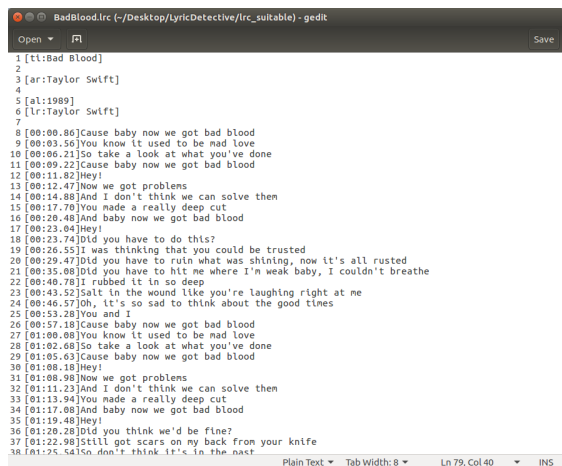
[图 3] 完成.lrc 文件列表创建，其间可进行添加、移除、清空操作



[图 4] 移除或清空后若再进行删除操作，会弹出提示信息

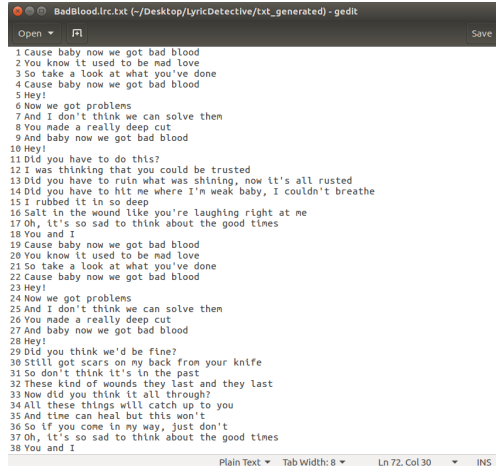


[图 5] 点击"Irc->txt"按钮，将当前歌词列表进行标识与时间标签分离



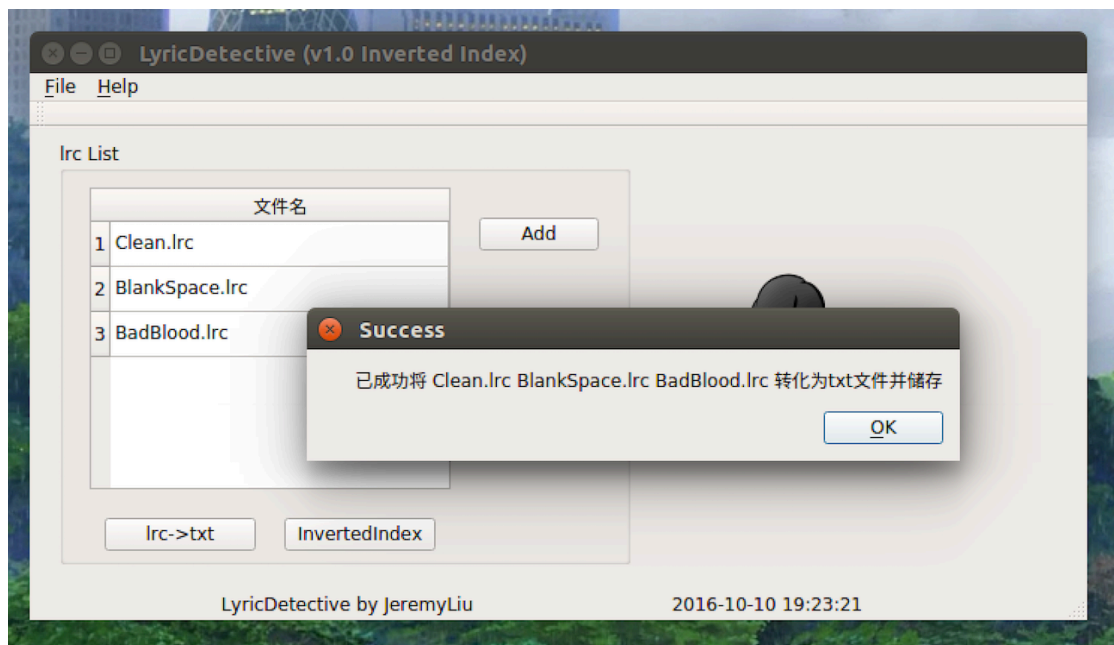
```
1 [tl:Bad Blood]
2
3 [ar:Taylor Swift]
4
5 [al:1989]
6 [lr:Taylor Swift]
7
8 [00:00.86]Cause baby now we got bad blood
9 [00:03.56]You know it used to be bad love
10 [00:06.21]So take a look at what you've done
11 [00:09.22]Cause baby now we got bad blood
12 [00:11.82]Hey!
13 [00:12.47]Now we got problems
14 [00:14.88]And I don't think we can solve then
15 [00:17.78]You made a really deep cut
16 [00:20.48]And baby now we got bad blood
17 [00:23.04]Hey!
18 [00:23.74]Did you have to do this?
19 [00:26.55]I was thinking that you could be trusted
20 [00:29.47]Did you have to ruin what was shining, now it's all rusted
21 [00:35.08]Did you have to hit me where I'm weak baby, I couldn't breathe
22 [00:40.78]I rubbed it in so deep
23 [00:43.52]Salt in the wound like you're laughing right at me
24 [00:46.57]Oh, it's so sad to think about the good times
25 [00:53.28]You and I
26 [00:57.18]Cause baby now we got bad blood
27 [01:00.08]You know it used to be bad love
28 [01:02.68]So take a look at what you've done
29 [01:05.63]Cause baby now we got bad blood
30 [01:08.18]Hey!
31 [01:08.98]Now we got problems
32 [01:11.23]And I don't think we can solve then
33 [01:13.94]You made a really deep cut
34 [01:17.08]And baby now we got bad blood
35 [01:19.48]Hey!
36 [01:20.28]Did you think we'd be fine?
37 [01:22.98]Still got scars on my back from your knife
38 [01:25.54]So don't think it's in the past
```

[图 6.a] 原.lrc 文件格式样例



```
1 Cause baby now we got bad blood
2 You know it used to be bad love
3 So take a look at what you've done
4 Cause baby now we got bad blood
5 Hey!
6 Now we got problems
7 And I don't think we can solve then
8 You made a really deep cut
9 And baby now we got bad blood
10 Hey!
11 Did you have to do this?
12 I was thinking that you could be trusted
13 Did you have to ruin what was shining, now it's all rusted
14 Did you have to hit me where I'm weak baby, I couldn't breathe
15 I rubbed it in so deep
16 Salt in the wound like you're laughing right at me
17 Oh, it's so sad to think about the good times
18 You and I
19 Cause baby now we got bad blood
20 You know it used to be bad love
21 So take a look at what you've done
22 Cause baby now we got bad blood
23 Hey!
24 Now we got problems
25 And I don't think we can solve then
26 You made a really deep cut
27 And baby now we got bad blood
28 Hey!
29 Did you think we'd be fine?
30 Still got scars on my back from your knife
31 So don't think it's in the past
32 These kind of wounds they last and they last
33 Now did you think it all through?
34 All these things will catch up to you
35 And time can heal but this won't
36 So if you come in my way, just don't
37 Oh, it's so sad to think about the good times
38 You and I
```

[图 6.b] 分析过滤后生成对应.txt 样例



[图 7] 点击“InvertedIndex”按钮，得到生成的倒排索引结果输出到 result.txt 中

```
result.txt (~/Desktop/LyricDetective/txt_generated) - gedit
Open Save

1 " : [BlankSpace.lrc, 2] [Clean.lrc, 2]
2 "Oh : [BlankSpace.lrc, 3] [Clean.lrc, 28]
3 'Cause : [BlankSpace.lrc, 3] [Clean.lrc, 6]
4 - : [Clean.lrc, 1]
5 -sheery : [BlankSpace.lrc, 1] [Clean.lrc, 1]
6 10 : [Clean.lrc, 11]
7 Ain't : [BlankSpace.lrc, 2] [Clean.lrc, 6]
8 All : [BadBlood.lrc, 2] [BlankSpace.lrc, 1] [Clean.lrc, 4]
9 And : [BadBlood.lrc, 11] [BlankSpace.lrc, 10] [Clean.lrc, 10]
10 Band-aids : [BadBlood.lrc, 8] [BlankSpace.lrc, 1] [Clean.lrc, 9]
11 Be : [BlankSpace.lrc, 11] [Clean.lrc, 9]
12 Blank : [BlankSpace.lrc, 1] [Clean.lrc, 14]
13 Boy : [BlankSpace.lrc, 4] [Clean.lrc, 10]
14 But : [BlankSpace.lrc, 6] [Clean.lrc, 6]
15 Cause : [BadBlood.lrc, 10] [BlankSpace.lrc, 6] [Clean.lrc, 6]
16 Cherry : [BlankSpace.lrc, 2] [Clean.lrc, 10]
17 Clean : [Clean.lrc, 1]
18 Did : [BadBlood.lrc, 6] [BlankSpace.lrc, 1] [Clean.lrc, 9]
19 Don't : [BlankSpace.lrc, 10] [Clean.lrc, 4]
20 Find : [BlankSpace.lrc, 3] [Clean.lrc, 10]
21 God : [BlankSpace.lrc, 2] [Clean.lrc, 6]
22 Got : [BlankSpace.lrc, 8] [Clean.lrc, 9]
23 Grab : [BlankSpace.lrc, 1] [Clean.lrc, 10]
24 Hey! : [BadBlood.lrc, 8] [BlankSpace.lrc, 1] [Clean.lrc, 9]
25 I : [BadBlood.lrc, 10] [BlankSpace.lrc, 18] [Clean.lrc, 28]
26 I'll : [BlankSpace.lrc, 8] [Clean.lrc, 10]
27 I'm : [BadBlood.lrc, 1] [BlankSpace.lrc, 9] [Clean.lrc, 6]
28 If : [BadBlood.lrc, 1] [BlankSpace.lrc, 3] [Clean.lrc, 3]
29 It : [Clean.lrc, 2]
30 Just : [Clean.lrc, 19]
31 Keep : [BlankSpace.lrc, 9] [Clean.lrc, 9]
32 Let : [Clean.lrc, 6]
33 Love's : [BlankSpace.lrc, 2] [Clean.lrc, 6]
34 Magic : [BlankSpace.lrc, 2] [Clean.lrc, 9]
35 Nice : [BlankSpace.lrc, 1] [Clean.lrc, 1]
36 No : [BlankSpace.lrc, 1] [Clean.lrc, 6]
37 Now : [BadBlood.lrc, 6] [Clean.lrc, 6]
38 Oh : [BadBlood.lrc, 3] [BlankSpace.lrc, 4] [Clean.lrc, 3]

Saving file '/home/jeremyliu/Desktop/LyricDetective/txt_... Plain Text Tab Width: 8 Ln 342, Col 84 INS
```

[图 8] 倒排索引生成结果 result.txt

## 五、核心代码说明

该软件需要解决的核心问题有：

- QTableWidget 中.lrc 列表的增、删和清空
- .lrc 文件中标识和时间标签的过滤
- 倒排索引的生成
- 生成结果的存储数据结构

### 5.1 QTableWidget 中.lrc 列表的增、删和清空

首先在 initial() 中定义 QTableWidget 的属性：

```
35 ui->twgFileList->setColumnCount(1);
36 ui->twgFileList->setRowCount(0);
37 QStringList headers;
38 headers << "文件名";
39 ui->twgFileList->setHorizontalHeaderLabels(headers);
40 ui->twgFileList->horizontalHeader()->setStretchLastSection(true);
41 ui->twgFileList->setSelectionBehavior(QAbstractItemView::SelectRows); // 设置选择行为时每次选择一行
```

在主窗体头文件中定义歌词文件名列表存储形式：

```
QVector<QString> listFileName
```

在 ADD 按钮对应的信号槽出发函数中进行文件打开与选择操作，并将文件名 append 进容器中：

```
63 // 添加lrc文件
64 void MainWindow::on_btnAddFile_clicked()
65 {
66     QString file_full = "";
67     QString file_name = "";
68     QString file_path = "";
69     QString file_suffix = "";
70
71     QFileInfo fileinfo;
72
73     file_full = QFileDialog::getOpenFileName(this, "add file", ".", "lrcfile(*.lrc)"); // 打开文件限定在lrc格式
74
75     fileinfo = QFileInfo(file_full);
76     // 文件名
77     file_name = fileinfo.fileName();
78     // 文件后缀
79     file_suffix = fileinfo.suffix();
80     // 绝对路径
81     file_path = fileinfo.absolutePath();
82
83     qDebug() << "File name is: " << file_name;
84
85     listFileName.append(file_name);
86
87     // TableWidget全部删除
88     int iLen = ui->twgFileList->rowCount();
89     for(int i=0; i<iLen; i++)
90     {
91         ui->twgFileList->removeRow(0);
92     }
93
94     // 重新将信息显示在TableWidget中
95     for(int i = 0; i < listFileName.length(); i++)
96     {
97         ui->twgFileList->setRowCount(i+1);
98         QString content;
99         content = listFileName[i];
100         ui->twgFileList->setItem(i, 0, new QTableWidgetItem(content));
101     }
102 }
```

由此，Remove 和 Clear 按钮操作实现即转换为对该 QVector 容器的操作。

需要注意的是，每次操作的最后都需要进行在 TableWidget 中的重新显示。

## 5.2 .lrc 文件中标识和时间标签的过滤

这项功能的关键点在于对字符串的识别，而恰恰在此我被僵持住了很久，原因在于，QT 中的字符串类型 QString 并没有标准 C++ 11 中 string 的诸如 find\_first\_of(), 最终我选择在 QT 环境中引入 std 库进行形式为 "std::String"，虽然产生了 QString 与 std::string 之间多次的相互转换，但使得问题得以解决。过滤逻辑如下：

```

63 // 歌词说明 ti:歌曲名称 ar:演唱者 al: by: 制作单位
64 getLyricHeader(str);
65
66 int m, n, p = 0;
67 std::string timeTemp[MAX_LYRCE_REPEAT_NUM]; // 保存信息
68 m = str_string.find_first_of('[');
69 n = str_string.find_first_of(']');
70 while (m >= 0 && m <= str_string.length() && n >= 0 && n <= str_string.length() && str != "")
71 {
72     timeTemp[p] = str_string.substr(m + 1, n - 1);
73     p++;
74     str_string = str_string.substr(n + 1, str_string.length());
75     m = str_string.find_first_of('[');
76     n = str_string.find_first_of(']');
77 }
78
79 QString contentTemp = QString::fromStdString(str_string); // important
80
81 for (int i = 0; i < p; i++)
82 {
83     if (lineNum == 1)
84     {
85         // qDebug() << "";
86     }
87     lyric[lineNum].startTime = QString::fromStdString(timeTemp[i]);
88
89     if (timeTemp[i + 1] != "") // 连续的
90     {
91         lyric[lineNum].endTime = QString::fromStdString(timeTemp[i + 1]);
92         // qDebug() << QString::fromStdString(timeTemp[i]);
93     }
94
95     if (lineNum - 1 >= 0 && i == 0) // 设置上一个的endTime
96         lyric[lineNum - 1].endTime = lyric[lineNum].startTime;
97     lyric[lineNum].lineNum = lineNum;
98     lyric[lineNum].lyricContent = contentTemp;
99     lineNum++;
100 }

```

最后将过滤结果分别保存至与 .lrc 文件名一致的 .txt 文件中：

```

132 QString txt_FileName = "/home/jeremyliu/Desktop/LyricDetective/txt_generated/" + file_CurrentProcessed_lrc + ".txt";
133 QFile file(txt_FileName);
134
135 if(! file.open(QIODevice::WriteOnly | QIODevice::Text))
136 {
137     QMessageBox::warning(NULL, "Warning", "无法录入文件 " + file_CurrentProcessed_lrc + ".txt");
138     return;
139 } else {
140     QString content;
141     // 录入内容
142     for(int i = 0; i < lineNum; i++)
143     {
144         if (lyric[i].startTime != "")
145         {
146             content += lyric[i].lyricContent + "\n";
147         }
148     }
149
150     // 清空结构体数组 否则之后的输出文件中包含之前的内容
151     memset(&lyric, 0, sizeof(OneLineLyric)*200);
152
153     // 输出到文件 (后期加messagebox)
154     QTextStream in(&file);
155     in << content;
156     file.close();
157 }

```

### 5.3 倒排索引的生成以及生成结果的存储结构

在此处的方法进行了借鉴，出处为：

<http://blog.csdn.net/okiwilldoit/article/details/51362839>

但我在其基础上进行了优化，主要着力点为：



- 调整为归一化的 Map 存储结构，方便了最后将结果通过 Iterator 遍历输出的时间复杂度。
- 重新调整了各个分步功能的实现位置，减少了全局变量的数量，使代码逻辑进一步清晰化。

```

58     if(strlen(word) == 0)
59     {
60         return;
61     }
62     std::string word_str = word;
63     std::string file_name_str = file_CurrentProcessed_incerted.toStdString();
64     // qDebug() << QString::fromStdString(word) << ":" << QString::fromStdString(file_name_str);
65
66     std::map<std::string, std::map<std::string, int>::iterator> it;
67     it = result_index_.find(word_str);
68
69     if(it == result_index_.end()) // not found
70     {
71         file_word_count.insert(std::pair<std::string, int>(file_name_str,1));
72         result_index_[word_str] = file_word_count;
73     }
74     else
75     {
76         file_word_count = result_index_[word_str];
77         file_word_count[file_name_str] ++ ;
78         result_index_[word_str] = file_word_count;
79         // qDebug() << result_index_[word_str];
80     }

```

最终，将得到的嵌套 Map 结构的倒排索引结构通过遍历器输出至 result.txt 文件中：

```

85     QString txt_FileName = "/home/jeremyliu/Desktop/LyricDetective/txt_generated/result.txt"; // 指定文件路径和文件名
86     QFile file(txt_FileName);
87
88     // 遍历输出map 录入内容
89     std::map<std::string, std::map<std::string, int>::iterator> iter_outer;
90     std::map<std::string, int>::iterator iter_inner;
91
92     QString content;
93
94     for (iter_outer = result_index_.begin(); iter_outer != result_index_.end(); ++iter_outer) {
95         std::map<std::string, int> inner_map = result_index_[iter_outer->first];
96         content += QString::fromStdString(iter_outer->first) + "\t\t\t";
97         for (iter_inner = inner_map.begin(); iter_inner != inner_map.end(); ++iter_inner) {
98             content += "[" + QString::fromStdString(iter_inner->first) + ", " + QString::number(iter_inner->second,10) + "] ";
99         }
100        content += '\n';
101    }
102    // qDebug() <<"....."<<content;
103
104    if(! file.open(QIODevice::WriteOnly | QIODevice::Text))
105    {
106        QMessageBox::warning(NULL, "Warning", "无法录入文件 result.txt");
107        return;
108    } else {
109        // 输出到文件
110        QTextStream in(&file);
111        in << content;
112        file.close();
113
114        QMessageBox::about(NULL, "Success", "成功录入文件 result.txt");
115    }

```

至此，该软件实现倒排索引的 1.0 版本全部完成，后期将会在界面 lrc\_list 的 Group 右侧添加布尔查询的用户操作端，从而实现从 .lrc 的解析、倒排索引的生成以及最后布尔查询的具体应用。

## 六、写在最后

感谢温延龙老师课上清晰的讲解，是对我倒排索引细节的掌握以及顺利制作完成该软件的基础。