基于双配置文件的深度网搜索

蒋秀才[a], 穆 斌[b]
(同济大学 a. 电信学院 计算机系; b. 软件学院, 上海 201804)

摘 要: 深度网搜索的核心部分就是深度网数据库接口的抽取和集成,虽然在理论上提出了很多种方案,并且在特定实验中也有着较好的效果,但是至今仍未将这些方法有效地整合到实际情况中去。首先提出了通过双配置文件的方式来简化深度网的实现,其次提出了一种基于编码方式的接口集成和映射的新方法,最后通过实验证明该框架和编码方法具有良好的实用效果。

关键词: 深度网; 配置文件; 编码; 映射
中图分类号: TP39 **文献标志码**: A **文章编号**: 1001-3695(2008)12-3621-03

Crawling Deep Web with two configurations

JIANG Xiu-cai[a], MU Bin[b]
(a. Dept. of Computer, College of Electronic & Information Engineering, b. College of Software, Tongji University, Shanghai 210804, China)

Abstract: The core of crawling Deep Web is the extraction and integration of Deep Web database interfaces, although many solutions have been formulated in theory, and even possessed lots of satisfied results in the particular experiments, they can't be integrated according to the practice successfully. At the beginning, this chapter demonstrated a novel framework with two configurations to simplify the implement of probing the Deep Web. In addition, put forward a novel approach for integrating and mapping those interfaces based on a code algorithm. Finally, used experiments to prove them well.
Key words: Deep Web; configuration; encode; mapping

深度网又称隐藏网、不可见网,用来定义那些不能通过静态网页链接获取,而是直接通过填入与数据库相连的表单接口动态生成的那部分网页。研究表明,传统搜索引擎只能搜索到约三分之一的深度网,这就意味着还有大量的深度网内容不被察觉。深度网不但含有丰富的内容(大约是静态网容量的 500 倍[1],约有 45 万个数据库和 125.8 万个查询接口[2]),而且所设计到的信息很具有专业性,因此发掘出深度网资源具有很大的现实意义。但是深度网的挖掘不同于静态网,这主要是数据的存储方式不同所导致的。在静态网中,数据大部分存储在半结构化的网页中,而在深度网中,数据主要存储在数据库中。研究表明,深度网中的数据结构化与非结构化比例大约为 3.4 1,并且约 94% 的查询接口隐藏在深度不会超过 3 的网络站点中[2],因此对于深度网要采用不同的搜索方法。

搜索深度网可以分为以下几个方面: a) 从网页上获取表单; b) 对表单进行关键字抽取并集成; c) 填充表单并提交; d) 分析返回的结果。当然每个部分还可以分成更小的部分,最近的一些研究工作也是基于这些方面的。然而,对于深度网的研究,各个部分的研究成果也各不一致,有些方面已经日趋成熟,如表单关键字的抽取和匹配以及表单的集成等;而有些方面只是一些探索和假设,如语义的分析等。

1 系统的基本结构

1.1 系统的框架

本文提出的深度网搜索的基本框架如图 1 所示。首先,用户将自己所要查询的信息进行归类集成,选取或输入与查询信息所关联的一系列关键词,然后发送给控制中心。控制中心控制处理器的处理,并根据配置文件 1 生成与用户相符的查询接口传递给用户进行查询输入。用户输入查询语句之后,将结果再次发送到控制中心,控制中心又将它发送到同一处理器。处理器再次读取配置文件 2 进行处理,并将得到的结果返回给页面处理器进行页面内容抽取筛选,最后内容进行重复处理后返回给用户。

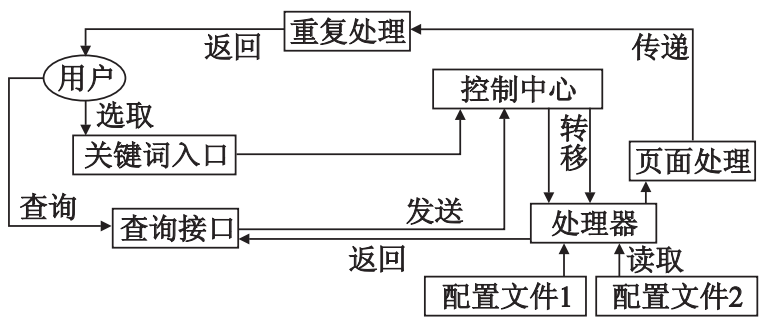

图1 系统框架

1.2 关键词入口和查询接口

用户要查询信息就要输入一系列的关键词作为对查询信息的描述,如购书,就要选择书名、作者、ISBN、价格等一些关键词,当然这里的关键词要与深度网的表单接口属性相联系。一旦你选择了关键词后,系统会根据这些关键词自动生成查询接口,然后就可以输入所要查询的信息,但是这里生成的查询接口并不是单个接口,而是某个域中一系列接口的集成。因此选择好关键词就意味着选择了一个集成查询接口。

收稿日期: 2008-03-28; 修回日期: 2008-05-04
作者简介: 蒋秀才(1983-),男,安徽马鞍山人,硕士研究生,主要研究方向为数据挖掘和搜索引擎(jxiucai@163.com);穆斌(1964-),男,副教授,主要研究方向为数据挖掘、语义网络.

1.3 控制中心

这个部分在该框架中起着控制和传递数据的作用。

1.4 配置文件 1

该配置文件是框架中的核心文件之一,主要用来存储集成接口的关键词以及这些关键词的限定域和类型等信息。对于关键词,它的值域可以分为有限型和无限型。对于有限型值域应该完全列举出来;而对于无限型值域,应该将出现频率高的值作为可能选择的默认值,这样就能极大地提高网页查找率和准确率,降低对关键词误用或乱用的可能性。

1.5 配置文件 2

该配置文件是框架中的另一个核心文件,也可以说是整个系统的中心点,因为它全面阐述了集成接口与各个子接口之间的联系,虽然只是涉及到接口间的映射关系,但是仍然需要从子接口的获取、集成等方面来得到这些映射关系。下面将探讨映射所涉及的方面以及映射关系的生成。

1.5.1 表单接口的获取

关于这个接口的获取,至今并没有一个行之有效的方法,因为无论是 IP 穷举法还是迭代法,其缺点和优点均是一样的明显。最近,文献[3,4]提出了采用 DynaBot 的方法来对深度网进行搜索和聚集,但是在整个 DynaBot 中主要依靠 SCD 来进行标准判断,而 SCD 的建立又具有不确定性,最后在 Source-Biased 分析阶段不但工作量大,而且效果也很差。

1.5.2 表单接口的融合

将一系列同领域的表单接口抽取出来之后,就要按照它们之间的联系进行集成。关于集成方面的研究,最近一直很热,也很成熟。其中,文献[5]阐述了在 e-commerce 中进行数据库查询接口的集成策略;文献[6]提出了相关属性的集成方法;文献[7]论证了通过对深度网接口模式匹配的方法来聚集接口;文献[8]论述了通过对属性寻找值的方法来进行匹配;文献[9]提出了一个 MetaQuerier 整合模型;文献[10]又论述了一个 WISE-integrator 模型,它比 MetaQuerier 模型更全面,所有这些方法均在某些方面很好地解释和集成了接口。关于表单的集成,本文在 1.9 节中提出了一种新的方法。

1.5.3 映射关系

映射关系体现了信息从集成接口如何分发到各个子接口,而映射效果的好坏将直接关系到各个子接口中填充信息的完整性和准确性,1.9 节将阐述一种高效的映射方法。

1.6 处理器

处理器是整个系统中真正的执行者,而两个配置文件则是指导处理器应该如何进行查询信息的分发,如何去搜索那些与接口相连的在线数据库。

1.7 页面处理

关于页面处理,文献[11]提出了一种 Thor 结构;文献[12]阐述了一种 Turbo wrapper 框架;文献[13]论述了一种 Testbed 模型。虽然它们在有效性方面有待提高,但是均能很好地抽取到一定的结果,而现在的抽取工作主要也是通过各种分装器来完成的。

1.8 重复处理

该过程属于系统中的优化过程,主要目的就是对抽取的结果进行去冗余和排序,返回最有可能满足客户的信息。

1.9 基于编码方式的接口集成和映射

1.9.1 树模型

设定一个接口模式为 $\eta=(\varepsilon_1,\varepsilon_2,\cdots,\varepsilon_i)$。其中:$\varepsilon_i$ 表示接口中所拥有的互不相同的属性。考虑一个同域的接口模式集合 $S=(\eta_1,\eta_2,\cdots,\eta_j)$,那么 $A=\bigcup_{n=1}^{j}\eta_n$ 则表示集合 S 中相异的属性集合。对于集合 A 中的每一个属性 ε_i,又可以用 $\delta_i=\sum_{i=1}^{j}N_{\eta_i}$ 来表示集合中该属性出现的次数。其中:$N_{\eta_i}\in(0,1)$,即如果该接口 η_i 含有属性 ε_i,那么 N_{η_i} 为 1,否则为 0,因此 A 中每个属性频率形成的集合为 $\varphi=(\delta_1,\delta_2,\delta_3,\cdots,\delta_q)$。最后再用 ϕ_t 表示第 t 个属性的权值,即 $\varphi_t=\delta_t/\sum_{p=1}^{q}\delta_p$,表示接口模式中该属性所占的比重。将这些权值进行降序排列,得到的集合设为 $W=(\varphi_1,\varphi_2,\varphi_\cdot,\varphi_q)$,与此顺序相对应的 A 中元素序列设为$(A_1,A_2,\cdots,A_q)$,当得到这些值后就用算法 1 来构造树。

算法 1

a) 如果属性 A_i 含有子元素设为$(A_{x_1},A_{x_2},\cdots,A_{x_d})$,那么从属性集合中去掉这些元素并且修改其父属性的权值,$\varphi_i'=\varphi_i+\Delta\Psi(\sum_{i=1}^{d}A_{x_i})$。其中:$\Delta\Psi$ 为设定的一个阈值。

b) 对剩下的元素再进行一次降序排列,设为 $F=(A_1,A_2,\cdots,A_m)$ 构成 m 棵二叉树 $T=\{T_1,T_2,\cdots,T_m\}$。其中:每棵 T_i 二叉树中只有一个权值为 ϕ_i 的根节点,它的左右子树均为空。

c) 在 F 中选取两棵根节点权值最小的树作为新构造的二叉树的左右子树(设定左子节点的权值小于右子节点的权值),新二叉树的根节点的权值为其左右子树的根节点的权值之和。

d) 从 F 中删除这两棵树,并将这棵新的二叉树加入到集合 F 中。

e) 重复过程 b),直到只剩下一棵树为止,这棵树就是本文所求的树。

f) 设左支路为 0,右支路为 1,将该树上的叶子进行编码,于是便得到了所有属性的编码集合 $M=(\beta_1,\beta_2,\cdots,\beta_m)$。

g) 将子元素以冗余码的方式添加到父属性的编码中。

采用算法 1,笔者将所有的属性均编译成了一段编码,根据本文的理论可知,$\gamma=\sum_{i=1}^{m}\varphi_i L_i$ 是最小的。其中:L_i 表示从根节点到第 i 个叶子的分支数目。

1.9.2 译码

一旦将属性编码之后,剩下的工作就是对用户输入的编码进行编译得到属性,可以由算法 2 来进行。

算法 2

输入:编码 φ;

输出:$\vartheta=\phi$,属性编码集合 $M=(\beta_1,\beta_2,\cdots,\beta_m)$,与之对应的属性集合为 $F=(A_1,A_2,\cdots,A_m)$,子元素的降序排列集合 $L=(A_{x_1},A_{x_2},\cdots,A_{x_d})$

```
n = 0;
while ( n < length(φ) ) {
   start n;
   for β1 to βm{
   if( equal( β1)
      { n = n + length( βi) ; ϑ= ϑ∪ Ai;
      if( nextchar = ( )
      {  find subelements from L;
      ϑ = ϑ∪( subelements) ;
      n = n + Length( subelementCodes) + 2;  }
      }
   }
}
```

当用户从统一接口输入编码 φ后, 经过解析生成属性序列 , 再生成查询接口。该方法是根据用户的选择动态生成集成接口, 不必拘束于某一个集成接口, 因此具有很大的优越性。当然仍可以对算法的匹配进行改进: 设由属性 $F=(A_1, A_2, \cdots, A_m)$ 组成的树为 T_f, 然后再对 T_f 树进行高度为 $\vartheta=(\vartheta_1, \vartheta_2, \vartheta\cdot, \vartheta_q)$ 的商划分, 形成深度为 q 的索引, 而对于属性中的子元素, 以冗余码的形式放在主属性的后面, 这样就能最大限度地减少比较的次数。

1.9.3　接口的映射

统一接口动态形成后, 从统一接口映射到各个接口便是本文讨论的重点。本文设所有接口模式的相异属性集合为 $F=(A_1, A_2, \cdots, A_q)$, 对于每一个接口模式用二进制代码来表示。其形成和映射如算法 3 所示。

算法 3

a) 将集合 F 按照属性权值降序排列, 如果该属性含有子元素, 则从中去除子元素, 并以降序排列的方式添加到另一个与对应父属性相连的集合中, 变化后的 $F'=(A_1, A_2, \cdots, A_m)$, 子元素的排列为 $(A_{x_1}, A_{x_2}, \cdots, A_{x_d})$。

b) 对每一个接口模式设为 $\eta_t=(\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_i)$, 对于 F 中每个元素采用以下规则进行编码:

$$\tau_i=\begin{cases}1 & A_i\in\eta_t\\ 0 & A_i\notin\eta_t\end{cases}\quad(1\le i\le m)$$

因此该接口模式的编码为 $s_t=(\tau_1, \tau_2, \cdots, \tau_m)$。对于子元素采用同样方式以冗余码的形式添加到接口模式的编码中。

c) 设集合 $\Re=(s_1, s_2, s\cdot, s_j)$ 表示所有接口模式编码的集合, 然后进行卡诺图化简。对于父元素的集成要加上子元素的限制条件。

d) 对集成的编码采用索引相连, 运行从集成接口到子接口的映射时, 将客户选择的属性采用以上的方式进行编码后与索引中元素相与, 再对结果异或, 值为 0, 则继续匹配下去; 否则搜索另一元素, 直至找到相关子接口。

e) 去重复, 得到相异子接口, 这些子接口便是映射的结果。

2　实验结果

这次实验主要是关于 book 这个域的研究, 然后用召回率和准确率来验证框架和编码方法的实效性。关于实验中查询接口模式的抽取, 可以使用文献[6, 7, 9, 10] 来得到; 为了实验的简化, 实验中所用的接口模式直接采用文献[4] 所提供的数据, 总共有 55 个接口和 247 个属性。根据对各个属性统计, 其结果如表 1 所示。

表 1　接口模式的属性统计

attribute	title	author	ISBN	key word
frequent	54	46	44	31
P/%	21.8	18.6	17.8	12.5
attribute	publisher	subject	last name	price	format
frequent	14	11	8	7	7
P/%	5.8	4.5	3.2	2.8	2.8
attribute	category	first name	publication date	binding	publish
frequent	7	6	5	4	3
P/%	2.8	2.5	2.0	1.6	1.3

因为 first name 和 last name 是 author 属性的子元素, 本文将 first name、last name 进行删除, 并且修改 author 的 ratio。设 $\Delta\psi=0.75$, 则 $\delta'=0.186+0.75\times(0.025+0.032)=0.229$。以下用 A、B、C、D、E、F、G、H、I、J、K、L 来代替这些排序后的属性, 按照算法 1 来构造树模型, 其最终结果如图 2 所示。

根据图 2 可以给属性编码, 如下所示: A: 11, B: 10, C: 001, D: 011, E: 0000, F: 00011, G: 01011, H: 01010, I: 01001, J: 01000, K: 000101, L: 000100。又因为 A 含有子元素, 需要给 A 加上一段冗余码, (00) 表示无子元素, (10) 表示只含有 first name, (01) 表示只含有 last name, 而(11) 表示两个子元素均含有。对于算法匹配的改进, 可以设定 = { 3} , 那么可以分为两部分, 第一部分为(10, 11, 001, 000* , 011, 010*) ; 第二部分为(0, 100, 101, 11) 、(00, 01, 10, 11) 。

利用算法 3 可以对每个接口生成以下编码(只选取其中的 10 个来解释) , 如表 2 所示。

表 2　接口模式编码

scheme name	code	scheme name	code
scheme1	0(00) 100 1110 0000	scheme6	0(11) 111 0010 0000
scheme2	1(00) 110 1010 0000	scheme7	1(00) 111 0010 0000
scheme3	1(00) 100 0010 0000	scheme8	1(00) 110 1110 0000
scheme4	1(00) 011 0010 0000	scheme9	1(00) 101 1010 0000
scheme5	1(00) 110 0010 0000	scheme10	0(11) 101 0010 0000

由上面的编码可知, 属性 G、H、I、J、K、L 可以不予考虑, 只需关心剩下的那些属性。根据卡诺图化简可以得到如下的集合类: 0(11) 1* 100{ 0(11) 10100, 0(11) 11100} ; 1(00) * 1100{ 1(00) 01100, 1(00) 11100} ; 1(00) 1101* { 1(00) 11010, 1(00) 11011} ; 1(00) 1* 000{ 1(00) 10000, 1(00) 11000} ; 1(00) 11* 00{ 1(00) 11000, 1(00) 11100} ; 0(00) 10011; 1(00) 10110。

以上集合便是本文所求的一级映射索引, 当属性很多不好集成时, 可以采用部分编码集成的办法, 即有选择地集成部分编码。编程后, 用户界面如图 3 所示。在最后的结果抽取中, 使用分装器来抽取, 得到的召回率和准确率分别为 80% 和 91%, 达到了笔者预期的效果。

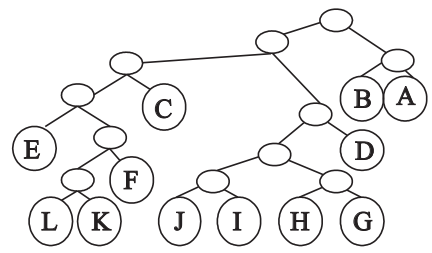

图2　book域的树模型

图3　用户界面

3　结论

通过以上的实验验证, 虽然召回率不高, 且部分工程是由手工来完成的, 但该深度网搜索框架还能够较好地完成笔者所期望的效果。文献[15] 阐述了一个 HIWE 模型, 但在本文所提出的框架中只需要两个简单的配置文件就可以完成所有的功能。不仅如此, 该框架将查询实现与接口的实现通过两个配置文件相分离, 这样有助于更好地完善接口的发现、集成等问题。另外对新增加的接口很容易融合到以前的集成接口中, 更重要的是, 本文所提出的基于编码的接口集成有利于接口自动化集成的实现。当然, 该框架中的某些环节还有很大的优化空间, 关于框架的进一步改进也在笔者下一步研究之中。

4　结束语

Scirus 和 Google Scholar 无疑是当前最有影响力的两个专业的搜索引擎, 它们都是致力于对某一领域的信息进行深层的挖掘, 并且对网页信息采用了结构化信息抽取。毫无疑问, 它们都是深度网的实例化, 它们的成功也正预示着深度网远大的前景。

(下转第 3627 页)

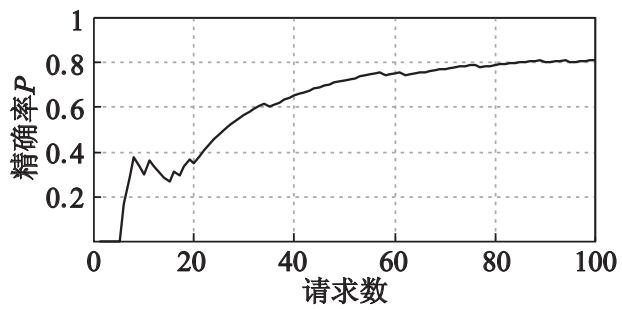

图4　系统精确率实验结果

4　结束语

通过上述实验验证不难发现,基于隐式经验约束理论规则的服务发现方法在服务发现的实践中是可行的,尤其是对智能服务发现和服务组合技术的发展能起到积极的推动作用。笔者在今后的工作中可以针对其中的近似度算法作一定的改进,使系统性能进一步得到提高,还能在隐式经验约束理论规则的自动生成方面开展进一步的研究工作。与此同时,因为本系统的设计都是在假定服务请求者与发现者之间是相互信任的基础上展开的,因此,在今后的工作中也可以针对安全方面对系统作进一步的改进和提高。

参考文献:

[1] ARTIN D. OWL-S: semantic markup for Web services[EB/OL]. (2004) [2008-03-20]. http://www.w3.org/Submission/OWL-S.

[2] KELLER U, LARA R, POLLERES A. WSMO Web service discovery [EB/OL]. (2004) [2008-03-22]. http://www.wsmo.org/2004/d5/d5.1.

[3] BOVA R, PAIK H Y, BEWATALLAH B, *et al*. Task memories and task forums: a foundation for sharing service-based personal processes [C]//Proc of International Conference on Service Oriented Computing Berlin: Springer-Verlay, 2007:365-376.

[4] KERRIGAN M. Web service selection mechanisms in the Web service execution environment[C]//Proc of ACM Symposium on Applied Computing. New York: ACM Press, 2006:1664-1668.

[5] MANIKRAO U S, PRABHAKAR T V. Dynamic selection of Web services with recommendation system[C]//Proc of International Conference on Next Generation Web Services Practices. Washington DC: IEEE Computer Society, 2005:117-121.

[6] SHERCHAN W, LOKE S W, KRISHNASWAMY S. A fuzzy model for reasoning about reputation in Web services[C]//Proc of ACM Symposium on Applied Computing. New York: ACM Press, 2006:1886-1892.

[7] CLAYPOOL M, LE P, WASEDA M, *et al*. Implicit interest indicators[C]//Proc of ACM Intelligent User Interfaces Conference. New York: ACM Press, 2001:33-40.

[8] BIRUKOU E. IC-Service: a service-oriented approach to the development of recommendation systems[C]//Proc of ACM Symposium on Applied Computing. New York: ACM Press, 2007:1683-1688.

[9] KOKASH N, BIRUKOU A, D'ANDREA A. Web service discovery based on past user experience[C]//Proc of International Conference on Business Information Systems. Berlin: Springer-Verlag, 2007:95-107.

[10] BLANZIERI E, GIORGINI P, MASSA P, *et al*. Implicit culture for multi-agent interaction support[C]//Proc of the 9th International Conference on Cooperative Information Systems. London: Springer-Verlag, 2001:27-39.

[11] SECO N, VEALE T, HAYES J. An intrinsic information content metric for semantic similarity in word-net[C]//Proc of the 16th European Conference on Artificial Intelligence. [S.l.]: IOS Press, 2004:1089-1090.

(上接第 3623 页)

参考文献:

[1] ERGMAN M K. The Deep web: surfacing hidden value[J]. Journal of Electronic Publishing, 2002,7(1):8912-8914.

[2] HE Bin, MITESH P, ZHANG Zhen, *et al*. Accessing the Deep Web: a survey[J]. Communication of the ACM, 2007,50(5):94-101.

[3] ROCOO D, CAVERLEE J, LIU Ling, *et al*. Exploiting the Deep Web with DynaBot: matching, Probing, and ranking[C]//Proc of the 14th International Conference on World Wide Web. New York: ACM Press, 2005:1174-1175.

[4] ROCOO D, LIU Ling, CRITCHLOW T. Focused crawling of the Deep Web using service class descriptions[D]. Carrollton: University of West Georgia, 2005.

[5] HE Hai, MENG Wei-yi, YU C, *et al*. WISE-integrator: an automatic integrator of Web search interfaces for e-commerce[C]//Proc of the 29th International Conference on Very Large Databases. [S.l.]: VLDB Endowment, 2003:503-505.

[6] KABRA G, LI Cheng-kai, CHANG K C C. Query routing: finding ways in the maze of the Deep Web[C]//Proc of International Workshop on Challenges in Web Informational Retrieval and Intergration. 2005:64-73.

[7] WU Wen-sheng, DOAN A H, YU C. Merging interface schemes on the Deep Web via clustering aggregation[C]//Proc of the 5th IEEE International Conference on Data Mining. Washington DC: IEEE Computer Society, 2005:801-804.

[8] WU Wen-sheng, DOAN A H, YU C. WebIQ: learning from the Web to match Deep Web query interfaces[C]//Proc of the 22nd International Conference on Data Engineering. 2006:44.

[9] HE Bin, ZHANG Zhen, CHANG K C C. Knocking the door to the Deep Web: integrating Web query interfaces[EB/OL]. (2007-09). http://metaquerier.cs.uiuc.edu/.

[10] HE Hai, MENG Wei-yi, YU C, *et al*. WISE-integrator: a system for extracting and integrating complex Web search interfaces of the Deep Web[C]//Proc of the 31st International Conference on Very Large Databases. [S.l.]: VLDB Endowment, 2005:1314-1317.

[11] CAVERLEE J, LIU Ling, BUTTLER D. Probe, cluster and discover: focused extraction of QA-pagelets from the Deep Web[C]//Proc of the 20th International Conference on Data Engineering New York: ACM Press, 2004:103-114.

[12] CHUANG S L, CHANG K C C, ZHAI C X. Collaborative wrapping: a turbo framework for Web data extraction[C]//Proc of the 23rd International Conference on Data Engineering. 2007:1261-1262.

[13] YAMADA Y, CRASWELL N. Testbed for information extraction from Deep Web[C]//Proc of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. New York: ACM Press, 2004:346-347.

[14] The UIUC Web integration repository[EB/OL]. (2003). http://metaquerier.cs.uiuc.edu/repository.

[15] RAGHAVAN S, GARICA-MOLINA H. Crawling the hidden Web, 2000-36[R]. Stanford: Stanford University, 2000.