doi:10.11959/j.issn.1000-436x.2016049

基于属性值序列图模型的 deep Web 新数据发现策略

鲜学丰[1,2,3]，崔志明[1,2]，赵朋朋[2]，方立刚[1,3]，杨元峰[1,3]，顾才东[1,3]
（1. 江苏省现代企业信息化应用支撑软件工程技术研发中心，江苏 苏州 215104；
2. 苏州大学智能信息处理及应用研究所，江苏 苏州 215006；3. 苏州市职业大学计算机工程学院，江苏 苏州 215104）

摘 要：针对数据源新产生数据记录的增量爬取问题，提出了一种 deep Web 新数据发现策略，该策略采用一种新的属性值序列图模型表示 deep Web 数据源，将新数据发现问题转化为属性值序列图的遍历问题，该模型仅与数据相关，与现有查询关联图模型相比，具有更强的适应性和确定性，可适用于仅仅包含简单查询接口的 deep Web 数据源。在此模型的基础上，发现增长节点并预测其新数据发现能力；利用互信息计算节点之间的依赖关系，查询选择时尽可能地降低查询依赖带来的负面影响。该策略提高了新数据爬取的效率，实验结果表明，在相同资源约束前提下，该策略能使本地数据和远程数据保持最大化同步。

关键词：deep Web；新数据发现；数据获取
中图分类号：TP392 **文献标识码：**A

Deep Web new data discovery strategy based on the graph model of data attribute value lists

XIAN Xue-feng[1,2,3], CUI Zhi-ming[1,2], ZHAO Peng-peng[2], FANG Li-gang[1,3], YANG Yuan-feng[1,3], GU Cai-dong[1,3]
(1. Jiangsu Province Support Software Engineering R&D Center for Modern Information Technology Application in Enterprise, Suzhou 215104,China;
2. Institute of Intelligent Information Processing and Application, Soochow University, Suzhou 215006, China;
3. School of Computer Engineering, Suzhou Vocational University, Suzhou 215104, China)

Abstract: A novel deep Web data discovery strategy was proposed for new generated data record in resources. In the approach, a new graph model of deep Web data attribute value lists was used to indicate the deep Web data source, an new data crawling task was transformed into a graph traversal process. This model was only related to the data, compared with the existing query-related graph model had better adaptability and certainty, applicable to contain only a simple query interface of deep Web data sources. Based on this model, which could discovery incremental nodes and predict new data mutual information was used to compute the dependencies between nodes. When the query selects, as much as possible to reduce the negative impact brought by the query-dependent. This strategy improves the data crawling efficiency. Experimental results show that this strategy could maximize the synchronization between local and remote data under the same restriction.

Key words: deep Web, new data discovery, data acquisition

1 引言

目前，主流搜索引擎还只能搜索 Internet 表面可索引的信息，在 Internet 深处还隐含着大量通过主流搜索引擎无法涉及的海量信息，这些信息称之为深层网页(deep Web，又称为 invisible Web 或 hidden Web)。根据 Bright Planet 研究表明，deep Web 信息量非常庞大，是可索引 Web 信息的 500 倍，并

收稿日期：2015-04-20；修回日期：2015-10-28
通信作者：崔志明，CZM@jssvc.edu.cn
基金项目：国家自然科学基金资助项目（No.61440053, No.61472268, No.41201338）；江苏省自然科学基金资助项目（No.BK2012164）；苏州市科技计划基金资助项目（No.SYG201342, No.SYG201343, No.SS201344）
Foundation Items: The National Natural Science Foundation of China (No.61440053, No.61472268, No.41201338), The Natural Science Foundation of Jiangsu Province (No.BK2012164), Suzhou Foundation for Development of Science and Technology (No.SYG201342, No.SYG201343, No.SS201344)

且这些 deep Web 内容 95%都是可以通过 Internet 无需付费注册就可以公开访问的。deep Web 的信息一般存储在服务端 Web 数据库中，与静态页面相比通常信息量更大、主题更专一、信息质量和结构更好。为了方便用户快捷高效地使用 deep Web 信息，国内外学者对 deep Web 数据集成进行了广泛的研究。目前，deep Web 信息集成主要有 2 种实现方案。一种方案是基于元搜索的方法，针对某个领域提供统一的查询接口，将用户查询经过语义映射转发到各个 deep Web 数据源上，返回的结果经过抽取、语义标注、去重合并呈现给用户。该方案不需维护本地数据库，但存在如下不足：用户的查询响应慢，响应时间不可控，由远程数据源的服务质量决定；建立和维护统一查询接口模式与各个数据源接口模式的语义映射代价高。另一种方案是 deep Web 数据本地化集成方案[1~4]，该方案将 deep Web 数据库中内容爬取出来，经过数据抽取、语义标注、实体识别和去重等处理后，使其以结构化的形式存储于本地数据库，它能在最短时间内响应用户的查询要求。Madhavan 等[1]提出了一种新的基于 DataSpace 的数据集成框架 PayGo，该集成框架具有 Pay-As-You-Go 和演化特性，具有构建成本低、领域独立、演化等特点。目前，第 2 种方案正受到越来越多国内外研究学者的关注，将成为 deep Web 数据集成研究的主流。该方案中的关键问题是如何让本地数据和远程数据源中数据保持同步。本文将致力于该关键问题的研究，在相同更新资源条件下，使本地数据和远程数据保持最大化同步。

由于 deep Web 是自治的、独立更新的，其数据经常处于频繁更新的状态，而用户总是希望能够得到当前 Web 数据库中最新的内容。因此需要定期地更新本地数据拷贝，以保持和远程数据源同步。由于不同的 deep Web 数据源或同一个 deep Web 数据源中的数据记录变化频率是不一样的，按统一频率更新本地存储的所有数据，这是非常耗费资源的(包括带宽、远程数据源服务器资源等)。deep Web 处于快速动态更新的状态，使增量维护变得更加复杂，因此亟需提出新的方法来自动增量更新本地 deep Web 数据，从而在相同资源约束前提下，提高本地数据的时新性和新数据的发现效率。deep Web 数据增量爬取主要包含 2 部分内容：系统已集成的本地数据（消失和改变的记录）的增量更新和新数据发现。目前，新数据的增量发现问题还有待进一步研究，因此本文将针对该问题开展研究，在相同资源约束前提下，获得尽可能多的新数据，使本地数据和远程数据保持最大化同步。

2　相关研究

互联网中信息量的快速增长使增量信息爬取技术成为网上信息获取的一种有效手段，针对浅层网页(surface Web)的网页变化和增量爬取技术已得到广泛的关注和研究[5~7]。然而 surface Web 和 deep Web 存在较大的差异，数据增量爬取的最大区别为：surface Web 页面有固定的 URL，更新一个本地网页只需根据这个 URL 重新访问。然而 deep Web 数据记录无固定的 URL，更新无法根据固定的 URL 来访问。对于一个 deep Web 数据记录，只能通过在 deep Web 数据源的查询接口上提交与该数据记录相关的查询，才能更新该数据记录。因此，不能直接应用 surface Web 的增量爬取技术来实现 deep Web 数据增量爬取，不得不研究新的方法解决 deep Web 数据增量爬取问题。

目前，国内外学者对 deep Web 数据增量爬取也开展了一些探索性研究，文献[8]提出一种基于查询关联图模型的增量数据爬取方法，该方法首先建立数据源的查询关联图模型，从而将增量爬取任务转化为图遍历过程，然后通过分析 deep Web 数据源的历史版本选择查询来增量爬取新记录。该图模型的复杂性由数据源的数据记录数和查询接口查询能力决定。文献[9]同样基于查询关联图模型，但与文献[8]的查询选择依据不同，文献[9]通过分析 deep Web 数据源的样本选择查询爬取新记录。该图模型与 deep Web 数据源提供的查询接口能力密切相关，对于 2 个具有相同内容 deep Web 数据源，如果它们的查询接口在查询能力上不同，那么所产生的查询关联图模型也不相同。因此，这种图模型不能独立表示 deep Web 数据源的内容，具有一定的不确定性；同时该图模型不适用于仅仅包含简单查询接口的 deep Web 数据源。文献[10]把 deep Web 数据爬取表示为集合覆盖问题，通过机器学习方法获得增量回报模型，然后根据增量回报模型自动选择查询爬取增量记录。还有一些针对直接集成 deep Web 网页集成系统的增量爬取的研究[11~13]，文献[11]提出一种基于 URL 分类的 deep Web 增量爬虫，该爬虫根据 deep Web 网页的内容将其分为列表页面和叶子页面，所有 URL 也被分为列表 URL 和叶子

URL，该爬虫根据列表页面的更新频率和叶子页面的变化频率爬取最新的 deep Web 页面。这类研究的 deep Web 集成方式与本文不同，它属于 deep Web 数据集成研究的另一个分支。虽然国内外学者已对 deep Web 数据增量爬取问题进行了一定的研究，但这些研究的新数据发现效率还有待进一步提高。deep Web 新数据发现问题研究目前仍处于探索阶段，其中，许多问题仍需进一步深入研究。

本文提出了一种基于属性值序列图模型的 deep Web 新数据发现策略，该策略首先将 deep Web 数据源的本地数据表示为数据记录属性值序列图，然后根据历史数据产生增长节点，并设计了增长节点的新数据发现能力计算方法，最后基于增长节点的新数据发现效率进行选择查询节点以爬取新数据记录。实验结果表明基于属性值序列图模型的 deep Web 新数据发现策略在相同资源约束前提下，提高了新数据的发现效率，使本地数据和远程数据保持最大化同步。

3 属性值序列图模型

3.1 属性值序列图模型定义

定义 1 deep Web 数据源的属性值序列图模型。结构化的 deep Web 数据源（WDB）可看作一张关系数据表 DB，DB 包含的数据记录为 $Rec=\{r_1,r_2,\cdots,r_n\}$，每条记录包含 m 个属性 $A=\{a_1,a_2,\cdots,a_m\}$，WDB 中所有不相同的属性值组成的集合为 DAV。在 Deep Web 数据增量爬取中，对于一个给定 WDB，它的属性值序列图模型可定义为一种带权的有向连通图，表示为 $G(V,S,T,E)$，其中 V 是节点的集合，每一个节点 $v_i\in V$ 都与 WDB 中的属性值 $av_i\in DAV$ 一一对应。$I(v_i)$表示节点 $v_i\in V$ 的入度，$O(v_i)$表示节点 $v_i\in V$ 的出度。每个节点附带一个权值 vw 表示该节点对应属性值在 WDB 的数据记录集合 Rec 出现的次数，如果不考虑同一个属性值在一个数据记录中出现多次的情况，则节点 $v_i\in V$ 的权值 $vw(v_i)$为在 WDB 中出现属性值 $av_i\in DAV$ 的数据记录数。S、T 分别为出现在单条数据记录中的起点属性值和终点属性值的集合，E 为有向边的集合，E 中的元素表示数据记录中属性值的接续关系，一个有向边$(v_i, v_j)\in E$ 当且仅当 av_i 和 av_j 接续出现在一个数据记录 $r_i\in WDB$ 中（如图 1 所示）。每条边也附带一个权值 ew 表示该边所关联的节点间的关联度，边$(v_i,v_j)\in E$ 的权值 $ew(v_i,v_j)$为包含有向边$(v_i,v_j)\in E$ 的数据记录数。

属性值序列图模型 G 具有如下特点。

1) 在属性值序列图 G 中，$S\subset V$，$T\subset V$ 且有 $|S|\geq 1$，$|T|\geq 1$，$S\cap T$ 可以不为 NULL。

2) 对于属性值序列图 G 任意一个节点 v，有 $I(v)=O(v_i)\geq 1$。

3) 每条数据记录在属性值序列图 G 中被表示为一个有序闭环。

4) 对于属性值序列图 G 中的任意一个节点 $v_i\in V$，根据 S、T、数据记录的长度以及与 $v_i\in V$ 相关联的有向边，可以确定属性值序列图 G 中包含节点 $v_i\in V$ 的所有数据记录。

5) 边$(v_i,v_j)\in E$ 和$(v_j,v_i)\in E$ 属于不同的边，$(v_i,v_j)\in E?(v_j,v_i)\in E$。

3.2 图模型的构建

对一个给定的结构化的 deep Web 数据源，假设在前期的数据爬取中已获得了该 WDB 的全部数据记录，获得的数据记录集为 $Rec=\{r_1,r_2,\cdots,r_n\}$。首先基于 Rec 中一条数据记录生成初始属性值序列图 G（如图 1 所示），然后依次从 Rec 中提取各数据记录，不断向 G 中的添加节点和边，同时也更新节点和边的权值，直到 Rec 中所有数据记录都已添加到 G，最终得到 WDB 的属性值序列图。属性值序列图构建示意如图 1 和图 2 所示。

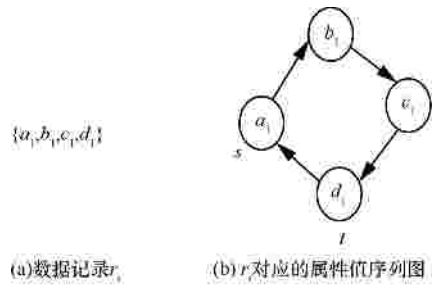


图 1 一条数据记录生成的属性值序列

图 1(a)为一个数据记录 r_i，包含 4 个属性值。图 1(b)为图 1(a)数据记录对应的属性值序列图，a_1 为起始节点，d_1 为终止节点。有向边$(v_i,v_j)\in E$ 当且仅当 av_i 和 av_j 接续出现于一个数据记录 $r_i\in WDB$ 中，数据记录 r_i 在属性值序列图中被表示为一个有序闭环。在图 1(b)中所有节点和边的权重都为 1。

图 2 显示了一个结构化 deep Web 数据源（假定 WDB 包含 4 条记录，每条记录包含 4 个属性值）和它所对应的属性值序列图。图 2(b)中节点 d_1 的权重 $vw=3$，节点 a_2、b_2、c_1 的权重 $vw=2$，其他所有

节点的权重 vw=1；图 2(b)中边$(c_1, d_1)\in E$和$(a_2, b_2)\in E$的权重 ew=2，其他边的权重都 ew=1。属性值序列图的节点和边的权重记录了它们在已构建的所有记录中的统计信息，这些信息将便于 deep Web 数据的增量爬取。

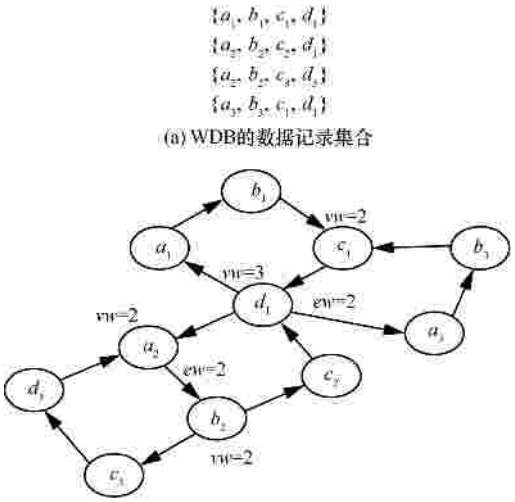


图 2　WDB 的属性值序列

对于一个给定的 deep Web 数据源，在第一次数据爬取结束后，通过上述图模型的构建方法可以得到该 WDB 的属性值序列图。随着时间的推移 deep Web 数据源会产生大量的新数据记录，需要进行增量爬取。因此，本文以属性值序列图模型为基础，研究新的方法来自动增量爬取新数据（新数据发现），以尽可能小的代价增量爬取 deep Web 数据源中新产生的数据记录。从而在相同资源约束前提下，使本地数据与远程数据保持最大化同步。

4　deep Web 新数据发现策略

4.1　deep Web 新数据发现的总体思路

事物发展的趋势可以分为 3 种：递增、递减和平稳。趋势的原因各式各样，比如我国的人均收入、银行的存款额每年随着时间而增长，而我国贫困人口的数据趋势逐年递减，某地区的平均温度以及平均降水量是基本平稳的。值得注意的是，几乎所有的事物在不同的发展阶段都要经过不同的趋势，一般来说初期具有向上增长的趋势，经过一段时间的成长达到成熟期，成熟期呈现平稳的趋势，到了末期则有向下减少的趋势。因此，可以通过分析事物过去的发展情况，来预测将来的发展趋势。假定 deep Web 数据源中包含某个关键词的数据记录数的变化趋势已符合事物发展的一般规律，基于这个假定本文提出一种 deep Web 新数据发现策略，该策略的基本思路为通过分析 deep Web 数据源在最近 L 个历史版本对应的属性值序列图，估计当前属性值序列图 $G(t_{k-1})$中哪些节点（查询关键词）为增长节点，增长节点为目前在 WDB 中与该节点相匹配的数据记录数处于递增阶段的节点，换句话说，WDB 在时刻 t_{k-1} 到时刻 t_k 期间产生的新数据记录较大可能包含的节点。然后，对增长节点的新数据发现能力进行评估，最后选择 Top-k 个具有最高新数据发现效率的增长节点作为查询节点，通过提交查询节点对应的关键词爬取新数据记录。下面介绍基于属性值序列图模型的 deep Web 新数据发现策略。假定从本地数据库获得 deep Web 数据源的 k 个历史版本，deep Web 新数据发现问题能被描述为：给定的 $WDB(t_0)$, $WDB(t_1)$, …, $WDB(t_{k-1})$，如何尽可能多地爬取在$\triangle t=t_k-t_{k-1}$ 时间区间内产生的新数据记录。新数据发现算法的具体描述如下。

算法 1　基于属性值序列图模型的 deep Web 新数据发现算法

输入： $WDB(t_{k-L})$, $WDB(t_{K-(L-1)})$, …, $WDB(t_{k-1})$ //deep Web 数据源最近 L 个历史版本

输出： NRS；// NRS 为爬取的新数据记录的集合

Algorithm $NewrecordsCrawler(WDB(t_{k-L}), WDB(t_{K-(L-1)}),\cdots,WDB(t_{k-1}))$

1) begin

2) 构建 $WDB(t_{k-L})$, $WDB(t_{K-(L-1)})$, …, $WDB(t_{k-1})$ 对应的属性值序列图 $G(t_{k-L})$，$G(t_{K-(L-1)})$，…，$G(t_{k-1})$

3) $IncrementV=IncrementVertexGeneration(G(t_{k-L}), G(t_{K-(L-1)}),\cdots,G(t_{k-1}))$ //增长节点选择

4) $PIncrementV=NewDataEstimate(IncrementV)$ //增长节点新数据发现能力估计

5) while *terminate criteria* do //停止条件

6) $q_i=QueryVertexSelection(PIncrementV)$ //查询节点选择

7) $Q(WDB,q_i)=Query(q_i,WDB)$ //在 WDB 上执行查询 q_i

8) $rs=RecordExtraction(Q(WDB,q_i))$ //抽取执行查询 q_i 获得的数据记录

9) $nrs=rs\setminus WDB(t_{k-1})$ //查询 q_i 爬取的新记录

10) $NRS=NRS\cup nrs$

11) $Cost(WDB)=Cost(WDB)+Cost(q_i, WDB)$

12) end while

13) return *NRS*

end

算法 1 **为** deep Web **新数据发现算法，算法首先构建** deep Web **数据源最近** L **个历史版本的属性值序列（**2**)行），然后根据这** L **个历史版本的信息从** $G(t_{k-1})$**中选择增长节点并估计增长节点的新数据发现能力（**3**)~**4**)行），最后根据增长节点的新数据发现效率选择查询节点，并提交查询节点对应的关键词爬取新数据记录（**3**)~**5**)行）。一次查询的新数据爬取流程为：首先采用** *QueryVertexSelection*( )**从增长节点中选择一个节点作为查询节点，提交该查询节点对应的关键词到当前的** deep Web **数据源，获得返回结果页面（**6**)~**7**)行）。然后从结果页面中抽取该次查询获得的所有数据记录，并筛选出新的数据记录加入到** *NRS* **中（**8**)~**9**)行），最后把该次数据爬取的代价** $Cost(q_t, WDB)$**计入爬取** WDB **已耗费的总代价** $Cost(WDB)$**。算法不断重复该新数据爬取过程直到满足停止条件，完成该次新数据发现。**

该算法主要由 4 **个主要部分组成：** *IncrementVertexGeneration*( )**、** *NewDataEstimate* ( )**、** *QueryVertexSelection*( )**和** *RecordExtraction*( )**。**

IncrementVertexGeneration( )**：通过分析** Deep Web **数据源的最近** L **个历史版本的属性值序列图，预测当前版本的属性值图** $G(t_{k-1})$**中节点的变化趋势并选择增长节点，增长节点为** $G(t_{k-1})$**中与** WDB **在时刻** t_{k-1} **到时刻** t_k **期间产生的新数据记录具有最大可能性相匹配的节点。**

NewDataEstimate( )**：估计增长节点可能产生新记录的数量。**

QueryVertexSelection( )**：从增长节点集合中选择最高新数据发现效率的节点。**

RecordExtraction( )**：从结果页面中抽取数据记录。**

RecordExtraction( )**已经被广泛研究，目前提出了许多解决方法，将不再讨论数据记录抽取问题。本文主要讨论** deep Web **新数据发现的** *IncrementVertexGeneration*( )**、** *NewDataEstimate* ( )**和** *QuerySelection*( ) **这** 3 **部分。**

新数据发现的停止条件如下。

1) **如果系统分配给** WDB **的新数据爬取资源耗尽，即** $Cost(WDB)=\sum_{i=1}^{n} Cost(q_i,WDB) \geq c_k$**，** c_k **为分配给** WDB **的总资源，则结束** WDB **的数据爬取。**

2) **对于数据源** WDB**，如果** *QueryVertexSelection*( )**最近连续选择的** a **个节点（查询关键词）发现新数据记录的效率低于阈值** e**。**

3) **查询节点集合为空，没有可以提交的查询节点。**

接下来将详细介绍增长节点选择、增长节点新数据发现能力估计和查询节点选择这 3 **个关键问题。**

4.2 增长节点选择

deep Web **数据源当前本地版本** $WDB(t_{k-1})$**的属性值图为** $G(t_{k-1})$**，**deep Web **新数据发现的关键问题为：在时刻** t_k**，如何从图** $G(t_{k-1})$**中选择一个节点，在** WDB **上提交该节点对应的查询关键词可以获得尽可能多的新记录。根据** 4.1 **节叙述的事物发展的趋势的一般规律，** $G(t_{k-1})$**中的所有节点根据目前所处的发展阶段可分为** 3 **种类：增长类、衰减类和平稳类。对于** $G(t_{k-1})$**中属于这** 3 **类的节点分别称为：增长节点、衰减节点和平稳节点。如果图** $G(t_{k-1})$**中的一个节点** v **的出度（入度）在** deep Web **数据源最近的** L **个历史版本中都非常稳定，根据事物发展的一般规律，那么该节点在** $WDB(t_k)$**中存在大量新记录与之相匹配的机会将很小，换句话说在时刻** t_{k-1} **到时刻** t_k **期间** WDB **不可能产生大量与该节点相匹配的数据记录，这类节点即为“平稳节点”，如果节点** v **的出度（入度）在** deep Web **数据源最近的** L **个历史版本中的度逐渐减小或节点** v **在最近的** L **个历史版本中，度的平均值与第** $L+1$ **版本相比减小，则称该类节点为“衰减节点”，如果节点** v **的出度（入度）在** deep Web **数据源最近的** L **个历史版本中的度逐渐增加，或节点** v **在最近的** L **个历史版本中度的平均值大于第** $L+1$ **版本，则称该类节点为“增长节点”，显然，在时刻** t_k**，**WDB **中与增长节点相匹配的数据记录数最有可能增加，即** $G(t_{k-1})$**中的增长节点在** WDB **中更有可能与新产生的数据记录相匹配。本文使用式（**1**）计算当前版本** $WDB(t_{k-1})$**中节点** v **从时刻** t_{k-L} **到时刻** t_{k-1} **区间的增长度**

$$Increment(v)=\sum_{i=1}^{L-1}\frac{1}{e^i}\left(\frac{D_{k-i}(v)-D_{k-(i+1)}(v)}{D_{k-i}(v)+e}\right) \quad (1)$$

其中， $D_{k-i}(v)$**为图** $G(t_{k-i})$**中节点** v **的入度（出度），即在** $WDB(t_{k-i})$**中与节点** v **对应关键词相匹配的数据记录数；** $\frac{1}{e^i}$ **表示节点** v **的增长度的权重，与当前版本越接近的版本之间的增长度越重要；** $\frac{D_{k-i}(v)-D_{k-(i+1)}(v)}{D_{k-i}(v)+e}$ **表**

示 $WDB(t_{k-i})$与 $WDB(t_{k-(i+1)})$2 个版本之间的增长度；$D_{k-i}(v)$- $D_{k(-i+1)}(v)$为在 $WDB(t_{k-i})$中与节点 v 相匹配的新记录数；对于 e，用一个例子说明，假定 $G(t_{k-i})$中 2 个节点 v_1 和 v_2，$D_{k-i}(v_1)=6$，$D_{k(-i+1)}(v_1)=2$，$D_{k-i}(v_2)=12$，$D_{k(-i+1)}(v_2)=4$，显然应该优先选择 v_2，因为在 deep Web 数据源上提交 v_2 对应的关键词能获得比提交 v_1 对应的关键词更多的新记录，e 使 $Increment(v_2)>Increment(v_1)$，同时避免当 v?$G(t_{k-i})$，$D_{k-i}(v)=0$ 时，公式除以 0 的情况。节点 v 的增长度如果大于增长阈值a ，则认为节点 v 为增长节点。增长节点产生算法的具体描述如下。

算法 2　增长节点产生算法

输入： $G(t_{k-L})$, $G(t_{k-(L-1)})$, …, $G(t_{k-1})$, // WDB 的最近 L 个历史版本的属性值图

输出： *IncrementV*　// *IncrementV* 为增长节点集合

Algorithm *IncrementVertexGeneration* ($G(t_{k-L})$, $G(t_{k-(L-1)})$,…, $G(t_{k-1})$)

1) begin

2) for each vertex v in $G(t_{k-1})$ do

3)　Compute *Increment* (v)；

4)　if *Increment* (v) ≥a then　//a 为增长节点的阈值，如果 *Increment* (v) ≥a ，则 v 为增长节点

5)　*IncrementV* = *IncrementV*∪v　//添加节点 v 到增长节点集合

6)　end if

7) end for

8) return *IncrementV*

end

在算法 2 中，首先输入 deep Web 数据源最近 L 个历史版本的属性值序列图 $G(t_{k-L})$,$G(t_{k-(L-1)})$,…, $G(t_{k-1})$，计算 deep Web 数据源当前版本属性值序列图 $G(t_{k-1})$中每个节点的增长度，如果节点的增长度大于等于增长阈值a ，则将节点加入到增长节点集合 *IncrementV* 中。通过上述算法得到当前版本属性值序列图 $G(t_{k-1})$中与产生新记录相匹配可能性较大的所有增长节点。通过增长度计算 $G(t_{k-1})$中所有节点定性分为增长节点、衰减节点和稳定节点 3 类，在下一节中将预测增长节点的新数据发现能力，即在 $WDB(t_k)$中与增长节点相匹配的新数据记录数量。

4.3　增长节点的新数据发现能力估计

目前，人们对事物发展趋势（如价格、宏观经济等）的预测主要采用的预测模型有灰色系统预测模型、时间序列预测模型、回归预测模型、神经网络模型等[14]。由于在实际应用中常常存在序列相关性、异方差性、非线性等问题，不可避免地存在信息丢失。神经网络预测模型由大量的处理单元以适当的方式互联构成，能模拟人的大脑及其活动，具有非线性和自适应性的特点。在实际应用中神经网络预测模型与其他预测模型相比通常具有较好的拟合效果和精度。同时，神经网络预测模型具有如下优点：1) 神经网络预测模型预测精度较高，并可以实现并行计算；2) 神经网络预测模型相当于一个黑盒，不要知道输入和输出变量之间的映射关系，只需对给定的训练数据进行训练，来自动建立输入和输出变量之间的映射关系，直到实际输出值与期望值的误差满足所需的精度要求；3) 神经网络预测模型在建模的过程中只需要选取适当的输入变量和输出变量以及相应的拓扑结构就能建立合适的模型，能减少人工干预，降低人为假设的可能。

由于 deep Web 数据变化的复杂性，本文采用 BP 神经网络模型来预测增长节点的新数据发现能力。同时，针对那些结果页面包含与查询关键词相匹配的数据记录数的 deep Web 数据源，采用一种更简洁的方法计算增长节点的新数据发现能力，即增长节点预提交方法。接下来将分别介绍这 2 种增长节点新数据发现能力计算方法。

4.3.1　基于 BP 神经网络的增长节点新数据发现能力预测

对于 *IncrementV* 中一个节点 v，预测其在 $WDB(t_k)$与之相关联的数据记录数，即在 $G(t_k)$中的度。假定节点 v 在最近 L 个历史版本的属性值图中度为：{ $D_{k-L}(v)$, $D_{k-(L-1)}(v)$,…,$D_{k-1}(v)$}。使用 BP 神经网络预测 $D_k(v)$的步骤如下。

1) 利用历史数据创建训练样本

用{ $D_{k-L}(v)$, $D_{k-(L-1)}(v)$,…,$D_m(v)$}作为建立 BP 神经网络预测模型的训练样本，其中，将 $D_{k-L}(v)$作为起始，每连续 h 个度作为一个样本的输入向量，接下来的 $g(g\geq 1)$个度作为相应的期望输出向量，这样的输入向量和期望输出向量组成一个样本，然后起始位置向后退 $h(h\geq 1)$个，即从 $D_{k-L+h}(v)$作为起始，再用上述方法建立第 2 个样本，依次类推，最终创建出所有的训练样本。

2) BP 神经网络结构确定

本文预测采用 3 层网络结构：分别为输入层、隐含层和输出层。

输入层：输入层有 h 个节点，对应训练样本的输入向量。

隐含层：采用经验式 $m=\mathrm{lb}h$（m 为隐含节点数，h 为输入层节点数）确定隐含层节点数。

输出层：输出层有 g 个节点，对应训练样本的期望输出向量。本文设置 $g=1$，仅仅预测 $D_{k-L+h+1}(v)$ 的值，如果需要一次预测 $D_{k-L+h+1}(v)$和 $D_{k-L+h+2}(v)$ 的值，则可以设置 $g=2$。

3) 训练相关参数选取与新数据发现能力预测

选取对数 S 型函数作为激励函数，选取梯度下降 BP 训练函数，最大迭代次数 100，精度为 0.000 1。由于 BP 神经网络选取的 S 型函数作为激励函数，而 S 型激励函数的值域为[0,1]，因此需要采用归一法将样本的输入和输出向量规范到[0,1]区间。然后利用处理好的样本对神经网络进行训练，得到训练好的神经网络，再利用训练好的神经网络进行预测，得出 $D_{k-L+h+1}(v)$的预测值。在时刻 t_k，对于 IncrementV 中的每一个增长节点 v，可以通过 BP 神经网络得到 $D_k(v)$的预测值，因此，增长节点 v 的新数据发现能力为：$D_k(v)-D_{k-1}(v)$。

使用基于 BP 神经网络的增长节点新数据发现能力预测方法，需要对每一个增长节点训练神经网络，当增长节点数量较大时，虽然 BP 神经网络训练可以离线进行，但仍然将十分耗时。因此，针对那些结果页面包含与查询关键词相匹配的数据记录数的 deep Web 数据源，本文采用一种更简洁的方法计算增长节点的新数据发现能力，即增长节点预提交方法。

4.3.2 增长节点预提交的方法

对于一个 deep Web 数据源，当查询关键词提交到 WDB，WDB 将返回与该查询关键词相匹配的数据记录组成的结果页面，据观察，大多数结果页除了包含与查询关键词相关的数据记录外，还包含 WDB 中与该查询关键词相匹配的数据记录总数。图 3 为在互动出版网（www.china-pub.com）的查询接口提交查询关键词“java”后，互动出版网返回的结果页面（互动出版网是一个图书类电子商务网站）。如图 3 所示，在结果页面中除了显示部分数据记录外，还包含在互动出版网中与查询关键词“java”匹配的记录总数（见粗线方框“使用 java 搜索，共有 3 413 种商品”）。通过统计分析发现，现实世界中大多数的 deep Web 数据源的结果页面中包含与提交到它的查询关键词相匹配的数据记录总数，记为 *Query-R-Num*。

为了进行高效的新数据发现，需要获得在时刻 t_k，$WDB(t_k)$中与增长节点相匹配的数据记录数，即在 $G(t_k)$中增长节点的度 $D_k(v)$。因此，本文利用 deep Web 数据源的结果页面中包含与提交到它的查询关键词相匹配的数据记录总数的特点，在时刻 t_k，对于 IncrementV 中的每一个增长节点 v，进行一次查询提交，获得该节点 v 查询提交的一个结果页面，然后从结果页面中抽取 $WDB(t_k)$中与增长节点 v 相匹配的数据记录数 *Query-R-Num*，*Query-R-Num* 等价于 $G(t_k)$中增长节点 v 的度 $D_k(v)$。因此，在时刻 t_k，增长节点 v 的新数据发现能力为：*Query-R-Num*-$D_{k-1}(v)$。

图 3 WDB 返回的结果页面

与基于 BP 神经网络的增长节点新数据发现能力预测方法相比较，增长节点预提交方法不需要通过历史数据训练预测模型，只需要对所有增长节点进行一次查询提交，爬取一个结果页面即可获得时刻 t_k 在 WDB 中与增长节点相匹配数据记录的准确数量。因此，增长节点预提交方法与基于 BP 神经网络的增长节点新数据发现能力预测方法相比较，增长节点预提交方法使用更加简单方便，同时代价较低。更重要的是基于预测的方法存在一定的误差，而增长节点预提交方法则更为准确。因此，对那些结果页面包含与查询关键词相匹配的数据记录数的 deep Web 数据源，本文使用增长节点预提交方法获得增长节点的新数据发现能力，否则，使用基于 BP 神经网络的方法预测增长节点的新数据发现能力。

4.4 查询节点选择

通过预测或预提交得到 *IncrementV* 中每个节点在 $G(t_k)$中可能的入度（出度）后，可以通过 $D_k(v)-D_{k-1}(v)$得到每个增长节点的新数据发现能力，即时刻 t_k，在 $WDB(t_k)$上提交节点 v 能获得新数

据记录数量。根据 deep Web 新数据发现策略的爬取代价 NR_Cost，查询节点（关键词）选择的目标为：每次在 $IncrementV$ 中选择具有最高新数据发现效率的节点，节点的新数据发现效率为爬取最少的总数据记录，获得最多的新数据记录，即对于 $G(t_{k-i})$中 2 个节点 v_1 和 v_2，如果 v_1 和 v_2 爬取相同数量的新数据记录，v_1 爬取的总数据记录数小于 v_2，则认为 v_1 的新数据发现效率高于 v_2。显然，应该优先选择 v_1。节点 v 的新数据发现效率定义为

$$effective(v)=\frac{D_k(v)-D_{k-1}(v)}{D_k(v)} \tag{2}$$

其中，$D_{k-1}(v)$为在时刻 t_{k-1} 图 $G(t_{k-1})$中增长节点 v 的入度（出度），$D_k(v)$为在时刻 t_k 在 deep Web 数据源上提交与增长节点 v 相对应的查询关键词所返回的总记录数。$D_k(v)$可由 4.3 节得到。$D_k(v)$- $D_{k-1}(v)$为时刻 t_k 在 $WDB(t_k)$上提交节点 v 能获得新数据记录数量。

在理论上，只需按照 $IncrementV$ 中每个增长节点的新数据发现效率从大到小选择节点，在 deep Web 数据源上提交与该节点对应的查询关键词，爬取新记录。不断重复该过程，直到满足停止条件。然而，上述新数据爬取的查询选择方法在进行查询选择时没有考虑选择提交的下一个查询节点（关键词）与已经选择提交的查询节点之间的依赖关系。根据文献[15]，在实际应用中，属性值之间的依赖关系十分普遍，例如：许多作者通常一起发表论文，当一个作者名字被提交查询后，其他的作者名字即便是具有较高新数据爬取效率，也不是一个好的选择。提交它们作为查询不但不能爬取更多的新数据，反而需要系统处理大量重复数据。因此，在进行查询节点选择时，必须要考虑查询节点与其他已提交查询节点之间的依赖关系，降低查询依赖带来的负面影响。

本文使用 Mutual Information 计算 2 个查询节点（关键词）之间的依赖性，2 个查询节点 v_1 和 v_2 的依赖度可由它们共同出现在 $G(t_{k-1})$中一个有序圈（一条数据记录）中的概率决定。对于 $IncrementV_{to\text{-}query}$ 中的每个节点 v_i 与所有已提交的节点之间的依赖关系 $r(v_i,IncrementV_{queried[1,2,\cdots,m]})$可由式(3)计算得到

$$r(v_i,IncrementV_{queried[1,2,\cdots,m]})=\mathrm{Max}\ln\left(\frac{P(v_i,v_j\mid G(t_{k-1}))}{P(v_i\mid G(t_{k-1}))P(v_j\mid G(t_{k-1}))}\right) \tag{3}$$

其中，$IncrementV_{to\text{-}query}$ 为 $IncrementV$ 中还没有被选择提交的增长节点集合，$IncrementV_{queried[1,2,\cdots,m]}$为所有已提交的 m 个查询节点集合，$v_j\in r(v_i,IncrementV_{queried[1,2,\cdots,m]})$，$P(v_i|G(t_{k-1}))$和 $P(v_j|G(t_{k-1}))$分别为 v_i 和 v_j 在 $G(t_{k-1})$中出现的概率，$P(v_i,v_j,|G(t_{k-1}))$为 v_i 和 v_j 在 $G(t_{k-1})$中同时出现在一个有序圈（一条数据记录）中的概率。$IncrementV_{to\text{-}query}$ 中所有节点的依赖度 r 都基于 $G(t_{k-1})$计算得到。

本文在进行查询选择时需要惩罚那些具有较强依赖关系的节点，因此，对于 $IncrementV_{to\text{-}query}$ 中的每个节点赋予它一个优先级分数 $s(v, IncrementV_{queried[1,2,\cdots,m]})$，$IncrementV_{to\text{-}query}$ 中的所有节点按照 $s(v,IncrementV_{queried[1,2,\cdots,m]})$降序排列，deep Web 数据增量爬虫每次从 $IncrementV_{to\text{-}query}$ 中选择 $s(v,IncrementV_{queried[1,2,\cdots,m]})$最大的节点作为查询节点。$s(v, IncrementV_{queried[1,2,\cdots,m]})$的计算式为

$$\begin{aligned}&s(v,IncrementV_{queried[1,2,\cdots,m]})\\&=effective(v)(1-r(v_i,IncrementV_{queried[1,2,\cdots,m]}))\end{aligned} \tag{4}$$

该方法从 $IncrementV_{to\text{-}query}$ 中每次选择具有最大 s 值的节点作为查询节点提交后，需要重新计算 $IncrementV_{to\text{-}query}$ 中所有节点的 s 值，以便选择下一个节点，这样计算量将非常大，因此，在具体应用中设置一个重计算节点 s 值的步长 u，即提交 u 次查询节点后，再重新计算 $IncrementV_{to\text{-}query}$ 中所有节点的 s 值。该方法在适当牺牲系统效率的情况下，大幅降低系统的计算代价。在新数据爬取过程中步长 u 的值可以动态调整，以提高新数据爬取的效率。

5　实验

为了对本文提出的基于属性值序列图模型的 deep Web 新数据发现策略的性能进行评估，主要从以下 3 个方面进行实验测试：1）算法关键参数（a、e 和 u）分析；2）新数据发现策略的性能分析；3）AVM 与 HVM、IHM 算法的同步率比较。本节首先介绍实验所使用的数据集，然后给出各项实验以及实验结果分析。

5.1　数据准备

为了评估本文提出方法的性能，本文采用专利领域和鞋类电子商务领域的 5 个真实 deep Web 数据源的历史版本数据，这些历史版本数据来至搜鞋客

（www.soxieke.com）和明智慧——生物医药科技信息服务平台（www. Mingzh.com）2 个数据集成系统。鞋类电子商务领域选择的 Deep Web 数据源为：好乐买（www.okbuy.com）、乐淘（www.letao.com）和搜鞋客（系统已集成的鞋类数据约 95 万余条数据）；专利领域选择的 deep Web 数据源为：中国知识产权局专利数据源的生物医药类别（www.sipo.gov.cn/zljs）和明智慧（系统已集成的七国两组织生物医药专利数据约 369 万余条数据）。这些数据取于系统 2013 年 5 月 1 日往后的系统集成数据，对于专利领域，每间隔 2 个星期选择一个时间点，获得连续 20 个历史版本数据；由于电子商务领域的数据变化快于专利领域，因此，对于鞋类电子商务领域，每间隔一个星期选择一个时间点，获得连续 20 个历史版本数据。下面将基于以上数据集进行实验，以验证本文提出方法的有效性。

5.2 新数据发现的效率评估指标

deep Web 新数据发现的主要目标为在尽可能小的代价下获得尽可能多的新数据，为了评估新数据发现的效率，2 个因素必须被考虑：覆盖率和爬取代价。

新数据发现策略的覆盖率为

$$NR\_Coverage=\frac{N_{Crawled\_N\_R}}{N_{New\_R}} \tag{5}$$

其中，$N_{Crawled\_N\_R}$ 为增量爬虫发现的新数据记录数，N_{New_R} 为时间 t 到时间 t+1 期间实际产生的新数据记录数。

新数据发现策略的爬取代价为

$$NR\_Cost=\frac{N_{Crawled\_N\_R}}{N_{Total\_R\_R}} \tag{6}$$

其中，$N_{Crawled\_N\_R}$ 为增量爬虫发现的新数据记录数，$N_{Total\_R\_R}$ 为爬虫发现 $N_{Crawled\_N\_R}$ 个新数据记录总共爬取的数据记录数。

显然，如果给定一个 *NR_Cost* 值，*NR_Coverage* 越大，则爬虫新数据发现的效率越高。

5.3 实验结果

5.3.1 算法关键参数（a 、e 和 u）分析

新数据发现算法中 a 、e 和 u 是影响算法效率的关键参数，在验证新数据发现策略的性能前，必需考虑如何能为算法选择合适的 a 、e 和 u 值。下面将主要分析 a 和 e 的选取，实验在上述 5 个数据源的最近一个历史版本上进行，在不限制提交次数的前提下，分析 a 和 e 的不同取值对覆盖率的影响，然后在考虑平衡新数据发现代价和效率的基础上，探讨 a 和 e 值的选取。

实验首先使用不同 a 值评估其对覆盖率的影响。从图 4 中可以看出对于每个固定的 e 值，随着 a 值的增加覆盖率也逐步增加，在开始阶段覆盖率随着 a 值的增加覆盖率增加较为明显，后期随着 a 值的继续增加覆盖率增长并不明显。就其原因为当 a 值较小时会导致新数据发现算法过早的终止，会忽略掉一部分新数据；当 a 值到达一定值后提交更多的查询仅能获得极少的新数据，所以这以后覆盖率趋于平衡，并不随 a 值的增加而明显增加。实验结果表明对于一个数据源 a 值有一个临界值，在这个临界值之前 a 值的增加对覆盖率的影响非常明显，而之后则仅有极小的影响。根据算法新数据发现代价会随着 a 值的增加，尤其是当 a 值大于临界之后 a 值的增加会使代价显著增加。综合考虑新数据发现代价和效率 2 个因素，临界值则应为该数据源新数据发现 a 参数最理想的取值。

然后实验分析不同 e 值对覆盖率的影响，如图 4 所示，在相同 a 值的情况下，e 值越小覆盖率越高，其变化趋势与 a 值对覆盖率的影响相似，当 e 值到达一个临界值后继续减小 e 值，对覆盖率的影响变的非常小，因此，对于一个数据源 e 值也有一个临界值。与 a 值对代价影响一致，随着 e 值的增加，尤其是 e 值大于临界之后，e 值的增加会使代价显著增加。在 5 个不同的数据源上的实验结果得到了基本一致的结论，因此，在后续的新数据发现算法效率比较时，选择 a 和 e 的临界值作为最理想取值。

对于 u 值本文通过类似的实验，得出的结果与 a 和 e 相似，并不是 u 值越大新数据覆盖率就越高，对于一个数据源 u 值同样也有一个临界值，由于实验过程和结论类似，本文在这里不再赘述。

从图 4 可以看出 a =8 和 a =10 时的覆盖率是略大于 a =6 的，同样 e=1%和 e=3%时的覆盖率是略大于 e=5%的，这些实验是在不限制次数（代价）的情况下得出的。当在资源约束条件下，必须平衡考虑新数据发现的代价和效率，因此临界值则应为该数据源新数据发现 a 参数的最理想的取值。从实验可以看出不同的数据源临界值可能有细微的差别，例如对于 e，在 Sipo、Okbuy 和 Letao 上 e=7%时的实验结果与 e=1%、3%、5%都比较接

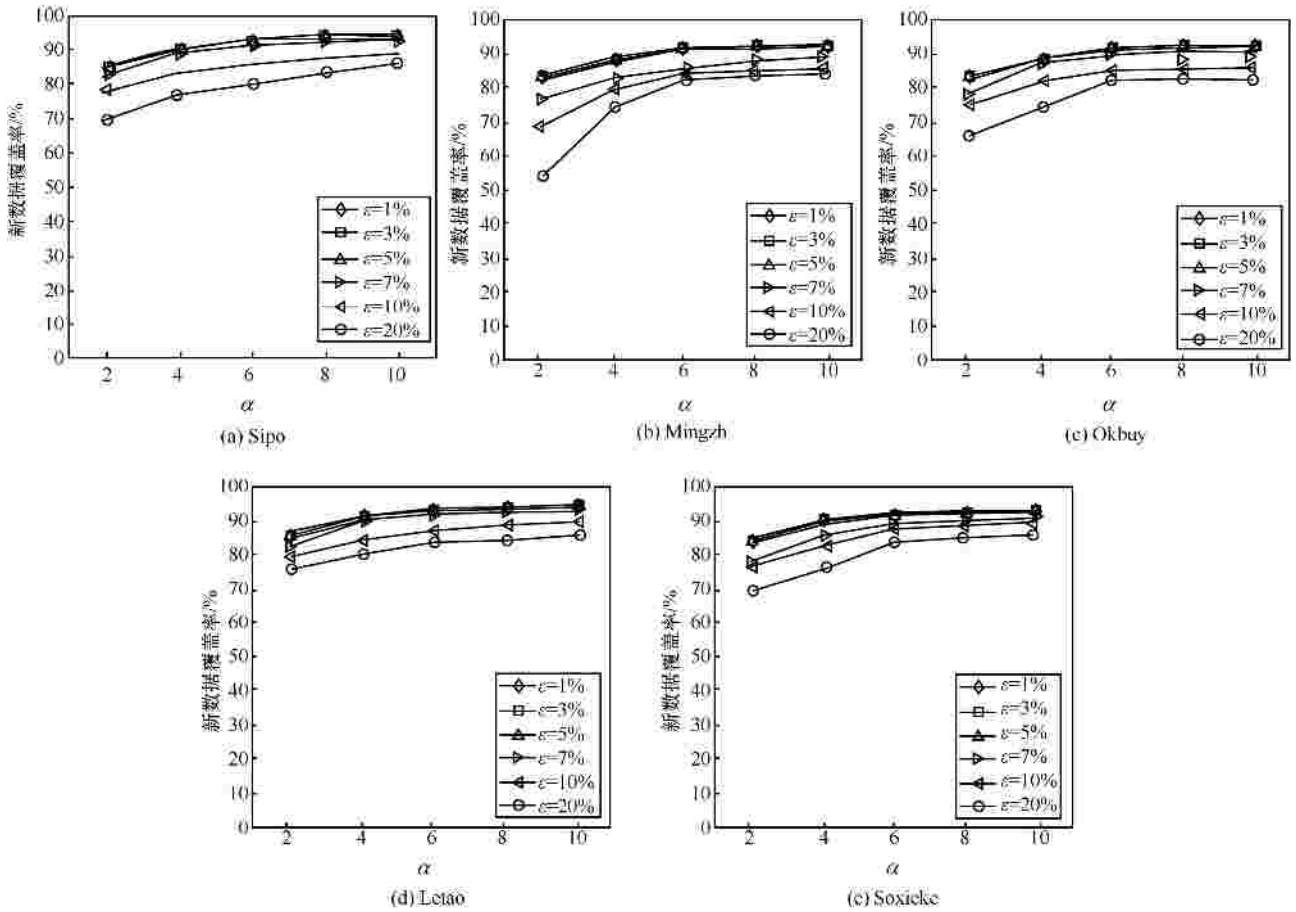


图 4　*a* 和 *e* 参数值对新数据发现算法性能的影响

近，但在其他 2 个数据源上却存在一定差距。在参数选择时为了更具代表性，本文 *e* 值设置为 5%，其他参数选择时基于同样的原则。因此，本文将新数据发现算法最理想参数设置 *a* 值为 6，*e* 值为 5%，重计算节点 *s* 值的步长 *u* 值为 12。由于本文在 5 个不同数据源上的实验结果表明，*a* 、*e* 和 *u* 的理想参数具有较强的相似性，因此，本文的参数值设置具有一定的通用性，可作为其他数据源的参考。

5.3.2　新数据发现策略的性能分析

1) 新数据发现算法的有效性分析

首先比较本文提出的基于属性值序列图模型的 deep Web 新数据发现策略（AVM）与现有 2 种 deep Web 数据增量爬取方法 HVM(model history version)[8]，IHM(incremental harvest model)[10]的性能并分析其有效性。上述 3 种方法通过各自策略选择下一个最可能发现新数据的关键词，以发现新数据，反复执行该过程，直到满足停止条件。该实验的停止条件为实验 deep Web 数据源的新数据覆盖率达到 95%，为了便于比较当某一种爬取方法的新数据覆盖率达到停止条件时，同时已结束其他 2 种增量爬取方法的爬取。为了避免 AVM 满足停止条件而过早停止，保证实验在相同的停止条件下进行比较，在实验中 *a* 值设置为 10，阈值 *e* 设置为 1%，重计算节点 *s* 值的步长 *u* 值设置为 12。

图 5 给出了 3 种方法在 5 个实验数据源最近一个历史版本上的实验结果。从图 5 可以得出基于属性值序列图模型的 deep Web 新数据发现策略在 5 个数据源上都最先获得 95%左右的新数据。在 Sipo 数据源上当其 Sipo 的新覆盖率达到 95%时，HVM 达到 83%，IHM 为 85.5%，在其他数据源上，基于属性值序列图模型的 deep Web 新数据发现策略已取得了与 IHM 数据源类似的结果。与 HVM 和 IHM 相比 AVM 的具有更好有效性和效率。

从 Sipo 数据源的实验中可以看出当查询提交 900 次左右时，Sipo 的覆盖率已达到 90%以上，在这个查询次数下取得这样的新数据覆盖率已经是非常理想的结果，而其他 2 种算法此时取得覆盖率

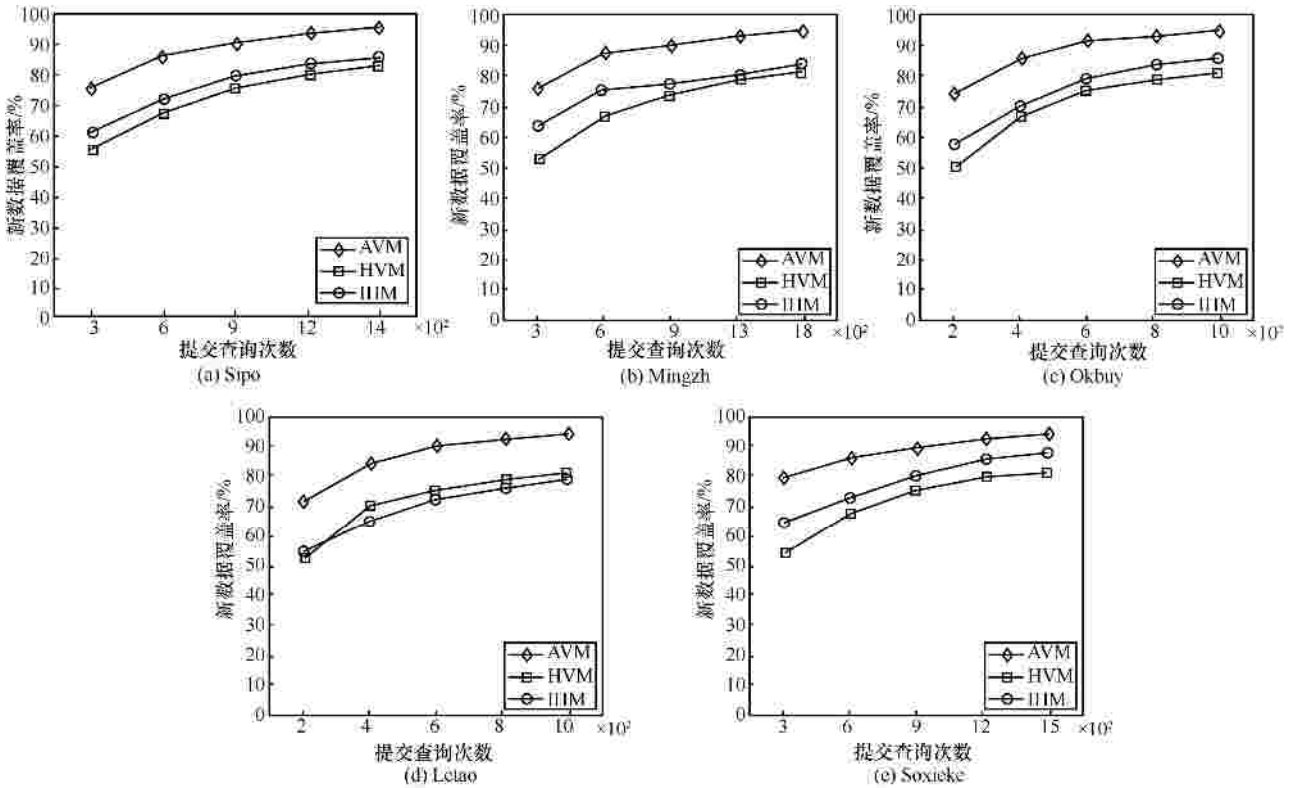


图 5 AVM、HVM 和 IHM 的增量爬虫的有效性比较

与 AVM 相差甚远，甚至提交成倍的查询次数也很难取得相同的覆盖率。在其他数据源上，AVM 算法都在较少的提交次数下获得了 90%以上的覆盖率。因此说明本文提出的 AVM 算法能在较少查询提交次数下获得较高的覆盖率，具有较强的有效性。

2) 在多个历史版本上验证新数据发现策略的效率

在以下 2 个实验中根据 5.3.1 节的实验结论，设置新数据发现算法的 a 值为 6，e 值为 5%，重计算节点 s 值的步长 u 值为 12。

在上述 5 个数据源上，利用新数据发现效率评估指标来评价本文提出的新数据发现策略的性能。该实验使用 $WDB(t_{k-6})$，$WDB(t_{k-5})$，$WDB(t_{k-4})$，$WDB(t_{k-3})$，$WDB(t_{k-2})$和 $WDB(t_{k-1})$最近连续 6 个历史版本数据作为统计信息来爬取 $WDB(t_k)$中新产生的数据记录。在时刻 t_7、t_{10} 和 t_{15} 这 3 个时间点上进行实验得到的结果如表 1 所示。本文提出的新数据发现策略在新数据发现策略的覆盖率（NR_Coverage）和新数据发现策略的爬取代价（NR_Cost）2 个方面都取得了非常高的效率。在这 5 个数据源上的所有实验表明，本文提出新数据发现策略的覆盖率 NR_Coverage 都超过了 87%，最高为 92.5%。新数据发现策略的爬取代价 NR_Cost 最高的仅为 39.7%。

表 1 新数据发现策略的性能

观察点	数据源	*NR_Coverage*	*NR_Cost*
时间点 t_7	Sipo	92.5%	35.2%
	Mingzh	91.5%	38.5%
	Okbuy	87.7%	39.3%
	Soxieke	89.5%	36.6%
	Letao	87.1 %	38.6%
时间点 t_{10}	Sipo	89.6%	32.5%
	Mingzh	90.2%	39.7%
	Okbuy	89.3%	35.8%
	Soxieke	91.%	38.7%
	Letao	88.9%	35.9%
时间点 t_{15}	Sipo	90.6%	36.1%
	Mingzh	90.8%	39.3%
	Okbuy	88.5%	37.4%
	Soxieke	89.3%	38.9%
	Letao	87.5%	36.9%

5.3.3 AVM 与 HVM、IHM 算法的同步率比较

在相同增量爬取资源约束下，分别在 5 个数据源上，比较本文提出 AVM 与现有 2 种 deep Web 数

据增量爬取方法 HVM 和 IHM 的数据同步率。

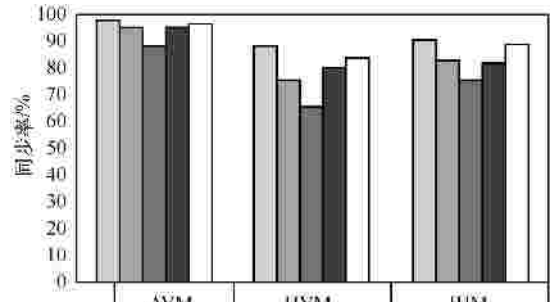
定义 2　本地数据与远程数据的同步率。假定在时刻 t_k，经过增量爬取后得到的本地数据为 $WDB(t_k)$，而此时远程数据源中的实际数据为 $WDB(t_k)$。则本地数据与远程数据的同步率 $Synchronization(t_k)$可定义为

$$Synchronization(t_k)=\frac{|WDB(t_k)\cap WDB'(t_k)|}{|WDB(t_k)|} \quad (7)$$

其中，$|WDB(t_k)|$为在时刻 t_k，WDB 的数据记录总数；$|WDB(t_k)\text{n } WDB'(t_k)|$为在时刻 t_k，本地数据与远程数据相同数据记录总数。

在实验中，对每个数据源设置增量更新资源为 20 万条数据记录，即在时刻 t_k，增量爬虫从 $WDB(t_k)$中爬取 20 万条数据记录更新 $WDB(t_{k-1})$，增量爬取后得到本地数据为 $WDB'(t_k)$。然后比较本地数据 $WDB'(t_k)$与远程数据 $WDB(t_k)$的同步率。在实验中，从 $WDB(t_k)$中爬取 20 万条数据记录可通过分析 WDB 的第 k 个历史版本获得，并不需要真实的爬取过程。在 t_9 时刻获得的实验结果如图 6 所示。

从图 6 中可以看出，本文提出的 AVM 方法，在 5 个数据源上都取得了非常高的数据同步率，并都优于 HVM 和 IHM。在 Sipo 数据源，AVM、HVM 和 IHM 3 种增量爬取方法的同步率比较接近，AVM 略优于 IHM 和 HVM。但在 Letao 数据源的实验结果与在 Sipo 数据源上存在较大的差异，在 Letao 数据源，AVM 和 IHM 明显优于 HVM，在相同更新资源的约束下，HVM 取得的数据同步率最低，仅为 75.2%，AVM 和 IHM 分别为 94.4%和 82.7%。对于 HVM 方法，本文在这些数据源分别选择不同的查询接口进行实验得到差异较大的实验结果，究其原因 HVM 方法与查询接口能力密切相关，不能独立表示 deep Web 数据源的内容，具有一定的不确定性。在 Mingzh 数据源，AVM 明显优于 HVM 和 IHM，但是与其他 4 个数据源比较，这 3 种方法的同步率都较低，究其原因为 Mingzh 的数据量远远大于其他数据源，在相同的更新资源下，取得的同步率应该低于其他数据源。从在 Mingzh 数据源上的实验结果中可以看出，当数据量较大时，本文方法的效率优势更加明显，能较好地适应大数据量情况下的增量更新。在其他时间点上的实验与在 t_9 时刻的实验结果类似，本文将不再这里赘述。综上所述，本文提出的新数据发现方法都取得了非常高的同步率。



	AVM	HVM	IHM
Sipo	98.3%	88.2%	90.6%
Letao	95.1%	75.2%	82.7%
Mingzh	98.3%	65.9%	75.4%
Soxicke	95.1%	79.6%	82.0%
Okbuy	96.3%	83.0%	88.3%

图 6　AVM、HVM 和 IHM 的增量爬虫的数据同步率性能比较

6　结束语

由于 deep Web 是自治的、独立更新的，其数据经常处于频繁更新的状态（不断有新数据记录产生），同时也有一些数据记录消失或改变，而用户总是希望能够得到当前 deep Web 数据源中最新的内容。因此需要定期地爬取远程数据源的数据，更新本地数据拷贝，以保持本地数据与远程数据同步。deep Web 数据增量爬取需要处理新产生的、消失的和改变的 3 类数据记录。本文以属性值序列图模型为基础，针对新产生记录的爬取，提出了一种新数据发现策略，实验表明在相同资源约束前提下，可有效提高本地数据的时新性和新数据的发现效率，使本地数据和远程数据保持最大化同步。同时本文提出的属性值序列图模型仅与数据相关，可适用于仅仅包含简单查询接口的 deep Web 数据源，拓展了新数据发现的应用范围。

参考文献：

[1] MADHAVAN J, COHEN S, DONG X L, et al . Web-scale data integration: you can afford to pay as you go[C]//The 3rd International Conference Innovative Data Systems Research. Asilomar, CA, c2007: 342-350.

[2] MADHAVAN J, KO D, KOT L, et al. Google's deep-Web crawl[C]//The 34th International Conference on Very Large Data Bases. Auckland, New Zealand, Springer, c2008: 1241-1252.

[3] PAVAI G, GEETHA T V. A unified architecture for surfacing the con-tents of deep Web databases[C]//International Conference on Ad-

vances in Communication. Network, and Computing, Chennai, India, c2013.

[4] ANDREA C, DAVIDE M, RICCARDO T. Keyword search in the deep Web[C]//AMW2015 Alberto Mendelzon International Workshop on Foundations of Data Management .Lima Peru, c2015: 205-208.

[5] EDWARDS J, MCCURLEY K, TOMLIN J. An adaptive model for optimizing performance of an incremental Web crawler[C]//The 10th Conference on World Wide Web. Hong Kong, China, c2001: 106-113.

[6] SINGHAL N, DIXIT A, SHARMA A K. Design of a priority based frequency regulated incremental crawler[J]. International Journal of Computer Applications, 2010, 1 (1): 42-47.

[7] JAGANATHAN P, KARTHIKEYAN T. Highly efficient architecture for scalable focused crawling using incremental parallel Web crawler[J]. Journal of Computer Science, 2015, 11 (1): 120-126.

[8] LIU W, XIAO J G, YANG J W. Incremental structured Web database crawling via history versions[C]//The 11th International Conference on Web Information Systems Engineering. c2010: 524-533.

[9] LIU W, XIAO J G, YANG J W. A sample-guided approach to incremental structured Web database crawling[C]//International Conference on Information and Automation ,Harbin, c2010: 890-895.

[10] HUANG Q Y, LI Q Z, LI H, et al. An approach to incremental deep Web crawling based on incremental harvest model[J]. Procedia Engineering, 2012, (29): 1081-1087.

[11] ZHANG Z X, DONG G Q, PENG Z H, et al. A framework for incremental deep Web crawler based on URL classification[J]. Lecture Notes in Computer Science, 2011, 6988: 302-310.

[12] 张志潇.面向领域的 Deep Web 的增量爬取[D].济南 :山东大学,2012. ZHANG Z X. Domain-specific deep Web incremental crawler[D]. Ji-Nan: Shandong University,2012.

[13] YOGESH K, MANOJ K R, JITENDRA D. Novel approach for data source integration system update strategy in hidden Web[J]. International Journal of Engineering Universe for Scientific Research and Management.2015,2(7):1-5.

[14] 徐国强. 统计预测和决策[M]. 上海；上海财经大学出版社.2008. XU G Q. Statistical forecasting and decision-making [M]. Shanghai: Shanghai University of Finance and Economics press, 2003.

[15] WU P, WEN J R, LIU H, et al. Query selection techniques for efficient crawling of structured Web sources[C]//The 22th International Conference on Data Engineering. Atlanta, GA, USA, c2006: 47-56.

作者简介：

鲜学丰（1980-），男，四川南充人，博士，苏州市职业大学副教授，主要研究方向为 Web 数据管理、数据挖掘和智能信息处理。

崔志明（1961-），男，上海人，苏州大学教授、博士生导师，主要研究方向为智能信息处理和计算机网络。

赵朋朋（1980-），男，江苏南通人，博士，苏州大学副教授，主要研究方向为 deep Web 和 Web 数据挖掘。

方立刚（1980-），男，安徽黄山人，博士，苏州市职业大学副教授，主要研究方向为计算机网络和 Web GIS。

杨元峰（1973-），男，江苏盐城人，苏州市职业大学副教授，主要研究方向为智能信息处理。

顾才东（1963-），男，宁夏吴忠人，苏州市职业大学教授，主要研究方向为智能信息处理和物联网。