

Deep Web 爬虫爬行策略研究

郑冬冬, 崔志明

(苏州大学 智能信息处理及应用研究所, 江苏 苏州 215006)

摘要: 如今 Web 上越来越多的信息可以通过查询接口来获得, 为了获取某 Deep Web 站点的页面用户不得不键入一系列的关键词集。由于没有直接指向 Deep Web 页面的静态链接, 当前大多搜索引擎不能发现和索引这些页面。然而, 近来研究表明 Deep Web 站点提供的高质量的信息对许多用户来说是非常有价值。这里研究了怎样建立起一个有效的 Deep Web 爬虫, 它可以自动发现和下载 Deep Web 页面。由于 Deep Web 惟一“入口点”是查询接口, Deep Web 爬虫设计面对的主要挑战是怎样对查询接口自动产生有意义的查询。这里提出一种针对查询接口查询自动产生问题的理论框架。通过在实际 Deep Web 站点上的实验证明了此方法是非常有效的。

关键词: Deep Web; Deep Web 爬虫; 查询选择; 查询效能; 适应性爬行算法

中图法分类号: TP393 **文献标识码:** A **文章编号:** 1000-7024(2006)17-3154-05

On research of deep web crawler's crawling strategy

ZHENG Dong-dong, CUI Zhi-ming

(Institute of Intelligent Information Processing and Application, Soochow University, Suzhou 215006, China)

Abstract: As an ever-increasing amount of information on the web today is available through search interfaces, users have to key in a set of keywords in order to access the pages from certain web sites, which are often referred to as the hidden web or the deep web. Since there is no static links to the hidden web pages, search engines cannot discover and index such pages. However, according to recent studies, the content provided by many hidden web sites is often of very high quality and can be extremely valuable to many users. How to build an effective hidden web crawler that can autonomously discover and download pages from the hidden web is studied. Since the only "entry point" to a hidden web site is a query interface, the main challenge to a hidden web crawler is how to automatically generate meaningful queries for issue to the site. A theoretical framework to investigate the query generation problem for the hidden web and we propose effective policies for generating queries automatically is provided. Experiment shows that these policies are effective.

Key words: deep web; deep web crawler; query selection; query efficiency; adaptive algorithm

0 引言

近来研究表明网络上大部分内容是不能通过静态链接^[2]获取的, 特别是大部分隐藏在搜索表单之后的页面只有通过用户键入一系列关键词才可以获得。这些页面被称为 Hidden Web 或 Deep Web, 由于当前的搜索引擎不能索引到或不能在它们的返回结果中出现这些页面, 因此对用户来说这部分页面是隐藏的。Deep Web 最初由 Dr. Jill Ellsworth 于 1994 年提出, 指那些由普通搜索引擎难以发现其信息内容的 Web 页面。2001 年, Christ Sherman、Gary Price 对 Deep Web 定义为: 虽然通过互联网可以获取, 但普通搜索引擎由于受技术限制而不能或不作索引的那些文本页、文件或其它通常是高质量、权威的信息。据最近对 Deep Web 的调查^[1]得到了以下有意义的发现:

(1) Deep Web 大约有 307 000 个站点, 450 000 个后台数据

库和 1 258 000 个查询接口。它仍在迅速增长, 从 2000 年到 2004 年, 它增长了 3~7 倍。

(2) Deep Web 内容分布于多种不同的主题领域, 尽管电子商务是主要的驱动力量, Web 数据库的发展趋势不仅在此领域, 同时, 在非商业领域相对占更大比重。

(3) 当今的爬虫并非完全爬行不到 Deep Web 后台数据库内容, 当前主要的搜索引擎已经覆盖 Deep Web 大约 1/3 的内容。然而, 在覆盖率上当前搜索引擎存在技术上的本质缺陷。

(4) Deep Web 中的后台数据库大多是结构化的, 其中结构化的是非结构化的 3.4 倍之多。

(5) 虽然一些 Deep Web 目录服务已经开始索引 Web 数据库, 但是它们的覆盖率比较小, 仅为 0.2%~15.6%。

(6) Web 数据库往往位于站点浅层, 94% 之多的大量 Web 数据库可以在站点前 3 层发现。

收稿日期: 2005-10-13。

基金项目: 教育部高校博士学科点科研基金项目 (20040285016); 江苏省高技术研究基金项目 (BG2005019)。

作者简介: 郑冬冬 (1980—), 男, 河南焦作人, 硕士研究生, 研究方向为网络与数据库技术、Web 数据挖掘; 崔志明 (1961—), 男, 江苏苏州人, 教授, 博士生导师, 研究方向为智能化信息处理、计算机网络与数据库应用。

可以看出 Deep Web 中信息量要比 Surface Web 信息量的多,同时由于 Deep Web 页面信息是由后台数据库动态产生的,数据库多是结构化的关系数据库,因此信息的质量要比非结构化的页面要高。本文研究如何设计 Deep Web 爬虫,它可以自动下载 Deep Web 页面,使搜索引擎可以索引这些页面。传统的爬虫是根据页面中的超链接来发现页面的,所以传统的搜索引擎不能索引 Deep Web 页面。一个有效的 Deep Web 爬虫将对信息获取有很重要的影响。

考虑到 Deep Web 的惟一入口点是搜索表单,要设计 Deep Web 爬虫主要存在两方面问题:一方面爬虫必须能理解和模型化查询接口;另一方面爬虫必须能对查询接口提出有意义的查询。前一个问题已经由 Raghavan 和 Garcia-Molina 在文献 [3] 提出,同时他们提出一种学习寻找查询接口表单的方法。本文主要针对后一问题进行研究,即为了发现和下载 Deep Web 页面爬虫怎样自动产生查询。

当搜索表单提供了每一表单元素所有可能的值时,解决方案是很直接的。穷尽所有可能值的组合就可以了。然而,当表单含有自由文本输入元素时,由于文本的值域是无限集,所以就不可能穷尽所有可能值。在此情况下,爬虫可以在不理解搜索表单元素语义的情况下自动提出有意义的查询吗?本文提出了对 Deep Web 爬虫设计的一种理论框架,提出了查询自动产生问题几种策略。

1 系统基本结构

本节形式化地描述了一个 Deep Web 爬虫的理论框架。描述了在 Deep Web 站点上的一些假设并解释了用户怎样和站点进行交互。基于此交互模型,提出了一种 Deep Web 爬虫爬行算法。最后形式化地定义了 Deep Web 爬虫爬行问题。

1.1 Deep Web 数据库模型

网络上存在关于多方面主题的 Deep Web 数据源。根据信息类型,可以把 Deep Web 站点后台数据库分类为文本数据库和结构化的数据库。文本数据库主要指包含有许多纯文本文档资料的数据库,例如 news.sohu.com。由于纯文本文档通常没有很好的结构,大多数文本数据库提供一个简单的搜索接口,用户可以在简单在此接口(如图 1 所示)中键入一关键词。相反一个结构化的数据库通常由有关联的多个属性的数据对象组成,它支持多属性的搜索界面(如图 2 所示)。本文研究对单一属性关键词查询的文本数据库爬虫设计^[7]。

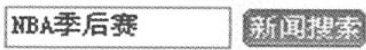


图 1 单一属性的搜索接口界面

用户一般通过如下 3 个步骤来访问 Deep Web 后台数据库:首先,用户通过站点提供的查询接口提出查询,如“NBA 季后赛”。然后,用户得到一个查询结果的索引页面列表(如图 3(a)所示)。最后,从索引列表页面选出用户感兴趣的页面,从而获得真正的数据页面(如图 3(b)所示)。

1.2 一般的 Deep Web 爬虫爬行算法

考虑到 Deep Web 站点的惟一入口点是搜索表单, Deep

题名:	<input type="text"/>	出版社:	<input type="text"/>
著者:	<input type="text"/>	ISBN/ISSN 号:	<input type="text"/>
丛书名:	<input type="text"/>	索取号:	<input type="text"/>
主题词:	<input type="text"/>	起始年代:	<input type="text"/>
语种类别:	<input type="text" value="所有语种"/>	文献类型:	<input type="text" value="所有类型"/>
<input type="button" value="开始查询"/>		<input type="button" value="重新输入"/>	

图 2 多属性的搜索接口界面



(a) 针对“NBA 季后赛”查询匹配部分页面

据新华社华盛顿 5 月 8 日电(记者梁希仪) 这哪像季后赛、哪像半决赛? NBA 季后赛半决赛东西部的第一场较量就以悬殊分差结束。

马刺队以 103: 81 大胜超音速队。这一结果难免让人想起前一天火箭队以 40 分之差输给小牛队而被淘汰那一幕, 因为这根本不是双方的真实差距。马刺队其实在第二节还剩下两分钟就已经锁定胜局。在 35: 22 领先第一节后, 马刺队在第二节打出一个 17: 2 的高潮, 将比分一下变成 58: 28。整整领先 30 分之多。之后, 马刺队尽管不敢麻痹大意, 第四节分差也被缩小到 16 分, 但最后仍然赢得轻松惬意。在东部迈阿密, 热火队虽然错过了第二节曾有过的 17 分优势, 而且全场 15 次失误, 11 次罚球不中, 两员主将奥尼尔和维德却没有多少过人之处, 但仍然以 106: 86 击败奇才队。

(b) 针对第 3 项查询匹配的页面

图 3 查询页面

Web 爬虫通过上述的 3 个步骤来获取结果页面。即爬虫首先针对站点接口产生一个查询; 然后下载结果的索引列表页面; 最后根据结果的索引下载真正的页面。通常情况下, Deep Web 爬虫在爬行时间和网络资源使用上都是有限的, 爬虫一般迭代执行上述过程, 直到耗尽其所有的资源。

图 4 指出了 Deep Web 爬虫一般的爬行算法。为简化问题, 假设 Deep Web 爬虫仅处理单一属性关键词查询问题。爬虫首先决定下次将使用那一个关键词, 然后是发送查询获得结果索引页, 最终基于这些超链接, 爬虫下载具体页面内容。反复执行这个过程, 直到其耗尽所有的资源。此算法最为关键的一步是爬虫针对查询接口选择下次将提出什么样的查询关键词用于表单提交。如果爬虫能提供成功的查询, 它将返回很多匹配页面同时它将使用最少的资源提前完成爬行任务。

算法 1 爬行 Deep Web 站点

Procedure

(1) while(there are available resources)do

(2) qi = SelectTerm() //选一关键词发送到 Deep Web 站点

(3) R(qi) = QueryWebSite(Qi) //发送查询, 获取结果索引列表页面

(4) Download(R(qi)) //下载用户自己感兴趣的页面内容

(5) done

图 4 Deep Web 爬虫爬行算法

相反,如果爬虫提出的是一个与查询任务完全无关的关键词,它将不返回任何与任务相关的页面同时浪费很多资源。因此,爬虫对表单下次提交关键词的选择问题将在很大程度上影响其效能。下一节将给出查询关键词选择问题的形式化定义。

1.3 爬行问题形式化定义

理论上,查询关键词选择问题可以被如下形式化的定义:假设爬虫从站点下载的页面集为 S (如图 5 所示) 用一个点来代表 S 中的每一个页面,对于每个潜在的查询 q_i 返回的结果页面集可以认为是 S 的一个子集。对于每个查询 q_i 都有一定的代价,如资源消耗量等。在这种定义下其目标是通过最小的代价来寻找能覆盖最多点的查询。这个问题等价于图论^[4]中集覆盖问题。

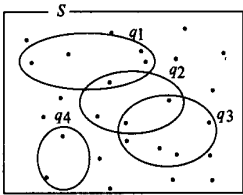


图 5 集覆盖的最优解问题

在此形式化定义中主要涉及两个问题。首先是,实际上针对每次查询 q_i 爬虫并不知道将有哪些和将有多少页面被返回。即 q_i 返回的子集是事先不知道的。不清楚这个子集,爬虫将不能决定什么样的查询关键词将有最大的结果覆盖率。再一个是,集覆盖最优解问题被认为是一个 NP 难问题^[4],目前还没有找到一个在多项式时间内解决此问题的有效算法。

本文提出一种在可计算代价内近似最优解的次优解算法。尽管不知道对于每次查询 q_i 将有多少页面被返回,综合实验观察本文提出一种查询选择算法,它可以预测将有多少页面被返回。基于此信息查询选择算法可以选择“最好的或最有希望的”的查询关键词。下一节将详细说明预测方法和查询选择算法。

1.4 性能评价标准

在给出查询选择问题之前,先形式化的定义出性能/代价评估标准。

给定查询 q_i , 使用 $P(q_i)$ 表示对于查询 q_i 站点返回的结果页面数占所有可能结果页面数的比例。例如:如果一个站点总共有 10 000 个页面。若使用查询 q_i = “medicine” 返回了 3 000 个结果页面,则 $P(q_i)$ = 0.3。使用 $P(q_1 \wedge q_2)$ 来表示由查询 q_1 和 q_2 均返回的页面数量占所有可能结果页面数的比例,即 $P(q_1)$ 和 $P(q_2)$ 的交集;同样使用 $P(q_1 \vee q_2)$ 来表示由查询 q_1 和 q_2 一共返回的页面数量占所有可能的结果页面数的比例,即 $P(q_1)$ 和 $P(q_2)$ 的并集。

使用 $Cost(q_i)$ 来表示查询 q_i 的各种代价总和。代价评定标准要么是时间,网络带宽,要么是表单提交次数等,或者是由这些量组成的一个函数。本文提出的算法实际上是没有考虑到代价函数的。通常情况下,查询代价包含多方面的因素,包含提交查询到站点的查询代价,检索索引页面代价和下载实际页面的代价。下载页面的代价是与返回的匹配文档数量成比例的。然而,设提交一次查询的代价为 C_q , 下载每一篇文

档的代价 C_d 。查询 q_i 的总代价可表示为: $Cost(q_i) = C_q + C_r P(q_i) + C_d P(q_i)$, 其中 C_q, C_r, C_d 为常量。

一般情况下,由 q_i 查询所返回的页面中有一部分可能由前一次查询已经下载。这样爬虫就可以跳过这些页面,直接下载新产生的页面,上面的代价函数可修改为: $Cost(q_i) = C_q + C_r P(q_i) + C_d P_{new}(q_i)$ 。

下一节将研究怎样来估计 $P(q_i)$ 和 $P_{new}(q_i)$ 来评估查询 q_i 的总代价。由于实验中算法实际上没有考虑查询代价问题。假设一个一般的代价函数 $Cost(q_i)$, 做了上面的定义后,可以把 Deep Web 爬虫爬行问题形式化定义为:在查询关键词 q_1, \dots, q_n 中寻找一个关键词使得 $P(q_1 \vee \dots \vee q_n)$ 值最大,其约束条件是 $\sum_{i=1}^n Cost(q_i) \leq t$, 其中 t 是提供给爬虫的最大资源数。

2 查询关键字选择策略

查询关键字选择的目标是使用最少的代价下载最大数量的后台数据库内容,考虑如下 3 种策略:

(1) 随机选择:从一词典中随机选择关键词用于提交表单查询。希望通过随机查询返回一定数量的匹配文档资源。

(2) 根据词频选择:通过分析普通的 Web 文档集来获得每一关键词的频率贡献。基于词频贡献,首先使用词频最高的关键词用于 Deep Web 后台数据库的查询提交,然后检索返回的结果。下次使用词频次之的关键词,反复执行这个过程,直到爬虫耗尽其所有资源。希望在 Web 文档集中的词在后台数据库中词频也较高,从而返回更多的匹配文档资源。

(3) 适应性策略:分析以前查询返回的结果文档,然后估计下次使用哪个关键词可以返回最多的文档资源。基于此分析,提出“最有希望”的查询,迭代此过程,直到资源耗尽或者是满足用户要求。

2.1 估计匹配的页面数量

为识别“最有希望”的查询关键词,需要针对下次查询 q_i 估计将返回多少新的文档。假设已经提出查询 q_1, \dots, q_{i-1} , 需要对下次每种潜在查询 q_i 估计 $P(q_1 \vee \dots \vee q_{i-1} \vee q_i)$ 值,比较出其中的最大值。可以通过改写估计函数的形式来计算这个值。

$$\begin{aligned} & P((q_1 \vee \dots \vee q_{i-1}) \vee q_i) \\ &= P(q_1 \vee \dots \vee q_{i-1}) + P(q_i) - P((q_1 \vee \dots \vee q_{i-1}) \wedge q_i) \\ &= P(q_1 \vee \dots \vee q_{i-1}) + P(q_i) - P(q_1 \vee \dots \vee q_{i-1}) P(q_i | q_1 \vee \dots \vee q_{i-1}) \end{aligned}$$

在上面的公式中,可以通过分析查询 q_i 之前已下载的页面来准确计算 $P(q_1 \vee \dots \vee q_{i-1})$ 和 $P(q_i | q_1 \vee \dots \vee q_{i-1})$ 值。 $P(q_1 \vee \dots \vee q_{i-1})$ 是查询 q_i 之前已下载的所有页面总和, $P(q_i | q_1 \vee \dots \vee q_{i-1})$ 是查询 q_i 出现在 q_1, \dots, q_{i-1} 下载页面中的条件概率,也可从分析已下载页面中得到此值。因此要计算 $P((q_1 \vee \dots \vee q_{i-1}) \vee q_i)$ 仅需计算 $P(q_i)$ 的值。考虑如下两种情况来估计此值:

(1) 独立估计量:假设下次查询每个潜在的关键词 q_i 是独立于 q_1, \dots, q_{i-1} 的,此时有 $P(q_i) = P(q_i | q_1 \vee \dots \vee q_{i-1})$ 。

(2) Zipf 估计量:在文献[5]中, Ipeirotis 等提出了一种方法来估计某一文档子集中的特定术语在整个文档集中出现了多少次。此方法利用了一个事实,文本集中的词频规律对此问题有很大贡献。也就是说,根据术语出现的频率排列所有术语的等级,最高等级的标记为 1,次之标记为 2,依此类推。这样在文本集中的术语频率 f 被定义为

$$f = \alpha(\gamma + \beta)^{-r}$$

式中： γ ——术语的排列等级， α, β, r ——由此文本集决定的常量。可以使用上述公式来估计 $P(q_i)$ 的值。

2.2 查询选择算法

Deep Web 爬虫设计的目标是使用有限资源从后台数据库中下载最大数量的惟一的文档资源。为了达到此目标 Deep Web 爬虫设计主要考虑两方面的因素：一方面是选择查询关键词 q_i ，使其可以返回最大数量的新文档；另一方面是查询 q_i 所花费代价问题。例如：对于两个不同的查询选择 q_i 和 q_j ，若在相同代价的情况下，查询 q_i 比 q_j 返回更多数量的相关的新页面，则认为查询 q_i 比 q_j 更有“希望”。基于此发现，这里给出了 Deep Web 爬虫查询选择算法效能评价标准的一个形式化的定义。

$$\text{Efficiency}(q_i) = P_{\text{new}}(q_i) / \text{Cost}(q_i)$$

这里 $P_{\text{new}}(q_i)$ 代表查询 q_i 返回新的页面数占站点可能总页面数的一个比例， $\text{Cost}(q_i)$ 代表查询 q_i 所花费的总代价。直觉上效能评价标准反映的是对于查询 q_i 单位代价下可以获取多少新的文档页面。Deep Web 爬虫可以以此来估计每个查询 q_i 的效能，下一次将要选择的关键词就是效能值最大的那个关键词。图 6 给出了查询选择算法，原则上此算法是个“贪心”算法，它可以获得潜在的最优解。

算法 2

Parameters: T: 潜在的查询关键词列表

Procedure

- (1) For each Tk in T do
- (2) Estimate $\text{Efficiency}(\text{Tk}) = P_{\text{new}}(\text{Tk}) / \text{Cost}(\text{Tk})$
- (3) done
- (4) return Tk with maximum $\text{Efficiency}(\text{Tk})$

图 6 贪婪选择查询关键词算法

可以使用上述算法计算每种潜在查询的效能值。查询 q_i 返回的新文档数占可能文档总数的比例为

$$\begin{aligned} P_{\text{new}}(q_i) &= P(q_i \vee \dots \vee q_{[i-1]} \vee q_i) - P(q_i \vee \dots \vee q_{[i-1]}) \\ &= P(q_i) - P(q_i \vee \dots \vee q_{[i-1]}) P(q_i | q_i \vee \dots \vee q_{[i-1]}) \end{aligned}$$

同样地也可类似的估计 $\text{Cost}(q_i)$ ，若 $\text{Cost}(q_i) = C_q + C_i P(q_i) + C_d P_{\text{new}}(q_i)$ 。可以通过估计 $P(q_i)$ 和 $P_{\text{new}}(q_i)$ 来估计 $\text{Cost}(q_i)$ 。

2.3 查询效能值计算的统计方法

在估计查询效能时，反复执行上述迭代算法是非常耗费时间的。本节介绍一种通过维护代价较小的查询统计表的方法来高效计算 $P(q_i | q_i \vee \dots \vee q_{[i-1]})$ 的值。

其主要设计思想是： $P(q_i | q_i \vee \dots \vee q_{[i-1]})$ 值可以通过查询关键词 q_i 在已经由查询 $q_i, \dots, q_{[i-1]}$ 下载的文档中出现的次数来计算出。记录各关键词在已下载文档中出现的次数于一个表中（如图 7 所示）所示。表的左边一列含有所有可能的查询关键词，右边一列记录了已下载文档中包含此关键词的文档个数。例如：假设当前查询统计表为图 7(a)， $P(\text{数字化} | q_i \vee \dots \vee q_{[i-1]}) = 32/106 = 0.3$ 。当对查询接口提出一个新的查询 q_i 时，图 7(b) 表示了新下载页面后每个候选关键词所出现的文档数。可根据图 7(a)、7(b) 简单地实现旧表到新表 C 的更新。从新表可以看出， $P(\text{数字化} | q_i \vee \dots \vee q_i) = 42/160 = 0.26$ 。通过维护查询统计表来计算查

询效能值要比直接使用公式计算代价要小得多。

Term Tk	N(Tk)
数字化	32
程序设计	65
操作系统	106
总页面数: 106	

(a) 查询 q_i 之前

Term Tk	N(Tk)
数字化	32+10=42
程序设计	65+54=120
操作系统	106+0=106
C++	0+32=32
总页面数为: 106+54=160	

(c) 查询 q_i 执行之后

Term Tk	N(Tk)
数字化	10
程序设计	54
C++	32
新页面数: 54	

(b) 使用查询 q_j 时

图 7 更新查询统计表

2.4 爬行限制返回结果页面数的站点

在一些情况下，当一个查询匹配很多的页面时，Deep Web 站点可能仅返回这些页面中的一部分。显然站点的这种限制对爬虫的爬行策略有直接的影响。首先，由于每次查询只能检索到特定数量的页面，为了下载更多的页面，爬虫将需要使用更多次的查询和用光潜在的所有资源。其次，在前面提到的查询选择算法对每次查询 q_i 得到 $P(q_i | q_i \vee \dots \vee q_{[i-1]})$ 值就不准确了，也就是说对每次查询 q_i 只可以找到整个文本数据库中一部分文档，将影响对下次查询选择的决定。因此，有必要提出一种仅从返回的部分页面来估计 $P(q_i | q_i \vee \dots \vee q_{[i-1]})$ 正确的值的方法。

假设在 Deep web 站点上已经执行过的查询 $q_i, \dots, q_{[i-1]}$ 返回的页面数量小于站点提供的最大数量，同时也下载了以前查询的所有页面。下次将要提交查询 q_i ，由于站点对返回结果数量的限制，检索到的页面集设为 q'_i ，它小于查询 q_i 在不限返回页面数量情况下返回的页面数。为了下次获得准确的估计值需要更新查询统计表。尽管我们得到的是 q'_i ，对下次查询 $q_{[i+1]}$ 有

$$\begin{aligned} P(q_{[i+1]} | q_i \vee \dots \vee q_i) \\ &= 1/P(q_i \vee \dots \vee q_i) * [P(q_{[i+1]} \wedge (q_i \vee \dots \vee q_{[i-1]})) + \\ &\quad P(q_{[i+1]} \wedge q_i) - P(q_{[i+1]} \wedge q_i \wedge (q_i \vee \dots \vee q_{[i-1]}))] \end{aligned}$$

可以通过估计 $P(q_i)$ 的值来计算 $P(q_i \vee \dots \vee q_i)$ 的值。通过检查由查询 $q_i, \dots, q_{[i-1]}$ 已下载的文档来计算 $P(q_{[i+1]} \wedge (q_i \vee \dots \vee q_{[i-1]}))$ 和 $P(q_{[i+1]} \wedge q_i \wedge (q_i \vee \dots \vee q_{[i-1]}))$ 的值。由于 $P(q_{[i+1]} \wedge q_i)$ 值是未知的，可以通过如下方程来计算它

$$\frac{P(q_{[i+1]} \wedge q_i) - P(q_i)}{P(q_{[i+1]} \wedge q'_i) - P(q'_i)}$$

从上面的等式可以计算出 $P(q_{[i+1]} \wedge q_i)$ 的值。

3 实验评估

本节将通过实验来评价上述的 Deep Web 爬虫 3 种爬行策略的效能。适应性选择策略是从下载文档资源中学习新的关键词或术语，其查询关键词的选择由上面所述查询效能值决定。为简化实验细节，假设每次查询的代价是个常量。其中对返回页面数的估计这里使用上述的第 1 种方法，假设每个关键词查询 q_i 是独立于 $q_i, \dots, q_{[i-1]}$ 的，此时有 $P(q_i) = P(q_i | q_i \vee \dots \vee q_{[i-1]})$ 。尽管这种估计方法很简单，但实验表明这种方法很实用。对于普通词频选择策略，实验计算来自 154 个站点的覆

盖多个主题^[6]的550万的页面集中的词频贡献。在文档中出现频率最高的关键词将作为下次查询的选择,次关键词用于下次的选择,依此类推。对于随机选择策略,对各Deep Web站点使用一个来自网络的词集,从中随机选择下次所要查询的关键词。

实验使用所爬行的Deep Web站点是:网易www.163.com,新浪网www.sina.com.cn,和搜狐网news.sohu.com。图8显示了在网易上每种策略的所提交查询次数与检索到的文档覆盖率之间的关系,在其它两站点上实验结果与图8类似。其中横轴代表查询所返回的页面数占整个站点文档数的比例,纵轴代表查询数。从图中可以明显的看出,普通词频选择策略和适应性选择策略覆盖率要远远高于随机选择策略。其次发现当Deep Web站点是关于某个特定主题的时候,普通词频选择策略覆盖率要低于适应性选择策略的覆盖率。例如对于某某站点,仅采用83次查询适应性选择策略就可以达到80%以上的覆盖率,而要获得同样的覆盖率普通词频选择策略要查询106次。

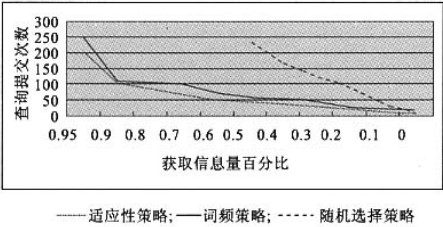


图8 www.163.com 上各种策略信息覆盖率比较

4 结束语

本文研究了Deep Web爬虫自动爬行和下载页面的理论框架,同时提出了3种爬行策略,实验表明这些策略在获取Deep Web页面上有良好的效果。尤其是适应性选择策略通过使用约100次查询可下载Deep Web站点90%以上的信息量。

下一步的工作将是怎样把这种针对单一属性接口爬虫的爬行策略扩展到多属性结构的查询接口上,毕竟Deep Web中,多属性接口后台数据库提供的是结构化的信息,其内容质量更高。除此之外,本文仅提出了针对单一属性接口爬虫爬行策略的一个理论框架,要设计一个可以自动爬行Deep Web站点的爬虫还存在许多实际的问题需要解决。例如:本文的爬

虫首先假设要爬行的接口站点已经知道,然而实际中涉及如何发现这些搜索接口问题,关于这点在文献[8]有所介绍。另一就是一些Deep Web站点含有无限的结果页面,其中有很多内容是不相关的内容,对于这个问题页面相似度算法也许有所帮助。通过更进一步的分析下载的页面内容以了解后台数据库模式,然后使用机器学习等方法分类Deep Web后台数据库^[5,8-11]以集成信息^[12]提供更好的信息检索服务。

参考文献:

[1] Bin He, Mitesh Patel, Zhen Zhang, et al. Accessing the deep web: A survey [EB/OL]. 2004.<http://eagle.cs.uiuc.edu/tr/dwsurvey-tr-hpzc-ju104.pdf>.

[2] Chang K C C, He B, Li C, et al. Structured databases on the web: Observations and implications [C]. *SIGMOD Record*, 33 (3), 2004-09.

[3] Raghavan S, Garcia-Molina H. Crawling the hidden web [C]. *Roma, Italy: Proceedings of the 27th International Conference on Very Large Data Bases*, 2001.129-138.

[4] Cormen T H, Leiserson C E, Rivest R L. Introduction to algorithms [M]. 2nd Edition. MIT Press/McGraw Hill, 2001.

[5] Ipeirotis P, Gravano L. Distributed search over the hidden web: Hierarchical database sampling and selection [C]. *VLDB*, 2002.

[6] Ntoulas A, Cho J, Olston C. What's new on the web? The evolution of the web from a search engine perspective [Z]. *WWW*, 2004.

[7] Barbosa L, Freire J. Siphoning hidden-web data through keyword-based interfaces [C]. *SBBD*, 2004.

[8] Cope J, Craswell N, Hawking D. Automated discovery of search interfaces on the web [C]. *14th Australasian conference on Data Base technologies*, 2003.

[9] He B, Chang K C C. Statistical schema matching across web query interfaces [C]. *SIGMOD Conference*, 2003.

[10] Ipeirotis P G, Gravano L, Sahami M. Probe, count, and classify: Categorizing hidden web databases [C]. *SIGMOD*, 2001.

[11] Liu V Z, Luo J C, Richard C, Chu W W. Dpro: A probabilistic approach for hidden web database selection using dynamic probing [C]. *ICDE*, 2004.

[12] Wang Jiying. Information discovery, extraction and integration for the hidden web [C]. 2002.

(上接第3153页)

参考文献:

[1] 柴晓路. Web 服务架构与互操作技术 [M]. 北京:清华大学出版社, 2002.


[2] 郑小平. .Net 精髓——Web 服务原理与开发 [M]. 北京:人民邮电出版社, 2001.1

[3] 胡新荣, 聂刚. 基于 Web Services 的分布式应用研究 [J]. 控制工程, 2004, (3): 243-250.

[4] 林格, 迈克昆廷, 威尔顿. C# 字符串和正则表达式参考手册 [M]. 北京:清华大学出版社, 2003.

[5] 班纳基, 卡罗纳. C# Web 服务高级编程——使用 .NET Remoting 和 ASP.NET 创建 Web 服务 [M]. 北京:清华大学出版社, 2002.

[6] 韩立巧, 张传生. 基于 XML 技术的 Web 服务的接口设计 [J]. 计算机工程与设计, 2003, (9): 47-49.

作者: 郑冬冬, 崔志明, ZHENG Dong-dong, CUI Zhi-ming
作者单位: 苏州大学, 智能信息处理及应用研究所, 江苏, 苏州, 215006
刊名: 计算机工程与设计 
英文刊名: COMPUTER ENGINEERING AND DESIGN
年, 卷(期): 2006, 27(17)
被引用次数: 8次

参考文献(12条)

1. Bin He; Mitesh Patel; Zhen Zhang [Accessing the deep web: A survey](#) 2004
2. Chang K C C; He B; Li C [Structured databases on the web: Observations and implications](#) [外文期刊] 2004(03)
3. Raghavan S; Garcia-Molina H [Crawling the hidden web](#) [外文会议] 2001
4. Cormen T H; Leiserson C E; Rivest R L [Introduction to algorithms](#) 2001
5. Ipeirotis P; Gravano L [Distributed search over the hidden web: Hierarchical database sampling and selection](#) [外文会议] 2002
6. Ntoulas A; Cho J; Olston C [What's new on the web? The evolution of the web from a search engine perspective](#) 2004
7. Barbosa L; Freire J [Siphoning hidden-web data through keyword-based interfaces](#) 2004
8. Cope J; Craswell N; Hawking D [Automated discovery of search interfaces on the web](#) 2003
9. He B; Chang K C C [Statistical schema matching across web query interfaces](#) [外文会议] 2003
10. Ipeirotis P G; Gravano L; Sahami M [Probe, count, and classify: Categorizing hidden web databases](#) 2001
11. Liu V Z; Luo J C; Richard C; Chu W W [Dpro: A probabilistic approach for hidden web database selection using dynamic probing](#) 2004
12. Wang Jiying [Information discovery, extraction and integration for the hidden web](#) 2002

本文读者也读过(9条)

1. 曾伟辉, 李森, 曾伟辉 [深层网络爬虫研究综述](#) [期刊论文] - 计算机系统应用 2008, 17(5)
2. 林超, 赵朋朋, 崔志明, LIN Chao, ZHAO Peng-peng, CUI Zhi-ming [Deep Web数据源聚焦爬虫](#) [期刊论文] - 计算机工程 2008, 34(7)
3. 李涛, 陈鹏, 李哲, LI TAO, CHEN PENG, LI ZHE [深度Web资源探测系统的研究与实现](#) [期刊论文] - 微计算机信息 2007, 23(33)
4. 徐文杰, 陈庆奎, XU Wen-jie, CHEN Qing-kui [增量更新并行Web爬虫系统](#) [期刊论文] - 计算机应用 2009, 29(4)
5. 黄聪会, 张水平, 胡洋, HUANG Cong-hui, ZHANG Shui-ping, HU Yang [主题Deep Web爬虫框架研究](#) [期刊论文] - 计算机工程与设计 2010, 31(5)
6. 田野, 丁岳伟, TIAN Ye, DING Yue-wei [基于关键词相关度的Deep Web爬虫爬行策略](#) [期刊论文] - 计算机工程 2008, 34(15)
7. 郑冬冬, 赵朋朋, 崔志明, ZHENG Dongdong, ZHAO Pengpeng, CUI Zhiming [Deep Web爬虫研究与设计](#) [期刊论文] - 清华大学学报(自然科学版) 2005, 45(9)
8. 王冉冉, 王刚, 黄青松, WANG Ran-ran, WANG Gang, HUANG Qing-song [基于Deep Web的信息采集系统](#) [期刊论文] - 计算机技术与发展 2007, 17(10)
9. 张亮, 陆余良, 刘金红, ZHANG Liang, LU Yu-liang, LIU Jin-hong [Deep Web入口探测与分类方法研究](#) [期刊论文] -

引证文献(8条)

1. 唐波 网络爬虫的设计与实现 [期刊论文] - 电脑知识与技术 2009 (11)
2. 李贵, 韩子扬, 郑新录, 李征宇 基于Apriori算法的Deep Web网页关系挖掘研究 [期刊论文] - 山东大学学报 (理学版) 2011 (5)
3. 黄聪会, 张水平, 胡洋 主题Deep Web爬虫框架研究 [期刊论文] - 计算机工程与设计 2010 (5)
4. 张云冬, 徐和祥, 胡运发, 邓河 基于个性化图书馆的Deep Web Crawler研究与实现 [期刊论文] - 计算机应用与软件 2009 (4)
5. 周杨 支持Ajax的Deep Web爬虫研究与设计 [期刊论文] - 计算机系统应用 2012 (2)
6. 周二虎, 张水平, 胡洋 基于Deep Web检索的查询结果处理技术的应用 [期刊论文] - 计算机工程与设计 2010 (1)
7. 曾伟辉, 李淼, 曾伟辉 深层网络爬虫研究综述 [期刊论文] - 计算机系统应用 2008 (5)
8. 董旻, 方曙 Deep Web信息抽取研究 [期刊论文] - 图书情报工作 2007 (10)

本文链接: http://d.g.wanfangdata.com.cn/Periodical_jsjgcysj200617015.aspx