

# 基于改进排序算法的用户查询优化的研究

吴家皋<sup>1,2</sup>, 刘杰<sup>1,2</sup>, 钱科宇<sup>1,2</sup>, 李云<sup>1,2</sup>

(1. 南京邮电大学 计算机学院, 江苏 南京 210003;  
2. 江苏省无线传感网高技术研究重点实验室, 江苏 南京 210003)

**摘要:**互联网的迅速发展使信息检索的环境发生了重大变化。而基于互联网的搜索引擎的排序算法直接关系到用户在新的环境里进行信息检索的使用体验。文中提出一种将 PageRank 算法、分类技术、文档 TF-IDF (词频-逆向词频) 值相结合的方法, 对排序算法进行改进。该算法对于用户查询的关键词进行预分类, 判断用户的输入关键字最可能属于的文本类型。基于此优先从 Solr 库中取出类别相似的数据, 使得主题相关的文本靠前显示。实验结果表明, 该排序算法具有较快的查询响应时间和较高的查准率。

**关键词:**搜索引擎; PageRank; 排序; 分类; 词频-逆向词频

中图分类号: TP301

文献标识码: A

文章编号: 1673-629X(2015)07-0049-05

doi: 10.3969/j.issn.1673-629X.2015.07.011

## Research on User's Query Optimization Based on Improved Sorting Algorithm

WU Jia-gao<sup>1,2</sup>, LIU Jie<sup>1,2</sup>, QIAN Ke-yu<sup>1,2</sup>, LI Yun<sup>1,2</sup>

(1. College of Computer, Nanjing University of Posts and Telecommunications,  
Nanjing 210003, China;  
2. Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks,  
Nanjing 210003, China)

**Abstract:** The rapid development of the Internet makes information retrieval environment has undergone major changes. The ranking algorithm based on Internet search engine directly influences the user experience in a new environment for information retrieval. In this paper, an improved sorting algorithm was proposed in which the PageRank algorithm, classification techniques, documentation TF-IDF (Term Frequency-Inverse Term Frequency) values were combined to improve the sorting algorithm. The keywords of the user's queries are pre-classified to predict which class are the user's text input keywords most likely belong to. Similar data are taken from Solr library based on this, making the front display text relevant to the subject. Experiments results show that the sorting algorithm has faster query response time and high precision.

**Key words:** search engine; PageRank; sort; classification; TF-IDF

## 0 引言

随着计算机系统性能的提高和网络技术的不断进步, 万维网得到了蓬勃发展, 成为全球最大的信息资源库。用户要在如此庞大杂乱的万维网资源中查找所需要的信息, 就像大海捞针一样, 搜索引擎技术恰好解决了这一难题。搜索引擎是基于万维网平台, 提供网络信息检索服务的工具。用户给出关键词作为查询请

求, 搜索引擎在万维网上收集、整理信息, 并且按照用户需求返回相关的查询结果, 帮助人们拒绝和忽略大量无关信息, 从而起到信息导航的作用。然而由于当今搜索引擎相关性排序算法并不完善, 用户通常需要从大量的返回结果中手工挑选相关网页, 搜索引擎的导航功能没有发挥明显优势。

目前大多数搜索引擎仍以 PageRank<sup>[1-2]</sup> 等经典算

收稿日期: 2014-08-26

修回日期: 2014-11-28

网络出版时间: 2015-06-23

基金项目: 国家自然科学基金资助项目(61373139); 江苏省自然科学基金(BK2012833); 江苏省高校自然科学基金(12KJB520011); 南京邮电大学科研基金(NY213160)

作者简介: 吴家皋(1969-), 男, 副教授, 博士, CCF 会员, 研究方向为计算机网络、GIS 应用等。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20150623.1031.027.html>

法为基础,进行改良,加入各自偏重的参数形成综合的排序模型。文献[3-6]考虑使用主题相关参数来对 PageRank 值进行改进,但其不能解决 PageRank 偏向于旧网页的特性。文献[7]基于链接分析来改进排序,其忽略了网页之间内容之间的相似度。文献[8]基于用户偏好进行预测,判断用户的下一个浏览网页,这种方法具有一定的局限性,不能反映整个 Web 的实际情况。

文中针对上述状况提出将传统 PageRank 算法、分类技术<sup>[9-10]</sup>、TF-IDF<sup>[11]</sup>值相结合的方法,来优化排序算法。传统 PageRank 算法具有偏于旧网页和主题漂移的现象,文中运用分类技术来解决主题漂移的现象,结合分类和 TF-IDF 值来解决其偏于旧网页的问题。对于用户查询的关键字进行预测,判断用户的输入关键字最可能属于哪一类的文本,基于此优先从 Solr 库中取出类别相似的数据,使得主题相关的文本靠前显示。实验表明,该排序算法具有较快的查询响应时间和较高的查准率。

## 1 算法设计

### 1.1 系统结构

图 1 是加入改进排序的用户查询流程图。具体流程如下:首先运用网络爬虫,从 Web 库中爬取需要的主题文本,作为实验数据;接着对于爬取到的文本计算其内容间的相似度、文本的 TF-IDF 值、文本和类别之间的相似度,运用上述值来改进 PageRank 值,从而得到文本数据的排序值;最后结合文本数据的排序值将其索引到数据库中。当用户通过浏览器发出 HTTP 请求时,系统从 HTTP 请求中提取关键字,利用基于文本训练得到的模型去匹配关键字,得到用户查询关键字最可能属于哪一类别,基于类别和 PageRank 值优先从数据库中取出相关数据,将其返回给用户。文中主要运用改进的 PageRank 算法和文本预分类,来提高用户查询的查准率和查询响应时间。接下来介绍改进的排序算法和文本的预分类所用到的模型。

### 1.2 排序算法

#### 1.2.1 相似度的计算

对于网页进行分类需要有如下假设:

(1)假设每个网页都是可以分类的,每个网页有一个主类别,即网页最趋于哪个类别;

(2)每个类别之间具有不同程度的联系,这个联系用相似度来表示;

(3)Web 库中总共包括  $K$  类,Web 中总共有  $N$  个页面。文本相似度的具体计算如下:对于从网页  $u$ ,算法先利用分类器,将  $u$  的内容作为分类器的输入,利用分词工具将输入的文本信息进行分词处理,用分词后

得到的词或者字分别去匹配 Web 库中所有类别中的每个类别  $c$  特征词集合,统计特征词集合中的单词在文本中是否出现,出现为 1,不出现为 0。基于这些频度形成文本的一个特征向量,计算文本特征向量和类别的特征词集合形成的特征向量(亦即是单位向量)的余弦相似性,公式如下:

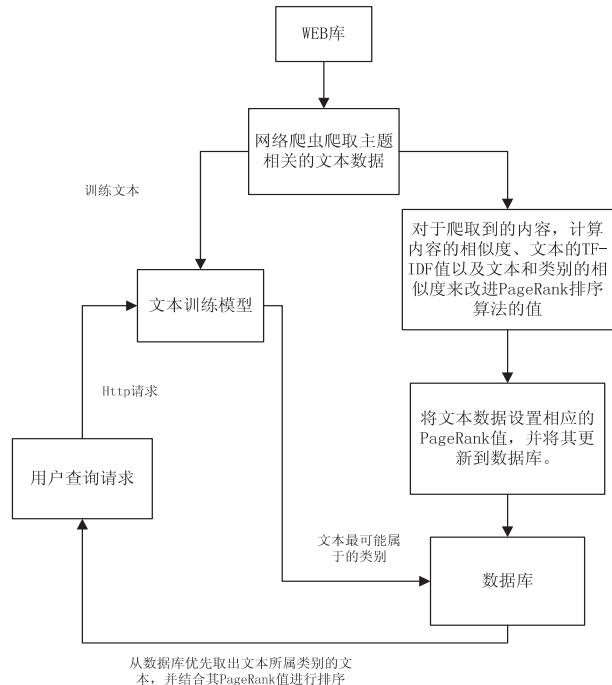


图 1 加入改进排序的用户查询流程图

$$P(u \rightarrow c) = \cos \theta = \frac{U \cdot C}{\|U\| \cdot \|C\|} = \frac{\sum_{i=1}^m U_i \times C_i}{\sqrt{\sum_{i=1}^m (U_i)^2} + \sqrt{\sum_{i=1}^m (C_i)^2}} \quad (1)$$

其中,  $u$  代表 Web 库中输入的网页;  $c$  代表 Web 库中的某一个类别;  $C$  代表语料库中特征词集合中的特征词形成的向量;  $U$  代表网页  $u$  是否包含特征词集合中的单词形成的向量;  $m$  代表特征向量的维度;  $U_i$ 、 $C_i$  代表特征向量第  $i$  维上的值。Web 库中某一网页和 Web 库中所有的类别都按上述公式形成这一网页和类别的相似度的向量。

利用上述公式,将网页  $u$  和  $v$  的内容作为输入。假设分类结果中  $u$  最趋于  $q_1$  ( $u$  的主类别),  $v$  最趋于  $q_2$  ( $v$  的主类别),  $P(u \rightarrow q_1)$ 、 $P(v \rightarrow q_2)$  分别代表  $u$  趋于  $q_1$  和  $v$  趋于  $q_2$  的概率,则  $u$  和  $v$  的文本相似度<sup>[12-13]</sup>可以由公式(2)表示:

$$P(u, v) = \frac{1}{D} \left( 1 - \frac{|P(u \rightarrow q_1) - P(v \rightarrow q_1)|}{P(u \rightarrow q_1) + P(v \rightarrow q_1)} \right) \cdot \left( 1 - \frac{|P(v \rightarrow q_2) - P(u \rightarrow q_2)|}{P(u \rightarrow q_2) + P(v \rightarrow q_2)} \right) \cdot SC \quad (2)$$

其中,  $SC = \frac{\text{sizeof}(Q_1 \cap Q_2)}{\text{sizeof}(Q_1 \cup Q_2)}$  代表  $q_1$  类和  $q_2$  类的相似度<sup>[14-16]</sup>;  $Q_1$  和  $Q_2$  分别代表  $q_1$  类和  $q_2$  类的特征向量;  $D = \sum_{i=1}^N P(u, a_i)$ , 其中所有的  $a_i$  组成了 Web 库的所有页面, 运用  $D$  将网页之间内容的相似度归一化, 将网页间内容相似性运用到 PageRank 中得到公式(3):

$$PR(u) = (1 - d) + d \left( \sum_{i=1}^n P(t_i, u) \cdot PR(t_i) \right) \quad (3)$$

其中,  $t_i$  代表指向网页  $u$  的某一网页;  $n$  代表指向网页  $u$  的网页数目;  $d$  为控制系数取值在 0 到 1 之间的数, 通常为 0.85。

TF-IDF (Term Frequency - Inverse Document Frequency) 是一种用于资讯检索与文本挖掘的常用加权技术。TF-IDF 是一种统计方法, 用于评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数呈正比增加, 但同时会随着它在语料库中出现的频率呈反比下降。TF-IDF 加权的各种形式常被搜索引擎应用, 作为文件与用户查询之间相关程度的度量或评级。因此, TF-IDF 倾向于过滤掉常见的词语, 保留重要的词语。文中 TF-IDF 值的具体计算方法如下: 通过分词工具将所有网页中的文本数据进行分词处理。

(1) TF 的计算: 设网页  $u$  的文本中分词后的单词共计  $S$  个, 统计分词后, 词  $z$  在一个文档中出现的频率为  $F$ , 则词  $z$  的 TF 值为:  $TF(z) = F/S$ ;

(2) IDF 的计算: 单词  $z$  在  $M$  个文本中出现了, 则单词  $z$  的 IDF 值为:  $IDF(z) = \log \frac{N}{M+1}$ ;

(3) 则词  $z$  的 TF-IDF 值为:  $TFIDF(z) = TF(z) \cdot IDF(z)$ 。

将文本分词后得到的单词的 TF-IDF 值的和除以  $S$  得到的值作为该网页的 TF-IDF 值, 从而得到 Web 中所有网页的 TF-IDF 向量, 记为  $TI$ , 表示如下:

$$TI(u) = \frac{\sum_{z \in u} TFIDF(z)}{S} \quad (4)$$

其中,  $z$  代表文本  $u$  分词后的某个单词。

将网页和类别的相似性、网页内容之间的相似度、网页的 TF-IDF 值运用到改进的 PageRank 中得到新的公式:

$$PR(u, c) = (1 - d) \cdot (P(u \rightarrow c) + TI(u)) + d \left( \sum_{i=1}^n P(t_i, u) \cdot PR(t_i) \right) \quad (5)$$

其中,  $P(u \rightarrow c)$  代表网页  $u$  和类别  $c$  的相似度;

$TI(u)$  代表网页  $u$  在整个 Web 库中的 TF-IDF 值。

基于上述公式来计算网页的 PageRank 值, 运用这个值对于网页进行排序, 优化用户查询的准确率和查询时间。

### 1.2.2 文本模型的训练

文中通过分词工具将 Web 库中每个文本信息进行分词处理, 用分词后得到的词或者字去匹配类别语料库中的单词, 统计每个类别中特征词集合中的特征词在哪些文本中出现过, 出现为 1, 不出现为 0, 从而得到特征词集合中的特征词在某一类别文本中总的出现次数, 进而得到特征词在所有类别的文本中出现的次数。定义特征词在某类文本中出现的比例为 TP (即特征词在单个文本出现的次数/特征词在所有文本中出现的次数所得值), 令 Web 库中每个类别的平均样本数目  $savg = N/K$ , 则特征词平均出现的次数  $avg = savg \cdot TP$ , 将  $avg$  归一化到  $[-1, 1]$  之间, 将归一化的  $avg$  值作为特征词对于某类文本的权重, 进而得到特征词对于所有类别文本由归一化的  $avg$  值构成的权重向量。将  $avg$  值归一化到  $[-1, 1]$  区间的具体方法如图 2 所示。

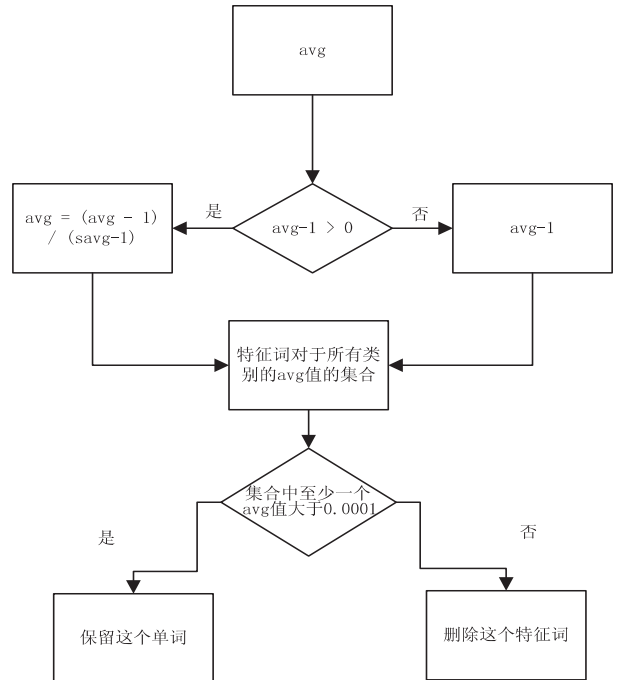


图2 归一化  $avg$  到区间  $[-1, 1]$

对于上述描述举例说明, 假设 Web 中总共包含了 60 个文本, 3 个类别, 显然  $savg$  为 20。假设类别 1 中某个特征词 (比如游戏) 在类别 1 的文本中出现 10 次, 在类别 2 的文本中出现 1 次, 在类别 3 的文本中出现 3 次, 则对于类别 1 的 TP 值为  $10/(10+1+3) = 0.71$ , 类别 2 的 TP 值为  $1/(10+1+3) = 0.07$ , 类别 3 的 TP 值为  $3/(10+1+3) = 0.21$ 。特征词对于类别 1 的  $avg$  值为  $20 \times 10/14 = 14.29$ , 同理可以得到特征词对于类别 2、3

的 avg 值。对于类别 1 的 avg 值归一化为  $(20 \times 10 / 14 - 1) / (20 - 1) = 0.74$ , 同理对于类别 2、3 也同样可以得到其归一化的值。

上面求出了语料库单词在各个类别的文本中的权重,并根据阈值 0.000 1 剔除了一些和类别相关度很低的词,统计语料库特征词集合中的哪些特征词在某个文本中出现,将这些特征词关于所有类别的权重向量累加后得到的权重向量作为该文档在各个类别上的权重向量,将其保存到本地,作为 SVM 的训练模型。

### 1.3 算法实现

#### 1.3.1 数据集的处理

文中的数据集来源于 Google 的爬虫爬取互联网的文本信息。文本主要类别有地震、暴雨、体育、游戏、房产、娱乐的文本信息。对于爬取到的文本信息,将其内容放到 html 文件中,html 具有如下几个标签:

<head>: 标签用于定义文档的头部,它是所有头部元素的容器。<head> 中的元素可以引用脚本、指示浏览器在哪里找到样式表、提供元信息等等。

<meta>: 元素可提供有关页面的元信息 (meta-information), 比如针对搜索引擎和更新频度的描述和关键词。

<title>: html 文档的标题。

<a>: 定义超链接,用于从一张页面链接到另一张页面。其属性 href 用于指示链接的目标。

<div>: 可以把文档分割为独立的、不同的部分。它可以用作严格的组织工具,并且不使用任何格式与其关联。文中用于存放爬取到的文本内容。

#### 1.3.2 文本训练模型

文中训练文本用到的模型为 LIBSVM,将 1.2.2 节得到的文本训练模型加上特定格式,形成 LIBSVM 需要的文本训练模型。运用这个模型去匹配用户的查询请求,预测查询请求最可能属于的类别。

LIBSVM 所需要的文本训练模型如下:

lable index<sub>1</sub> :value<sub>1</sub> index<sub>2</sub> :value<sub>2</sub> ... index<sub>n</sub> :value<sub>n</sub>

其中,lable 为文本所属的类别标号;index<sub>i</sub> 为类别号(所有的 index<sub>i</sub> 组成了 Web 中包含的类别);value<sub>i</sub> 为文本在类别号上对应的权重,也即 1.2.2 节所得到的权重向量。

#### 1.3.3 索引文本数据

文中对于网页中的文本数据建立索引采用的是 Solr<sup>[17]</sup>,运用 Solr 将文本数据分词后索引到 Solr 库中,并将优化后的排序算法得到的值设置为网页的排序权重。Solr 是一个高性能,采用 Java5 开发,基于 Lucene 的全文搜索服务器。同时对其进行了扩展,提供了比 Lucene 更为丰富的查询语言,同时实现了可配置、可

扩展并对查询性能进行了优化,并且提供了一个完善的功能管理界面,是一款非常优秀的全文搜索引擎。企业可以基于 Solr 开发出自己的专用搜索引擎。搜索结果文档通过 Http 利用 XML 加到一个搜索集合中。查询该集合也是通过 Http 收到一个 XML/JSON 响应来实现。它的主要特性包括:高效、灵活的缓存功能,垂直搜索功能,Solr 高亮显示搜索结果,通过索引复制来提高可用性。Solr 提供一套强大的 Data Schema 来定义字段、类型和设置文本分析、提供基于 Web 的管理界面等。Solr 有自己的 Solr 库,对于文本信息建立的索引存储在 Solr 库中,Solr 中提供一系列的 API 让用户来访问 Solr 库,并且可以动态更新 Solr 库。综上,Solr 能够胜任本实验对于文本索引的任务。

## 2 实验

### 2.1 实验数据集和实验环境

实验数据是从网上爬取的主类别为地震、暴雨、体育、游戏、房产、娱乐的文本信息,每个类别分别有 120 篇,将其中 100 篇作为训练文本,其余 20 篇作为测试文本。

实验环境:实验 PC 机为台式机,CPU 为 Inter Q9300,主频为 2.50 GHz,Java 环境为 jdk1.6,采用 eclipse-ee 作为开发平台,Solr 作为索引建立的工具,Tomcat6.0 作为 Solr 的 Web 服务器,Chrome 作为用户访问 Solr 的工具。

### 2.2 实验评价准则

#### 2.2.1 查询响应时间

为了计算加入排序和分类的查询效率,引入了查询响应时间:发出查询请求到 Solr 返回文本信息所需的时间,选取 5 个关键字(地震、游戏、篮球、软件下载、暴雨)进行查询,分别进行 10 次,计算查询的平均时间,具体如图 3 所示。

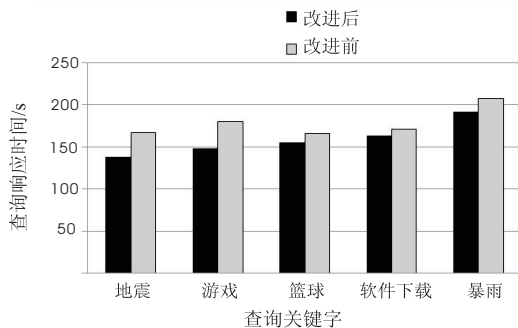


图 3 关键字查询的响应时间

从图 3 可以明显看出,加入了预分类后的查询时间一般是要优于未加入改进的排序的方案,这是由于预分类后数据库取出同类别的数据而不是整个数据库中匹配到的所有数据,故查询的响应时间要优于在整



个数据库中进行搜索。

2.2.2 查准率

查准率是评价检索模型好坏的量化指标,文中通过对关键字搜索到主题相关的数目与搜索到的总的数目的比例作为查准率(前30条)。图4列举了6个关键字搜索结果及其查准率。

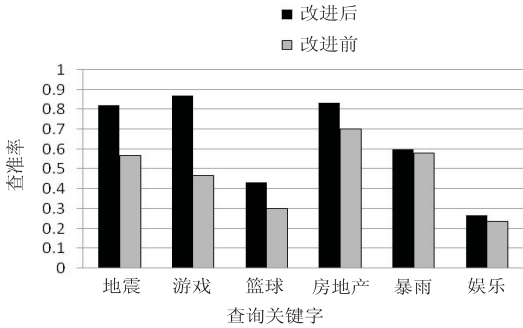


图4 关键字查询的查准率

从图中可以看出,加入了改进的排序算法,主题相关的靠前显示了,地震查询到的主题不相关的是关于房地产的,游戏中查询结果包含了娱乐的信息,篮球查准率低的原因是体育中篮球的篇幅有限,还有其他运动,房地产匹配到了地震的信息,暴雨中夹杂了体育中的暴雨天气等,娱乐的相关信息包含了游戏、篮球、桌球等的信息。结合图4,可以看出加入了主题相关性,查准率有了较大提升。

3 结束语

针对传统 PageRank 的不足,提出将分类运用到排序中去,并基于预测来优化查询的效率。实验表明,加入了分类的排序算法,能一定程度上提升查询的响应时间及其查询的准确率,但是由于实验数据较小,没有对大数据集进行测试。下一步计划将这个排序算法运用到大数据集中,在查准率中有交叉出现的现象,这也是一个需要解决的问题。

参考文献:

[1] Brin S, Page L. The anatomy of a large-scale hypertextual Web search engine[J]. Computer Networks and ISDN Systems, 1998, 30(1): 107-117.

[2] Page L, Brin S, Motwani R, et al. The PageRank citation rank-

ing: bringing order to the web[R]. Stanford; Stanford Digital Library Technologies Project, 1998.

[3] Belkin N J, Croft W B. Information filtering and information retrieval: two sides of the same coin? [J]. Communications of the ACM, 1992, 35(12): 29-38.

[4] 王福海. 基于 PageRank 的主题过滤算法改进[J]. 科技信息, 2011(15): 77-77.

[5] 陈建峡, 黄日, 马忠宝. 基于 PageRank 的 Lucene 排序算法优化与实现[J]. 计算机工程与科学, 2012, 34(10): 123-127.

[6] 舒忠梅, 左亚尧, 张祖传. 时态信息的语义抽取与排序方法研究及系统实现[J]. 计算机工程与科学, 2014, 36(8): 1609-1614.

[7] 原福永, 张园园. 基于链接分析的相关排序方法的研究和改进[J]. 计算机工程与设计, 2007, 28(7): 1630-1631.

[8] 林泓, 刘朋, 李晶晶, 等. 基于概率的 PageRank 改进算法[J]. 武汉理工大学学报, 2009, 31(3): 81-83.

[9] Gold C, Sollich P. Model selection for support vector machine classification[J]. Neurocomputing, 2003, 55(1-2): 221-249.

[10] 吴德, 刘三阳. 支持向量域多分类器[J]. 西安交通大学学报, 2012, 46(6): 87-91.

[11] 施聪莺, 徐朝军, 杨晓江. TFIDF 算法研究综述[J]. 计算机应用, 2009, 29(B06): 167-170.

[12] Haveliwala T H. Topic-sensitive pagerank: a context-sensitive ranking algorithm for Web search[J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(4): 784-796.

[13] 詹志建, 杨小平. 基于语言网络和语义信息的文本相似度计算[J]. 计算机工程与应用, 2014, 50(5): 33-38.

[14] How B C, Narayanan K. An empirical study of feature selection for text categorization based on term weightage[C]//Proceedings of the 2004 IEEE/WIC/ACM international conference on web intelligence. [s.l.]: IEEE Computer Society, 2004: 599-602.

[15] Ko Y, Park J, Seo J. Improving text categorization using the importance of sentences[J]. Information Processing & Management, 2004, 40(1): 65-79.

[16] 刘英伟, 秦永彬. 基于余弦相似性的 m-类分类器设计与算法实现[J]. 计算机与数字工程, 2014, 42(3): 351-354.

[17] Apache. Solr[EB/OL]. [2014-06-20]. <http://lucene.apache.org/Solr/>.

# 基于改进排序算法的用户查询优化的研究

作者：[吴家皋](#)，[刘杰](#)，[钱科宇](#)，[李云](#)，[WU Jia-gao](#)，[LIU Jie](#)，[QIAN Ke-yu](#)，[LI Yun](#)

作者单位：[南京邮电大学 计算机学院](#)，[江苏 南京210003](#)；[江苏省无线传感网高技术研究重点实验室](#)，[江苏 南京 210003](#)

刊名：[计算机技术与发展](#)

英文刊名：[Computer Technology and Development](#)

年，卷(期)：2015(7)

引用本文格式：[吴家皋](#).[刘杰](#).[钱科宇](#).[李云](#).[WU Jia-gao](#).[LIU Jie](#).[QIAN Ke-yu](#).[LI Yun](#) [基于改进排序算法的用户查询优化的研究](#)[期刊论文]-[计算机技术与发展](#) 2015(7)