基于 Hadoop 平台的 XML 文档重复数据检测①

李振兴, 刘 波

(暨南大学 信息科学技术学院学院, 广州 510632)

摘 要: XML 数据越来越广泛地被用于信息交换与集成中, 其数据质量问题引起了人们的关注. 解决由数据质量引发的问题, 实体识别技术非常关键. 为了克服现有方法的不足, 在海量 XML 数据上进行高效的重复对象检测, 以实体识别技术为基础提出了基于 Hadoop 平台的 XML 文档重复检测算法, 它将所有标签节点统称为属性, 用实体来描述属性, 通过属性的比较, 快速地找到在某些属性上相同的所有实体对象, 并利用 Hadoop 应用框架处理海量数据的优势实现并行处理. 经过试验验证该方法良好的扩展性, 伸缩性和高效性.

关键词: XML; 数据质量; 重复检测; Hadoop; 分布式

XML Data Duplicate Detection Based on Hadoop Platform

LI Zhen-Xing, LIU Bo

(College of Information Science and Technology, Jinan University, Guangzhou 510632, China)

Abstract: As being more and more widely used for data exchange and integration, the XML data quality issues cause more concern. In order to overcome the problems caused by data quality, Entity Resolution(ER) is critical. To overcome the drawbacks of current methods's deficiency and perform entity resolution efficiently and effectively on massive XML data set, under the basis of Entity Resolution, an XML data duplicate detection based on hadoop platform algorithm is presented in this paper. The method uses entities to describe their atrributes. By the comparing of the attributes,we can find all the objects that have the same attributes quickly. Meanwhile, taking the advantage of the Hadoop platform which can process massive data parallel. From the experiments, the method has excellent performance in scalability, flexibility and efficiency.

Key words: XML; data quality; duplicate detection; Hadoop; distribute

xml 已经成为 Web 上数据交换与数据集成的主要标准, 其主要原因是其良好的可扩展性, 以及其结构的灵活性. 由于使用日趋广泛, xml 文档的数据质量问题引起了人们的关注, 如输入错误、各数据源中数据的结构差异、表达方式不同等因素令 xml 文档中存在着大量的数据冗余, 从而对数据发布、数据交互、数据集成及数据挖掘等操作产生不良影响[1].

为了克服由数据质量引发的问题, 研究人员已经提出许多技术[2-5], 其中实体识别起着重要作用. 实体识别是找到那些指向相同实体的数据对象. 当实体识别被应用于 xml 数据时, 最为关键的操作是实体数据对象的匹配. 实体数据对象匹配是指判断两个数据对象是否指向同一真实世界的实体. 实体识别技术的许多方法已经在关系数据上提出并使用[6-9], 但是由于 xml 数据结构多样, 致使这些方法并不适用. 近年来, 一些在 xml 数据上识别实体的方法被提出来.

第 1 种方法为了解决元素多样性问题[10], 先通过采用 XQuery 语言将结构不同的 xml 数据元素变换成统一的结构, 再将同一层次的内容合并为一个元素来处理. 但这种方法使原本具有不同标签的数据之间也进行相似性计算, 因此得到的精度不高.

第 2 种方法将 xml 数据抽象形成为一个树[1,11], 通过计算树之间的编辑距离得出对象相似度. 这种方法的时间复杂度在 $O(|T(e)|*n*n)$数量级上, 其中 e 为实体

① 收稿时间:2013-04-22;收到修改稿时间:2013-05-28

对象元素，*n* 为实体数量. 若实体的描述信息庞大，这种方法的效率将会非常低.

第 3 种方法将具有相似结构的 xml 元素进行合并[12]，提出了三类启发式聚类算法，分别使用全部比较聚类、选择比较聚类、M 树聚类方法来实现重复元素的合并. 主要不足在于没有解决 XML 数据结构多样性的问题.

总而言之，现有方法的缺点可以归纳为三点：第一，这些方法需要将所有实体对象一一比较，效率不高；第二，这些方法不能区别对待各个属性及结点，从而产生的精确度不高；第三，现有方法可扩展性差，不适用于海量数据.

为了克服现有方法的不足，在海量 xml 数据上进行高效的重复对象检测，我们提出一种基于实体描述属性的高效 xml 重复数据对象检测方法. 这种方法将待识别元素定义为实体对象，将标签、属性和子元素都看作实体属性，用实体属性来辨识实体. 此方法的优势体现在仅需比较在某些属性上值相同而无需比较所有的实体对象，并利用 Hadoop 分布式应用框架的优势进行处理，从而极大地提高了效率. 这种方法还通过抽取属性与结点值降低了 xml 数据中描述实体对象结构的复杂性.

1　基于Hadoop平台的XML文档重复数据检测系统结构

Hadoop 是由 Apache 基金会开发，并由其开源组织的一个分布以在大量廉价的硬件设备组成的集群上运行应用程序，为应用程序接口，同时用户在充分利用集群的威力高速运算和存储来开发分布了解分布式底层细节. Hadoop 系统是由 Java 程序语言开发出来提供了 C++、Java、Shell Command 等语言开发的访问界面. 使用中的插件相结合可以快速开发程序，程序员也可以将 Windows 操的程序安装到 Hadoop 的 Linux 集群上运行以完成分布式计算任务搜索以及数据处理，均能在 Hadoop 平台上方便的运行[13].

Hadoop 是一个能够对大量数据进行分布式处理的软件框架. 其保证了处理的可靠性、高效性、可伸缩性. Hadoop 是可靠的，因为它假设计算元素和存储会失败，为此，它维护了多个工作数据副本，以保证能够针对失败的节点重新分布处理. Hadoop 是高效的，因为它的工作运行方式是并行的，可以通过并行处理以加快处理数据的速度. Hadoop 还是可伸缩的，对于 PB 级数据也能够进行处理. 除此之于普通的 PC 机器上，对硬件要求不高，故成本低，任何人可以使 Hadoop 的核心是 HDFS 分布式文件系统、Map/Reduce 分布式 HDFS 提供数据存储，使用 Map/Reduce 实现并行数据处理.

使用 Hadoop 平台来对大规模 XML 文档数据进行重复检测能够有效满足应用需求. 一个 reduce 任务的数据流如图 1 所示. 虚线框表示节点，虚线箭头表示节点内部的数据传输，而实现箭头表示节点之间的数据传输.

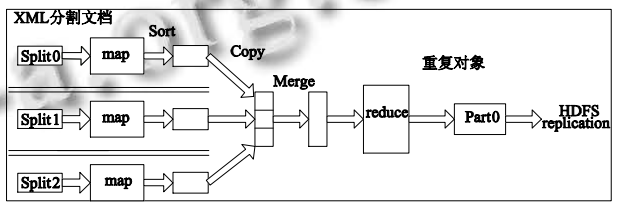

图 1　mapreduce 数据流

2　重复检测方法

2.1 待处理数据的预处理

在实体识别过程中，XML 数据不规则可能会引发一系列问题. 例如，没有主键便无法直接定位到对象，两个不同的 XML 数据源中对某个属性的属性名表述可能不同，而两个相同的属性名在不同的数据源中可能代表不同的意义，对象的父级及以上的结点降低了对象的检索速度. 为了利用本文的算法进行重复检测，需要对原始数据进行预处理，下面以图 2 中两个数据源的原始数据为例对预处理过程进行了说明.

```
数据源1                                  数据源2
<title >Java 编程思想</ title>           <book>算法导论</ book>
<time >                                  <author>本特利</ author >
    <year >2010</ year >                 <year>2011</ year >
    <month>09</ month>                   <month>05</ month>
</ time >                                < publisher>人民邮电出版社</ publisher>
<author>埃克尔</ author>                 <ISBN>036598715</ ISBN >
<publisher>机械工业出版社</ publisher>
<ISBN>023536985</ ISBN>
```

图 2　不规则源文件

第一步，使对象的所有属性处于同一层上.

实体对象具有的复杂层次结构降低了对象的处理效率，可以递归的从最里层的属性节点提升至父节点层次直至所有属性处于同一层上. 提升子节点时其父节点可以简单删除，以子节点来代替父节点. 如图 3 所示，将 time 的子属性提升至父节点层次，time 节点简单删除，不影响数据的有效性和正确性.

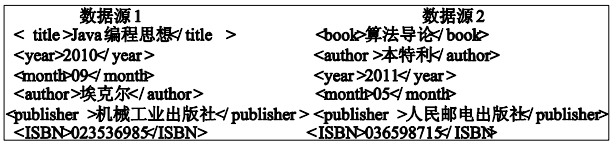

图 3 消除子节点

第二步, 统一属性名.

两个不同的 XML 数据源中对某个属性的属性名表述可能不同, 两个相同的属性名在不同的数据源中可能代表不同的意义. 我们的算法需要在不同的实体的相同的位置上代表相同的属性, 即意义相同. 因此有必要对意义相同但属性名不同的属性进行修改. 如图 2 中的 book 和 title 意义相同, 可将 book 改为 title,如图 4 所示.

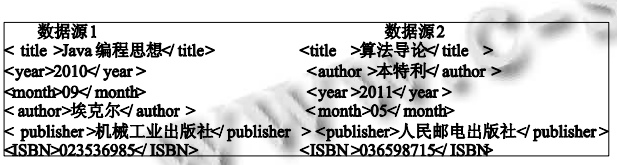

图 4 修改属性名

第三步, 调整属性间相对位置.

通过以上两步我们实现了使属性处于同一层和相同意义的属性具有了相同的属性名, 但我们的算法需要不同实体间的相同位置的属性有相同的意义, 以利于比较, 提高效率. 如修改图 5 使图 4 中的相同属性名处于相同位置.

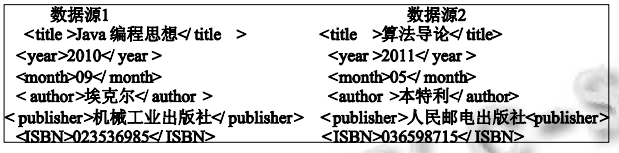

图 5 调整属性顺序

第四步, 将实体各属性改造成记录形式.

经过以上步骤, 我们的实体对象的属性名及属性位置都相同, 并且属性个数也相同(个数若不同以属性数最大者为准). 在此条件下, 完全可以将属性名及其符号去掉, 保留其属性值, 并以逗号分隔, 每个实体占一行, 因为我们的算法要比较的是其属性值. 如图 6 所示.

2.2 重复数据的检测算法

通过预处理我们将源文件改造成记录的形式, 各属性间用逗号分隔, 类似于数据库中的记录, 但无主键. 每行(即记录)代表一个数据对象, 两个数据对象如

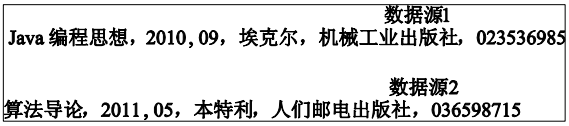

图 6 预处理后源文件

果描述同一个真实世界实体, 尽管它们很可能在某些属性的表现形式上有所不同, 但是它们的某些属性值很可能相同. 因此, 我们提出了重复检测算法, 该算法按照属性组织对象, 把在某个或某些属性上相同的对象放在同一个集合中, 这样就可以通过一次对数据源的遍历操作, 把各个对象插入相应的集合中, 然后只比较在对应属性上相同的对象.

同一对象的各属性对于对象的描述重要性可能不同, 为了更加精确地识别实体对象, 我们可以根据其不同重要性或人们关注的其特定属性来比较其是否相同, 假如实际需要某个或某些属性相同即认定两对象指向同一实体的话可以在输入时指定(即 *condition*). 实例中的 *condition* 指定为(1,2,3), 代表如果两对象的前三个属性相同就可以认定两对象指向同一实体, 大大增强了灵活性, 提高了扩展性. 算法 XML_RunDel 描述了检测过程.

算法名称: XML_RunDel

输入: 经过预处理的源文件 filename 及属性号组 condition

输出: 重复的对象数和重复对象组

方法:

```
1) map(string filename,string document){
2) list<string> T=tokenize(document,condition);
3) For each line in T {
4) Emit((string)condition,(string)line);//输出(key,value)对(condition,line)
5)          }
6)       }
7) Reduce(string condition,list<string> lines){
8) Integer sum=0;
9) For each line in lines {
10) sum=sum+1;                          //计算具有相同 key 的记录数
11)         }
12)      If(sum>1)                          //保留大于 1 的(key,value)对
```

13) Emit((Integer)sum,(string)lines);

14) }

15) }

例如: 源文件如下:

Java 编程思想,2010,09,埃克尔,机械工业出版社,023536985

算法导论,2011,05,本特利,人们邮电出版社,036598715

Java 编程思想,2010,09, 埃克尔,机械工业出版社,023536985

计算机网络,2004,01,Tanenbaum,清华大学出版社,730207815

假设 condition 为(1,2,3), 即如果属性的前三个属性相同即认定两对象表示同一实体.

按照以上算法步骤:

由 map 函数得到:

(Java 编程思想,2010,09,{(Java 编程思想,2010,09,埃克尔,机械工业出版社,023536985)})

(Java 编程思想,2010,09,{(Java 编程思想,2010,09,埃克尔,机械工业出版社,023536985)})

(算法导论,2011,05,{(算法导论,2011,05,本特利,人们邮电出版社,036598715)})

(计算机网络,2004,01,{(计算机网络,2004,01,Tanenbaum,清华大学出版社,730207815)})

由 reduce 函数得到:

2 (Java 编程思想,2010,09,埃克尔,机械工业出版社,023536985),(Java 编程思想,2010,09,埃克尔,机械工业出版社,023536985)

其中 2 表示有两项重复, 随后是两项重复记录.

3 实验验证

本文提出的算法是基于 hadoop 的核心算法 Map-Reduce 和分布式文件系统 HDHS 来实现并行计算, 充分利用多台处理机的计算能力, 能够极大提高效率. 为了验证本算法的有效性并测试影响效率的主要因素, 我们进行了大量的试验, 并对试验结果进行分析. 试验采用的硬件环境为 Intel(R) Pentium(R) Dual CPU 2.00GHz, 内存 1G, 操作系统为 enterprise linux 6, 系统代码用 java 实现, 在 hadoop-1.0.4 上运行.

3.1 单机模式下算法效率试验

在本实验中, 我们利用数据生成系统生成 9 个(D1-D9)含有不同数量实体对象的的 XML 数据集合. 它们分别含有 100K, 200K, 400K, 800K, 1000K, 2000K, 4000K, 8000K, 10000K 个实体对象. 这些实体对象的冗余程度适中, 都为 5%. 数据集合中的每个实体对象的结构中不存在孙级及以下的结点. 子级结点共由 7-10 个属性组成.

在单机环境下我们的算法运行于以上 9 个含有不同数量实体的数据, 此时 Hadoop 会完全运行在本地, 不需要与其它结点交互, 也不使用 HDFS 和加载任何 Hadoop 守护进程. 单机模式下开发应用逻辑, 而不与守护进程交互, 避免引起额外的复杂性. 单机环境下的所需时间如图 7.

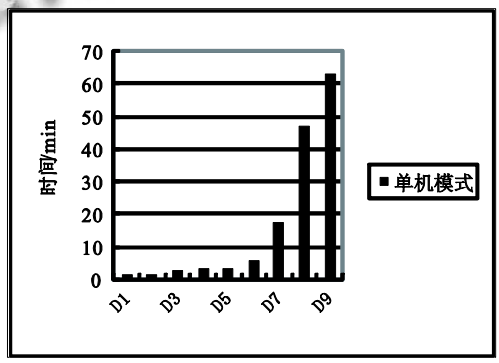

图 7 单机运行时间

由图 7 可见, 我们算法运行于含有 100K, 200K, 400K, 800K, 1000K, 2000K, 4000K, 8000K, 10000K 个实体对象的数据进行重复对象检查时, 所需要的时间分别为 1.2min,1.3min,2.6min,3.1min,3.5min,6.1min, 17.4min,47.3min,63.2min. 可见, 当数据中冗余的对象比率大致相同时, 数据集合的数量增大时, 重复检测所需要的时间显著增加, 同时, 随着数据量的增大, 单位数据集合的重复检测所需的时间显著减少, 充分体现的了 Hadoop 平台处理大规模数据集的优势. 运行显示在所给的算法下重复检测的正确率达到 100%. 这些验证了算法的效率和有效性.

3.2 单机模式与分布式模式的比较试验

以上验证了在单机模式下算法的性能, 虽然是在 Hadoop 平台上运行, 但严格说不是分布式, 没有体现出 Hadoop 分布式计算的优势. 因此我们验证了在全分布式条件下的重复检测, 并和单机模式作对比, 全分布模式下能充分利用 Hadoop 的分布式存储和分布式计算的优势, 能够极大的提高效率, 并有很好的伸缩

性和扩展性. 我们的全分布模式采用了 3 台机器, 其中一台做为集群的主节点, 驻留 Namenode, JobTracker 守护进程和 SNN 守护进程, 另外两台作为集群的从节点, 主要用来计算, 驻留 DataNode 和 Tasktracker 守护进程. 计算时间对比如图 8 所示.

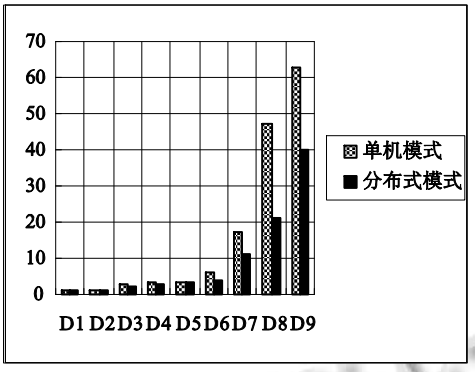

图 8 运行间比较图

由图 8 可以看出, 在数据量较小时单机模式和分布式模式的运行时间差别很小, 当数据量显著增大时分布式模式的计算时间相对于单机模式显著减少. 我们注意到在两台机器参与计算的条件下计算时间相对于单机速度并没有提高两倍, 这是由于我们的数据不是均匀的存储在集群中, 这就造成了数据的传输开销, 另外还有管理开销和启动开销. 同单机模式一样, 分布式模式下的重复检测的正确率也是 100%, 同时速度提升显著, 充分验证了本算法的高效性和有效性.

4 结论

本文提出了基于实体描述属性的分布式重复检测算法, 创新性在于将 hadoop 计算框架应用于海量 XML 数据的重复检测, 并提出了适合于 MapReduce 处理的算法, 并成功地应用于单机和分布式计算环境中, 该算法能够充分利用 Hadoop 中的 Map-Reduce 数据处理模型, 能够很容易地扩展到多个计算节点上处理数据, 实现分布式计算, 从而极大的提高了对大批量数据的处理速度, 经过实验验证, 该算法在 Hadoop 下表现出了其高效性, 扩展性和正确性. 进一步的工作就是实现重复数据的近似匹配, 本算法在比较属性时要求某个属性值完全相同时才为相同, 有些情况下是其属性值不完全相同但其意义相同, 如果能实现近似匹配就能更好的适应实际需要.

参考文献

1 Hernandez MA, Stolfo SJ. Real-world data is dirty:data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery, 1988, 2(1): 9–37.

2 Hassanzadeh O, Sadoghi M, Miller RJ. Accuracy of approximate string joins using grams. Proc. of the International. Workshop on Quality in Database(QDB). Vienna, Austria. 2007.11–18.

3 Hassanzadeh O. Benchmarking declarative approximate selection redicates. University of Toronto, Canada, 2007.

4 Whang SE, Menestrina D, Koutrika G. Entity resolution with iterative blocking. Proc. of the 35th SIGMOD International Conference on Management of Data. Rhode Island, USA. 2009. 219–231.

5 Weis M, Naumann F. Detecting duplicate objects in XML documents. Proc. of the IQIS. Pairs, France. 2004. 10–19.

6 Weis G, Naumann F. DogmatiX tracks down duplicates in XML. Proc. of the ACM SIGMOD 2005. New York, USA. 2005. 431–442.

7 Pluempitiwiriyawej C, Hammer J. Element matching across data-oriented XML sources using a multi-strategy clustering model. Data&Knowledge Engineering, 2004,48(3): 297–333.

8 王天亮,陈刚,徐宏炳.基于对象树相似匹配的 XML 重复对象检测.计算机科学,2006,33(11):162–166.

9 Karr AF. Exploratory data mining and data cleaning. Journal of the American Statistical Association, 2006, 101(473): 399–399.

10 Low WL, Lee ML, Ling TW. A knowledge-based approach for duplicate Elimination in data cleaning. Information Systems, 2001, 26(8): 585–606.

11 Marcel W, Roberto R. Efficient topology-aware overly network. ACM SIGCOMM Communication Review, 2003, 33(1): 101–106.

12 Aebi D, Perrochon L.Towards improving data quality. Proc. of the International Conference on Information Systems and Management of Data. Delhi, India. 1993.273– 281.

13 任宣萱.基于 hadoop 平台的作业调度研究[博士学位论文]. 天津:天津师范大学,2011.