

**Geeking搜索引擎项目报告**

团队名称：Geek

团队成员：

陈新荃2014E8018461048

高妍 201428013229077

林裕杰

肖卡飞

2014年12月14日

目录

[目录 2](#_Toc405806037)

[**一．项目总体介绍** 3](#_Toc405806038)

[**二．设计方案** 3](#_Toc405806039)

[1. 网页爬虫 3](#_Toc405806040)

[1.1 爬取策略 3](#_Toc405806041)

[1.2 具体实现 3](#_Toc405806042)

[2. 索引构建 3](#_Toc405806043)

[2.1 数据结构 3](#_Toc405806044)

[2.2 构建流程 4](#_Toc405806045)

[2.3 网页过滤 4](#_Toc405806046)

[2.4 文本分词 4](#_Toc405806047)

[3. 检索策略 4](#_Toc405806048)

[3.1 检索流程 4](#_Toc405806049)

[3.2 结果排序 4](#_Toc405806050)

[3.3 结果聚类 4](#_Toc405806051)

[4. 前端处理 5](#_Toc405806052)

[4.1 页面元素 5](#_Toc405806053)

[4.2 自动补齐 5](#_Toc405806054)

[4.3 搜索词推荐 5](#_Toc405806055)

[4.4 摘要快照及高亮 5](#_Toc405806056)

[**三．测试与评估** 5](#_Toc405806057)

[1. 测试环境 6](#_Toc405806058)

[2. 测试结果 6](#_Toc405806059)

[2.1 功能测试 6](#_Toc405806060)

[2.2 性能测试 6](#_Toc405806061)

[**四．创新点** 6](#_Toc405806062)

[1. 标题权重 6](#_Toc405806063)

[2. 中间字自动补全 6](#_Toc405806064)

[3. 【还有什么尽管添加】 6](#_Toc405806065)

[**五．经验与总结** 7](#_Toc405806066)

[1. 个人经验总结 7](#_Toc405806067)

[2. 组长总结 7](#_Toc405806068)

# **一．项目总体介绍**

🡪陈新荃

采用什么方法，实现了什么功能，基于什么语料，创新点

时间、代码量、github结果等

# **二．设计方案**

## 1. 网页爬虫

### 1.1 爬取策略

🡪咖啡

### 1.2 具体实现

🡪咖啡

第三方jar包等

## 2. 索引构建

🡪裕杰

总体介绍

### 2.1数据结构

🡪裕杰

数据库、索引在内存中的结构等

### 2.2构建流程

🡪裕杰

### 2.3 网页过滤

🡪陈新荃

### 2.4文本分词

🡪陈新荃

## 3. 检索策略

🡪裕杰

总体介绍

### 3.1检索流程

🡪裕杰

流程图

### 3.2 结果排序

🡪陈新荃

合并、排序等

### 3.3结果聚类

🡪裕杰

聚类探究与算法实现

## 4. 前端处理

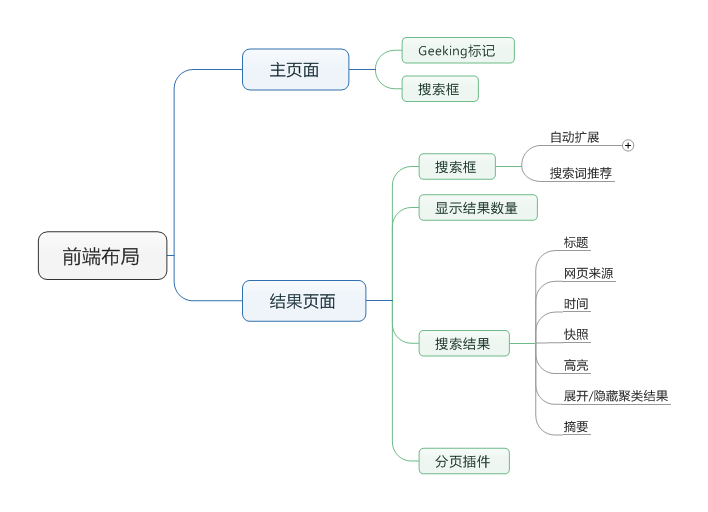
前端总体框架采用**Ajax+Jquery+JSP**的富客户端方式实现。使用**Javascript**绑定和处理所有数据，操作**Document Object Model**进行动态显示及交互。

AJAX具有无刷新更新技术、异步与服务器通信、前端与后端负载平衡和界面与应用分离等优点，是一个强大灵活的webservice工具。

除了基本的显示搜索结果页面外，实现功能包括**无刷新分页、自动补齐、搜索词推荐、结果聚类、网页快照**等功能。功能完善，能带给用户良好的体验。

### 4.1 页面元素

#### 4.1.1 前端布局



#### 4.1.2 主页面



#### 4.1.3 搜索结果页面



### 4.2 自动补齐

#### 4.2.1前端：

当用户输入查询词的时候，会在输入框的下方面动态显示一个下拉框。 同时，Jquery会监听用户在搜索框的每次输入，通过Ajax请求，从服务端获取查询结果，然后结果以JSON的数据类型返回，动态的显示在下拉框中。用户可以选择下拉框中的任意一个词，按回车键即可完成搜索。

#### 4.2.2后端：

Server端使用简单的jsp，在初始化的时候使用关键词词库构建了一颗哈希树，接收到查询请求后，开始查询，将查询结果按照热度进行排序后，以JSON的数据格式返回。最多返回5个单词。

#### 4.2.3 页面展示



### 4.3 搜索词推荐

#### 4.3.1前端

当发现某次搜索结果为空的时候，认为用户可能输入有误，会查找热词词典，返回最相似的最多三个单词，显示在搜索结果页面上，用户可以点击这些词进行搜索。

#### 4.3.2后台

为了将用户的输入和字典里的词相比较，找到相似度最高的单词，采用**N-gram算法和LevensteinDistance编辑距离**。

1. 首先来说一个N-Gram的概念， N-Gram是指将一个单词划分成若干等长的字串，每一个成为一个gram，n就是用来控制每个gram的长度的。

2. 对于一个单词我们往往会给不同的N做多次切割，这样便于做搜索建议和拼写检查。

3. 构建词典

a. 对于每个词典中的单词，根据两个值minN~maxN（minN, maxN是根据单词长度由专门的函数产生的）进行分割，介于两个值之间的N都算作一种分割，伪码如下：

for ng = minN to maxN

for i=0 to wordLen-ng +1

gram = word.substr(i,i+ng);

add this gram as a field to the current lucene document

end for

end for

b. 针对每个N，开头结尾的gram也加入索引域

举例来说：

对于three这个词，minN = 2, maxN =3;那么针对three这个词的对应document对象包含的Field有如下

gram2: th hr re ee

start2: th

end2: ee

gram3: thr hre ree

start3: thr

end3:ree

通过如上的方式将词典里的每个单词都进行了索引.

4. 拼写检查

a. 根据输入word构建查询条件

也是通过minN, maxN来讲输入进行gram化，每个对应的分割都构成一个查询Term，同时针对每种分割也有对应的起始和结束查询，过程和构建词典的过程一样。这些条件之间是或的关系。这个用BooleanQuery可以实现。

b. 开始从词典构建的索引中使用上述条件进行查询

I. 从查询结果中过滤掉自身（如果查询结果包含自身的话）

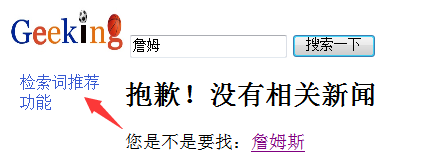
II. 计算查询结果中的每个单词和输入word的编辑距离，过滤掉部分距离过小的单词

III. 放入以编辑距离为衡量指标的优先队列中，队列长度人为确定，具体根据要查询的建议单词的多少确定

IV. 当队列长度满之后，返回结果，注意此处的结果要是以编辑距离增序的结果（使用优先队列的函数可以达到此要求）。

5. 编辑距离解释：编辑距离是衡量两个字符串相似度的值，介于0和1之间，0表示两个字符串完全相同，1表示两个字符串具有最大的不相似程度。常用的编辑距离有：LevensteinDistance和JaroWinklerDistance. 这里使用Levenshtein算法中，两个字符串之间的距离定 义为将一个字符串转换为另一字符串所需的最少编辑次数，允许的编辑操作有插入、删除、单个字符的替换。该算法由Vladimir Levenshtein在1965年提出，并以作者名来命名。

#### 4.3.3 效果展示



### 4.4摘要及高亮

#### 4.4.1前端

摘要目前是静态摘要，将新闻的导语作为摘要显示出来。并且查找到关键词，进行高亮显示，主要使用jquery动态添加标签，实现了高亮效果。

为了防止摘要过长，自动截取一部分显示。

向后台发起Ajax请求，返回分词时候的query列表，使用jquery插件，将title和正文中的query都高亮显示。

#### 4.4.2 后端

将query的结果分词后，接受客户端的Ajax请求，将分词后的列表返回给前端。

#### 4.4.3效果展示

### 4.5 结果聚类

#### 4.5.1 前端

#### 4.5.2 后端

#### 4.5.3 效果展示



### 4.6 网页快照

#### 4.6.1 前端

在“Geeking快照”中加入一个超链接，指向缓存的网页所在的地址。

#### 4.6.2 后台

服务端按照网站类目存放好过滤掉javascript的原始网页，按照目录，拼接成地址，返回到前端。

#### 4.6.3 效果展示

# **三．测试与评估**

🡪陈新荃

总体介绍，测试内容等

## 1. 测试环境

🡪陈新荃

软件、硬件、语料集、

## 2. 测试结果

### 2.1 功能测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目名称** | | Geeking搜索引擎 | **测试方法** | 黑盒,白盒,回归 | |
| **模块开发者** | | 高妍、林裕杰 | **程序版本号** | 0.2 | |
| **功能/模块** | **序号** | **功能点** | **测试用例数** | **发现bug数量** | **修复Bug数量** |
| **Web UI与交互模块** | 1 | 自动补齐功能 | 20 | 10 | 10 |
| 2 | 检索词推荐功能 | 20 | 15 | 15 |
| 3 | 检索结果聚类 | 20 | 12 | 12 |
| 4 | 整体布局调整 | 20 | 10 | 9 |
| 5 | 关键词高亮 | 25 | 10 | 10 |
| 6 | 无刷新分页 | 25 | 5 | 5 |
| 7 | 快照功能 | 25 | 2 | 2 |
| 8 | 聚类结果展开/隐藏 | 25 | 12 | 12 |
| 9 |  |  |  |  |
| **SpellCheck模块** | 1 | N-gram词典构建 | 25 | 12 | 12 |
| 2 | LevensteinDistance编辑距离。 | 25 | 2 | 2 |
| 3 | 关键词推荐处理逻辑 | 25 | 12 | 12 |
| **自测执行者** | | 高妍 | **日期** | 2014/12/9 | |
| **开发负责人** | | 陈新荃 | **日期** | 2014/12/9 | |

### 2.2 性能测试

🡪陈新荃

启动速度、响应速度

# **四．创新点**

## 1. 标题权重

🡪陈新荃

## 2. 中间字自动补全

🡪高妍

## 3. 【还有什么尽管添加】

# **五．经验与总结**

## 1. 个人经验总结

陈新荃：

高妍：

通过这次信息检索大作业，我对搜索引擎有了更深一步的理解，增强了编程能力，对于算法效率、可用性、可扩展性有了进一步的理解。掌握了JSP、Jquery、Ajax、Java等多项技术，深入理解了N-gram算法和LevensteinDistance编辑距离。其中spellcheck模块是通过阅读Lucene源码给了我跟多启发，在此基础上完成的。更重要的是，在团队合作开发的过程中，积累了团队合作经验，学会使用github，特别感谢组长陈新荃和林裕杰、肖卡飞两位队友，我们的队伍里没有大神，大家都是一起努力的菜鸟，在整个开发过程中，面对开发量大，难度高，缺乏经验这些困难，我们相互帮助，相互支持，相互鼓励，都一一克服了。在大家的共同努力下，我们很好的完成了任务。

林裕杰：

肖卡飞：

## 2. 组长总结

Balabala…