基于加权 XML模型的 XML数据与 DTD模式匹配*

李树青　程国达　王维民
(南京财经大学信息工程学院　南京 210046)

【摘要】首先说明利用加权 XML数据模型分别得到标准 XML参考实例和 XML数据实例的方法,并对 DTD约束修饰符的表达方法进行介绍。其次,详细阐述相似度算法的实现方法,重点说明在 XML数据实例中寻找与标准 XML参考实例的匹配节点算法和计算标准 XML参考实例与 XML数据实例的相似度算法。最后,对相关实验及其结论进行总结。

【关键词】加权 XML　DTD　相似度　模式匹配
【分类号】TP391

The Schema Matching of XML and DTD Based on Weighted XML Data Model

Li Shuqing　Cheng Guoda　Wang Weimin
(College of Information Engineering, Nanjing University of Finance & Economics, Nanjing 210046, China)

【Abstract】This paper first introduces standard XML reference instants and XML data instants based on the weighted XML data model. Then it displays the expression ways of constraints in DTD. Furthermore, the paper also shows the approaches on how to implement similarity algorithm, with an emphasis on how to find out a matching node with standard XML reference instants and to get the similarity algorithm of standard XML reference instants and that of XML data instants.

【Keywords】Weighted XML　DTD　Similarity　Schema matching

1 引　言

XML数据具有事先定义好的文法规则(Grammar),包含 DTD(Document Type Definition)和 Schema两种主要形式。这些内容都可以定义对应 XML数据中的元素和属性,并且对它们之间的结构关系也可以做详细的规定。和传统的关系型数据库一样,XML文法规则对于 XML文档验证、索引和检索来说都具有重要的辅助作用[1]。

其中,DTD是一种较为简单有效的 XML数据模式定义方法。它有两方面的作用:

(1)提供一种约束 XML数据结构特征的强制手段,通过检查 DTD中的模式定义,系统可以检测已有的 XML数据是否规范,并对不符合要求的 XML数据给出具体的错误信息。它的相关应用很常见,如各种 XML数据编辑器等。

收稿日期:2009-12-07
收修改稿日期:2009-12-22
* 本文系江苏省教育厅"青蓝工程"基金资助项目和江苏省教育厅高校自然科学研究项目"分布式数据流的挖掘及其应用"(项目编号:06KJD520073)的研究成果之一。

(2)利用 DTD 作为一种并不严格的建议标准,推荐不同的 XML 数据符合它的定义要求,但是如果 XML 数据不符合,系统也并不会给出错误信息,而是认为该 XML 数据在一定程度上不符合 DTD 的规范定义。事实上,在各种商业数据交换领域中,由于难以对不同的参与者进行有效的强制约束,所以即便是在特定领域中人们已经制定了相关的 DTD 规范,也难以确保所有的相关 XML 数据都能够严格符合 DTD 的要求。但由于数据交换的需要,此时亟需一种手段来对这些格式并不十分严谨但却在很大程度上遵守 DTD 模式要求的不同 XML 数据进行相似度匹配。这种问题也被称为 XML 数据的模式匹配(Schema Matching)。

与此相关的应用领域有很多,如根据 DTD 模式对 XML 数据进行分类,在 XML 文档聚类时抽取结构特征构造新的 DTD,基于结构特征信息进行 XML 数据检索,对 XML 文档的定题情报服务(Selective Dissemination of XML Documents)等。同时,该项研究有助于很多数据库领域的应用实现,如在数据仓库中将不同格式的 XML 数据映射成统一的模式,或者在数据库系统中利用已有的 DTD 模式对 XML 数据进行结构性验证[2]。

2 文献回顾

2.1 方法说明

从概念上说,一般而言,DTD 模式通过约束关系定义了一组相互连接的 XML 元素、子元素和属性,而 XML 数据与 DTD 模式的相似度匹配可以表达为一个函数,该函数接受 XML 数据与 DTD 模式这两个参数,返回一个用于表征相似程度的度量值[3]。

通常人们把某个特定应用领域中被大家认可的标准 XML 数据交换格式称为领域模式(Domain Schema),而把某个参与组织所发布的特定 XML 数据格式称为源模式(Source Schema)。

在现有的研究领域中,大量的研究工作主要集中于 XML 文档之间的相似度计算,对于 XML 数据及其 DTD 模式之间的相似度研究却不多见[2]。较早开展的相关研究主要集中在关系型数据模型和 ER 模型等领域[4],随后的相关研究主要集中于比较不同源模式之间的相似度[5]。近年来,随着 XML 数据应用的快速发展,XML 模式匹配问题受到人们的广泛关注[2]。

从总体来看,计算 XML 数据与 DTD 模式之间的相似度,可以采取两种不同的方法:

(1)将 DTD 看成是 XML 数据结构的抽象,该种方法可以被称为外延方法(Extensible Approach),此时取出一个需要比较的 XML 数据实例,并计算与其他 XML 数据实例的相似度,最后可以将全部计算结果中最大的相似度值作为最终的相似度。这种方法的好处在于可以利用现有的 XML 数据相似度计算方法来进行。

(2)将 DTD 看成是约束 XML 数据的一组规则集合,该种方法可被称为内涵方法(Intensional Approach)。该方法需要利用每个 XML 数据与 DTD 模式进行相似度的比较计算。和前一个方法相比,该方法计算量要小得多,但是需要人们研究和设计一种有效的模式匹配方法[4]。本文的研究也采取了这个思路。

2.2 研究方法介绍

(1)基于树编辑距离的匹配方法

传统的模式比较方法主要采用基于树编辑距离(Tree Edit Distance)的方法。在各种以数据为中心(Data-Centric)的 XML 数据组织模式中,由于数据结构较为严谨,便于数据交换和机器自动化处理,所以基于树编辑距离的方法适合处理该种类型的 XML 数据[6]。

Nierman 和 Jagadish 等人的方法是一种提出较早并且具有较高准确度的方法,特别是在判断部分 XML 文档内容即 XML 文档子树相似度方面效果较好[7]。对于算法性能而言,该方法具有和一般编辑距离方法相同的时间复杂度。后来的很多学者据此也提出了许多改进算法,如有的学者利用该方法进行 XML 文档聚类的研究工作[8];还有学者将 XML 文档和 DTD 结构都表示成有向标签树结构,并忽略所有属性元素,同时综合考虑各种 DTD 限制修饰符的作用,并利用基于编辑距离的方法提出了一种新的相似度测度方法[9]。

BiBranch 方法通过一个整数值来描述两者比较模式的差异度,该整数值越小,则相似度越大[10]。然而,该方法也存在很多问题。如 BiBranch 方法考虑了节点次序,而事实上在 XML 检索中往往关注是否命中节点本身,对位于同一节点下的直接下级节点往往并不在意它们之间的相互次序。再如 BiBranch 方法由于采用编辑距离的操作方法,单纯以节点编辑操作数量为差异度的衡量标准,没有很好地区分丢失节点和增加节

点两者的差异。有学者据此提出一种新的模式相似度比较方法(Source Schemas against a Domain, SSD),该方法侧重于覆盖面的比较,认为两个模式如果存在更多相同的元素,则相似度更大。同时,该方法考虑节点关系的影响程度,其中忽略同级节点元素的次序关系,对那些对结构关系没有影响的冗余节点也予以忽略[5]。

纵观上述研究,这些利用树编辑距离的方法往往具有较高的算法复杂度,而且每种方法都是适应不同应用系统的要求而提出的,缺乏通用性的特点。

(2)基于机器学习的匹配方法

这些方法往往结合使用多种匹配器(Matcher),每种匹配器都采用一种有效的独立方法。在实际使用中,人们可以根据需要灵活调整这些匹配器的组合,来构造更为有效的复合方法。

如一种较早提出的 XML模式匹配方法就是学习源描述法(Learning Source Description, LSD),它主要使用机器学习的方法来半自动化发现不同模式之间的映射关系,据此计算它们之间的相似度。它需要两个处理阶段:

①训练阶段,此时系统要求用户提供反映模式映射关系的数据源样本集合,并以此得到训练学习规则;

②映射阶段,利用不同的学习规则可以得到不同种类的相似度测度值,如名称匹配器(NameMatcher)主要根据元素属性的文本内容来计算相似度等[11]。

然而,该方法需要大量的人工操作,特别是在样本训练阶段。后续研究学者提出了很多新方法,无需预处理和复杂的机器学习过程,这些方法主要采用基于图论的一些模式匹配算法。如有学者提出一种基于句法的 DTD文档相似度判断方法,将整个 DTD结构看成一个有向无环根树(Rooted Directed Acyclic Graph),采用自下而上的方法:首先根据叶节点的名称来判断它们的相似度,然后计算上层中间节点的相似度,对于它们相似度的计算而言,主要是利用这些节点所含有的下级子图结构相似度来进行[12]。

还有学者通过定义几种能够捕获不同 DTD模式之间差异度的操作方法,并利用一种基于启发式功能的成本估算方法从中选择最优的操作步骤。对于每个源 DTD中的节点,该算法会在目标 DTD中相同节点层次和更低的节点层次中发现候选的匹配模式。但是该方法存在两个局限性:有些转换操作不能很好地适应一些较为模糊的数据结构;该方法没有很好地结合语义信息来提高其匹配精度[12]。

相反,另外一些学者就指出如果能够充分利用这些语义信息,就可以改善匹配精度,并减少结果中的不一致现象[13]。

(3)基于语义分析的方法

在 XML模式相似度研究领域,还有一个研究分支侧重于对 XML数据元素的信息内容进行分析。传统的内容方法一般都是将节点内容看成是字符串类型,然而事实上随着 XML所能表达的数据内容越来越丰富,人们逐渐需要给不同的节点内容指定不同的数据类型,如数值型和布尔型等。类型不一样,所对应的相似度计算方法也不一样。这种 XML模式相似度可以被称为 XML文法相似度[1]。

有学者利用通用语义模型(Universal Semantic Model)来表达 XML的模式和进行不同模式的相似度计算。该语义模型主要由三部分组成,分别是本体类别(Ontological Categories)、属性(Properties)和上下文限制(Contextual Constraints)[14]。还有学者利用组合方法计算 XML模式的相似度,该方法结合考虑了两种成分:通过最小共同类型计算得到的最大信息量;类型的声明信息[15]。还有学者提出了 7条简化原则来将 XML模式转化为可以存储相同信息和叶节点基数限制特征的最精简结构[16]。

一些学者不再考虑简单的 DTD模式,而是考虑约束内容更为丰富的 Schema模式。有人提出需要对复杂数据类型和自定义数据类型进行考虑,但是没有给出具体的处理方法[13]。还有学者却提出应当忽略这些复杂数据类型和用户自定义数据类型[17]。同时,上述学者都没有过多考虑诸如 ALL和 OR等 Schema操作符的处理方法及其对相似度计算的影响。

在更近的研究中,有学者对 Schema模式中的复杂数据类型进行研究,提出类型层次树(Type Hierarchy)的概念,即利用扩散限制关系将不同类型组织成一个完整的层次结构。对于两个复杂元素的相似度比较而言,测度主要利用类型层次树和元素的结构相似度,根据数据类型的名称来得到语义相似度。但是作者并没有将此方法扩展到完整的 Schema模式匹配计算过程中[15]。

值得注意的是,在这些已有的研究文献中,很多学者都对标准的 DTD模式标准进行了适应性的修改,如

为了有效地实现相似度计算,很多学者对一些常见的DTD规则进行简化,去除对规定是否可以存在的“?”修饰符和规定可以重复出现的“+”修饰符等的考虑;有的学者所考虑的XML文法都较简单,往往忽略对重复元素(Cardinality)和可选元素(Alternativeness)的约束处理。不同方法的性能具有很大的差异,对于如何有效衡量不同算法的有效性,有学者提出检验XML模式匹配算法有效性的一个简单标准就是根据任务中所需的人工处理工作量大小[18]。

3 利用加权XML模型表示DTD模式和XML数据实例

本文所提出的方法没有沿用上述传统模式,而是首先对标准DTD模式,即XML领域模式,定义一个XML标准参考实例,其中的每一个节点都被分配一个权值,对该值的分配采取权值扩散的策略和引入约束操作符的限制,并且利用这种节点权值来表达各种结构特征和约束信息,将这个XML标准参考实例称为标准XML参考实例,同时把各种XML源模式实例称为XML数据实例。利用标准XML参考实例作为与其他XML数据实例进行比较的依据,并以比较结果作为XML数据实例与DTD模式的相似度匹配结果。

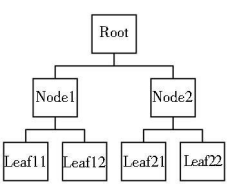
3.1 标准XML参考实例的定义方法

(1) 简单标准**XML**参考实例的定义方法

为说明方便,先给出一个DTD模式,注意它的所有节点都有Weight权值属性,如图1所示:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Leaf11 EMPTY>
<!ATTLIST Leaf11 weight CDATA #REQUIRED>
<!ELEMENT Leaf12 EMPTY>
<!ATTLIST Leaf12 weight CDATA #REQUIRED>
<!ELEMENT Leaf21 EMPTY>
<!ATTLIST Leaf21 weight CDATA #REQUIRED>
<!ELEMENT Leaf22 EMPTY>
<!ATTLIST Leaf22 weight CDATA #REQUIRED>
<!ELEMENT Node1 (Leaf11, Leaf12)>
<!ATTLIST Node1 weight CDATA #REQUIRED>
<!ELEMENT Node2 (Leaf21, Leaf22)>
<!ATTLIST Node2 weight CDATA #REQUIRED>
<!ELEMENT Root (Node1, Node2)>
<!ATTLIST Root weight CDATA #REQUIRED>
```

(a) DTD模式定义代码



(b) DTD模式的树状视图

图1 一个DTD模式样例

对于节点权值的分配方法,可以采取以下步骤:

①给每个叶节点分配一个数值为1的权值。

②采取自下而上的扩散策略,非叶节点的权值为所有直接子节点的权值之和。

具体表述如下:

$$weight_{Node_i}=\begin{cases}1 & if\ Node_i\ is\ leaf\ node\\ \sum weight_{Node_{ij}} & if\ Node_i\ is\ not\ leaf\ node\end{cases} \quad (1)$$

在公式(1)中,节点Nodeij为当前节点Nodei包含的所有直接下级一层节点。以下公式中的定义与此相同,将不再说明。

据此,可以得到图1中DTD模式的标准XML参考实例,如例1所示:

例1:

```
<Root weight="4">
  <Node1 weight="2">
    <Leaf11 weight="1"></Leaf11>
    <Leaf12 weight="1"></Leaf12>
  </Node1>
  <Node2 weight="2">
    <Leaf21 weight="1"></Leaf21>
    <Leaf22 weight="1"></Leaf22>
  </Node2>
</Root>
```

(2) 带有约束修饰符的标准**XML**参考实例定义方法

除了节点说明信息外,DTD模式通常还会对节点元素定义多种约束修饰符,其中重要的修饰符有如下5种:“?”、“*”、“+”、“|”和“,”。其中,“?”表示所修饰节点元素可以出现0次或者1次,“*”表示所修饰节点元素出现次数大于或等于0次,“+”表示所修饰节点元素出现次数大于或等于1次,“|”表示当前节点的下级节点元素只能并必须有一个元素出现,“,”表示所分隔的每个节点构成一个完整的序列。

值得注意的是,对于约束修饰符,由于主要用于限制XML数据实例的节点出现次数,所以难以通过权值分配在标准XML参考实例中加以体现。相反,应该在与标准DTD结构比较的XML数据实例中予以体现。因此,除“|”和“,”外,对于其他约束修饰符在此都不予考虑,得到标准XML参考实例。

对于“|”修饰符的多个节点元素,只能有一个出现在XML数据实例中,所以在标准XML参考实例中,利用这些供选择节点的平均权值作为扩散权值,具体计算如下:

$$weight_{Node_i}=avg(weight_{Node_{ij}}) \quad (2)$$

对于“, ”修饰符的多个节点元素, 前文已经说明过, 将采用所有相关节点的权值之和作为其上级非叶节点的权值, 在此不再赘述。

3.2 XML数据实例的表示方法

(1) 简单 XML数据实例的定义方法

假设有 XML数据实例如例 2所示:

例 2

```
<Root weight= "3">
  <Node1 weight= "1">
    <Leaf11 weight= "1"> </Leaf11>
  </Node1>
  <Node2 weight= "2">
    <Leaf21 weight= "1"> </Leaf21>
    <Leaf22 weight= "1"> </Leaf22>
  </Node2>
</Root>
```

该 XML数据实例并不完全对应如图 1所示的标准 DTD结构, 它有以下几个特点:

①XML数据实例与 DTD模式的结构性差异可以通过节点权值的差异体现出来, 而且权值的差异在一定程度能够反映结构的差异程度。

②该种权值可以表示节点层次结构对相似度的影响程度, 一般而言, 层次越高的节点影响程度越大。事实上, 在该种 XML数据实例中, 层次越高的节点往往具有越高的权值, 因此在相似度计算时将会产生更为明显的作用。

(2) 带有约束修饰的 XML数据实例定义方法

前文已经说明了几种约束修饰符的表达方法, 下面重点介绍一下其他 DTD约束修饰符在 XML数据实例中的表达方法:

①“?”表示方法

从概念上看, 虽然此修饰符表示所修饰元素最多只能出现 1次, 但是该元素也可以不存在。因此, 只有 XML数据实例中具有超过一个的相应节点元素, 才会被认为不规范。为了表达这种限制, 可以在 XML数据实例的权值扩散中, 对出现多次的节点元素, 将其所有子节点的权值之和扩散到当前节点中。显然, 当前节点包含的子节点重复次数越多, 则当前节点的权值与标准 XML参考实例的权值差异就会越大。具体计算如下:

$$weight_{Node_i} = \sum weight_{Node_{ij}} \qquad (3)$$

②“* ”表示方法

从概念上看, 该修饰符的约束能力近似于“?”, 只是在最多出现次数上没有限制, 约束力更为宽松。简单的处理方法可以不考虑此修饰符的影响, 采用忽略的方法。然而, 在各种实际应用中, 对于此修饰符所修饰的元素, 由于不同的 XML数据实例往往具有较多重复次数的元素值, 而且元素值差异较大, 所以有必要在权值设计上反映这种差异。采取的方法是对出现多次的节点元素, 只将其最小权值扩散到上级节点中。之所以选择最小值, 主要是以此来反映它与标准 DTD模式结构的相同性。具体计算如下:

$$weight_{Node_i} = \min(weight_{Node_{ij}}) \qquad (4)$$

③“+”表示方法

由于该修饰符要求所修饰元素至少要出现一次。因此, 只有当 XML数据实例不存在该节点时, 才会代表一种不规范的特征。事实上, 这种差异完全可以通过节点权值得以体现, 如果 XML数据实例相应节点没有子节点, 则当前节点无法获得扩散过来的权值。同时, 在后文的相似度计算方法中, 还会进一步考虑到这种节点丢失产生的结构差异度。因此, 本文没有考虑此种约束修饰符对权值扩散的直接影响。

大多数情况下不必考虑节点和属性等元素的先后次序[1], 所以本文对此问题没有考虑。

4 相似度算法的完整描述

利用上述方法得到标准 XML参考实例与待比较的 XML数据实例, 可以进一步计算两者的相似度。为此需要解决两个问题:

(1)在 XML数据实例中寻找与标准 XML参考实例的匹配节点;

(2)计算标准 XML参考实例与 XML数据实例的相似度。

4.1 在 XML数据实例中寻找与标准 XML参考实例匹配的节点

XML数据实例是通过采用不同的权值扩散策略来体现不同约束修饰符的约束能力。为此, 首先在 XML数据实例中准确定位相关节点的位置。由于不同的 XML数据实例与标准 DTD模式相比, 在结构上往往存在较大的差异, 所以需要设计一种算法来有效地找到对应的节点。

假设标准 XML参考实例为 x^r, XML数据实例为 x^i, 两者的相似度算法伪代码可以表示为:

```
//循环处理标准 XML参考实例中每一个叶节点路径
for each leafpath lpref_i of x^r {
  //如果该路径上的节点含有约束修饰符
  if( hasConstraintInPath ( lpref_i )) {
    int indexPath;  //与标准 XML参考实例中叶节点路径对应
      的 XML数据实例叶节点路径
```

```
int minED = MAX; //临时变量,存储编辑距离转换的最小值
int indexNode; //与标准 XML参考实例中节点对应的 XML数据实例节点
double minWeight; //临时变量,存储节点权值的最小差值
//循环处理 XML数据实例中每 个叶节点路径
//根据编辑距离来获取与标准 XML参考实例最相似的匹配节点路径
for each leafpath lpint_j of x^i
  if(minED > editDistance(lpref_i, lpint_j)){
    indexPath = j;
    minED = editDistance(lpref_i, lpint_j);
  }
//循环处理标准 XML参考实例中当前节点路径的每一个节点
//按照从叶节点到根节点的倒序方向处理,以确保处理过的节点不再被处理
for each node n_k of lpref_i in leaf-to-root order{
  //如果该节点含有约束修饰符
  if(hasConstraintInNode(n_k)){
    //循环处理 XML数据实例中最相似节点路径的每一个节点
    //获取与标准 XML参考实例中当前节点权值差异度最小的 XML数据实例节点
    //分配相应的约束修饰符
    for each node m_l of lpint_indexPath{
      int difference = abs(getWeight(n_k, lpref_i), getWeight(m_l, lpint_indexPath));
      if(minWeight > difference){
        indexNode = l;
        minWeight = difference;
      }
      addConstraint(m_indexNode);
    }
    //在 XML数据实例当前处理节点路径中删除已经添加过约束修饰符的节点,以确保处理过的节点不再被处理
    lpint_indexPath = substring(lpint_indexPath, indexOf(m_indexNode), length(lpint_indexPath));
  }
 }
}
```

说明如下:

(1)该方法用于在标准 XML参考实例中寻找与含有约束修饰符的当前节点最匹配的 XML数据实例节点,并将相应的约束修饰符标记下来,为后续的 XML数据实例权值扩散提供判断依据。

(2)该方法首先以每个标准 XML参考实例的叶节点完整路径作为处理单元,采用基于编辑记录的测度方法,在 XML数据实例中寻找最为合适的节点所在路径。然后,该方法寻找当前路径上权值差异度最小的节点,以此节点作为命中节点并分配相应的约束修饰符。

(3)XML数据实例中已经处理过的节点无需再次参与下一轮的比较运算,所以需要按照从叶节点到根节点的方法来遍历标准 XML参考实例每条叶节点路径上的所有节点,同时不再比较 XML数据实例中已经处理过的节点。

4.2 计算标准 XML参考实例与 XML数据实例的相似度

对于既有的标准 XML参考实例和 XML数据实例,仍然可以采用利用权值的差异度来表示结构差异度的思路,得到一种新颖的相似度计算方法。

假设标准 XML参考实例为 x^r,加权 XML数据实例为 x^i,两者的相似度算法伪代码可以表示为:

```
Collection S; //存储所有比较节点的集合
//存储 XML数据实例中所有叶节点与对应标准 XML参考实例中叶节点权值差值的数组
//该数组长度为 XML数据实例中叶节点 个数,表明要考虑所有 XML数据实例中叶节点的匹配情况
double[ ] minDifferenceValue = new double[countOfLeafnode(x^r)];
//循环处理标准 XML参考实例中每一个叶节点路径
for each leafpath lpref_i of x^r{
  //循环处理 XML数据实例中每一个叶节点路径
  for each leafpath lpint_j of x^i{
    int totalDifferenceOfLevel; //临时变量,存储当前叶节点路径上的权值差异总和
    //根据标准 XML参考实例和 XML数据实例,得到对应节点路径上的全部待比较节点
    S = getUnionSet(getAllNodeInPath(lpref_i), getAllNodeInPath(lpint_j));
    //累加对应节点路径上的全部待比较节点的权值差值
    for each s of S{
      totalDifferenceOfLevel += abs(getWeight(ln_i, s) - getWeight(ln_j, s));
    }
    //获取与标准 XML参考实例当前节点路径权值差值总和
```

```
            的最小值
            if(minDifferenceValue[index(lni)] > totaDifferenceOfLev-
              el){
              minDifferenceValue[index(lni)] = totaDifferenceOfLevel;
            }
          }
        }
        //以最小的权值差值总和作为最终的相似度测度数值
        //该相似度计算方法的测度结果为大于等于0的一个浮点数,该数值越小,则越相似
        return min(minDifferenceValue);
```

5 实　验

为了对上述方法的有效性进行验证,笔者在目前正在进行的一个基于个性化推荐服务的用户个人信息体管理系统(Personal Information Agent, PIA)上进行实验。实验的硬件平台环境为: CPU　Intel Core 2 Duo P8400,内存　PC3-8500 DDR3 2.0GB,　软件平台为: Windows Server 2003, SQL Server 2005, JDK1.6, Eclipse 3.3。

实验分为两个步骤: 对上述相似度测度基本方法的有效性进行测试; 在具体项目平台上测试用户满意度。

5.1　基本方法的测试

所有的标准 XML参考实例和 XML数据实例都是以表记录的方式存储在 SQL Server 2005数据库中。其中, XML数据实例所在表为 XMLInt,例 2所示 XML数据实例的存储信息如表 1所示:

表 1　XMLInt表的部分记录

ID	节点路径	节点权值	是否为叶节点	修饰符类型
1008	Leaf11/Node1/Root	0.3333333333333331	True	Null
1010	Leaf21/Node2/Root	0.3333333333333331	True	Null
1011	Leaf22/Node2/Root	0.3333333333333331	True	Null
1012	Node1/Root	0.3333333333333331	False	Null
1013	Node2/Root	0.6666666666666663	False	Null
1014	Root	1	False	Null

标准 XML参考实例所在表为 XMLRef,结构类似于 XMLInt,图 1所示 DTD所生成的标准 XML参考实例存储信息如表 2所示。

需要说明以下三点:

(1)两张表都存储了对应 XML数据中的所有节点路径及其相应权值,为了计算方便,节点路径采取倒序的表达方式。

表 2　XMLRef表的部分记录

ID	节点路径	节点权值	是否为叶节点	最匹配的XML数据实例节点路径	是否含有修饰符	修饰符类型
801	Leaf11/Node1/Root	0.25	True	1008	False	Null
802	Leaf12/Node1/Root	0.25	True	1008	False	Null
803	Leaf21/Node2/Root	0.25	True	1010	False	Null
804	Leaf22/Node2/Root	0.25	True	1011	False	Null
805	Node1/Root	0.5	False	1012	False	Null
806	Node2/Root	0.5	False	1013	False	Null
807	Root	1	False	1014	False	Null

(2)在 XMLRef表中,增加了一个表达最匹配的 XML数据实例节点路径的字段,字段值在计算中会随着比较对象的变化而更新,如表 2所显示的值就是与表 1所示 XML数据实例比较后的计算结果。之所以在 XMLRef中存储该信息,是因为通常认为一个遵守 DTD的 XML数据实例应该具有较多的对应 DTD规定元素, XML数据实例中缺少对应 DTD规定元素要比出现多余节点更能反映出一种不相似的状态。所以,通过遍历标准 XML参考实例的所有节点路径,在 XML数据实例中寻找最相似的节点路径,然后据此分配约束修饰符并为后续相似度计算提供基础。

(3) XMLRef表还有表示是否含有修饰符和约束修饰符类型的字段,可以在 XML数据实例分配约束修饰符时使用。

对于表 1和表 2所示的标准 XML参考实例和 XML数据实例,它们的相似度计算结果如表 3所示:

表 3　相似度计算结果

标准 XML参考实例比较节点	XML数据实例比较节点	相似度
Leaf11/Node1/Root	Leaf11/Node1/Root	0.25
Leaf21/Node1/Root	Leaf11/Node1/Root	0.416667
Leaf21/Node2/Root	Leaf21/Node2/Root	0.25
Leaf22/Node2/Root	Leaf22/Node2/Root	0.25

最终的相似度计算结果为 0.25,反映了部分相似的特点。

实验还对两种极端情况做了对比测试:

(1)比较完全不相似的 XML数据实例,使用数据为表 1所示的标准 XML参考实例和表 4所示的 XML数据实例,相似度结果如表 5所示。

表 4　一个完全不相似的 XML数据实例记录

ID	节点路径	节点权值	是否为叶节点	修饰符类型
1018	Instant/XML	1	True	Null
1019	XML	1	False	Null

表 5 相似度计算结果

标准 XML参考实例比较节点	XML数据实例比较节点	相似度
Leaf11 /Node1 /Root	Instant/XML	1.75
Leaf21 /Node1 /Root	Instant/XML	1.75
Leaf21 /Node2 /Root	Instant/XML	1.75
Leaf22 /Node2 /Root	Instant/XML	1.75

最终的相似度计算结果为 1.75,反映了高度不相似的特点。

(2)比较完全相似的 XML数据实例,使用数据为表 1所示的标准 XML参考实例和表 6所示的 XML数据实例,相似度结果如表 7所示:

表 6 一个完全相似的 XML数据实例记录

ID	节点路径	节点权值	是否为叶节点	修饰符类型
1115	Leaf11 /Node1 /Root	0.25	True	Null
1116	Leaf12 /Node1 /Root	0.25	True	Null
1117	Leaf21 /Node2 /Root	0.25	True	Null
1118	Leaf22 /Node2 /Root	0.25	True	Null
1119	Node1 /Root	0.5	False	Null
1120	Node2 /Root	0.5	False	Null
1121	Root	1	False	Null

表 7 相似度计算结果

标准 XML参考实例比较节点	XML数据实例比较节点	相似度
Leaf11 /Node1 /Root	Leaf11 /Node1 /Root	0
Leaf21 /Node1 /Root	Leaf21 /Node1 /Root	0
Leaf21 /Node2 /Root	Leaf21 /Node2 /Root	0
Leaf22 /Node2 /Root	Leaf22 /Node2 /Root	0

最终的相似度计算结果为 0,反映了完全相似的特点。

通过测试,得出如下结论:

(1)如果比较的标准 XML参考实例和 XML数据实例完全相同,则相似度为 0,两者越不相似,则相似度测度值越大,该值能够反映不同程度的结构差异。

(2)如果比较的标准 XML参考实例和 XML数据实例完全不相同,最终得到的相似度测度值会大于 1,而且该值会随着 DTD结构本身的复杂度而变大。

5.2 用户满意度测试

利用该方法在正在实施的基于个性化推荐服务的用户个人信息体管理系统(PIA)上进行用户满意度测试。该系统主要利用 Web用户的各种 Web访问信息和个性化特征信息来自动化地构建用户个性化本体,并以此进行个性化信息推荐服务。在实验中,采用 DTD结构来定义用户的个性化本体,如图 2所示:

实验首先得到系统中已有的 20个 PIA个性化本体信息,并进行适当的手工修正,要求 12位测试用户

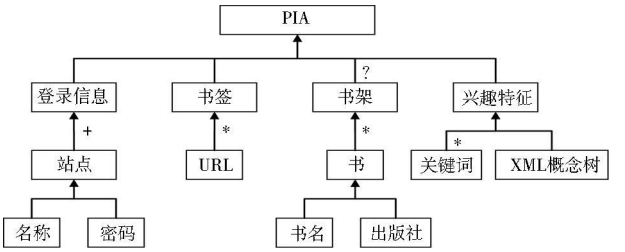


图 2 PIA结构

对所有的 PIA信息按照与 DTD模式的遵守规范程度进行排序。把排序汇总后的结果看成一个二维表格,如表 8所示:

表 8 测度结果表格说明

层次号	用户 1	…	用户 12
1	R_2	…	R_{15}
2	R_{11}	…	R_{11}
…	…	…	…
20	R_{19}	…	R_6

表格中的每一列表达一个测试用户的评价排序结果,每一行表达一个排序等级,层次号越小则越相似。其中的每一个单元格内容表达一个 PIA个性化本体信息,如 R_2 表示第二个 PIA个性化本体信息。

据此,定义了一个用户满意度的间接测度指标,即:

$$Index_{satisfaction}=\frac{\sum_{level\,i}\left(1-\frac{Distinct_i-1}{Total_i-1}\right)}{Total_i} \qquad (5)$$

其中,$Total_i$ 表示排序层次 i上所有的 PIA个性化本体信息个数,此处为常量 12,$Distinct_i$ 表示排序层次 i上彼此不同的 PIA个性化本体信息个数。显然,该测度值为一个 0到 1之间的小数,数值越大说明不同用户的评价结果越一致。

系统实验进行了 3次,平均用户满意度为 0.77,但是如果只考虑层次号小于 5和大于 15的结果,发现平均用户满意度更高,为 0.83。这在一定程度上说明了该方法能够较好地区别最满意的结果和最不满意的结果,对处于两者之间的相似结果判断能力相对较低。

6 结 语

本文提出了利用加权 XML数据模型实现 XML数据与 DTD模式的相似度匹配方法,并对该方法的理论基础、实现过程及其实验结论都做出了详细说明。初步的实验表明该方法满足要求,同时具有算法简单和缩放性强的特点。但是也发现了该方法所存在的问

题,特别是对于约束修饰符的表达和相似度算法中匹配节点的定位与寻找,笔者还在测试以期找到更优秀的替代方法。同时也发现,在进行 XML数据与 DTD模式相似度匹配比较时,除了要考虑元素的结构对应特征外,还要考虑一些语义信息,如 XML数据实例中元素的名称可能与 DTD模式中规定的标准名称虽然同义但是字面却存在差异。所以,在下一阶段的研究中,笔者将结合节点语义信息,从更多方面增强对 XML数据与 DTD规范一致性的判断准确性。

参考文献:

[1] Bertino E, Guerrini G, Mesiti M. A Matching Algorithm for Measuring the Structural Similarity Between an XML Documents and a DTD and Its Applications[J]. *Information Systems*, 2004, 29(1): 23-46.

[2] Tekli J, Chbeir R, Yetongnon K. An XML Grammar Comparison Framework-Technical Report[R/OL]. [2009-11-11]. http://www.u-bourgogne.fr/Dbconf/XGM/MatchersAndExperiments.pdf

[3] Rahm E, Bernstein P A. A Survey of Approaches to Automatic Schema Matching[J]. *The VLDB Journal*, 2001, 10(4): 334-350.

[4] Silvana Castano, Valeria De Antonellis, Sabrina De Capitani di Vimercati. Global Viewing of Heterogeneous Data Sources[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2001, 13(2): 277-297.

[5] Li J X, Liu J X, Liu C F, et al. Computing Structural Similarity of Source XML Schemas Against Domain XML Schema[C]. In *Proceedings of the 19th Conference on Australasian Database*, Gold Coast, Australia. Darlinghurst, Australia: Australian Computer Society, 2008: 155-164.

[6] Guerrini G, Mesiti M, Sanz I. An Overview of Similarity Measures for Clustering XML Documents[EB/OL]. [2009-12-01]. http://krono.act.uji.es/publications/pdf/gms.pdf

[7] Nierman A, Jagadish H V. Evaluating Structural Similarity in XML Documents[C]. In *Proceedings of the 5th ACM SIGMOD International Workshop on the Web and Databases*, 2002: 61-66.

[8] Dalamagas T, Cheng T, Winkel K, et al. A Methodology for Clustering XML Documents by Structure[J]. *Information Systems*, 2006, 31(3): 187-228.

[9] Tekli J, Chbeir R, Yetongnon K. Structural Similarity Evaluation Between XML Documents and DTDs[C]. In *Proceedings of the 8th International Conference on Web Information Systems Engineering (WISE'07)*, Nancy, France. Berlin Heidelberg: Springer-Verlag, 2007: 196-201.

[10] Yang R, Kalnis P, Tung A K H. Similarity Evaluation on Tree-structured Data[C]. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, Baltimore, Maryland. New York, USA: ACM Press, 2005: 754-765.

[11] Doan A, Domingos P, Halevy A Y. Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach[J]. *ACM SIGMOD Record*, 2001, 30(2): 509-520.

[12] Su H, Padmanabhan S, Lo M L. Identification of Syntactically Similar DTD Elements for Schema Matching[C]. In *Proceedings of the 2nd International Conference on Advances in Web-Age Information Management*. London, UK: Springer-Verlag, 2001: 145-159.

[13] Boukottaya A, Vanoirbeek C. Schema Matching for Transforming Structured Documents[C]. In *Proceedings of the 2005 ACM Symposium on Document Engineering*, Bristol, UK. New York, USA: ACM Press, 2005: 101-110.

[14] Yi S, Huang B, Chan W T. XML Application Schema Matching Using Similarity Measure and Relaxation Labeling[J]. *Information Sciences*, 2005, 169(1-2): 27-46.

[15] Fornica A. Similarity of XML-Schema Elements: A Structural and Information Content Approach[J]. *The Computer Journal*, 2008, 51(2): 240-254.

[16] Duta A C, Barker K, Alhajj R. RA: An XML Schema Reduction Algorithm[C]. In: *Proceedings of ADBIS*. 2006.

[17] Thang H Q, Nam V S. XML Schema Automatic Matching Solution[J]. *International Journal of Computer Systems Science and Engineering*, 2008, 4(1): 68-74.

[18] Do H H, Rahm E. COMA: A System for Flexible Combination of Schema Matching Approaches[C]. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002: 610-621.

(作者 E-mail: leeshuqing@163.com)