# J Search Engine

00648333 陈志杰　　00648332 揭忠

January 1, 2008

## 1 简介

J Search Engine是由陈志杰(Joyan),揭忠共同开发的一个模块化,低耦合度,高扩展性的微星实验性本地文本搜索引擎,目的在于展示搜索引擎的基本原理和结构,熟悉可扩展性软件设计方法.

## 2 使用方法

1. pre_process的第一个参数是需要递归建立索引的子目录，第二个参数是工程名。
2. 建立倒排索引成功后，会自动调用服务程序，监听本地的7891端口等待连接。
3. 如果早就建立好了索引，也可以直接调用”start.sh 目录名 工程名”启动服务。
4. 启动后，用telnet（windows底下的不支持，用linux下的）连接主机的7891端口即可。

## 3 主要流程说明

### 3.1 抓取部分

**抓取**:某根目录下的所有的文件遍历并且结合成一个比较大的.raw类型的文件(详情请参考crawl.cpp)，.raw类型的文件格式如下：

```
1  version:1.0
2  url:dir/path/foo.txt
3  length:12345
```

```
4
5   XXXXXXXXXX
6   XXXXXXXXXXX
7   ...
8   XXXXXX
9
10  version:1.0
11  ...
```

建立此文件以后就不需要再对原文件进行重新访问了，这是为了模仿实际情况，在实际的网络查询中，也不会直接去访问原文件。

## 3.2 预处理部分

1. **切词**:将每个文件都切成一个一个的词，以固定的格式储存在.raw.seg类型的文件中。具体格式如下：

```
1     DocID term1 term2 ...
2     DocID term1 term2 ...
```

2. 利用docoff处理raw文件，记录文档号和对应的偏移到.didx文件中，didx格式如下：

```
1     DocID 十六进制的偏移量例如：(00002432)
2     DocID 十六进制的偏移量
```

3. 利用didx文件改进raw.seg文件，改成准倒排文件的格式(也就是.piidx格式),主要格式如下：

```
1     term1 DocID
2     term2 DocID
3     ...
```

4. 利用linux自带的sort工具对piidx文件按照单词排序，整理成 .piidx.sort文件。

5. 利用invert在piidx.sort的基础上再一步改进，改成倒排文件的格式(也就是 .iidx格式),主要格式如下：

```
1     term DocID DocID Docid...
2     term DocID DocID ...
```

以上便是预处理的主要过程了，此步骤虽然简单，但是其优点也就是简单，在下面的过程中你将会看到一个又一个非常非常简单的步骤，但当其结合起来时确实现了一个比较复杂的功能。

## 3.3 查询服务器

为了只调用一次create_map()函数,减少查询的时间,该部分被我们做成了一个服务器,监听在7891端口,如果向他发送一个不以@打头的字符串,他就会启动搜索并返回结果.这个过程本应是udp的,但那样还要开发client端,这对于一个搜索引擎来说是无意义的(因为我们打算将来的2.0版本是直接基于B/S的),搜以我们暂时采用了tcp传输.
借到一个查询请求后,serv工作流程如下::
首先将预处理的文件利用map_creat函数将每个词与其在文档中的位置对应起来，这样便于以后的查询。
map类能够很好的将记录和记录的位置对应起来，对以后的查询有非常大的帮助。
然后，当输入某一短语的时候，首先调用cut_word函数对其进行切词处理，将其切成一个一个的词组(所以用户查询的时候只需要像百度那样连续输入即可)，每个词组之间以空格隔开。
接着，将此第一个词组的读入到string中，利用map查找到记录的地址，然后调用get_term函数从相应的地址中读取所需的内容，接着调用process_term函数计算此词组的权值，并储存在vector数组中，调用res_merge函数对此所有词组的vector数组进行处理，计算出所输入短语在每篇文章中的权值，然后serv函数按照权值大小将查询结果按照固定的模式输出。
输出格式为：

```
标题（即文件名）
摘要
路径

标题（即文件名）
摘要
路径

标题（即文件名）
摘要
路径
```

其中,对摘要中前后切词中文的处理是一个技术难点.

# 4 本作品的亮点

1. **模块化和可扩展性**:在设计和实现的时候,我们充分考虑到以后对她的扩展,所以尽量降低各组成部分之间的耦合度,简化接口,是各部分都能独立作为一个小工具.
2. **服务器化**:虽然简陋,但是却节省了建立查询树的时间,同时起到了很好的试验目的,为下一个基于B/S的并行版本的推出打下了结构和设计上的基础.

# 5 下个版本的改进

1. **对中文的统一转换**：目前的版本只能处理gb2312编码的文本和文件名，从下个版本开始，我们打算将所有的编码在抓取的时候就统一转化为UTF-8处理，这样就可以支持多种编码格式了。
2. **进一步服务器化**:将serv拆分为数据库服务器和用户交互服务器，前者加入并发特征，只负责对提交的查询请求返回根据权重排好序的文档id，后者接受用户连接，转发请求和生成摘要并呈现给用户等特征。
3. **改进排名算法**：将单词第一次出现在文章中的权重设为1000,以后每重复出现一次就加一，同时将只检索到部分词的文章也加入排名。

# 6 SVN地址

http://code.google.com/p/jsepku/

# 7 Bug Report

”Joyan”<ilcq@163.com>

# 8 分工

**陈志杰(Joyan)** shell编程部分，crawler部分，切词部分，cut_word函数，process_term函数，res_merge函数，serv.cpp文件，软件调试，文档编写。

**揭忠** 建立didx文件，piidx文件和iidx文件，query函数，create_map函数，get_term函数，源代码注释，文档编写。

# 9 声明

切词部分使用的是闫宏飞老师的TSE部分的切词实现，并做了少量修改（已在源文件中注明）。

# 10 个人心得

揭忠：通过此次大作业的编写，我充分体会到了将一个复杂问题简单化的方法，我认为此次作业的步骤划分绝对是一个亮点，以前总是认为一个函数实现的功能越多越好，通过此次的编写，才感觉到其实是越简单越好，充分感觉到步骤的重复性以及中间文本的重用。此次大作业通过对一些步骤的重用，将某些本来很复杂的程序很简单的实现了。