摘 要

随着 Internet 的飞速发展，网上的数据资源空前丰富。每天都会有成千上万的用户在网络上浏览和寻找自己所需的信息。然而，由于信息量的庞大，对于每个用户来说，如何能够及时快速地发现有用信息则变得异常困难。为了解决上述问题，Web 挖掘技术应运而生。其中，面向 Web 服务器日志的 Web 使用挖掘技术尤其得到了广大研究人员的关注。Web 日志数据记录了用户对 Web 站点的访问信息，对这些信息进行分析可以发现用户访问站点的浏览模式和访问习惯，对于页面重组、优化网站的结构，以及在电子商务智能的应用等方面都具有十分重要的意义。

本文对 Web 挖掘与 Web 使用挖掘进行了系统的分析和研究，并在已有研究的基础上改进并提出了两个新的算法。

本文的工作主要有以下几个方面：

(1) 对 Web 挖掘的基本理论知识和分类进行了总体研究，重点分析研究了 Web 使用挖掘的基本思想和经典算法。

(2) 在分析关联规则经典算法 Apriori 的基础上，提出一种基于事务矩阵的关联规则挖掘算法，通过将事务数据库映射为一个事务矩阵，对事务矩阵进行操作以得到所有的频繁项目集。理论分析和实验证明了新算法在性能上的优越性。将新算法应用于 Web 使用挖掘可以高效地发现用户之间、页面之间以及用户浏览页面和网上行为之间存在的潜在关系。

(3) 提出一种基于有向图的用户频繁访问模式挖掘算法，通过对 Web 事务数据库进行一次扫描，将所有页面之间的序列信息记录在有向图中，并从中挖掘所有的用户频繁访问模式。利用挖掘出的模式知识，可以帮助预测网页的访问情况，从而可以帮助合理地放置广告以针对特定用户群。

关键词：Web 挖掘；Web 使用挖掘；关联规则；事务矩阵；序列模式；有向图；用户频繁访问模式

Abstract

With the rapid development of the Internet, digital resource on the Internet becomes more and more abundant. Thousand and thousand users browse and search useful information for them on the Internet everyday. But, it's very difficult for users to find useful information in time because of the huge communication on the Internet. To solve this problem, Web mining techniques emerge as the times require. Especially, lots of researchers pay more attentions to the Web usage mining which faces Web server logs. Web logs record the visit information of Web site visitors; Therefore, we can obtain the browsing behavior and visiting habit of the visitors by analyzing the Web logs, which are significant for the page recombination, the structure optimization of Web site, the capability improvement of Web system and the application enhancement of Electronic Commerce.

This thesis presents the systematic analysis and research on the Web mining and Web usage mining. Based on the existing reseach, two novel algorithms are improved and proposed in this thesis.

The main researches of this thesis are as follows:

(1) A general research on the basic theory and classification of Web mining is done, and then the basic idea and classical algorithms of Web usage mining are analysed and researched primarily.

(2) On the basis of analyzing Apriori, a classical algorithm of association rules, a novel algorithm for association rule mining is put forward based on transaction matrix. The new algorithm maps the trasaction database into a trasaction matrix, and operates the trasaction matrix to gain all the frequent item sets. The performance ascendant of the algorithm is proven by theory-analysis and experiment. The algorithm can be applied in web usage mining to effectively discover the potential relations among users or pages, and the association between user traversal path and behavior on Internet.

(3) Another novel algorithm of user frequent traversal patterns is proposed

based on digragh. The algorithm scans the trasaction database only once, records the information of the sequences among the whole web pages in a digragh, and mines all the user frequent traversal patterns based on the digragh. The visit of the web pages could be forecasted by using the mined patterns, and accordingly the advertisement can be placed with reason to suit specifical user group.

Key words: Web Mining; Web Usage Mining; Association Rules; Transaction matrix; Sequential Patterns; Digraph; User Frequent Traversal Patterns

西南交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复印手段保存和汇编本学位论文。

本学位论文属于

1．保密□，在　　年解密后适用本授权书；

2．不保密√，使用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：冯贺　　指导老师签名：

日期：2008．6．15　　日期：2008．6．15

西南交通大学学位论文创新性声明

本人郑重声明：所呈交的学位论文，是在导师指导下独立进行研究工作所得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中作了明确的说明。本人完全意识到本声明的法律结果由本人承担。

本学位论文的主要创新点如下：

(1) 在分析关联规则经典算法 Apriori 的基础上，提出一种基于事务矩阵的关联规则挖掘算法，通过将事务数据库映射为一个事务矩阵，对事务矩阵进行操作以得到所有的频繁项目集。理论分析和实验证明了改进后的算法在性能上的优越性。

(2) 提出一种基于有向图的用户频繁访问模式挖掘算法，通过对 Web 事务数据库进行一次扫描，将所有页面之间的序列信息记录在有向图中并从中挖掘所有的用户频繁访问模式。并在具有典型结构的 Web Log 数据上进行了用户频繁访问模式挖掘的试验。

第 1 章 绪 论

1.1 研究背景与意义

Internet 技术的发展和普及使信息获取和发布的方式发生了巨大的飞跃和本质性的变化，World Wide Web(简称 WWW)成为信息传播的主要载体之一。Web 页面散布在世界各地的 Web 服务器上，每个服务器自主地管理自己的资源。目前，Web 的信息容量已超过 Gopher 和 WAIS 而成为全球最大的信息系统。可以感觉得到，Web 的容量增长迅速，根据过去的统计，平均每两个小时增加一台服务器，每天增加数百万个 Web 页。

根据 Internet 软件协会(http://www.isc.org)的统计，到 2002 年 1 月，Web 主机的数量已经超过了 1 亿 4 千万台。而根据市场调研公司 eTForecasts 新近发表的一份调研报告称，2005 年全球有 11.7 亿人使用 Internet[1]。

就中国的情况而言，根据 CNNIC 于 2007 年 1 月 23 日发布的第 19 次“中国互连网络情况统计报告” [2]中的数据，中国大陆的上网用户总人数已经达到 1.37 亿，比去年同期增长了 23.4%。CN 下注册域名总数超过 180 万，与 2005 年同期相比，增长幅度达到 64.4%。本次报告新增了对我国网页数、网页字节数等资源的调查内容，结果显示，截至 2006 年底，全国网页数和网页字节总数分别为 44.7 亿个和 122,306GB，与去年同期相比分别增长 86.3%和 81.7%。

面对这样铺天盖地的网络信息量，似乎能够满足人们对于信息的需求。但实际情况是，对于用户来说，Web 上绝大多数的信息是毫无用处的。随着时间的推移，人们越来越感觉到这个数字时代的图书馆并不像真正的图书馆那样支持有组织的信息管理和检索。恰恰相反，它只是一个杂乱无章的信息仓库。在这个仓库中有文本、书刊、科研资料、图片、广告、录像和录音等，转瞬即逝的信息和有持久意义的重要资料混杂在一起。如何从这些巨量的 Web 数据中发现有用的知识是数据挖掘和知识工程研究面临的新课题。

Web 用户在日常活动中产生大量的信息，这些信息可以被 Web 服务器自动收集并存储在访问日志中。Web 服务器日志记录了用户与服务器的交互信息，对于掌握 Web 服务器的运行情况、分析用户需求、维护系统安全、辅助

站点维护人员优化站点具有重要作用[3]。对日志采用统计分析和联机分析处理(Online analysis processing, OLAP)的方法，可以对常用数据进行汇总，提供关于用户行为的统计报告。但为了在更深的层次上理解用户的行为和 Web 站点的结构，得到诸如用户的访问模式和兴趣爱好等有用信息，就要用到数据挖掘的方法，Web 使用挖掘就是从 Web 访问日志中获取用户访问 Web 的规律并预测用户的网上行为。

在 Web 挖掘中，最重要的应用就是 Web 使用挖掘，即通过挖掘 Web 服务器的日志文件，以发现用户访问站点的浏览模式，从而进一步分析和研究日志记录的规律，以改进网站的组织结构及其性能，构建智能站点，面向用户提供个性化服务，发现潜在的用户群体，为企业制定更有效的市场营销策略，从而获得更大的竞争优势[4]。因此，进行 Web 日志挖掘技术的研究具有非常重要的意义，具体主要包括以下几个方面[5]：

(1) 提供个性化服务

基于 Web 日志挖掘的个性化服务是指 Web 服务器从单个用户的浏览信息发现用户的兴趣，向每位用户提供符合其兴趣要求的个性化界面。简单流程就是：把当前用户的会话（或已经保存的该用户的简档）和通过 Web 日志挖掘得到的使用模式相匹配，得到当前用户的兴趣偏好，然后根据当前用户的兴趣偏好向其推荐一组对象，这些对象包括用户可能感兴趣的链接、广告、产品或其他服务等。

(2) 系统改进

用户对 Web 是否满意取决于 Web 的性能和服务质量，对 Web 系统的特性数据进行分析，有助于更好地研究 Web 缓存、网络传输、负载平衡或数据分布等策略，从而得出结论以供 Web 系统性能改进。如可以提供 Web 流量行为的分析，利用它来进行 Web 缓存、实现存取平衡等。另外，随着电子商务以指数形式增长，安全问题成为 Web 服务的重点，Web 日志挖掘也可以提供有用的挖掘模式来检测 Web 站点侵入、欺骗等。

(3) Web 站点辅助设计

一个 Web 站点能否在内容和结构方面吸引用户，对于很多 Web 商家来说是个很关键的问题。Web 日志挖掘为 Web 站点设计者提供了详细的用户反馈，帮助他们根据实际用户的浏览情况，调整 Web 站点的拓扑结构和内容，对 Web 站点进行优化，从而更好的为用户服务。

(4) 商业智能

消费者是如何使用Web站点的，这对于Web商家来说也是非常重要的信息。在电子商务网站，把Web日志和顾客交易信息相结合进行挖掘，能够发现关联购买集合、顾客的购买趋势，以及潜在顾客对商品的兴趣，从而对商品信息在页面显示上进行调整以方便顾客浏览和购买，为顾客推荐相关商品，预测顾客的购买兴趣，还可以把潜在的客户转变成为实际的购买者。

1.2 Web使用挖掘研究现状

1.2.1 国外研究现状

Web数据挖掘的研究应用工作，自1996年由M.S. Chen，H.Mannila，T. Yan，O. Etzioni等[6-9]提出开始，到现在已经有10多年的发展历程。通过大量学者、技术人员的努力，已经取得了很大的成绩，这其中大量工作源于国外的一些学者和研究机构。对于Web使用挖掘，Mannila和Chen在研究过程中都假定去掉了图形文件、声音等多媒体文件，这样剩下的Web服务器日志就如实反映了用户在网站中的访问情况。Mannila把用户访问页面当作事件，从网站访问日志中试着寻找用户访问网站的周期。Chen则提出了最大前向参引模型，同时也提出用这种方法来分解用户访问的Session作为一个个的事务(transaction)，然后就可以在事务的基础上，挖掘用户访问模式。T.Yan研究了如何动态地根据用户的当前访问提供推荐页面。他首先对用户进行分类，然后根据同类用户访问过的页面情况，决定为当前用户提供的页面内容。

Zaki[10]提出了挖掘频繁序列的SPADE(Sequential Pattern Discovery using Equivalence classes)算法，SPADE算法只需要对数据库进行3遍扫描就可以发现所有的频繁序列。SPADE算法包括3个主要的特征：将数据的保存形式由事务为中心的方式变为以数据项为中心的方式；利用格理论将搜索空间分为多个小的搜索空间，使得每个小的搜索空间可以在内存完成处理过程；采用两种搜索策略——广度优先和深度优先进行模式搜索。

Shahabi等[11, 12]的日志挖掘系统依赖于客户端的数据收集，客户端的代理服务器返回用户请求的页面以及时间等数据。

1.2.2 国内研究现状

Chen[13, 14]等首先将数据挖掘技术应用于Web服务器日志文件，以期发现用户浏览模式。提出最大向前引用（Maximal Forward Reference，MFR）系列

的概念。将用户会话分割成一系列的事务，然后采用与关联规则相似的方法挖掘用户频繁访问路径。

Han 等[15]根据 Web 日志建立数据立方体，然后对数据立方体进行数据挖掘和联机分析处理，并且给出一个关于 Web 日志挖掘系统 WebLogMiner。通过对 Web 站点的日志记录进行预处理，将日志数据组织成传统的数据挖掘方法能够处理的事务数据形式，然后利用传统的数据挖掘方法（如关联规则发现算法）进行处理，该系统已经实现了关联规则、分类以及时间序列分析。

国内的 Web 使用挖掘主要侧重于理论研究。西安交大的沈均毅等[16]提出以 Web 站点的 URL（Uniform resource locator）为行、以 UserID 为列，建立 URL－UserID 关联矩阵，元素值为用户的访问次数，然后，对列向量进行相似性分析得到相似客户群体，对行向量进行相似性度量获得相关 Web 页面，对相关页面进行下一步处理，以发现频繁访问路径。西安交大的陆丽娜等[17]采用基于事务的方法，研究 Web 日志挖掘预处理及用户访问序列模式挖掘方法，提出了一种基于扩展有向树框架进行用户浏览模式识别的日志挖掘方法。

通过对收集到的文献的分析以及本文的工作来看，关联规则挖掘和序列模式挖掘算法[18]还有进一步改进从而提高性能的空间。

1.3 本文研究内容及章节安排

1.3.1 研究内容

Web 使用挖掘主要有两个方面：用户访问模式挖掘和个性化挖掘。用户访问模式挖掘一般是从 Web 日志中挖掘用户的访问模式和预测用户的访问趋势；个性化挖掘则倾向于分析单个用户的偏好，使得网站更加生动而独特[19]。本文主要以 Web 服务器里的日志作为挖掘对象，通过从 Web Log 文件中挖掘关联规则，来帮助对站点结构进行管理及优化；通过挖掘用户在 Web 环境中的访问模式来完善站点结构，并进一步为用户提供个性化服务。

具体说来，本文的研究工作主要有以下几点：

(1) 对 Web 挖掘的基本理论知识和分类进行了总体研究，重点分析了 Web 使用挖掘的基本思想和经典算法。

(2) 在分析关联规则经典算法 Apriori 的基础上，提出一种基于事务矩阵的关联规则挖掘算法，通过将事务数据库映射为一个事务矩阵，对事务矩阵进行操作以得到所有的频繁项目集。理论分析和实验证明了改进后的算法在

性能上的优越性。将改进后的算法应用于 Web 使用挖掘可以高效地发现用户之间、页面之间以及用户浏览页面和网上行为之间存在的潜在关系。

(3) 提出一种基于有向图的用户频繁访问模式挖掘算法，通过对 Web 事务数据库进行一次扫描，将所有页面之间的序列信息记录在有向图中并从中挖掘所有的用户频繁访问模式。利用挖掘出的模式知识，可以帮助预测网页的访问情况，从而可以帮助合理地放置广告以针对特定用户群。

1.3.2 组织框架

全文共分 4 章。

第 1 章：介绍了本文的研究背景及研究意义，分析了国内外对 Web 使用挖掘的研究现状，引出了本文的研究主题和研究内容。

第 2 章：概述 Web 挖掘的定义、特点和分类，重点介绍 Web 使用挖掘的数据抽象和挖掘过程。

第 3 章：在对经典关联规则挖掘算法 Apriori 进行研究和分析的基础上，针对 Apriori 算法的缺点，提出了一种基于事务矩阵的改进算法，并理论分析和实验证明了算法在性能上的优越性。

第 4 章：提出了一种基于有向图的用户频繁访问模式挖掘算法，介绍了进行用户频繁访问模式挖掘的数据预处理过程，并详细介绍了算法在预处理得到的 Web 事务数据库上进行挖掘的改进思想和具体实现，最后在具有典型结构的 Web Log 数据上进行了用户频繁访问模式挖掘的试验。

结论与展望：总结了论文的主要工作和结论，对将来的研究工作进行展望，提出了进一步的研究方向。

第2章 Web挖掘和Web使用挖掘研究

随着 Internet 技术的发展，Web 资源也在飞速地膨胀，如何开发和利用这些丰富的资源就成了人们普遍关注的问题。人们如何从这海量的数据中，查到自己想要的数据和信息，迫切需要一种新的技术能自动地从 Web 资源上发现、抽取和过虑信息，于是出现了 Web 挖掘技术。Web 挖掘是对 Web 文档的内容、Web 上可利用资源的使用情况以及资源之间的关系进行分析，从中发现有效、新颖、潜在有用、并且最终可理解的模式。

本章首先对 Web 挖掘的定义及其过程进行概述，然后介绍 Web 挖掘的 3 个分类：Web 内容挖掘(Web content mining)，Web 结构挖掘(Web structure mining)，和 Web 使用挖掘(Web usage mining)的内容、方法和意义，最后回到本文的主题——Web 使用挖掘，对其过程和应用进行阐述。

2.1 Web 挖掘

2.1.1 Web 挖掘的定义

Web 挖掘是一门交叉性学科，涉及数据挖掘、机器学习、模式识别、人工智能、统计学、计算机语言学、计算机网络技术、信息学等多个领域。

从广义的角度出发，可以对 Web 挖掘作如下定义[20]：

定义 1.1：Web 挖掘是指从大量非结构化、异构的 Web 信息源集合中发现有效、新颖、潜在可用及最终可理解的知识，包括概念、模式、规则、规律、约束及可视化等形式的非平凡过程。

以上定义借鉴了数据挖掘的传统定义，因此 Web 挖掘在部分方法和技术研究方面也与传统的数据挖掘相似，具有相通之处。

如果从实用性开发的角度来考虑的话，可以对 Web 挖掘做出如下定义[21]：

定义 1.2：Web 挖掘是针对包括 Web 页面内容、页面之间的结构、用户访问信息、电子商务信息等在内的各种 Web 数据源，在一定程度上对应用数据挖掘的方法以发现有用的知识帮助人们从 WWW 中提取知识，改进站点设计，更好地开展电子商务的应用。

2.1.2 Web 上的数据

World Wide Web 是巨大、分布、异构、半结构化、动态及基于超链接的超媒体文档构成的数据库。Web 页面用 HTML（Hypertext markup language）书写，由多种媒体对象和指向其他文档的指针(超链)组成。具体来讲，Web 上的数据主要有以下特点[22]：

(1) 从数据库的角度出发，Web 网站上的信息可以看作一个更大、更复杂的数据库。Web 上的每一个站点就是一个数据源，每个数据源都是异构的，因而每一站点的信息和组织都不一样，这就构成了一个巨大的异构数据库环境。

(2) 从数据管理的角度出发，Web 页面散布在世界各地的 Web 服务器上，形成了分布式数据源。每个服务器自主地管理自己的资源，没有统一的管理机制，这为数据的分析和处理带来了更高的难度。

(3) 从数据模型的角度出发，半结构化是 Web 上数据的最大特点。Web 上的数据非常复杂，没有特定的模型描述，每一站点的数据都分别独立设计，并且数据本身具有自述性和动态可变性。因而，Web 上的数据具有一定的结构性，但因自述层次的存在，是一种并非完全结构化的数据，也可称之为半结构化数据。

(4) 从数据内容的角度出发，Web 包含了各种信息和资源，有文本数据、超文本数据、图表、图像、音频数据和视频数据等各种多媒体数据，表现了 Web 数据的多样性和复杂性。

(5) 从数据更新的角度出发，Web 是一个动态性极强的信息源，不仅增长的速度极快，而且信息也在不断地快速更新，各站点的链接信息和访问记录的更新非常频繁。

(6) 从用户的角度出发，Web 面对的是一个庞大的、且在不断扩张的用户群体，每个用户具有不用的背景、兴趣和使用目的。大部分用户并不了解信息网络结构，不清楚搜索的高昂代价，极容易在“黑暗”的网络中迷失方向，也极容易在“跳跃式”的访问中翻乱不已，或者在等待信息的过程中失去耐心。

如果想要利用 Web 数据进行数据挖掘，首先，必须研究站点之间异构数据的集成问题，只有将这些站点的数据都集成起来，提供给用户一个统一的视图，才有可能从巨大的数据资源中获取所需的信息；其次，要解决 Web 上

的各种数据查询问题，如果不能有效地得到所需的数据，对这些数据进行分析、集成、处理也就无从谈起。

2.1.3 Web 挖掘的分类

Web 挖掘的研究依然遵循数据挖掘的研究思路，其一般过程可以分为 3 个阶段：

(1) 数据预处理：需要对收集的数据进行必要的预处理，例如清除脏数据。

(2) 模式发现：应用不同的 Web 挖掘算法发现用户的访问模式。

(3) 模式分析：从模式发现的模式集合中提取有价值的模式。

一般地，Web 挖掘可以分为 3 类[23]：Web 内容挖掘，Web 结构挖掘，和 Web 使用挖掘，如图 2-1 所示。

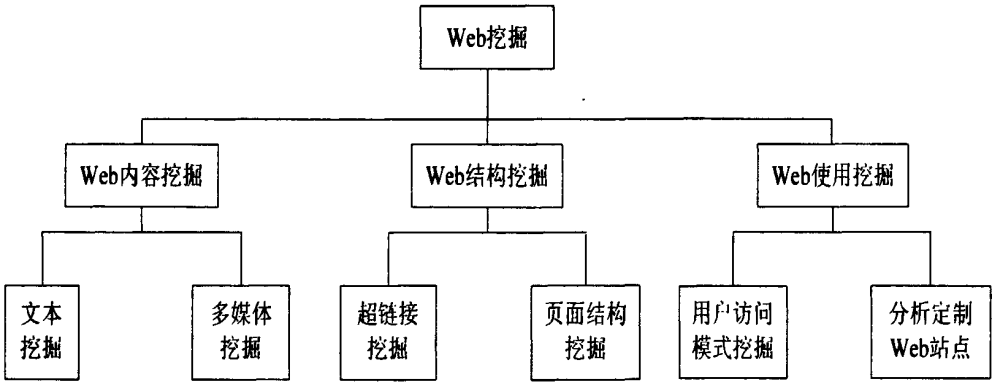

图2-1 Web挖掘的分类

表 2-1[24]展示了 3 类不同挖掘的特点。

表 2-1 Web 内容挖掘、结构挖掘和使用挖掘比较

	Web 挖掘			
	Web 内容挖掘		Web 结构挖掘	Web 使用挖掘
	信息检索领域	数据库领域		
数据	•文本文档 •超文本文档	•超文本文档	•链接结构	•Web 服务器日志 •Proxy 日志 •浏览器日志
数据特征	•非结构化 •半结构化	•半结构化 •Web 站点可以看作一个数据库	•链接结构	•交互式数据

续表

数据表示形式	•无序/有序的单词集合 •术语和短语 •概念/实体 •关系曲线	•对象交互模型 •关系曲线	•图	•关系表 •图
方法	•机器学习 •统计	•专利算法 •关联规则及变形	•专利算法	•机器学习 •统计 •关联规则及变形 •聚类 •序列模式
应用领域	•分类 •聚类 •寻找抽取规则 •寻找文本模式 •用户建模	•发现频繁子结构 •提取 Web 站点大纲	•分类 •聚类	•站点结构管理及优化 •网络销售 •用户建模 •推荐系统

2.1.3.1 Web 内容挖掘

Web 内容挖掘是从 Web 文档的内容或其描述中提取知识的过程，这是一个从非结构化文本信息中获取用户感兴趣或者有用的模式的过程[25]。Web 文档由多种类型的数据组成，如文本、图像、音频和视频等。这些数据又分为文本数据和多媒体数据两大类，从而将 Web 内容挖掘对应地分为 Web 文本挖掘和 Web 多媒体挖掘两类[26]。

Web 内容挖掘一般从两个不同的观点来进行研究。从信息检索(Information retrieval)的观点来看，Web 内容挖掘的任务是从用户的角度出发，怎样提高信息质量和帮助用户过滤信息。而从数据库的角度来讲，Web 内容挖掘的任务主要是试图对 Web 上的数据进行集成、建模，以支持对 Web 数据的复杂查询。

2.1.3.2 Web 结构挖掘

Web 结构挖掘是从 WWW 的组织结构、Web 文档结构以及链接关系中推导知识。Web 结构挖掘主要针对的是外部文档的超链接结构。挖掘 Web 结构

的目的是发现 Web 的结构和页面的结构及其蕴含在这些结构中的有用模式；对页面及其链接进行分类和聚类，找出权威页面；发现 Web 文档自身的结构，这种结构挖掘能更有助于用户的浏览，也利于对网页进行比较和系统化。文档之间的链接反映了文档之间的引用关系，一个网页被引用的次数体现了该页面的重要性。对 Web 进行结构挖掘，可以得到以下的信息：

(1) 同一网站里不同网页链接的频率；

(2) 同一网站里同一网页内部链接的频率；

(3) 不同网站间链接的频率。

Brin 等[27]提出用 Page-rank 方法来发现“权威”页面，其基本思想是：一个页面被多次引用，则这个页面很可能是重要的；各页面尽管没有被多次引用，但被一个重要页面引用，则这个页面很可能也是重要的。Kleinberg[28]提出了另一种挖掘“权威”页面的方法——Hub/Authority 方法。Hub 是指一个或多个 Web 页面，它提供了指向权威页面的链接集合，Hub 页面本身可能并不突出或者说可能没有几个链接指向它们，但是 Hub 页面却提供了指向某个突出站点的链接，起到了隐含说明某权威页面的作用。

2.1.3.3 Web 使用挖掘

Web 用户在日常活动中产生大量的信息，这些信息可以被 Web 服务器自动收集并存储在访问日志中。对用户的每次访问，Web 日志记录了访问的时间、用户的网络地址、目的信息的网络地址及传输的信息量等。Web 使用挖掘就是从 Web 访问日志中获取用户访问 Web 的规律并预测用户的网上行为[29]。同时，对于 Web 网站，可以通过分析其访问日志，发现潜在的威胁和漏洞，增强网站的安全性[30]。

2.2 Web 使用挖掘研究

2.1.1 Web 使用挖掘的定义

计算机网络技术将世界各个角落连接成一个信息资源巨大的 WWW 网，其用户数以亿计。随着硬件技术和计算机网络技术的发展，网络用户也日益增长。因此，进一步改善 Web 服务器系统的性能，提高 Internet 信息服务的质量是很有必要的；其次，用户群体也表现出多样性的特点，全球信息网大约有数亿个工作站，其用户具有不同的背景、不同的兴趣和目的。如何对网

络用户的行为规律加以分析，对信息系统的研究者提出了新的挑战。Web 使用挖掘已成为一个新的研究领域，在电子商务和个性化 Web 等方面有着广泛的应用。

Web 使用挖掘主要是通过挖掘 Web 日志记录，来发现用户访问 Web 页面的模式。通过分析和探究 Web 日志记录中的规律，可以识别电子商务的潜在用户，增强对最终用户的因特网信息服务的质量和交付，并改进 Web 服务器系统的性能[31]。

2.2.2 Web 使用挖掘的数据源

数据挖掘的关键步骤之一就是要构造一个适合目标任务的数据集。从服务器端、客户端、代理端，或从网站数据库中所收集的各种数据，它们不仅(数据)类型差别较大，而且相应的(数据)处理方法也各不相同。从不同数据源收集而来的数据反映了 Web 使用过程中的不同访问模式。客户端的数据通常反映单用户/多站点的访问行为；服务器端的数据则描述了多用户/单站点的访问行为；而代理端的数据则记载了多用户/多站点的使用情况。Web 日志是 Web 使用挖掘中最重要的数据源，它清楚记录了用户的访问浏览行为。

2.1.2.1 术语解释

W3C 组织(www.w3c.org)为描述用户在 Web 上的使用行为定义了若干概念[32]，虽然它不是一个标准，但是其中的一些概念还是得到了广泛接受。虽然这些概念在 Web 日志挖掘领域内的解释有所变化，但它依旧是理解 Web 使用数据特性的基础。我们简要介绍如下：

Web 服务器日志：Web 服务器在响应用户的请求时，将用户请求的文件发送出去的同时把这次请求写入日志，所以 Web 服务器日志记录了用户访问本站点的信息。一般分 3 部分：访问日志(Access Log)、代理日志(Agent Log)和引用日志(Referrer Log)。访问日志主要记录基本的请求信息，包括：用户的 IP 地址、时间戳、方法(如 GET, POST)、被请求文件的 URL、超文本传输协议(HTTP)的版本号、返回码(请求的状态，成功或错误码)、传输字节数。代理日志记录用户使用的操作系统以及浏览器类型。引用日志则记录用户的请求是来自那个 URL。

请求 (Request 或 Hit)：向 Web 服务器请求一个文件的动作，对于用户来说是站点中某个超链接，对于服务器来说一个请求对应一条日志记录。要

说明的一点是，一个 Web 页面可能包含多个文件，如 HTML 及声音和图像，此时用户虽然只点中一个超链接，但是相应的有多个请求，在日志中就有多行记录，一个 Web 页面对应的请求数可以通过查看 Web 日志得到。

网络爬虫(Spider 或 Crawler)：一种网络软件工具，通过分析 Web 站点的每个 HTML 文件，建立与 HTML 文件对应的超文本链接目录，然后跟踪每一个超链接，直到 Web 站点的所有页面都被遍历，最终获得 Web 站点的结构文件。

Cookie：首先由 Netscape 公司开发的一种用于追踪用户的机制。用户第一次访问站点时，Web 服务器为其分配一个唯一的标识符并保存在用户的计算机中，用户再次访问该站点时，浏览器将该标识符回送给 Web 服务器，由此来识别用户。

用户会话 (User Session)：一段时间内用户对一个或多个网站的访问请求。

服务器会话(Server Session)：用户会话中对应于本服务器的访问请求。

页视图(Page View)：用户点击一次超链接后在浏览器中得到的一个屏幕显示，页视图可能由许多文件构成。一个页视图代表用户的一次点击动作。

由于本文中所提到的 Web 使用挖掘一般只是面向一个服务器的，所以在本文中，用户会话和服务器会话的含义一致，指用户在一段时间内访问 Web 站点时所有请求的页面。用户会话文件中包含访问 Web 站点的用户、请求的页面及顺序、每一页阅读的时间等。

2.1.2.2 Web 日志简介

Web 服务器日志记录着用户访问该站点时每个页面的请求信息。日志记录的格式主要分为两种：通用日志格式(Commen Log Format，CLF)和扩展型日志格式(Extended Log Format，ECLF)。其主要结构如表 2-2 所示，其中标有*号的是扩展日志中增加的项。

表 2-2 Web 日志记录的结构表示例

属性域	描述
日期	用户请求页面的日期
时间戳	用户请求页面的具体时间
用户 IP 地址	客户端主机的 IP 地址或 DNS 出口

续表

用户名	客户端的用户名
*服务器名	服务器的名称
服务器 IP 地址	服务器端的 IP 地址
服务器端口	服务器端的端口号
方法	用户请求的方法
访问页	用户请求的页面
URL 查询	用户欲进行的查询
协议状态	HTTP 的状态标识
*发送字节数	发送数据的字节数
*接收字节数	接收数据的字节数
*所花时间	浏览耗费时间
*协议版本	HTTP 协议的版本
主机	服务器的操作系统
用户代理	服务的提供者
*Cookie	Cookie 标识号
*引用页	用户从哪一页跳转过来

在日志文件中，每条记录被称为项或条目。其中：

用户 IP 地址(Client IP)是发出请求的客户端的 IP 地址，在 Proxy 代理服务器的环境下为代理服务器的 IP 地址。

用户名(User name or User id)域一般不填写，只有当存取特定的文件，需要鉴别身份时才需要。

时间戳(Date or Time)表示 Web 服务器接受该请求的时间，在整个日志文件中，每一项以时间戳递增排列。

请求域（Request）包括请求方法、请求协议。其中请求的方法有：GET、POST 和 HEAD。 GET 从 Web 服务器得到对象；POST 向 Web 服务器发送信息；HEAD 仅请求一个对象的 HTTP 头。

URL 查询或者为服务器上文件系统上的一个静态的文件，或者为一个响应该请求的一个将要被调用的可执行程序。

协议状态域由 Web 服务器设置指示出响应该请求的行为：从 200 到 299 的代码一般指示成功响应；从 300 到 399 表征某种程度的重定向；400 到 499

指示错误；500 到 599 表示 Web 服务器有问题。

接收字节数(Size or Bytes)表示返回结果的字节数。

引用页域(Referer)表示上次被请求的页面，如果用户通过直接键入地址或通过书签(BookMark)访问，那么该域为空。

用户代理域(User-Agent)能够指出客户端的操作系统和浏览软件。在某些日志中，Referer 域和代理域不被记录。

其中标有*号的是扩展型日志中增加的项。在本文的日志分析过程中，并不是所有的条目都必须使用，我们只把日志记录中用到的日志条目投影出来。

2.2.3 Web 使用挖掘的过程

Web 使用挖掘是应用数据挖掘技术和方法从 Web 使用数据或 Web 日志数据中发现 Web 使用模式的过程。在 Web 使用挖掘过程中，主要运用人工智能、数据挖掘、信息论和数据库等领域的相关技术和知识，从 Web 日志中挖掘知识，来探索和分析 Web 使用记录数据中的规律和用户访问模式。

Web 使用挖掘一般分为 Web 数据预处理(Web Data Preprocessing)，模式发现(Pattern Discovery)和模式分析(Pattern Analysis) 3 个主要的阶段，如图 2-2 所示。

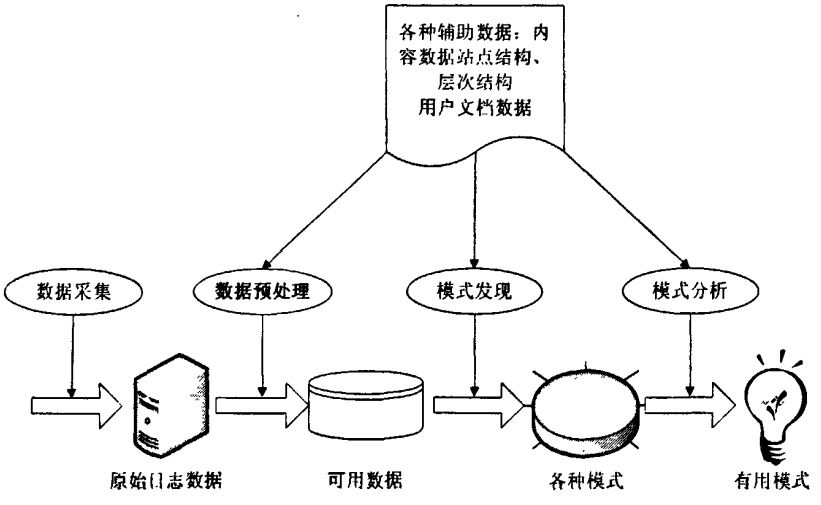

图2-2 Web使用挖掘的过程

2.2.3.1 数据预处理

在 Web 使用挖掘的过程中，有一个重要的基础就是 Web 数据的准确性，必须有准确的数据才能每次都正确地反映使用者的意图，从而可以使分析沿

着正确的方向进行。不过，由于本地缓存、代理服务器、防火墙的存在，使得直接在庞杂的 Web 日志数据上进行挖掘变得十分困难和不准确。因此，在实施数据挖掘之前，必须对 Web 日志文件进行预先处理。

实现这个功能的阶段就是数据的预处理阶段[33]，它主要完成的任务是把各种不同的 Web 使用记录数据抽取和映射为模式发现所需的数据形式，它是模式发现阶段和模式分析阶段的数据准备。数据预处理阶段通常分为以下几个部分：数据清理（数据净化）、用户识别、会话识别、路径补充和事务识别等。

2.2.3.2 模式发现

模式发现即是运用各种方法和算法从 Web 日志数据中挖掘和发现有效、新颖、潜在、有用及用户可能感兴趣的，最终可以理解的信息和知识。这一阶段使用的方法和算法不仅仅来自数据挖掘领域，还包括统计学、机器学习和模式识别等领域[34]。不过，从其他领域移植到 Web 挖掘领域的技术需要考虑到原有领域与 Web 挖掘领域可能会有不同的数据类型、不同的先验知识。所以，这些技术必须针对 Web 挖掘领域的特点做出相应的修改和完善才能进行成功的移植。

模式发现阶段使用的技术有：统计分析、路径分析、关联规则、分类聚类技术和序列模式等。其中路径分析技术是 Web 使用挖掘所特有的。关联规则是较重要的一种，它属于描述性模式，决策者只对满足一定支持度和可信度的关联规则感兴趣。因此，为了发现有意义的关联规则，需要给定两个阈值：最小支持度和最小可信度。

预处理阶段对用户事务进行了划分后，就可以根据具体的分析需求，选择访问模式发现的技术。模式发现阶段是数据挖掘的核心，也是技术难点所在。下面分别介绍模式发现阶段使用的各种技术。

1. 统计分析

统计分析是最常用的分析网站访问以抽取有关知识的方法。利用分析会话文件，可以对网页浏览、浏览时间和浏览路径长度等进行各种不同的描述性统计分析，如频率、均值和中间值等。目前已经有许多网站分析工具能够提供此类统计信息的周期性报告。尽管缺乏深度的分析，这类统计知识还是能够帮助改善网站系统的性能、提高系统安全性、促进网站内容更趋合理，

以及提供市场营销决策支持。

2. 路径分析

路径分析就是要从图中确定最频繁的路径访问模式或频繁的参考序列。

我们可以用许多不同类型的图形来进行路径分析，因为图形表达了各网页间定义的关系[35]。最常见的一种是表示网站物理布局的网站结构图，它把网页当作结点，把页面间的超文本链接当作连接的边。还可以根据网页类型来生成其他图形，在这类图形中，边代表页面间的相似度，或者在边上给出使用该超链接的人数。

目前所做的大部分工作是从图形的物理布局类型中确定常见的横断模式或大型参考访问序列。导航性质的事务、最大向前参考事务或用户会话都可以用来进行路径分析。可以从图形的物理布局中找到用户的浏览模式，发现 Web 站点中最经常被访问的路径，从而可以调整站点的结构。

3. 关联规则

在 Web 使用挖掘中，关联规则是指经常被一起访问的，支持度超过指定阈值的页面集合。它主要用于发现用户之间、页面之间以及用户浏览页面和网上行为之间存在的潜在关系。比如挖掘可能得出“浏览/company/products/ElectronicProduct.html 的用户 68% 都会浏览 /company/products/Software.html ”， 并且“ 浏览/company/products/Software.html 的用户 59% 都会在线下订单”的规则，那么显然，网络管理员应该在电子商品目录页面提供进入计算机软件目录页面的直接途径[36]。

4. 分类学习

在 Web 使用挖掘中，可以利用分类学习获得有关某类用户的概要描述。分类方法主要包括：决策树方法、简单贝叶斯方法、k-最近邻方法和支持向量机方法。例如，利用含有登记数据的访问数据，进行分类学习所获得知识模式可以是：30%提交在线订单的用户，年龄在 18～25 岁，家住北京。

5. 聚类分析

聚类技术将数据对象按特征相近的原则划分为多个类或簇。在 Web 使用挖掘领域有两种有趣的聚类：用户聚类和页面聚类。用户聚类就是将那些经常访问相同页面的用户群划分出来，他们具有相同的使用习惯和网上行为，可以对他们开展特定的广告策略或是个性化定制。页面聚类则发现内容相关

的页面组，为搜索引擎和 Web 服务商提供有用信息。

在 Web 使用挖掘中，聚类算法将用户浏览页面的总和视为数据空间，构造一个稀疏图。首先，根据每个页面的内容相似性和路径互联性，将数据对象分割为若干个 k-最近邻居子图(簇)，图中的每个点都代表一个页，子图的密度作为边的权重被记录下来。如果发现两个子图间的互联性和相似性与子图内部页面的互联性和相似性高度相关的话，则将二者合并为一个簇。

6. 序列模式

在 Web 使用挖掘中，利用序列挖掘方法可以帮助发现如下模式知识：50%的情况下，浏览彩电商品网页后就会浏览电脑商品网页。利用上述模式知识，可以帮助预测网页的访问情况，从而可以帮助合理地放置广告以针对特定用户群。

2.2.3.3 模式分析

模式分析是整个 Web 使用挖掘过程的最后一步，它的作用是过滤掉模式发现阶段产生的无用规则，从而抽取出用户最感兴趣的规则和模式。

由于 Web 使用挖掘在大多数情况下属于无偏向学习，它可能挖掘出所有的模式和规则。其中，有些模式是常识性的，普通的或最终用户不感兴趣的。所以，必须采用模式分析的方法使得挖掘出来的规则和知识具有有效性、可读性和最终可理解性。常见的模式分析方法有图形和可视化技术、联机分析处理和数据库查询机制等。

2.3 本章小结

Web 使用挖掘是 Web 挖掘一个重要研究内容，而 Web 挖掘则是数据挖掘和快速发展的 WWW 结合所形成的一个崭新的研究领域。本章主要介绍了 Web 挖掘和 Web 使用挖掘的基本理论和相关概念。首先介绍了 Web 挖掘的定义、特点和分类，然后对 Web 使用挖掘的定义、应用、数据源和过程进行了详细叙述。

第 3 章 一种新的基于事务矩阵的关联规则算法

关联规则挖掘是数据挖掘中最活跃的研究方法之一。最早是由 Agrawal 等人针对“购物篮”分析问题提出的，其目的是为了发现交易数据库中不同商品之间的联系规则。这些规则刻画了顾客购买行为模式，可以用来指导商家科学地安排进货、库存以及货架设计等。之后，诸多的研究人员对关联规则的挖掘问题进行了大量的研究。应用使用挖掘技术，从 Web Log 文件中挖掘关联规则，可以得到形如下例的一些规则：访问 pa1 页面的客户中，有 40%又访问了 pc1 页面；访问 pc2 页面的客户中，有 70%是从 pa2、pa3、pa4 和 pa5 页面进入系统的。这些规则可以帮助网站管理员对站点结构进行管理及优化。本章首先对关联规则相关知识给出了概述，然后对其典型的 Apriori 算法进行描述和分析，并在分析基础上提出了一种基于事务矩阵的关联规则算法，经理论分析和实验证明该算法性能优于 Apriori 算法。

3.1 关联规则概述

3.1.1 关联规则的背景介绍

事务之间存在着普遍联系，这是辩证法的重要内容之一。著名的“啤酒与尿布”的例子就反映了这种普遍联系。在这个例子中，商家最初的目的就是希望在货物篮数据中发现那些经常被同时购买的货品。当某些商品同时出现的次数超过用户设定的阈值时，就可以认为这些商品之间可能存在着关联关系，啤酒和尿布的关系就是一个关联关系。发现这种关联关系的过程就叫关联分析(association analysis)。

关联分析最初是统计学中经常用到的一个名词，指的是对两个或者更多变量之间可能存在的关联关系的分析。从广义的角度说，这种关系还可以是因果关系或者时序关系等。

对于关联关系来说，比较常见的是简单关联关系和时序关联关系。例如，“购买面包的顾客 90%也同时购买牛奶”就是一个简单关联。而“如果 AT&T 股票连续上涨两天且 DEC 股票不下跌，那么第三天 IBM 股票上涨的可能性为 75%”，这样的关联就属于时序关联，其发生和时间先后有关。

关联规则(accociation rule)的挖掘就是为了在数据库中发现关联关系，它是数据挖掘最先研究的问题之一，也是数据挖掘一个主要研究方向。实际上，关于数据挖掘方面的文章，尤其是早期的一些研究论文和领域应用，多数集中在对各类不同的关联规则的定义和挖掘算法的设计上。甚至可以说，提到数据挖掘，人们首先就会想到关联规则挖掘。关联规则是由 Agrawal、Imielinski 和 Swami 在 1993 年首先提出的[37]，起初是研究超市的顾客交易数据库中购买商品之间的关联规则的挖掘问题，即所谓的货篮数据的关联规则。到了 1994 年，Agrawal 和 Srikant 提出了关联规则挖掘的经典算法 Apriori[38]，这篇论文在其后被多次引用，成为这个领域最基本的算法。迄今为止，关联规则挖掘的研究论文已经达到了数千篇，而它的应用也由最初的货篮数据扩展到其他数据格式，规则的涵义也越来越多样化。

3.1.2 关联规则挖掘的基本概念

定义 3.1：设 $I=\{i_1, i_2, \ldots, i_m\}$为全体数据项（简称项）的集合，D 为全体事务的集合。其中，每个事务 T 是项的集合，使得 $T \subseteq I$，且每一个事务具有唯一的标识符，记为 TID。设 X 是一个项集，事务 T 包含 X 当且仅当 $X \subseteq T$。

项的集合称为项集(itemset)。项集包含的元素的个数称为项集的长度，长度为 k 的项集称为 k 阶项集(k-itemset)，或称 k-项集。

定义 3.2：关联规则是形如 $X \Rightarrow Y$ 的蕴涵式，其中 $X \subseteq I$，$Y \subseteq I$，并且 $X \cap Y = \phi$。项集之间的关联表示：如果 X 出现在一条交易中，那么 Y 在这条交易中同时出现的可能性比较高。

定义 3.3：规则 $X \Rightarrow Y$ 的支持度（support）是 D 中包含 X 和 Y 的百分比，从统计的角度看，就是概率 P（$X \cup Y$），表示 X 和 Y 同时出现的可能性。规则 $X \Rightarrow Y$ 的置信度(confidence)定义为 D 中包含 X 的事务的同时也包含 Y 的百分比，即条件概率 P(Y/X)。即：

$$\text{support}(X \Rightarrow Y) = P(X \cup Y)$$

$$\text{confidence}(X \Rightarrow Y) = P(Y/X)$$

对于给定的支持阈值和置信阈值，支持度和置信度都大于相应阈值的称为关联规则。项集满足最小支持度的称为频繁项集(frequent itemset)。频繁 k-项集的集合通常记为 L_k。

关联规则的挖掘分为两个步骤：

(1) 找出所有的频繁项集：根据定义，这些项集出现的频繁性至少和预

定义的最小支持计数一样。

(2) 由频繁项集产生关联规则：根据定义，这些规则必须满足最小支持度和最小置信度。

3.1.3 关联规则的类型

关联规则按不同的情况可以进行多种分类，常见的有以下几种[39]：

1. 基于规则中数据的抽象层次分类

基于规则中数据的抽象层次，可以分为单层关联规则和多层关联规则。

在单层的关联规则中，所有的变量都没有考虑到现实的数据是具有多个不同的层次的；而在多层的关联规则中，对数据的多层次性已经进行了充分的考虑。

例如：IBM 台式机⇒Sony 打印机，是一个细节数据上的单层关联规则；台式机⇒Sony 打印机，是一个较高层次和细节层次之间的多层关联规则。

2. 基于规则中涉及到的数据的维数分类

基于规则中涉及到的数据的维数，关联规则可以分为单维的和多维的。

在单维的关联规则中，只涉及到数据的一个维，如用户购买的物品；而在多维的关联规则中，要处理的数据将会涉及到多个维。换一种说法，单维关联规则是处理单个属性中的一些关系；多维关联规则是处理各个属性之间的某些关系。

例如：啤酒⇒尿布，这条规则只涉及到用户的购买的物品；性别=“女”⇒职业=“秘书”，这条规则就涉及到两个属性的信息，是二维数据上的一条关联规则。

3. 基于规则中处理变量的分类

基于规则中处理的变量的类别，关联规则可以分为布尔型和数值型。

布尔型关联规则处理的值都是离散的、种类化的，它显示了这些变量之间的关系。例如：性别=“女”⇒职业=“秘书”，就是一条布尔型关联规则。而数值型关联规则可以和多维关联或多层关联规则结合起来，对数值型字段进行处理，将其进行动态的分割，或者直接对原始的数据进行处理，当然数值型关联规则中也可以包含种类变量。例如：性别=“女”⇒avg (收入)=2300，涉及的收入是数值类型，所以是一个数值型关联规则。

3.2 关联规则的经典算法 Apriori

3.2.1 Apriori 算法简介

1994 年，Agrawal 和 Verkano 提出了关联规则挖掘的经典算法 Apriori，它是一种最有影响的挖掘布尔关联规则频繁项集的算法。Apriori 使用一种称作逐层搜索的迭代方法，k-项集用于探索 k+1-项集。首先，找出频繁 1-项集的集合。该集合记作 L_1。L_1 用于找频繁 2-项集的集合 L_2，而 L_2 用于找 L_3，如此下去，直到不能找到频繁 k-项集。找每个 L_k 需要扫描一次数据库。

为了提高频繁项集逐层产生的效率，一种称作 Apriori 性质的重要性质用于压缩搜索空间，Apriori 性质描述如下。

性质 3.1：频繁项集的所有非空子集都必须也是频繁的。

也就是说，如果项集 I 不满足最小支持度阈值 min_sup，则 I 不是频繁的，即 P(I)<min_sup。如果项 A 添加到 I，则结果项集(即 I∪A)不可能比 I 更频繁出现。因此，I∪A 也不是频繁的，即 P(I∪A)<min_sup。

3.2.2 Apriori 算法描述

Apriori 算法的思想为，在第一次循环时，通过扫描数据库得到频繁 1-项集，在之后的第 k(k>1)次循环中，对第 k-1 次循环产生的 k-1 阶频繁项集 L_{k-1} 实施 Apriori_gen 运算生成 k 阶候选项集 C_k。再次扫描数据库，得到 C_k 的支持数，从而得到 C_k 中支持数不小于最小支持数的 k 阶频繁项集 L_k。重复以上步骤，直到某一阶的频繁项集为空时算法停止。

下面给出 Apriori 算法的伪代码描述。

算法 3-1：Apriori 使用根据候选生成的逐层迭代找出频繁项集。

输入：事务数据库 D；最小支持度阈值 min_sup。

输出：D 中的频繁项集 L。

```
1)  L1={large 1-itemsets};
2)    for (k=2;Lk-1≠ϕ, k++){
3)    Ck=apriori_gen(Lk-1,min_sup);    // Ck 是 k 阶候选项集
4)      for each trasactions t∈D{      // scan D for counts
5)        Ct=subset(Ck,t);             // Ct 是所有 t 包含的候选项集元素
6)        For each candidates c∈Ct
```

```
7)            C.count++;
8)          }
9)        Lk={c∈Ck|c.count≥minsup_count}
10)       }
11)   Return L=∪kLk;
```

算法 3-1 中调用了 apriori_gen(L_{k-1},min_sup)。是为了通过频繁 k-1-项集产生 k 阶候选集。算法 3-2 描述了 apriori_gen 的过程。

算法 3-2：apriori_gen(L_{k-1},min_sup)——候选集产生。

输入：频繁 k-1-项集 L_{k-1}；最小支持度 min_sup。

输出：候选 k 阶项集 C_k。

```
1)  for each itemset p∈Lk-1
2)    for each itemset q∈Lk-1
3)      If (p.item1=q.item1,...,p.itemk-2=q.itemk-2,p.itemk-1<q.itemk-1) then{
4)        c=p∞q;              // 把 q 的第 k-1 个元素连到 p 后
5)        if has_infrequent_subset(c, Lk-1) then
6)          Delete c;         // 删除含有非频繁项目子集的候选元素
7)        Else add c to Ck;
8)        }
9)  Return Ck;
```

算法 3-2 中调用了 has_infrequent_subset(c,L_{k-1})，是为了判断 c 是否需加入到 k 阶候选集中。按照 Agrawal 的项目集格空间理论，含有非频繁项目子集的元素不可能是频繁项目集，因此应该裁减掉，以提高效率。

例如，如果 L_2={AB, AC, AD, BD}，对于新产生的元素 ABC 不需要加入到 C_3 中，因为它的子集{BC}不在 L_2 中，而 ABD 应该加入到 C_3 中，因为它的所有 2-项子集都在 L_2 中。

算法 3-3 描述了这个过程。

算法 3-3： has_infrequent_subset(c,L_{k-1})——判断候选项集的元素。

输入：候选 k 阶项集 c；频繁 k-1-项集 L_{k-1}。

输出：一个 Bool 变量。

1) for each (k-1)-subset s of c

2) if s∉ L_{k-1} then

3) return TRUE;

4) return FALSE;

如上所述，apriori_gen 做两个动作：连接和剪枝。在连接部分，L_{k-1} 与 L_{k-1} 连接产生可能的候选(第 1-4 步)。剪枝部分(第 5-7 步)使用 Apriori 性质删除具有非频繁子集的候选。非频繁子集的测试在过程 has_infrequent_subset 中。

3.3 Apriori 算法的性能瓶颈分析

在许多情况下，Apriori 的候选产生-检查方法大幅度压缩了候选项集的大小，有着很好的性能，但在实际应用中还存在以下问题：

◆ 需要重复地扫描数据库，用来计算每次侯选项集的支持数。随着项集阶数的增加，侯选项集的个数会逐渐减少，包含这些侯选项集的事务也会越来越少，但扫描的事务量并没有减少。

◆ 可能需要产生大量的侯选项集。例如，如果有 10^4 个频繁 1-项集，那么 Apriori 算法需要产生多达 10^7 个侯选 2-项集，并检查它们的频繁性。此外，为发现长度为 100 的频繁模式，如$\{a_1, ..., a_{100}\}$，会产生多达 2^{100}（$\approx 10^{30}$）个侯选项集。

因此，通过减少原始事务数据库的扫描次数和侯选项集的数量，可以进一步提高算法的效率。本章下一节提出的改进算法正是通过对事务数据库进行一次扫描，将其映射为一个事务矩阵，在矩阵上进行操作得到所有的频繁项集来提高算法的性能。

3.4 基于事务矩阵的关联规则挖掘算法的提出

3.4.1 算法思想及所基于的推论

3.4.1.1 算法思想

针对 Apriori 算法的缺点，本文提出了一种基于事务矩阵的关联规则挖掘算法，通过将事务数据库映射为一个事务矩阵，对事务矩阵进行操作以得到

所有的频繁项目集，整个算法分为两步：第一步由事务数据库生成的事务矩阵产生频繁 1-项集并记录相关信息；第二步，由频繁 k-1-项集产生频繁 k-项集（k>1）。

3.4.1.2 算法涉及的基本定义和推论

定义 3.4：设 $I=\{i_1, i_2, \ldots, i_m\}$是全体数据项的集合，$\{T_1, T_2, \ldots, T_n\}$构成全体事务的集合 D。可以将 D 映射为一个矩阵，其中如果 T_i $(1\le i\le n)$中包含项目 I_j $(1\le j\le m)$，则矩阵的第 j 行第 i 列的值为 1，反之取 0。称这个矩阵为 D 的事务矩阵，记做 T_{m*n}。

由关联规则的相关定义和 Apriori 性质可以得到以下推论：

推论 3.1：对于事务 T_i $(1\le i\le n)$，由频繁 k-项集产生频繁 k+1-项集时，如果 T_i 包含的项集的长度小于 k+1,可以修剪掉该事务项。

推论 3.2：设 I_j $(1\le j\le m)\in L_k$, 记 $|I_j|$ 为 L_k 中包含 I_j 的频繁项的个数，如果 $|I_j|<k$, 则在由频繁 k-项集产生频繁 k+1-项集时可以剪裁掉该数据项。

证明：设项集 $X=\{\ldots, I_j, \ldots\}\in L_{k+1}$，则由性质 3.1 可得在 L_k 中包含 I_j 的项集的个数至少为 $C_k^1=k$，所以推论 2 是成立的。证毕。

推论 3.3：记$|L_k|$为频繁 k-项集的个数，如果$|L_k|<k+1$，则没有必要进行侯选 k+1-项集的产生。

证明：用反证法证明。假设当$|L_k|<k+1$ 时，由 L_k 产生 C_{k+1} 进而得到 L_{k+1}，又由性质 3.1 可得对于 L_{k+1} 中的一个频繁 k+1-项集，它包含 k+1 个互不相同的频繁 k-项集，这与假设的条件是矛盾的。证毕。

3.4.2 算法描述

算法 3-4 给出本文所提出新算法的伪代码描述：

算法 3-4：基于事务矩阵的关联规则挖掘算法。

输入：事务数据库 D；最小支持度阈值 min_sup(最小支持数 minsup_count=|D|*min_sup)。

输出：所有频繁项集。

1) 扫描事务数据库，得到 T_{m*n};
2) 分别对 T_{m*n} 的行向量和列向量计数，修剪掉支持数小于最小支持数的项对应的行向量，记录频繁 1-项集并得到 T_{p*q}; //由频繁 1-项集修剪得到的矩阵

```
3)  while(|L_k|>k+1) then{              // 推论 3.3;k>1
4)    for(i=0;i<p;i++)                  // 对行向量和列向量记数
5)        count_r[i]= T[i,0]+…T[i,q-1];
6)    for(i=0;i<q,i++)
7)        count_c[i]= T[0,i]+…+T[p-1,i];
8)    for(i=0;i<p;i++)
9)        if count_r[i]<k-1 then      //count_r[i]为 I_i所对应的行向量的代数和
10)         delete T[i,j] from T[p-1,q-1];  // 推论 3.1
11)   for(i=0;i<q;i++)
12)       if count_c[i]<k  then     //count_c[i]为 T_i所对应的列向量的代数和
13)         delete T[j,i] from T[p-1,q-1];  // 推论 3.2
14)   重新得到 T_p*q;
15)   for(i=0;i<=p-k;i++){          //每 k 个行向量逻辑与运算得到所有频繁 k-项集
16)     for(j=k-1;j<p;j++){
17)       if count=T[0,0]*T[k-2,0]*T[j,0]+…
              +T[0,q-1]*T[k-2,q-1]*T[j,q-1]≥minsup_count then
18)         L_k=行向量对应的数据项的集合∪L_k; }
19)     }
20) }
```

我们以图 3-1 所示的事务数据为例来说明算法的具体处理。

TID	Itemset
T100	ABEF
T200	BE
T300	BC
T400	ABD
T500	ACF
T600	ABCE
T700	ABC

图 3-1 事务数据

图 3-1 共包含 7 条事务，分别用 T100、T200、…、T700 表示，{ABEF}、{BE}、…、{ABC}是每条事务所对应的项目集合。

扫描数据库，由定义 3.4 可以得到如图 3-2 所示的事务矩阵(即算法 3-4 的第 1 步)。矩阵的第 1-6 行分别表示事务数据中的各项是否被某条事务所包含，第 1-7 列分别表示事务数据中的各条事务是否包含某一项。

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	
A	1	0	0	1	1	1	1	5
B	1	1	1	1	0	1	1	6
C	0	0	1	0	1	1	1	4
D	0	0	0	1	0	0	0	1
E	1	1	0	0	0	1	0	3
F	1	0	0	0	1	0	0	2
	4	2	2	3	3	4	3	

图 3-2 事务矩阵

分别对得到的事务矩阵的行向量和列向量进行记数，该矩阵的行向量的代数和就是每个项目的支持数，列向量的代数和就是项集的长度。然后对事务矩阵进行修剪，删除支持数小于最小支持数的项所对应的行向量(第 2 步)，修剪后的矩阵如图 3-3 所示。

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	
A	1	0	0	1	1	1	1	5
B	1	1	1	1	0	1	1	6
C	0	0	1	0	1	1	1	4
E	1	1	0	0	0	1	0	3
F	1	0	0	0	1	0	0	2
	4	2	2	2	3	4	3	

图 3-3 修剪后的事务矩阵

由频繁 k-1-项集产生频繁 k-项集。通过对行向量的逻辑与运算(例如 $1001111 \wedge 1111011 = 1001011$)得到的向量中 1 的记数为 4，大于最小支持数，由此可得 AB 为频繁 2-项集，通过事务矩阵行向量的两两相与可得到频繁 2-项集的集合为{AB, AC, AE, BC, BE, AF}(第 15-19 步)。

根据推论 3.1 和推论 3.2，由频繁 2-项集产生频繁 3-项集时，可以删除矩阵的第 5 行和第 2、3、4 列，得到图 3-4 所示的矩阵(第 4-14 步)。

$$\begin{array}{c} \\ A \\ B \\ C \\ E \end{array}\begin{array}{c} \begin{array}{cccc} T_1 & T_5 & T_6 & T_7 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

图 3-4 修剪后的事务矩阵

对该矩阵的每三行进行逻辑与运算，可得频繁 3-项集的集合为{ABC, ABE}，即项集{ABC}和{ABE}在事务集合中的支持数不小于最小支持数。此时$|L_3|=2<4$(第 4 步)，由推论 3.3 可知，频繁项集的最大阶数为 3，整个过程结束。

3.4.3 算法分析

3.4.3.1 与 Apriori 算法的比较

上述基于事务矩阵的关联规则挖掘算法只对数据库进行一次扫描，其时间复杂度为 $O(nm^k)$，m 为修剪后的事务矩阵的行数，n 为修剪后的事务矩阵的列数，k 为待发现的频繁项集的阶数。

Apriori 算法在执行过程中，需要多次扫描数据库，算法的时间复杂度为 O(|D|)，|D|为事务数据库中包含的事务数量。在实际应用中，大型事务数据库中交易的数量一般远远大于商品的数量，即|D|>>n。

新算法将事务数据库映射为一个事务矩阵，把事务之间的关联反映到事务矩阵上，利用事务矩阵行向量的逻辑运算产生所有的频繁项集，并且根据一系列推论对事务矩阵进行合理的修剪，进一步减少了运算量，从而大大减少了系统开销，执行效率明显优于 Apriori 算法。

3.4.3.2 与现有算法的比较

针对 Apriori 算法的缺点，有关学者对此作了改进，并从不同的角度提出了矩阵与关联规则挖掘结合的改进算法，将事务数据库转换为基于内存的矩阵，在矩阵上找出所有的频繁项集，从而大大减少了数据库的扫描次数，提高了算法的效率。但是，这些算法各有优缺点。

文献[40]没有进行矩阵的压缩。文献[41]的矩阵数据结构较为合理，也进行了矩阵压缩，但压缩得不彻底。

本文改进了矩阵的数据结构：在一个单纯的事务矩阵中，添加一个辅助行和一个辅助列来分别对行向量和列向量进行计数，方便对矩阵进行修剪，删除相应的行和列从而实现矩阵的压缩。此外，笔者还对 Apriori 算法的性质进行引申得到了一系列的推论，从而有助于进行彻底的矩阵压缩，进一步提高算法的效率。

3.5 测试和实验

3.5.1 测试环境及实验数据

本文的实验环境为 AMD SP2800+，512DDR 内存，操作系统为 Windows XP Professional，所有程序用 VC++.NET 实现。

为了综合测试所提出算法的性能，本文分别采用 UCI 标准数据集 Mushroom[42]和合成数据集 T10I4D100K。为使比较的结果更为合理，本文选用的数据集是测试关联规则算法常用的数据集 Mushroom，该数据集可以从 UCI Machine Learning Repository 上免费获得。真实数据集的特点是频繁项目集分布密集，即使在较大的最小支持度下也会产生大量的频繁项目集。合成数据集 T10I4D100K 由 IBM 数据生成器[43]产生，其中的参数表示如下：|T|表示事务的平均长度，|I|表示潜在的频繁项目集的平均最大长度，|D|表示数据库中的事务数。

3.5.2 实验结果分析和比较

采用 Mushroom 数据集在不同的最小支持度下，算法的执行时间如图 3-5 所示。

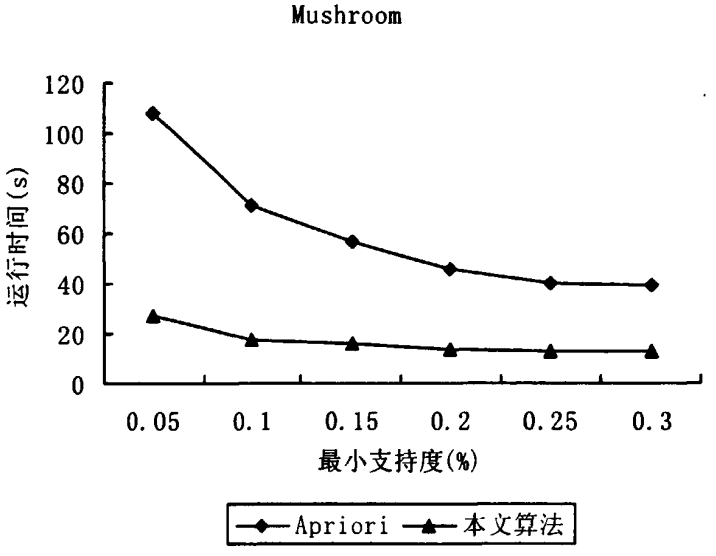

图 3-5 Mushroom 数据集上运行时间比较

接下来，采用 T10I4D100K 合成数据集，取定最小支持度为 1%，算法运行时间随事务数变化的测试结果如图 3-6 所示。

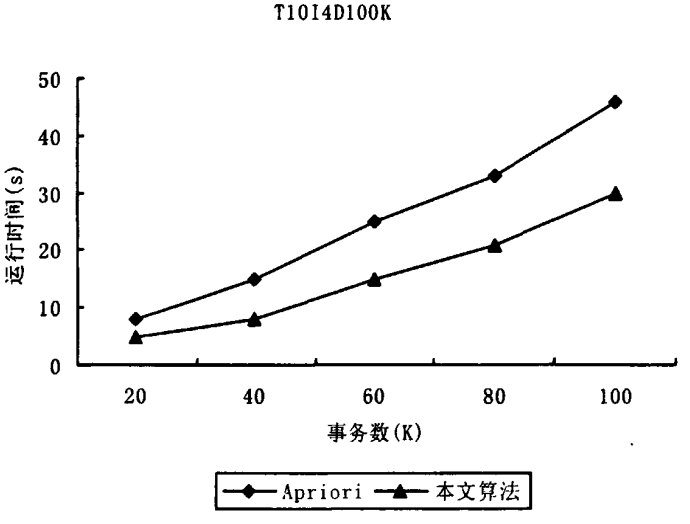

图 3-6 随事务数变化运行时间比较

由图 3-5 的实验结果可以看出，与 Apriori 算法相比，最小支持度越小时，本文算法的优势就越明显。这是由于最小支持度较小时，产生的频繁项集的规模就较大，相应的候选项目集的数量就变大了，扫描数据库的次数也增多了，而新算法采用矩阵向量进行逻辑运算的高效性就更明显。

图 3-6 的实验结果表明，随着事务规模的增大，新算法的优势也增加了。因为新算法只对数据库进行一次扫描，而 Apriori 算法随着事务规模的增大扫

描次数和时间也相应增大了。

综上所述，本文所提出算法的执行效率明显优于 Apriori 算法，并且随着最小支持度和事务数的变化，执行时间的变化比较稳定。

3.6 本章小结

本章的主要内容是在对经典关联规则挖掘算法 Apriori 进行研究和分析的基础上，针对 Apriori 算法的缺点，提出了一种基于事务矩阵的改进算法，并理论分析和实验证明了算法在性能上的优越性。将改进后的算法应用于 Web 使用挖掘可以高效地发现用户之间、页面之间以及用户浏览页面和网上行为之间存在的潜在关系。

第 4 章 一种新的用户频繁访问模式挖掘算法

随着条形码技术的发展以及在零售业中的普及使用，使得零售业很方便地收集和存储大量的商品销售信息。这些信息中，通常都会包含在某次商品销售过程中用户购买商品的时间、购买的物品、当然还有用户的信息(一般用户信息多以用户的信用卡或会员卡的形式保留)。我们可以从这些商业销售信息中分析出一些用户的购买模式，比如，在购买了自行车的所有用户中，可能有的用户会在两个月后购买打气筒；四年前购买了大众轿车的用户，可能在今年采取贴旧换新的购车行动，等等。可见，这些分析结果对商家而言是多么的重要，于是面对零售业中对商品销售数据分析[42]的需要，作为数据挖掘一个重要范畴的序列模式挖掘诞生了。而如今，序列模式挖掘已被广泛地应用于很多领域的数据分析、预测和知识提取中，比如自然灾害预测、生物数据分析、疾病诊断、客户购买行为能力预测、以及网站访问流量的分析等。

由于 Web 日志的访问模式是一种序列模式，因此将序列模式挖掘的方法应用于 Web 使用挖掘具有重要意义[43]。本章首先介绍了序列模式挖掘的基本概念并对其经典算法进行了分析研究，并在此基础上提出了一种基于有向图的用户频繁访问模式挖掘算法，通过理论分析和实验证明了该算法的性能优势。

4.1 序列模式挖掘概述

4.1.1 概念与问题描述

事件序列是数据的一种常见形式，如果将一个顾客在商场的一次购买视为一个事件，则该客户在一段时间内的若干次购买就形成了一个购买事件序列。事件序列中知识发现的一个重要问题，就是要从很多事件序列中发现那些经常出现的序列。例如，在一个超市交易数据库中，可能会发现这样的信息：在一段时间内，不少顾客先买了洗衣机，然后买了冰箱，接着买了空调，这三类产品就很可能是一条有用的序列模式。这种微观序列模式的发现在广泛采用信用卡和顾客卡进行消费的情况下更有意义。如果以信用卡或者客户卡作为顾客的标识，就可以得到顾客在一段时间内的购买记录，然后分析经

常出现的购买事件序列。从宏观的角度看，也可以用商场或超市所在地区的顾客群作为一个整体来分析，发现的模式是各类产品的销售模式及季节性趋势。实际上，当把一个城市或城区作为一个对象进行研究时，往往会有更富启发性的发现，相关供应链企业可以根据这样的序列模式确定营销战略。在涉及到全球供应链管理层次时，研究对象可以是一个国家或地区。

序列模式挖掘是近年来一个很活跃的研究领域，它最初是从发现描述一个事件序列的连续发生所遵循的规则开始的，目前已经提出多种序列模式的挖掘算法。下面介绍序列模式挖掘的一些基本概念。

交易数据库D的每条记录表示一次交易，包含以下域：顾客标识(Cust_id)、交易时间(Tran_time)、该交易所购买的商品集合(items)。每个商品称为一个数据项，简称项(item)，项的非空集合称为数据项集(itemset)，简称项集。项集所包含的项的个数称为项集的长度，长度为 k 的项集称为 k 阶项集(k-itemset)或称 k-项集。假设把项按字典顺序排列，k 阶项集 X 可以表示为{X[1], X[2], ..., X[k]}或(X[1], X[2], ..., X[k])。

数据库中每次交易的商品集合就构成一个项集。举例来说，假设顾客小冯的信用卡卡号为 0123456789，在 2007 年 6 月 30 日 20 点 55 分在某超市购买了香皂、牙膏、牛奶和饼干，那么，他的信用卡的卡号就可以作为顾客标识，这在整个数据库中是唯一的，交易时间就是 2007 年 6 月 30 日 20 点 55 分，交易的商品集合就是{香皂，牙膏，牛奶，饼干}，这个商品集合也就是项集，而且是个 4 阶项集。如果顾客小冯使用信用卡在稍后几天还有第二次交易，时间是 2007 年 7 月 4 日 10 点 30 分，购买的商品有方便面、豆腐干、花生米、啤酒和香烟，那么，第二次交易的项集就是{方便面，豆腐干，花生米，啤酒，香烟}。表 4-1 表示一个简单的顾客交易数据库，包括两位顾客的 5 条交易记录。

表 4-1 以客户号(Cust_id)及交易时间(Tran_time)排序的数据库

顾客标示(Cust_id)	交易时间(Tran_time)	物品(items)
0123456789	2007 年 6 月 30 日 20 点 55 分	香皂，牙膏，牛奶，饼干
0123456789	2007 年 7 月 4 日 10 点 30 分	方便面，豆腐干，花生米，啤酒，香烟
0123456789	2007 年 7 月 7 日 18 点 20 分	苹果，牛奶，巧克力
9876543210	2007 年 6 月 20 日 9 点 10 分	苹果，酸奶，纯净水
9876543210	2007 年 7 月 2 日 17 点 25 分	啤酒，卫生纸，袜子

定义 4.1：序列(sequence)是项集的一个非空有序集，也就是一个集合的集合，记为$\{s_1, s_2, ..., s_n\}$，其中s_i(i=1,2,...,n)是一个项集，表示在同一条交易中出现的商品。一个序列包含的项集的个数成为序列的长度，长度为 k 的序列称为 k 阶序列(k_sequence)。

特别地，一个项集就是一个一阶序列。在序列模式挖掘中，通常将构成序列的项集按照交易时间的先后顺序排列，把序列表示成$<s_1, s_2, ..., s_n>$的形式。序列中的项集有时间顺序，但不一定是连续出现的，序列 A 的第 i 个项集记作：A[i]。

定义 4.2：把一个顾客的所有交易按照交易时间的升序排列成一个序列$<T_1, T_2, ..., T_n>$，成为顾客序列(customer sequence)。

定义 4.3：设 X 是项集，T 是交易数据库 D 中的一条交易，如果 $X \subseteq T$，那么称 X 包含在交易 T 中。给定两个序列$A=<a_1,a_2,...,a_m>$和$B=<b_2,b_2,...,b_n>$，$m \le n$，如果存在一组整数$i_1<i_2<...<i_m$，使得$a_1 \subseteq b_{i1}$，$a_2 \subseteq b_{i2}$，...，$a_m \subseteq b_{im}$，则称序列 B 包含序列 A，或称 A 是 B 的子序列。在一个序列集合中，如果序列 A 不是其他任何序列的子序列，则称序列 A 为最大序列(maximal sequence)。

例如，<(3)，(4，5)，(8)>和<(7)，(3，8)，(9)，(4，5，6)，(8)>是两个序列，由于$(3) \subseteq (3，8)$，$(4，5) \subseteq (4，5，6)$，$(8) \subseteq (8)$，因此序列<(7)，(3，8)，(9)，(4，5，6)，(8)>包含序列<(3)，(4，5)，(8)>。但是序列<(3)，(5)>和<(3，5)>之间不存在包含关系。前者表示项 3 和项 5 是先后购买的，而后者则表示项 3 和项 5 是同时购买的。

定义 4.4：如果一个顾客序列中至少一个交易包含项集 X，则称该顾客序列支持 X。项集 X 的支持数是交易数据库 D 中支持 X 的顾客数，记为 X.sup，项集的支持度就是支持数与 D 中顾客总数之比。对于序列 A，如果一个顾客序列包含 A，则称该顾客支持 A。序列 A 的支持数是 D 中支持 A 的顾客数，记为 A.sup。同样，A 的支持度等于支持数与 D 中顾客总数之比。

定义 4.5：支持阈值表示项集或序列在统计意义上的最低重要性。如果交易数据库的顾客数是固定的，可以用最小支持数 minsup_count 来代替支持阈值。如果项集 X 的支持数不小于最小支持数，即 $X.sup \ge minsup\_count$，则称 X 是频繁项集。如果序列 A 的支持数不小于最小支持数，即 $A.sup \ge minsup\_count$，则 A 是大序列或频繁序列。

根据前面的定义，大项集也就是一阶大序列，大序列中的每一个项集都必须具有最小支持数，即任何大序列都是由大项集构成的。

定义 4.6：序列模式是具有最小支持数的最大序列，也就是要满足两个条件：支持数不小于最小支持数；不是其他大序列的子序列。

定义 4.7：序列规则可以表示成 A⇒B 的形式，其中 A 和 B 都是序列。令 C=<A, B>，规则的支持数定义为 C.sup，规则的置信度定义为 C.sup/A.sup。支持度指规则出现的频度，而置信度主要指规则的强度。令 minconf 为置信阈值，代表规则的最低可靠性。当一个规则的支持度和置信度分别不小于支持阈值和置信阈值时，称其为强规则。

序列模式挖掘多数以发现序列模式为目的，但也可以得到序列规则。序列模式的挖掘可以描述为：在一个给定的顾客交易数据库的所有顾客序列集中，寻找满足用户指定的支持阈值的序列模式，或者是找到满足用户指定的支持阈值和置信阈值的序列规则。两类问题的步骤基本相同，本章主要对第一类问题进行研究。

4.1.2 序列模式挖掘过程

序列模式挖掘的问题源于关联规则挖掘，是关联规则挖掘的延伸，对它的研究开始于 1995 年。在关联规则挖掘的 Apriori 算法的基础上，Agrawal 和 Srikant 提出了几种序列模式的挖掘算法，即 AprioriAll、AprioriSome 及 DynamicSome。序列模式的挖掘主要基于频繁项集的挖掘，基本思想是将交易数据库转换为以顾客序列为记录的数据库，再利用频繁项集搜索算法计算频繁项集，作为一阶大序列，然后依次搜索所有阶的大序列，最后从大序列中删除子序列，最终得到序列模式。

序列模式挖掘通常分以下 5 个步骤进行：排序阶段(sort phase)、大项阶段(large itemset phase)、转换阶段(transformation phase)、序列阶段(sequence phase)和最大项序列阶段(maximal phase)。其中，序列阶段的任务是计算所有的大序列，这个步骤是序列模式挖掘的核心，我们将在下一节对大序列计算的典型算法进行介绍。

4.2 序列模式挖掘算法分类及典型算法

4.2.1 序列模式挖掘算法分类

序列模式的发现算法基本上可以分为两大类。

第一类是基于 Apriori 特征的、逐层(lever-wise)的发现算法，包括 AprioriAll

算法和 GSP[44]算法等。PSP 算法是 GSP 算法的改进。AprioriAll 及 AprioriSome 算法利用频繁模式的所有的非空子集都是频繁的这一 Apriori 性质，对搜索空间进行了压缩。GSP(Genaralized Sequential Patterns)算法是一种典型的基于 Apriori 性质的算法，算法将候选序列组织在哈希树(hash-tree)中，而 PSP 算法将候选序列组织在前缀树(prefix-tree)中，因而提高了效率。经典算法基于的数据组织为水平(horizontal)方式，而 SPADE 算法基于垂直(vertical)数据方式，并且使用了一种格搜索技术和简单时序连续操作来发现所有的序列模式。这类算法最先由 R.Agrawal 等人提出。

第二类是 J.Han 等人提出的基于模式增长的算法，包括 FreeSpan [45]算法、PrefixSpan[46] 算法等。这类算法采取了一种分而治之(divide-and-conquer)的模式增长策略，基于现有的模式序列将序列数据库递归地投影到一序列更小的投影数据库，只需通过探索每个投影数据库中局部的频繁部分使序列模式得到增长。下面我们分别选取两类算法中有代表性的经典算法，介绍用它们来解决序列模式挖掘的思路。

4.2.2 AprioriAll 算法

AprioriAll 算法是 Agrawal 和 Srikant 首先在文献[42]中提出的，同时提出的算法还有 AprioriSome 以及 DynamicSome 算法。之后，Apriori 类算法又经过了很多不同角度的改进，算法性能在某种程度上提高了很多。

AprioriAll 算法的过程与频繁项集的搜索过程类似，也是由 1 阶频繁序列构造 2 阶候选序列，扫描数据库计算候选序列的支持数，得到 2 阶频繁序列，这样依次循环，直至找到所有的频繁序列为止。AprioriAll 算法的详细过程描述如下：

算法 4-1：AprioriAll 使用根据候选序列生成的逐层迭代找出频繁序列。

输入：顾客事务数据库 S；最小支持度阈值 min_sup。

输出：S 中所有频繁序列 L。

1) L_1={frequent items}; //1 阶大序列

2) for ($k=2; L_{k-1} \neq \phi, k++$){

3) C_k=Get_candidate(L_{k-1}); //构造候选序列

4) for each customer sequence c∈Dt{

5) C_t=subset(C_k,c); //计算顾客序列 c 所包含的候选序列

```
6)      for each s∈C_t
7)          s.sup=s.sup+1;          //累计支持数
8)      }
9) L=∪_k L_k;
```

为了减少候选序列的个数，候选序列的构造用 Get_candidate 函数来实现，它与本文第 3 章介绍的关联规则挖掘算法 Apriori 中候选项集构造函数 apriori_gen 类似。

性质 4.1：如果 A 是频繁序列，那么 A 的所有子序列也必定是频繁序列。反之，如果一个序列的某个子序列不是频繁序列，那么这个序列也一定不是频繁序列。

这是因为如果一个顾客序列包含序列 A，那么必然也包含 A 的所有子序列。因此，A 的支持数必然不大于它的子序列的支持数。根据这个性质，get_candidate 通过自连接和剪枝两个步骤来构造候选序列。

(1) 自连接步

将 L_{k-1} 与 L_{k-1} 连接产生候选序列 C_k。设序列 s_1 与序列 s_2 属于长度为 k-1 的频繁序列。为了方便起见，我们在生成序列数据库的时候，就将项集中的项按照字母序排列。如下情况序列 s_1 与序列 s_2 是可连接的：如果将序列 s_1 的第一项删除后得到的序列与将序列 s_2 的最后一项删除后得到的序列相同，则将 s_1 与 s_2 连接。序列 s_1 与序列 s_2 连接产生的序列，就是将 s_2 的最后一项加入到 s_1 中。加入时分以下两种情况：如果 s_2 的最后一项是 s_2 的单独的一个项集，则它在 s_1 中也作为单独的项集，否则将它作为一项加入到 s_1 的最后一个项集中。在连接 L_1 与 L_1 时，s_2 的最后一项分别作为一个单独的项集和项集的一部分加入到 s_1 中。

(2) 剪枝步

删除支持度小于最小支持度阈值的候选序列 C_k 中所有的子序列。C_k 是 L_k 的超集，这意味 C_k 的成员可以是不频繁的，但是所有的长度为 k 的频繁序列都在 C_k 中。扫描数据库，确定 C_k 中每个候选序列的支持度，从而确定 L_k。

然而，C_k 可能很大，这样就涉及很大的计算量。为了压缩 C_k，我们利用了性质 4.1：任何非频繁的 k-1 项集都不可能是频繁 k 项集的子集。因此，如果一个候选的长度为 k 的序列，其长度为 k-1 的子序列不在 L_{k-1} 中，则该候选序列也不可能是频繁的，从而可以从 C_k 中删除，以减少扫描计算候选序列的

支持度的工作量。

下面，举例说明一下 get_candidate 函数的处理过程。假设 L_3={<1 2 3>，<1 2 4>，<1 3 4>，<1 3 5>，<2 3 4>}，利用 get_candidate 方法构造 4 阶候选序列。方法如下：

(1) 对 3 阶频繁序列做自连接

<1 2 3><1 2 4>→<1 2 3 4>，<1 2 4 3>

<1 3 4><1 3 5>→<1 3 4 5>，<1 3 5 4>

因此，C_4={<1 2 3 4><1 2 4 3><1 3 4 5><1 3 5 4>}。

(2) 检查每个候选序列的子序列

由于<1 2 4 3>的子序列<2 4 3>$\notin L_3$，从 C_4 中删除<1 2 4 3>。同样，<1 3 4 5>的子序列<3 4 5>$\notin L_3$，从 C_4 中删除<1 3 4 5>，而<1 3 5 4>的子序列<3 5 4>$\notin L_3$，从 C_4 中删除<1 3 5 4>。最后，C_4={<1 2 3 4>}。

AprioriAll 算法容易生成庞大众多的候选序列，并且在计算候选序列支持度时需要反复多次扫描数据库，因此造成了很大的开销。同时，随着数据库的序列模式长度的增长，候选序列的数量会成指数级增长，这对 AprioriAll 算法而言，发现长序列模式是困难的。

4.2.3 PrefixSpan 算法

Pei 等在文献[46]中提出了基础投影的 PrefixSpan(Prefix-projected Sequential pattern mining)算法，该算法首次将前缀以及投影的概念引入到了序列模式挖掘领域。PrefixSpan 算法通过实验和理论的验证可以发现数据库用户序列支持的所有的频繁序列，同时又能极大地减小在生成候选序列时的系统开销。更值得一提的是，PrefiSpan 算法利用投影将原数据库进行分割的方式，明显地提高了挖掘数据库序列模式的效率，所以，从目前情况看，在需要对大型的序列数据库做序列模式挖掘的时候，PrefixSpan 算法以其在性能及效率等方面的优势都可以作为首选算法考虑。

在介绍 PrefixSpan 算法思想之前，我们先将首次在序列模式挖掘中用到的一些概念做个介绍。

(1) 前缀

两个序列α=<a_1 a_2...a_n>，β=<b_1 b_2...b_m>，$m \le n$，如果当 $i \le m-1$ 时，使得 $b_i=a_i$，$b_m \subseteq a_m$，且 a_m 中(a_m-b_m)的项均出现在 b_m 之后。此时就称β是α的一个前缀。

例如，序列α=<(a)(a,b,c)(a,c)(d)(c,f)>，β=<(a)(a,b,c)(a)>，β就是α的一个前缀。而序列α=<(a)(a,b,c)(a,c)(d)(c,f)>，β=<(a)(a,b,c)(c)>，就不满足这样的关系。

(2) 投影

序列α和序列β，当且仅当β是α的一个子序列，α′也是α的一个子序列且满足以β为前缀，而且对α′而言不存在超序列α″也是α的子序列且也以β为前缀。此时就称序列α′是序列α关于β的投影。

例如，序列α=<(a)(a,b,c)(a,c)(d)(c,f)>，β=<(b,c)(a)>，则序列α关于β的投影α′为：α′=<(b,c)(a,c)(d)(c,f)>。

(3) 后缀

两个序列α′=<a_1 a_2...a_n>，β=<a_1 a_2...a_{m-1} a_m′>，m≤n，且β是α的一个前缀，序列α′是序列α以β为前缀的投影。那么，序列γ=<a_m″ a_{m+1}...a_n>，就是序列α以β为前缀的后缀，其中，a_m″=(a_m-a_m′)。

例如，序列α=<(a)(a,b,c)(a,c)(d)(c,f)>，β=<(a)(a,b,c)(a)>，那么序列γ=<(_,c)(d)(c,f)>即为序列α以β为前缀的后缀。

PrefixSpan 算法与之前所有算法的不同之处在于，它并不以原数据库为参考来考虑所有可能出现的频繁序列，而是只检查前缀序列是否频繁。如果前缀频繁，则把其相应的后缀序列建成投影数据库。同样，在每个投影数据库中，只检查该投影数据库中前缀序列是否频繁。如此递归，整个过程中不需要生成候选序列。而且，通过 PrefixSpan 算法中的递归过程，前缀在不断地扩展，以此可以产生序列数据库中所有的频繁序列。

下面给出 PrefixSpan 算法的伪代码描述，其中，D_S 为序列数据库；min_sup 是最小支持度阈值；a 为一个序列模式，初始值为<>；l 为 a 的长度，初始值为 0；$D_S|_a$ 是 a 在序列数据库 D_S 上的投影。我们执行该算法时调用 Call PrefixSpan(<>，0，D_S)。

算法 4-2：PrefixSpan 通过递归调用找出所有频繁序列。

输入：顾客事务序列数据库 D_S；最小支持度阈值 min_sup。

输出：D_S 中所有频繁序列 L。

方法：

(1) PrefixSpan(a, l, $D_S|_a$){

(2) 扫描 $D_S|_a$ 一次；

---

(3) b 是长度为 1 的序列，如果 b 是频繁的，则加入到长度为 1 的频繁序列

(4) 集合 B；

(5) For each b∈B{

(6) b 对 a 做序列扩展或项集扩展形成序列 a′，将 a′加入序列 L，输出 a′；}

(7) For each a′{

(8) 建立其投影数据库 $D|_{a'}$，call PrefixSpan(a′,l+1,$D_S|_{a'}$)；}

(9) }

PrefixSpan 算法的执行步骤可以描述如下：

(1) 对序列数据库扫描一次得到全部频繁项目 n，即所有长度为 1 的频繁序列集合。

(2) 将频繁序列完整的集合分为 n 个具有不同前缀的频繁序列的子集。

(3) 通过构造相应的投影数据库，并在其中递归地挖掘发现频繁序列的子集。

从算法的过程描述中不难看出，整个算法采用广度遍历的方式来递归生成各个投影数据库，不需要产生候选序列模式，大大缩减了检索空间。与原始的序列数据库相比，投影数据库的规模不断减小。算法的主要开销在于投影数据库的构造，如果存在大量的序列模式，并且需要为每一个序列模式建立一个投影数据库，那么开销就比较大。PrefixSpan 算法花费大量的资源去产生中途的临时数据库，但其记录前缀的思想，在用户序列的长度普遍较长时，对提高数据库扫描速度很有利。

4.3 基于有向图的用户频繁访问模式挖掘新算法

4.3.1 用户频繁访问模式

4.3.1.1 问题模型

由 2.2 节中对 Web 使用挖掘的研究可以知道，通过对原始的 Web 日志文件进行预处理，Web 日志文件中的每条记录是由用户访问的页面组成的序列。可见，用这些序列构建 Web 日志访问序列数据库是序列数据库的一个特殊形式，因此对 Web 日志访问序列模式，我们需要在常规序列模式模型定义的基础上给出更有针对性的定义。

定义 4.8：我们将用户访问的某个 Web 页面记为 p，Web 日志文件中的每条记录可以看成是由某个用户连续访问的若干页面组成的，页面按用户访问的先后顺序排列，称为访问序列。记用户会话集为 D_{SS}，$P=\{p_1,p_2,\ldots,p_m\}$是 D_{SS} 中全体 Web 页面组成的集合。一个用户会话 S 是一个页面集，它是 P 的子集，每个用户会话都与一个唯一标识符 Sid(Session Identifier)相联系。不同的用户会话一起组成用户会话集 D_{SS}。一条 Web 日志文件中的访问序列 $S=<p_i, p_j, \ldots, p_k>$，其中 $i, j, k \in P_n$，n 称为访问序列的长度。长度为 n 的序列也被叫做 n 阶序列。

定义 4.9：给定一个用户会话的集合 $D_{SS}=\{s_1,s_2,\ldots,s_n\}$，称为 Web 访问序列数据库。$s_i(1\le i\le n)$是访问序列。访问序列 s 在 D_{SS} 上的支持度 support(s)就是 Web 访问序列数据库中包含 s 的访问序列的数目。用户给定最小支持度阈值 min_sup，如果 support(S)≥min_sup，则序列 s 叫做频繁序列或者用户的访问序列模式。

定义 4.10：给出一个 Web 访问序列数据库，对其中每个用户会话执行最大向前引用法，找出所有 MFP 并存储在数据库中，称为 Web 事务数据库。用户频繁访问模式挖掘就是给定一个 Web 事务数据库和最小支持度阈值，挖掘数据库中所有的频繁序列。

4.3.1.2 用户频繁访问模式挖掘步骤

挖掘用户频繁访问模式(或者频繁访问路径)是 Web 使用挖掘的一个重要研究内容，它主要是通过了解用户的访问模式来完善站点结构，并进一步为用户提供个性化服务。

频繁访问路径挖掘的基本思想是[47]：经过数据清理（数据净化）、用户识别、会话识别、路径补充的数据预处理阶段得到的用户会话文件，借助预扩展有向树模型，识别出用户会话文件中的所有最大前向访问路径，然后从中发现频繁出现的连续子序列，从而得到频繁访问路径。最大前向路径 MFP[48]是指扩展有向树中的无后退访问行为的最大的连接页面的序列，即这些页面中没有任何页面在之前被访问过。频繁访问路径是 MFP 中支持度大于指定的最小支持度阈值的连续页面序列。路径的支持度是指包含该路径的用户会话的个数。显然，支持度越大，说明使用该条访问路径的用户越多。

用户频繁访问模式的挖掘步骤如表 4-2 所示。

表 4-2 频繁访问模式挖掘的步骤

步骤		结果
1	WebLog	服务器日志文件
2	日志文件导入数据库	原始日志数据
3	数据净化	净化后的日志数据
4	用户会话识别	用户会话文件
5	MFP 识别	Web 事务数据库
6	频繁访问路径识别	用户频繁访问路径

4.3.1.3 最大前向路径

最大前向路径(MFP)系列的概念是由 Chen M S 等人提出的。Chen 等首先将数据挖掘技术应用于 Web 服务器日志挖掘，发现用户的浏览模式。通过发现 MFP，将用户会话分割成一系列事务，然后采用与关联规则相似的方法挖掘频繁的浏览路径。

用户在浏览网站时，通过页面之间的超链接和图标按钮访问各相关页面，完成“前进”和“回退”的操作。所以，一些页面被重复访问的原因可能是它的位置，而不是内容。例如，用户想到达一个同级页面，通常先点击“回退”图标，然后连接到目的页面，而不是新打开一个 URL。因此，要想从 Web 服务器的 Log 中抽取出有意义的用户访问模式，需要考虑这种“回退”访问，发现真正的访问行为。假定“回退”引用主要是方便浏览操作，而不是以浏览页面内容为目的，我们的工作集中在发现前向引用模式。有一点要注意：一个“回退”引用是指同一用户再次访问前面访问过的页面。“回退”引用的出现标志着一个前向引用路径的结束。于是，定义最大向前引用路径是在用户会话中的第一页到回退的前一页组成的路径。

最大前向引用法识别用户访问事务的步骤为：首先将每个用户会话中的访问页面序列表示成一棵扩展有向树，然后利用扩展有向树找出 MPF 集合，这是进行用户频繁访问模式序列挖掘的基础。

例如，用户会话{A, B, C, D, C, B, E, F, A, G}的扩展有向树如图 4-1 所示。所谓的扩展有向树分两部分，一部分是通常意义的有向树，即树中用实线表示的部分；另一部分是扩展部分，在树中用虚线表示，这部分是遍历时的后

退路径。扩展部分只是为了便于清楚地说明用户的行为，实际上有用的只是有向树部分。

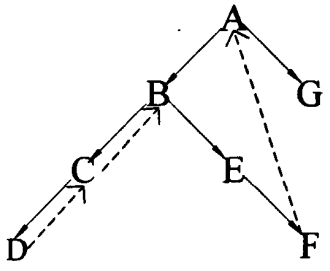

图4-1 扩展有向树示例

运用 MFP 算法，可以得到其中的最大向前引用为：{A, B, C, D},{A, B, E, F},{A, G}。根据文中定义的用户行为模型，用户访问网站的活动可以用(起始页，目的页)的方式来表示。对于一个新的访问会话，它的起始点可以定义为空节点。用访问序列{(s_1, d_1), (s_2, d_2), …, (s_n, d_n)}来表示一个用户的访问序列，然后生成其中的最大向前引用。首先按照用户的 ID 排列 Log 中的访问信息，可以得到访问序列{(s_1, d_1), (s_2, d_2), …, (s_n, d_n)}，对于每一个用户来说(s_i, d_i)满足时间序。MFP 算法用于获取每个用户的最大向前引用。

定义 D_F 为存放最大向前引用的数据库，即 Web 事务数据库，则 MFP 算法执行过程描述如下：

算法 4-3：MFP 算法。

```
Step1: int i=1;
       String y=null;
       Flag F=1 to indicate a forward traversal;
Step2: A=Si,B=di;
       If(A==null)
         {
           Write out current String y (if not null) to database DF;
           y=B;
           Go to step5;}
Step3: if(B==some reference in String y)   //this is a backtrace to a previous reference
          {
           If(F==1)
```

Write out String y to database D_F;

Discard all the reference after i-th one in String y; /*Discard all the backtrace reference after the first one*/

F=0;

Go to Step5; }

Step4: append B to the end of String y;

If(F==0)

F=1;

Step5: Set i=i+1;

If(sequence is not completed scaned)

Go to Step2;

对每个用户会话执行最大向前引用法，找出所有 MFP 之后，则挖掘用户频繁访问序列模式的问题，就转化为在所有的用户会话的 MFP 中发现频繁出现的连续子序列的问题。

4.3.2 新算法及其分析

4.3.2.1 相关定义及算法基本思想

推论 4.1：如果 $s=\{p_1, p_2, \ldots, p_i\}$是 i 阶频繁访问序列，则$\{p_1, p_j\}$, $\{p_2, p_j\}$, …, $\{p_i, p_j\}$皆为 2 阶频繁访问序列是序列 $s'=\{p_1, p_2, \ldots, p_i, p_j\}$为 i+1 阶频繁访问序列的必要条件。

证明：由性质 4.1，如果 $s=\{p_1, p_2, \ldots, p_i, p_j\}$为 i+1 阶频繁序列，则一定有 $\{p_1, p_j\}$, $\{p_2, p_j\}$, …, $\{p_i, p_j\} \in L_2$，反之，则不一定。证毕。

定义 4.11：有向图 DG(V, {A})定义为：① 有向图中的顶点集 V 为 Web 事务数据库中 p_i的集合$\{v_1, v_1, \ldots, v_n\}$，每一个顶点包含两个属性：顶点名称和指向关联顶点的指针；② 有向图中的弧集是连接顶点的弧<vi, vj>的集合。每一条弧包含一个属性：弧的名称<vi, vj>。

定义 4.12：在 DG 中，如果有$\{v_i, \ldots, v_{i+j}\} \in V$，且这 j 个顶点之间弧的总数为 C_n^2 条，把这个子图称为 DG 的 n 阶频繁序列模式生成图(Frequent Sequential Pattern Generation Gragh, FSPGG)。对于一个 n 阶频繁序列模式生成图，如果在 DG 中除这 n 个顶点以外的任意一个顶点 v_r，与这个 n 阶频繁序

列模式生成图不能构成 n+1 阶频繁序列模式生成图，则称这个图为最大频繁序列模式生成图。

定义 4.13：定义页面的支持事务向量为 BV，向量的长度等于数据库中的事务数，如果第 i 个事务表示的访问序列包含页面 Pj，则向量的第 i 个比特位取值为 1，否则取 0。

基于上述定义，本文提出了一种基于有向图的用户频繁访问模式挖掘算法(User Traversal Patterns Mining Based-on Digragh, UTPMBD)，其基本思想是：扫描 Web 事务数据库一遍得到所有访问页面的支持序列向量。按照如下的规则构造有向图 DG：计算 BVpi∧BVpj，如果结果中 1 的个数≥min_sup，即{pi, pj}在 Web 事务数据库中的支持度大于最小支持度阈值时，则{pi, pj}为 2 阶频繁序列，记录相关信息并增加顶点 vi,vj 及弧<vi,vj>。遍历 DG，可以得到其中的所有最大频繁序列模式生成图。由推论 4.1 可知，n 阶最大频繁序列模式生成图的 n 个顶点所对应的页面访问序列，有可能成为 n 阶频繁访问序列模式，计算 $BV_1 \wedge BV_2 \wedge \ldots \wedge BV_n$，如果结果得到的比特向量中 1 的个数不小于最小支持度阈值，则为 n 阶频繁序列模式。

基于算法的基本思想，UTPMBD 算法的整个处理过程大致可以分为 3 个步骤：

第 1 步，扫描 Web 事务数据库一次，得到所有页面的支持事务向量，进而得到所有的 1 阶频繁序列；

第 2 步，构造 DG，并记录相关信息；

第 3 步，遍历 DG 得到所有的 k(k>2)阶最大频繁序列模式生成图，计算得到所有的用户频繁访问模式。

4.3.2.2 算法描述

下面，给出 UTPMBD 算法中关键步骤的伪代码描述。

算法 4-4：构造有向图 DG。

输入：Web 事务数据库 D_F 中包含所有页面 $p_1,p_2,\ldots p_m$ 的支持序列向量 $BV_1,BV_2,\ldots BV_m$；最小支持度阈值 minsup。

输出：有向图 DG。

方法：

(1) $L_2=\phi$;

```
(2)  for(i=0;i<m-1;i++){
(3)      for(j=i+1,j<m,j++){
(4)        if (the number of '1' in (BVi∧BVj))≥minsup then{
(5)          增加顶点 v_{i+1},v_{j+1} 和弧<v_{i+1},v_{j+1}>到 DG;
(6)          L2=L2∪{p1,p2};}
(7)      }
(8)   }
(9)  Return DG;
```

算法 4-4 描述了构造 DG 的过程，在将每个事务中的页面按序添加到 DG 以后，就可以对 DG 进行遍历从而得到所有的最大频繁序列模式生成图，进而得到所有的用户频繁访问模式。下面给出这个步骤的伪代码描述。

算法 4-5：遍历 DG 得到所有的用户频繁访问序列。

输入：DG；最小支持度阈值 minsup。

输出：用户频繁访问序列。

方法：

```
(1) join v_l in FSPGG; //按字典顺序由 v_l 开始遍历，FSPGG 为最大频繁模式生成图
(2) while((v_i=v_l→unvisitedadjvex)≠ϕ) then{
(3)   join v_i in FSPGG;
(4)   while( vi→unvisitedadjvex≠ϕ) then{
(5)     vj=vi→unvisitedadjvex;      /*从 v_i 的未被访问的邻接点中按字典顺序选择
      下一个顶点*/
(6)     if (is_FSPGG(FSPGG, vj)) then{
(7)       join vj in FSPGG;
(8)       G=G∪FSPGG; }            //G 为最大频繁模式生成图的集合
(9)   }
(10) for each FSPGG∈G
(11)   frequent_seqential_generation(FSPGG);

Procedure is_FSPGG(FSPGG, v_k)
(1) if( for each v_i∈FSPGG except v_j   v_i.adjvex∩v_k≠ϕ) then{ /*v_j 为 FSPGG 中按序排
```

```
的最后一个顶点 */
(2)  join vk in FSPGG;
(3)  if(vk.unvisitedadjvex≠ϕ)  then{
(4)    vl=vk.unvisitedadjvex;
(5)    is_FSPGG(FSPGG, vl)}   /*递归调用 is_FSPGG 判断 vk 的邻接点是否可以
     加入 FSPGG*/
(6)  else return true; }
(7) else return false;

Procedure frequent_sequential_generation(FSPGGn) //FSPGGn 为 n 阶频繁模式生成图
(1) for(i=n;i>3;i--){
(2)  if the number of "1" in BVpi∧...BVpk>minsup then{ /*pi,...pk 为 FSPGG 中的各
     个顶点所表示的页面 */
(3)    L=L∪<pi,...pk>;
(4)    break;}
(5)  else for each FSPGGn-1∈FSPGGn  /*FSPGGn-1 为 n 阶 FSPGG 包含的 n 个 n-1
     阶 FSPGG*/
(6)    Frequent_sequential_generation( FSPGGn-1);
(7) }
```

图 4-2(a)是一个用户会话集，采用最大向前引用算法处理后得到的 Web 事务集如图 4-2(b)所示。我们就以它为例来说明算法的具体处理。

会话名	访问序列
S100	A,B,C,D,B,E
S200	A,B,C,D,B,E,G
S300	C,F,C,D

事务名	访问序列
T100	A,B,C,D
T200	A,B,E
T300	A,B,C,D
T400	A,B,E,G
T500	C,F
T600	C,D

(a)用户会话集　　(b)采用 MFS 算法后的访问事务集

图 4-2 用户会话集及预处理

第一步，扫描 Web 事务数据库一遍，得到所有页面的支持序列向量。对于图 4-2(b)所示的事务集，有 BV_A = 111100，BV_B =111100，BV_C=101011，BV_D=101001，BV_E=010100，BV_F =000010，BV_G=000100。设最小支持度为 2，对于每个页面的支持事务向量，其中 1 的个数即为页面的支持度。F、G 的支持度皆为 1<minsup，所以 1 阶频繁序列为<A>，<B>，<C>，<D>，<E>。

第二步，将第一步得到的所有 1 阶频繁序列按其访问的先后顺序两两进行“∧”运算。如果计算结果中 1 的个数大于最小支持度，则增加相应的顶点和它们之间的弧到 DG(算法 4-4 第 2-9 步)。例如，$BV_A \wedge BV_B$=111100，计算结果中 1 的个数为 4>minsup，所以增加 v_1, v_2, <v_1, v_2>。执行完第二步得到的有向图如图 4-3 所示，<A, B>, <A, C>, <A, D>, <A, E>, <B, C>, <B, D>, <B, E>, <C, D>为所有的 2 阶频繁序列。

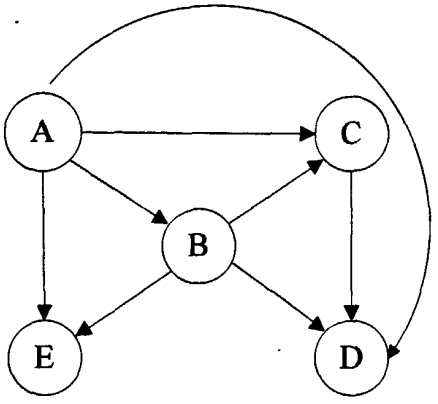

图4-3 算法得到的有向图

第三步，从顶点 A 开始执行算法的第三步，按字典排序 A 的下一个邻接顶点为 B，B 指向的下一个邻接顶点为 C。这时，有 $C \cap A.adjvex \neq \phi$，即 DG 中有一条从 A 指向 C 的弧，所以有{A, B, C}∈FSPGG(算法 4-5 第 1-7 步)。接下来判断：因为 $D=C \rightarrow unvisitedadjvex$，且 $D \cap A.adjvex \cap B.adjvev \neq \phi$,扩展 D 到 FSPGG，即{A, B, C, D}∈FSPGG(Procedure is_FSPGG 第 1-5 步)。再有 $BV_A \wedge BV_B \wedge BV_C \wedge BV_D$=101000，其中 1 的个数为 2≥minsup，所以<A, B, C, D>∈L_4(算法 4-5 第 10-11 步)。继续执行算法得到<A, B, E>∈L_3。由于最长的序列是 4 阶，因此 4 阶频繁序列都是最大序列。首先删除 4 阶序列< A, B, C, D >的子序列，剩下的 3 阶频繁序列< A, B, E >是最大序列。然后删除<A, B, E>的所有子序列。到此为止，已经没有可以再删除的子序列，最后得到的用户频繁访问模式为{< A, B, E >, < A, B, C, D >}。

4.3.2.3 算法分析

本章提出了一种基于有向图的用户频繁访问模式挖掘算法，该算法只对 Web 事务数据库进行一次扫描，并将所有页面之间的序列信息记录在有向图中，通过对有向图的遍历得到最大频繁序列模式生成图集，对其中的元素所表示的页面在事务数据库中的支持事务向量做逻辑“与”运算，从而得出所有的最大序列，也就是用户的频繁访问路径。

算法在空间上的代价主要是存储 DG（有向图），而 DG 只需要用 N 个顶点和 K 条弧来记录有关频繁序列的所有信息。算法在时间上的代价主要取决于对 DG 的遍历。对图进行搜索时，遍历 DG 的时间复杂度是由图中顶点的个数 N、邻接顶点的个数 K 以及频繁序列的长度 L 决定的，它的平均估计执行时间的复杂度为 O(N*K*L)。

4.3.3 测试和实验

实验使用和本文 3.5.1 节相同的测试环境，实验数据采用某大学网站的 Web 服务器日志文件，部分 Web 日志格式如图 4-4 所示。

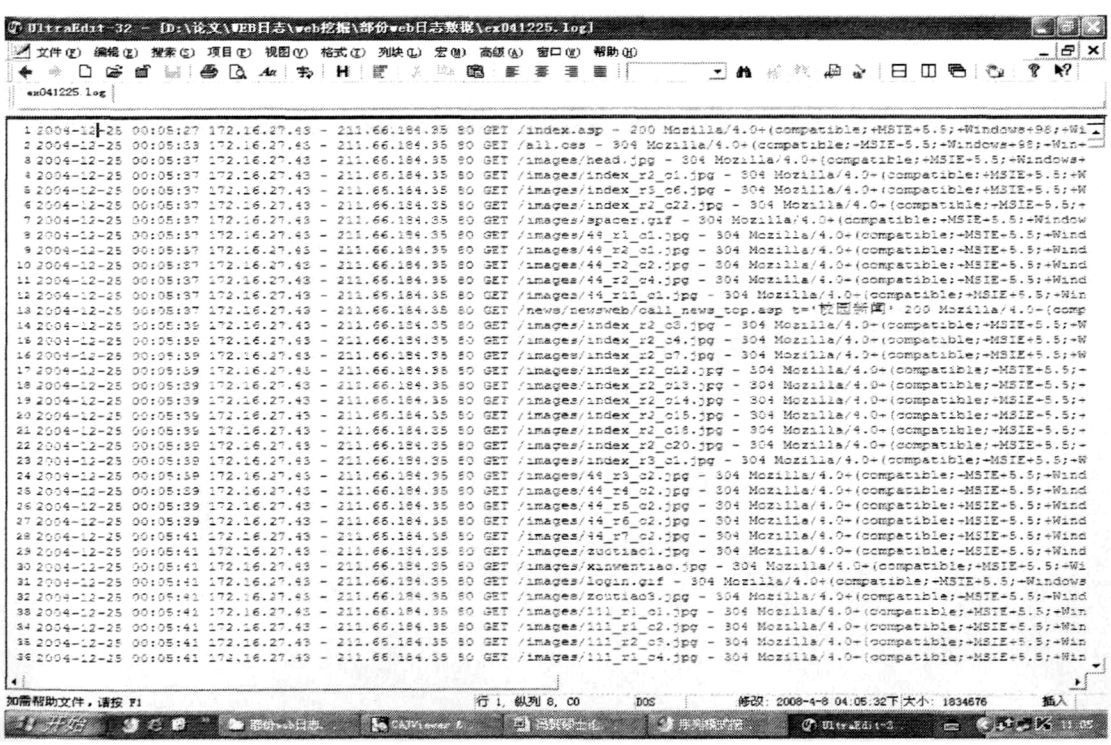

图 4-4 部分 Web 日志

由于 PrefixSpan 算法的性能特征比较适合于用户频繁访问模式的挖掘，

所以本文以 PrefixSpan 算法作为对比算法进行实验来比较性能。本文选取该大学网站其中一天的 Web 日志文件，共包含 125889 条访问记录，经过数据预处理并执行最大向前引用算法后，共得到 MFP19563 条。在不同的最小支持度下，算法的执行时间如图 4-5 所示。

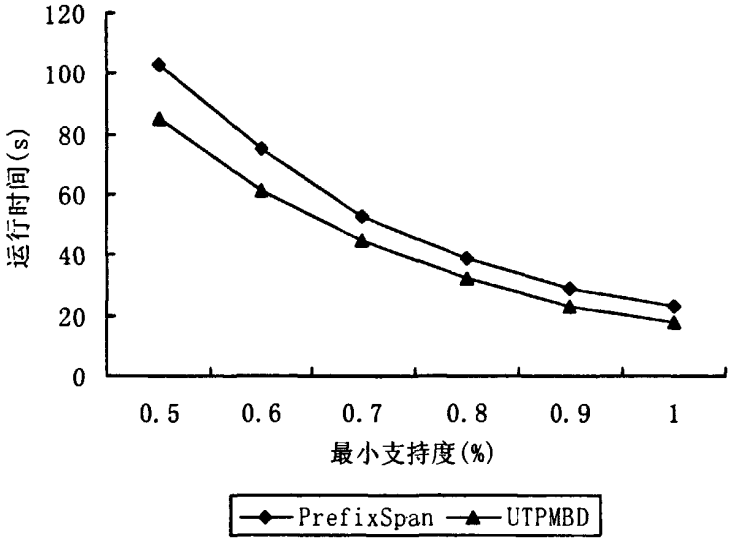

图 4-5 不同支持度运行时间比较

接下来，我们从该大学网站日志中选取多个 Web 日志文件并对其进行处理得到 MPF 分别为 10K、20K、...、60K 条的用户访问序列。从文献[48]可以知道，当最小支持度小于 1%时，随着最小支持度的减小，PrefixSpan 算法的性能优势明显增大。同时，设定一个较小的最小支持度，算法会产生大量的频繁序列，从而使得算法效率的比较效果更加明显。所以，这里取定最小支持度为 0.5%，图 4-6 为本算法与 PrefixSpan 算法的内存使用情况比较。

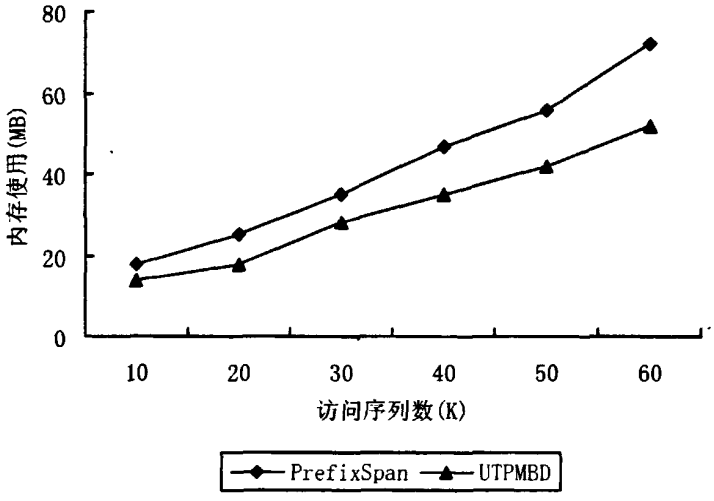

图 4-6 随访问序列数变化内存使用比较

由图 4-6 的实验结果可以看出，新算法在内存使用上的优势随着用户序列规模的增大而增加了。这是因为 PrefixSpan 算法需要花费大量的资源去产生中途的临时数据库，随着用户序列规模的增大，产生的临时数据库的规模也要增加。UTPMBD 算法在空间上的代价主要是存储 DG，而 DG 只需要用 N 个顶点和 K 条弧来记录有关频繁序列的所有信息。综合图 4-5、4-6 的实验结果，表明 UTPMBD 算法在时空性能上优于 PrefixSpan 算法。

4.4 本章小结

本章提出了一种基于有向图的用户频繁访问模式挖掘算法，首先对序列模式挖掘进行了概述并研究分析了其中有代表性的经典算法，然后介绍了进行用户频繁访问模式挖掘的数据预处理过程，并详细介绍了新算法在预处理得到的 Web 事务数据库上进行挖掘的改进思想和具体实现，最后在具有典型结构的 Web Log 数据上进行了用户频繁访问模式挖掘的试验，证明了算法的有效性。

结论与展望

Web 使用挖掘是 Web 挖掘领域中一个重要的研究方向，它对于发现用户浏览网站的行为规律，改善页面之间的超链接结构，提高整个 Web 系统的性能，以及在电子商务智能的应用等方面都具有十分重要的意义。

本文在分析研究 Web 使用挖掘的含义和相关技术的基础上，提出了两个优化改进的算法。总的来说，本文的工作主要有以下 3 个方面：

(1) 首先对 Web 挖掘的基本理论知识和分类进行了总体研究，重点分析了 Web 使用挖掘的基本思想和经典算法。

(2) 在分析关联规则经典算法 Apriori 的基础上，提出一种基于事务矩阵的关联规则挖掘算法，通过将事务数据库映射为一个事务矩阵，对事务矩阵进行操作以得到所有的频繁项目集。理论分析和实验证明了改进后的算法在性能上的优越性。

(3) 提出一种基于有向图的用户频繁访问模式挖掘算法，通过对 Web 事务数据库进行一次扫描，将所有页面之间的序列信息记录在有向图中并从中挖掘所有的用户频繁访问模式。并在具有典型结构的 Web Log 数据上进行了用户频繁访问模式挖掘的试验。

经分析研究表明，以上的技术和方法不仅对发现用户访问 Web 站点的行为规律是行之有效的，而且对研究 Web 站点结构的管理及优化也有一定的参考价值。

Web 使用挖掘是一个相对崭新的研究领域，对它的研究工作涉及多方面的理论、方法和技术，本论文对其中一些关键问题进行了有益的研究和探讨，但还有许多问题需要进一步的研究。

(1) 数据的预处理是 Web 使用挖掘技术的重要步骤，在今后的工作中需要继续研究用户识别、会话识别和事务识别等预处理步骤的技术和算法，进一步完善数据预处理的过程，提高日志预处理的质量和效率。

(2) 本文提出的所有算法，都只将研究重点放在了提高算法的效率上。今后的研究重点，应当是在进一步提高算法效率的基础上，重点挖掘对用户更有意义的频繁模式。

(3) 将本文提出或改进的算法与现有的一些通用的 Web 挖掘技术集成起来，应用于实际的 Web 挖掘系统的设计和实现，把理论知识转化为现实的生产力，高质高效地挖掘出对系统的使用者有价值的模式或知识。

致 谢

本论文是在我的导师陶宏才副教授的悉心指导下完成的。本论文从选题到最后完成，期间倾注了陶老师大量的汗水和心血。陶老师用孜孜以求的治学精神和学术上的敏锐洞察力教会了我进行科学研究的方法和途径，在此我要向陶老师表示最衷心的感谢！同时陶老师治学态度严谨、对工作极端负责、对学生关怀备至，这些都将使我终生受益！

要感谢实验室的师兄弟和师姐妹们，各位的存在，使实验室充满了活力，也使我在一个愉快的学习环境中完成本论文。同时，信息科学与技术学院的各位老师和同学也给了我大力的支持，在此表示感谢。

最后，要感谢家人在我求学生涯中所给予的物质和精神上的无私支持，使我安心致力于学习与研究，正是因为他们对我无限的支持，让我始终充满了前进的动力！

参考文献

[1] 陈安, 陈宁, 周龙骧等 编著. 数据挖掘技术及应用[M]. 北京：科学出版社, 2006

[2] 第 19 次中国互联网络发展状况统计报告 [EB/OL]. http://www.cnnic.net.cn/html/Dir/2007/01/22/4395.htm

[3] 高毅龙. Web 服务器访问日志的保存方法及其实现[J]. 计算机工程, 1999, 25(9): 36-39

[4] Jiawei Han，Micheline Kamber 编著. 范明，孟小峰 等译. 数据挖掘概念与技术[M]. 北京：机械工业出版社, 2001

[5] 朱明 编著. 数据挖掘[M]. 合肥：中国科学技术大学出版社, 2002

[6] Etzioni O. The Word Wide Web: Quagmire or gold mine [J]. Communications of ACM, 1996, 39(11): 65 – 68

[7] M.S.Chen, J.S.Pard, P. S.Yu. Efficent Data Mining for Path Traversal Patterns in a Web environment [C] // Proc. of the 16th IEEE intern'l Conf. On DistrIbuted Computing Systems, May 27-30,1996: 385-39

[8] D. Gunopulos, H. Mannila, R. Khardon and H. Toivone. Data mining, hypergragh transversals, and machine learning [C] // Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART symposium on Priciples of database systems,1997: 209-216

[9] Tak Yan, Mattew Jacohsen. Hector Garcia-Molina and Umeshwar Dayal. From User Access Patterns to Dynamic Hypertext Linking [C] // In Proceedings on the 5th International World Wide Web Conference.Paris, France, 1996

[10] Mohammed Javeed Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences [J]. Machine Learning, 2001,42(1/2): 31-60

[11] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation [C] // In Workshop on Research Issues in Data Engineering, Birmingham, England, 1997

[12] A. Zarkesh, J. Adibi, C. Shahabi, R. Sadri, and V. Shah. Analysis and design of server informative www-sites [C] // In Sixth International Conference on Information and Knowledge Management, Las Vegas, Nevada, 1997

[13] M.S.Chen，J.S.Mark. Efficient Data Mining for Path Traversal Pattern [C] // In Proceedings of IEEE INFOCOM '98，pp.209-221，April 1998

[14] A.Nanopoulos. Finding Generalized Path Patterns for Web Log Data Mining [C] // In Proceedings of the best-buropean Conference on Advances in a Databases and Information Systems，pp.215-228，May 2000

[15] O.R.Zaiane，J.Han. Discovering Web access patterns and trends by applying OLAP and data mining technology on Web logs [C] // In Proc of Advances in Digital Libraries Conf. Santa Barbara, CA., 1998

[16] 宋擒豹, 沈均毅. Web 日志挖掘的高效多能挖掘算法[J]. 计算机研究与发展, 2001,38(3):328-333

[17] 陆丽娜, 魏恒义, 杨怡玲等. Web 日志挖掘中的序列模式识别[J]. 小型微型计算机系统, 2000,21(5):481-483

[18] 高飞，谢维信. 互联网上的数据挖掘[J]. 计算机科学, 2001,28(5):81-84

[19] J.Pei, J.Han, B.Mortazavi-Asl, and H. Zhu. Mining Access Pattern Efficiently from Web Logs [C] // Proc Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Kyoto, Japan, 2000: 396-407

[20] 王继成，潘金贵，张福炎. Web 文本挖掘技术研究[J]. 计算机研究与发展，2000,37(5): 513-520

[21] 王实，高文，段立娟. Internet 上的文本数据挖掘[J]. 计算机科学，2000,27(4): 32-36

[22] Jonathan B,Ronny K. Tutorial on E-commerce and Clickstream Mining [C] // Fisrt SIAM International Conference on Data Mining, April 2001

[23] D. Arotariter, S. Mitra. Web Mining: A Survey in the Fuzzy Framwork [J]. Fuzzy Sets and Systems, 2004,148(1): 5-19

[24] 杨怡玲. 基于 Web 日志挖掘技术的智能 Web 站点研究[D]. 博士论文. 上海交通大学, 2002

[25] Osmar Rachid Zaiane. Resource and Knowledge Discovery from the Internet and Multimedia Repositories. [EB/OL]. http://www.cs.aue.auc.dk/datamining/papers/osmarzaianephd.pdf

[26] 庄越挺, 潘云鹤, 吴飞. 网上多媒体信息分析与检索[M]. 北京：清华大学出版社, 2002

[27] Sergey Brin. Dynamic Itemset Counting and Implication Rules for Market Data [C] // Proceedigns ACM SIGMOD International Conference on Management of Data, Tucson, USA, ACM Press, 1997: p 255-264

[28] J. Kleinberg. Authoritative Sources in a Hyperlinked Environment [J]. Journal of the ACM,1997,46(5):604-632

[29] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, Pang-Ning Tan. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Datas [J]. SIGKDD Explorations, 2000,1(2): 12-23

[30] T.Joachims, D.Freitag, T.Mitchell. WebWatcher: A Tour Guide for the World Wide Web [C] // Proceesdings of IJCA197, August 1997

[31] 付国瑜. 基于 Web 日志的数据挖掘研究[D]. 硕士论文. 重庆大学, 2007

[32] Logging Control In W3C httpd [EB/OL]. http://www.w3.org/Daemon/User/ConFig/Logging.html, 1995

[33] 张娥, 郑斐峰, 冯耕中. Web 日志数据挖掘的数据预处理方法研究[J]. 计算机应用研究, 2004,(2): 58-60

[34] 郭秀娟. 数据挖掘方法综述[J]. 吉林建筑工程学院学报, 2004,21(1): 49-53

[35] 夏庆, 马元元, 孙志辉. 路径遍历模式挖掘方法的改进[J]. 兰州大学学报(自然科学版), 1999,8(35): 370-371

[36] 陈 健, 印 鉴. Web 使用挖掘技术研究综述[J]. 计算机工程, 2005,31(9): 4-6

[37] Rakesh Agrawal, Tomasz Imielinski, Arun Swami. Mining Association Rules Between Sets of Items in Large Databases [C] // ACM SIGMOD,1993: 207-216

[38] Rakesh Agrawal, Ramakrishnan Srikant. Fast Algorithm for Mining Association Rules [C] // Proceedings of 20th Int. Conf. Very Large Data Bases(VLDB), Morgan Kaufmann Press,1994: 487-499

[39] 蔡伟杰, 张晓辉, 朱建秋等. 关联规则挖掘综述[J]. 计算机工程, 2001,27(5): 31-33, 49

[40] 李超, 余昭平. 基于矩阵的 Apriori 算法改进[J]. 计算机工程, 2006,32(23):

68-69

[41] 曾万聃, 周绪波, 戴勃等. 关联规则挖掘的矩阵算法[J]. 计算机工程, 2006,32(2): 45-47

[42] UCI Machine Learning Repository: Mushroom Data Set [EB/OL]. http://archive.ics.uci.edu/ml/datasets/Mushroom

[43] Quest Synthetic Data Generation Code [EB/OL]. http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html

[44] Agrawal R, Srikant R. Mining sequential pattems [C] // Proeeedings of the 11th Intemational Conference on Data Engineering, 1995: 3-14

[45] 何炎祥, 孔维强, 向剑文等. WebLog 访问序列模式挖掘[J]. 计算机工程与应用, 2003,(27):206-209

[46] R.Agrawal, R.Srikant. Mining Sequential Patterns:Generalizations and Performance Improvments [C] // Proc, 5th Int, Conf, Extending Database Technology(EDBT), Avignon,France, March.1996: 3-17

[47] Jiawei Han, Jian Pei. FreeSpan:Frequent Pattern-Projected Sequential Pattern Mining [C] // Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining(KDD'00), Boston, MA:ACM Press, Aug. 2000: 355-359

[48] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach [J]. IEEE Transactions on Knowledge and Data Engineering, 2004,11(16): 1424-1440

[49] N.Jain, E.Han, J.Srivastana. Web Mining: pattern discovery from World Wide Web trasactions [R]. Technical Report TR96-050, Department of Computer Science, University of Minnesota, http://maya.cs.depaul.edu~mobasherpaperswebminer-tr96.ps, 1996

[50] M.S.Chen, J.S.Park, P.S.Yu. Data Mining for Path Traversal Patterns in a Web Enviroment [C] // Proceeding of the 16th International Conference on Distributed Computing Systems, pp.385-392, July 1996

[51] 姚洪波, 杨炳儒. Web 日志挖掘数据预处理过程技术研究[J]. 微计算机信息(管控一体化), 2006, 22 (6-3): 234-236

[52] 战立强. 频繁模式挖掘算法研究[D]. 博士论文. 哈尔滨工程大学, 2007

[53] 胡作霆, 董兰芳, 王洵. 图的数据挖掘算法研究[J]. 计算机工程, 2006,3(32): 76-78

[54] D.W.Cheung, B.Kao, J.W.Lee. Discovering User Access Patterns on the World-Wide-Web [C] // Proc. First Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD-97), Singapore, 1997: 303-316

[55] 王利. Web 使用挖掘方法及其在个性化学习系统中的应用研究[D]. 硕士论文. 苏州大学, 2006

[56] Takeehiro Nakayama, Hiroki Kato. Discovering the Gap between Web Site Disigners' Expectations and User's Behavior [J]. Computer Networks, pp.811-822, June 2000

攻读硕士学位期间发表的论文

[1] 冯贺，陶宏才. 一种基于事务矩阵的关联规则挖掘算法[J]. 电脑学习，拟于 2008 年第 6 期发表