软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(5):1271−1295 [doi: 10.13328/j.cnki.jos.005105]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于分层抽样的重叠深网数据源选择*

江俊彦[1,2], 彭智勇[2], 吴小莹[2], 彭承晨[1], 王 敏[1]

[1](武汉大学 计算机学院,湖北 武汉 430072)

[2](软件工程国家重点实验室(武汉大学),湖北 武汉 430072)

通讯作者: 吴小莹, E-mail: xiaoying.wu@whu.edu.cn

摘 要: 深网查询在Web上众多的应用,需要查询大量的数据源才能获得足够的数据,如多媒体数据搜索、团购网站信息聚合等.应用的成功,取决于查询多数据源的效率和效果.当前研究侧重查询与数据源的相关性而忽略数据源之间的重叠关系,使得不同数据源上相同结果的数据被重复查询,增加了查询开销及数据源的工作负载.为了提高深网查询的效率,提出一种元组水平的分层抽样方法来估计和利用查询在数据源上的统计数据,选择高相关、低重叠的数据源.该方法分为两个阶段:离线阶段,基于元组水平对数据源进行分层抽样,获得样本数据;在线阶段,基于样本数据迭代地估计查询在数据源上的覆盖率和重叠率,并采用一种启发式策略以高效地发现低重叠的数据源.实验结果表明,该方法能够显著提高重叠数据源选择的精度和效率.

关键词: 数据源选择;分层抽样;数据源重叠率估计;回归

中图法分类号: TP311

中文引用格式: 江俊彦,彭智勇,吴小莹,彭承晨,王敏.基于分层抽样的重叠深网数据源选择.软件学报,2017,28(5):1271−1295. http://www.jos.org.cn/1000-9825/5105.htm

英文引用格式: Jiang JY, Peng ZY, Wu XY, Peng CC, Wang M. Overlapping deep Web data source selection based on stratified sampling. Ruan Jian Xue Bao/Journal of Software, 2017,28(5):1271−1295 (in Chinese). http://www.jos.org.cn/1000-9825/5105.htm

Overlapping Deep Web Data Source Selection Based on Stratified Sampling

JIANG Jun-Yan[1,2], PENG Zhi-Yong[2], WU Xiao-Ying[2], PENG Cheng-Chen[1], WANG Min[1]

[1](Computer School, Wuhan University, Wuhan 430072, China)

[2](State Key Laboratory of Software Engineering (Wuhan University), Wuhan 430072, China)

Abstract: Many Web applications, such as multimedia data integration and online business data aggregation, require deep Web querying to integrate information from many data sources on the Web. The success of such applications is largely determined by the efficiency and effectiveness of querying methods over relevant sources. Existing studies on multiple data source integration have focused on ranking the relevance of queries w.r.t data sources without considering the impact of overlap among the sources over data source selection, resulting in not only query processing overhead but also increased workloads on data sources. In order to improve query efficiency on overlapping data sources, this work proposes a tuple-level stratified sampling approach for overlapping data source selection. The approach has two stages: the offline stage and the online stage. In the offline stage, tuple-level stratified sampling is applied to obtain sample tuples. In the online stage, samples are used to estimate query coverage and overlap among multiple data sources. A heuristic method is also designed to discover data sources with low overlap. Experimental results show that the proposed approach is more efficient and effective than the state of the art methods for selecting overlapping data sources.

* 基金项目: 国家自然科学基金(61232002, 61202035); 湖北省科技支撑计划(2015BAA127)

Foundation item: National Natural Science Foundation of China (61232002, 61202035); Science and Technology Support Program of Hubei Province (2015BAA127)

收稿时间: 2016-01-31; 修改时间: 2016-04-21; 采用时间: 2016-05-24; jos在线出版时间: 2016-07-30

CNKI网络优先出版: 2016-08-01 09:39:01, http://www.cnki.net/kcms/detail/11.2560.TP.20160801.0939.007.html

Key words: data source selection; stratified sampling; tata source overlap estimation; regression

深网(deep Web)是指不能通过一般的搜索引擎查找内容的 Web 站点.深网数据通常存储在后台的数据库中,必须通过表单或者 Ajax 接口进行查询.其他类型的深网数据包括非文本文件(如音频、图像等)、动态内容等.深网数据源是 Web 上很重要的一类数据源.早在 2001 年的研究[1]表明:深网共有约 7 500TB 的数据,是表层网的 500 倍,分布在约 20 万个深网数据源中.查询深网数据源面临如下挑战[2]:1) 完全覆盖单个领域的数据需要集成成千上万个数据源;2) 不同数据源之间的数据存在很大的重叠;3) 多数深网数据源是非合作的,数据分布未知,查询次数有限(如 Yahoo!Auto 限制每个 IP 地址每天最多查询 1 000 次)[3].因此,选择合适的数据源进行查询不仅能够减少查询的开销,提高用户的满意度,而且可以减轻数据源的工作负载.

当前,数据源的选择着重于相关数据源的选择[4].相关数据源选择方法基于数据库摘要,估计查询与数据源之间的相关性,选择相关性得分前 N 的数据源.这些方法假定数据源之间的重叠在数据源选择时可以忽略,并在查询结果合并时被去除.但在存在大量数据源时,这个假定并不成立[2]:忽略数据源之间的重叠造成不同数据源上相似查询结果的重复查询,增加被查询数据源上的工作负载,浪费数据源的访问资源.我们以例 1 来简单说明重叠数据源选择问题.

例 1:如图 1 所示,给定查询 Q 和 5 个 Web 数据库 A,B,C,D,E.查询 Q 总共有 4 个查询结果元组,对应的在这 5 个数据源上的查询结果数量为 3,3,2,0,1.假如用户想要查询所有的查询结果,实际上不知道数据源之间关系时,会按数据源查询结果数量一直向下查,A,B,C,E,D,得到所有的 4 个元组.其中,查询 B,C,D 时没有新增元组.事实上,如果知道数据源之间的重叠关系,我们只需查询 A 和 E 即可得到所有的结果元组.

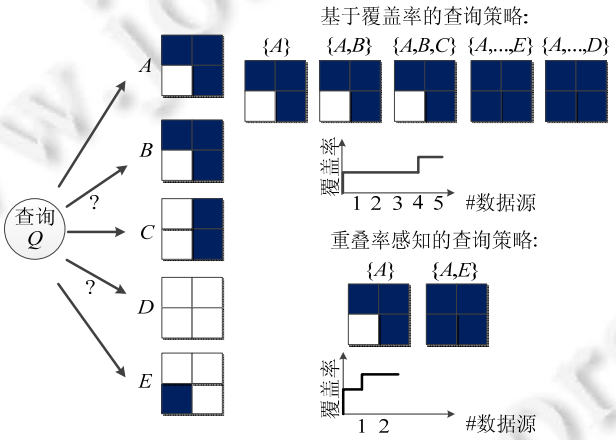


Fig.1 Example of overlaping data source selection

图 1 重叠数据源选择实例

尽管利用预先获取的数据源之间的重叠关系能够极大地提高数据源选择的效率,然而有效地估计数据源之间的重叠是非常困难的.例 1 中,5 个数据源之间(不仅仅是两两之间)的重叠变量总共有 2^5=32 个变量,当数据源数量较大时,需要估计的重叠变量数量将呈指数级爆炸增长,这使得重叠数据源选择的过程非常困难.

当前,重叠数据源选择工作大多假设给定覆盖率和重叠率[5],而实际中,覆盖率和重叠率的获取都是不精确的.现有的考虑覆盖率和重叠率的重叠数据源选择方法主要有基于抽样的方法[6]和基于最大熵的方法[7].

1) 基于抽样的方法

基于抽样的方法以 StatMiner(文献检索系统)[6]为主,StatMiner 研究基于查询的分层抽样方法解决静态 Top-K 数据源选择问题,使得选择的 K 个数据源中不同结果元组最多.StatMiner 对整个查询空间的查询进行分层抽样得到样本查询,基于查询结果(即样本数据)将查询和数据源都划分为类(即集合),学习查询类和数据源集

合类之间的覆盖率和重叠率.该方法有3点不足:首先,StatMiner生成样本的方法是基于查询分层抽样,而非基于元组分层抽样,这使得未被选择的查询没有查询结果;其次,StatMiner 需要学习所有的重叠率,适用于少量的数据源,而不适用于大量数据源的情况;最后,StatMiner 通过离线学习好的覆盖率和重叠率来对数据源静态排序,因而在实际选择时会因为较大的统计数据误差影响深网查询的效果.

2) 基于最大熵的方法

基于最大熵的方法以 OASIS[7]为主.OASIS 研究动态数据源排序问题,使得用户尽快检索到更多不同的结果元组.它利用数据库查询优化技术估计各数据源的覆盖率,并基于最大熵方法估计多数据源之间的重叠率,在线运行查询并不断校准数据源的重叠率,动态选择数据源.该方法能够容忍不精确和不完整的重叠率,但它假定覆盖率是给定且精确的.其不足在于:首先,在数据源非合作的情况下,无法利用数据库查询优化器来获得精确的数据源覆盖率;其次,在线估计数据源的统计数据会产生较大的时间开销,并不适用于深网数据源.图2给出了我们在实验中考察不同数量数据源时求解最大熵问题所花费的时间开销(我们随机生成20个查询,在不同数量的数据源下调用 matlab 运行最大熵算法(即 OASIS)获取的时间).

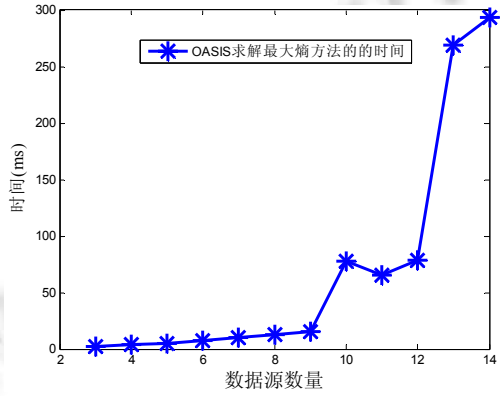


Fig.2 Time cost of OASIS approach

图2 OASIS 的时间开销

本文研究通过选择部分的数据源集合,以更少的查询代价来尽可能多地获得所有的查询结果.针对现有工作中未考虑或仅考虑部分统计数据(覆盖率和重叠率)的不足,本文首次提出元组级别的分层抽样方法,解决重叠数据源选择问题.具体方法如下(方法框架如图3所示).

- 首先,离线抽样获得分层样本.相对于简单随机抽样而言,分层抽样能够得到更精确的统计数据.由于不同数据源存在不同的元组数量,使用分层抽样的方法来估计查询在数据源上的统计数据并不容易,为此,我们提出一种误差约束的分层抽样框架,能够保证查询的覆盖率和重叠率的误差,从而保证重叠数据源选择的效率.主要策略是:基于决策树方法在查询属性中选择区分度大的分层属性,给定误差界确定每一层中抽取元组的数量,并随机抽取给定数量的样本.
- 其次,在线估计查询的覆盖率和重叠率,迭代地选择剩余覆盖率最大的数据源.
  1) 尽管通过样本可以直观地估计查询在数据源上的结果数量,然而估计查询的覆盖率和重叠率并不简单.我们基于历史查询,通过分类和回归估计查询在所有数据源的查询结果,得到查询在各数据源上的覆盖率.
  2) 直观来看,除非样本数量非常大,否则用多数据源的样本来估计多数据源重叠率的准确性非常低.因此,我们提出一种迭代式的重叠率估计方法:利用已查询数据源的查询结果和未查询的数据源的样本估计数据源之间的重叠率,提高重叠率估计的精度.
  3) 由于最大剩余策略需要比较所有未查询数据源的样本和已选数据源之间的重叠率,时间复杂度

较高,我们提出一种“K 近邻”的启发式重叠数据源选择算法,以提高查询效率.

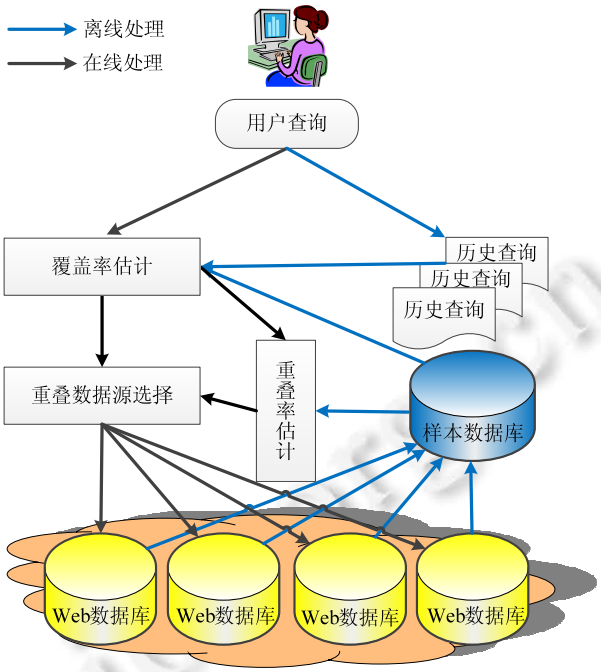


Fig.3 Approach framework

图 3 本文的方法框架

本文创新如下:(1) 我们研究了误差约束的分层抽样方法,解决了重叠数据源选择问题;(2) 我们提出了一种在线迭代的数据源重叠率估算方法,提高了覆盖率估计的精度;(3) 我们在最大剩余策略下提出一种启发式的重叠数据源选择算法,提高了数据源选择的效率;(4) 基准数据集和真实数据集上的实验结果表明,本文的方法能够在保证查询结果误差率的同时提高查询重叠数据源的效率.

1 相关工作

1.1 数据源摘要

在数据库领域,数据源摘要往往通过直方图、小波、Sketches 等[8]方法来构建,但这些方法均假设在数据源合作时,事先知道数据源的数据分布,并能对数据源进行聚合查询.

大多数 Web 数据源是非合作的,仅能利用现有的查询接口来访问数据源.现有很多工作处理深网数据源上的抽样和聚合估计:文献[3]提供了对 COUNT 和 SUM 查询的无偏估计方法,文献[9]描述了从深网中有效抽取随机样本进行聚合查询估计的方法,文献[10]提出在动态深网中进行有效的抽样来进行聚合查询估计.上述工作主要考虑给定单个查询进行抽样估计查询的聚合统计数据,不适合于抽取一批样本处理多查询的场景.

常见的数据源分析工作主要利用分层抽样方法[6,11,12].分层抽样是指将数据空间划分为多个层次,然后在每层中随机抽样,通过在数据分布广的层次抽取较多的样本,在数据分布窄的层次抽取较少的样本来提高聚合查询估计的准确率,因此在处理选择度低的查询时有较高的准确率.文献[6](StatMiner)考虑深网数据源中基于查询的分层抽样,分层方式是考虑所有查询空间的查询,选择一部分的查询并获得查询结果作为样本数据.这种方法仅能处理少量查询.其他深网上的分层抽样工作大多集中在数据挖掘领域:文献[11]考虑在结构化深网上进行关联规则挖掘和差分规则挖掘,但其抽样目的在于整体最小化抽样代价和估计误差;文献[12]考虑在结构化

深网上利用分层抽样方法进行 *K*-means 聚类分析,其目标在于通过抽样更好地获得真实数据集的聚类信息(聚类的中心和抽样比例).上述工作仅考虑在单数据源上分层抽样,处理数据挖掘等分析任务,无法应用于重叠数据源.

1.2 数据源选择

在非结构化数据源中,数据源的选择主要是基于大文档和小文档的方法,现有研究[13,14]都是基于小文档方法,即从数据源获得代表性的样本文档集合,在样本文档集合上建立倒排索引来进行查询,获得相关数据源集合.在非结构化数据源中,重叠数据源的估计是通过比较样本文档集合之间的重叠估计真实的重叠.文献[15]考虑在重叠文档数据源中,通过抽样利用 bloom 过滤器来估计两两数据源中的文档的重叠从而选择低重叠的数据源,但这种方法无法估计多数据源之间的重叠.

在结构化数据源(Web 数据库)中,同样可以构建数据库的摘要选择数据源[16]:通过查询获取查询结果,获得属性值-频率的统计数据,建立直方图.但在深网数据源中,我们无法直接获得属性值-频率的聚合估计值来建立直方图;同时,直方图数据无法用来估计数据源之间的重叠率.

基于抽样的重叠数据源选择工作主要是 StateMiner[6]方法.StatMiner 假设数据源和查询的关联被转化为类(集合)层次;基于样本数据,可以学习出类间(而非数据源间)覆盖率和重叠率的统计数据,选择 top-*K* 数据源集合,使得不同的查询结果数量最多.本文与 StatMiner 方法的不同在于:

1) 本文考虑选择一部分数据源来得到所有查询结果,使得总体查询开销最小.而 StatMiner 目标是选择 top-*K* 个数据源.
2) StatMiner 是基于查询进行分层抽样,在整个查询空间中进行抽样,再去得到查询结果作为样本.而我们的方法是对给定查询层次,基于其查询结果进行抽样,在每层简单随机抽样,使得结果数量少的查询仍然能够得到比较精确的结果.
3) StatMiner 假设所有的查询都是需要通过离线学习好的覆盖率和重叠率来对数据源静态排序,适用于少量的数据源.但实际中,由于产生较大的统计数据误差影响数据源选择的效率;同时,在大量数据源选择时,需要计算大量的统计数据,因而非常耗时.而本文利用离线样本和在线的数据源查询结果即可得到较为精确的统计数据,可以处理大量数据源的选择.

基于统计模型的重叠数据源选择工作主要是基于最大熵的方法 OASIS[7],OASIS 考虑动态的数据源排序,通过最大熵模型估计数据源之间的重叠率,在线执行查询,更新统计数据,得到下一步要查询的数据源.本文与 OASIS 的不同在于:

1) OASIS 目标是对数据源进行排序,使用户尽快得到更多的查询结果,需要查询所有的数据源.而本文的方法仅需查询一部分数据源就可以得到所有的结果.
2) OASIS 仅根据查询优化器来获得数据源的覆盖率和部分数据源的重叠率,忽略对样本数据的获取;同时,在线查询数据源获取查询在数据源上的统计数据,增加了查询的开销.而我们通过对数据源进行分层抽样,能够估计查询在数据源上的统计数据,减少访问数据源的次数.
3) OASIS 在估计数据源的重叠率时需要花费较大的时间开销,如图 2 所示,在仅有 15 个数据源时,OASIS 估计重叠率耗时 280s.而我们利用离线样本和在线查询结果,能够精确、快速地在线估计重叠率,适用于大量深网数据源.
4) OASIS 之所以能够迭代地更新统计数据,在于其假定可以实时发送查询到数据源获得统计数据,而这不适用于非合作深网数据源(限制每天查询的次数).
5) OASIS 仅考虑一个查询在多个数据源上的重叠率.本文研究不同选择度查询在多个数据源上的重叠率.

除了重叠数据源选择的研究,其他数据源选择工作考虑了数据源的质量[17]、时新性[18]、可信性[19]、收益[20]等因素.Rekatsinas 等人[17]研究数据源质量评估和数据源选择问题,他们利用知识库来评价数据源的质量,将数据源选择问题建模为多目标优化问题(如数据质量和查询代价),并允许用户交互式地选择数据源.Rehatsinas 等

人[18]还研究了动态数据源选择问题,使得集成代价约束下选择的数据源返回的结果集合新鲜性最大.他们利用历史数据建模数据源的变化规律,并估计数据源的时新性,利用时新性优化目标的子模性质提出一种有效的局部搜索方法选择时新的数据源.Balakrishnan 等人[19]考虑数据源可信性评价问题,他们在查询多数据源时,向多个数据源发送同样的查询集合,基于查询结果一致性来评价数据源的可信性.Dong 等人[20]研究如何选择数据源集合来平衡数据集成的精度和集成代价,他们提出了一种单调的数据集成模型,选择最大化总体的集成数据的精度的数据源集合.和上述工作相垂直,本文主要考虑重叠数据源选择问题,未来我们将考虑在重叠数据源选择问题中加入数据源的质量等因素.

综上所述,当前重叠数据源的研究无法同时很好地估计覆盖率和重叠率.本文提出一种元组水平的分层抽样方法,能够很好地估计覆盖率和重叠率,并提出一种启发式的重叠数据源选择算法,可以降低查询时间.

2 基础知识和问题定义

2.1 深网数据库模型

为了讨论方便且限于文章篇幅,本文主要讨论类别数据,并且不考虑有空值的元组.但我们的方法不止适用于类别数据,同样也可以适用于数值数据,比如数值数据可被离散化为类别数据.令$\Omega=\{d_1,d_2,\ldots,d_h\}$为给定的深网数据源(或数据库)集合,每个深网数据源$d\in\Omega$,d有$|d|$个元组$t_1,t_2,\ldots,t_{|d|}$和n个属性$A_1,A_2,\ldots,A_n$(我们通过数据集成中的模式映射技术 Global-As-View[21]建立数据源间的模式映射关系),d 中没有重叠元组(数据表通常通过定义主键约束来避免出现数据的冗余存储,因此不会出现重叠元组).令 $Dom(\cdot)$表示返回 1 个或多个属性的值域函数, $Dom(A_i)$表示属性 A_i 的领域;$Dom(A_i,A_k)$表示属性 A_i 和 A_k 值域的笛卡尔乘积;$|Dom(A_i)|$表示 $Dom(A_i)$的基数(cardinality),即属性 A_i 可能取值的数量.后文我们也称查询结果的数量为查询的基数.

用户仅能通过查询接口,即填写网页中的表单访问数据源 d.因此,一个用户查询 Q 为

$$\text{SELECT * FROM } d \text{ WHERE } A_{i1}=v_{i1} \,\&\ldots\&\, A_{is}=v_{is},$$

其中,v_{ik} 为 $Dom(A_{ik})$中的取值.在访问限制的约束下,尽管在一个周期中我们无法响应所有的查询,但可以利用多周期[10]来获得对应的数据源上的所有元组.

$Q(d)$为 Q 在数据源 d 上满足查询 Q 的元组集合,$|Q(d)|$为 Q 在 d 上的基数.类似地,$Q(\Omega)$为 Q 在Ω上的查询结果元组集合,$|Q(\Omega)|$为 Q 在Ω上的基数(我们假定 $Q(\Omega)$是去除冗余元组后的集合).

定义 1(覆盖率). 数据源 d 关于 Q 的覆盖率为 $C(d)=|Q(d)|/|Q(\Omega)|$.令 $D\subseteq\Omega$为数据源集合Ω的子集,D 关于 Q 的覆盖率定义为 $C(D)=\left|\bigcup_{d\in D}Q(d)\right|/|Q(\Omega)|$(我们假定覆盖率和重叠率是相对于给定的$\Omega$而言).

定义 2(重叠率). 数据源集合 $D\subseteq\Omega$关于查询 Q 的重叠率为 $O(D)=\left|\bigcap_{d\in D}Q(d)\right|/|Q(\Omega)|$, 其中,$\left|\bigcap_{d\in D}Q(d)\right|$为 D 中每个数据源都满足查询 Q 的元组的数量.

定义 3(查询代价). Q 查询数据源 d 的代价为 $cost(Q,d)=ta(d)+tr(Q(d))$,其中,$ta(d)$是查询数据源 d 的连接代价,$tr(Q(d))$为查询结果 $Q(d)$的传输代价.Q 在查询数据源集合 $D\subseteq\Omega$的查询代价为 $cost(Q,D)=\sum_{d\in D}cost(Q,d)$.

2.2 符号定义(见表1)

Table 1 List of symbols

表 1 符号定义

$\Omega=\{d_1,d_2,\ldots,d_h\}$	数据源空间(包含 h 个数据源)	$cost(Q,D)$	查询 Q 访问数据源集合 D 的代价
D	多数据源集合	Φ	查询 Q 的 *WHERE* 条件中的属性列
$t_1,\ldots,t_m$	数据源 $d\in\Omega$中的元组集合	$d(\Phi)$	$d\in\Omega$中所有Φ取值的集合
$A=\{A_1,A_2,\ldots,A_n\}$	数据源 $d\in\Omega$中的属性集合	$d(x)$	$d\in\Omega$中Φ值为 x 的元组集合
Q	用户查询	$S_x(d)$	$d(x)$中抽取的样本元组集合
$topK$	深网中查询结果页面出现的元组的最大数量	N_x	$d(x)$中抽取的样本元组数量
$C(D)$	数据源集合 D 上查询的覆盖率	$S(d)$	$d\in\Omega$中抽取的样本元组集合
$O(D)$	数据源集合 D 上查询的重叠率	$N(d)$	$S(d)$中样本元组的数量

2.3 问题描述

定义 4(重叠数据源选择问题). 给定数据源集合Ω、数据源样本 $S=\{S(d)|d\in\Omega\}$、历史查询集合(实验中,我们利用抽样过程中生成的查询作为历史查询),重叠数据源选择问题为:选择数据源子集 $\bar{D}\subseteq\Omega$, 在约束条件 $C(\bar{D})=1$ 时,最小化总查询代价 $cost(Q,\bar{D})$.

定理 1.

1) 重叠数据源选择问题是集合覆盖问题的推广[22],是 NP 难问题;

2) 贪心方法(基于最大剩余覆盖)求解集合覆盖问题是多项式时间的 $\ln\left(\left|\bigcup_{d\in\Omega}d\right|+1\right)$ 近似算法.

证明:

1) 重叠数据源选择问题的数学规划问题为

$$\left.\begin{aligned}&\text{Minimize } \sum_{d\in\Omega}x_d\cdot cost(Q,d)\\&\text{Subject to } C\left(\bigcup_{d\in\Omega}x_d d\right)=1\end{aligned}\right\} \tag{1}$$

$$x_d\in\{0,1\},d\in\Omega \tag{2}$$

其中,令 $U=Q(\Omega)$为 Q 在所有数据源上不同查询结果的全域,那么对于任一数据源 $d\in\Omega$,$Q(d)$是 U 的子集.因此,所有的$\Gamma=\{Q(d)|d\in\Omega\}$自然地构成了 U 的子集族,而 $cost(Q,d)$即为费用函数:$\Gamma\to\mathbb{R}^+$.因此,重叠数据源选择问题可归约为:找Γ的子集族,覆盖 U(对应约束(1)),使得该子集族的总体费用 $cost(Q,\bar{D})$最小(对应目标函数).

2) 由文献[23]可证.

直观地看,数据源间查询结果之间的重叠越少,总查询代价就越少.因此,解决重叠数据源选择问题需要考虑如下两个问题:

① 估计查询在数据源上的统计数据(覆盖率和重叠率);

② 基于给定的统计数据如何选择数据源.

针对问题①,由于在线查询数据源计算查询在数据源上的覆盖率和重叠率代价很大,因此,采用离线对数据源分层抽样来估计查询在各数据源的基数(见第 3.1 节);在估计数据源的覆盖率时,本文利用分类和回归的方法估计查询在Ω上的基数(见第 3.2 节);而在估计查询在多数据源上的重叠率时,我们基于离线样本和已选数据源的查询结果来估计数据源之间的重叠率(见第 3.3.1 节).针对子问题②,本文基于估计的重叠率提出启发式的最大剩余覆盖优先的重叠数据源选择算法(见第 3.3.2 节).

3 基于分层抽样的重叠数据源选择

本节我们详细描述基于分层抽样的重叠数据源选择算法.首先介绍数据源分层抽样,然后提出数据源覆盖率计算,最后介绍重叠数据源选择.

3.1 误差约束的数据库分层抽样

本文采用分层抽样而非均匀抽样的原因是:均匀抽样给不同的层分配层次数量比例的样本,当抽样比例低、层中的数据量少时,无法抽取层次中的样本.现有的分层抽样方法[24]尽管可以通过获取属性的值域来确立层次,但它们无法通过查询接口来进行随机抽样,因此无法有效地控制查询结果的误差.本文扩展Arjun[3]随机“下钻”获取随机样本的方法来实现任意数量样本的分层抽样.据我们所知,现有利用分层抽样方法来解决数据源选择问题的只有 StatMiner[6],但 StatMiner 是基于查询的分层抽样,使得抽样中未被选择的查询层次没有查询结果,产生较大的查询结果误差.下面我们先给出分层抽样的描述,然后给出我们的基于误差抽样的方法.

给定数据源 d(d 有$|d|$个元组和 n 个属性 $A=\{A_1,A_2,\ldots,A_n\}$)、分层属性集合$\Phi\subseteq A$.在 d 的所有元组中,定义 $d(\Phi)$为属性集合Φ上不同值 x 的集合,因此,对应于每个 x 都有 d 中的元组集合 $d(x)=\{t:t\in d$ 且 t 在属性值集合Φ上取值为 $x\}$,记 d 中属性集Φ上分组数量为$|d(\Phi)|$.

分层抽样的任务为:给定查询基数的相对误差ε>0,从 d 中选择样本 $S(d)\subseteq d$.对每个具体的层次 $d(x)$,存在对应的样本集合 $S_x(d)\subseteq S(d)$为 $d(x)$的子集,在 $S_x(d)$上运行查询估计查询基数.下面我们将给出单个数据源误差约束的分层抽样方法.

A. 确定抽样层次Φ:

选择分层属性集合可以由系统设计者自行制定或者使用自动化方法(比如决策树学习)基于信息增益的策略对属性选择排序.对于分层属性Φ,我们通过属性值域发现方法[25]得到所有的层次$\{x_0,\ldots,x_{|d(\Phi)|-1}\}$.

例 2:在 TPC-W 基准数据库 *Book* 表中,我们基于属性Φ=(*Year*,*Subject*)进行分层抽样,*Year* 有 70 个不同值,*Subject* 有 24 个不同值,总共最多有 1 680 个层次.比如,*Book* 中所有元组在Φ上的值为(2000,'*HISTORY*')的元组构成一个层次.

由于各数据源元组数量不同,因此对不同数据源抽取的样本数量不一.我们提出基于查询结果误差约束的抽样方法,利用 Chernoff Bound(Chernoff bound.https://en.wikipedia.org/wiki/Chernoff_bound)来控制抽样的数量.核心想法是:在每一层中抽取样本的数量,能够保证估计的查询结果误差在给定范围.

B. 确定每层样本量 N_x:

给定$x\in\{x_0,\ldots,x_{|d(\Phi)|-1}\}$,$d(x)$,查询 Q,置信度 0<δ<1,构造指示函数:任取元组 $t_i\in d(x)$,如果 $t_i\in Q(d(x))$,Y_i=1;否则,y_i=1.则 $\Pr[Y_i=1]=p=\dfrac{|Q(d(x)|}{|d(x)|}$,$\Pr[Y_i=0]=1-\dfrac{|Q(d(x))|}{|d(x)|}$,则 $Y_1,\ldots,Y_{N_x}$ 是独立的泊松观测.

令 $\bar{Y}=\sum_{i=1}^{N_x}Y_i$,$\mu=E[\bar{Y}]=N_xp$,对于任意给定的相对误差$\varepsilon$>0,由 Chernoff Bound 有:

$$\Pr[|\bar{Y}-E[\bar{Y}]|\geqslant N_xp\varepsilon]\leqslant 2\exp(-N_xp\varepsilon^2/(2+\varepsilon))=1-\delta \tag{3}$$

因此,层次 x 确定抽取的样本数量为

$$N_x=\min\begin{cases}\left\lceil\dfrac{(2+\varepsilon)|d(x)|}{|Q(d(x))|\varepsilon^2}\cdot\ln\dfrac{2}{1-\delta}\right\rceil\\ |d(x)|\end{cases} \tag{4}$$

总的样本数量 $N=\sum_{x\in\{x_0,\ldots,x_{|d(\Phi)|-1}\}}N_x$.由公式(4),$N_x$ 与|$d(x)$|成正比,与|$Q(d(x))$|成反比,并且当|$Q(d(x))$|=|$d(x)$|时得到最小数量的样本,对应的查询为查询所有 d_x 中的结果(SELECT * FROM d_x).对于层内元组数量大于 $KS(\varepsilon)=\left\lceil\dfrac{(2+\varepsilon)}{\varepsilon^2}\cdot\ln\dfrac{2}{1-\delta}\right\rceil$的层次而言,其抽样数量都是 $KS(\varepsilon)$;而对于层内元组数量小于 $KS(\varepsilon)$的层次,其抽样数量为|$d(x)$|.

例 3:给定置信度δ=0.8,相对误差ε=0.2,假定查询 Q 为 SELECT * FROM *Book*,则层次 x_1=(2000,'*HISTORY*')中共有 757 个元组,由公式(4),我们需要抽取 127 个元组;层次 x_2=(1970,'*POLITICS*')共有 12 个元组,由公式(4),需抽取 12 个元组.

C. 每层简单随机抽样选择样本 S_x:

由于我们仅能利用查询接口获取随机样本,我们利用文献[3]中提出的随机"下钻"生成查询的方法来获取样本,对每个 x,从 $d(x)$中简单随机抽样,构成样本 $S_x(d)$.其中,每个样本元组由如下步骤生成:

Step 1. 给定 x,我们获得查询 Q:Φ=x 在 d 上的基数|$Q(d(x))$|.当|$Q(d(x))$|>*topK*(*topK* 为每个网页中最大的输出元组数量),转至 Step 2;否则,转至 Step 3.

Step 2. 当|$Q(d(x))$|>*topK* 时,采用随机下钻的方法生成新的查询 Q':Φ=x & A_i=v_i,其中,A_i 为 $A\backslash\Phi$中随机选择的属性,v_i 为随机选择的属性值,转至 Step 1.

Step 3. 当|$Q(d(x))$|$\leqslant$*topK* 时,我们从查询结果中随机选择一个元组(我们通过蓄水池抽样[26]的方法来获得这些样本),如果该元组已被选择,则回到 Setp 2;否则,将该元组加入样本集合.

重复上述 3 步,直到获得 N_x 个不同的元组为止.

例 4:给定查询 $Q(d)$:*Year*=2000 & *Subject*='*HISTORY*',|$Q(d)$|=757,因此,我们需要去"下钻".考虑未被选择的属性,随机选择 *Publisher*=1(编号为 1 的出版社),生成新的查询查询 Q':*Year*=2000 & *Subject*='*HISTORY*' &

Publisher=1,其结果数量为 15,小于页面最大的元组数量 *topK*(实验中我们设定为 20),从中随机选择一个元组.然后继续从 $Q(d)$开始选择不重叠的元组,直至找到127个元组.实验中,为了提高抽样的效率,我们选择 $Q(d)$的前$\lceil N_x/topK \rceil$个页面来获取 N_x 个结果元组.

通过分层抽样,整个样本空间 $S(d)$是 d 中所有不相交的分层样本 $S_x(d)$的并集,在Φ上的分层样本由 N_x 确定.我们将抽取的样本数据存入图 3 所示的样本数据库中,并且在样本数据库中,对于抽取的每个元组,增加抽样比属性(*sampleratio*)为 $N_x/|d(x)|$,以及元组来自数据源的 *sourceid*.对于 $d(\Phi)$中的每个值 x,如果$|d(x)| \leqslant N_x$,即抽样比 *sampleratio* 为 1,那么样本包含所有从原始表中抽取的元组,该层提供精确的查询结果;当$|d(x)|>N_x$ 时,我们用抽取的 N_x 个元组估计查询的基数.利用抽取的样本 $S(d)$来估计查询 Q 在 d 上基数的期望为

$$E[Q]=\sum_{x\in\{x_0,\ldots,x_{|d(\Phi)|-1}\}}|Q_x|\cdot\frac{N_x}{|d(x)|} \tag{5}$$

其中,Q_x 为查询 Q 在分层样本 S_x 的结果元组集合.$\frac{N_x}{|d(x)|}$ 为抽样的概率,对应的方差为

$$Var(Q)=\sum_{x\in\{x_0,\ldots,x_{|d(\Phi)|-1}\}}\left(1-\frac{N_x}{|d(x)|}\right)\left(\frac{|d(x)|}{|d|}\right)^2\frac{Var(Q_x)}{|d(x)|} \tag{6}$$

其中,$Var(Q_x)$为查询 Q 在分层样本 S_x 的方差.

对于基本的聚合算子 AVG,SUM,COUNT,由于在每一层中的抽样是基于简单随机抽样,因此由公式(5)可知 $E[Q]$是无偏的;同时,N_x 直接确定了 Q 估计的基数,由公式(6)可知,这些聚合算子的方差反比于 N_x.

3.2 估计查询在数据源上的覆盖率

估计查询在数据源上的覆盖率需要计算查询在Ω上的基数$|Q(\Omega)|$,我们选择基于查询模板的机器学习方法计算$|Q(\Omega)|$[27].我们通过抽取查询在属性相关特征上的值,利用基于回归的方法学习不同查询在Ω上的基数.这里的查询模板是指基于 SQL 查询对应的 select 子句、from 子句和 where 子句中分别对应的属性和表.

在计算单个数据源上的查询基数时,我们考虑用抽样样本而不用学习的方法,是因为我们抽取一次样本就可以计算多次不同查询的结果.而在估算多个数据源上的查询结果时,由于总体样本数据量比较大,因此难以用基于样本的方法来算总体的查询结果数量,因此在估计总体的查询数量时,我们采用现有的黑盒式的机器学习方法,利用现有的机器学习工具包 liblinear(http://www.csie.ntu.edu.tw/~cjlin/liblinear/)来实现.下面是机器学习方法的训练阶段和预测阶段的处理流程.

A. 训练阶段

1) 对查询进行分类,按照查询的约束的类型来分类;2) 提取查询的特征,包括属性、属性值、属性个数、查询结果数量;3) 基于分组的查询,每组查询进行线性回归,确定回归模型 LR(logistic regression)(http://en.wikipedia.org/wiki/Logistic_regression)的参数.

B. 预测阶段

1) 对查询基于模板选择回归模型;2) 提取查询中的模板特征;3) 基于学习出来的 LR 模型和查询特征值估计查询在所有数据源上的结果数量$|Q(\Omega)|=LR(Q)$.

例 5:给定历史查询 $Q(\Omega)$:SELECT * FROM Ω WHERE PubYear between 1930 and 1935 & *Subejct*='*HISTORY*' & *Cover*='*Hardback*',我们基于训练阶段的结果得到特征向量(1,1,1,0,30,6,3,6),其中,向量中的前3个1代表 where 条件中出现的属性 *Pubyear*,*Subject* 和 *Cover*,0 代表未出现的属性 *Publisher*,30 和 6 代表出现的年份和持续的时间,6,3 和 6 代表 *Pubyear*,*Subject* 和 *Cover* 值在所有取值中的顺序.查询的$|Q(\Omega)|$值为 737.

基于抽样样本和学习的总体样本数据大小估算 Q 在数据源 d_i 上的覆盖率为 $C(d_i)=|Q(d_i)|/|Q(\Omega)|$.然后,我们对覆盖率由大到小进行排序.不失一般性,我们假设得到的数据源为 $d_1,\ldots,d_h$.

- 数据源样本抽样讨论

观测:通过 TPC-W 数据集上的实验我们发现,$|Q(\Omega)|$和 $cost(Q)$之间存在如下规律.

1) 当|$Q(\Omega)$|≥9835 时(图 4,查询选择度约为 0.1),得到 $Q(\Omega)$需要查询Ω.原因在于,当 $Q(\Omega)$较大时,数据分布较广,此时|$Q(\Omega)$|和 $cost(Q)$成正比例关系(如图 5 所示).比如,当我们想要获得|Ω|时,我们需要去查询所有的数据源.

2) 当|$Q(\Omega)$|<9835 时(如图 4 所示),我们通过样本估计|$Q(\Omega)$|的误差较大,想要准确估计|$Q(\Omega)$|需要查询较多的数据源.比如,我们想要查询单个元组,从大量的数据中抽样难以估计准确的查询基数,进而需要查询更多的数据源.

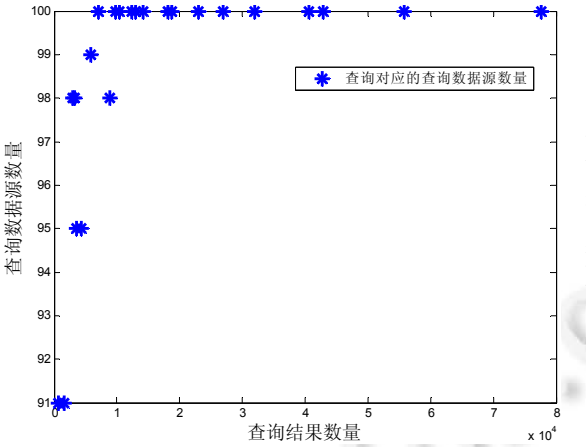


Fig.4 Number of queried data sources

图 4 查询的数据源数量

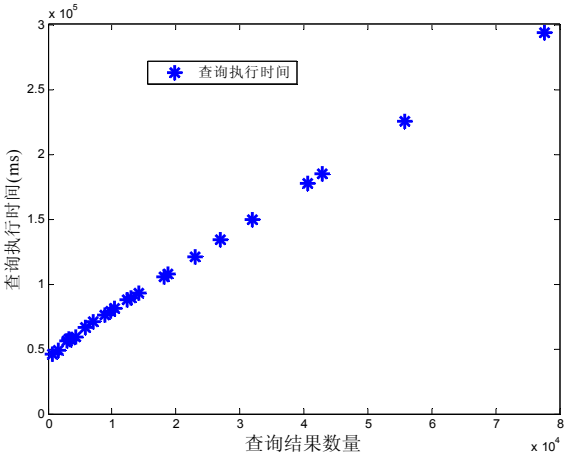


Fig.5 Query execution time

图 5 查询执行时间

结论:例 3 假定查询为查询所有数据源上的元组,但我们需要处理不同选择条件的查询,因此需要选择合适的查询基数决定抽样数量.查询基数较大时,由公式(4)知抽取的样本数量较少,用样本回答其他查询基数小的查询误差会比较大;查询基数数量少时,我们抽取的样本数量比较多,可以很好地回答所有的查询,但对应的抽样数量大.因此结合图 4,我们折中考虑查询选择度为 0.1 来推导我们实验中抽样数量.

3.3 重叠数据源选择

3.3.1 数据源重叠率估计

精确估计查询在数据源之间的重叠是很困难的,我们将讨论两种基于样本估计查询在数据源上重叠率的方法:完全基于样本的方法和部分基于样本的方法.

完全基于样本的方法是通过不同数据源之间的样本估计数据源之间的重叠率.以图 6 为例,按照覆盖率排序,我们知道 $C(d_1)>C(d_2)>C(d_3)$,那么先查询 d_1,然后选择查 d_2 还是 d_3.选择数据源的原则是考虑最大剩余覆盖的数据源,即考虑|$Q(d_2)-Q(d_1)$|和|$Q(d_3)-Q(d_1)$|的大小,完全基于样本的策略是:通过样本中重叠的元组来估计原始的数据源重叠元组的数量.假设查询 Q 在数据源样本 $S(d_1)$和 $S(d_2)$上查询结果元组集合为 $Q(S(d_1))$和 $Q(S(d_2))$,那么我们可以推导出[15]查询 Q 在据源 d_1 和 d_2 中的重叠率为

$$O(Q(d_1),Q(d_2))\approx\frac{|Q(d_1)|\times|Q(d_2)|\times|Q(S(d_1))\cap Q(S(d_2))|}{|Q(S(d_1))|\times|Q(S(d_2))|\times|Q(\Omega)|} \tag{7}$$

其中,|$Q(d_i)$|可以通过样本 $S(d_i)$,基于公式(5)估计出来(i=1,2).通过公式(7),我们可以估计|$Q(d_2)-Q(d_1)$|=|$Q(d_2)$|−|$O(Q(d_1),Q(d_2))$|×|$Q(\Omega)$|.类似地,我们可以估算|$Q(d_3)-Q(d_1)$|.然而由公式(7)可知:当样本数量很少时,要么结果数量为 0,要么结果非常大,通过这种方法估计查询在不同数据上的结果数量误差非常大.因此,我们将利用部分基于样本的方法来估计数据源之间的重叠率.

部分基于样本的方法是通过已查询的数据源结果和未查询数据源的样本来估计数据源之间的重叠率.如图 7 所示,按照覆盖率排序,我们知道 $C(d_1)>C(d_2)>C(d_3)$,需要估计|$Q(d_2)-Q(d_1)$|和|$Q(d_3)-Q(d_1)$|的大小来选择剩

余覆盖最大的数据源.我们已获得 $Q(d_1)$,通过比较 $Q(d_1)$和 $Q(S(d_2))$的相同元组来估计 $Q(d_1)$和 $Q(d_2)$中重叠元组的数量.

$$O(Q(d_1),Q(d_2))\approx\frac{|Q(d_2)|\times|Q(d_1)\cap Q(S(d_2))|}{|Q(S(d_2))|\times|Q(\Omega)|} \tag{8}$$

其中,$|Q(d_i)|$可以通过样本 $S(d_i)$,基于公式(5)估计出来(i=1,2).

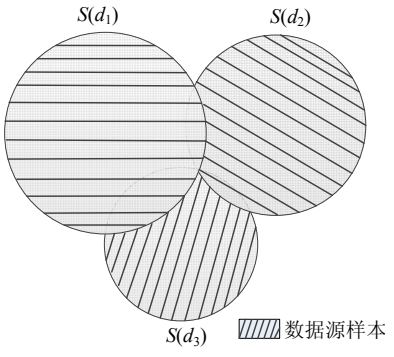


Fig.6 Overlap estimation based on sample

图 6 基于样本的重叠率估计

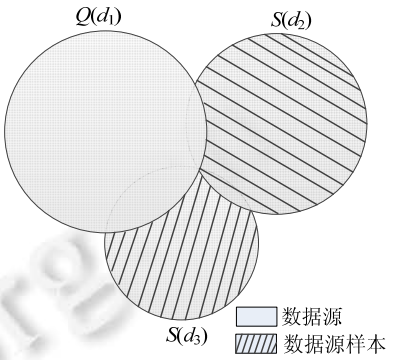


Fig.7 Overlap estimation based on sample and query result

图 7 基于样本和查询结果的重叠率估计

通过公式(8),我们可以估计$|Q(d_2)-Q(d_1)|$,进一步可以估计$|Q(d_3)-Q(d_1)|$的大小.这样,我们只需一个数据源的样本就能估计两个数据源的重叠元组数量.同时,样本数量越多,我们估计的重叠率就越准确.从而我们有如下定理.

定理 2. 假定已查询的数据源为 D_{Sel},查询结果集合为 $Q(D_{Sel})$,d_k 为未选数据源集合 D_{Unsel} 中的任一数据源,则查询 Q 关于 D_{Sel} 和 d_k 的重叠率估计量为 $O(Q(D_{Sel}),Q(d_k))\approx\dfrac{|Q(d_k)|\times|Q(D_{Sel}))\cap Q(S(d_k))|}{|Q(S(d_k))|\times|Q(\Omega)|}$.

证明:令 $M=Q(D_{Sel})\cap Q(d_k)$,$m_k=Q(S(d_k))\cap M$.对 d_k 简单随机抽样,有$|m_k|=|Q(S(d_k))|\times M/|Q(d_k)|$.已知 $Q(D_{Sel})$和$Q(S(d_k))$,我们用$|Q(D_{Sel})\cap Q(S(d_k))|$来近似$|m_k|$,即有 $O(Q(D_{Sel}),Q(d_k))\approx\dfrac{|Q(d_k)|\times|Q(D_{Sel}))\cap Q(S(d_k))|}{|Q(S(d_k))|\times|Q(\Omega)|}$. □

尽管我们给出了部分基于样本的数据源重叠率估计方法,然而重叠率估计的准确性主要由查询和抽样样本确定.直观上,各数据源仅有完整查询结果的部分数据,因此查询在不同数据源上的基数很大,抽样数量越多,估计的准确性越高.值得说明的是,随着查询的数据源越来越多,合并的查询结果集合越来越趋近于完整的结果,这使得未被选择的数据源和已得到的查询结果之间的重叠率估计越来越准确.后面,我们将通过实验来说明查询选择度和抽样数量对查询在数据源上的重叠率影响.

3.3.2 重叠数据源选择算法

正如我们在第 3.3.1 节中介绍的,完全利用样本的方法会产生较大的误差,因此,我们利用基于部分样本的方法来估计数据源之间的重叠.由于数据源选择问题是集合覆盖问题的推广,直观的想法就是利用贪心算法来求解数据源选择问题,我们已在定理 1 中说明:在准确计算数据源查询结果时,贪心算法求解集合覆盖问题是多项式时间的 $\ln\left(\left|\bigcup_{d\in\Omega}d\right|+1\right)$ 方法.但由于抽样导致估计查询结果的精度不同,每次选择剩余覆盖最大的数据源是不精确的,因此,我们无法直接在利用样本估计覆盖率的情况下给出基于样本的贪心方法的近似保证,但我们将在实验部分讨论贪心方法的误差.

给定估计的覆盖率和备选的 h 个数据源.选择数据源的基本想法是:基于覆盖率大小顺序,选择未选数据源中(相比已选数据源)剩余覆盖最大的数据源.首先,我们给出基于样本的最大剩余覆盖数据源优先的算法 OverlappingSourceSelection(如图 8 所示),即每次从未被查询的数据源中选择估计的剩余覆盖率最大的数据源.

假设已查询的数据源集合为 D_{Sel},合并的查询结果为 $Q(D_{Sel})$,未查询的数据源集合为 D_{Unsel},我们需要估计每个数据源 $d_i \in D_{Unsel}$ 和 D_{Sel} 之间的重叠率.算法如图 8 所示.算法的第 3 行~第 5 行估计未选数据源和已选数据源查询结果间的重叠率;第 6 行是选择新增元组数量最多的数据源;第 7 行~第 9 行运行查询计算数据源 d_t 中新增元组的数量,并移除已查询的数据源;第 10 行是跳过新增覆盖率为 0 的数据源.算法在总体数据源覆盖率为 1 或 D_{Unsel} 为空时停止.算法 1 的时间复杂度为 $O(h^2)\times f(Q)+cost(Q,D_{Sel})$,其中,$f(Q)$为每次估计已查询结果 RS 和未查询数据源 d_t 之间重叠率的时间.由于总共有 h 个数据源而每次都需要和剩下所有的数据源进行比较,总共最多需要比较 $h(h-1)/2$ 次,因此,在本地估计重叠率的总的时间为 $O(h^2)\times f(Q)$,而在线查询的代价为 $cost(Q,D_{Sel})$,即,查询所有选择的数据源 D_{Sel} 对应的开销.

算法 1. *OverlappingSourceSelection*(Q,Ω,S).
Input: Q /*查询*/ $\Omega=\{d_1,d_2,\ldots,d_h\}$ /*所有数据源*/
$S=\{S(d_1),\ldots,S(d_h)\}$ /*数据源样本*/
Output: D_{Sel} /*已选数据源*/
RS /*已选数据源的查询结果元组集合*/
1 $D_{Sel}=\{d_1\}$, $RS=Q(D_{Sel})$, $\delta=0$, $D_{Unsel}=\Omega\backslash d_1$;
2 **while** $C(D_{Sel})<1$ **OR** $D_{Unsel}\neq\varnothing$ **do**
3 **foreach** $d_j\in D_{Unsel}$ **do**
4 $O(Q(d_j)\cap Q(D_{Sel}))\approx O(Q(S(d_j)),RS)$;
5 **end**
6 $t=\arg\max_{d_j\in D}(C(d_j)-O(d_j\cap D_{Sel}))$;
7 **if** $C(d_t)-O(d_t\cap D_{Sel})>0$
8 $RS=Q(D_{Sel})\cup Q(d_t)$;
9 $D_{Sel}=D_{Sel}\cup d_t$, $D_{Unsel}=D_{Unsel}\backslash d_t$;
10 **else** $D_{Unsel}=D_{Unsel}\backslash d_t$;
11 **end**
12 **end**
13 **return** RS;

Fig.8 Overlapping data source selection algorithm

图 8 重叠数据源选择算法

由于算法 1 考虑 $Q(D_{Sel})$与每个 $d_i\in D_{Unsel}$ 的重叠,时间复杂度较高,因此我们每次仅考虑 D_{Unsel} 中覆盖率较大的前 k 个数据源,提出算法 2,即 *KOverlappingSourceSelection* 算法.直观的原理是,覆盖率较大的数据源更有可能存在剩余覆盖大的数据源.假定 D_{Unsel} 中覆盖率最大的前 k 个数据源为 $K(D_{Unsel})$,k 值太小,只能找到局部最优的剩余覆盖大的数据源;而 k 值较大,估计重叠率会产生较大开销.因此,我们通过实验确定如何选择 k 值.实验中我们发现,当 k=1 时 m 对应的算法效果最好,具体算法如图 9 所示.

算法 2. *KOverlappingSourceSelection*(Q,Ω,S).
Input:
Q /*Query*/ $\Omega=\{d_1,d_2,\ldots,d_h\}$ /*All Sources*/
$S=\{S(d_1),\ldots,S(d_h)\}$ /*Samples of Sources*/
Output: D_{Sel} /*Selected Sources*/
RS /*Result tuples of Already Queried Sources*/
1 $D_{Sel}=\{d_1\}$, $RS=Q(D_{Sel})$, $\delta=0$, $D_{Unsel}=\Omega\backslash d_1$;
2 **while** $C(D_{Sel})<1$ **OR** $D_{Unsel}\neq\varnothing$ **do**
3 $(RS,d_j)=OverlappingSourceSelection(Q,K(D_{Unsel}),S)$;
4 **end**
5 **return** RS;

Fig.9 K overlapping data source selection algorithm

图 9 K 重叠数据源选择算法

由于算法 2 中比较已选数据源和重叠数据源之间的重叠率只需考虑前 k 个数据源,因此对应的时间复杂度为 $O(kh)\times f(Q)+cost(Q,D_{Sel})$,其中,$f(Q)$为每次估计计算已查询结果 RS 和未查询数据源 d_t 之间重叠率的时间.由于总共有 h 个数据源,而每次都需要和未被选择的数据源中覆盖率最大的 k 个进行比较,总共最多需要比较 kh 次,因此在本地估计重叠率的总的时间为 $O(kh)\times f(Q)$,而在线查询的代价 $cost(Q,D_{Sel})$为查询数据源集合 D_{Sel} 的代价.

4 实验结果及分析

4.1 实验准备

TPC-W 数据集.我们用 TPC-W 基准[28]生成 97 671 个不同的元组,存储在关系表 *Book*(*book_id*,*pubyear*,*publisher*,*subject*,*cover*)中.我们考虑将这些元组划分到 100 个数据源(即 100 个数据表),每个元组随机分配到 1 个~20 个数据源中.但选择的 20 个数据源的范围通过 Zipf 分布确定,对应的参数 z=1.各数据源统计数据见表 2.我们在 *Book* 关系表中选择相对区分度大的 *pubyear* 和 *subject* 作为分层属性.通过查询在线书店(京东、当当等),我们观测数据源的平均的连接时间为 200ms~800ms,每个元组的平均传输时间为 0.3ms.实验中,利用这些统计数据模拟数据源的连接时间和数据的传输时间.此外,通过观察在线书店的页面,我们设置 *topK*=20.

Abebooks 数据集.我们使用公开的在线数据 Abebooks 数据集[29],该数据集包含 895 个数据源,1 265 个不同的元组,共 24 819 个元组.我们使用文献[7]中的方法将其扩展成 1 260 个数据源,1 265 个不同的元组,共 33 865 元组.各数据源统计数据见表 2.我们选择相对区分度大的 *pubyear* 和 *subject* 作为分层属性.通过在线测试,数据源的连接时间为 141ms~300ms,每个元组的传输时间为 0.35ms,这里,*topK*=20.

Table 2 Statistics of the data sets

表 2 数据集的统计数据

	TPC-W 数据集		Abebooks 数据集	
	数据源覆盖率	Pairwise 重叠率	数据源覆盖率	Pairwise 重叠率
min	0.018 7	0.002	0.000 8	0
averge	0.105 3	0.013 5	0.021 2	0.000 5
std	0.079 4	0.020 3	0.082 9	0.008 3
max	0.245 2	0.106 4	0.866 4	0.773 1

抽样方法.我们比较 3 种主要的抽样方法:简单随机抽样、基于查询的分层抽样和基于元组的分层抽样方法.我们建立样本表 *Sample*(*book_id*,*pubyear*,*publisher*,*subject*,*cover*,*sampleratio*,*sourceid*),相对于 *Book* 表,我们增加了字段抽样概率 *sampleratio* 和数据源 *sourceid*.

简单随机抽样.随机选择一个初始查询条件(查询属性和属性值),当查询结果大于单个网页显示的最大元组值 *topK* 时,我们随机增加查询条件(查询属性和属性值),直到查询结果小于等于 *topK* 为止,保存查询结果.重复这一过程,直到获得所需数量的样本为止.

基于查询的分层抽样(即 StatMiner[6]提出的抽样方法).在所有的层次中,我们随机选择 1 个层次,获取并保存该层次所有的查询元组.重复上述过程,随机选择另一层次,直到获得所需数量的样本.值得注意的是,由于每次查询只能获得单个网页上的查询元组,因此当单个层次中的结果数量远大于 *topK* 时,我们需要发送多个查询来获得该层次的所有查询元组.

基于元组的分层抽样.本文提出的方法.实验中,如果确定抽取的样本数量 N_x,我们通过查询该层次查询结果中的前$\lceil N_x/topK \rceil$个页面来获取 N_x 个结果元组.

算法.我们实现并比较如下数据源选择算法.

- Random:随机选择数据源的数量.
- Coverage[5](*Coverage*):按覆盖率大小降序选择最大覆盖率的数据源.
- Maxent 算法[6](OASIS)(OASIS 使用 LINDO 软件,但免费版 Lindo 最多能够计算 300 个变量,因此我们使用 Matlab 来求解最大熵问题):利用最大熵模型在线估计数据源之间的重叠率,查询数据源不断校正数据源的覆盖率,选择最优剩余覆盖的数据源.其中,用 Matlab 中的非线性 interior-point 算法(http://cn.mathworks.com/help/optim/ug/fmincon-interior-point-algorithm-with-analytic-hessian.html)求解最大熵问题.
- OverlappingSourceSelection(OSS):本文提出的基于样本最优算法.
- KOverlappingSourceSelection(KSS):本文提出的基于样本的启发式数据源选择算法.

- Full-knowledge(最优算法)(*Optimal*):基于精确的统计数据以及最大剩余覆盖策略查询数据源得到的时间.

我们用 Java 1.6 在 Windows7 上(3.2GHZ Intel Core i5 CPU 和 8G 的 RAM)实现了上述所有的算法,并用 PostgreSQL 9.4 数据库来存储数据.

评价标准.我们随机选出 20 条不同选择度的查询作为测试查询,其中,查询结果的选择度分布在 0.006~0.79 之间.为简单起见,将选择度在 0.006~0.1 之间的查询称为低选择度查询(0.1 对应查询结果数量为 9 835),选择度在 0.1~0.2 之间的查询称为中选择度查询,0.2~0.79 之间的查询称为高选择度查询.实验中,考虑 3 类选择度查询中的均值.我们考虑如下两个方面的评价尺度:(1) 在抽样方法中,我们考虑抽样的代价(即获得样本所需要发送数据源的查询数量)和查询基数的误差(包括覆盖率相对误差和重叠率相对误差);(2) 在数据源选择的过程中,我们考虑查询性能(查询的平均执行时间和查询的数据源数量)和查询基数的相对误差.

4.2 TPC-W合成数据集

首先需要说明的是,基于查询的抽样方法不能直接获得其潜在的数据分布,因此无法直接分析查询在其样本上的误差.因此,我们对每个数据源分别用不同的抽样方法抽取 5%,7.5%,10%,12.5%的元组作为样本数据.

4.2.1 统计数据精度对比

表 3 和表 4 分别展示了不同的抽样方法随样本数量变化时,不同选择度查询的覆盖率和重叠率的平均相对误差.

Table 3 Relative error of coverage of queries with different selectivities on TPC-W dataset

表 3 TPC-W 数据集上不同选择度查询对应的覆盖率的相对误差

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	3.72 597	2.519 192	1.278 375
	7.5%	3.446 358	2.344 643	1.203 055
	10%	3.281 781	2.341 838	1.237 787
	12.5%	3.087 59	2.244 073	1.218 875
基于查询的分层抽样方法	5%	0.479 971	0.286 146	0.221 066
	7.5%	0.311 769	0.189 628	0.159 537
	10%	0.329 346	0.183 228	0.147 664
	12.5%	0.257 54	0.121 993	0.158 55
基于元组的分层抽样方法	5%	0.051 175	0.029 572	0.017 057
	7.5%	0.035 854	0.022 802	0.013 375
	10%	0.052 166	0.020 331	0.009 4
	12.5%	0.012 283	0.005 312	0.002 57

Table 4 Relative error of overlap of queries with different selectivities on TPC-W dataset

表 4 TPC-W 数据集上不同选择度查询对应的重叠率的相对误差

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	3.617 978	2.477 992	1.263 479
	7.5%	3.279 878	2.296 687	1.189 95
	10%	3.172 23	2.311 785	1.224 768
	12.5%	2.955 611	2.216 346	1.205 545
基于查询的分层抽样方法	5%	0.420 986	0.287 543	0.218 621
	7.5%	0.293 393	0.177 304	0.157 186
	10%	0.310 065	0.182 72	0.146 362
	12.5%	0.237 326	0.154 075	0.123 39
基于元组的分层抽样方法	5%	0.059 538	0.049 228	0.045 927
	7.5%	0.058 951	0.048 822	0.045 071
	10%	0.061 202	0.046 481	0.042 628
	12.5%	0.055 694	0.035 186	0.028 889

从表 3 的实验结果来看,

(1) 本文提出的基于元组的分层抽样方法对应的查询覆盖率误差最小;简单随机抽样方法估计的覆盖率误差最大,均大于 1.其原因在于,简单随机抽样相对于基于元组的分层抽样会在元组数量少的层次中

抽取较少的元组,因此误差较大;而基于查询的分层抽样方法仅选择一部分层次代表整个数据源,其误差也大于基于元组的抽样方法的查询覆盖率误差,但相对于简单随机抽样而言基于查询层次来抽样,使得新的查询在已选的层次上对应的误差较小,而在未选的层次上误差较大,但总体的误差比均匀抽样时的误差要低.

(2) 3 种抽样方法的覆盖率误差都会随着样本数量的增加而减小,覆盖率误差均随着查询选择度的增加而减小.

从表 4 中容易看出,基于元组的分层抽样方法对应的查询覆盖率误差最小.其原因在于,样本数量增加,查询选择度增加,使得满足查询的样本元组越多,估计的覆盖率就越准确.表 4 的实验结果整体上和表 3 中的趋势一致,然而由于查询在数据源之间重叠结果一般小于查询在对应数据源上的查询结果,对应的相对误差较大,这使得重叠率的相对误差比覆盖率的误差更大,尤其是对分层抽样方法(覆盖率更准确).

表 5 展示的是在少量(10 个)数据源时最大熵方法和分层抽样方法估计的重叠率的相对误差(我们仅考虑 10 个数据源是因为 matlab 求解最大熵问题非常耗时),其中,最大熵方法和分层抽样方法中都是首先通过分层抽样方法估计的覆盖率再估计重叠率.从实验结果来看:(1) 最大熵方法估计的重叠率的误差远大于基于元组分层抽样方法估计的重叠率的误差;(2) 基于分层抽样估计的相对误差随着查询选择度的增加和样本数量增加而减小,但最大熵的方法估计的重叠率误差却维持在较高的水平.其原因在于:

- 首先,由于估计的覆盖率(即最大熵方法的输入)本身存在误差.
- 其次,由于 OASIS 是通过伸缩性地放松约束条件求解最大熵问题,因此同样的输入会产生较大的误差;而基于元组的分层抽样方法能够利用在线的查询结果来不断校准数据源之间的重叠率,因此对应的误差较小.

Table 5 Relative error of overlap of queries with different selectivities on TPC-W dataset (10 data sources)

表 5 不同选择度查询对应的重叠率的相对误差(10 个数据源)

查询选择度(%)		低选择度查询	中选择度查询	高选择度查询
最大熵方法	5	0.554 757 143	0.588 466 667	0.585 014 286
	7.5	0.587 6	0.579 133 333	0.616 642 857
	10	0.567 771 429	0.581 744 444	0.620 842 857
	12.5	0.600 8	0.594 422 222	0.611 457 143
基于元组的分层抽样方法	5	0.117 220 433	0.164 845 786	0.136 407 085
	7.5	0.113 384 968	0.144 323 954	0.112 750 37
	10	0.095 979 348	0.129 168 432	0.099 355 271
	12.5	0.064 309 235	0.096 109 623	0.053 030 278

表 6 展示了不同抽样方法生成相同比例样本数据所需的查询数量.从实验结果来看,基于分层抽样方法所需要的查询数量最少;而随着样本数量的增加,查询的数量在不断增加.值得注意的是:(1) 基于元组分层抽样的方法在 7.5%和 10%的样本时查询数量是一样的,这是因为此时每一层中查询的元组都在相同的页面中;(2) 简单随机抽样方法由于需要不断“下钻”,因此需要更多的查询,而基于元组的抽样方法和基于查询的抽样方法随着样本数量的增加而越来越趋近,这是因为两种方法都是通过对层次中的元组进行抽样所致(选择页面中的元组所致).

Table 6 Query number required by sampling given number of tuples on TPC-W dataset

表 6 TPC-W 数据集上抽取给定数量元组所需的查询数量

抽样方法	5%	7.5%	10%	12.5%
简单随机抽样	13 439	21 104	29 648	39 182
基于元组的分层抽样	7 100	8 520	8 520	11 360
基于查询的分层抽样	8 644	9 497	10 176	11 382

4.2.2 执行时间对比

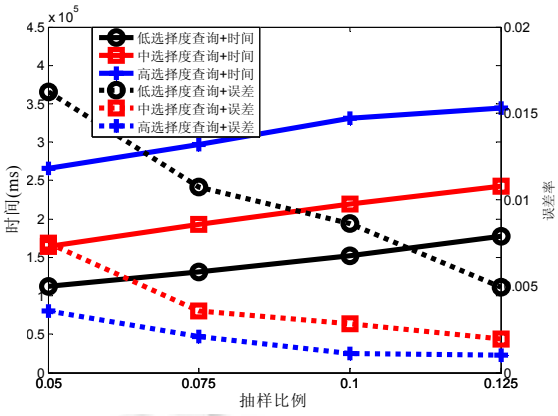
图 10(a)和图 10(b)分别对应不同选择度查询和不同样本下 OSS 算法的效率、查询基数误差和查询的数据

源数量.从图 10(a)我们可以看到:

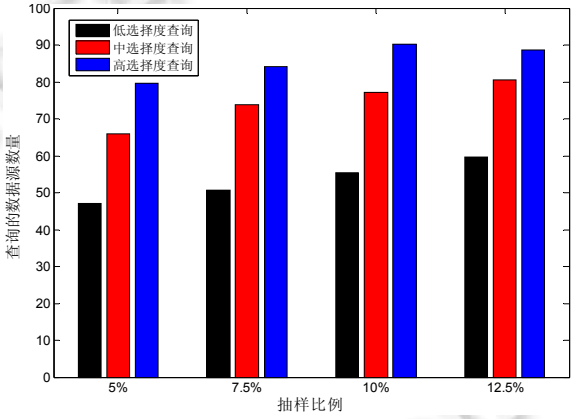
(1) 查询选择度越高,则查询执行时间增大,查询结果的相对误差越小,在 12.5%的样本上,所有查询对应的误差率能够保证在 0.5%以下.值得注意的是,对于高选择度查询,查询的执行时间随着样本数量增加而变缓,其原因在于,样本数量增加导致覆盖率和统计率估计的误差较小,使得查询的数据源数量变少(如图 10(b)所示).

(2) 随着样本元组数量变大,查询结果的误差在不断减小,其中,高选择度查询误差最小,低选择度查询的误差最大.如图 10(b)所示,随着样本数量的增加,OSS 查询的数据源数量逐渐增加并逐渐趋于稳定.

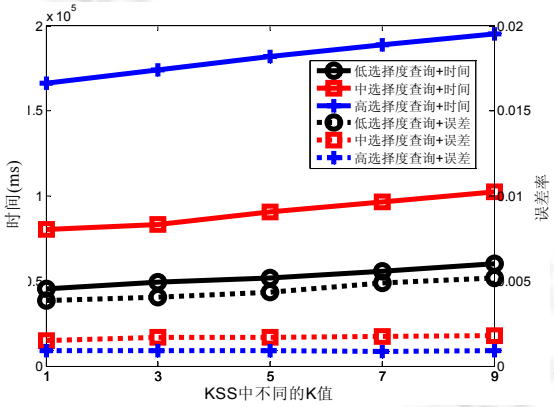
图 10(c)和图 10(d)分别对应不同选择度查询和不同样本下 KSS 算法在不同的 k 值时所对应的查询执行时间、查询结果误差和查询数据源数量.从图 10(c)可以看到,随着 k 值增大,不同选择度查询的执行时间增加.这是因为 KSS 需要估计多个数据源和已选数据源结果之间的重叠率,但对应的查询结果误差并不是随着 K 值的增加而减小,因为估计的数据源查询结果元组数量较少使得估计的最大剩余覆盖对应的误差变大(图 10(c)中,低选择度查询的误差在增加).图 10(d)显示,查询的数据源数量随着 k 值的增大基本不变.其原因在于,按最大剩余覆盖选择数据源基本上选 k=1 时对应的数据源.



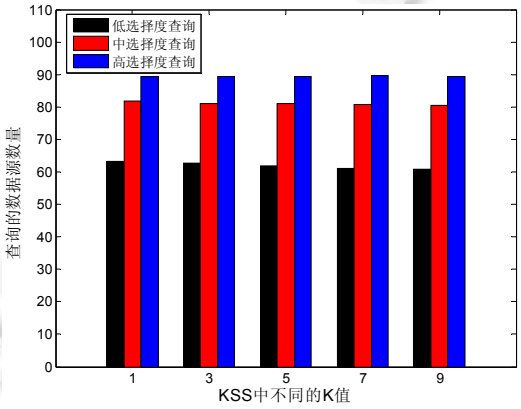
(a) OSS 算法在查询和样本上的执行时间



(b) 查询的数据源数量



(c) KSS 中不同的 K 值对查询效率和结果误差的影响



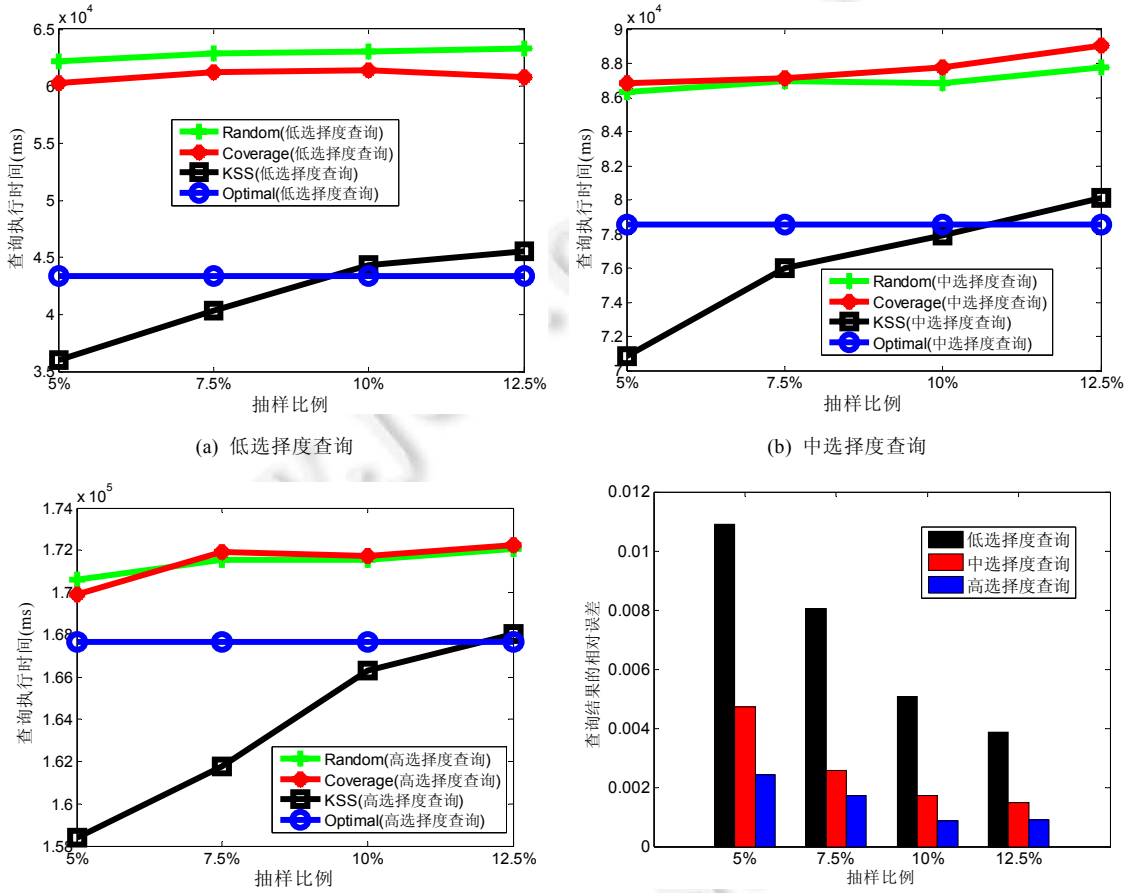
(d) 查询的数据源数量

Fig.10 Algorithm analysis of OSS and KSS on TPC-W dataset

图 10 OSS 和 KSS 在 TPC-W 数据集上的算法分析

综合图 10 中 OSS 和 KSS 的分析,KSS 的执行时间和查询基数误差上均优于 OSS.而 KSS 在 k=1 时对应的误差率和查询执行时间均为最优.下面我们用使用 KSS 算法(k=1)与现有方法进行比较.

图 11 展示的是 KSS 和其他不同的数据源选择算法在查询时间、误差上的分析.从图 11(a)~图 11(c)不难看到:KSS 算法的执行时间在小数据量样本时,优于最优的时间,代价是较大的误差(如图 11(d)所示);而 Coverage 算法在低选择度查询时优于 Random 算法,但在其他选择度查询时性能近似.其原因在于,中选择度和高选择度查询需要查询的数据源较多,因而相对的查询的数据源数量近似,查询时间趋于一致;随着样本数量增大,KSS 的执行时间不断增加,其并超过 Optimal 的查询时间.值得注意的是,图 11(a)~图 11(c)中,KSS 和 Optimal 相交的点对应的样本数量随查询选择度增加而增加.其原因在于:(1) 样本数量增大,估计查询的覆盖率和重叠率时间增大;(2) 样本数量越多,查询选择度越大.结合图 11(d)不难得出,其对应的结果误差在不断减小,查找的元组就越来越多.总体上,我们可以将查询结果误差控制在 1.2%以内.



(a) 低选择度查询 (b) 中选择度查询

(c) 高选择度查询 (d) KSS 对不同查询在不同样本上的结果误差的影响

Fig.11 Performance of selection algorithms with different queriesons on TPC-W dataset

图 11 TPC-W 数据集上不同选择算法在不同查询上的性能

表 7 分析了仅有 10 个数据源时,KSS(K=1)和 OASIS 算法在查询效率上的对比(OSASIS 需要估计所有的重叠率变量,使用 matlab 运算最大熵算法耗时很长).从表 7 中不难看出,估计数据源重叠变量的时间占据了 OASIS 的主要时间,平均每次估计耗时 15 375ms,远大于 KSS 查询数据源的时间;同时也可以看出,KSS 算法近似于 Optimal 算法的运行时间.

表 8 和表 9 分别分析了 KSS 在不同抽样方法下对应的查询执行时间和查询基数误差.

从表 8 我们可以看到,基于元组的分层抽样方法和基于简单随机抽样方法的查询执行时间近似,但均高于

基于查询的分层抽样的时间.其原因在于基于查询的分层抽样仅抽取一部分的层次,因而对未抽取的层次估计的查询基数为0,从而忽略很多数据源,导致查询时间快,查询基数的误差较大(见表9).尽管基于元组的分层抽样方法和简单随机抽样方法在查询执行时间上近似,但在查询结果误差上,前者显然要优于后者.其原因在于,基于元组的抽样方法能够提供更精确的覆盖率和重叠率估计.值得说明的是,在估计覆盖率和统计率时,简单随机抽样方法估计的统计数据的误差要远大于基于查询的分层抽样方法的误差;在估计总体的查询结果误差时,简单随机抽样的结果要劣于后者.其原因在于,简单随机抽样是均匀地在每层抽取等比例的样本,导致估计的覆盖率和重叠率率远大于真实的统计数据,使得在运行 KSS 时不会跳过某些数据源;而基于查询的分层抽样方法由于仅选择一部分的层次,估计的覆盖率和率低于真实的数值,使得 KSS 容易跳过某些数据源,最终,总体的查询结果误差较大.

Table 7 Query execution time of KSS and OASIS on TPC-W dataset

表 7 TPC-W 数据集上 KSS 和 OASIS 方法的查询执行时间

查询选择度		低选择度查询	中选择度查询	高选择度查询
OASIS	5%	253 592.6	316 017.3	331 485
	7.5%	260 713.6	315 760.1	337 465.6
	10%	287 647.3	325 682.9	332 741.7
	12.5%	284 816.7	340 537	324 426.4
KSS (K=1)	5%	6 607.857	9 729.429	18 773.44
	7.5%	6 554.571	9 697.429	18 623.89
	10%	7 030	9 311.143	19 084.44
	12.5%	6 929.143	9 718.714	19 221.11
Optimal	5%	5 661.429	8 161.429	18 236.78
	7.5%	5 661.429	8 161.429	18 236.78
	10%	5 661.429	8 161.429	18 236.78
	12.5%	5 661.429	8 161.429	18 236.78

Table 8 Query execution time of KSS on different queries on TPC-W dataset

表 8 TPC-W 数据集上不同查询运行 KSS 时的执行时间

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	34 671.29	71 912.14	162 678.4
	7.5%	38 468.29	73 130.71	162 769.1
	10%	41 501.29	77 760.71	167 641.4
	12.5%	43 698.43	79 659.29	171 940.3
基于查询的分层抽样方法	5%	9 278.429	10 070.86	16 490.44
	7.5%	7 242.571	6 983.143	10 938.67
	10%	6 835.571	6 432.429	10 402.89
	12.5%	5 812.857	6 096.286	9 098.667
基于元组的分层抽样方法	5%	35 960.86	70 863.71	158 403.7
	7.5%	40 361.29	76 017.43	161 749.9
	10%	44 361.14	77 909.86	168 027.7
	12.5%	45 536.57	80 104.57	166 303

Table 9 Relative error of KSS on different queries on TPC-W dataset

表 9 TPC-W 数据集上不同查询运行 KSS 时的查询基数误差

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	0.030 867	0.004 25	0.001 617
	7.5%	0.021 072	0.003 217	0.001 925
	10%	0.012 055	0.001 659	0.001 054
	12.5%	0.005 192	0.001 866	0.000 4
基于查询的分层抽样方法	5%	0.523 806	0.625 136	0.626 861
	7.5%	0.586 263	0.673 057	0.675 228
	10%	0.631 447	0.691 349	0.683 217
	12.5%	0.639 523	0.690 941	0.698 566
基于元组的分层抽样方法	5%	0.010 907	0.004 722	0.002 437
	7.5%	0.008 058	0.002 577	0.001 715
	10%	0.005 085	0.001 718	0.000 868
	12.5%	0.003 855	0.001 492	0.000 888

4.3 Abebooks真实数据集

本节我们分析 Abebooks 真实数据集上的实验结果.和 TPC-W 数据集一样,我们对每个数据源分别用不同的抽样方法抽取 5%,7.5%,10%,12.5%的元组作为样本数据.

4.3.1 统计数据精度对比

表 10 和表 11 分别展示了 Abebooks 数据集上不同抽样方法随样本数量变化时,不同选择度查询的覆盖率和重叠率的平均相对误差.

Table 10 Relative error of coverage of queries with different selectivities on Abebooks dataset

表 10 Abebooks 数据集上不同选择度查询对应的覆盖率的相对误差

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	0.826 846	0.481 515	0.252 412
	7.5%	0.803 942	0.454 967	0.230 271
	10%	0.789 227	0.442 356	0.230 192
	12.5%	0.773 76	0.436 392	0.218 91
基于查询的分层抽样方法	5%	0.441 735	0.389 076	0.194 656
	7.5%	0.372 306	0.278 748	0.154 656
	10%	0.397 257	0.226 432	0.134 656
	12.5%	0.255 714	0.184 253	0.114 568
基于元组的分层抽样方法	5%	0.055 787	0.024 493	0.013 328
	7.5%	0.045 263	0.025 425	0.013 958
	10%	0.043 003	0.020 124	0.006 325
	12.5%	0.012 239	0.004 181	0.002 898

Table 11 Relative error of overlap of queries with different selectivity on Abebooks dataset

表 11 Abebooks 数据集上不同选择度查询对应的重叠率的相对误差

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	0.878 542	0.712 195	0.543891
	7.5%	0.864 689	0.709 524	0.534 16
	10%	0.863 809	0.700 421	0.490 764
	12.5%	0.863 781	0.654 398	0.389 761
基于查询的分层抽样方法	5%	0.561 268	0.321 264	0.256 3
	7.5%	0.457 544	0.199 354	0.235 451
	10%	0.355 44	0.172 109	0.148 481
	12.5%	0.284 269	0.164 393	0.101 852
基于元组的分层抽样方法	5%	0.085 347 8	0.060 957 9	0.038 541 7
	7.5%	0.085 220 3	0.058 551	0.036 898 1
	10%	0.081 357 7	0.058 507 4	0.034 388 2
	12.5%	0.073 432 8	0.058 142 3	0.032 420 6

表 10 的实验结果展示了与 TPC-W 数据集上相似的结果.

(1) 本文提出的基于元组的分层抽样方法对应的查询覆盖率误差最小,简单随机抽样方法估计的覆盖率误差最大,但简单随机抽样的误差要远小于 TPC-W 数据及上的误差.其原因在于,真实数据集上数据源的元组数量较少,因此简单随机抽样相对来说趋近于基于查询的抽样.

(2) 3 种抽样方法的覆盖率误差都会随着样本数量的增加而减小,覆盖率误差均随着查询选择度的增加而减小.

表 11 的实验结果也容易看到,基于元组的分层抽样方法对应的查询覆盖率误差最小.其原因在于,样本数量增加,查询选择度增加,使得满足查询的样本元组越多,估计的覆盖率就越准确.

表 12 展示的是在 Abebooks 数据集上少量(10 个)数据源时,最大熵方法和分层抽样方法估计的重叠率的相对误差.从实验结果来看,也展示了与 TPC-W 数据集上相似的性质:(1) 最大熵方法估计的重叠率的误差远大于基于元组分层抽样方法估计的重叠率的误差;(2) 基于分层抽样估计的相对误差随着查询选择度和样本数量的增加而减小.

表 13 展示了不同抽样方法生成相同比例样本数据所需的查询数量:基于查询的分层抽样方法需要的查询数量最少,基于元组的分层抽样方法其次,而简单随机抽样方法最多.其原因在于,简单随机抽样方法由于需要

不断“下钻”,因此需要更多的查询,而基于查询的方法仅需确定层次属性值对应的查询,单个数据源仅需较少的查询来访问数据量少的数据源.

Table 12 Relative error of overlap of queries with different selectivities on Abebooks dataset (10 data sources)

表 12 Abebooks 数据集上不同选择度查询对应的重叠率的相对误差(10 个数据源)

查询选择度		低选择度查询	中选择度查询	高选择度查询
最大熵方法	5%	0.380 952	0.342 857	0.226 19
	7.5%	0.380 952	0.295 238	0.291 667
	10%	0.380 952	0.295 238	0.247 024
	12.5%	0.380 952	0.295 238	0.247 024
基于元组的分层抽样方法	5%	0.083 411	0.055 112	0.032 917
	7.5%	0.083 411	0.033 251	0.021 615
	10%	0.067 53	0.035 868	0.017 463
	12.5%	0.067 53	0.041 104	0.015 578

Table 13 Query number required by sampling given number of tuples on Abebooks dataset

表 13 Abebooks 数据集上抽取给定数量元组所需的查询数量

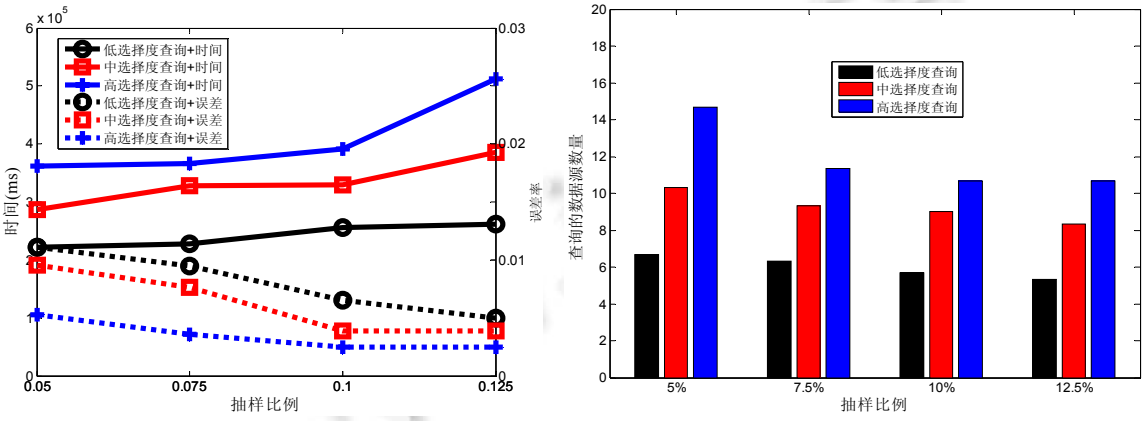
抽样方法	5%	7.5%	10%	12.5%
简单随机抽样	33 865	33 865	33 865	33 865
基于元组的分层抽样	19 227	21 786	23 429	24 096
基于查询的分层抽样	15 623	17 027	17 990	18 663

4.3.2 查询执行时间对比

图 12(a)和图 12(b)分别对应不同选择度查询和不同样本下 OSS 算法的效率、查询基数误差和查询的数据源数量.我们可以看到,查询选择度越高,(1) 查询执行时间增大,查询结果的相对误差越小,选择度越高的查询结果误差越小(如图 12(a)所示);(2) 查询的数据源数量在不断减少并逐渐趋于稳定,选择度高的查询的数据源数量越多(如图 12(b)所示).

图 12(c)和图 12(d)分别对应不同选择度查询和不同样本下,KSS 算法在不同 k 值下所对应的查询执行时间、查询结果误差和查询数据源数量.从图 12(c)可以看出,随着 k 值的增大,不同选择度查询展现了与合成数据集上不同的性质:(1) 随着 K 值的增加,查询时间缓慢增长;(2) 查询结果误差不断减小,并趋近于稳定;(3) 查询的数据源数量在缓慢减少.其原因在于,从 OSS 中我们发现,在 Abebooks 上,对于任意查询,我们仅需少量数据源(不超过 20)即可查询到所有的结果(如图 12(d)所示),因此,我们不需要去查询后面那些覆盖率低且增益少的数据源.

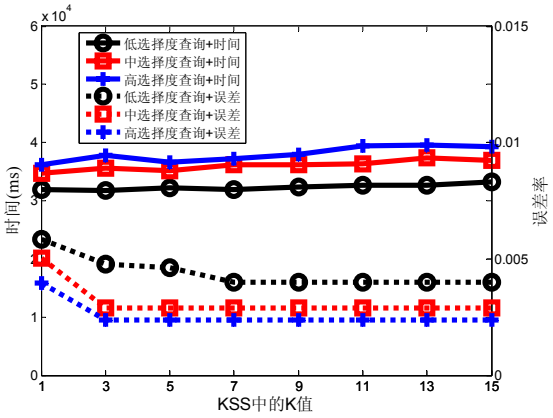
综合图 12 中 OSS 和 KSS 的分析,KSS 的执行时间和查询基数误差上均优于 OSS.而 KSS 在 k=3 时对应的误差率和查询执行时间均为最优.下面我们用使用 KSS 算法(k=3)与现有方法进行比较.



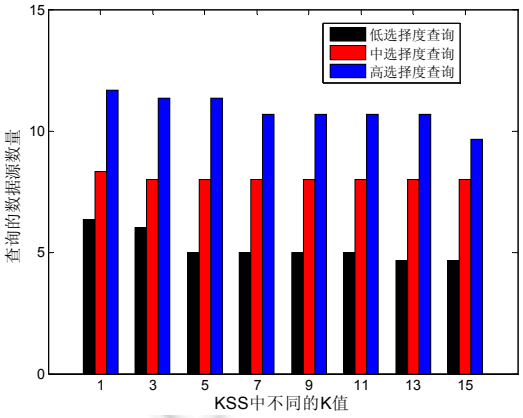
(a) 低选择度查询 (b) 中选择度查询

Fig.12 Algorithm analysis of OSS and KSS on Abebooks dataset

图 12 OSS 和 KSS 在 Abebooks 数据集上的算法分析



(c) 高选择度查询

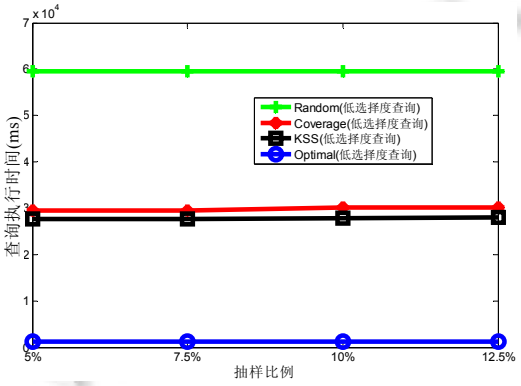


(d) KSS 对不同查询在不同样本上的结果误差的影响

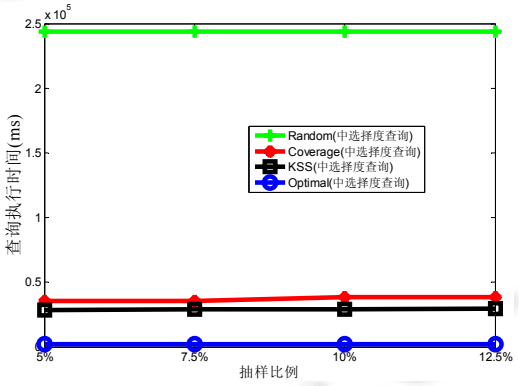
Fig.12 Algorithm analysis of OSS and KSS on Abebooks dataset (Contiuned)

图 12 OSS 和 KSS 在 Abebooks 数据集上的算法分析(续)

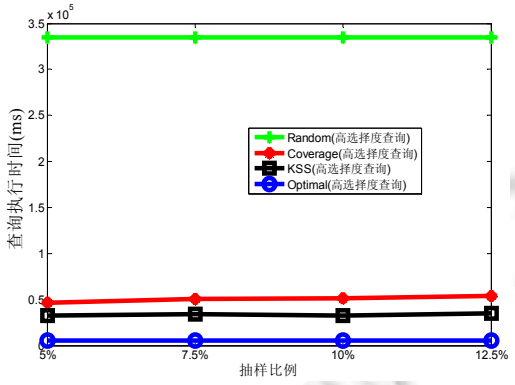
图 13 展示的是 Abebooks 数据集上 KSS 和其他不同的数据源选择算法在查询时间、误差上的分析.



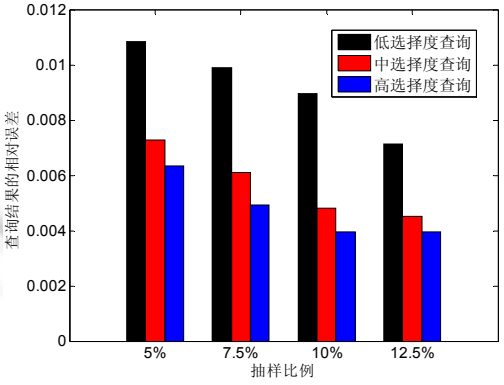
(a) 低选择度查询



(b) 中选择度查询



(c) 高选择度查询



(d) KSS 对不同查询在不同样本上的结果误差的影响

Fig.13 Performance of selection algorithm with different queries on Abebooks dataset

图 13 Abebooks 数据集上不同选择算法在不同查询上的性能

从图 13(a)~图 13(c)不难看出:

(1) KSS 的执行时间高于 Optimal,低于 Coverage 和 Random;同时,Coverage 和 KSS 算法的执行时间随着样本比例增加而缓慢增加,其原因在于需要查询样本的数量相对增加.

(2) KSS 略高于 Coverage 的执行时间,其原因在于 KSS 能够利用数据源间的重叠来减少查询开销.

图 13(d)展示的是不同选择度查询对应的查询结果误差.随着样本数量和查询选择度的增加,查询结果的误差越来越小,大部分的误差都在 1.2%以下.

表 14 分析了仅有 10 个数据源时,KSS(K=3)和 OASIS 算法在查询效率上的对比.从表 14 中不难看出:

- 估计数据源重叠变量的时间占据了 OASIS 的主要时间,平均每次估计耗时 14 837ms,远大于 KSS 查询数据源的时间.
- 同时也可以看出,KSS 算法近似于 Optimal 算法的运行时间.

Table 14 Query execution time of KSS and OASIS on Abebooks dataset

表 14 Abebooks 数据集上 KSS 和 OASIS 方法的查询执行时间

查询选择度		低选择度查询	中选择度查询	高选择度查询
OASIS	5%	95 947.83	280 216	318 307.2
	7.5%	83 249	293 589.5	306 373.2
	10%	110 985.8	277 415	305 481.5
	12.5%	103 270.7	273 098	290 945.8
KSS (K=3)	5%	1 421.167	3 081	4 368.5
	7.5%	1 421.167	3 081	4 368.5
	10%	1 421.167	3 081	4 368.5
	12.5%	1 421.167	3 081	4 368.5
Optimal	5%	1 278.5	2 129	2 865.833
	7.5%	1 338.5	2 613.5	3 791.833
	10%	1 924.5	2 917	3 808.667
	12.5%	1 961.5	2 718.5	4 668

表 15 和表 16 分别分析了 KSS 在不同抽样方法下对应的查询执行时间和查询结果数量误差.

从表 15 不难看出:

- 由于数据源的数据量少,同时查询所有数据源所需要的数据源数量较少,使得这 3 种方法的查询执行时间近似.
- 但由于分层抽样方法更高的覆盖率和重叠率的估计精度,使得 KSS 的误差相对较小,同时查询的执行时间更长.

和 TPW-W 数据集类似,表 16 展示了 Abebooks 上不同抽样方法生成相同比例样本数据所需的查询数量:基于查询的分层抽样方法需要的查询数量最少,基于元组的分层抽样方法其次,而简单随机抽样方法最多.

Table 15 Query execution time of KSS on different queries on Abebooks dataset

表 15 Abebooks 数据集上不同查询运行 KSS 时的执行时间

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	30 281	31 150	32 470
	7.5%	29 741.17	32 104.33	34 317
	10%	29 708.17	32 779.33	34 650.6
	12.5%	32 453.83	33 217	39 421.6
基于查询的分层抽样方法	5%	30 452.83	31 650	33 237.2
	7.5%	29 959.33	31 917.33	33 958.8
	10%	29 449.67	31 422.67	35 271.4
	12.5%	29 631.33	31 584.67	35 478.2
基于元组的分层抽样方法	5%	42 473.17	38 070.33	35 383.4
	7.5%	31 682	34 195	36 648.2
	10%	30 783.167	32 911	33 724.4
	12.5%	29 511	32 074	35 082.6

Table 16 Relative error of KSS on different queries on Abebooks dataset

表 16 Abebooks 数据集上不同查询运行 KSS 时的查询基数误差

查询选择度		低选择度查询	中选择度查询	高选择度查询
简单随机抽样方法	5%	0.470 101	0.038 296	0.008 351
	7.5%	0.429 647	0.031 398	0.009 628
	10%	0.410 55	0.028 474	0.008 904
	12.5%	0.390 713	0.023 805	0.007 432
基于查询的分层抽样方法	5%	0.582 727	0.237 965	0.010 606
	7.5%	0.547 009	0.146 22	0.010 605
	10%	0.493 445	0.135 195	0.009 157
	12.5%	0.482 772	0.101 245	0.006 745
基于元组的分层抽样方法	5%	0.105 864	0.037 965	0.009 824
	7.5%	0.098 654	0.024 322	0.008 608
	10%	0.093 445 2	0.024 168 9	0.007 628 3
	12.5%	0.087 980 7	0.022 423 7	0.006 412 7

综上所述,基于元组的抽样方法尽管抽样代价趋近于基于查询的抽样方法,但能保证较高的覆盖率和重叠率的估计误差;虽然 OASIS 能够伸缩性地解决多重叠数据源排序问题,但是在深网数据源的环境下,我们提出的 KSS 算法能够得到更高的效率和更小的误差.

5 总结与展望

在海量深网数据源查询的过程中,探索和利用数据源之间的重叠能够极大提高深网查询的效率.本文提出一种元组水平的分层抽样方法来估计和利用查询在数据源上的统计数据,选择高相关、低重叠的数据源.该方法分为两个阶段:离线阶段,我们基于元组水平对数据源进行分层抽样,获得样本数据;在线阶段,我们基于样本数据迭代地估计查询在数据源上的覆盖率和重叠率,并采用一种启发式策略以高效地发现低重叠的数据源.本文在基准数据集和真实数据集上进行实验,结果表明,在能够保证用户精度需求的同时,与传统方法相比,能够显著提高效率.

References:

[1] Bergman MK. White paper: The deep Web: Surfacing hidden value. Journal of Electronic Publishing, 2001,7(1). [doi: 10.3998/3336451.0007.104]

[2] Dalvi N, Machanavajjhala A, Pang B. An analysis of structured data on the Web. Proc. of the VLDB Endowment, 2012,5(7): 680−691. [doi: 10.14778/2180912.2180920]

[3] Dasgupta A, Jin X, Jewell B, Zhang N, Das G. Unbiased estimation of size and other aggregates over hidden Web databases. In: Elmagarmid AK, Agrawal D, eds. Proc. of the 2010 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2010). New York: ACM Press, 2010. 855−866. [doi: 10.1145/1807167.1807259]

[4] Wan CX, Deng S, Liu XP, Liao GQ, Liu DX, Jiang TJ. Web data source selection technologies. Ruan Jian Xue Bao/Journal of Software, 2013,24(4):781−797 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4374.htm [doi: 10.3724/SP.J.1001.2013.04374]

[5] Florescu D, Koller D, Levy A. Using probabilistic information in data integration. In: Jarke M, Carey MJ, Dittrich KD, Lochovsky F, Loucopoulos P, Jeusfeld MA, eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases (VLDB'97). Athens: Morgan Kaufmann Publishers, 1997. 25−29.

[6] Nie Z, Kambhampati S, Nambiar U. Effectively mining and using coverage and overlap statistics for data integration. IEEE Trans. on Knowledge and Data Engineering, 2005,17(5):638−651. [doi: 10.1109/TKDE.2005.76]

[7] Salloumj M, Dong X, Srivastava D, Tsotras V. Online ordering of overlapping data sources. Proc. of the VLDB Endowment, 2014, 7(3):133−144. [doi: 10.14778/2732232.2732233]

[8] Cormode G, Garofalakis M, Haas P, Jermaine C. Synopses for massive data: Samples, histograms, wavelets, sketches. Foundations and Trends in Databases, 2012,4(1-3):1−294. [doi: 10.1561/1900000004]

[9] Dasgupta A, Das G, Mannila H. A random walk approach to sampling hidden databases. In: Chan CY, Ooi BC, Zhou AY, eds. Proc.of the 2007 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2007). New York: ACM Press, 2007. 629−640. [doi: 10.1145/1247480.1247550]

[10] Liu W, Thirumuruganathan S, Zhang N, Das G. Aggregate estimation over dynamic hidden Web databases. Proc. of the VLDB Endowment, 2014,7(12):1107−1118. [doi: 10.14778/2732977.2732985]

[11] Liu T, Wang F, Agrawal G. Stratified sampling for data mining on the deep Web. Frontiers of Computer Science, 2012,6(2): 179−196. [doi: 10.1007/s11704-012-2859-3]

[12] Liu T, Agrawal G. Stratified *k*-means clustering over a deep Web data source. In: Yang Q, Agarwal D, Pei J, eds. Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2012). Beijing: ACM Press, 2012. 1113−1121. [doi: 10.1145/2339530.2339705]

[13] Arguello J, Diaz F, Callan J, Crespo JF. Sources of evidence for vertical selection. In: Allan J, Aslam JA, Mark Sanderson M, Zhai CX, Zobel J, eds. Proc. of the 32nd Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Boston: ACM Press, 2009. 315−322. [doi: 10.1145/1571941.1571997]

[14] Arguello J, Callan J, Diaz F. Classification-Based resource selection. In: Cheung D, Song I, Chu W, Hu XH, Lin J, eds. Proc. of the 18th ACM Conf. on Information and Knowledge Management (CIKM 2009). Hong Kong: ACM Press, 2009. 1277−1286. [doi: 10.1145/1645953.1646115]

[15] Shokouhi M, Zobel J. Federated text retrieval from uncooperative overlapped collections. In: Kraaij W, Vries A, Clarke C, Fuhr N, Kando N, eds. Proc. of the 30th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. Amsterdam: ACM Press, 2007. 495−502. [doi: 10.1145/1277741.1277827]

[16] Gravano L, Garcia-Molina H, Tomasic A. GIOSS: Text-Source discovery over the Internet. ACM Trans. on Database Systems, 1999,24(2):229−264. [doi: 10.1145/320248.320252]

[17] Rekatsinas T, Dong XL, Getoor L, Srivastava D. Finding quality in quantity: The challenge of discovering valuable sources for integration. In: Kossmann D, Ailamaki A, Stonebrake M, eds. Proc. of the 7th Biennial Conf. on Innovative Data Systems Research (CIDR). Asilomar, 2015.

[18] Rekatsinas T, Dong XL, Srivastava D. Characterizing and selecting fresh data sources. In: Dyreson CE, Li FF, Özsu MT, eds. Proc. of the 2014 ACM SIGMOD Int'l Conf. on Management of data (SIGMOD 2014). Snowbird: ACM Press, 2014. 919−930. [doi: 10.1145/2588555.2610504]

[19] Balakrishnan R, Kambhampati S. SourceRank: Relevance and trust assessment for deep Web sources based on inter-source agreement. In: Srinivasan S, Ramamritham K, Kumar A, Ravindra MP, Bertino E, Kumar R, eds. Proc. of the 20th Int'l Conf. on World Wide Web (WWW 2011). New York: ACM Press, 2011. 227−236. [doi: 10.1145/1963405.1963440]

[20] Dong XL, Saha B, Srivastava D. Less is more: Selecting sources wisely for integration. Proc. of the VLDB Endowment, 2012,6(2): 37−48. [doi: 10.14778/2535568.2448938]

[21] Doan AH, Halevy A, Ives Z. Principles of Data Integration. Waltham: Elsevier, 2012.

[22] Vazirani VV. Approximation Algorithms. Berlin: Springer-Verlag, 2003. 15−26. [doi: 10.1007/978-3-662-04565-7]

[23] Chvatal V. A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 1979,4(3):233−235. [doi: 10.1287/moor.4.3.233]

[24] Lohr S. Sampling: Design and Analysis. 2nd ed., Boston: Nelson Education, 2009. 73−115.

[25] Jin X, Zhang N, Das G. Attribute domain discovery for hidden Web databases. In: Sellis TK, Miller RJ, Kementsietsidis A, Velegrakis Y, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011). Athens: ACM Press, 2011. 553−564. [doi: 10.1145/1989323.1989381]

[26] Vitter JS. Random sampling with a reservoir. ACM Trans. on Mathematical Software, 1985,11(1):37−57. [doi: 10.1145/3147.3165]

[27] Malik T, Burns R, Chawla N. A black-box approach to query cardinality estimation. In: Hellerstein J, Stonebraker M, Weikum G, eds. Proc. of the 3rd Biennial Conf. on Innovative Data Systems Research (CIDR). Asilomar, 2007. 56−67.

[28] TPC-W benchmark. 2000. http://www.tpc.org/tpcw/

[29] Abebooks dataset. 2007. http://www.lunadong.com/fusionDataSets.htm

附中文参考文献:

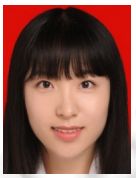
[4] 万常选,邓松,刘喜平,廖国琼,刘德喜,江腾蛟.Web 数据源选择技术.软件学报,2013,24(4):781−797. http://www.jos.org.cn/1000-9825/4374.htm [doi: 10.3724/SP.J.1001.2013.04374]

江俊彦(1987－),男,湖北黄冈人,博士生,主要研究领域为 Web 数据管理,查询调度,数据集成.

彭智勇(1963－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为复杂数据管理,可信数据管理,Web 数据管理.

吴小莹(1973－),女,博士,副教授,博士生导师,CCF 专业会员,主要研究领域为数据管理,数据查询处理和优化,关键字查询,模式挖掘,语义网,数据集成.

彭承晨(1994－),男,硕士生,主要研究领域为 Web 数据管理.



王敏(1992－),女,硕士生,主要研究领域为 Web 数据管理.