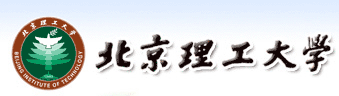
# JZSearch精准搜索引擎技术白皮书





[**http://www.nlpir.org/**](Http://hi.baidu.com/drkevinzhang/)

**2015-5**

**For the latest information about JZSearch, please visit Http://www.nlpir.org/**

Document Information

|  |  |  |  |
| --- | --- | --- | --- |
| Document ID | JZSearch-WHITEPAPER | Version | V1.0 |
| Security level | Public 公开 | Status | Creation and first draft for comment |
| Author | 张华平 | Date | May 31, 2015 |
| Publisher | / | Approved by |  |

Version History

Note：The first version is”v0.1”. Each subsequent version will add 0.1 to the exiting version. The version number should be updated only when there are significant changes, for example, changes made to reflect reviews. The first figure in the version 1.x denotes current review status by. 1. x denotes review process has passed round 1 etc .Anyone who create, review or modify the document should describe his action.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author/Reviewer | Date | Description |
| V1.0 | Kevin Zhang | 2011-8-21 | first complete draft for comment. |
| V1.1 | Kevin Zhang | 2011-9-15 | 增加了缓存机制的说明 |
| V2.0 | Kevin Zhang | 2011-12-5 |  |
| V2.5 | Kevin Zhang | 2012-1-5 | 增加可视化界面部分，及案例篇 |
| V2.6 | Kevin Zhang | 2012-5-24 | 增加多编码支持，增加情感扫描识别，新增1.5；新增3.1.12，修改4.2.1 修改4.3.1 |
| V2.7 | Kevin Zhang | 2012-5-28 | 增加了Complex运算符号 |
| V2.7 | Kevin Zhang | 2012-5-28 | 语法中：列出所有的索引内容 |
| V2.8 | Kevin Zhang | 2012-6-8 | 增加同义词联想功能 |
| V2.9 | Kevin Zhang | 2013-9-28 | 增加搜索聚类、比较功能，见3.1.17-3.1.19 |
| V3.0 |  | 2015-5-31 | 增加了Unique去重语句 |

# 目录

[JZSearch精准搜索引擎技术白皮书 1](#_Toc327357335)

[目录 3](#_Toc327357336)

[一、入门篇：了解基本知识 6](#_Toc327357337)

[1.1、JZSearch精准搜索引擎介绍 6](#_Toc327357338)

[1.2、JZSearch精准搜索引擎性能测试 7](#_Toc327357339)

[1.3、字段类型定义 8](#_Toc327357340)

[1.4、精准搜索排序策略 9](#_Toc327357341)

[1.5、支持的字符编码 9](#_Toc327357342)

[二、通用篇：五步配置搭建搜索服务 9](#_Toc327357343)

[2.1、采用可视化界面配置（适用于Windows,安装包比较大，需要可另外索取下载） 10](#_Toc327357344)

[2.1.1、运行star.Bat出现初始界面，配置数据库信息 10](#_Toc327357345)

[2.1.2、选择要建立索引的表，点击“执行”、“创建索引” 11](#_Toc327357346)

[2.1.3、弹出如下界面：配置好字段信息:点“创建索引文件” 11](#_Toc327357347)

[2.1.4:选择索引服务，然后选择项目，点击建立索引： 12](#_Toc327357348)

[2.1.5：索引建立成功之后，即可开启索引服务，测试索引： 13](#_Toc327357349)

[2.2、采用手工配置（适用于各种操作系统） 13](#_Toc327357350)

[2.2.1 第一步：配置数据库读取参数 14](#_Toc327357351)

[2.2.2 第二步：数据库字段信息列表导入 15](#_Toc327357352)

[2.2.3 第三步：建立字段信息文件 15](#_Toc327357353)

[2.2.4 第四步：自动建立索引 15](#_Toc327357354)

[2.2.5 第五步：启动搜索服务 15](#_Toc327357355)

[2.2.6 第六步：启动客户端服务 15](#_Toc327357356)

[2.2.7 其他脚本 15](#_Toc327357357)

[2.2.8 Linux环境 16](#_Toc327357358)

[三、进阶篇：了解内核 17](#_Toc327357359)

[3.1、JZSearch检索语法说明 17](#_Toc327357360)

[3.1.1 搜索与运算符AND 17](#_Toc327357361)

[3.1.2 搜索或运算符OR 17](#_Toc327357362)

[3.1.3 搜索非运算符NOT 18](#_Toc327357363)

[3.1.4 搜索邻近运算符NEAR 18](#_Toc327357364)

[3.1.5 搜索智能模糊运算符FUZZY 18](#_Toc327357365)

[3.1.6 搜索范围运算符RANG 18](#_Toc327357366)

[3.1.7 搜索最小值运算符MIN 19](#_Toc327357367)

[3.1.8 搜索最大值运算符MAX 19](#_Toc327357368)

[3.1.9 搜索前缀运算符PREFIX 19](#_Toc327357369)

[3.1.10 搜索精准运算符PRECISION 19](#_Toc327357370)

[3.1.11 搜索排序运算符 SORT 20](#_Toc327357371)

[3.1.12 复合运算符COMPLEX 20](#_Toc327357372)

[3.1.13 列出所有索引的结果内容LISTALL 21](#_Toc327357373)

[3.1.14 正负面情感搜索 21](#_Toc327357374)

[3.2 JZSearch搜索后台服务系统搭建 22](#_Toc327357375)

[3.3 JZSearch客户端搭建与管理指南 23](#_Toc327357376)

[3.3.1 客户端管理命令语法 23](#_Toc327357377)

[3.3.2 命令行方式 24](#_Toc327357378)

[3.3.3 C语言API方式 25](#_Toc327357379)

[3.3.4 JAVA语言调用搜索客户端 25](#_Toc327357380)

[四、高级篇：API定制开发 26](#_Toc327357381)

[4.1 字段定义接口 26](#_Toc327357382)

[4.1.1 JZIndexer\_FieldAdd 添加一个字段 26](#_Toc327357383)

[4.1.2 JZIndexer\_FieldSave 保存已经设置的字段信息 27](#_Toc327357384)

[4.1.3 JZIndexer\_FieldLoad 读取已经设置的字段信息文件 28](#_Toc327357385)

[4.2 索引接口 28](#_Toc327357386)

[4.2.1 JZIndexer\_Init 精准索引器初始化 28](#_Toc327357387)

[4.2.2 JZIndexer\_Exit 精准索引器系统退出 29](#_Toc327357388)

[4.2.3 CJZIndexer 精准索引器类 30](#_Toc327357389)

[4.2.3.1 CJZIndexer::CJZIndexer精准索引器类构建函数 31](#_Toc327357390)

[4.2.3.2 CJZIndexer::MemIndexing精准索引器类内存索引函数 32](#_Toc327357391)

[4.2.3.3 CJZIndexer::FileIndexing精准索引器类文件索引函数 33](#_Toc327357392)

[4.2.3.4 CJZIndexer::BigFileIndexing精准索引器类大文件索引函数 35](#_Toc327357393)

[4.2.3.5 CJZIndexer::IdIndexing精准索引器类ID索引函数 36](#_Toc327357394)

[4.2.3.6 CJZIndexer::IntIndexing精准索引器类整型索引函数 37](#_Toc327357395)

[4.2.3.7 CJZIndexer:: LongIndexing精准索引器类 64位长整型索引函数 38](#_Toc327357396)

[4.2.3.8 CJZIndexer::FloatIndexing精准索引器类浮点型索引函数 40](#_Toc327357397)

[4.2.3.9 CJZIndexer::AddDoc精准索引器类文档添加函数 41](#_Toc327357398)

[4.2.3.10 CJZIndexer::Save精准索引器类保存函数 42](#_Toc327357399)

[4.2.3.11 CJZIndexer::Merge精准索引器类索引合并函数 43](#_Toc327357400)

[4.2.3.12 CJZIndexer::Export精准索引器类索引导出函数 44](#_Toc327357401)

[4.3 检索接口 45](#_Toc327357402)

[4.3.1 JZSearch\_Init 精准搜索器初始化 45](#_Toc327357403)

[4.3.2 JZSearch\_Exit 精准搜索器系统退出 46](#_Toc327357404)

[4.3.3 JZSearch\_Reload 精准搜索器系统增量加载 47](#_Toc327357405)

[4.3.4 JZSearch\_Export 精准搜索器系统索引内容导出函数 48](#_Toc327357406)

[4.3.5 JZSearch\_Merge 精准搜索器系统索引归并优化函数 49](#_Toc327357407)

[4.3.6 搜索结果的数据记录结构 50](#_Toc327357408)

[4.3.7 CJZSearcher精准搜索器类 51](#_Toc327357409)

[4.3.7.1 CJZSearcher::CJZSearcher精准索引器类构建函数 52](#_Toc327357410)

[4.3.7.2 CJZSearcher::Search精准索引器类搜索函数 52](#_Toc327357411)

[4.3.7.3 CJZSearcher::Search精准索引器类搜索函数 53](#_Toc327357412)

[4.3.7.4 CJZSearcher::DocDelete精准索引器类索引文档删除函数 54](#_Toc327357413)

[4.4 利用JZSearch开发程序，搭建搜索引擎服务指南 55](#_Toc327357414)

[五、案例篇 56](#_Toc327357415)

[5.1 中国邮政集团名址信息中心首页的邮址垂直搜索 56](#_Toc327357416)

[5.2 河北标准化研究院的标准搜索 56](#_Toc327357417)

[5.3 中国对外承包工程商会的知识搜索门户 57](#_Toc327357418)

[5.4 富基融通的商品比价搜索 58](#_Toc327357419)

[5.5 WBK微博人物搜索 59](#_Toc327357420)

[六、问答篇：FAQ及小技巧 60](#_Toc327357421)

[6.1：环境问题 60](#_Toc327357422)

[6.1.1 JZSearch支持Linux吗？ 60](#_Toc327357423)

[6.1.2 Linux环境使用太不方便，有什么技巧？ 60](#_Toc327357424)

[6.2：索引问题 60](#_Toc327357425)

[6.2.1 在Windows下，针对MySQL数据库的索引老是创建不成功，访问不了数据库，为什么？ 60](#_Toc327357426)

[6.3：搜索问题 60](#_Toc327357427)

[6.3.1 老是搜索不到结果或者搜索到的结果老是不变的，怎么回事？ 60](#_Toc327357428)

[七、作者篇 61](#_Toc327357429)

# 入门篇：了解基本知识

## 1.1、JZSearch精准搜索引擎介绍

JZSearch精准搜索引擎由北理工副教授张华平博士精心设计，具有专业精准、高扩展性和高通用性的特点。可支持文本、数字、日期、字符串等各种数据类型的高效索引，支持丰富的查询语言和查询类型，支持少数民族语言的搜索。目前已经应用于中国邮政搜索引擎、河北省标准搜索引擎、富基融通（纳斯达克上市公司：EFUT）商品搜索。

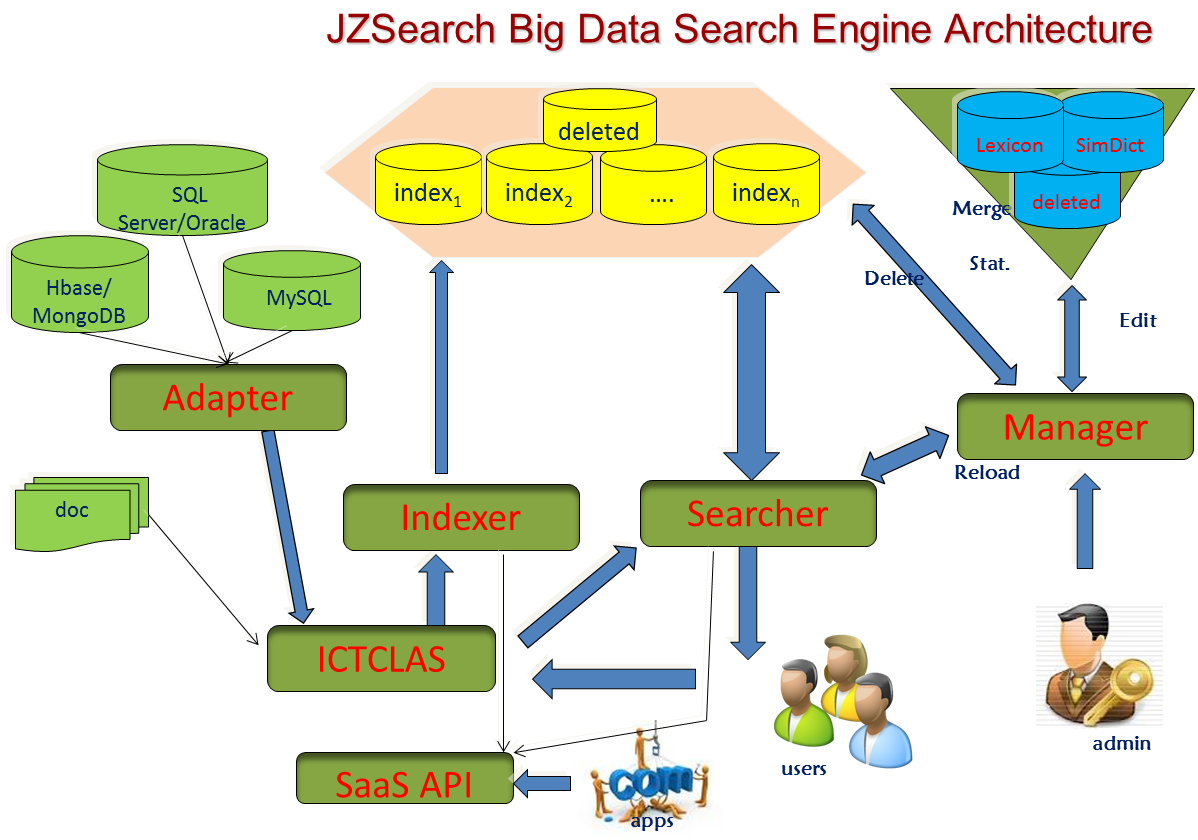
同时，全文搜索中间件通过可视化界面，可以快速地配置相关参数，启动搜索服务，并提供测试程序，可以无缝地与现有数据库系统融合，实现全文搜索与相关的数据库管理应用系统。

其主要特性包括：

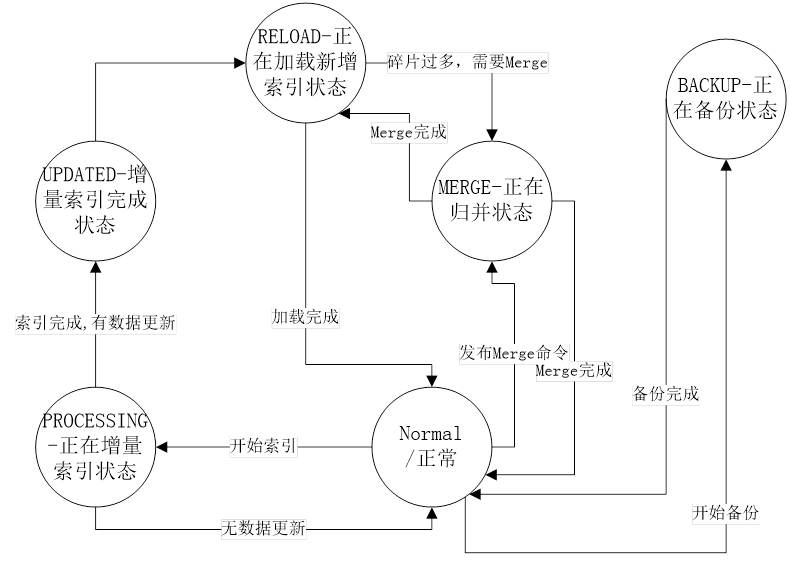
* 可以按照任意指定字段的排序，支持指定字段的搜索，也可以搜索多个字段，以及复杂表达式的综合搜索；
* 支持精确匹配以及模糊匹配，默认为精确匹配，忽略字母大小写进行模糊匹配；
* 内嵌正负面情感等极性分析，也可以支持类别搜索；
* 语义联想搜索：如搜索“马铃薯”可以同时返回“土豆”的内容，搜索“北京市”可以返回“北京”或者“首都”的内容；语义联系词表用户可以根据业务需要进行定制；
* 支持增量索引:系统可以在搜索服务不停的前提下，继续索引新的数据，索引完成后，可以搜索新的数据；
* 自动备份与恢复机制，在建立索引和自动优化之前，系统会将已有的索引文件自动备份；在当前索引文件被破坏无法搜索的前提下，系统将自动恢复上次搜索正常的备份文件；
* 自动缓存机制：系统自动保存最近常用的搜索条件与结果，再次搜索时将直接推送搜索结果内容，可以将搜索响应速度提升30%以上；缓存会随着新的索引数据自动更新，不存在缓存延迟问题；
* 自动优化机制：在系统索引碎片较多时，系统会自动优化归并；
* 实现的是多线程搜索服务；
* 兼容当前所有厂商的数据库系统，其中SQL Server, Oracle, MySQL，DB2等。
* 支持Windows/Linux/FreeBSD等操作系统，支持C/C++/C#/Java二次开发。

JZSearch大数据精准搜索引擎的技术架构如下图所示：

JZSearch大数据精准搜索引擎的技术架构如下图所示：



JZSearch大数据精准搜索引擎的状态转移图如下图所示：



## 1.2、JZSearch精准搜索引擎的优势与特色

下面，我们结合已经大规模公开应用的中国标准图书馆搜索（http://121.28.48.213:9999/byweb/search!list.action）为例，进行说明。

示例一：搜索“马铃薯”，系统会自动搜索出标准数据库里面的标题以及各类信息，左边会自动对搜索结果进行统计分析。

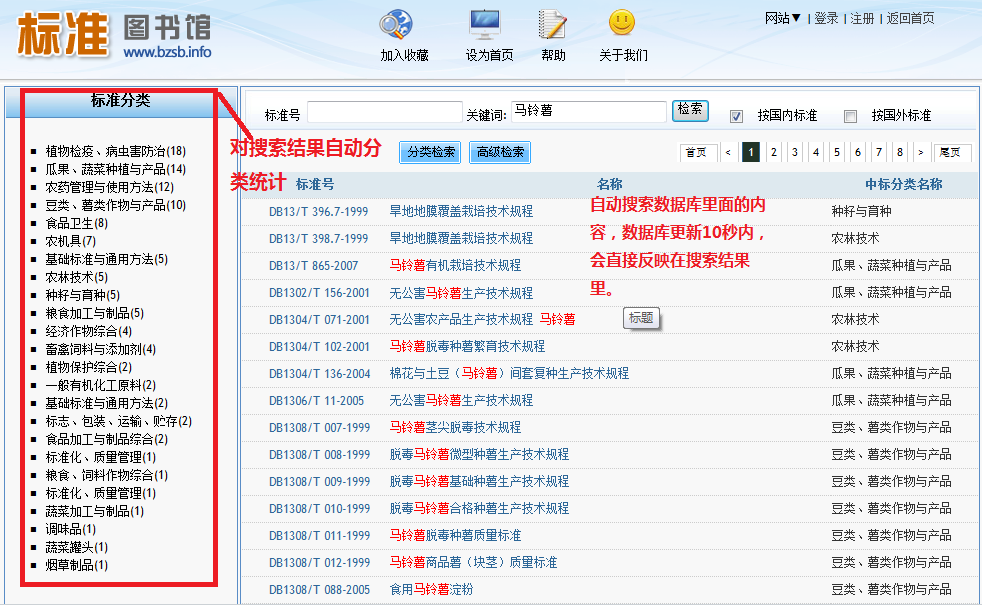


图1.语义搜索功能示例

示例二：高级搜索，JZSearch精准搜索引擎可以针对任意类型的字段进行各类综合搜索。复杂的搜索响应时间均在100毫秒左右。



示例三：搜索“土豆”，系统自动搜索出“马铃薯”等信息



示例四：搜索“GB 1003”，系统不会给出 GB 10030等不准确的结果。实现了精准的匹配与理解。



与传统的开源搜索引擎Lucence与Sphinx系统，以及TRS等已有的搜索厂商，通过以上的例子，我们不难看出JZSearch精准搜索引擎的优势与特色在于以下五点：

1. **语义精准搜索**：系统不是简单的关键词匹配，会自动根据语义知识进行联想，搜索用户真正需要的信息；并能实现数字、字母的精准搜索，而不是模糊匹配(搜索100，而不会给出1001等不当结果)。
2. **与业务无缝衔接**：JZSearch精准搜索引擎能够兼容现有的所有数据库，无需开发，即可实现与现有业务数据库的无缝衔接，实现任意字段的搜索分析，为数据库提供毫秒级的搜索服务；而新增或者修改的数据在10秒内，JZSearch搜索引擎就能捕获并反映在搜索结果中。非常适合企业业务数据的垂直搜索过滤。
3. **更强搜索功能**：JZSearch可以便利的搜索负面信息，对搜索结果自动统计分析，还可以实现指定文字范围内的搜索，如：搜索招标公告中的公司名称附件的法人信息，而不是泛泛地搜索，可以实现信息的精准定位。
4. **系统自主维护**：JZSearch搜索引擎可以处理TB级别的大数据业务，索引速度10MB/s，搜索响应1秒以内；整个过程无需人工干预，系统部署后，自动索引、自动搜索、自动增量更新、自动优化、自动备份与恢复，实现7\*24小时在线运行。
5. **各类成功案例**：目前，基于JZSearch公开运行的搜索引擎已经有3年多的历史，各类政府、企业及互联网的典型成功案例包括：中国邮政集团的邮政搜索引擎（所有数据2个亿，每年搜索次数超过1个亿）；中国证监会的舆情搜索服务（覆盖所有财经网站）；河北省标准化研究院的标准搜索引擎（同时在线服务10万家的企业）；纳斯达克上市公司富基融通的商品搜索引擎（服务中国70%的零售业）；缔元信网站统计数据搜索服务（每天新增数据500GB）。

## 1.3、JZSearch精准搜索引擎性能测试

|  |  |
| --- | --- |
| 项目 | 参数 |
| 测试机器配置 | ThinkPad x220i 内存 2G， Intel Core i3-2310M CPU @2.10GHz 2.10GHZ |
| 测试数据 | NLPIR新闻语料库（中文3,686,479KB 英文 2,249,607KB） |
| 使用内存 | 1GB |
| 占用CPU | 25% |
| 索引时间 | 1,115秒 |
| 索引速度 | 5,323KB/s |
| 搜索速度 | <1秒可控制在毫秒级别 |

注：测试在个人使用的笔记本电脑运行，在服务器端的速度可以超过10MB/s。

## 1.4、字段类型定义

JZSearch兼容当前基本的数据类型，如下表。

|  |  |  |
| --- | --- | --- |
| **类型名** | **类型值** | **说明** |
| FIELD\_TYPE\_TEXT | 1 | 文本类型 |
| FIELD\_TYPE\_INT | 2 | 整型（32位整型数据） |
| FIELD\_TYPE\_LONG | 3 | 长整型（64位整型数据） |
| FIELD\_TYPE\_DATETIME | 4 | 日期时间型（目前等同于FIELD\_TYPE\_INT） |
| FIELD\_TYPE\_FLOAT | 5 | 浮点型 |
| FIELD\_TYPE\_BIGTEXT | 6 | 存储在大文件中 |

示例如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 数据库字段名称 | 索引名称 | 数据类型 | 是否需要索引 | 是否需要存储原文 | 是否需要通配索引 | 是否需要摘要并红显 | 备注信息 |
| 1 | bigfilename@offset@length@e:\语料库资源\corpus\新闻语料库 | content | bigtext | 1 | 1 | 1 | 1 | 0 |
| 2 | id | id | text | 0 | 1 | 0 | 0 | 0 |
| 3 | title | title | text | 1 | 1 | 1 | 1 | 0 |
| 4 | url | url | text | 0 | 1 | 0 | 0 | 0 |
| 5 | publishtime | publishtime | date | 1 | 1 | 0 | 0 | 0 |
| 6 | click | click | int | 1 | 1 | 0 | 0 | 0 |

说明如下：

**1. 数据库字段名：**为数据库中的实际字段名称，必须一致，否则无法访问数据库；如不访问数据库，该字段可以不设置。

**2. 索引名：**为后续检索的字段名称，由用户自定义，必须和提交搜索的名称一致，否则解析错误，无法正确搜索；

**3. 数据类型:**支持数据库的常见格式，其中要求非字符型字段不用有索引，只用于过滤或者存储；参见数据字段类型表；

**4．是否需要索引：**表示的是搜索系统在索引过程中是否保留原文，并在检索结果中是否出现；

**5. 是否需要存储原文：**存储在文件中，方便搜索结果显示该字段；

**6. 是否为通配符索引：**在不指定字段名称进行统配搜索（搜索 \*）时，是否覆盖该字段。

**7. 是否需要摘要并红显：**在结果显示时，需要对该字段进行动态摘要，并红显。

**8.bigtext类型：**主要解决大文件存储多个小文本的情况，适合于海量文本的处理，对应数据库的时候，需要依次按照顺序指定如下信息：bigfilename@offset@length@e:\语料库资源\corpus\新闻语料库 分别对应的是 大文件名，偏移量，文本内容长度 即指定的文件路径名，中间采用“@”链接。注：路径名最好是绝对路径。

## 1.5、精准搜索排序策略

精准搜索引擎提供四种排序方式：

|  |  |  |
| --- | --- | --- |
| **类型名** | **类型值** | **说明** |
| docid | 1 | 按照docid顺序排序，即先建索引的文档优先 |
| relevance | 2 | 按照相关度排序 |
| reverse\_docid | 3 | 按照docid逆序排序，即后建索引的文档优先 |
| <field\_name>descend/ascend | 4 | 按照指定的数值型字段进行排序，同时可以指定从小到大，或者从大到小（默认为从大到小） |

## 1.6、支持的字符编码

|  |  |  |
| --- | --- | --- |
| **类型名** | **类型值** | **说明** |
| INDEX\_ENCODING\_GBK | 0 | 默认为GBK编码，其中只有简体字 |
| INDEX\_ENCODING\_UTF8 | 1 | UTF8编码 |
| INDEX\_ENCODING\_BIG5 | 2 | BIG5编码 |
| INDEX\_ENCODING\_GBK\_FANTI | 3 | GBK编码，但是简体字繁体字混杂，选择此项，系统将实现繁简体跨界搜索。 |

# 通用篇：五步配置搭建搜索服务

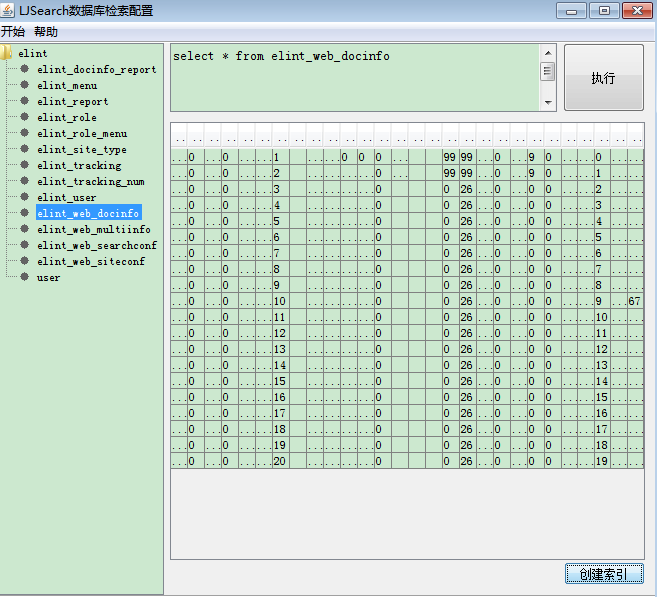
本篇介绍如何利用JZSearch精准搜索引擎工具包，不进行程序定制开发，搭建一套针对特定数据库的垂直搜索引擎服务。

## 2.1、采用可视化界面配置（适用于Windows,安装包比较大，需要可另外索取下载）

### 2.1.1、运行star.Bat出现初始界面，配置数据库信息



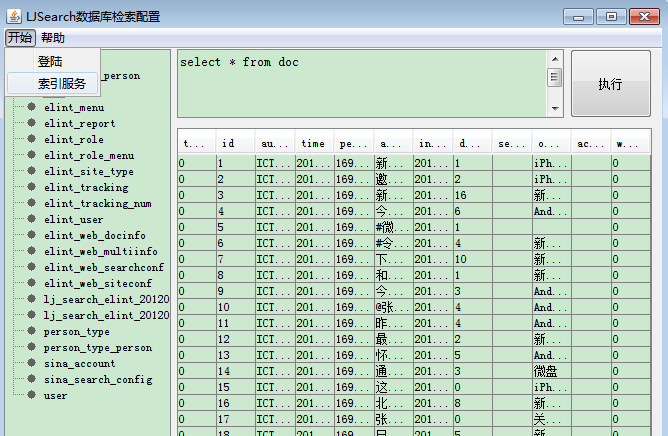
### 2.1.2、选择要建立索引的表，点击“执行”、“创建索引”

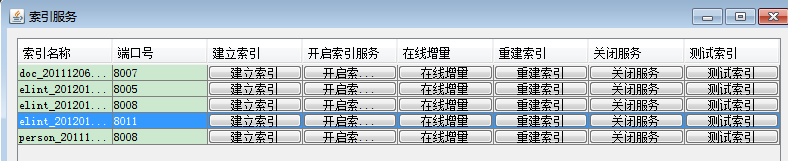


### 2.1.3、弹出如下界面：配置好字段信息:点“创建索引文件”

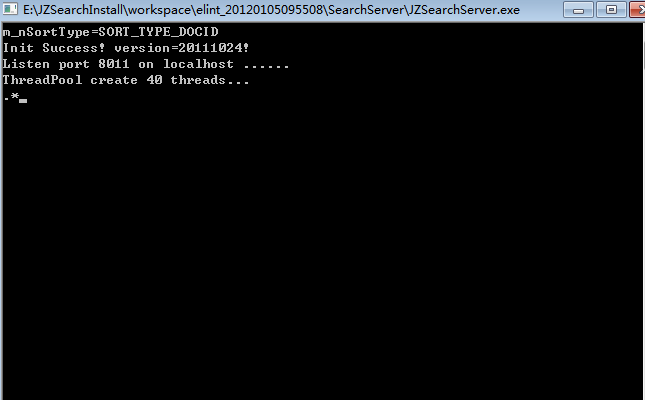


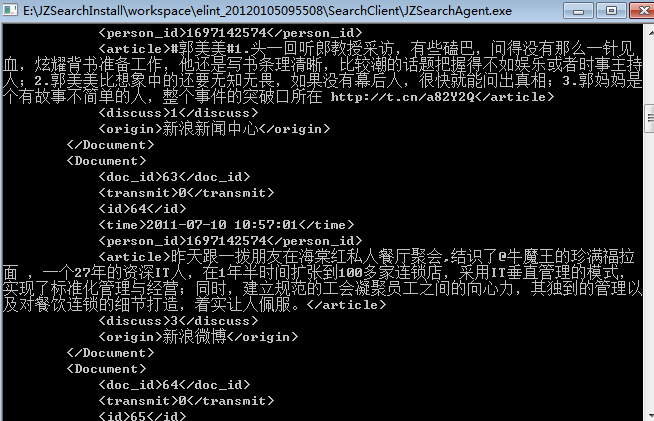
### 2.1.4:选择索引服务，然后选择项目，点击建立索引：





### 2.1.5：索引建立成功之后，即可开启索引服务，测试索引：





## 2.2、采用手工配置（适用于各种操作系统）

利用DBIndex目录下对应的可执行文件JZIndexer，搭建数据库搜索引擎服务的步骤（Windows下支持ODBC方式读取数据库，Linux下目前支持MySQL与Oracle）**。**

### 2.2.1 第一步：配置数据库读取参数

**注：如果不是直接读取数据库建立索引，则略过此步。**

配置数据库信息，结果写入index\dbConf.xml

配置信息如下所示：

<DatabaseConf>

<DBConnectString>DRIVER={MySQL ODBC 3.51 Driver};SERVER=192.168.1.202;PORT=3306;DATABASE=elint;UID=root;Pwd=root;</DBConnectString> //mysql

<Host>192.168.1.202</Host>#数据库服务器IP

<Username>root</Username>

<Password>root</Password>

<Database>elint</Database>

<Table>elint\_web\_docinfo</Table>

<SortField>inserttime</SortField>//用于排序的字段

<IncrementField>inserttime</IncrementField>

//用于做增量更新的字段,必须保证后续新增加的字段该值大于前面字段的值

<MaxMem>524000000</MaxMem>//最大内存数

<Compare>>=</Compare>

<ICTCLAS>on</ICTCLAS>

<RecordPerPage>100000</RecordPerPage>//每批处理的记录数

<Language>Chinese</Language>//少数民族语言Minor，

<Encoding>GBK</Encoding>//编码：GBK,UTF8,BIG5,GBK-FANTI,sqlserver版本上拷过来的

<KeyField></KeyField>//关键字索引字段名称列表，必须有一个文本类型的字段，用于分析判断记录是否已经存在，需要更新数据

</DatabaseConf>

其中：

**DBConnectString**为连接数据库的字符串，各个不同的数据库系统的字符串都不太一致，如：

**MYSQL：**

DRIVER={MySQL ODBC 3.51 Driver};SERVER=192.168.1.103;PORT=3306;DATABASE=elint;UID=root;Pwd=wysxzw;

**ACCESS：**

Provider=Microsoft.Jet.OLEDB.4.0;Data Source=E:\LJSearch\db1.mdb

**SQL Server:**

Provider=SQLOLEDB; Server=127.0.0.1;Database=LOS; uid=sa; pwd=sa;

**Username**为连接数据库的用户名；

**Password**为连接数据库的密码；

**Database**为指定的数据库名；

**Table**为需要建立全文搜索的表名，也可以是视图名称；

**SortField**为数据库建立索引时的记录排序字段名称；在本例中，是按照数据库入库的时间字段inserttime来排序的。

**IncrementField**用于增量索引时区分新旧记录的字段，要求后入库记录的该字段值必须大于先入库记录的值；在本例在，是按照数据库入库的时间字段inserttime来区分先后增量索引的。

**MaxMem**为系统建立索引时可以分配的最大内存，内存越大，建立索引越快，32位环境下最大不能超过1GB；需要比较大的内存，建议采用64位环境下对应的程序，同时，需要结合机器特点，内存小的机器，分配大内存往往导致建立索引不成功；另外，系统需要建立索引的字段。

Compare增量索引时采取的比较字符串,一般采用>=，采用=时，只更新上批次最后的记录；

**ICTCLAS**为系统建立索引时是否采用ICTCLAS智能分词系统；

Language处理的语言，默认为中英文混杂，选择CHINESE，少数民族语言设置为Minor；

RecordPerPage为系统索引时，每批加载的记录数；默认为100万，如果每天记录较多，建议修改为10万，从而提高速度。

KeyField: 关键字索引字段名称列表，必须有一个文本类型的字段，用于分析判断记录是否已经存在，需要更新数据

**注：为保障数据库连接成功，需要在索引服务器（可以和数据库服务器不是同一台机器）上，在“系统配置/管理工具”中建立连接该数据库的ODBC。**

### 2.2.2 第二步：数据库字段信息列表导入

配置字段信息，结果写入index\FieldInfo.txt

FieldInfo.txt内容，参见1.3中的示例表。

### 2.2.3 第三步：建立字段信息文件

执行bin\buildTable.bat建立字段信息，结果写入index\Field.dat

### 2.2.4 第四步：自动建立索引

注：如果不是直接读取数据库建立索引，则略过此步，需要自己调用API编程，自行建立索引。

执行bin\buildIndex.bat建立索引

### 2.2.5 第五步：启动搜索服务

自己建立的索引，也可以启动本服务。

执行bin\startServer.bat启动搜索服务；

### 2.2.6 第六步：启动客户端服务

执行bin\startClient.bat启动客户端服务；

### 2.2.7 其他脚本

1.buildOnlineIndex:建立在线索引

2.cleanIndex:清楚索引垃圾信息；

3.RebuildAllIndex:重新建立索引，不进行增量更新；

4.stopServer：停止搜索服务；

5.update:更新升级程序

### 2.2.8 Linux环境

1.新将linux-packet下面的install.sh执行下面命令：

chmod +x install.sh

2.执行安装命令

./install.sh

3.其他过程类似于Windows，不同的是这里面调用的是linux-packet\bin下面的脚本

# 进阶篇：了解内核

## 3.1、JZSearch检索语法说明

JZSearch的检索语法类似于SQL语句，由下面格式的语句组成

{[FIELD] fieldname/\* [搜索运算符] value1 value2 ... Value\_n}\*| [SORT] sort\_type

说明如下：

1. [FIELD] 为搜索系统保留字；
2. Fieldname为系统设置的字段名称，该字段必须在3.2 JZSearch\_FieldAdd中定义过，否则非法操作，系统不接受查询请求；其含义为在该字段上进行搜索运算；
3. Filedname也可以为“\*”,含义为通配符搜索，不需要指定特定字段，所有涵盖在通配搜索范围的字段都纳入搜索的范畴；
4. 搜索运算符包括：

AND/OR/NOT/NEAR/RANG/MIN/MAX/PREF/PREC/FUZZY，具体含义见后面的详细描述；

5. Value1 Value2等为参数列表

6. 大括号的语句可以反复叠加使用，各个语句之间是逻辑与的关系；

7. 各个部分之间参与空格或者<TAB>键分开，无论是保留字还是参数均不区分大小写；

8. [SORT] sort\_type 用于设定当前搜索结果的排序方式

### 3.1.1搜索与运算符AND

表示的含义: 后续参数列表的逻辑与关系，举例说明如下：

[FIELD] title [AND] 基金 公司

含义：搜索title字段中同时包含“基金”与“公司”的文档记录；

### 3.1.2搜索或运算符OR

表示的含义: 后续参数列表的逻辑或关系，举例说明如下：

[FIELD] title [OR] 基金 公司

含义：搜索title字段中包含“基金”或“公司”的文档记录；

[FIELD] \* [OR] 基金 公司

含义：搜索任意字段中包含“基金”或“公司”的文档记录；

### 3.1.3搜索非运算符NOT

表示的含义: 后续参数列表的逻辑非关系，举例说明如下：

[FIELD] \*[AND] 基金 [FIELD] title [NOT] 公司 中心

含义：搜索任意字段中包含“基金”，但是标题中不包含“公司”也不包含“中心”的文档记录；

### 3.1.4搜索邻近运算符NEAR

1. 格式为：[FIELD] fieldname/\* [NEAR] value1 value2 value3
2. 后续参数为3个，value1 value2 value3；前两个参数value1 value2为搜索内容，第三个参数value3为邻近的距离值；只有两个参数时，默认第三个参数value3为10.

3）表示的含义: 搜索内容value1 value2必须同时存在，而且距离不能超过value3个词，类似AND，但比AND严格。

举例说明如下：

[FIELD] \* [NEAR] 尚福林 卖国贼 9

含义：搜索任意字段中同时包含“尚福林”和“卖国贼”的文档，同时要求两者的距离不能超过9个词，采用该语法信息过滤将更有针对性；

### 3.1.5搜索智能模糊运算符FUZZY

1）表示的含义: 搜索方式为模糊匹配，尽可能地匹配搜索串，但模糊处理，不做严格的匹配。

举例说明如下：

[FIELD] \* [FUZZY] 张华平 kevinzhang 精准搜索内核

含义：搜索任意字段中"张华平 kevinzhang 精准搜索内核"的文档，按照相关度自动排序；

### 3.1.6搜索范围运算符RANG

1. 格式为：格式为：[FIELD] fieldname/\* [RANG] value1 value2
2. 后续参数为2个，value1 value2
3. 表示的含义: 字段fieldname的值必须在value1 value2之间的范围内；fieldname字段必须为数值型字段，否则非法；
4. 注意：如果字段为datetime型，其表示形式为：2012/06/20\_00:57:17，需要在日期与时间之间加上“\_”

举例说明如下：

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童

价格在1.0~9.0之间，且商品名包含“儿童”、“牛奶”的记录

[field] title [and] 北京 [field] publishtime [range] 2012/06/20\_00:57:17 2011/06/22\_14:34:55 [sort] publishtime desc

含义：搜索标题包含“北京”，且发表时间在2012/06/20\_00:57:17 2011/06/22\_14:34:55之间的搜索结果，按照publishtime倒序排列。

### 3.1.7搜索最小值运算符MIN

1. 格式为：格式为：[FIELD] fieldname/\* [MIN] value1
2. 后续参数为1个，value1
3. 表示的含义: 字段fieldname的值必须大于或等于value1；fieldname字段必须为数值型字段，否则非法；
4. 注意：如果字段为datetime型，其表示形式为：2012/06/20\_00:57:17，需要在日期与时间之间加上“\_”

举例说明如下：

[FIELD] price [MIN] 1.0 [FIELD] name [AND] 牛奶 儿童

价格不低于1.0，且商品名包含“儿童”、“牛奶”的记录

### 3.1.8搜索最大值运算符MAX

1. 格式为：格式为：[FIELD] fieldname/\* [MAX] value1
2. 后续参数为1个，value1
3. 表示的含义: 字段fieldname的值必须不大于value1；fieldname字段必须为数值型字段，否则非法；
4. 注意：如果字段为datetime型，其表示形式为：2012/06/20\_00:57:17，需要在日期与时间之间加上“\_”

举例说明如下：

[FIELD] price [MAX] 9.0 [FIELD] name [AND] 牛奶 儿童

价格不高于9.0，且商品名包含“儿童”、“牛奶”的记录

### 3.1.9搜索前缀运算符PREFIX

1. 格式为：格式为：[FIELD] fieldname/\* [PREF] value1
2. 后续参数为1个，value1
3. 表示的含义: 字段fieldname的内容必须以value1开头；fieldname字段必须为字符型字段，否则非法；

举例说明如下：

[FIELD] name [PREF] 张

姓名字段name必须以“张”作为前缀开头

### 3.1.10搜索精准运算符PRECISION

1. 格式为：格式为：[FIELD] fieldname/\* [PREC] value1
2. 后续参数为1个，value1
3. 表示的含义: 字段fieldname的内容必须精准匹配value1；fieldname字段必须为字符型字段，否则非法；

举例说明如下：

[FIELD] name [PREC] abc

姓名字段name必须以“abc”精准匹配，如“abcd”不作为匹配结果；

[FIELD] id [PREC] 123

姓名字段id必须以“123”精准匹配，如“1234”或者“0123”均不作为匹配结果；

### 3.1.11搜索排序运算符 SORT

1) 格式为：[SORT] sort\_type 用于设定当前搜索结果的排序方式

sort\_type参见下表：

|  |  |  |
| --- | --- | --- |
| **类型名** | **类型值** | **说明** |
| docid | 1 | 按照docid顺序排序，即先建索引的文档优先 |
| relevance | 2 | 按照相关度排序 |
| reverse\_docid | 3 | 按照docid逆序排序，即后建索引的文档优先 |
| <field\_name>descend/ascend | 4 | 按照指定的数值型字段进行排序， |

2) [SORT]可以出现在搜索语句的起始、中间以及末尾；如果不指定的话，系统会按照系统配置的排序方式进行排序；不做任何指定的前提下，系统按照DOCID排序；

3）sort\_type不区分大小写，只需要输入前五个字符即可。

4）<field\_name>descend/ascend field\_name必须是数值型，并设置为信息过滤。

5）示例如下：

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童 [SORT] docid

价格在1.0~9.0之间，且商品名包含“儿童”、“牛奶”的记录，结果按照docid排序

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童 [SORT] relev

价格在1.0~9.0之间，且商品名包含“儿童”、“牛奶”的记录，结果按照相关度排序

[FIELD] price [RANG] 1.0 9.0 [FIELD] name [AND] 牛奶 儿童 [SORT] rever

价格在1.0~9.0之间，且商品名包含“儿童”、“牛奶”的记录，结果按照docid逆序排序

### 3.1.12复合运算符COMPLEX

1. 格式为：[FIELD] fieldname/\* [COMPLEX] keyword1|| keyword2||…||keywordN word1|| word2||…||wordMdist
2. 后续参数至少2组:

keyword1|| keyword2||…||keywordN:关键词列表(中间用||隔开)

word1|| word2||…||wordM:关键词列表(中间用||隔开)

关键词列表至少需要两组，可以多组兼容

dist: 表示距离，不设置则不考虑之间的距离

1. 表示的含义:

{ keyword1 OR keyword2 OR…OR keywordN} AND {word1 OR word2OR…OR wordM}

如有距离，则表示集合{ keyword1 OR keyword2 OR…OR keywordN}与集合{word1 OR word2OR…OR wordM}，出现的词必须在dist范围以内。

举例说明如下：

[field] title [complex] 台湾||解放军||孕妇 甲流||人事

含义：要求title字段内，同时出现{台湾||解放军||孕妇}{甲流||人事}

[field] content [complex] 统计局||中国统计局||CPI骗人||砖家 10

含义：要求content字段内，同时出现{统计局||中国统计局||CPI }{骗人||砖家},且两者之间的距离必须在10个词内。

### 3.1.13列出所有索引的结果内容LISTALL

1. 格式为： [CMD] LISTALL
2. 表示的含义:

不给定搜索条件，要求列出所有的索引内容；此时field及其他and or操作无效。

3）可以对结果按照3.1.11的要求排序

举例说明如下：

[cmd] listall

含义：列出所有的索引内容,默认按照docid从小到大排序

[cmd] listall [sort] docid\_reverse

含义：列出所有的索引内容，按照docid从大到小逆序

### 3.1.14正负面情感搜索

正负面搜索采用特殊的搜索词，可以用前面所述的任何一种操作符号

特殊的搜索词参见下表：

|  |  |
| --- | --- |
| **类型名** | **说明** |
| ##负面JZSearch## | 搜索负面信息 |
| ##正面JZSearch## | 搜索正面信息 |
| ##色情JZSearch## | 搜索色情信息：目前还没有开放 |
| ##政治反动类JZSearch## | 搜索政治反动类信息：目前还没有开放 |
| ##毒品类JZSearch## | 搜索毒品类信息：目前还没有开放 |
| ##赌博类JZSearch## | 搜索赌博类信息：目前还没有开放 |
| ##法轮功类JZSearch## | 搜索法轮功类信息：目前还没有开放 |
| ##民运类JZSearch## | 搜索民运类信息：目前还没有开放 |
| ##反日类JZSearch## | 搜索反日类信息：目前还没有开放 |

示例如下：

[field] content [near] 张华平 ##负面JZSearch## 12

表达的意思为：在content字段搜索 “张华平” 和负面信息相隔12个词范围内的文档

[field] content [and] 张华平 ##正面JZSearch## [sort] publishtime desc

表达的意思为：在content字段搜索 包含“张华平” 和正面信息的文档，且按照发表时间倒序排列。

### 3.1.15删除显示运算符Deleted

1. 格式为： [deleted] on/off
2. 表示的含义:

搜索结果中包含删除文档的部分是否需要显示出来，如果为on将显示，否则不显示。

举例说明如下：

[FIELD] title [AND] 基金 公司 [deleted] on

含义：将所有基金公司的信息找出来，如果删除文档中有相关的内容也要显示出来。

### 3.1.16 两表关联搜索语法

<index>

<no>1</no> //说明：索引在搜索服务中的编号

<main>1</main>//主索引，搜索结果全部由该索引产生

<join>person\_id\_int</join>//关联字段名，外键用于关联附属表

<cmd> [cmd] list all [sort] publishtime desc</cmd>//搜索语句

</index>

<index>

<no>2</no>//说明：索引在搜索服务中的编号

<main>0</main>//从索引

<join>url\_id\_long</join>//关联字段名，与上面的person\_id\_int对应

<cmd> [field] groupName [or] ORGAN\_6E1F9622F90B5668D76F8E01E824C00D\_GROUP</cmd>//搜索语句

</index>

执行过程：在第二个索引中执行[field] groupName [or] ORGAN\_6E1F9622F90B5668D76F8E01E824C00D\_GROUP后，生成符合要求的person\_id\_int列表，作为查询条件之一，过滤第一个索引中搜索出来的结果。

### 3.1.17搜索聚类运算符Cluster

1. 格式为：格式为：[CLUSTER] fieldname1 fieldname2 …fieldnameN [Count] Num
2. 表示的含义: 将搜索结果分别按照字段fieldname1fieldname2 …fieldnameN的内容聚类；fieldname字段必须为字符型字段且系统已经保存，否则非法；
3. Num表示输出的聚类结果数目，不设置默认为10.

举例说明如下：

[field] \* [and] 安全 [cluster] keywords people location [count] 10

搜索“安全”，将搜索结果分别按照keywords people location字段聚类；这些字段里面的内容按照“#”分割，如keywords为“<keywords>举报#信息#工作#奖励#互联绿色情#</keywords>”

搜索结果会增加最后一段:

<SearchCluster>

<fieldcount>3</fieldcount>

<Cluster>

<fieldname>keywords</fieldname>

<value>中国信息安全/12#中国安全/8#信息安全/6#技术/5#安全/5#HotDB/4#discussion/4#processing/4#conferences/4#北京大学/4#</value>

</Cluster>

<Cluster>

<fieldname>people</fieldname>

<value>戴浩/8#方安全/3#哈尔/2#罗锋/2#郭建宇/2#赵泽良/2#胡啸/2#舒曼/2#吴琳琳/2#叶红吴亚非刘春林司张胜赵战生赵晓春刘品新吕欣/2#</value>

</Cluster>

<Cluster>

<fieldname>location</fieldname>

<value>中国/21#中国中华人民共和国/12#北京/7#中国江苏阜宁/4#大都/3#中国湖北省宜昌市/3#安定门/2#美国/2#台中/2#中国北京北京市江苏阜宁/2#</value>

</Cluster>

</SearchCluster>

其中每个值由 [word]/[freq]#组成。

### 3.1.18分组运算符Groupby

1. 格式为：格式为：[Groupby] fieldname1 fieldname2 …fieldnameN
2. 表示的含义: 将搜索结果分别按照字段fieldname1fieldname2 …fieldnameN的值进行分组；fieldname字段必须为数值型字段，否则非法；

举例说明如下：

[field] \* [and] 安全 [groupby] type\_id

搜索“安全”，将搜索结果分别按照type\_id字段分组；

搜索结果会增加最后一段:

<Groupby>

<fieldcount>1</fieldcount>

<group>

<fieldname>keywords</fieldname>

<groupcount>3</groupcount>

<item>

<value>4</value><count >3</count>

</item>

<item>

<value>23</value><count >2</count>

</item>

<item>

<value>8</value><count >1</count>

</item>

</group>

</Groupby>

### 3.1.19搜索结果比较运算符Compare Between With

1. 格式为：格式为：[Compare] fieldname1 [between] search\_cmd1 [with] search\_cmd2
2. 表示的含义: 将[field]fieldname2 value1 {其他命令}与[field]fieldname2 value2 {其他命令}的搜索结果进行比较，比较对应的fieldname1字段值的异同。

举例说明如下：

[Compare] keywords [between] [field] \* [and]华为 [field] type [and] news [with] [field] \* [and]联想 [field] type [and] news

搜索新闻中“华为” “联想”的文章搜索结果中，比较对应的keywords字段的异同，

[Compare] keywords [between] [field] \* [and]华为 [field] publishtime [range] 2012/1/1 2012/12/31 [with] [field] \* [and]华为 [field] publishtime [range] 2013/1/1 2013/12/31

搜索结果格式如下所示；

搜索结果如下：

<compare>

<same>信息/15/3#网络/3/15#标准/8/3#技术/7/1#信息安全/5/1#互联网/2/4#电子信息/3/1#中国/2/2#服务/2/2#服务信息中心标/2/2#</same>

<diff1>中国信息安全/12#信息技术/5#HotDB/4#conferences/4#机/4#北京大学/4#标准化/4#研究/4#discussion/4#researchers/4#</diff1>

<diff2>高校/5#人才/5#军/5#网罗/5#打造/5#薛蛮子/5#科技/5#民警/3#风范/3#微博/3#</diff2>

</compare>

### 3.1.20搜索结果自动去重处理Unique

1. 格式为：格式为：[Unique] fieldname1 fieldname2 .. fieldnameN
2. 表示的含义: 将搜索结果按照字段fieldname1 fieldname2 .. fieldnameN对搜索结果进行去重处理，也就是说不会重复出现<fieldname1 fieldname2 .. fieldnameN>对应完全一致的搜索结果。

举例说明如下： [field] \* [and] 图 [unique] key

表述的是：统配搜索"图"，搜索结果自动key必须都不一样

## 3.2 JZSearch搜索后台服务系统搭建

1. 可以支持Windows、Linux、FreeBSD等多种环境，支持普通PC机器即可运行，目前支持C/C++调用，索引程序不建议采用非C程序实现。搜索前端目前已经有C/C++/Php/Java等实现版本；
2. 基于JZSearch已经实现了后台的搜索服务，可以与数据库无缝融合。
3. 命令行： JZSearchServer.exe [搜索目录1] [搜索目录2] ... [搜索目录n]
4. 说明：搜索服务需要寻找指定目录下的SearchServer.xml，根据SearchServer.xml的内容提供搜索服务
5. SearchServer.xml示例如下：

<?xml version="1.0" encoding="GB2312"?>

<SearchServerConf>

<ReleaseVersion>

<version>3.0</version>

<lastModify>2011-8-28</lastModify>

</ReleaseVersion>

<Config>

<ServerIP>127.0.0.1</ServerIP>#检索服务器IP地址

<Port>8003</Port>##检索服务进程的端口号

<dictpath>D:\NLPIR\release\JZSearchInstall\dict</dictpath>##词典等目标文件

<ICTCLAS>on</ICTCLAS>

<index>

<indexfile>D:\NLPIR\test\uyghan\JZSearch</indexfile>##原始标准数据及索引结果文件

<FieldInfo>D:\NLPIR\test\uyghan\fieldinfo.dat</FieldInfo>##字段信息文件

<SORT>DOCID</SORT>

##排序方式：

##DOCID=按照DOCID顺序排序（即文档建索引的顺序），默认方式；DOCID=按照DOCID排序

##REVERSE\_DOCID=按照DOCID逆序排序（即后建索引的结果排名靠前）

##RELEVANCE=按照相关度排序

<Encoding>UTF8</Encoding>//编码：GBK,UTF8,BIG5,GBK-FANTI

<GroupMaxCount>50</GroupMaxCount>//分组最大条目数

<MexMem>1024000000</MexMem>//最大内存数

<Language>Minor</Language>//少数民族语言

</index>

</Config>

</SearchServerConf>

## 3.3 JZSearch客户端搭建与管理指南

### 3.3.1 客户端管理命令语法

客户端的命令都是采用搜索服务控制命令实现的，服务端控制命令类似于搜索语法，具体细节如下：

|  |  |
| --- | --- |
| **命令格式** | **说明** |
| [CMD] RELOAD | 表示重新加载新增的索引内容，系统会自动维护，不建议使用 |
| [CMD] MERGE | 表示需要对索引数据进行优化归并，系统会自动维护，不建议使用 |
| [CMD] QUIT | 表示要将系统搜索服务停止 |
| [CMD] Version | 查询搜索引擎的版本号 |
| [CMD] Backup | 索引自动备份，系统会自动维护，不建议使用 |
| [CMD] Restore | 索引自动恢复，系统会自动维护，不建议使用 |
| [CMD] del id1 id2 ... idn | 从索引中删除后续编号的文档 |
| [cmd] WHERE- DEL【搜索语句】 | 将符合【搜索语句】条件的记录删除 |
| [cmd] WHERE- UPDATE【搜索语句】 | 将符合【搜索语句】条件的记录删除，但保留最新的一条 |

### 3.3.2 命令行方式

命令行： JZSearchAgent.exe 参数 【Linux有配套程序】

参数说明：

0.参数不区分大小写；

1.参数为空，系统接收用户实时输入的命令行，命令行参考搜索语法手册**、JZSearch检索语法说明；**

2.QUIT:表示要将系统搜索服务停止；

3.Reload:表示重新加载新增的索引数据；

4.Merge:表示需要对索引数据进行优化归并；

5.VER:查询搜索引擎的版本号

6.BACKUP: 索引自动备份

7.RESTORE: 索引自动恢复

8.delete: 从索引中删除后续参数的文档

9.INDEX:启动索引，并自动加载进入搜索进程

需要事先配置索引的脚本，Windows下命名为 buildIndex.bat

Linux下命名为buildIndex.sh

### 3.3.3 C语言API方式

具体存放在SearchAgentAPI/Win-C目录下

JZSEARCHAGENTAPI\_API int jzsearch\_query\_client(const char \*XMLfile,const char \*query\_line,int nStart,int nPageCount,const char \*sResultName);

//SearchAgent.XML的存放地址

//query\_line:查询的语句 [FIELD] \* [AND] 钢铁 ;\*表示除标准号之外的通配符；2.AND后面可以放多个值；

//nStart：起始结果的行号，从0开始

//nPageCount:每页结果数目

//sResultName:result.xml存放的文件名

//

### 3.3.4 JAVA语言调用搜索客户端

具体存放在SearchAgentAPI/Java目录下

### 3.3.5 调用搜索服务的协议说明

查询协议

1. Start：查询起始记录号，整型数据，四个字节；
2. maxPageCount：当前页面记录总数，整型数据，四个字节；
3. query.size：查询语句的长度，整型数据，四个字节；
4. query：查询语句，字符串类型，长度：query.size

注意：所有的整数类型都需要转换成网络协议支持的格式，如C语言中，需要调用函数htonl进行格式转换。

搜索结果协议

1. count：搜索结果总数，整型数据，四个字节；
2. start\_ret：返回的起始记录号，整型数据，四个字节；
3. pagecount\_ret：返回的当前页面记录总数，整型数据，四个字节；
4. col\_count：每条记录的字段数，也成为列数，整型数据，四个字节；
5. buf\_size：内容缓存的长度，整型数据，四个字节；
6. 申请二维整型数组pVecLength[pagecount\_ret][col\_count],依次读取col\_count\* pagecount\_ret个整型数据，每个四个字节；pVecLength[i][j]表示的是第i个搜索结果第j个字段的内容长度，内容从7中的content读取；i，j从0开始编号；
7. 申请大小为buf\_size的内容串content，读取buf\_size个字符；按照6中pVecLength[i][j]的值对缓存的内容串进行分割即可。
8. 如果有搜索分组操作，继续下面的操作；否则，可直接结束
9. 分组信息读取，详细见下面的C语言代码

m\_pResult->group\_count=ntohl(\*((int\*)(pBuf+m\_nBufSize)));

m\_nBufSize+=sizeof(tuint);

m\_pResult->p\_vecGroupResult=new \_tGroupResult[m\_pResult->group\_count];

int i,j;

for (i=0;i<m\_pResult->group\_count;i++)

{

m\_pResult->p\_vecGroupResult[i].field\_id=ntohl(\*((int\*)(pBuf+m\_nBufSize)));

m\_nBufSize+=sizeof(tuint);

m\_pResult->p\_vecGroupResult[i].item\_count=ntohl(\*((int\*)(pBuf+m\_nBufSize)));

m\_nBufSize+=sizeof(tuint);

m\_pResult->p\_vecGroupResult[i].p\_count\_list=newint [m\_pResult->p\_vecGroupResult[i].item\_count];

m\_pResult->p\_vecGroupResult[i].p\_val\_list=new FILTER\_DATA\_TYPE [m\_pResult->p\_vecGroupResult[i].item\_count];

for (j=0;j<m\_pResult->p\_vecGroupResult[i].item\_count;j++)

{

m\_pResult->p\_vecGroupResult[i].p\_count\_list[j]=ntohl(\*((int\*)(pBuf+m\_nBufSize)));

m\_nBufSize+=sizeof(tuint);

m\_pResult->p\_vecGroupResult[i].p\_val\_list[j]=ntoh\_double(\*((FILTER\_DATA\_TYPE\*)(pBuf+m\_nBufSize)));

m\_nBufSize+=sizeof(FILTER\_DATA\_TYPE);

}

}

注意：所有的整数类型都需要将网络协议支持的格式转换成开发语言支持的格式，如C语言中，需要调用函数ntohl进行格式转换。

# 高级篇：API定制开发

## 4.1 字段定义接口

### 4.1.1 JZIndexer\_FieldAdd 添加一个字段

**JZSearchAPI\_API bool JZIndexer\_FieldAdd(const char \*sFieldName,const char \*sDBFieldName,int nFieldType,bool bIndexed,bool bRetrieved,bool bGeneral=false,bool bAbstracted=false);**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZIndexer\_FieldAdd | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

const char \*sFieldName:字段名称，在索引建立与搜索过程中，字段的唯一标示符

const char \*sDBFieldName:：对应于数据库的字段名称；在数据库搜索的时候，用于获取数据库的数据

int nFieldType：数据类型，参加3.1节字段类型定义表

bIndexed: 该字段是否需要索引，true，需要索引，以后可以搜索该字段，否则，不提供搜索

bRetrieved:搜索结果中是否要输出该字段内容

bGeneral:是否纳入通配搜索的范畴

**bAbstract:**是否进行搜索摘要提取并红显

**Remarks**

**Example**

//设置字段信息

**JZIndexer\_FieldAdd("title",NULL,FIELD\_TYPE\_TEXT,true,true,true,true);**

**//对标题建索引，需要搜索，需要通配搜索，需要存储，需要红显**

**JZIndexer\_FieldAdd("content",NULL,FIELD\_TYPE\_TEXT,true,true,true,true);**

**//对内容建索引，需要搜索，需要通配搜索，需要存储，需要红显**

**JZIndexer\_FieldAdd("id",NULL,FIELD\_TYPE\_INT,true,true,true,false);**

**JZIndexer\_FieldAdd("idf",NULL,FIELD\_TYPE\_FLOAT,true,true,true,false);**

**//对内容建索引，需要搜索**

JZIndexer\_FieldSave("./index/FieldInfo.dat");

**Output**

### 4.1.2 JZIndexer\_FieldSave 保存已经设置的字段信息

JZSearchAPI\_API bool JZIndexer\_FieldSave(const char \*sFieldInfoDataFile);

保存的字段信息数据文件

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZIndexer\_FieldSave | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

const char \*sFieldInfoDataFile：字段信息文件，系统特定的格式，建立索引和搜索都需要作为参数。

**Remarks**

**Example**

//设置字段信息

JZIndexer\_FieldAdd("title",NULL,FIELD\_TYPE\_TEXT,true,true,true);

//对标题建索引，需要搜索

JZIndexer\_FieldAdd("content",NULL,FIELD\_TYPE\_TEXT,true,true,true);

//对内容建索引，需要搜索

JZIndexer\_FieldAdd("id",NULL,FIELD\_TYPE\_INT,true,true,true);

JZIndexer\_FieldAdd("idf",NULL,FIELD\_TYPE\_FLOAT,true,true,true);

//对内容建索引，需要搜索

**JZIndexer\_FieldSave("./index/FieldInfo.dat");**

**Output**

### 4.1.3 JZIndexer\_FieldLoad 读取已经设置的字段信息文件

JZSearchAPI\_API bool JZIndexer\_FieldLoad(const char \*sFieldInfoDataFile);

读取已经保存好的字段信息数据文件

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZIndexer\_FieldLoad | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

const char \*sFieldInfoDataFile：字段信息文件，系统特定的格式，建立索引和搜索都需要作为参数。

**Remarks**

**Example**

**JZIndexer\_FieldLoad"./index/FieldInfo.dat");**

**Output**

JZSearchAPI\_API bool JZIndexer\_FieldLoad(const char \*sFieldInfoDataFile);

## 4.2 索引接口

### 4.2.1 JZIndexer\_Init 精准索引器初始化

JZSearchAPI\_API bool JZIndexer\_Init(const char \*sDictPath=0,const char \*sFieldInfoFile=0,intencoding=INDEX\_ENCODING\_GBK\_FANTI);

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZIndexer\_Init | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

//sDictPath：词典文件所在位置；为空时，采用n-gram索引方法

//sFieldInfoFile:域字段信息，用于支持多域索引，为空则只支持一个字段，参见3.3节JZIndexer\_FieldSave.

//encoding:设置编码，具体可以参见1.5

**Remarks**

1. 索引系统初始化，必须初始化后，才能使用CJZIndexer

2. 初始化失败的原因包括：

1）数据文件不齐全：sDictPath需要齐备的文件包括irreg2regularEnglish.map、dictionary.wordlist、dictionary.pdat、english.pdat

2）字段定义数据文件非法：即不是系统自动生成的字段信息

3）授权过期

3.失败时，可以查看当前目录下以日期命名的log日志文件，有具体的提示；

**Example**

**JZIndexer\_Init("E:\\Projects\\LJSearch\\dat",0,"LingjoinICTCLAS2010\*~!@#$%&&%$#@!~\*");**

**Output**

### 4.2.2 JZIndexer\_Exit 精准索引器系统退出

**JZSearchAPI\_API bool JZIndexer\_Exit();**

**//索引器系统退出**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZIndexer\_Exit | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

None

**Remarks**

1. 执行JZIndexer\_Exit，才能释放资源

**Example**

**JZIndexer\_Exit();**

**Output**

### 4.2.3 CJZIndexer 精准索引器类

**//建立索引的类**

**class JZSearchAPI\_API CJZIndexer {**

**public:**

**bool Export(const char \*sFilename,const char \*sWordList);**

**//内存大小控制**

**CJZIndexer(const char \*pcIndexFile,int nMaxMemSize=512000000 );**

**~CJZIndexer(void);**

**// TODO: add your methods here.**

**int MemIndexing(const char \*pText,const char \*sFieldName=0,int nMemSize=0);**

**//索引一段内存，，**

**//sFieldName：字段名称，为空则表示无字段信息**

**int FileIndexing(const char \*sTextFilename,const char \*sFieldName=0);**

**//索引一个文本文件，**

**//sFieldName：字段名称，为空则表示无字段信息**

**int IdIndexing(int term\_id,const char \*sFieldName=0);**

**//词ID索引,**

**//sFieldName：字段名称，为空则表示无字段信息**

**int IntIndexing(int value,const char \*sFieldName=0);**

**//对数值进行索引,**

**//sFieldName：字段名称，为空则表示无字段信息**

**int FloatIndexing(float value,const char \*sFieldName=0);**

**//对数值进行索引,**

**//sFieldName：字段名称，为空则表示无字段信息**

**bool Save();**

**//索引保存的名称**

**//索引保存的名称**

**int Merge();**

**//索引合并**

**int AddDoc();//增加文档**

**private://以下部分为系统使用，应用开发者不要改写，只能读取数据**

**int m\_nHandle;//索引器的Handle，无需调用申请**

**};**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZIndexer\_Exit | <JZSearchAPI.h> |

**Return Value**

None

**Parameters**

None

**Remarks**

1. 索引器类，可以并行多线程处理

**Example**

**Output**

### 4.2.3.1 CJZIndexer::CJZIndexer精准索引器类构建函数

**CJZIndexer(const char \*pcIndexFile,int nMaxMemSize=512000000 );**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：CJZIndexer** | <JZSearchAPI.h> |

**Return Value**

None

**Parameters**

const char \*pcIndexFile: 指定的索引文件名，系统根据该文件名自动生成相应的数据文件，检索时使用，索引与检索的文件必须一致。

int nMaxMemSize=512000000：系统给定的最大内存大小，默认为512MB。精准搜索索引时内存由系统自行分配与管理。

**Remarks:**

系统异常时，可以查看当前目录下以日期命名的log日志文件，有具体的提示；

**Example**

CJZIndexer \*pIndexer=new CJZIndexer("./index/Test0809",50000000);

**Output**

### 4.2.3.2 CJZIndexer::MemIndexing精准索引器类内存索引函数

**int MemIndexing(const char \*pText,const char \*sFieldName=0,int nMemSize=0);**

索引一段内存

//sFieldName：字段名称，为空则表示无字段信息

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：MemIndexing** | <JZSearchAPI.h> |

**Return Value**

int

**Parameters**

const char \***pText**: 待索引的内存文本内容起始地址。

const char \***sFieldName**：指定的给字段名称，需要跟3.2定义的字段名称一致。

Int **nMemSize**: 待索引的内存文本内容长度，为0，则按照字符串结束标准'\0'作为内存结束标记。

**Remarks:**

1. 对文本内容进行索引；系统默认只处理前5KB的内容，超出部分予以截断；
2. 字段可以是文本字段也可以是数值型字段；
3. 索引完成后，需要调用AddDoc方有效；

**Example**

CJZIndexer \*pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

**pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引**

pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引

pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果Output

### 4.2.3.3 CJZIndexer::FileIndexing精准索引器类文件索引函数

int FileIndexing(const char \*sTextFilename,const char \*sFieldName=0);

//索引一个文本文件，

//sFieldName：字段名称，为空则表示无字段信息

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：FileIndexing** | <JZSearchAPI.h> |

**Return Value**

int

**Parameters**

const char \*sTextFilename: 待索引的文本文件名。

const char \***sFieldName**：指定的给字段名称，需要跟3.2定义的字段名称一致。

**Remarks:**

1. 系统自动读取文件内容，并建立索引；系统默认只处理前5KB的内容，超出部分予以截断
2. 字段可以是文本字段也可以是数值型字段；

3. 索引完成后，需要调用AddDoc方有效；

**Example**

CJZIndexer \*pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

**pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引**

pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果Output

### 4.2.3.4 CJZIndexer::BigFileIndexing精准索引器类大文件索引函数

int BigFileIndexing(const char \*sBigFilename,unsigned int nOffset,unsigned int nSize,const char \*sFieldName=0);

//索引一个Big File中指定的偏移量和长度的文本文件，

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：BigFileIndexing** | <JZSearchAPI.h> |

**Return Value**

int

**Parameters**

//sBigFilename: Big File的文件名；

//nOffset:偏移量；

//nSize: 文件长度

//sFieldName：字段名称，为空则表示无字段信息

**Remarks:**

1. 系统自动读取文件内容，并建立索引；系统默认只处理前5KB的内容，超出部分予以截断

2. 索引完成后，需要调用AddDoc方有效；

**Example**

if (pFieldInfo->path.size()>0)

{

strcpy(pFilename,pFieldInfo->path.c\_str());//get path

strcat(pFilename,PATH\_DELEMETER);//分隔符

}

strcat(pFilename,pUnit);//filename

pUnit = row[nColID++];//get offset

nOffset=0;

if (pUnit!=0)

{

nOffset=atoi(pUnit);

}

pUnit = row[nColID++];//get length

nLen=0;

if (pUnit!=0)

{

nLen=atoi(pUnit);

}

#ifdef LJSEARCH\_DEBUG

printf("m\_pIndexer->BigFileIndexing(%s,%d,%d,%s)\n",pFilename,nOffset,nLen,pFieldInfo->index\_name.c\_str());

#endif

m\_pIndexer->BigFileIndexing(pFilename,nOffset,nLen,pFieldInfo->index\_name.c\_str());

//Big File Indexing

### 4.2.3.5 CJZIndexer::IdIndexing精准索引器类ID索引函数

int IdIndexing(int term\_id,const char \*sFieldName=0);

//词ID索引,

//sFieldName：字段名称，为空则表示无字段信息

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**IdIndexing | <JZSearchAPI.h> |

**Return Value**

Int

**Parameters**

int term\_id：词ID，ID需要大于0,超出词典词总数，调用无效

const char \***sFieldName**：指定的给字段名称，需要跟3.2定义的字段名称一致。

**Remarks:**

1. 字段可以是文本字段也可以是数值型字段；
2. 索引完成后，需要调用AddDoc方有效；

**Example**

CJZIndexer \*pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

**pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引**

pIndexer->IdIndexing(term\_id,"term\_id");//对词ID建立索引

pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果Output

### 4.2.3.6 CJZIndexer::IntIndexing精准索引器类整型索引函数

int IntIndexing(int value,const char \*sFieldName=0);

//对整型数值进行索引,

//sFieldName：字段名称，为空则表示无字段信息

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**IntIndexing | <JZSearchAPI.h> |

**Return Value**

int

**Parameters**

int value：数值

const char \***sFieldName**：指定的给字段名称，需要跟3.2定义的字段名称一致。

**Remarks:**

1. 此处字段的类型必须是FIELD\_TYPE\_INT；
2. 索引完成后，需要调用AddDoc方有效；

**Example**

CJZIndexer \*pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引

**pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引**

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果Output

### 4.2.3.7 CJZIndexer:: LongIndexing精准索引器类 64位长整型索引函数

#ifndef \_WIN32

int LongIndexing(long long value,const char \*sFieldName=0);

//对64位数值进行索引,

//sFieldName：字段名称，为空则表示无字段信息

#else

int LongIndexing(\_\_int64 value,const char \*sFieldName=0);

//对64位数值进行索引,

//sFieldName：字段名称，为空则表示无字段信息

#endif

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**LongIndexing | <JZSearchAPI.h> |

**Return Value**

int

**Parameters**

value：64位数值

const char \***sFieldName**：指定的给字段名称，需要跟3.2定义的字段名称一致。

**Remarks:**

1. 此处字段的类型必须是FIELD\_TYPE\_LONG；
2. 索引完成后，需要调用AddDoc方有效；

**Example**

CJZIndexer \*pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引

**pIndexer->LongIndexing(doc\_id,"id");//对ID建立索引**

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果Output

### 4.2.3.8 CJZIndexer::FloatIndexing精准索引器类浮点型索引函数

int FloatIndexing(float value,const char \*sFieldName=0);

//对浮点型数值进行索引,

//sFieldName：字段名称，为空则表示无字段信息

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**FloatIndexing | <JZSearchAPI.h> |

**Return Value**

int

**Parameters**

float value：数值

const char \***sFieldName**：指定的给字段名称，需要跟3.2定义的字段名称一致。

**Remarks:**

1. 此处字段的类型必须是FIELD\_TYPE\_FLOAT；
2. 索引完成后，需要调用AddDoc方有效；

**Example**

CJZIndexer \*pIndexer=new CJZIndexer("./index/Test0809",50000000);

//对文档列表进行索引

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引

pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引

**pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引**

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果Output

### 4.2.3.9 CJZIndexer::AddDoc精准索引器类文档添加函数

int AddDoc();//增加文档

当前文档索引完成，提交文档。准备索引下一篇文档;系统

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**AddDoc | <JZSearchAPI.h> |

**Return Value**

Int

成功则返回内部的doc\_id号；否则为0

**Parameters**

None

**Remarks:**

1. 对当前文档索引完成后，需要执行AddDoc提交，方可处理下一篇文档；
2. 内部将维护一个默认的doc\_id；AddDoc执行后，doc\_id将自动加1；
3. 一个索引器默认处理最大的文档数目为2亿，如果超过该上限，系统执行AddDoc将返回0，需要重新启动一个索引器进行所有，实现多个索引器并行多机器搜索。

**Example**

nLineId=1;

while(fgets(sLine,30690,fp)!=0)

{

start=clock();

pIndexer->MemIndexing(sLine,"content");//对文件内容建立索引

end=clock();

nTotalTime+=end-start;

nTotalLength+=strlen(sLine);

pIndexer->IntIndexing(doc\_id,"file\_index");

pIndexer->IntIndexing(nOffset,"offset");

**if(!pIndexer->AddDoc())**

{//当前文档索引完成，提交文档。准备索引下一篇文档

printf("Enough buffer!");

break;

}

if (nLineId%100==0)

{

printf("Indexing %s:Line %d Completed!\n",vecFileList[doc\_id].c\_str(),nLineId);

}

nLineId++;

nOffset=ftell(fp);

}

### 4.2.3.10 CJZIndexer::Save精准索引器类保存函数

bool Save();

//索引保存的名称

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**Save | <JZSearchAPI.h> |

**Return Value**

Bool 成功则返回true；否则为false

**Parameters**

None

**Remarks:**

1. 保持索引文件；

**Example**

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引

pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

**pIndexer->Save();//保存结果**

### 4.2.3.11 CJZIndexer::Merge精准索引器类索引合并函数

int Merge();

//索引合并

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**Merge | <JZSearchAPI.h> |

**Return Value**

Int

**Parameters**

None

**Remarks:**

1. 将小的索引文件碎片，合并为完整较大的文件，以确保搜索的效率；
2. 一般在增量索引比较多次的情况下，调用

**Example**

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引

pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果

**pIndexer->Merge();//索引文件归并处理**

### 4.2.3.12 CJZIndexer::Export精准索引器类索引导出函数

JZSearchAPI\_API bool Export(const char \*sExportFile,const char \*sWordList);

导出的索引数据文本，以便核查调试使用，索引比较大，一般不建议用户自行使用。

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZIndexer：：**Export | <JZSearchAPI.h> |

**Return Value**

Bool :成功为true，否则为false

**Parameters**

const char \*sExportFile: 导出的文本文件名称

const char \*sWordList: 数据词典目录下的dictionary.wordlist文件。

**Remarks:**

1. 处理失败后，可以查看当前目录下的当天日志文件；
2. 一般不建议使用

**Example**

for (int doc\_id=nDocID\_Start;doc\_id<vecFileList.size();doc\_id++)

{//索引一个文件

pIndexer->MemIndexing(vecFileList[doc\_id].c\_str(),"title");//对文件标题建立索引

pIndexer->FileIndexing(vecFileList[doc\_id].c\_str(),"content");//对文件内容建立索引

pIndexer->IntIndexing(doc\_id,"id");//对ID建立索引

pIndexer->FloatIndexing(doc\_id,"idf");//对IDF建立索引

pIndexer->AddDoc();

}

pIndexer->Save();//保存结果

**pIndexer->Export("Export0729.txt","E:\\Projects\\LJSearch\\dat\\dictionary.wordlist");//导出索引文件的实际内容**

## 4.3 检索接口

### 4.3.1 JZSearch\_Init 精准搜索器初始化

JZSearchAPI\_API bool JZSearch\_Init(const char \*pcIndexFile,const char \*sDictPath=0,const char \*sFieldInfoFile=0,intencoding=INDEX\_ENCODING\_GBK\_FANTI));

检索系统初始化，必须初始化后，才能使用CJZSearcher

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZSearch\_Init | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

const char \*pcIndexFile: 指定的索引文件名，系统根据该文件名自动生成相应的数据文件，检索时使用，索引与检索的文件必须一致。

sDictPath：词典文件所在位置；为空时，采用n-gram索引方法

sFieldInfoFile:域字段信息，用于支持多域索引，为空则只支持一个字段，参见3.3节JZIndexer\_FieldSave.

//encoding:设置编码，具体可以参见1.5

**Remarks**

1. 搜索系统初始化后，才能使用精准搜索类CJZSearch；
2. 2. 初始化失败的原因包括：

1）数据文件不齐全：sDictPath需要齐备的文件包括irreg2regularEnglish.map、dictionary.wordlist、dictionary.pdat、english.pdat

2）字段定义数据文件非法：即不是系统自动生成的字段信息

3）授权过期

3.失败时，可以查看当前目录下以日期命名的log日志文件，有具体的提示；

**Example**

**JZSearch\_Init("E:/语料库资源/Corpus/搜索语料库/index\_ch/Test0807","E:\\Projects\\LJSearch\\dat","E:/语料库资源/Corpus/搜索语料库/index/FieldInfo.dat");**

printf("TestGBKChinese JZSearch\_Init OK!\n");

**Output**

### 4.3.2 JZSearch\_Exit 精准搜索器系统退出

**JZSearchAPI\_API bool JZSearch\_Exit();**

**//系统退出**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **JZSearch**\_Exit | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

None

**Remarks**

1. 执行**JZSearch**\_Exit，才能释放资源

**Example**

**JZSearch\_Exit();**

**Output**

### 4.3.3 JZSearch\_Reload 精准搜索器系统增量加载

**JZSearchAPI\_API bool JZSearch\_Reload();**

**//系统增量更新后，重新加载**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **JZSearch**\_**Reload** | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

None

**Remarks**

1. 增量索引完成后，需要执行**JZSearch**\_**Reload**，在不影响当前搜索服务的前提下提供新的搜索服务

**Example**

indexing("E:\\Projects\\LJSearch\\LJSearch\\TestDLL\\Corpus\\List.txt",vecFileList);

//增量索引，采用CJZIndexer建立索引的一个函数，其目的是建立指定文件列表内的索引文件

printf("indexing(\"\\List2.txt\" OK! Time=%f\n",duration);

**JZSearch\_Reload();**

**//动态加载，进行新的搜索服务**

### 4.3.4 JZSearch\_Export 精准搜索器系统索引内容导出函数

**JZSearchAPI\_API bool JZSearch\_Export(const char \*sExportFile,const char \*sWordList);**

**//导出内部的数据，以便核查**

**JZSearchAPI\_API bool JZSearch\_Export(int nTermID,const char \*sExportFile,const char \*sWordList);**

**//导出内部的数据，以便核查**

**功能类似于4.3.10 CJZIndexer::Export，内部调试使用，不推荐用户使用**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **JZSearch**\_**Export** | <JZSearchAPI.h> |

**Return Value**

Return true if init succeed. Otherwise return false.

**Parameters**

const char \*sExportFile: 导出的文本文件名称

const char \*sWordList: 数据词典目录下的dictionary.wordlist文件。

**Remarks**

1. 增量索引完成后，需要执行**JZSearch**\_**Reload**，在不影响当前搜索服务的前提下提供新的搜索服务

**Example**

indexing("E:\\Projects\\LJSearch\\LJSearch\\TestDLL\\Corpus\\List.txt",vecFileList);

//增量索引，采用CJZIndexer建立索引的一个函数，其目的是建立指定文件列表内的索引文件

printf("indexing(\"\\List2.txt\" OK! Time=%f\n",duration);

**JZSearch\_Reload();**

**//动态加载，进行新的搜索服务**

### 4.3.5 JZSearch\_Merge 精准搜索器系统索引归并优化函数

**JZSearchAPI\_API bool JZSearch\_Merge();**

**//系统自动优化，对索引信息进行归并处理，将多个小的索引文件合并为新的索引文件**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **JZSearch**\_**Reload** | <JZSearchAPI.h> |

**Return Value**

Return true if succeed. Otherwise return false.

**Parameters**

None

**Remarks**

1. 索引归并相对耗时，建议小的索引文件比较多的时候再根据一定的业务策略调用该函数；

2. 系统会自己进行优化处理，无须用户干预

**Example**

**JZSearch\_Rerge();**

**//系统进行归并**

### 4.3.6搜索结果的数据记录结构

**typedef struct tRESULT\_RECORD {//搜索结果结构，用于检索计算使用**

**int doc\_id;**

**int offset;//在域字段内的偏移量**

**double score;//排序用的打分机制**

**}RESULT\_RECORD;**

**typedef RESULT\_RECORD \* RESULT\_RECORD\_VECTOR;**

**搜索结果XML文件格式如下：**

**<?xml version="1.0" encoding="gb2312" standalone="yes" ?>**

**<LJSearch-Result>**

**<Result-Number>**

**<Total-Number>1</Total-Number>//符合条件的搜索结果总数**

**<Return-Number>1</Return-Number>//返回符合条件的搜索结果总数**

**<Start-Position>0</Start-Position>//结果集合中的起始编号**

**</Result-Number>**

**<Result>**

**<Document>**

**<doc\_id>0</doc\_id>//字段名doc\_id 系统内部的doc\_id编号，由AddDoc函数顺序递增**

**<title>E:\Projects\LJSearch\LJSearch\TestDLL\Corpus1021\Corpus\Meteorological Bulletin Automatic Generation based on Spatio-Temporal Reasoning.txt</title>//字段名title由3.2 FieldAdd定义的需要检索出的字段**

**<content>Meteorological Bulletin Automatic Generation based on Spatio-Temporal Reasoning Abstract Meteorological bulletin has more and more diversified, large scale, highly integrated requirements and potential demands from whole society. The strong professional ef-forts involved in transforming the variety of special meteorological data to natural language text are becoming more challeng-ing in providing sophisticated and easily understood weather features. This paper presents a new Meteorological bulletin automatic generation method based on spa-tio-temporal reasoning. To enhance an ex-act and non-redundant description for complex meteorological data, and for spe-cial future tendency dynamics in emerged interesting areas.</content>**

**<id>0</id>**

**<idf>0.000000</idf>**

**</Document>**

**</LJSearch-Result>**

### 4.3.7 CJZSearcher精准搜索器类

**class JZSearchAPI\_API CJZSearcher{**

**public:**

**CJZSearcher(int sort\_type=SORT\_TYPE\_DOCID);**

**~CJZSearcher(void);**

**// TODO: add your methods here.**

**void Search(const char \*query\_line,int nStart,int nPageCount,const char \*sResultName=0);**

**//query\_line: 查询表达式**

**//nStart:记录起始地址**

**//nPageCount：当前页返回结果数目**

**//nPageCount=-1:当前页需要返回所有的结果数目**

**//sResultName：结果存储的XML地址**

**const RESULT\_RECORD\_VECTOR Search(const char \*query\_line,int \*p\_nResultCountRet);**

**//query\_line: 查询表达式**

**//p\_nResultCountRet:搜索结果总数**

**bool DocDelete(int doc\_id);**

**private://以下部分为系统使用，应用开发者不要改写，只能读取数据**

**int m\_nHandle;**

**int m\_nSortType;//排序方法编号**

**char \*m\_pResultBuf;**

**};**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| JZIndexer\_Exit | <JZSearchAPI.h> |

**Return Value**

None

**Parameters**

None

**Remarks**

1. 索引器类，可以并行多线程处理

**Example**

**Output**

### 4.3.7.1 CJZSearcher::CJZSearcher精准索引器类构建函数

**CJZSearcher(int sort\_type=SORT\_TYPE\_DOCID);**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZSearcher：：CJZSearcher** | <JZSearchAPI.h> |

**Return Value**

None

**Parameters**

**int sort\_type=SORT\_TYPE\_DOCID**: 设定搜索的排序方式

**Remarks:**

1. 系统异常时，可以查看当前目录下以日期命名的log日志文件，有具体的提示；
2. 必须调用JZSearch\_Init，系统可以同时实例化多个 CJZSearcher,采用不同的排序方式；

**Example**

JZSearch\_Init("./index/Test0809","E:\\Projects\\LJSearch\\dat","./index/FieldInfo.dat");

printf("TestGBKChinese JZSearch\_Init OK!\n");

**CJZSearcher \*pSearcher=new CJZSearcher(SORT\_TYPE\_DOCID);//按照相关度排序**

printf("TestGBKChinese new CJZSearcher OK!\n");

**Output**

### 4.3.7.2 CJZSearcher::Search精准索引器类搜索函数

**void Search(const char \*query\_line,int nStart,int nPageCount,const char \*sResultName=0);**

**根据查询表达式，生成搜索结果文件**

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZSearcher：：Search** | <JZSearchAPI.h> |

**Return Value**

None

**Parameters**

query\_line: 查询表达式，需要符合搜索语法，参见6、JZSearch检索语法

nStart:记录起始地址

nPageCount：当前页返回结果数目，-1时，需要返回所有的结果数目

sResultName：结果存储的XML文件地址

**Remarks:**

1. 系统异常时，返回结果为空，可以查看当前目录下以日期命名的log日志文件，有具体的提示；

**Example**

strcpy(sLine,"[FIELD] title [AND] 解放军");//甲型H1N1流感

pSearcher->Search(sLine,0,-1,"Result.xml");//搜索title字段，将包含“解放军”的所有记录保存到Result.xml文件中。

Output

### 4.3.7.3 CJZSearcher::Search精准索引器类搜索函数

**const RESULT\_RECORD\_VECTOR Search(const char \*query\_line,int \*p\_nResultCountRet);**

类似于5.8.2,不同的是，该函数返回的是结果记录集数组，而不是文件本身，开发者可以根据返回的doc\_id进行进一步的搜索处理

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZSearcher：：Search** | <JZSearchAPI.h> |

**Return Value**

None

**Parameters**

**query\_line: 查询表达式**

**p\_nResultCountRet:搜索结果总数存储的指针**

**Remarks:**

1. 系统异常时，返回结果为空，可以查看当前目录下以日期命名的log日志文件，有具体的提示；

**Example**

strcpy(sLine,"[FIELD] title [AND] 解放军");//甲型H1N1流感

int nResultCount=0;

RESULT\_RECORD\_VECTOR pResult =pSearcher->Search(sLine,&nResultCount);//搜索title字段，将包含“解放军”的所有记录保存到pResult数组中。

Output

### 4.3.7.4 CJZSearcher::DocDelete精准索引器类索引文档删除函数

**bool DocDelete(int doc\_id);**

删除内部编号为doc\_id的文档

|  |  |
| --- | --- |
| **Routine** | **Required Header** |
| **CJZSearcher：：DocDelete** | <JZSearchAPI.h> |

**Return Value**

bool

**Parameters**

**int doc\_id；内部编号参数**

**Remarks:**

1. 系统异常时，返回结果为空，可以查看当前目录下以日期命名的log日志文件，有具体的提示；
2. doc\_id 是系统内部字段维护的，由AddDoc实现该字段的自动维护
3. 调用该函数之前，一般都需要事先检索，并获取检索记录中的doc\_id;

**Example**

**pSearcher->DocDelete(3908);//删除3908号文档**

**pSearcher->DocDelete(2110);//删除2110号文档**

Output

## 4.4 利用JZSearch开发程序，搭建搜索引擎服务指南

搭建搜索引擎服务的步骤（建议尽量采用通用篇：五步配置搭建搜索服务）：

|  |  |  |
| --- | --- | --- |
| 步骤 | 处理内容 | 参见 |
| **建立索引过程** | | |
| Step 1 | 定义字段索引信息 | 参见3.2 JZIndexer\_FieldAdd |
| Step 2 | 保存字段索引信息， | 参见3.3 JZIndexer\_FieldSave |
| Step 3 | 索引器初始化 | 参见4.1 JZIndexer\_Init 精准索引器初始化 |
| Step 4 | 建立索引器类 | 4.3.1 CJZIndexer::CJZIndexer |
| Step 5 | 扫描数据库或者文件中的文档，依次针对每篇文档建立索引 | 参见4.3.2~4.3.6； |
| Step 6 | 处理完成文档，调用AddDoc | 4.3.7 |
| Step 7 | 索引完毕，调用Save | 4.3.8 |
| **建立搜索后台服务过程** | | |
| Step 1 | 按照已索引的内容搭建搜索服务 | 参见第7章 |
| **建立搜索前段服务过程** | | |
| Step 1 | 按照客户端，选择C或者Java开发客户端程序 | 参见第8章 |
| **增量索引过程** | | |
| Step 1 | 完全参照第一次建立索引的过程，注意增量索引文件必须和第一次索引文件名一致 | 参见**建立索引过程** |
| Step 2 | 通过客户端命令行方式 | 参见3.3 JZIndexer\_FieldSave |

## 4.5 索引数据辅助分析

Bin目录下的部分高级调试功能

exportIndex.bat：导出各个索引词对应的文档数目，可以辅助我们分析索引词的质量，及我们索引文档的词语宏观统计特点。用户也可以利用类似的命令做各种分析。exportIndex.bat的其格式如下：

JZIndexer.exe e 【索引带路径的文件名】【字段带路径的文件名】【词典所在的路径】【词典wordlist的路径与文件名】 【导出的文件名】i

注意：如果在Linux下，需要将JZIndexer.exe改成 ./JZIndex

# 案例篇

JZSearch已应用到了多个海量搜索系统中，目前已经应用于中国邮政搜索引擎、河北省标准搜索引擎、中国对外承包工程商会的知识搜、富基融通（纳斯达克上市公司：EFUT）商品搜索、北京市网控办的WBK微博监控系统中。

## 5.1 中国邮政集团名址信息中心首页的邮址垂直搜索

访问地址为：[www.cpdc.com.cn](http://www.cpdc.com.cn)其中邮编搜索、企事业单位查询的内核均由JZSearch实现，目前每年的访问量超过3000万。



图1 基于JZSearch的中国邮址搜索引擎展示效果

## 5.2 河北标准化研究院的标准搜索

该系统的搜索部分均由JZSearch对其国内外标准数据库及全文文件库进行索引，提供全省上万家企事业单位与政府机构搜索，搜索段为C/S模式。



图2 基于JZSearch的标准文献垂直搜索展示效果

## 5.3 中国对外承包工程商会的知识搜索门户

该系统主要采用JZSearch搜索对中国对外承包商会内部100台办公电脑的文档进行索引，建立内网知识搜索门户，目前已经稳定运行2年多。



图3 基于JZSearch的中国对外承包工程商会的知识搜索

## 5.4 富基融通的商品比价搜索

该系统主要采用JZSearch搜索对庞大的商品库进行同一性判别，并提供比价搜索功能，每日新增数据1000万，同时支持在线搜索1万并发。该系统已经内嵌到其MyStore项目中。



图4 基于JZSearch的富基融通商品比价搜索系统

## 5.5 WBK微博人物搜索

WBK为北京市网控办的微博监控系统，内置的微博搜索与人物搜索由JZSearch支持，搜索效果如下：



图5 基于JZSearch的微博人物搜索系统

# 问答篇：FAQ及小技巧

## 6.1：环境问题

### 6.1.1 JZSearch支持Linux吗？

支持Windows, Linux等各类操作系统，目前有Win7, WinXP版本，及Linux 64位版本，如需其他环境，可以直接登录www.nlpir.org 寻求技术支持。

### 6.1.2 Linux环境使用太不方便，有什么技巧？

字段数据生成、索引生成均可以在Windows环境下通过可视化界面实现，生成的数据在Windows和Linux都是兼容的。

## 6.2：索引问题

### 6.2.1 在Windows下，针对MySQL数据库的索引老是创建不成功，访问不了数据库，为什么？

Windows下为了兼容各个数据库，JZSearch采用了ODBC访问数据库，MySQL数据库的索引不成功，往往是本地没有安装ODBC driver，可以访问MySQL官网下载，下载地址为：<http://www.mysql.com/downloads/connector/odbc/>。

### 6.2.2 采用bigfile字段，为什么？

### 6.2.3 数据库增删改如何适应，为什么？

## 6.3：搜索问题

### 6.3.1 老是搜索不到结果或者搜索到的结果老是不变的，怎么回事？

搜索不到结果，主要原因包括：

1. 搜索服务没有启动，或者客户端和服务端的IP及端口不匹配；
2. 搜索语法表达式有问题，如：field等关键字拼写错误，或者索引字段名称拼写错误。详细情况可以查看搜索服务目录下的日志文件，有详细的错误说明。
3. 搜索内容确实不存在搜索结果，建议换成通用的搜索词实验。

# 作者篇



张华平 博士 副教授 硕导

北京理工大学大数据搜索与挖掘实验室 主任

地址：北京海淀区中关村南大街5号 100081

电话：+86-10-68918642

Email:kevinzhang@bit.edu.cn

MSN: pipy\_zhang@msn.com;

网站: http://www.nlpir.org (自然语言处理与信息检索共享平台)

博客:http://hi.baidu.com/drkevinzhang/

微博: http://www.weibo.com/drkevinzhang/

Dr. Kevin Zhang (张华平，Zhang Hua-Ping)

Associate Professor, Graduate Supervisor

Director, Big Data Search and Mining Lab.

Beijing Institute of Technology

Add: No.5, South St.,Zhongguancun,Haidian District,Beijing,P.R.C PC:100081

Tel: +86-10-68918642

Email:kevinzhang@bit.edu.cn

MSN: pipy\_zhang@msn.com;

Website: http://www.nlpir.org (Natural Language Processing and Information Retrieval Sharing Platform)

Blog:http://hi.baidu.com/drkevinzhang/

Twitter: <http://www.weibo.com/drkevinzhang/>

