搜索引擎优化报告

题 目 名 称：搜索引擎

姓　　名：李杰

学　　号：32016070129

专业（方向）：计算机科学与技术

指 导 教 师：秦爱明老师

2018 年 12 月

目录

基于 Lucene 的搜索引擎优化

一 优化目的

1 了解 Lucene 的更多用法；

2 熟悉 Web 编程与网页制作方法；

3 使搜索内容更加理想化、全面化，增加用户体验。

二 优化内容

1 页面优化；

2 页码优化；

3 根据标题去重；

4 同义关键词推荐；

5 按时间返回搜索结果。

三 环境和仪器（软件、硬件）：

1 硬件环境：普通联网的 PC 机；

2 操作系统：Win10；

3 相关软件：JDK1.8、Eclipse、Tomcat8.5、lucene-core-2.9.4.jar、Bootstrap 框架、JQuery 框架、HashMap、同义词词林表。

四 优化详解

1 页面优化

1）搜索界面优化

为了使搜索界面在简洁的同时不失美观，引用了 Bootstrap 及 JQuery 框架对前端页面进行美化，做了一个简要的导航栏。搜索界面如图 1 所示。

图 1 搜索界面

在搜索界面还加了一个天气预报框，基于一个开源的 API，详细地址见技术参考。其包括定位地点，天气状况，气温等。主要代码如下：

```
<iframe scrolling="no"
    src="https://tianqiapi.com/api.php?style=td&skin=pitaya&
    color=FFFFFF&city=北京"frameborder="0" width="70"
height="32">
  </iframe>
```

2）显示界面优化

该界面主要显示用户搜索的结果，效果如图 2 所示。加了一个输入框，默认文本为用户搜索的关键词，用户也可在该页面输入关键字进行搜索。主要代码如下：

```
<%
  String parm = request.getParameter("keys");
  parm = new String(parm.getBytes("gbk"), "utf-8");
%>
  <a href="/SESearchEngine"><img class="logo" alt=""
src="<%=basePath%>images/logo01.png"/></a>
<form action="indexSearch.jsp" method="GET">
  <input class="keys" value="<%out.print(parm); %>" type="text"
name="keys">
  <input type="submit" value="搜  索" class="seaBtn">
</form>
```

还在原来的基础上加了一个发布日期，并提供给用户基于时间的降序搜索；也加一个版块用于关键字相似推荐，如搜索北京可返回首都等近义词的搜索结果。该两个功能见后面讲解。

图 2 显示界面

2 页码优化

翻页功能使用开源框架 kkpager，其 Github 地址及详解见技术参考。效果

如图 3 所示。

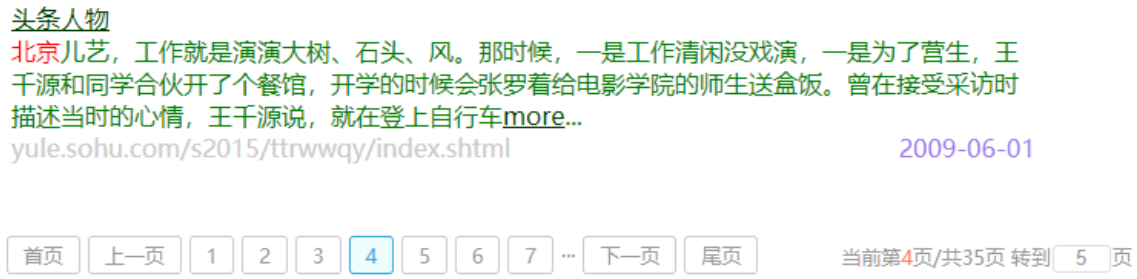

图 3 页码效果

其提供了上下页跳转，首尾页跳转，特定页跳转等。做法如下：

1）引入 JQuery 及 kkpager 框架：

```
<script type="text/javascript" src="<%=basePath%>js/jquery-1.10.2.min.js"></script>
<script type="text/javascript" src="<%=basePath%>js/kkpager/kkpager.js"></script>
<script type="text/javascript" src="<%=basePath%>js/index.js"></script>
<link rel="stylesheet" type="text/css" href="<%=basePath%>js/kkpager/kkpager_blue.css">
<link rel="stylesheet" type="text/css" href="<%=basePath%>css/search.css">
```

2）将总页数，总条数及请求参数传入函数，代码如下：

```
<script type="text/javascript">
page(<% out.print(length/10 + 1); %>, <% out.print(length); %>, "<%
out.print(parm); %>")
</script>
```

在将相关的链接前部及尾部传入，详情见项目代码。

3 根据标题去重

该功能主要基于 Java 的抽象接口 Map<K,V>，在相同键时覆盖原值，即可达到去重的目的。为了提高检索速度，使用实现类 HashMap，并将标题 title 作为键(key)，将 url，summary，time 封装为一个实体类 Info 并作为值(value)。最后遍历获得的 HashMap 即为去重后的结果。

1）实体类 Info 代码：

```
public class Info {
    private String url;
    private String summary;
    private String time;
    public Info(String url, String summary, String timeT) {
        this.url = url;
        this.summary = summary;
        if (timeT.contains("年")) {
```

```
                timeT = timeT.replace("年","-");
                timeT = timeT.replace("月","-");
            }
            if (timeT.length() < 9) {
                timeT = timeT+01;
            }
            this.time = timeT;
        }
        public String getUrl() {
            return url;
        }
        public void setUrl(String url) {
            this.url = url;
        }
        public String getSummary() {
            return summary;
        }
        public void setSummary(String summery) {
            this.summary = summery;
        }
        public String getTime() {
            return time;
        }
        public void setTime(String time) {
            this.time = time;
        }
}
```

2）封装为 HashMap 的主要代码：

```
    Map<String, Info> hMap = new HashMap<>();
    for(int i = 0; i < results.length; i++) {
            int position = results[i].indexOf("|");
            String title = results[i].substring(0, position);
            position = results[i].indexOf("mirror");
            String url = results[i].substring(position+7, results[i].indexOf("|",
position+1));
            position = results[i].indexOf("|", position+1);
            String context = results[i].substring(position+1);

            ……

            summary = context.substring(summaryStart, summaryEnd);
            position = results[i].indexOf("发布日期");
            String time = "";
            if(position > 1) {
                time = results[i].substring(position+5, position+15);
```

```
                time = time.trim();
            }
            Info info = new Info(url, summary, time);
            hMap.put(title, info);
        }

        ……
```

3)遍历 HashMap 输出去重结果：

```
        ……
        for(Entry<String, Info> info : hMap.entrySet()) {
        if(j < i) {
                j++;
                continue;
        }else{
                if(i == end)break;
                out.print("<a href=\"http://"+info.getValue().getUrl()+"\">"+"<font
color=\"#003c00\">"+info.getKey()+"</a>"+"</font><br/>");
                out.print("<font
color=\"#008000\">"+info.getValue().getSummary()+"<a
href=\"http://"+info.getValue().getUrl()+"\">"+"<font
color=\"#003c00\">more</a>"+"</font>...</font><br/>");
                out.print("<font
color=\"#cccccc\">"+info.getValue().getUrl()+"</font><font class=\"time\"
color=\"#9F79EE\">"+info.getValue().getTime()+"</font><br/><br/><br/>");
                i++;
            }
        }

        .......
```

4 同义关键词推荐

此功能主要基于《HIT－IRLab 同义词词林（扩展版）》。该词林共收录日常用语 10000 句，约 35 万词语，有 17817 行同义词语，并且有完整的编码格式。可以作为分词、词性标注、文本搜索研究等方面的标准训练语料。部分词林如图 4 所示。

获取到用户输入的关键词后，遍历该词林搜索该关键字并获得其同义词语，然后依次遍历同义词语并将其作为关键词搜索相应文档链接，去重后在可能的情况下选择 4 条的 title 与 url 作为同义词推荐给用户。当输入关键词老师时的相似推荐如图 5 所示。

```
17804   La04B01= 劳驾 劳神 劳动 烦劳 麻烦 烦 劳 累 辛苦 分神 费心 费神
17805   La04C01= 抱歉 对不起 对不住
17806   La05A01= 久仰 久仰大名 久慕盛名
17807   La05B01= 久违 少见
17808   La05C01= 怠慢 失敬 失礼 不周 怠
17809   La05C02= 失迎 有失远迎
17810   La05D01= 请便 听便 自便 悉听尊便
17811   La06A01= 恭喜 恭贺 贺喜
```

图 4 同义词林

图 5 老师关键词的相似推荐

1）遍历搜索同义词林并返回同义词主要代码如下：

```
public class SimilarityWord {
    //读取该类下的.txt 文档
    InputStream is=this.getClass().getResourceAsStream("WordTree.txt");
    ArrayList TYCCL = new ArrayList();
    InputStreamReader read1 = new InputStreamReader(is, "utf-8");
    BufferedReader TYCCL_br = new BufferedReader((read1));

    public SimilarityWord() throws IOException {
        String TYCCL_Line = null;
        for (; (TYCCL_Line = TYCCL_br.readLine()) != null; ) {
            String currentLineWords[] = TYCCL_Line.split(" ");
            TYCCL.add(currentLineWords);
        }
        TYCCL_br.close();
    }
    // 找同义词
    public ArrayList<String> findSimilarity(String ConceptA) throws
IOException {
        ArrayList<String> result    = new ArrayList<>();
```

```
            String CurrentTongyicicilinLine = null;
            Map map_ConceptA = new HashMap();
            Iterator itA = TYCCL.iterator();
            for(int j=0;j<17817;j++){
                String currentLineWords[] = (String []) itA.next();
                for (int i = 1; i < currentLineWords.length; i++) {
                    if (currentLineWords[i].equals(ConceptA)) {
                        if (currentLineWords.length > 6) {
                            for (i = 1; i < 6; i++) {
                                result.add(currentLineWords[i]);
                            }
                        }else {
                            for(i = 1; i < currentLineWords.length; i++){
                                result.add(currentLineWords[i]);
                            }
                        }
                        if (result.contains(ConceptA)) {
                            result.remove(ConceptA);
                        }
                        break;
                    }
                }
            }
            return result;
        }
    }
```

2）遍历相似词做搜索并作相似推荐的主要代码如下：

```
……
<%
ArrayList<String> simWords = new SimilarityWord().findSimilarity(keys);
            if(simWords.size() > 0) {
            // 遍历
            for(String simWord : simWords) {
                results = new QueryIndex().getQueryResult(simWord);
                if(results.length > 1) {
            Map<String, String> hashMap = new HashMap<String, String>();
                    for(String simInfo : results) {

                        ……

                                if(title.length() > 0) {
                                    hashMap.put(title, url);
                                }
                            }
```

```
        out.print("<div class=\"simkey\"><h5>key:"+simWord+"</h5></div>");
                                out.print("<ul class=\"simTab\">");
                                i = 0;
                                for(Entry<String, String> entry : hashMap.entrySet()) {
                                    if(i == 4)break;
                                    out.print("<li><a
href=\"http://"+entry.getValue()+"\">"+entry.getKey()+"</a></li>");
                                    i++;
                                }
                            out.print("</ul><br></br>");
                            }
                    }
                }
        %>
```

最后，将同义词搜索结果输出展示在界面即为相似词推荐。

5 按时间返回搜索结果

该功能主要是根据爬取的网页数据找出发布日期并保存在实体类 Info 里。因为在之前去重的时候是用 HashMap<String, Info>保存检索信息，故此处只需按时间降序 HashMap 数据即可。但因为 HashMap 是无序的且直接排序较难，故先将其作为对象保存在 ArrayList 列表里，在调用集合类 Collections.sort()方法并实现匿名内部类 Comparator 的 compare()方法来对数据排序，最后将时间转换为标准格式并比较返回有序列表即可。

1）按时间降序检索信息代码如下：

```
public class TimeSearch {
    public static List<Map.Entry<String,Info>> sort(Map<String, Info> map){
        List<Map.Entry<String, Info>> list = new
ArrayList<Map.Entry<String, Info>>(map.entrySet());
    Collections.sort(list, new Comparator<Map.Entry<String, Info>>() {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        public int compare(Entry<String, Info> o1, Entry<String, Info> o2) {
                try {
                    Date dt1 = format.parse(o1.getValue().getTime());
                    Date dt2 = format.parse(o2.getValue().getTime());
                    if (dt1.getTime() > dt2.getTime()) {
                        return -1;
                    } else if (dt1.getTime() < dt2.getTime()) {
                        return 1;
                    } else {
                        return 0;
                    }
```

```
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                    return 0;
                };
            });
            return list;
        }
}
```

2）当用户点击按时间检索时，调用排序好的信息列表并遍历返回给用户即可。主要代码如下：

```
Map<String, Info> hMap = new HashMap<>();
……
Info info = new Info(url, summary, time);
hMap.put(title, info);
……
List<Map.Entry<String, Info>> list = TimeSearch.sort(hMap);
……
```

当搜索信息为北京且用户点击按时间搜索时效果如图 6 所示。

图 5 老师关键词的相似推荐

五 结论与不足

通过本次搜索引擎优化实验，对搜索引擎有了更多的认识，不管是页面上的美化或是需求的满足，还是算法如去重，推荐检索，按时间、浏览量、匹配度等均有了更近一步的了解。但在 Url、文本相似度去重，亦或是模糊查询等还存在明显的不足，希望再接再厉。

六 技术参考

[1] 天气预报 API：http://www.tianqiapi.com/
[2] 开源翻页 kkpager：https://github.com/pgkk/kkpager