Geeking：基于胜者表的体育新闻搜索引擎系统

林裕杰[1,2] 陈新荃[3] 高 妍[4] 肖卡飞[5] 胡红祥[1] 花 强[6]

[1](中国科学院深圳先进技术研究院 深圳 518055)
[2](中国科学院大学深圳先进技术学院 深圳 518055)
[3](中国科学院上海高等研究院 上海 201210)
[4](中国科学院计算技术研究所 北京 100190)
[5](中国科学院沈阳计算技术研究所 沈阳 110168)
[6](河北大学数学与信息科学学院 保定 071002)

摘 要 文章介绍了体育新闻搜索引擎系统 Geeking 的框架结构和各项功能，其结构分为网页爬取、胜者表构建、检索处理、用户界面 4 个部分，其主要功能包含查询词校正、自动补全、检索结果排序、相似新闻聚类以及显示页面中关键词高亮并提供网页快照。输入查询请求时，系统根据搜索日志和新闻热词自动补全查询词，搜索不到相关结果时校正查询，给出推荐的查询词。检索新闻文档时，使用胜者表快速查找查询词项的相关文档，综合 tf-idf 权重和新闻标题、发布时间等因素计算文档的相关性并按得分排序。在相似新闻聚类中，结合最长公共子序列和编辑距离衡量新闻标题之间的相似度，以新闻标题相似度代表新闻文档的相似度。测试结果表明，基于胜者表的 Geeking 搜索引擎系统各项功能协调效果好，检索响应速度快。

关键词 搜索引擎；体育新闻；胜者表；编辑距离；聚类；查询词校正
中图分类号 TP 391.3 **文献标志码** A

Geeking: a Sports News Search Engine System Based on Champion List

LIN Yujie[1,2] CHEN Xinquan[3] GAO Yan[4] XIAO Kafei[5] HU Hongxiang[1] HUA Qiang[6]

[1](*Shenzhen Institutes of Advanced Technology*, *Chinese Academy of Sciences*, *Shenzhen* 518055, *China*)
[2](*Shenzhen College of Advanced Technology*, *University of Chinese Academy of Sciences*, *Shenzhen* 518055, *China*)
[3](*Shanghai Advanced Research Institute*, *Chinese Academy of Sciences*, *Shanghai* 201210, *China*)
[4](*Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)
[5](*Shenyang Institute of Computing Technology*, *Chinese Academy of Sciences*, *Shenyang* 110168, *China*)
[6](*College of Mathematics and Information Sciences*, *Hebei University*, *Baoding* 071002, *China*)

Abstract In this paper, a sports news search engine, Geeking, was introduced, which contains four functional models: web crawling, champion list building, search processing and user interface. Geeking could provide

收稿日期：2015-11-25 **修回日期**：2016-01-13
基金项目：国家自然科学基金(61433012, U1435215, 11171086)；河北省自然科学基金(F2013201064)
作者简介：林裕杰，硕士研究生，研究方向为数据挖掘、算法设计与分析；陈新荃，硕士研究生，研究方向为云计算；高妍，硕士研究生，研究方向为机器学习；肖卡飞，硕士研究生，研究方向为海量数据分析；胡红祥，硕士研究生，研究方向为数据挖掘和推荐系统；花强(通讯作者)，教授，研究方向为神经网络、机器学习，E-mail：huaq@hbu.cn。

query correction, query auto-completion, search results sorting, news clustering, keywords highlighting and snapshot visualization. Given a query, the system automatically completes the query according to the search logs and the news hot keywords. If there was no return of result, the system could correct the query and provided the recommended query terms. The related documents were searched quickly according to the champion list. Based on the tf-idf values and other factors like news headlines and release time, the documents' relevance was calculated. For the clustering of similar news, the longest common subsequence and levenshtein distance were used to measure the similarity between news headlines and the similarity of news headlines could be regarded as the similarity between documents. Test results were given to show that Geeking is fast and stable.

Keywords search engine; sports news; champion list; levenshtein distance; clustering; query term correction

1 引 言

传统搜索引擎没有对不同领域的各个主题设立一个专门的搜索栏目，如百度等搜索引擎有“新闻”这个栏目，但没有在这个栏目下设更加细致的划分，如“体育新闻”、“军事新闻”等。在信息爆炸的当代，人们希望更加准确快捷地获取需要的特定领域的信息，因此搜索需求会逐渐领域化、个性化[1]。用于搜索特定领域或主题的搜索引擎称之为垂直搜索引擎[2]，Geeking 就是体育新闻领域的垂直搜索引擎。Geeking 由 Java 语言编写，能够爬取各大门户网站的体育新闻(包括搜狐、腾讯、网易、MSN 等)，对搜索结果按相关度排序，并对相似新闻聚类，以及实现查询词校正、自动补全、关键词高亮、网页快照等功能。

对搜索引擎的评价和优化需要结合用户行为的分析[3]。研究表明，85% 的搜索引擎用户只查看返回结果的第一页[4]。这一方面说明搜索引擎对返回结果排序相当重要，另一方面也说明用户不需要全部的相关文档。因此，在不让用户感受到前 K 个结果的相关度有所降低的基础上，采用非精确 top K 机制，其在保证返回和真实的 top K 文档得分非常接近的 K 篇文档的前提下，相比精确 top K 机制能够显著降低计算复杂度[5]。Geeking 搜索引擎使用胜者表(Champion List)可实现非精确 top K。

信息检索文档得分经典的计算方法是基于 tf-idf 的向量空间模型 VSM(Vector Space Model)，其中，tf 代表词项频率(term frequency)；idf 代表逆文档频率 (inverse document frequency)；tf-idf 代表二者的乘积。在向量空间坐标系里，文档的每一维用一个特征词表示，文档的相似度用向量之间的余弦夹角来衡量[6]。Geeking 搜索引擎针对体育新闻的特点，在 tf-idf 的基础上增加了新闻标题、发布时间等影响因素来计算查询和文档的相关度。

文档的聚类在计算相似度时一般需要比较文档的主要内容，Geeking 根据对百度等搜索引擎相似新闻聚类的调研结果，以新闻标题的相似度代表文档之间的相似度，提高了计算效率。短文本相似度计算方法中，按照所依据特征大致分为 3 类：基于字面相似、基于统计关联和基于语义相似。常用的基于字面相似的方法有计算最长公共子序列(Longest Common Subsequence，LCS)和编辑距离(Levenshtein Distance，LD)等，其中，编辑距离算法常用于字符串的模糊匹配，是一种经典而广为使用的方法[7]。本文基于最长公共子序列 LCS 和 LD 编辑距离计算新闻标题的相似度，并给出了一个合理的计算相似度值的式子。

传统的自动补全功能通过字典树实现，只

能按照前缀匹配，而 Geeking 的自动补全功能基于哈希树实现了查询词的任意段的匹配，且用 *k*-gram 实现了查询词校正，提高了用户体验。

2　网页爬虫

2.1　爬虫模块

本搜索引擎系统的爬虫模块 GeekSpider 基于开源 JAVA 爬虫 WebCollector 实现。WebCollector 是一个便于二次开发的 JAVA 爬虫框架(内核)，它提供精简的 API(Application Programming Interface，应用程序编程接口)，只需少量代码即可实现功能强大的爬虫[8]。GeekSpider 在此内核上采用基于广度优先算法的多线程网页获取策略，并通过一定正则条件约束，剔除获取到本地的网页的图片信息，Script 脚本信息等。最终以网页的 url 命名保存在本地，其中“:”用“$”代替，“/”用“#”代替，如：http$##sports.163.com#04#0918#23#10JMFUK400051CA0.html。

2.2　爬虫流程

爬虫流程如图 1 所示。

以爬取 Sports163 为例，流程如下：

(1)继承 WebCollector 中的 BreadthCrawler 父类，派生出 Spider163 类；

(2)设定爬取种子入口，一般把爬虫的种子设为网站的首页；

(3)定义正则约束条件，给定爬取约束范围并且过滤网页中的图片格式信息；

(4)爬取参数的设定(断点爬取、爬取深度、爬取线程数等)；

(5)自定义 visit 函数访问获取的 URL 页面信息，并通过自定义的 CreatHtml 类的 outputFile 方法把网页信息保存在指定目录下，其中在保存本地的过程中剔除不必要的 Script 脚本信息，为快照的实现减小文本规模。

按照类似方法，获取 MSN、QQ、Sohu 等

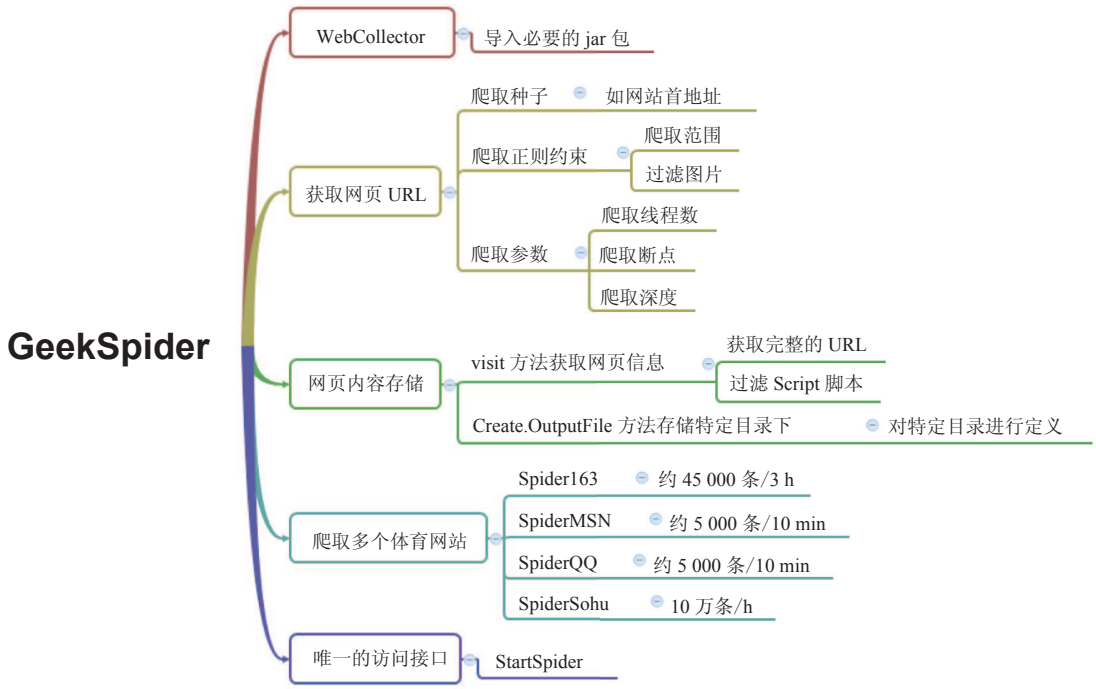

图 1　爬虫流程

Fig. 1　Web crawling process

门户网站的体育新闻信息，过滤掉 url 相同的网页，将原始网页存储在文件系统中。如 2.1 节所述，原始网页以 html 文件形式存储，以其 url 命名，并以分类目录方式存储，具体如图 2 所示。

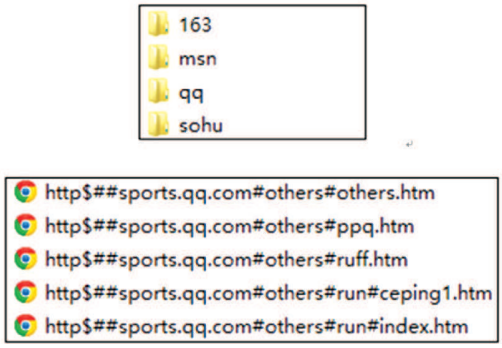

图 2 原始网页文件列表

Fig. 2 Directory structure of the raw pages

3 胜者表构建

3.1 构建流程

胜者表是索引表构建的最后一个步骤，流程如下：

(1) 在 Mysql 数据库中创建相关表格，包括：PagesIndex(网页信息表)、ForwardIndex(正向索引表)、TermsIndex(词项 ID 映射表)、InvertedIndex(倒排索引表)和 ChampionList(胜者表)，原始网页仍存储在文件系统中；

(2) 读取原始网页文件，提取网页标题、关键字等信息写进网页信息表 PagesIndex；

(3) 提取网页正文进行分词，并根据分词结果写进词项 ID 映射表 TermsIndex 和正向索引表 ForwardIndex；

(4) 读取正向索引表，合并相同词项，记录词项在文档中的位置信息，统计 tf-idf，写进倒排索引表 InvertedIndex；

(5) 读取倒排索引表，对每个词项的倒排索引记录按 tf-idf 值由高到底排序，截取出前 r 个(r 根据 idf 值而定)文档信息，写进胜者表 ChampionList。

3.2 网页信息提取

网页信息提取部分采用开源第三方工具 htmlparser[9]，并结合自定义的正则表达式，加以封装，实现对网页信息的提取，包括 title(标题)、description(导语)、keywords(关键词)、public time(发布时间)以及网页正文等。其中搜狐、网易、腾讯、MSN 四大门户网站的新闻网页中的 title、description 和 keywords 内容均可以通过相同的网页标签定位并获取。但是对于 public time，四大网站的标签均不同，并且对于同一个网站，其不同年份的 public time 标签也都不同，因此此处并未采用 htmlparser 来提取，而是使用了自定义的正则表达式。

原始网页经过提取出标题等信息后写进数据库中的网页信息表 PagesIndex，其存储结构如表 1 所示。PagesIndex 各属性介绍如下。

文档 ID：主键；

URL：候选键，与文档 ID 一一对应；

标题：用于网页排序、相关度计算和结果显示；

关键字：用于热词推荐；

日期：网页发布日期，用于网页排序、结果显示；

类型：门户网站类型，用于快照时根据类型进入原始网页文件目录读取文件；

摘要：用于高亮显示。

表 1 网页信息表存储结构

Table 1 PagesIndex storage structure

属性名	实例	类型
文档 ID	0	int
URL	http$##msn.spprts.ynet.com#2.1.0#27778_3.html	varchar
标题	精彩评论_体育_MSN 中国	varchar
关键字	[中国，军团，北京，奥运会，姚明]	varchar
日期	2014-11-10	varchar
类型	msn	varchar
摘要	深刻的解读，独特视角看奥运	text

3.3　网页正文分词并映射词项 ID

3.3.1 分词方法和词典大小

文本分词应用于信息检索、机器翻译、问答系统、文本分类等领域，分词准确率对自然语言处理系统的效果有重要的影响。相关研究表明运用分类联合模型能提高分词速度，而融合模型能取得更好的准确率[10]。Geeking 搜索引擎系统中的分词模块使用第三方开源中文分词工具 Ansj。Ansj 是基于中国科学院的 ictclas 中文分词算法实现的，比其他常用的开源分词工具(如 mmseg4j)的分词准确率高。Ansj 主要分词方法有：基本分词(BaseAnalysis)、精准分词(ToAnalysis)、nlp 分词(NlpAnalysis)、面向索引的分词(IndexAnalysis)[11]。四种方法的对比情况如表 2 所示。

表 2　Ansj 四种分词方法功能比较

Table 2　Feature comparison for 4 segmentation methods of Ansj

分词方法	用户自定义词典	数字识别	人名识别	机构名识别	新词发现
BaseAnalysis	No	Yes	No	No	No
ToAnalysis	Yes	Yes	Yes	No	No
NlpAnalysis	Yes	Yes	Yes	Yes	Yes
IndexAnalysis	Yes	Yes	Yes	No	No

考虑到 BaseAnalysis 功能单一，因此不采用该方法。而对于 IndexAnalysis，特点是充分考虑了歧义句的因素，比如对“主副食品”分词，能够产生五个分词结果：“主副食品”、“主副食”、“副食”、“副食品”、“食品”。鉴于其分词结果会派生出很多新词，极大增加倒排索引的规模，因此不在本搜索引擎系统中采用。NlpAnalysis 虽功能强大，能够支持新词发现，但是执行时间相对较长，并且非常消耗内存。而 ToAnalysis 在易用性、稳定性、准确性、以及分词效率上都取得了一个不错的平衡，也是作者亲自推荐的方法。综合上述考虑，本搜索引擎系统索引构建过程对文本的分词采用 ToAnalysis 方法，为保持一致性，对查询语句的分词也采用 ToAnalysis。

Ansj 默认分词词典有 386 211 个词项，约 6 MB。停用词词典需要自行添加，本搜索引擎系统添加的停用词有 1 534 个，共 10.2 KB。

3.3.2 映射词项 ID

将文档正文进行分词并过滤停用词之后，将词项映射成词项 ID，写进词项 ID 映射表 TermsIndex，如表 3 所示。

表 3　词项 ID 映射表存储结构

Table 3　TermsIndex storage structure

属性名	实例	类型
词项 ID	11	int
词项	体育	varchar

3.4　构建正向索引表和倒排索引表

根据分词结果，构建正向索引表，记录网页文档中依次出现了哪些词项，其存储结构如表 4 所示。正向索引表包含属性文档 ID 和词项 ID 序列，其中词项 ID 序列字段用“#”分隔各个词项 ID。

表 4　正向索引表存储结构

Table 4　ForwardIndex storage structure

属性名	实例	类型
文档 ID	1	int
词项 ID 序列	11#2#13…	text

读取正向索引表，合并相同词项，记录词项在文档中的位置信息，统计词项频率 tf 和逆文档频率 idf，计算其乘积 tf-idf，写进倒排索引表。倒排索引记录了每个词项出现在哪些网页文档中，以及针对每篇文档的 tf-idf 权重，其存储结构如表 5 所示。倒排索引表中包含两个属性，词项 ID 和文档 ID 序列，文档 ID 序列用“#”分隔

表 5　倒排索引表存储结构

Table 5　InverteIndex storage structure

属性名	实例	类型
词项 ID	1	int
文档 ID 序列	2|2.73|2,13,25,62#4|5.85|5,18,40,81#……	text

若干个文档 ID 信息，序列格式为：文档 ID|tf-idf 值|词项位置列表#。倒排索引构建速度快，在搜索引擎的索引方法中占据十分重要的地位[12]。

词项频率 tf 指的是某个给定的词语在给定的文档中出现的频率，该值是对词数(term count)的归一化，以防止它偏向长文本。其计算公式如下：

$$tf_{t,d}=\frac{n_{t,d}}{\sum_k n_{k,d}} \tag{1}$$

其中，$tf_{t,d}$ 表示词项 t 在文档 d 的词项频率；分子是词项 t 在文档 d 中出现的次数；而分母则是文档 d 中所有词项的出现次数之和。

逆文档频率 idf 度量的是词项的罕见程度，词项 t 的 idf 值等于总文档数目除以包含该词项的文档的数目，再将得到的商取对数，公式如下：

$$idf_t=\log\frac{|D|}{|\{j:t\in d_j\}|} \tag{2}$$

利用 tf 能获取特定文档内的高频词项，利用 idf 可以获取文档集的罕见词项，因此，tf 和 idf 的乘积 tf-idf 权重倾向于过滤掉常见的词项，保留重要的词项。

3.5 构建胜者表

倒排索引表每个词项 ID 对应的文档 ID 序列太长，包含了许多 tf-idf 权值较小的文档，在计算文档相关度时时间复杂度太大，因此该系统采用胜者表机制。胜者表也称为优胜表(fancy list)或高分文档(top docs)。

读取倒排索引表，对每个词项的倒排索引记录按 tf-idf 值由高到底排序，截取出前 r 个文档 ID 信息(r 根据词项 idf 值而定，且设定 $r\leqslant 1\ 000$)，写进胜者表 ChampionList，其存储结构如表 6 所示。胜者表和倒排索引表不同的地方在于胜者表只记录词项的倒排索引表中 tf-idf 权重最高的前 r 个文档 ID，故在检索过程相对需要计算相关度的文档数目少，时间复杂度低。采用胜者表的检索机制时，在得到查询语句分词后的查询词项后，对这些查询词项在胜者表中的索引文档做相关度计算，从中选出 top K 篇文档作为返回结果。胜者表属于非精确 top K 的一种方法，非精确 top K 机制在保证返回结果和精确 top K 相差不大的前提下，显著降低了计算复杂度。

表 6 胜者表存储结构

Table 6 Storage structure of ChampionList

属性名	实例	类型
词项 ID	1	int
tf-idf 最大的前 r 个文档	4|5.85|5,18,40,81#2|2.73|2,13,25,62#……	text

4 检索处理

4.1 检索流程

启动服务器，初始化搜索服务，包括初始化配置和数据库、加载停用词词典、热词词典和词项 ID 映射表到内存，如图 3 所示。数据库中其他表格无法全部加载进内存，否则内存溢满。

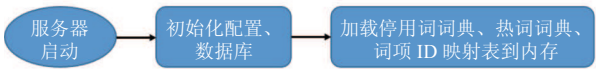

图 3 服务器初始化过程

Fig. 3 Server initialization process

系统检索的过程从服务器初始化之后开始，检索流程如图 4 所示。

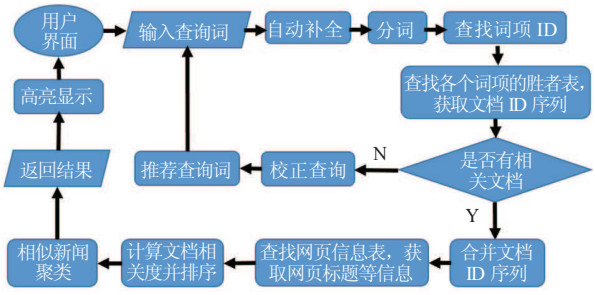

图 4 系统检索流程

Fig. 4 System retrieval process

(1)用户输入查询词；

(2)根据热词词典和历史搜索记录对查询词自动补全；

(3) 对查询词分词；

(4) 根据分词结果从词项映射表查找词项 ID；

(5) 根据词项 ID 查找胜者表，获取各个查询词项对应的文档 ID 序列；

(6) 如果查找不到相关文档，执行查询词校正，查找相似词项，并将推荐的查询词返回给前端等待下一轮查询；

(7) 将文档 ID 序列按照其包含的文档数目升序排列做合并操作；

(8) 查找相关文档集合中每篇文档的网页信息表，获取文档的标题、发布时间等信息；

(9) 计算相关文档得分并排序；

(10) 对排序后的相关文档集聚类，将聚类后的最终结果返回给前端；

(11) 前端显示搜索结果，且用红颜色高亮显示网页信息中与查询词一致的词项。

4.2　查询词自动补全

传统的自动补全功能是通过字典树实现的，只能按照前缀匹配，Geeking 的自动补全功能基于哈希树实现了查询词任意位置字段的匹配。哈希树根据热词词典构造，热词词典包含新闻热词和用户搜索记录，系统初始化时查找网页信息表的“关键字”字段生成热词词典，之后根据用户搜索记录持续添加用户的搜索热词。前端 Jquery 会监听用户在搜索框的每次输入，通过 Ajax 请求，Server 端根据哈希树查找补全结果，将补全结果按照热度进行排序，以 JSON 数据格式返回。

4.3　查询词校正

当搜索不到用户查询的相关结果时，认为用户可能输入有误，执行查询词校正，搜索相似词项，并将推荐的查询词返回给前端。为了将用户的输入和字典里的词相比较，找到相似度最高的词项，采用 *k*-gram 算法和编辑距离 Levenstein Distance。

4.3.1 构建 *k*-gram 索引

k-gram 是指将一个词项划分成若干等长为 *k* 的字串，每一个成为一个 gram，*k* 就是用来控制每个 gram 的长度。对于一个词项我们往往会给不同的 *k* 做多次切割，这样便于做搜索建议和拼写检查。对词典中的每个词项，根据预定值 k_{min} 和 k_{max}（k_{min} 和 k_{max} 根据词项长度等因素决定），介于两个值之间的 *k* 值都作一类分割，并且将开始和结尾的字符“$”也加入索引域，“$”是一个特殊字符，表示词项开始或结束。举例来说：对于“詹姆斯”这个词，若 $k_{min}=2$，$k_{max}=3$，那么索引到“詹姆斯”这个词的所有 *k*-gram 如下：

gram2: $ 詹，詹姆，姆斯，斯 $

start2: $ 詹

end2: 斯 $

gram3: $ 詹姆，詹姆斯，姆斯 $

start3: $ 詹姆

end3: 姆斯 $

通过如上的方式将词典里的每个词项都进行了 *k*-gram 索引。

4.3.2 查找校正词

LD 编辑距离用来度量词项的相似性：在将一个字符串转换为另一个字符串时，采用的基本操作是单字符插入（insert）、删除（delete）和替换（replace），每个基本操作增加 1 个单位编辑距离[13]。如“cute”和“cat”的 LD 为 2，因为“cute”转换为“cat”至少需要 2 个编辑步骤：通过 replace 操作将“u”替换为“a”变成“cate”，再经过 delete 删除“e”变成“cat”。

(1) 根据输入的查询词构建查询条件

根据 k_{min} 和 k_{max} 对查询词做 *k*-gram 切割，每个分割的 gram 都构成一个查询词项 term，每个 term 为 1 个查询条件，它们之间是逻辑或的关系，用 Boolean query 可以实现。

(2) 查找 *k*-gram 索引并计算编辑距离

根据编辑距离查找校正词的算法如下：

①使用 (1) 的查询条件从 *k*-gram 索引中进行查询对应词项；

②从查询结果中过滤掉自身(如果查询结果包含自身的话)；

③计算查询结果中的每个词项和输入词的编辑距离，过滤掉距离大于阈值的词项；

④把满足阈值要求的校正词放入以编辑距离为衡量指标的优先队列中，队列长度预先设定；

⑤当队列长度满了之后，返回结果，结果以编辑距离升序排列。

4.4 合并相关文档 ID 序列

在得到各个查询词项 ID 后，查找各个查询词项在胜者表的文档 ID 序列记录，将各个文档 ID 序列做合并操作，合并前按照其包含的文档数目升序排列，能减少合并计算量。合并后的相关文档 ID 序列依然保留其 tf-idf 等信息。

4.5 计算文档相关度得分

本搜索引擎系统基于 tf-idf 权值计算查询语句和相关文档的相关度，并且根据体育新闻的时效性，增加了时间权重。以及考虑到新闻标题和关键字在新闻网页占据相对重要的地位，认为如果查询词出现在某篇文档的标题或者关键字中，则该文档的相关度相对更高。因此我们采取的相关度计算公式为

$$score(q,d)=\sum_{i=1}^{|q|}(tf_{i,q}\times tf\text{-}idf_{i,d}+10\times ctk_{i,d})+w(d,curMill) \tag{3}$$

其中，$score(q,d)$ 表示文档 d 对查询 q 的相关性得分；$|q|$ 表示查询向量 q 包含的词项数目；$tf_{i,q}$ 表示 q 的第 i 个词项在 q 中的 tf 值；$tf\text{-}idf_{i,d}$ 表示 q 的第 i 个词项在文档 d 中的 tf-idf 值；$ctk_{t,d}$ 是指查询词 t 在标题和关键字中出现的次数，每出现一次，权重增加 10，如果查询语句在标题中出现的词项越多，那么权重分数将会更高。$w(d,curMill)$ 是时效性权重，计算公式为

$$w(d,curMill)=\left(\frac{curMill}{curMill-pubMill_d}\right)^2 \tag{4}$$

其中，$curMill$ 指从 Epoch(1970 年 1 月 1 日 00:00:00 UTC)开始到现在所经过的秒数；$pubMill_d$ 指从 Epoch 开始到网页发布那天所经历过的秒数。公式表明，如果网页发布时间距离用户搜索当天越近，那么权重分数将会更高，反之更低。

可见，计算文档相关度得分不仅仅需要 4.4 节得到的相关文档 ID 序列中的 tf-idf 信息，还需要根据文档 ID 读取网页信息表，获取网页的标题、关键字、发布时间等信息。计算完文档得分后按得分进行排序。

4.6 相似新闻聚类

4.6.1 体育新闻聚类调研

对同一个主题的新闻报道，若内容相似度极高的新闻，新闻搜索引擎会有“相同新闻”字样的链接。该系统的体育新闻聚类是指对这些“相同新闻”的聚类。如图 5 右下角，红色矩形框内的“35 条相同新闻”是百度对该新闻与之高度相似的新闻的聚类。通过对百度等搜索引擎相似新闻聚类结果的分析，得出如下结论：

(1)“相同新闻”列表的新闻网页排列次序和其未聚类前的相对顺序相同。同类新闻内的排序依据包括发布时间、关键字等，这些排序依据和搜索引擎对全部搜索结果的排序依据一致；

(2)高度相似的新闻之间的标题相似性高；

(3)相似性高的标题其对应的新闻也高度相似。

根据结论(1)，网页聚类前应该已经按照相关度排序了。

图 5 百度新闻搜索结果聚类示意

Fig. 5 Clustering results for searching news by Baidu

4.6.2 体育新闻相似度计算

根据上面的体育新闻聚类调研新闻结论(2)和结论(3)，我们用新闻标题相似度代表新闻网页的相似度。且考虑到新闻标题是短文本，若使用基于 tf-idf 的向量空间模型效果不佳，短文本基于字面相似的相似性计算方法常用的是计算 LCS 和 LD 编辑距离。注意 LCS 和最长公共子串(要求连续)是不同的，比如“cute”和“cat”的最长公共子串为“c”或“t”，长度为 1，而 LCS 为“c, t”，长度为 2。 LD 参见 4.3.2 节的介绍。

标题 A 和标题 B 的相似度 $sim(A,B)$ 计算如下式，其中 $LCS(A,B)$ 表示 A 和 B 的 LCS 长度，$LD(A,B)$ 表示 A 和 B 的 LD 编辑距离。

$$sim(A,B)=\frac{LCS(A,B)}{LCS(A,B)+LD(A,B)} \tag{5}$$

4.6.3 体育新闻聚类过程

根据调研，体育新闻类簇内各个新闻相似性极高，而直观上，一个类簇内部点间距离跟类簇外其他点距离相比更小[14]，故假定体育新闻每个类簇的直径小于类簇之间的距离。为减少时间复杂度，可以设定一个相似度阈值，将一个类簇的新闻聚完后，再从剩下的新闻中去聚下一个类，如此反复直到新闻全部聚类完成。算法描述如下：

(1)输入数据集为搜索结果根据相关度排序的 N 篇文档 *pageList*；

(2)从 0 开始依次遍历 *pageList* 取其第 k 篇网页 *pageList*[k]，构造 1 个新数组 *claster*，将 *pageList*[k] 放入 *claster*；

(3)从 $k+1$ 开始依次取 *pageList* 中第 i 篇网页 *pageList*[i]，计算 *pageList*[k] 和 *pageList*[i] 的标题相似度；

(4)若相似度满足阈值要求，则将 *pageList*[i] 加入 *claster*，并从 *pageList* 删除 *pageList*[i]；

(5)判断步(2)是否遍历完 *pageList*，否则回到步(2)，是则将已经完成的 *claster* 加到 *result* 中；

(6)判断步(1)是否遍历完 *pageList*，否则回到步(1)，是则输出 *result*。*result* 包含若干个新闻类簇 *claster*。

在输入的新闻数据集大小为 N，聚成的类簇个数为 C 时，上述聚类算法的时间复杂度为 $O(CN)$，通常情况下 C 小于 N，该算法与时间复杂度为 $O(N^2)$ 的单连接聚类算法相比，时间复杂度低，且不会出现单连接算法的链式效应。该聚类算法至多只需要计算 $\frac{N\times(N-1)}{2}$ 个文档对。这是因为该算法时间复杂度最差的情况出现在聚成的类簇个数等于 N 时，则第 k 篇文档需要计算 $N-k$ 篇文档的相似度，总共需要计算的文档对则为：

$$(N-1)+(N-2)+\cdots+0=\frac{N\times(N-1)}{2} \tag{6}$$

5　前端处理

前端总体框架采用 Ajax+Jquery+JSP 的客户端方式，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。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、查询词校正、任意位置字段自动补全、结果聚类、网页快照等，能带给用户良好的体验。

5.1　页面元素

前端用户界面的布局结构如图 6 所示，搜索主页面如图 7 所示。

5.2　自动补全的前端处理

自动补全的后端处理方法见 4.2 节介绍。在前端当用户输入查询词时，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，查询结果以 JSON 的数据类型返回，动态地显示在下拉框中。用户可以选择下

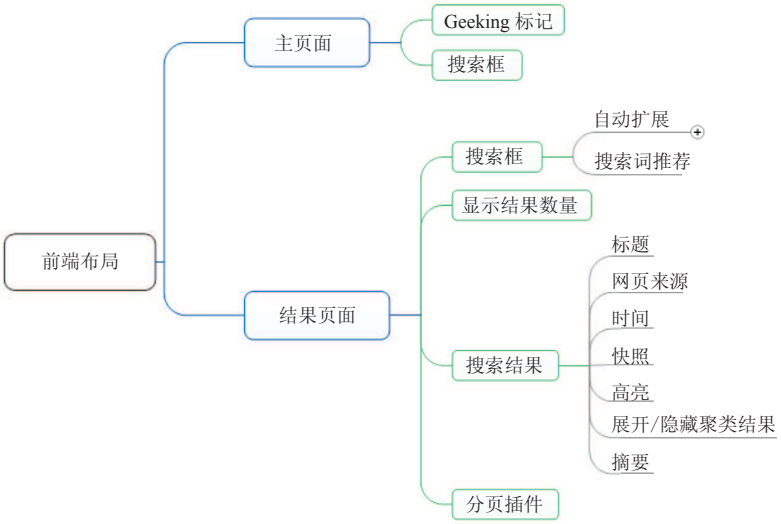

图 6 前端布局结构

Fig. 6 Front page layout structure

图 7 主页面

Fig. 7 Main page

拉框中的任意一项完成搜索，如图 8 所示：用户输入“姆”，自动补全出现“詹姆斯”和“詹姆斯 骑士”。

图 8 查询词自动补全

Fig. 8 Query term auto-completion

5.3 查询词校正的前端处理

查询词校正后端处理方法见 4.3 节。如图 9 所示，用户搜索“詹姆”，搜索结果为空，给出推荐的查询词“詹姆斯”，点击查询词链接可以直接完成新一轮搜索。返回的校正词至多显示 3 个。

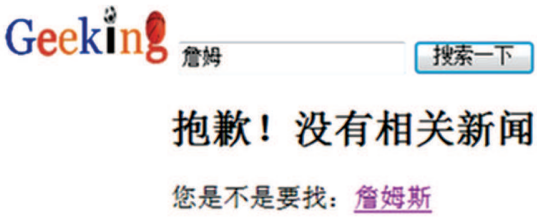

图 9 查询词校正

Fig. 9 Query term correction

5.4 返回结果显示处理

返回结果显示页面如图 10 所示，前端通过 Ajax 请求，得到查询结果后，显示如下信息：

(1) 相关新闻总数目；

(2) 网页标题、摘要、门户来源、发布时间；其中摘要是静态摘要，自动截取新闻的导语显示出来；标题和摘要中与查询词一致的词项用红色高亮显示，通过 jquery 动态添加标签而成；

图 10　搜索结果高亮显示

Fig. 10　Hightlight search results

(3) 聚类结果，点击图中的“显示*条相同新闻”链接会展开该类别的其余新闻；

(4)“Geeking 快照”，链接根据网页 url 指向存储在本地的原始网页文件。

6　测试

本项目测试主要分为两个部分：功能测试和性能测试。其中功能测试包括了黑盒、白盒及回归测试，涵盖了项目开发过程中主要模块的功能测试以及 bug 修复统计。性能测试包括了索引构建、服务启动和检索过程中分词、排序、聚类等每一步的时间开销。

6.1　测试环境和数据集

(1) 硬件环境

CPU：Intel Core（TM）i7-4500U 2.40GHz

Memory：8 GB DDR3

硬盘：机械硬盘

(2) 软件环境

操作系统：Windows 8.1 Enterprise

Web 服务器： apache-tomcat-8.0.23

数据库：Mysql 5.1

JDK：1.8.0

(3) 数据集

网页数据来源数目如表 7 所示。

表 7　网页数据集

Table 7　Web page data set

搜狐	网易	腾讯	MSN	总计
45 831	15 669	37 622	5 791	104 913

数据表存储规模如表 8 所示。

表 8　数据库表规模

Table 8　Database size

数据表	记录条数
网页信息表	61 725
正向索引表	61 725
词项 ID 映射表	291 402
倒排索引表	283 587
胜者表	283 587

6.2　检索功能测试

功能测试方法为黑盒、白盒及回归。此处功能测试并不基于上述测试环境，而是整个系统开发过程中主要功能基本实现之后，进行功能测试时所遇到的 bug 和修复统计，具体见表 9。测试结果表明搜索引擎的各项功能均能正常使用。

表 9　功能测试

Table 9　Function test

序号	功能	测试用例数	Debug 完成
1	自动补全	150	Yes
2	查询词校正	150	Yes
3	检索词推荐	150	Yes
4	相关文档查找	100	Yes
5	检索结果排序	100	Yes
6	检索结果聚类	100	Yes
7	聚类结果展开/隐藏	100	Yes
8	关键词高亮	100	Yes
9	无刷新分页	100	Yes
10	快照功能	100	Yes
11	整体布局调整	--	Yes

基于上述所示的测试环境和数据集，以用户输入查询词“中国奥运冠军”为例，返回 349 篇相关文档，检索过程所耗费的时间如表 10 所示。测试结果表明系统响应速度较快。

上例的整个检索过程花费时间为 1.422 秒，响应速度较为理想。其中步骤 2 和步骤 3 占用时

间最多，因为步骤 2 和步骤 3 分别涉及查询数据库胜者表和网页信息表，根据 6.1 节的数据集显示，数据库中胜者表有 283 587 条记录，网页信息表有 61 725 条记录，数据库表规模越大，查询时间越久。而步骤 1 虽涉及词项 ID 的映射，但由于系统初始化时已经将词项 ID 映射表读进内存，故此步骤不会花费太多时间。

表 10 检索流程和耗费时间

Table 10 Retrieval process and the time-consuming

序号	检索流程	时间(ms)
1	分词并映射词项 ID	31
2	查找相关网页	879
3	查找网页信息	326
4	计算网页相关度	52
5	网页排序	35
6	网页聚类	98
7	前端高亮显示	1
合计		1 422

7 总结与展望

7.1 论文总结

本搜索引擎系统具备一般搜索引擎的主要功能，算法设计较为高效。检索算法设计中，运用哈希树自动补全任意位置字段的查询词，且基于 *k*-gram 实现了查询词校正；使用了胜利表加快 top *K* 相关文档的查找速度；根据新闻特点，排序基于 tf-idf 权重并增加了新闻标题和发布时间等因素来计算相关度；相似新闻聚类算法计算相似性时结合了 LCS 和 LD，时间复杂度小于单连接算法；前端显示结果全面，并高亮显示网页信息中与查询词一致的词项。经测试，该系统各项功能整体协调效果良好，检索响应速度快。

7.2 展望

测试结果反应了数据库的数据集越大，查询时间越久，因此如果在更大量级的数据集下，数据库的查询将会消耗更多时间，这将是影响该系统搜索响应速度的重要因素。本系统需要改进使用分布式计算系统如 spark 来部署整个搜索引擎系统，以提高大规模数据集的处理能力。并且，在大规模数据集下，检索过程中使用到的处理方法，如自动补全和查询词校正等需要相应做出改进，优化其计算复杂度。另外，本搜索引擎的相关文档排序还未考虑用户的搜索反馈，后面会根据用户的搜索反馈行为做出搜索优化。

参 考 文 献

[1] 冯竣佷. 基于特定领域元搜索的网页排名算法研究 [D]. 上海: 华东师范大学, 2013.

[2] 杜雷. 垂直搜索引擎网络爬虫的研究与设计 [D]. 北京: 北京邮电大学, 2015.

[3] 邢千里. 搜索引擎用户点击模型研究 [D]. 北京: 清华大学, 2014.

[4] Oudinet J. Search Engine Ranking [EB/OL]. [2015-11-10]. http://www.lrde.epita.fr/dload/20060524-Seminar/oudinet-search-engine-ranking.pdf.

[5] Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval [M]. Cambridge: Cambridge University Press, 2008.

[6] Croft B, Metzler D, Strohman T. Search Engine: Information Retrieval in Practice [M]. New Jersey: Addison Wesley, 2009: 237.

[7] 姜华, 韩安琪, 王美佳, 等. 基于改进编辑距离的字符串相似度求解算法 [J]. 人工智能及识别技术, 2014, 40(1): 222-223.

[8] 开源中国. JAVA 爬虫 WebCollector [EB/OL]. [2015-06-08]. http://www.oschina.net/p/webcollector.

[9] Oswald D. HTML Parser [EB/OL]. 2006-09-17 [2015-06-20]. http://htmlparser.sourceforge.net/.

[10] 刘一佳, 车万翔, 刘挺, 等. 基于序列标注的中文分词、词性标注模型比较分析 [J]. 中文信息学报, 2013, 27(4): 31-35.

[11] 孙健. Ansj 中文分词 [EB/OL]. [2014-12-08]. https://github.com/NLPchina/ansj_seg.

[12] Liu B. Web 数据挖掘 [M]. 韩定一译. 北京: 清华大学出版社, 2009: 150-151.

[13] Cormen TH, Leiserson CE, Rivest RL, et al. 算法导论 [M]. 潘金贵, 顾铁成, 李成法, 等译. 北京: 机械工业出版社, 2006: 208-212.

[14] Bishop CM. Pattern Recognition and Machine Learning [M]. Germany: Springer, 2007.