基于SVM的主题爬虫技术研究

李 璐[1] 张国印[2] 李正文[2]

(军工保密资格审查认证中心实验室 北京 100089)[1]

(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)[2]

摘 要 随着互联网的快速发展,网络信息呈现海量和多元化的趋势。如何为互联网用户快速、准确地提取其所需信息,已成为搜索引擎面临的首要问题。传统的通用搜索引擎虽然能够在较大的信息范围内获取目标,但在某些特定领域无法给用户提供专业而深入的信息。提出基于SVM分类的主题爬虫技术,其将基于文字内容和部分链接信息的主题相关度预测算法、SVM分类算法和HITS算法相结合,解决了特定信息检索的难题。实验结果表明,使用基于SVM分类算法的爬取策略,能够较好地区分主题相关网页和不相关网页,提高了主题相关网页的收获率和召回率,进而提高了搜索引擎的检索效率。

关键词 SVM,主题爬虫,爬取策略,HITS

中图法分类号 TP302.1 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.025

Research on Focused Crawling Technology Based on SVM

LI Lu[1] ZHANG Guo-yin[2] LI Zheng-wen[2]

(Defence Industry Secrecy Examination and Certification Center Laboratory, Beijing 100089, China)[1]

(College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)[2]

Abstract With the rapid development of Internet, network information comes to be massive and diversity. How to provide the information required by users rapidly and exactly is the first task of the search engine. Traditional general search engine can provide the information in the general area, but in the special area, it cannot provide the professional and in-depth information for users. In this paper, the focused crawler based on the SVM classification algorithm was proposed for a solution to the problem of information retrieval in the special area, which makes use of the topic relevance predict algorithm based on the content and partial link information, the SVM classification algorithm and the HITS algorithm. The experiment shows that the crawling strategy based on the SVM classification algorithm can distinguish the topic related pages and topic unrelated Web pages better, improve the harvest rate and recall rate, and furthermore, the retrieval efficiency of search engines is improved.

Keywords SVM, Focused crawler, Crawling strategy, HITS

1 引言

随着互联网的迅速发展和日益普及,网络信息平台所能提供的内容越发丰富多彩,用户在搜索所需信息时面临搜索难度增加及信息筛选所需消耗的大量时间和精力[1]也随之而来。搜索引擎的出现解决了海量信息检索的难题[2]。搜索引擎通过爬虫(采集器)[3]进行资源的搜集。网络爬虫通过网络连接进行网页文档的爬取和收集,即从预先给定的URLs(Uniform Resource Location)入手,利用HTTP协议爬取所需的HTML文档,并分析这些HTML文档中所包含的超链接[4],再次抓取未访问的链接及其包含的资源。如此进行,直至没有新的URL[5]。因此,目前常见的综合大型搜索引擎普遍具有以下优点:

1)自动化高;

2)维护方便;

3)获取范围广[6]。

但也正是由于大部分搜索引擎更倾向于检索领域、范围过于广泛的信息,因此在某些特定领域的查询结果方面反而不够深入和专业化,体现在整个采集过程中主题性不够突出,爬取获得的页面分类过于混乱等方面[7]。

针对目前搜索引擎的不足,本文提出了一种基于SVM[8]的主题爬虫技术,可实现面向主题的信息搜索。采用基于主题搜索算法设计爬虫,对主题相关的网页文档进行专项深入搜集。通过爬取算法预测、分析所访问的网页文档与主题相关度,并决定是否爬取这些网页文档中所包含的超链接。基于SVM的主题爬虫可以有效减少无关页面文档的爬取,节约带宽的占用率,加强了检索的规范程度,从而能提高搜索引擎的检索效率。

到稿日期:2014-03-03 返修日期:2014-05-25

李 璐(1985-),女,博士,工程师,主要研究方向为网络信息安全、信息检索,E-mail:lilu_1231@163.com;张国印(1962-),男,博士,教授,CCF会员,主要研究方向为网络信息安全、信息检索等;李正文(1985-),男,硕士,主要研究方向为信息检索。

2 主题相关度预测算法的研究

主题相关度预测模块针对即将爬取网页的主题相关程度进行预测，并按照所得相关程度进行 URL 队列的排序，主题相关度越高，网页的优先级越高，由此可减少对主题相关度低的网页进行的爬取工作。

主题相关度预测算法中涉及到父网页、锚文本、链接附近的信息与主题的相关度以及父网页中的兄弟 URL 与主题的相关度。

1)子网页从父网页集合中获得主题相关度 *score_parent*。

父网页 u 与主题相关时，它所指向的网页的主题相关率相对较高[9]。根据此条件可以对网页 u 中子网页的主题相关度进行预测，即父网页 u 的相关度得分将通过其前向链接直接传递给子网页 v，或通过其子网页间接将相关度得分传递给其子孙网页。但在主题相关度的传递过程中，祖先网页的主题相关度需不断衰减，因此通过网页的前向链接，网页主题相关度达到向下传递。子网页 v 的相关度得分部分继承于父网页，因此子网页从父网页集合中所继承的主题相关度计算如式(1)、式(2)所示：

$$score\_parent(u,v)=\sum_{u\in B(v)}\frac{w(u)}{|F(u)|} \tag{1}$$

$$score\_parent(v)=\frac{\sum_{u\in B(v)} score\_parent(u,v)}{|B(v)|} \tag{2}$$

其中，$B(u)$表示指向网页 u 的网页集合；$F(u)$表示网页 u 指向的网页集合；$w(u)$表示网页 u 的主题相关度；*score_parent*(u,v)表示网页 v 从网页 u 中继承的主题相关度得分；*score_parent*(v)表示子网页从父网页集合中继承的主题相关度得分。

由式(1)可知，父网页的主题相关度均分给其子网页；子网页从其所有父网页中继承主题相关度。

2)在父网页集合中，指向网页 v 的链接 URL 的锚文本与主题的相关度 *score_anchor*，链接 URL 邻接区域的文字信息与主题的相关度 *score_context*。

由于 *score_anchor* 和 *score_context* 内容信息与所指向的子网页的主题相关，因此有助于子网页 v 与主题的相关度的预测。指向子网页 v 的父网页可能同时存在多个，因此 *score_anchor* 和 *score_context* 的计算需要综合考虑父网页中的链接锚文本和链接邻近区域的上下文信息。*score_anchor*、*score_context* 的计算如式(3)－式(6)所示。

$$score\_anchor(u,v)=\frac{vec_{anchor}(u,v)\cdot vec_{Topical}}{|vec_{anchor}(u,v)||vec_{Topical}|}=\frac{\sum_{k=1}^{n}w_{anchor}(u,k)\times w(t,k)}{\sqrt{\sum_{k=1}^{n}w_{anchor}(u,k)^2\sum_{k=1}^{n}w(t,k)^2}} \tag{3}$$

$$score\_context(u,v)=\frac{vec_{context}(u,v)\cdot vec_{Topical}}{|vec_{context}(u,v)||vec_{Topical}|}=\frac{\sum_{k=1}^{n}w_{context}(u,k)\times w(t,k)}{\sqrt{\sum_{k=1}^{n}w_{context}(u,k)^2\sum_{k=1}^{n}w(t,k)^2}} \tag{4}$$

$$score\_anchor(v)=\frac{\sum_{u\in B(v)}w(u,v)\cdot score\_anchor(u,v)}{|B(v)|} \tag{5}$$

$$score\_context(v)=\frac{\sum_{u\in B(v)}w(u,v)\cdot score\_context(u,v)}{|B(v)|} \tag{6}$$

其中，$vec_{anchor}(u,v)$表示父网页 u 中指向子网页 v 的接锚文本向量；$vec_{Topical}$表示主题向量；*score_anchor*(u,v)表示子网页 v 从父网页 u 指向子网页 v 的链接锚文本中获得的主题相关度得分；*score_context*(u,v)表示子网页 v 从父网页 u 指向子网页 v 的链接上下文信息中获得的主题相关度得分；*score_anchor*(v)表示网页 v 从其所有父网页锚文本中获得的主题相关度得分；*score_context*(v)表示网页 v 从其所有父网页的链接上下文信息中获得的主题相关度得分；$B(v)$表示指向网页 v 的网页集；$w_{anchor}(u,k)$表示指向网页 u 的链接锚文本向量的第 k 个分量；$w(t,k)$表示主题向量 t 的第 k 个分量；$w_{context}(u,k)$表示指向网页 u 的链接上下文信息向量的第 k 个分量；$w(u,v)$表示父网页 u 的权重值。

3)网页 v 从父网页 u 中兄弟链接获得主题的相关度 *score_sibling*。

网页 v 的主题相关度受其兄弟网页的主题相关度影响，即与指向网页 v 的同网页的其他链接 URL(兄弟网页)中，如果兄弟网页与主题相关，则网页 v 与主题相关度就高。因此，受兄弟网页影响的网页 v 主题相关度公式如式(7)、式(8)所示：

$$score\_sibling(v)=\delta\cdot\frac{\sum_{t\in U_s}w_r(t)}{|U_s|} \tag{7}$$

$$w_r(t)=\begin{cases}w(t), & \text{若网页 } t \text{ 已爬取}\\ w_{predict}(t), & \text{否则}\end{cases} \tag{8}$$

其中，$w_{predict}(t)$表示预测未爬取网页 t 的主题相关度得分；$w(t)$表示已爬取网页 t 的主题相关度；$w_r(t)$表示网页 t 的主题相关度；U_s 表示网页 v 的父网页 u 中 URL 集合；$|U_s|$表示 U_s 集合的数量；δ 表示衰减因子。

4)未爬取网页 v 的预测主题相关度通过对 *score_parent*、*score_anchor*、*score_context* 和 *score_sibling* 采取加权求和计算，即综合考虑父网页、锚文本、链接附近的信息及兄弟链接与主题的相关度，公式如下：

$$score\_predict(v)=w_1\cdot score\_parent(v)+w_2\cdot score\_anchor+w_3\cdot score\_context(v)+w_4\cdot score\_sibling(v)$$

$$\sum_{i=1}^{4}w_i=1$$

其中，w_i 为相关度分量的权重值；*score_predict*(v)表示预测未爬取网页 v 的主题相关度。

3 爬取策略研究

网络数据实时更新的动态特性，导致了网页内容与 Web 链接关系存在即时变化。因此，多次递归爬取过程是爬虫设计的首选策略。通过 SVM 分类器对已爬取的历史网页库信息进行筛选处理获取主题相关网页，根据 HITS 算法对相关网页库中的 Web 链接信息计算出 Web 对应的权威度(authority)和中心度(hub)，根据所获得的信息，预测待爬取网页与主题的相关度，从而进行再次爬取。

3.1 首次爬取算法

1)种子的选取。在主题爬虫首次爬取时，采取人工选取种子的方式：将关键词输入通用搜索引擎中，并在搜索结果中进行人工筛选，与主题相关的网页作为种子集合，为首次爬取提供起始种子。

2)首次爬取算法的设计。首次爬取算法中,仅包含一个多线程共享的优先级队列。由于在首次爬取中可参考信息有限,因此采用基于文字内容信息和部分链接信息的主题相关度预测算法,在爬取网页中抽取新的 URL 进行主题相关度的预测评价,并将所抽取的 URL 按预测的主题相关度得分添加到优先级队列中。

通过动态地维护已爬取网页集合所构成的 Web 子图信息,为预测待爬取网页的主题相关度提供参考。因此,在该 Web 子图信息中记录着已爬取网页之间的链接结构和主题相关度等信息。

首次爬取算法如算法 1 所示。

算法 1:

```
Crawl_first(G,Seeds)
    Priority_Queue〈priority_queue data structure〉 Queue //优先级队列
    Collection 〈url-page pairs〉 collection //url,page 对
    Hash_tablevisited_table// 已访问 url 列表
      Queue←seeds urls //种子 urls 插入优先级队列
    while Queue is not empty and crawl_num<MAX_CRAWLING_PAGE
    do
    crawl_url←seed url in Queue but not in the visited_table
    visited_table←crawl_url
    PAGE←fetch_page (crawl_url)
    Collection←store (crawl_url,PAGE )
    Queue←urls in PAGE but not in collection and Queque //按优先级插入
    crawl_num++
    end
    Return collection
```

算法 1 中,priority_queue 为最大优先级队列。在 priority_queue优先级队列所维护的集合中,每个元素均对应一个优先级 *key*。priority_queue 优先级队列支持以下方法:

Queue_Insert(*Set*,*e*,*key*):将优先级为 *key* 的元素 *e* 插入到集合 *Set* 中;

Queue_Maximum(*Set*):返回集合 *Set* 中优先级最高的元素;

Queue_Extract_Max(*Set*):返回集合 *Set* 中优先级最高的元素,并将其从集合 *Set* 中删除;

Queue_Increase_key(*Set*,*e*,*key*):将集合 *Set* 中元素 *e* 的优先级置为 *key*。

priority_queue 优先级队列通过最大堆实现,其基本操作的时间复杂度为 O(lg(*n*))(*n* 为元素个数),因此具有较高的效率。

3.2 再次爬取算法

SVM 分类器对爬取的网页集合进行过滤,提取与主题相关的网页,并加入训练集中,可不断提高 SVM 分类器的分类准确度。同时,通过这些主题相关网页的集合,构造主题子图 *G*(focused subgraph)。然后在主题子图 *G* 内,应用链接分析算法,寻找与主题相关的权威网页和中心网页集合。

在使用 HITS 算法时,应考虑链接重要度差别性的存在。按链接所在的域分,可分为横向链接(Transverse Link)和内部链接(Intrinsic Link)。横向链接是不同域名间网页的链接,内部链接是相同域名内的链接。横向连接要比内部链接重要,因此可以只考虑横向链接,忽略内部链接或者给内部链接赋予较低的权重。

令 *A* 表示主题子图 *G* 的邻接矩阵,则其在忽略链接重要度时,如式(9)所示:

$$A[u][v]\begin{cases}1, & u\rightarrow v\\ 0, & u\not\rightarrow v\end{cases} \tag{9}$$

其中,$u\rightarrow v$ 表示存在 u 到 v 的边;$u\not\rightarrow v$ 表示 u 到 v 的边不存在。在考虑横向连接和内部链接重要度权重时,邻接矩阵 *A* 如式(10)所示:

$$A[u][v]=\begin{cases}1, & u\rightarrow v \text{ transverse link}\\ c, & u\rightarrow v \text{ intrinsic link}\\ 0, & u\not\rightarrow v\end{cases},0<c<1 \tag{10}$$

种子提取算法如算法 2 所示。

算法 2:

```
get_seeds (seeds_Authority,seeds_Hub,classified_collection)
(1)初始化 Authority(u),Hub(u)为任意非 0 值,其中,u∈ classified_
   collection,1<u<n
(2)While distance>ε
   do
     for each u ∈V
     do
     Authority_i(u)=∑_{v∈B(u)} Hub_{i-1}(v) · δ(v,u)
     Hub_i(u)=∑_{v∈F(u)} Authority_{i-1}(v) · δ(u,v)
   end
▷compute the length(La,Lh) of authority and hub
▷normalize the vector(Authority_i(u),Hub_i(u)) of authority and hub
distance_Authority=√(∑_{k=1}^{n}(Authority_i(k)-Authority_{i-1}(k))^2)
distance_Hub=√(∑_{k=1}^{n}(Hub_i(k)-Hub_{i-1}(k))^2)
distance=distance_AuthorityScore+distance_HubScore
end
(3)seeds_Authority=
      select max n1 authority pages from classified_collection
      seeds_Hub=select max n2 hub pages from classified_collection
```

其中,

$$\delta(u,v)=\begin{cases}1, & u\rightarrow v \text{ transverse link}\\ c, & u\rightarrow v \text{ intrinsic link}\end{cases},0<c<1。$$

seeds_Authority:表示在与主题相关网页集合中,权威度较高的 n_1 个网页形成的种子。权威度高的种子集合有利于更新权威度高的网页集合。

seeds_Hub:表示与主题相关网页集合中,中心度较高的 n_2 个网页形成的种子。中心度高的种子集合有利于发现新的主题资源。

再次爬取算法中,包含线程共享的优先级队列。其中一个是优先级队列 Queue[1],队中元素为 seeds_Authority 种子集合。另一个是优先级队列 Queue[2],队中元素为 seeds_Hub 种子集合。分别以 Queue[1]、Queue[2]为优先级队列,启动双线程。在优先级队列 Queue[*i*]中,选取最高优先级的 URL 爬取,然后解析新爬取的网页,提取链接,插入优先级队列 Queue[*i*]中。采用基于文字内容和部分链接的主题相关度预测算法,预测新提取的链接与主题相关度。

再次爬取算法如算法 3 所示。

算法 3:

```
Crawl_first(G,Seeds_Authority,Seeds_Hub)
        Priority_Queue〈priority_queue data structure〉 Queue1,
```

```
Queue2 //优先级队列
    Collection〈url-page pairs〉collection1, collection2 //url, page 对
Hash_tablevisited_table //已访问 url 列表
Queue1←urls in Seeds_ Authority with url's priority // Seeds_ Authority 中 url 插入队列
Queue2←urls in Seeds_ Hub with url's priority // Seeds_ Hub 中 url 插入队列
Thread[i]: //i is 1 or 2
while Queue[i] is not empty and crawl_num[i]<MAX_CRAWLING_PAGE
  do
  crawl_url←url in Queue[i] but not in the visited_table
  visited_table←crawl_url
  PAGE←fetch_page (crawl_url)
  Collection[i]←store (crawl_url,PAGE)
  Queue←urls in PAGE but not in collection1 and Queque[i] //按优先级插入
  crawl_num[i]++
end
```

4 实验结果与分析

4.1 环境配置

本文基于 TSE 和 LibSVM 进行实验。其中，TSE 是支持相关网页爬取策略[10]，LibSVM 可针对已爬取网页集合进行主题分类[11]。实验所需的软/硬件环境如表 1 所列。

表 1 实验平台软/硬件配置

实验环境	版本/配置
CPU	Intel 双核 1.6G
内存	1G
硬盘	80G
Linux	Redhat 4
LibSVM	1.0
TSE	1.0
Tomcat	6.0

4.2 数据分析

本文通过宽度优先算法、HITS 算法、基于 SVM 分类的主题爬虫算法的爬取策略进行收获率和召回率的比较。

收获率(Harvest rate)为衡量其性能的最重要指标，收获率越高，与主题相关的网页所占的比例越大。其计算公式如下：

$$Harvest\ rate=\frac{\sum_{related\ to\ topic}\text{Web pages}}{\sum_{obtained}\text{Web pages}}$$

SVM-topical、HITS、Breadth-First 算法的主题相关网页收获率如图 1 所示。

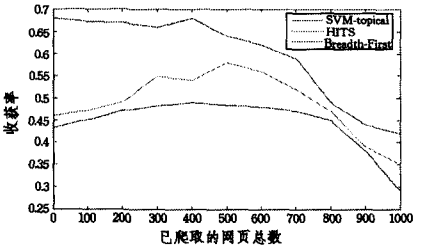

图 1 SVM-topical、HITS、Breadth-First 算法的主题相关网页收获率

图 1 的横轴记录爬取的网页数量，纵轴记录对应主题相关网页的收获率。从实验数据可知，SVM-topical 算法的主题相关网页收获率最高，其次为 HITS 算法，而 Breadth-First 算法最低。其中，Breadth-First 算法基于宽度优先的思想，虽然与当前爬取网页深度接近的网页即重要度较高的网页会被优先获取，但没有考虑所爬取网页与主题相关的因素，因此，主题相关网页的收获率最低。HITS 算法是基于链接分析的网页爬取策略，通过权威度或中心度高的网页挖掘新的资源，但没有考虑文字内容的影响因素，易导致发生主题漂移的问题。因此，主题相关网页的收获率会随着爬取网页数量的增加而下降。本文所采用的 SVM-topical 算法，针对已爬取的网页集合使用 SVM 分类器进行筛选，去除与主题无关的网页，然后采用 HITS 算法寻找权威度或中心度高的网页，将其作为种子，用于下次爬取。同时，加入文字内容的影响因素，即基于文字内容信息和部分链接信息的主题相关度预测算法，去除与主题不相关的网页。因此相对前两种策略，主题相关网页的收获率有所提高。

召回率(Target Recall)是衡量爬虫性能的另一个重要标准。召回率计算公式如下：

$$R(t)=\frac{|C(t)\cap T|}{|T|}$$

其中，$R(t)$表示主题爬虫在爬取 t 网页后的召回率；$C(t)$表示主题爬虫已爬取的 t 网页组成的集合；T 为从网上随机搜集的主题相关网页组成的集合。

SVM-topical、HITS、Breadth-First 算法目标网页的召回率如图 2 所示。

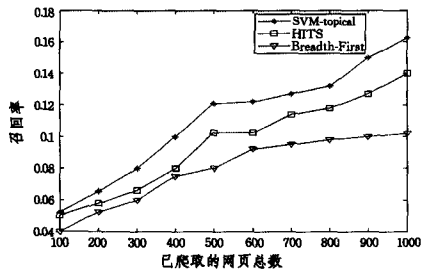

图 2 SVM-topical、HITS、Breadth-First 算法目标网页的召回率

图 2 的横轴记录爬取的网页数量，纵轴记录对应主题相关网页的召回率。由实验数据可知，SVM-topical 算法的主题相关目标网页的召回率最高，其次为 HITS 算法，而 Breadth-First 算法为最低。在网页爬行数量较少时，3 种爬取策略的主题相关目标网页的召回率相差不大。由于 SVM-topical 算法更有利于扩大爬取的范围、挖掘新的与主题相关的资源，避免了主题漂移[12]等问题，因此，随着所爬取网页数量的增加，本文的 SVM-topical 算法的召回率逐渐高于其他两种算法。

SVM-topical、HITS、Breadth-First 算法爬取主题相关网页数与时间关系如图 3 所示。

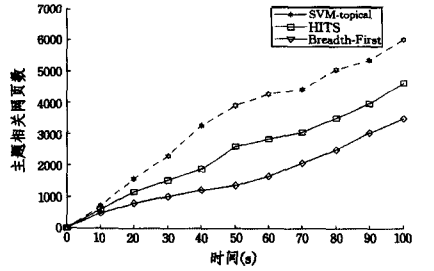

图 3 SVM-topical、HITS、Breadth-First 算法爬取主题相关网页数与时间关系

图 3 的横轴记录爬取时间，纵轴记录爬取主题相关网页的数量。由实验数据可知，在相同时间内，SVM-topical 算法抓取的主题相关目标网页数最高，其次为 HITS 算法，而 Breadth-First 算法最低。在时间较短时，3 种爬取策略爬取的主题相关目标网页数量相差不大。由于 SVM-topical 算法有利于主题相关网页的挖掘，因此，随着所爬取时间的增加，本文的 SVM-topical 算法爬取的网页数量逐渐高于其他两种算法。

结束语 本文提出了一种基于 SVM 分类算法的主题爬虫技术，通过两次爬取获取与主题相关的网页。其中，在第一次爬取时，采用种子选取的方式，将与主题相关的关键词加入到爬虫中，之后在获取的页面列表中提取与主题相关的一类集合作为种子。第二次爬取时，采用 HITS 算法对已爬取的网页集合进行计算处理获取种子。通过这种基于预测的主题相关性算法，在考虑网页文本的同时，也借助网页之间的链接关系，选取权威度或中心度高的页面作为种子。其中，权威度高的网页与主题相关性高，而中心度高的网页有利于挖掘出主题相关网页。经实验表明，基于 SVM 分类算法的主题爬虫技术可提高主题相关网页收获率和召回率，进而提高搜索引擎的检索效率。

参 考 文 献

[1] Boanjak M, Oliveira E, et al. TwitterEcho: a distributed focused crawler to support open research with twitter data[C]// WWW'12 Companion Proceedings of the 21st International Conference Companion on World Wide Web. 2012

[2] Kazai G. In Search of Quality in Crowdsourcing for Search Engine Evaluation[J]. Advances in information retrieval, Lecture Notes in Computer Science, 2011, 66(11): 165-176

[3] 许笑，张伟哲，张宏莉，等. 广域网分布式 Web 爬虫[J]. 软件学报，2010，21(5)：1067-1082

[4] 张宪超，徐雯，高亮，等. 一种结合文本和链接分析的局部 Web 社区识别技术[J]. 计算机研究与发展，2012，49(11)：2352-2358

[5] de Groc C. Babouk: Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction[J]. Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE/WIC/ACM International Conference, 2011, 3(1), 497-498

[6] 张伟哲，张宏莉，许笑，等. 分布式搜索引擎系统效能建模与评价[J]. 软件学报，2012，23(2)：253-265

[7] 王上，于海，王钲旋. Deep Web 垂直搜索引擎设计与实现[J]. 计算机研究与发展，2009，46：359-365

[8] 蒋华荣，郁雪. 应用遗传算法优化子空间的 SVM 分类算法[J]. 计算机科学，2013，40(11)：255-260，275

[9] 黄仁，王良伟. 基于主题相关概念和网页分块的主题爬虫研究[J]. 计算机应用研究，2013，30(8)：2377-2380

[10] 李晓明，闫鸿飞，王继民. 搜索引擎：原理技术与系统[M]. 北京：科学技术出版社，2004：29-33

[11] Chang Chih-chung, Lin Chih-jen. LIBSVM: A library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 280-292

[12] 李稚楹，杨武，谢治军. PageRank 算法研究综述[J]. 计算机科学，2011，38(Z10)：185-188

(上接第 89 页)

[11] Hsieh J-W, Kuo T-W, Chang L-P. Efficient identification of hot data for flash memory storage systems[J]. ACM Transactions on Storage, 2006, 2(1): 22-40

[12] Huang P-C, Chang Y-H, Kuo T-W, et al. The behavior analysis of flash-memory storage systems[C]// Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing. 2008: 529-534

[13] Jiang S, Zhang L, et al. S-ftl: An efficient address translation for flash memory by exploiting spatial locality[C]// IEEE 27th Symposium on Mass Storage Systems and Technologies. 2011

[14] Kim J, Kim J M, Noh S, et al. A space-efficient flash translation layer for compactflash systems[J]. IEEE Transactions on Consumer Electronics, 2002, 48(2): 366-375

[15] Kim J M, Min S L, Choi J, et al. A low-overhead highperformance unified buffer management scheme that exploits sequential and looping references[C]// Proceedings of the 4th Symposium on Operating Systems Design and Implementation. 2000: 119-134

[16] Lee S, Shin D, Kim Y-J, et al. Last: locality-aware sector translation for nand flash memory-based storage systems[J]. SIGOPS Oper. Syst. Rev., 2008, 42: 36-42

[17] Lee S-W, Park D-J, Chung T-S, et al. A log buffer - based flash translation layer using fully-associative sector translation[J]. ACM Transactions on Embedded Computing Systems, 2007, 6(3): 1-27

[18] Li Y, Xu J, Choi B, et al. Stablebuffer: optimizing write performance for dbms applications on flash devices[C]// Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10). 2010

[19] Lin L, Li X, Jiang H, et al. Amp: An affinity-based metadata prefetching scheme in large-scale distributed storage systems[C]// Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid. 2008

[20] Madhyastha T M, Reed D A. Input/output access pattern classification using hidden markov models[C]// Proceedings of the Fifth Workshop on I/O in Parallel and Distributed Systems (IOPADS'97). 1997

[21] Nath S, Gibbons P B. Online maintenance of very large random samples on flash storage[J]. The VLDB Journal, 2010, 1(1): 970-983

[22] Park D, Du D. Hot data identification for flash-based storage systems using multiple bloom filters[C]// 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies. 2011

[23] Park S, Park J, Kim S, et al. A pattern adaptive nand flash memory storage structure[J]. IEEE Transactions on Computers, 2010, 61(1): 134-138

[24] Repository U T. Oltp application i/o [OL]. http://traces.cs.umass.edu

[25] Zhu Y, Jiang H. Race: A robust adaptive caching strategy for buffer cache[J]. IEEE Transactions on Computers, 2008, 57(1): 25-40

[26] 唐世庆，李云龙，田凤明，等. 基于 Hadoop 的云计算与存储平台研究与实现[J]. 四川兵工学报，2014，35(8)：97-100