

Web Service 架构下的分布式关联规则挖掘研究

剧立伟, 姜 浩, 蒲安建

(东南大学 计算机科学与工程学院, 江苏 南京 211189)

摘 要:随着信息技术的不断发展,大量的分布系统和跨平台系统不断涌现,同时也给数据挖掘带来了一定难度。针对分布的、异构的应用环境,提出了一个基于 Web Service 架构的数据挖掘框架。该框架能极大地实现分布异构环境下的数据挖掘。在此框架基础上设计了一个改进的 PF 增长算法,算法主要针对分布异构环境下的关联规则挖掘并具有较高的效率,同时也证实了框架的可行性。

关键词:Web 服务;分布式数据挖掘;FP 增长算法

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2009)04-0031-04

Distributed Associational Rules Data Mining Based on Web Service

JU Li-wei, JIANG Hao, PU An-jian

(School of Computer Science & Engineering, Southeast University, Nanjing 211189, China)

Abstract: With the development of information technology, a number of distributed systems are presented, at the same time, it got some difficulties for data mining. for the distribution and heterogeneous environments, proposes a framework of data mining based on Web Service, which can greatly achieve the distributed data mining. On the basis of this framework, an improved FP-GROWTH algorithm is presented, which is effective and confirmed the feasibility of the framework.

Key words: Web Service; distributed data mining; FP-GROWTH algorithm

0 引 言

随着信息技术的不断发展,信息技术的应用也不断普及,出现了信息爆炸时代。如何在大量的信息中寻找有用的信息也就凸现地十分重要。近年来,数据挖掘作为一个年轻而又充满活力的学科得到了广泛的关注。简单地说,数据挖掘就是指从大量数据中提取或“挖掘”知识^[1]。随着研究的进展,人们也提出了很多研究成果,并极大地促进了数据挖掘的发展。

但是,传统的数据挖掘主要面向集中的数据源,在大量分散和异构环境下的数据挖掘主要还是采用集中转储到一个大的数据仓库的方式来进行的,效率低下,并且在数据传输过程中也存在安全隐患。事实上,现今很多数据源都是分散和异构的,一个企业的数据源可能运行在不同环境下,环境平台的互操作性很差,并且大量数据的转储也比较费时费力,一定程度上影响了企业的效率。如何在网络环境下在庞大分布和异构的数据源中挖掘信息也就成为了现在研究的热点问题。

Web Service 的出现给这个问题带来了一线曙光。

Web 服务是一种优秀的分布式计算技术,其是以独立于平台的方式,通过标准的 Web 协议,可以有程序访问的应用程序逻辑^[2]。Web 服务是基于组件的分布式技术变革的必然产物,与 DCOM 和 CORBA 相比,其在跨平台、跨 Internet 和适应 Internet 可伸缩性方面有强大的优势,能尽可能地达到复用和互操作的目的。将数据挖掘和 Web 服务结合起来将极大地提高挖掘的灵活性和效率,最大程度上提升企业的效益。

文中在此基础上提出了一个基于 Web 服务的数据挖掘架构,提出了一个改进的 PF 增量关联规则挖掘算法,一定程度上证实了该架构的可行性。

1 Web 服务介绍

1.1 Web 服务基本原理

Web 服务其实就是位于应用程序代码和应用程序用户间的一个接口^[3]。它的作用相当于一个抽象层,将应用平台与编程语言相关的细节分隔开,其相当于一个黑匣子,可以被用户使用而不需要关心它如何实现。这意味着任何支持 Web 服务的语言都可以访问应用程序提供的功能。

收稿日期:2008-07-21

作者简介:剧立伟(1983-),男,河北唐山人,硕士研究生,研究方向为数据挖掘、Web Service;姜 浩,副教授,研究方向为工作流应用研究、Web 挖掘研究等。

1.2 Web 服务体系结构

Web 服务主要依赖 SOAP、WSDL 和 UDDI 三者的交互。它的体系结构如图 1 所示。Web 服务提供方通过 WSDL 描述所提供的服务,并将这一描述告知 Web 服务注册中心。注册中心根据 WSDL 的描述,依照 UDDI 的协议更新服务目录并在 Internet 上发布。用户在使用 Web 服务前想要向注册中心发出请求,得到 Web 服务提供者的地址和服务接口信息,然后使用 SOAP 协议与服务提供者建立连接,绑定服务后进行通信。

Web 服务的技术主要建立在 XML 的规范上,这保证了这一体系结构的平台无关性、语言无关性和人机交互性能。SOAP 规范中定义了 SOPA 协议与 HTTP 协议的绑定规定,这主要是由于 HTTP 协议时 Internet 上应用最广泛的通信协议,可以穿越多数防火墙。Web 服务可以将企业的应用重新包装和发布。企业内部可以根据需要构造自己的服务平台。SOAP 处理器将后台应用程序处理后得到的数据打包成一个符合 SOAP 规范的 XML 文档,通过 HTTP 发送给 Web 服务的用户。而用户发出的请求也是以符合 SOAP 规范的 XML 文档发出。由此可见,用户不需要了解远程服务的细节就可以使用服务,整个交互都以 XML 文档方式进行,使服务的实现有了极大的灵活性。

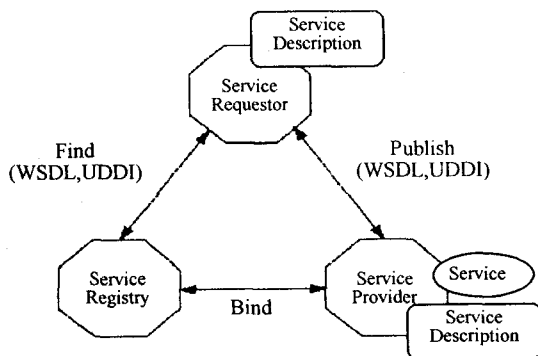


图 1 Web Service 的体系结构

2 基于 Web Service 的数据挖掘框架

该体系结构建立在 Web 服务分布式计算体系之上,各个局部数据挖掘模块都是独立作为 Web 服务注册和发布的^[4](如图 2 所示)。它具有良好的可移植性,能够跨越平台和数据结构的差异性,并能跨越防火墙和服务器通信,使用灵活,具有很好的用户透明性^[5]。该体系结构主要包括以下几个模块:

* 用户 GUI:与用户交互的界面,可以订制各种期望的服务,并展现结果。

* 全局汇总模块:根据局部数据挖掘处理的结果进行分析处理,最终得到期望的全局知识表示。

* 注册中心模块:注册中心主要完成服务的注册和查询功能,用户可以从注册中心获取相关的服务或发布自己的挖掘服务。

* 对外服务模块:完成服务的封装,并向注册中心注册,完成对服务请求的绑定。

* 本地挖掘模块:完成对本地数据源的数据挖掘,作为服务提供给请求者。

* 预处理:按照一定规则完成对本地数据源的转换和抽取等功能。

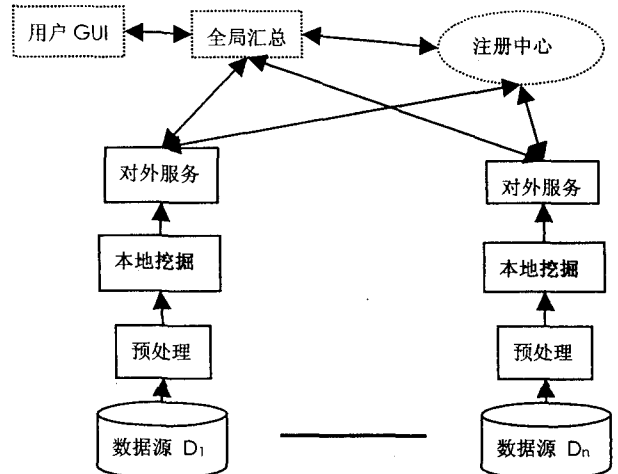


图 2 基于 Web service 的挖掘框架

3 关联规则挖掘

3.1 相关概念

定义 1:交易集: $T = \{t_1, t_2, \dots, t_m\}, t_i (1 \leq i \leq m)$ 为 DB 中的交易。

数据项集: $I = \{i_1, i_2, \dots, i_n\}, i_j (1 \leq j \leq m)$ 为 T 中的数据项集。

项集:如果 $IS \subseteq I$,就称 IS 为项集。

设 $X, Y \subseteq I$, 且 $X \cap Y = \emptyset$, 如一笔交易中既含有 X , 又含有 Y , 称 $X \Rightarrow Y$ 。

定义 2:支持度:如果 $X \Rightarrow Y$ 在 T 的 $S\%$ 的交易中成立,即称 $X \Rightarrow Y$ 的支持度为 $S\%$ 。

或者 $\text{Support}(X \Rightarrow Y) = P(X \cup Y)$ 。

$$S\% = \frac{||t|t \text{ 中含有 } X, Y||}{|t|} \times 100\%$$

如果 $X \Rightarrow Y$ 的支持度超过用户给定的最小支持度阈值 ($\min \sim \sup$), 就称 $|X, Y|$ 为频繁项集。

定义 3:可信度:说明 $X \Rightarrow Y$ 的必然程度。 $\text{confidence}(X \Rightarrow Y) = P(X \mid Y)$, 即: T 中含有 X 同时含有 Y 的交易为 $C\%$ 。

$$C\% = \frac{||t|t \text{ 中含有 } X, Y||}{||t|t \text{ 中含有 } X||} \times 100\% = \frac{S(X \Rightarrow Y)}{S(X)}$$

$\times 100\%$

如果 $X \Rightarrow Y$ 的支持度和可信度都超过用户给定的最小支持度和最小可信度 ($\min\text{-conf}$) 阈值, 就称 $X \Rightarrow Y$ 为关联规则。

性质 1: 频繁项集的任何非空子集都是频繁的。

根据定义, 如果项集 IS 不满足 $\min\text{-sup}$, 则 IS 不是频繁的。即 $P(IS) < \min\text{-sup}$ 。如果项 A 添加到 IS , 其结果项集 (即 $IS \cup A$) 不可能比 IS 更频繁出现。因此, $(A \cup IS)$ 也不是频繁的。即: $P(IS \cup A) < \min\text{-sup}$ 。

推论 1: 如果项集 IS 中包含一个不是频繁项集的非空子集, 则 IS 必定不是频繁项集。

性质 2: 一个频繁项集至少在一个局部挖掘中是频繁项集。

定义 4: 局部数据库中含有的交易数记为 $\text{Loc-}C$ 。

定义 5: 局部最小支持度: $\text{Loc-}\min\text{-sup} = \min\text{-sup} \times \text{Loc-}C$

定义 6: 局部支持度计数 $\text{Loc-}SC$: 局部数据库中包含某个频繁项集的交易数。

定义 7: 全局支持度计数 SC : 数据库中包含某个频繁项集的交易数。

$$SC = \sum_{i=1}^N \text{Loc-}SC_i, \text{ 其中 } N \text{ 为分布数据源数目。}$$

发现关联规则需要经历以下两个步骤: (1) 找出所有频繁项集; (2) 根据频繁项集生成满足最小可信度阈值的关联规则。

3.2 DEX-PF 算法

作为关联规则挖掘的经典算法 Apriori 算法, 众所周知效率较差, 其主要的时间消耗在频繁集的生成上, 在这种分布的环境下, 其效率会更差。考虑到这个原因, 对一些改进的分布算法进行了探讨^[6], 而文中采用了 FP 增长算法来发现频繁模式。

FP 增长算法^[1]主要采用分治策略。首先, 将提供频繁项的数据库压缩到一棵频繁模式树 (FP-tree), 但保留项集关联信息。然后, 将压缩后的数据库划分成一组条件数据库, 每个关联到一个频繁项或者“模式段”, 并分别挖掘每个条件数据库。FP 增量算法只需要对数据库进行两次扫描, 而且避免了产生大量的候选集的问题。研究表明, FP 增量算法对挖掘长和短的频繁模式, 都是有效的和可规模化的, 并且大约比 Apriori 算法快一个数量级。

上述算法主要讨论了集中环境下的情况, 针对文中的分布环境下的情况, 对 FP 增量算法进行了改进, 称为 DEX-PF 算法, 即分布环境下的扩展 PF 增长算法。该算法只需要三次扫描数据库, 在分布环境下具有较高的效率。

DEX-FP 算法描述:

输入: 最小支持度阈值 $\min\text{-sup}$ 。

输出: 全局频繁项集。

1) 用户输入 $\min\text{-sup}$, 全局汇总模块向各个局部模块发出挖掘请求。

2) 各个局部挖掘模块扫描各自本地数据源 DB_i , 统计出各自的交易数 $\text{Loc-}C_i$ 和各个项的局部支持度计数 $\text{Loc-}SC_i$ 。然后, 发送到全局汇总模块处理。

3) 全局汇总模块根据收集的局部交易数和局部支持度计数, 计算出全局频繁 1 项集。对于某个项 K ,

$$\text{如果 } \frac{\sum_{i=1}^N \text{Loc-}SC_i(k)}{\sum_{i=1}^N \text{Loc-}C_i} \geq \min\text{-sup} \text{ 成立, 则项 } K \text{ 属于全局频繁 1 项集。}$$

计算出所有全局频繁 1 项集后, 全局汇总模块将其发送到各个局部挖掘模块处理。

4) 局部挖掘模块根据全局频繁 1 项集, 再次扫描数据库, 根据 FP 增长算法生成各自的局部频繁项集。在构造 FP-tree 时, 采用修剪办法。例如, 全局频繁 1 项集按照局部支持度降序排序后为 $\{ABD\}$, 而某条交易记录按照排序为 $\{EBFD\}$, 则只需要根据 $\{BD\}$ 在构造 FP-tree。根据推论 1, 可以知道这个裁减办法是正确的, 一定程度上也减小了 FP-tree 的规模。

5) 各个局部模块计算出局部频繁项集后, 将其以及其局部支持度计数发送到全局汇总中心处理。

6) 全局汇总模块根据收集的局部频繁项集和局部支持计数, 将频繁集按照包含的项数降序排序, 按照类似第三步的思路, 对某个频繁项集 K , 如果其满足

$$\frac{\sum_{i=1}^N \text{Loc-}SC_i(k)}{\sum_{i=1}^N \text{Loc-}C_i} \geq \min\text{-sup}, \text{ 则其属于全局频繁项集。}$$

根据性质 1, 频繁项集 K 的所有非空子集也属于全局频繁项集, 可以归并到全局频繁项集中。如果剩余的局部频繁集为空, 转到第 9 步。

7) 第 6 步中裁减剩下的局部频繁项集是可能的全局频繁集, 全局汇总模块把它们再次发送到局部模块, 由局部模块计算出局部支持计数送回。

8) 全局汇总模块根据支持计数结果计算出剩余的全局频繁项集。

9) 输出全局频繁项集。

3.3 DEX-FP 算法示例

下面以一个例子详细说明 DEX-FP 算法。

假设有三个局部数据源 DB_1 、 DB_2 、 DB_3 , 各自有若干条记录, 如表 1 所示。设定最小支持度为 40%, 计算过程如下:

表 1 数据库示例

DB ₁	AE	ABDE	BCE	B	
DB ₂	BDE	E	ABDE	AE	BDE
DB ₃	DE	CD	BE	BDE	

1) 各个局部挖掘模块计算各项的支持度计数,由全局汇总模块处理,计算出全局频繁 1 项集合。结果如表 2 所示。

表 2 全局频繁 1 项集

	DB ₁	DB ₂	DB ₃	全局计数	是否频繁 1 项集
A	2	2	0	4	否
B	3	3	2	8	是
C	1	0	1	2	否
D	1	3	3	7	是
E	3	5	3	12	是

2) 根据全局频繁 1 项集 B,D,E,各个局部模块用 PF 增长算法计算局部频繁集。结果如表 3 所示。

表 3 局部频繁项集

DB ₁	BE:2		
DB ₂	BD:3	DE:3	BDE:3
DB ₃	BE:2	DE:3	

根据表 3 可以知道 DE 属于全局频繁集。剩余的是候选频繁集,返回各个局部挖掘模块计算局部计数,计算结果如表 4 所示。

表 4 候选频繁项集计算结果

	DB ₁	DB ₂	DB ₃	全局计数	是否全局频繁项集
BE	2	3	2	7	是
BD	1	3	1	5	是
BDE	1	3	1	5	是

3) 最终输出全局频繁项集为: {B}、{D}、{E}、{BD}、{BE}、{DE}、{BDE}。

(上接第 30 页)

时的需求,能用在数控等实时性要求高的系统中去。

6 结束语

文中讨论了标准 Linux 在实时性方面的不足,研究了基于 ADEOS 的 RTAI 的实现机制,并对 RTAI-Linux 进行了构建和实时性能测试,结果表明 RTAI-Linux 的实时性能够满足硬实时系统的要求。目前,RTAI-Linux 在数控系统等对实时性要求高的领域已经获得了一定的应用。文中的创新点在于将 ADEOS 与 RTAI 应用到标准 Linux 上,提高 Linux 实时性能。

参考文献:

- [1] 阮鸿芳,钟家骥. Linux 与硬实时扩展系统——RTAI 的分析与研究[J]. 微计算机信息,2007(2):44-46.
- [2] Mantegazza P. DIAPM RTAI Programming Guide 1.0[EB/

4 结束语

分布式、网络化是数据挖掘的发展趋势之一。文中基于 Web Service 架构设计了一个分布式数据挖掘体系,并在此基础上探讨了一个扩展的 FP 增长算法,该算法具有一定的实用价值和效率,并且可以改进后作为并行算法的一种实现^[7]。但是,基于分布环境和 Web 服务的复杂性,算法还有许多细节需要进一步探讨,比如如何解决数据传输效率问题,如何组合不同挖掘服务等问题。但随着信息技术的不断发展,作为一种发现信息的有效手段,相信该技术会得到更大的发展。

参考文献:

- [1] Han JiaWei, Kamber M. 数据挖掘—概念与技术[M]. 北京:机械工业出版社,2007.
- [2] 顾宁. web service 原理与研发实践[M]. 北京:机械工业出版社,2006.
- [3] 梁宇奇,柴晓路. Web Service 技术、架构和应用[M]. 北京:电子工业出版社,2003.
- [4] 侯敬军. 基于 Web 服务的分布式数据挖掘系统研究[D]. 武汉:华中科技大学,2005.
- [5] 李爱军,郭学俊. 基于 Web 服务的异构数据交换方案设计与实现[J]. 计算机技术与发展,2006,16(7):79-81.
- [6] 金春霞,白秋产. 基于 Web Service 技术分布式并行数据挖掘的研究与实现[J]. 现代电子技术,2008,31(10):42-44.
- [7] 张潇,恽爽,陆桑璐,等. 并行数据挖掘研究[J]. 计算机工程,2003,29(17):58-59.

OL]. 2007-01-10. <http://www.rtai.org>.

- [3] 白小明,邱桃荣. Linux 的嵌入式实时操作系统的研究[J]. 微计算机信息,2006(2):78-80.
- [4] 毛得操,胡希明. Linux 内核源代码情景分析[M]. 杭州:浙江大学出版社,2001.
- [5] 陈文星,张辉宜,陶陶,等. 嵌入式 Linux 的实时性改进技术[J]. 计算机技术与发展,2006,16(10):114-117.
- [6] 马季兰,刘勇. 嵌入式 Linux 操作系统的实时性研究[J]. 计算机技术与发展,2007,17(8):80-83.
- [7] 李小群,赵慧斌,叶以民. 基于 Linux 的实时操作系统[J]. 计算机学报,2003,14(7):1203-1212.
- [8] Haute T. Fedora Core 3 2.6.9-adeos HOWTO[CP/OL]. 2005-01. <http://www.csrose-hulman.edu/>.
- [9] Monteiro J. A Guide RTAI Installation Complete Guide [CP/OL]. 2008-02. <https://www.rtai.org>.
- [10] 褚文奎,张凤鸣,樊晓光. 嵌入式 Linux 系统实时性能测试研究[J]. 系统工程与电子技术,2007(8):1385-1388.