基于条件偏好的XML查询结果排序方法

孟祥福 张霄雁 彭晏飞 朱旭光
(辽宁工程技术大学电子与信息工程学院 葫芦岛 125105)

摘 要 为了解决XML查询的信息过载问题,提出了基于条件偏好的XML多查询结果排序方法。该方法把用户指定的内容查询谓词作为上下文条件,然后在原始XML数据和查询历史上利用概率信息检索模型推测当前用户偏好,评估结果元素中被查询指定的属性单元值与未指定的属性单元值之间的关联关系以及未指定的属性单元值与用户偏好之间的相关程度,进而构建查询结果元素打分函数;在此基础上,利用打分函数计算结果元素的排序分值,并以此对查询结果进行排序。实验结果表明,提出的排序方法具有较高的排序准确性,能够较好地满足用户需求和偏好。

关键词 XML,属性单元,条件偏好,概率信息检索模型,排序
中图法分类号 TP391 **文献标识码** A

Contextual Preferences-based Approach for XML Query Results Ranking

MENG Xiang-fu ZHANG Xiao-yan PENG Yan-fei ZHU Xu-guang
(College of Electronic and Information Engineering, Liaoning Technical University, Huludao 125105, China)

Abstract To deal with the problem of information overload for XML data queries, this paper proposed a contextual preferences-based XML query results ranking approach. The value-based query predicates user specified are treated as the context, and then the probabilistic information model is used on the dataset and query history to speculate the user preferences, estimate the correlations between the specified and unspecified attribute unit values and the relevance between the unspecified attribute unit values and user preferences as well. Next, scoring function of query results is constructed and the ranking score which is computed by the scoring function is used to rank the query results. Results of experiments showed that the ranking approach proposed has high ranking accuracy and can meet the user needs and preferences as well.

Keywords XML, Attribute unit, Contextual preference, Probabilistic information retrieval model, Ranking

1 引言

随着WWW的迅速膨胀和Internet的普遍应用,访问Web已成为人们获取信息的重要手段。XML规范是当今Web上信息表示与交换的标准,大量的异构数据因而也就集成在XML数据库中[1]。对于蕴含海量信息的XML数据库来说,在用户查询要求不是很严格的情况下,通常会返回大量的查询结果,也就是"信息过载"问题。如果把这些信息随机返回给用户,那么用户将不得不对大量的查询结果元素逐一对比分析,这是一件枯燥并且费时费力的工作。如果能把返回的XML结果元素按其对查询要求和用户偏好的相关度进行排序,将会把用户从繁重枯燥的对比分析任务中解脱出来,因此对XML查询结果排序研究有着重要的意义。

在信息检索领域,由于Web搜索结果经常出现信息过载现象,因此研究者们对Web查询结果的排序已经展开了较为深入的研究,其中向量空间模型[2,3]、概率信息检索模型[4,5]和统计语言模型[2,6]均得以成功的应用。在关系数据库领域,查询结果排序方法研究可分成4类:第一类是根据单个属性(如日期或价格等)对查询结果进行升序或降序排列,然而在实际应用中大多数用户可能会同时考虑多个属性来做出综合评判;第二类方法是基于用户相关反馈的排序方法[7,8];第三类是基于用户明确指定偏好(如构建用户偏好描述文件)的排序方法[9,10],然而现实中很多用户不能或不愿明确给出自己的偏好;第四类是通过挖掘查询历史来推测用户隐含偏好从而实现自动排序[11,12],本文基于该思想提出了一种适用于XML查询结果的自动排序方法。

近年来,随着XML的广泛应用,XML查询结果排序方法的研究开始受到关注[13-20]。文献[13]利用语义本体计算查询结果元素内容与XPath内容查询谓词之间的语义相似度,并以此对查询结果进行排序。文献[14,15]提出了基于 $tf*idf$ 思想的XML查询结果排序方法,对于一个XPath查询,先将其分解成若干查询片段,然后利用tf * idf思想计算XML文档中满足每个查询片段的节点的排序分数,最后合并满足所有查询片段并且具有前 K 个最高排序分数的节点作

本文受国家青年科学基金项目(61003162)资助。
孟祥福(1981—),男,博士,讲师,主要研究方向为Web数据库和XML个性化柔性查询技术,E-mail:marxi@126.com;**张霄雁**(1983—),女,硕士,助教,主要研究方向为Web数据库与XML查询优化技术;**彭晏飞**(1975—),男,硕士,副教授,主要研究方向为XML数据查询与排序技术;**朱旭光**(1980—),男,硕士,讲师,主要研究方向为XML关键字近似查询技术。

为查询结果，计算 $tf*idf$ 值的同时考虑了 XPath 查询的路径和内容。文献[16]提出了一种基于边界阈值的 XML 数据 Top-K 查询方法，该方法利用 XML 的模式信息来评估结果元素与 XPath 查询之间的相关性，并利用边界阈值使相关查询结果尽早返回。文献[17-20]提出了基于关键字查询的 Top-K 排序方法，该类方法首先计算出 XML 文档中所有包含关键字的 XML 元素的最小共同祖先(LCAs)，然后识别出以 LCAs 为根的子树并将其作为查询结果，最后查询结果元素按其是否直接包含关键字作为评分依据排序返回。然而，上述排序方法大多考虑的是查询结果在结构和内容上与查询要求之间的相似性，很少考虑用户偏好，而实际上用户偏好在很大程度上影响着查询结果的排序效果，并且用户偏好与上下文条件密切相关，在不同的上下文条件下用户偏好通常有所不同。因此，本文研究基于条件偏好的 XML 数据查询结果排序方法。

本文第 2 节介绍 XML 查询相关定义；第 3 节提出 XML 查询结果排序方法；第 4 节给出查询结果排序的实现方法；第 5 节描述实验并给出实验结果；最后是总结和展望。

2 XML 查询相关定义

定义 1(XML 数据树) 由 XML 文档按一定映射关系转化成的无向多叉树称为 XML 数据树，简称数据树，用一个四元组 $T=(r_T,N,E,R)$ 表示，其中，r_T 表示数据树的根节点，N 表示树中的所有节点集合，E 表示所有边的集合，R 代表结构关系的集合[21]。

定义 2(查询树) 由 XML 查询语句转化而成的无向多叉树称为查询模式树，简称查询树，用一个四元组 $Q=(r_Q,V,E,R)$ 表示，其中 r_Q 表示查询树的根节点，V 表示查询树中的所有节点集合，E 表示所有边的集合，R 代表结构关系的集合。查询树中节点之间的边有两种结构关系：父子关系和祖先后代关系。查询树中的根节点指明了查询的范围，叶节点描述了用户所要查找的具体信息，中间节点指明了具体信息所在的上下文。

定义 3(小枝模式查询) 一个小枝模式查询 t_Q 定义为 $t_Q=(r_t,V_t,E_t,R_t)$，它是一棵查询树；E_t 分为两个互不相交的集合，分别表示所有孩子边(/)和后裔边(//)。

给定数据树 T 和小枝模式查询 t_Q，t_Q 在 T 中的一个匹配是指从小枝模式中节点到数据树中元素的一个映射，并且：

(1) 保持节点类型一致，即 t_Q 中节点与映射的树中节点类型一致。

(2) 保持节点关系一致，即对小枝查询中的任意两个节点，它们对应的映射节点间的关系和其在查询树中的关系一致[22]。

定义 4(查询匹配) 查询 Q 在 XML 数据树 T 中的一个查询匹配 M 为一个 n 元组 $(m_1,m_2,\cdots,m_n)$，且查询匹配 M 中的元素与查询树中节点的标签类型以及节点间的结构关系相匹配，其中 m_i 是 XML 数据树中的元素节点，n 为查询树 Q 中的节点个数。

定义 5(属性单元) XML 文档中的属性节点和叶子节点统称为属性单元。属性节点在解析的过程中被记录下来，对应每个属性节点的属性单元的名字是该属性节点的名字，值是该属性节点的取值。如果不同元素的属性节点名称相同，那么属性单元的名称的形式为〈元素名称-属性节点名称〉。叶子节点在解析的过程中被记录下来，对应每个叶子节点的属性单元的名字是该叶子节点的名字，值是该叶子节点的取值。

定义 6(查询历史) 查询历史是用户以往提出的查询条件的集合，能够在很大程度上反映出用户对某(些)项内容的兴趣偏好。查询历史可表示为 $W=\{(U_1,Q_1,T_1),\cdots,(U_k,Q_k,T_k)\}$，其中 U_i 代表 SessionID，Q_i 代表查询历史中第 i 条查询记录，T_i 代表第 i 条查询记录的提交时间。

传统的用 XPath 表示的小枝模式查询形式由路径(paths)和基于内容的查询谓词(value-based/content predicates)两部分构成。查询谓词的形式为 $A\theta Y$，其中 A 代表属性单元，θ 代表规则关系(如=，>，<，≥，≤，≠，(not) between)，Y 代表操作数(用户查询要求中指定的查询值)。在实际应用中，由于用户偏好通常与其提出的基于内容的查询谓词密切关系，因此本文主要研究如何根据查询历史和用户提出的查询谓词来推测隐式用户偏好，并将其用于查询结果的排序过程中。

3 XML 查询结果排序方法

本节首先介绍概率信息检索模型(Probabilistic Information Retrieval，PIR)，然后提出基于 PIR 模型的 XML 查询结果元素相关性评估方法。

3.1 概率信息检索模型

在信息检索领域，概率信息检索模型被广泛用于文档与查询之间的相关性评估。为了评估 XML 查询结果元素与用户查询和偏好之间的相关度，本文在概率信息检索模型(PIR，Probabilistic Information Retrieval)基础上进行改进，使其适用于 XML 数据下的相关性评估。概率信息检索利用了概率论中如下两个基本公式：

贝叶斯公式：$p(a|b)=\dfrac{p(b|a)p(a)}{p(b)}$

推论：$p(a,b|c)=p(a|c)p(b|a,c)$

在信息检索中，考虑一个文档集合 T，对于一个给定的查询 Q，令 R 代表相关的文档集合，$\bar{R}=T-R$ 代表不相关的文档集合。为了对 T 中的任一文档 t 进行相关性评估，需要根据文档 t 的特征信息(如 t 中词条的出现频率等)计算 t 对查询 Q 的相关性概率，即 $p(R|t)$。在概率信息检索中，文档按照它们与查询之间的相关性降序排列，其中，文档与查询之间的相关性(也可看成是 t 的排序分值)被定义为：

$$Score(t)=\frac{p(R|t)}{p(\bar{R}|t)}=\frac{\dfrac{p(t|R)p(R)}{p(t)}}{\dfrac{p(t|\bar{R})p(\bar{R})}{p(t)}}\propto\frac{p(t|R)}{p(t|\bar{R})} \tag{1}$$

式中，$Score(.)$ 为文档 t 的相关性打分函数。通常情况下，对于每一个文档 t 来说，$p(R)$ 和 $p(\bar{R})$ 均是相等的，所以相当于两个常量，而不影响排序函数的最终结果。现在的问题是，如何在 R 和 $\bar{R}$ 是未知的(因为在查询期间 R 和 $\bar{R}$ 是未知的)情况下计算这些概率。在信息检索中，解决这一问题的常用技术有：通过用户相关反馈估计 R，用 T 近似代表 $\bar{R}$(因为 R 通常要比 T 小很多，所以 $\bar{R}$ 与 T 比较接近)[23]。

3.2 基于 PIR 模型的 XML 元素相关性评估

为了使 PIR 模型能够适用于 XML 结果元素的相关度打

分，本文把 XML 数据树 T 中的每个被查询元素 t 看成是一个文档。然后，对于一个给定的查询 Q，目的是为每个元素 t 获取它的排序分值 $Score(t)$，最后根据排序分值对元素进行排序。

3.2.1 XML 数据下的 PIR 模型

给定 XML 小枝模式查询 Q，令 X 是 XML 数据树中被查询指定的属性单元集合，Y 是 XML 数据树中未被查询指定的属性单元集合，于是被查询元素 t 可由两个部分表示，即 $t(X)$ 和 $t(Y)$，其中，$t(X)$ 是元素 t 中对应于属性单元集 X 上的值的集合，$t(Y)$ 是元素 t 中对应于属性单元集 Y 上的值的集合。用 X 和 Y 代替 t，用 T 代替 $\bar{R}$（如上所述，相对于 T 来讲 R 只是很小的一部分，因此可用 T 近似替代 $\bar{R}$），则式(1)可改写为：

$$Score(t)\propto\frac{p(t|R)}{p(t|T)}=\frac{p(X,Y|R)}{p(X,Y|T)}=\frac{p(Y|R)}{p(Y|T)}\cdot\frac{p(X|Y,R)}{p(X|Y,T)} \tag{2}$$

因为所有结果元素中的 X 都相同（即 $R\subseteq X$），则 $P(X|Y,R)=1$，所以上式可转化为：

$$Score(t)\propto\frac{p(Y|R)}{p(Y|T)}\cdot\frac{1}{p(X|Y,T)} \tag{3}$$

例 1　考虑一个二手车数据集，包括 100000 条二手车记录，利用 IBM XML data generator 生成 CarDB. xml，其结构如图 1 所示。

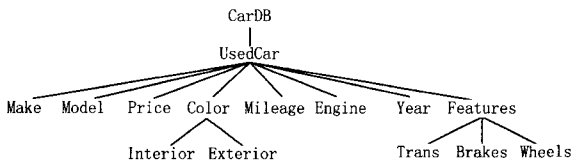

图 1　CarDB 文档结构图

基于 CarDB. xml，考虑一个 XPath 查询“Q:-/CarDB/UsedCar/[Make=Toyota][Price between 10000 and 12000]”。Toyota 是日本著名的丰田汽车制造商，由该公司生产的汽车型号包括锐志-Reiz、凯美瑞-Camry 和卡罗拉-Corolla 等，其中 Reiz 和 Camry 是比较受欢迎的汽车型号，Corolla 次之。因此，提出上述查询的用户可能也希望购买到型号(Model)为 Reiz 或 Camry 的二手车，而 Corolla 可能就不是他们所期望的。所以，p(Model=Reiz|R)并且 p(Model=Camry|R)的值都应该较大，而 p(Model=Corolla|R)的值可能相对较小。进而，如果 T 中可选的 Camry 型号车相对于 Reiz 型号车多，即分母 p(Model=Camry | Make=Toyota ∧ Price between 10000 and 12000, T)大于 p(Model=Reiz | Make=Toyota ∧ (Price between 10000 and 12000, T)，那么根据式(3)得到的最后排序结果将是包含 Reiz 的元素排前面(即 Reiz 的相关性较大)，Camry 次之，Corolla 最后。实际上，式(3)的两个因数分别评估了未被查询指定的属性单元值的全局重要性和条件重要性，即式(3)的第一个因数评估了未被查询指定的属性单元值 Reiz、Camry 和 Corolla 的全局重要性，而第二个因数评估了这 3 个属性单元值与查询谓词“[Make=Toyota][Price between 10000 and 12000]”中指定的属性单元值之间的依赖程度。

3.2.2 有限独立模型

为了进一步转换打分函数，首先采用有限形式的独立性假设来对公式做进一步推导。

有限独立性假设：给定一个小枝模式查询，假设其包含的内容查询谓词是合取形式，即 $\wedge_{j\in(1\cdots k)}(A_j=a_j)$。对于一个元素 t，假设 t 中对应于属性单元集合 X(和 Y)内部的值之间是独立的，但 X 和 Y 之间的值的依赖是允许的。也就是说，在有限独立性假设条件下，$p(X|C)$(或 $p(Y|C)$)可以分别被写成 $\prod_{x\in X}p(x|C)$ 和 $\prod_{y\in Y}p(y|C)$，其中 C 是与 X(或 Y)不相交的集合，即 $C\cap X=\emptyset$(或 $C\cap Y=\emptyset$)，C 中只能包含 T 或 R 上对应于属性单元集合 Y(或 X)中的值。

基于上述假设，可将打分函数进一步转换为：

$$Score(t)\propto\frac{p(Y|R)}{p(Y|T)}\cdot\frac{1}{p(X|Y,T)}=\prod_{y\in Y}\frac{p(y|R)}{p(y|T)}\cdot\prod_{x\in X}\prod_{y\in Y}\frac{1}{p(x|y,T)} \tag{4}$$

3.2.3 消除独立性假设

基于有限独立性假设，式(4)平衡了一些依赖关系，但它的推导仍然需要独立性假设。在 XML 数据中，属性单元值之间的依赖关系对元素的相关性评估有着很重要的影响，因此需要消除独立性假设。注意，虽然 XML 数据属性单元的值之间存在依赖关系[24]，但查询历史中的查询记录之间不存在依赖关系，于是函数依赖只能影响式(4)中的分母，而不会影响分子。基于上述分析，下面介绍独立性假设的消除方法。

(1) 消除 Y 中的独立性假设

考虑 Y 中的函数依赖，假设 $y_i\rightarrow y_j$ 是 Y 中的属性单元 $\langle y_i,y_j\rangle$ 上的函数依赖，这表示对于所有的属性单元值 $\langle a_i,a_j\rangle$，$\{t\mid t.y_i=a_i(t.y_j=a_j\}=\{t\mid t.y_i=a_i\}$ 成立。于是，$p(y_i,y_j|R)$ 可简化成 $p(y_i|R)\ p(y_j|y_i,R)=p(y_i|R)$。通常，式(3)中的 $\frac{1}{p(Y|T)}$ 可简化成 $\prod_{y\in Y'}\frac{1}{p(y|T)}$，其中 $Y'=\{y\in Y\mid\nrightarrow\exists y'\in Y,FD:y'\rightarrow y\}$。

(2) 消除 X 中的独立性假设

函数依赖同样存在于属性单元集合 X 中，则式(3)中的表达式 $\frac{1}{p(X|Y,T)}$ 可以简化为 $\prod_{y\in Y'}\prod_{x\in X'}\frac{1}{p(x|y,T)}$，其中 $X'=\{x\in X\mid\nrightarrow\exists x'\in X,FD:x'\rightarrow x\}$。把上述推导应用于式(4)，则式(4)可改写为(其中 X' 和 Y' 如上定义)：

$$Score(t)\propto\prod_{y\in Y}p(y|R)\prod_{y\in Y'}\frac{1}{p(y|T)}\prod_{y\in Y'}\prod_{x\in X'}\frac{1}{p(x|y,T)} \tag{5}$$

(3) 基于查询历史的 $p(y|R)$ 评估

$p(y|R)$ 的评估需要知道查询结果集 R 的相关信息，但是在执行查询时它是未知的，因此需要考虑使用其他辅助信息对其进行评估。由于查询历史能够在很大程度上反映用户的查询需求和偏好信息，因此本文利用了查询历史来评估 $p(y|R)$。如前所述，查询历史 W 代表一个特殊的“记录”集合，其中的每条“记录”代表一个历史查询，该“记录”包含了以往查询在不同属性单元上指定的值。假设查询 Q 在不同属性单元上指定的值用集合 X 表示，则可把 R 近似看成是所有在 W 中查询过 X 的“记录”集合，因此通过考察 W 中包含查询过 X 的历史查询记录，就能够获得查询结果集 R 的所有(或大部分)特征信息。例如，基于 CarDB. xml，用户需要购买一辆价位在 $10000 左右的“Toyota”二手车，用内容查询谓词表示为“[Make='Toyota'] [Price=10000]”。那么查询历史就有可能反映出这样一些信息：过去许多用户提出的查询要求除了指定生产商(Make)为“Toyota”和价位(Price)在

$10000 左右之外，还指定了车的型号(Model)为“Camry”，即“[Make=‘Toyota’][Price=10000] [Model=‘Camry’]。所以，对于当前提出查询“[Make=‘Toyota’][Price=10000]”的用户来说，他的偏好就很可能是“Model=‘Camry’”，因此系统提供的结果应该是 Model 为 Camry 车的比其他 Model 的车排序要靠前。

于是，对于查询 Q(Q 指定了属性单元集合 X)，用 $p(y \mid X,W)$ 替换 $p(y|R)$，得到公式：

$$\begin{aligned}Score(t) &\propto \prod_{y\in Y}\frac{p(y|X,W)}{p(y|X,T)}\prod_{y\in Y}\prod_{x\in X}\frac{1}{p(x|y,T)}\\ &=\prod_{y\in Y}\frac{\frac{p(X,W|y)p(y)}{p(X,W)}}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}\frac{1}{p(x|y,T)}\\ &\propto\prod_{y\in Y}\frac{p(X,W|y)}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}\frac{1}{p(x|y,T)}\\ &=\prod_{y\in Y}\frac{p(W|y)p(X|W,y)p(y)}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}\frac{1}{p(x|y,T)}\\ &\propto\prod_{y\in Y}\frac{p(y|W)\prod_{x\in X}p(x|y,W)}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}\frac{1}{p(x|y,T)}\\ &\propto\prod_{y\in Y}\frac{p(y|W)}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}\frac{p(x|y,W)}{p(x|y,T)}\end{aligned}\tag{6}$$

最终可以写成：

$$Score(t)\propto\prod_{y\in Y}\frac{p(y|W)}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}\frac{p(x|y,W)}{p(x|y,T)}\tag{7}$$

式(7)是假设不存在函数依赖的前提下得到的元素最终排序分值。考虑函数依赖，可以得到：

$$Score(t)\propto\prod_{y\in Y}p(y|W)\prod_{y\in Y}\frac{1}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}p(x|y,W)\prod_{y\in Y}\prod_{x\in X}\frac{1}{p(x|y,T)}\tag{8}$$

式中，X'和 Y'如式(5)所定义。

3.3 特殊情形讨论

本节给出排序函数在实际应用中需要处理的两个特殊情形：(1) 查询历史缺失的情况；(2) 查询指定的属性单元取值为数值型值。下面分别对这两种特殊情形进行讨论，并给出解决方法。

(1) 查询历史缺失

现在讨论查询历史不可用的情况。考虑式(7)，由于在查询历史缺失情况下，$p(x|W)$对于所有不同的属性单元值 x 都相同，并且对应的 $p(x|y,W)$对于所有不同的 x 和 y 也都相同，因此可将其看成是常数，不影响排序结果，这样式(7)可简化成：

$$\begin{aligned}Score(t)&\propto\frac{1}{p(Y|T)}\cdot\frac{1}{p(X|Y,T)}\\ &=\prod_{y\in Y}\frac{1}{p(y|T)}\prod_{y\in Y}\prod_{x\in X}\frac{1}{p(x|y,T)}\end{aligned}\tag{9}$$

式(9)的含义与信息检索中的 IDF(inverse document frequency，倒排文档频率)思想相似。也就是说，式(9)中第一个因数表示那些在数据集中很少出现的未指定的属性单元值 y 将获得更高的分数；第二个因数表示那些未指定的属性单元值与指定的属性单元值之间函数依赖程度较低的元素将获得更高的分数。例如，如果用户查找低价位的二手车，那么具有较新生产日期的二手车将获得更高的分数，因为同时具有低价位和新生产日期的元素在数据集中并不常见。

(2) 数值型值的处理

前面讨论了属性单元中包含文本型值的处理方法。而在实际应用中，XML 数据树的属性单元取值也可能是数值型值。例如，在 CarDB. xml 数据集中，叶节点 Price 和 Mileage 上的取值包含了数值型值。

本文处理数值型值的方法是利用直方图技术对数值进行离散化处理。该技术把数据分割成若干区间，然后把每个区间视为一个文本型值。数值区间划分算法描述见算法 1。

算法 1 数值区间划分算法

输入：取值为数值型值的属性单元 A_i 和它的取值范围，XML 数据树中属性单元 A_i 的个数$|T|$，待划分的数值区间个数 n

输出：n 个数值区间的上界集合 $U=\{u_1,\cdots,u_n\}$

1. 令 U 为一个大小为 n 的数组，用于存储每个数值区间的上界；
2. 令属性单元 A_i 在 T 上的取值范围为$[a_{low},a_{up})$；
3. 设置 $\lambda=|R|/n$；
4. 令数值区间下界 $low=a_{low}$，上界 $up=a_{up}$；
5. do
6. 用查询谓词“[A_i between low and up]”查询 T 并记录结果元素个数 c；
7. if($c\leqslant\lambda$) then
8. 记录 up 的值并将其按序存入 U 中；
9. 设置 low=up，$up=a_{up}$；
10. else
11. 设置 up=low+(up−low)/2；
12. while($low<a_{up}$)
13. return U

算法 1 首先确定每个数值区间所包含的元素数阈值 λ，该值为 T 中属性单元 A_i 的个数与设定的数值区间数之比(步骤 3)，并设置最初的数值区间下界 low 和上界 up 的值分别为属性单元 A_i 取值范围中的最小值和最大值(步骤 4)。然后，用查询谓词“[A_i between low and up]”查询 T 并记录结果元素个数 c(步骤 6)，如果 c 小于阈值 λ，记录 up 的值并将其存入数组 U 中；否则，缩小查询范围重新查询 R。如此反复直到 c 小于阈值 λ，这样每个数值区间中的元素数都会不超过阈值 λ(步骤 7－步骤 12)。最后，返回数值区间的上界集合 U。利用该算法，可为 T 中每个数值型属性单元划分数值区间。这样，在利用式(7)计算元素排序分数时，可将每个数值区间看成是一个文本值来处理。

4 实现方法

上述讨论的 XML 查询结果排序方法，其实现过程可分为两个处理阶段：预处理阶段和在线排序处理阶段。预处理阶段的主要任务是计算排序函数所需的不同文本值 x 和 y 的原子概率 $p(y|W)$，$p(y|T)$，$p(x|y,W)$和 $p(x|y,T)$，并将计算结果存储在原子概率表中；在线排序处理阶段的主要任务是根据查询条件和属性单元值的原子概率表，再根据排序函数计算出查询结果集中每个元素的排序分数，进而对其进行降序排列。

(1) 原子概率计算

对于文本型值，$p(y|W)$和 $p(y|T)$的值可直接利用统计方法计算得到，$p(x|y,W)$和 $p(x|y,T)$的值可通过在数据集和查询历史中利用关联规则挖掘算法来计算得到。对于数值型值，首先利用直方图方法来为数据集和查询历史中的数值划分数值区间，然后把每一个数值区间作为一个文本值，最后通过扫描数据集和查询历史来计算出每个数值区间的原子概率 $p(y|W)$，$p(y|T)$，$p(x|y,W)$和 $p(x|y,T)$。注意，这里的

x 和 y 分别代表每个数值区间中出现频率最高的数值。

(2) 排序处理

对于查询结果中的每个元素，利用预处理阶段计算得到的原子概率值，根据排序函数计算元素的排序分数，最后利用快速排序算法对查询结果进行排序。排序的实现算法描述见算法 2。

算法 2 查询结果排序算法

输入：查询结果集 R，原子概率表

输出：排序结果

1. 令 B={}为保存查询结果集的缓存；
2. for 每个元素 t∈R do
3. for 每一属性单元 A do
4. if A 的取值是文本型值 then
5. 从原子概率表中获取 p(x|W)，p(x|T)，p(y|W)，p(y|T)，p(x|y,W)和 p(x|y,T)的值；
6. else /*若 A 的取值为数值型值*/
7. 利用预先划分的数值区间，计算 p(x|W)，p(x|T)，p(y|W)，p(y|T)，p(x|y,W)，p(x|y,T)值；
8. end for
9. 计算 $\text{Score}(t)=\frac{p(y|W)}{p(y|T)}\prod_{x\in X}\frac{p(x|W)}{p(x|T)}\prod_{y\in Y}\prod_{x\in X}\frac{p(x|y,W)}{p(x|y,T)}$；
10. end for
11. B=sort(R,Socre(t))；/*对结果元素按照 Score(t)的大小降序排列*/
12. return B

5 性能实验评价

本节介绍实验环境和实验数据，阐述实验方法和实验结果并对其进行分析。

5.1 实验环境

所有实验都是在 Intel(R) Core(TM2) 6300 @ 1.86GHz CPU，1G 内存，120G Bytes&7200rpm 硬盘和 Microsoft Windows XP 操作系统下进行的；使用 SAX2 解析 XML 文档；用 Java 编程语言和 eclipse3.0 编译工具开发实验系统。

数据集：测试文档由 IBM XML data generator[25] 产生，生成以下两个 XML 数据集：

(1) HouseDB 数据集：从 Yahoo! realestate 网站随机抽取了 237620 条房产信息记录，利用 IBM XML data generator 生成 HouseDB. xml 数据集，该数据集包括 237620 个 House 元素。

(2) CarDB 数据集：从 Yahoo! Autos 网站随机抽取了 100000 条二手车记录，利用 IBM XML data generator 生成 CarDB. xml 数据集，该数据集包括 100000 个 UsedCar 元素。HouseDB 和 CarDB 的文档结构如图 2 所示。

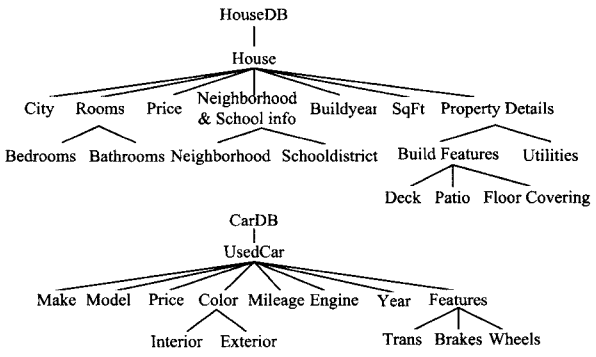

图 2 HouseDB 和 CarDB 的文档结构图

查询历史：首先建立了基于 HouseDB 和 CarDB 数据集的 Web 查询系统，跟踪了 3 个月的时间，获取到一定数量的查询历史，经过对查询历史进行修剪，最终分别为 HouseDB 和 CarDB 数据集保留了 2000 条和 2500 条用户查询作为查询历史。在此基础上，在各自查询历史和原始数据上利用统计方法和关联规则挖掘算法，分别为 HouseDB 和 CarDB 计算原子概率 $p(y|W)$，$p(y|T)$，$p(x|y,W)$和 $p(x|y,T)$，其中，关联规则的信任度阈值设置为 0.2。

5.2 结果排序的准确性测试

本实验目的是以用户调查方式评价结果排序的准确率。邀请了 15 位测试者，每位测试者根据自己的需求和偏好，对 HouseDB 和 CarDB 分别提出 1 条测试查询。

由于要求测试者从整个数据集中查找所有与查询相关的 House 和 UsedCar 元素是不现实的，因此采用了如下策略：对于 HouseDB 和 CarDB 上的每个测试查询 Q_i，生成一个相应的数据集 H_i，H_i 中包含了 60 个与该查询相关的和不相关的适当元素组合(对于测试查询 Q_i，分别使用随机排序方法、元素对查询指定属性单元值的满足程度排序方法和本文排序方法，取各自排序结果中的前 20 个元素相混合并去掉重复而构成数据集 H_i)。然后，将所有测试查询和对应的数据集都提供给每个测试者，他们的任务是从数据集 H_i 中标出与测试查询 Q_i 最为相关的前 10 个元素。在此基础上考察由本文排序方法返回的前 10 个元素与测试者标注的 10 个元素之间的重叠程度，用准确性(Accuracy)评价这种重叠程度，准确性定义为：

$$Accuracy=\frac{|top-10\ elements\cap relevant\ elements|}{10} \quad (10)$$

式中，$|top-10\ elements\cap relevant\ elements|$代表排序结果的前 10 个元素与测试者标注的 10 个元素之间相同的元素数。*Accuracy* 的值介于 0 到 1 之间，值越大表明两个集合之间相同的元素数越多，排序方法的准确性就越高，也就是说用户认为最为相关的结果被排到了前面。本文方法在 CarDB 和 HouseDB 数据集上的排序准确性如图 3 所示。

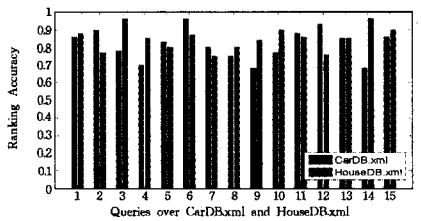

图 3 CarDB 和 HouseDB 数据集上的排序准确性

实验结果表明，排序方法在 CarDB 和 HouseDB 数据集上的平均排序准确性分别为 0.85 和 0.82。这是因为本文方法在排序过程中综合考虑了结果元素对初始查询和用户偏好的满足程度，从而能够有效地对查询结果进行排序。由此可见，所提排序方法能够较好地满足用户需求和偏好。

5.3 查询历史大小对排序准确性的影响测试

该实验目的是测试查询历史的大小(即查询历史中所包含的查询记录个数多少)对排序准确性的影响情况。在该实验中，分别将 CarDB 和 HouseDB 的历史查询记录个数调整为 2000、1000、500、200 和 100，然后在不同历史查询记录个数的情况下测试本文排序方法的准确性(见图 4)，测试结果

取 15 个测试查询的排序准确性的平均值。

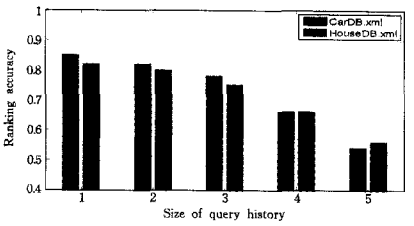

图 4　不同查询历史大小下的排序准确性

从图 4 的比较结果可以看出，查询历史越丰富，排序准确性越高。实验结果还表明，历史查询记录如果少于 100 条，则几乎不能有效地获取用户偏好，因此本文排序方法的准确性将会受到一定影响。不过，本文方法在查询历史缺失的情况下将转化为信息检索领域的 IDF 排序方法（如第 3.3 节讨论）；此外，排序函数还允许人工调整，因此即便是在查询历史缺失或不充分的情况下，排序函数也依然能达到不错的效果。

结束语　本文提出了一种基于条件偏好的 XML 数据多查询结果排序方法。主要贡献包括以下方面：(1) 提出了一种利用概率信息检索模型推测条件用户偏好并以此对查询结果进行排序的方法；(2) 给出了所提查询结果排序方法的实现方法。实验结果表明，提出的查询结果排序方法具有较高的排序准确性，能够较好地满足当前用户的需求和偏好。下一步工作将开展 XML 数据的多查询结果个性化分类方面的研究工作。

参 考 文 献

[1] Bray T. Extensible Markup Language(XML) 1.0[CP/OL]. http://www.w3.org/TR/REC-xml/,2008-11-26

[2] Baeza-Yates R, Ribeiro-Neto B. Modern information retrieval [M]. MA: Addison-Wesley, 1999

[3] Grossman D, Frieder O. Information retrieval - algorithms and heuristics [M]. Berlin: Springer, 2004

[4] Sparck J K, Walker S, Robertson S E. A probabilistic model of information retrieval: development and comparative experiments—Part 1 [J]. Information Processing and Management, 2000, 36(6): 779-808

[5] Sparck J K, Walker S, Robertson S E. A probabilistic model of information retrieval: Development and comparative experiments—Part 2 [J]. Information Processing and Management, 2000, 36(6): 809-840

[6] Croft W B, Lafferty J. Language modeling for information retrieval [M]. Norwell, MA: Kluwer, 2003

[7] Rui Y, Huang T, Merhotra S. Content-based image retrieval with relevance feedback in MARS [C]// Proceedings of IEEE International Conference on Image Processing. 1997: 815-818

[8] Wu L. Faloutsos C, Sycara K, et al. FALCON: feedback adaptive loop for content-based retrieval [C]// Proceedings of VLDB Conference. 2000: 297-306

[9] Kieling W, Kostler G. Preference SQL-Design, implementation, experiences [C]// Proceedings of the VLDB Conference. 2002: 990-1001

[10] Chomicki J. Preference formulas in relational queries [J]. ACM Trans on Database Systems, 2003, 28(4): 427-430

[11] Agrawal S, Chaudhuri S, Das G, et al. Automated ranking of database query results [J]. ACM Transactions on Database Systems, 2003, 28(2): 140-174

[12] Chaudhuri S, Das G, Hristidis V, et al. Probabilistic information retrieval approach for ranking of database query results [J]. ACM Trans. on Database Systems, 2006, 31(3): 1134-1168

[13] Theobald A, Weikum G. The index-based XXL search engine for querying XML data with relevance ranking [C]// Proceedings of the EDBT Conference. 2002: 477-495

[14] Marian A, Amer-Yahia S, Koudas N, et al.. Adaptive processing of top-k queries in XML [C]// Proceedings of the ICDE Conference. 2005: 162-173

[15] Amer-Yahia S, Koudas N, Marian A, et al. Structure and content scoring for XML [C]// Proceedings of the VLDB Conference. 2005: 361-372

[16] Li J X, Liu C F, Zhou R. Efficient top-k search across heterogeneous XML data sources [C]// Proceedings of the DASSFA Conference. 2008: 314-329

[17] Guo L, Shao F, Botev C, et al. XRANK: ranked keyword search over XML documents [C]// Proceedings of the SIGMOD Conference. 2003: 16-27

[18] Cohen S, Mamou J, Kanza Y, et al. Xsearch: a semantic search engine for xml [C]// Proceedings of the VLDB Conference. 2003: 45-56

[19] Sun C, Chan C Y, Goenka A K. Multiway SLCA-based keyword search in xml data [C]// Proceedings of the WWW Conference. 2007: 1043-1052

[20] Xu Y, Papakonstantinou Y. Efficient LCA based keyword search in XML data [C]// Proceedings of the EDBT Conference. 2008: 535-546

[21] 衡星辰，覃征，邵利平，等. 基于两阶段查询重写的 XML 近似查询算法 [J]. 电子学报，2007，35(7)：1271-1278

[22] 徐超，张东站. sTwig——一种基于流的 XML 小枝匹配算法 [J]. 计算机研究与发展，2010，47：86-92

[23] Wu L, Faloutsos C, Sycara K, et al. FALCON: feedback adaptive loop for content-based retrieval [C]// Proceedings of VLDB Conference. 2000: 297-306

[24] Millist W V, Liu J X, Liu C F. Strong functional dependencies and their application to normal forms in XML [J]. ACM Transactions on Database Systems, 2004, 29(3): 445-462

[25] IBM Corporation XML data generator [EB/OL]. http://www.alphaworks.ibm.com/tech/xmlgenerator