面向分面导航的层次概念格模型及挖掘算法

何 超[1),2)] 程学旗[1)] 郭嘉丰[1)]

[1)](中国科学院计算技术研究所 北京 100190)

[2)](中国科学院研究生院 北京 100190)

摘 要 分面导航利用动态多维分类目录组织查询结果,从而有效减轻数据库资源定位过程中的信息过载.现有的分面导航限制用户每次增删一个查询关键字,无法满足对具有丰富语义的导航操作的需求.另一方面,高效的动态目录生成算法的缺乏阻碍了分面导航在大规模数据中的应用.该文提出了层次概念格,对分面导航中不同浏览状态之间的关系进行建模.基于该层次概念格模型,该文设计了若干新的导航操作以支持用户在不同浏览状态之间更灵活地跳转,从而更有效地进行知识发现.为获取该层次概念格以支持导航的灵活性和实时性,该文提出了层次概念格的高效挖掘和索引算法 L-Miner. L-Miner 以深度优先方式挖掘所有节点,每得到一个新节点,就更新已挖掘节点之间的边.通过对底层格节点的倒排索引,L-Minder 可以高效地进行边更新.实验结果表明:L-Miner 的速度远快于现有算法,而其构建的索引结构的存储代价更低.

关键词 分面导航;概念格;频繁闭项集挖掘;数据挖掘;探索式信息检索

中图法分类号 TP391 **DOI 号**: 10.3724/SP.J.1016.2011.01589

Mining Hierarchical Concept Lattice for Faceted Navigation

HE Chao[1),2)] CHENG Xue-Qi[1)] GUO Jia-Feng[1)]

[1)](*Institute of Computing Technology, Chinese Academic of Sciences, Beijing* 100190)

[2)](*Graduate University of Chinese Academic of Sciences, Beijing* 100190)

Abstract Faceted navigation is effective in reducing information overload in the process of resource identification, by organizing query result into dynamic multi-dimensional categories. Existing approaches allow users to add or delete only one query keyword at each step, which cannot meet the demands for semantic-rich navigation operations. Moreover, the lack of efficient algorithms for generating dynamic categories makes faceted navigation non-scalable to large datasets. This paper proposes a hierarchical concept lattice for modeling the relationship between different navigation states. Then, a series of navigation operations are proposed to support more flexible transitions between navigation states and hence achieve more effective knowledge discovery. To guarantee real-time response, this paper also devises an efficient algorithm L-Miner for mining and indexing the hierarchical concept lattice. L-Miner discovers all the nodes in a depth-first way and updates the edges between all the generated nodes each time a new node is detected. Empirical studies demonstrate that L-Miner is much faster than existing approaches, utilizing a much smaller indexing structure.

Keywords faceted navigation; concept lattice; frequent close itemset mining; data mining; exploratory search

---

收稿日期:2010-11-17;最终修改稿收到日期:2011-06-14.本课题得到“八六三”高技术研究发展计划项目基金(2010AA012500)和国家自然科学基金(60933005,61003166)资助.**何 超**,男,1982 年生,博士研究生,主要研究兴趣为数据挖掘、探索式信息检索和分面搜索. E-mail: hechao@software.ict.ac.cn. **程学旗**,男,1971 年生,研究员,博士生导师,主要研究领域为信息检索、社会计算和分布式计算. **郭嘉丰**,男,1980 年生,博士,主要研究兴趣为 Web 搜索和挖掘、机器学习和社会网络.

1 引 言

互联网的快速发展使得人们可以在线访问大量数据库,如电子商务网站和数字图书馆.这些数据库存储了海量资源,且规模不断扩大.由于传统搜索引擎难以索引这些资源,人们称其为"暗网"(Deep Web).据统计,暗网比浅层网络大若干量级[1].为增强可用性,暗网站点通常提供基于关键字的搜索.渐增式/探索式搜索是常见的用户搜索行为,即以少量关键字作为初始查询,然后通过扩展查询不断细化查询结果,直至目标资源出现.用户进行渐增式搜索的主要原因有两点:(1)用户通过探索查询结果来不断明确自己的查询意图;(2)利用查询结果中出现的关键字,用户可以更准确地表达查询意图.在渐增式搜索过程中,用户面临的主要挑战是大量的查询结果导致信息过载,从而难以有效扩展查询.

分面导航通过查询结果分类来有效减轻暗网关键字搜索中的信息过载[2],并日益成为电子商务网站(如淘宝和 Amazon)和数字图书馆(如 ACM Digital Library)的主要信息检索方式.分面(facet)指对资源进行分类的特定维度,表示为类别层次结构.分面导航同时从多个分面对查询结果进行分类.例如,伯克利大学的 Flamenco 项目[3]从"性别,出生地,国家,奖项,年份"五个分面对诺贝尔奖得主进行分类.与传统的单层次目录树相比,分面目录的分类语义更加规范和清晰,且易于扩展.

分面导航通过交互式检索来进一步减轻信息过载.分面中类别是动态变化的,只有出现在查询结果中的类别才显示给用户(包括该类别对应查询结果中的资源个数).相应地,用户选择推荐的分面类别作为关键字进行检索或细化查询.例如,当用户在 Flamenco 中查询"经济学奖",因为所有 55 位获此奖者均为男性,所以性别分面中只显示"男性(55)".与静态目录相比,分面导航不仅降低了用户需要查看的类别数目,而且能有效避免查询结果为空.

现有的分面导航存在以下不足.首先,用户无法得到当前查询结果的分类语义范围,也无法发现推荐的分面类别(及其组合)对应的细化结果之间可能存在的包含关系.这限制了用户对查询结果的进一步探索,同时制约了查询结果过滤的有效性.其次,用户每次仅能增删一个查询关键字,或对一个已有查询关键字进行泛化或细化,无法执行语义更丰富的跳转.再次,用户无法对多个历史查询进行有效的分析.

针对上述问题,本文提出了面向分面导航的层次概念格模型,来实现具有丰富语义的分面导航操作.若将分面导航过程中查询及其结果作为浏览状态,层次概念格是对浏览状态空间的一种本体描述.具体而言,层次概念格是无环有向图,格中节点对应所有具有相同查询结果的浏览状态,节点之间的有向边表示它们对应的查询结果之间存在极小包含关系.因此,层次概念格是一张包含所有浏览状态的地图,分面导航就是通过动态分面目录来引导用户在不同浏览状态之间跳转.

为获取该层次概念格以支持导航的灵活性和实时性,本文进一步提出了格挖掘算法 L-Miner 来生成并索引格.L-Miner 采用基于深度优先的格增量挖掘框架.该框架通过自顶向下且深度优先方式遍历生成格中所有节点,并通过动态维护当前已生成节点的直接后继来同步生成边.L-Miner 利用倒排表索引少量已生成节点,并结合格的性质,来高效地完成上述框架下的两个关键操作,冗余检测和边更新.当格生成后,该倒排索引还可以支持高效的格导航算法.基于大量公开测试数据集的实验结果表明,L-Miner 的格挖掘速度远快于现有算法,而索引却小得多.

本文在第 2 节给出关于分面导航、形式概念格和格挖掘算法方面的相关工作;在第 3 节中,我们提出面向分面导航的层次概念格模型,以及基于层次概念格的导航操作;在第 4 节中,我们提出一种层次概念格的挖掘算法 L-Miner,并在第 5 节中对 L-Miner的性能进行实验分析;最后在第 6 节中对全文进行总结,并给出后继研究计划.

2 相关工作

2.1 分面导航

分类目录是帮助用户探索数据的有效方式.分面导航是一种新的基于分类目录浏览数据的方式.与传统方式相比,分面导航具有以下两个主要特点:(1)分类目录结构呈森林状,具有多维分类语义;(2)目录结构随查询结果动态变化,只保留出现在查询结果中的类别.

传统的目录结果是树状的,其类别通常涉及数据的不同特征,例如,对于音乐数据,目录结构中可能同时出现关于歌手和年份的类别.分面目录基于属性特征对传统目录进行规范化分解,其结构是森

林状的，其中每棵树代表基于一个属性的分类目录，称为一个分面. 例如，歌手分面和年份分面表示为两棵树. 与传统分类目录相比，分面目录的分类语义更清晰，且易于维护.

分面导航的另一个特点是目录结构随查询动态变化. 在传统导航式信息检索中，分类目录是静态的，可能包含成千上万的类别，尽管出现在查询结果中的类别可能很少. 这不仅加重了用户浏览目录的代价，而且用户点击某个类别进行查询结果过滤时，经常遇到空结果，从而降低了导航的有效性.

目前，分面导航不仅被广泛应用于电子商务网站和数字图书馆，而且逐渐成为研究热点. 目前的研究主要集中在两方面. 一方面是推荐给用户更有针对性的动态目录. 早期的动态目录显示所有对应非空细化结果的类别，却仍面临类别数目过多的问题. 文献[4-5]在生成动态目录前会向用户提问，根据回答推荐类别. 在文献[2,6-7]中，动态目录中不同类别对应的细化结果之间的重合度较小，该推荐策略可以极小化期望的用户代价，包括查看的类别数以及鼠标点击次数. 文献[8]生成个性化的动态目录. 文献[9-11]将分面导航和在线联机分析处理相结合. 文献[12]自动给出对应当前搜索结果的最小长度类别集，从而提高浏览效率.

另一方面是自动生成分面目录，并通过构建类别之间的层次关系来降低用户的浏览代价. 文献[13-14]分别利用监督学习和无监督学习从文本数据中构建分面目录. 文献[15] 利用无监督学习自动构建分面目录来组织用户提供的关键字.

其它的相关研究还包括，文献[16-17]利用分面导航帮助用户浏览半结构化数据；文献[18]提出一种能适应不同屏幕大小的分面导航方式. 代表性的分面导航系统有 Flamenco[3]、SIMILE① 和 Haystack②.

现有的分面导航操作只允许用户每次增删一个查询关键字，无法满足用户对具有丰富语义的导航操作的需求. 本文中，我们将设计若干新的导航操作以支持用户在不同浏览状态之间更灵活地跳转，从而更有效地进行知识发现.

2.2　形式概念格

形式概念格(Formal Concept Lattice)[19] 是根据对象及其属性自动构建的一个本体. 形式化的表达如下：给定三元组$\langle A,B,f\rangle$，其中 A 是对象集，B 是属性集，函数 $f:A\times B\to\{0,1\}$，其中 $f(a,b)=1$ 当且仅当对象 a 具有属性 b. 对于对象集 $O\subseteq A$ 和属性集 $X\subseteq B$，(O,X) 是一个概念当且仅当：① O 是具有 X 中所有属性的所有对象构成的集合；且② $\forall Y\supset X$，$\exists o\in O$，o 不具有 Y 中所有属性. 此时 O 称为该概念的外延，X 称为该概念的内涵. 所有概念根据外延的集合包含关系构成格，即对任意多个概念$(O_1,X_1),(O_2,X_2),\cdots,(O_n,X_n)$，存在：① 最小概念$(O_m,X_m)$，满足 $\forall i\in[1,n]$，有 $O_m\supseteq O_i$；和② 最大概念(O_M,X_M)，满足 $\forall j\in[1,n]$，有 $O_M\subseteq O_j$. 传统概念格不考虑属性间存在层次关系.

本文通过扩展传统的形式概念格，提出层次概念格模型对分面导航的用户浏览状态空间进行建模. 基于层次概念格，我们可以高效地完成所提出的具有丰富语义的导航操作.

2.3　频繁闭项集挖掘

频繁闭项集挖掘[20-31] 是数据挖掘中的核心问题之一，是众多数据挖掘任务的基础. 在此，我们可以利用频繁闭项集挖掘算法得到层次概念格中的所有概念(概念和频繁闭项集一一对应)，然后通过两两比较概念来计算层次概念格的边. 由于层次概念格中节点数目通常高达几十万，这种两步计算方式的代价太高.

CHARM-L[31] 是最近提出的频繁闭项集格挖掘算法. 它采用单步方式计算格，每生成一个新节点，就更有已有节点之间的边. 因此，节点和边同时生成. 与两步方式比，单步方式的效率更高. CHARM-L 通过倒排表索引已挖掘的频繁闭项集来高效地实现频繁闭项集格挖掘中的两个关键操作，冗余检测和边更新.

我们提出的 L-Miner 算法只索引了部分频繁闭项集，但结合层次概念格的性质可以更高效地完成上述两个关键操作. 因此，L-Miner 以更小的存储代价实现了更快的格挖掘.

3　面向分面导航的层次概念格模型

分面导航是基于分面目录对查询结果分类并引导用户探索查询结果的有效方式. 本节利用层次概念格对用户浏览状态空间进行建模，并基于层次概念格提出具有丰富语义的导航操作，帮助用户在导航过程中进行知识发现.

3.1　用户浏览状态空间

在传统分面导航过程中，给定分面目录和关系

① http://simile.mit.edu
② http://haystack.csail.mit.edu

数据库表,用户以目录中已有类别为关键字进行查询.分面导航系统将返回属于指定类别集的所有资源(即数据库表中的元组).此外,分面导航系统根据查询结果过滤分面目录,仅保留出现在查询结果中的类别,从而引导用户进一步细化查询.

传统分类目录通常是一棵树,树中节点代表分类类别,其孩子节点代表对属于该节点的资源(元组)的细分.值得注意的是,祖先节点和后代节点不一定存在子类型关系.例如,关于音乐专辑的传统分类目录中,代表2010年的类别"2010"的孩子节点可能为代表歌手的类别"王菲"、"刘德华"等.

分面目录是一个多维分类目录,通常表示为森林.森林中每棵树代表一个分类维度.同一棵树上的祖先节点和后代节点之间存在超类型关系.在上面的音乐专辑例子中,代表年份的所有类别在同一棵树中,构成年份分面,而代表歌手的所有类别在另一棵树中,构成歌手分面.通过这种多维分类结构,分面目录的分类语义更加清晰,且易于维护.

我们将分面目录形式化定义为关于类别的偏序集,类别之间的偏序关系和它们在分面目录中的层次关系一一对应.换言之,分面目录是该类别偏序集对应的哈斯图①.

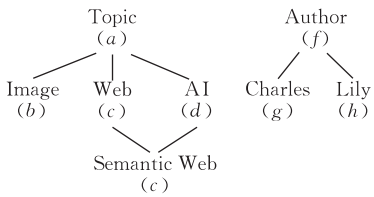
定义 1. 令$\mathcal{C}$是所有类别构成的集合.$<_c$是$\mathcal{C}$上的偏序关系,$\forall u,v\in\mathcal{C}$,若$v$是$u$的超类,则$u<_c v$.$u\leq_c v$表示$u<_c v$或$u=v$.称偏序集$(\mathcal{C},\leq_c)$为分面目录.

例如,图1(a)中关于论文的分面目录包含两个分面,分别代表论文主题"Topic"和论文作者"Author"两个分类维度.该分面目录表示为偏序集$(\{Topic\ (a),Image\ (b),Web\ (c),AI\ (d),Semantic\ Web\ (e),Author\ (f),Charles\ (g),Lily\ (h)\},<_c)$,其类别之间的所有偏序关系为$\{b<_c a,c<_c a,d<_c a,e<_c a,e<_c c,e<_c d,g<_c f,h<_c f\}$.显然,不同分面上的任意两个类别$u$和$v$互相独立,即$u\neg<_c v$且$v\neg<_c u$,其中"¬"表示取非.

给定类别集$X\subseteq\mathcal{C}$,若其同时包含存在子类型关系的两个类别$u,v(u<_c v)$,则$X-\{v\}$表示的分类语义和X一致.因此,我们记$concise(X)=\{u:u\in X\wedge\neg\exists v\in X(v<_c u)\}$,表示$X$的最简描述;记$verbose(X)=\{u:u\in\mathcal{C}\wedge\exists v\in X(v\leq_c u)\}$,表示$X$的最繁描述.

定义 2. $\forall X,Y\subseteq\mathcal{C}$,若$concise(X)=concise(Y)$,则$X$与$Y$的分类语义相同.若$X=concise(X)$,称$X$是不可约类别集.

除分面目录外,分面导航还假定每个资源r(元组)所属类别集均已知(通常根据资源的属性值可得),表示为$categories(r)=\{u:u\in\mathcal{C}\wedge r\text{ 属于类别 }u\}$.显然,$categories(r)$是最繁描述,事实上,我们只需指定其最简描述$concise(categories(r))$即可.例如,在图1(b)中,论文#1的所属类别集$categories(\#1)=\{a,b,c,f,g,h\}$,其最简描述为$\{b,c,g,h\}$.



(a) 分面目录

论文编号	论文所属类别
1	b,c,g,h
2	e,h
3	b,c,g,h
4	b,e,h
5	b,e,g,h
6	e,g

(b)

图1 论文数据库

给定分面目录$(\mathcal{C},<_c)$和资源所属类别集,我们将分面导航过程中用户的浏览状态定义为由查询及其结果构成的二元组.本文中,我们假定查询是不可约类别集.查询结果为同时属于每个查询关键字(类别)的所有资源.令$\mathcal{T}$为所有资源构成的集合,记查询Q的资源集合为$resources(Q)=\{r:r\in\mathcal{T}\wedge Q\subseteq categories(r)\}$.

定义 3. 用户的浏览状态表示二元组$\langle Q,resources(Q)\rangle$,其中$Q$为查询.设$\langle Q_1,S_1\rangle\rangle$和$\langle Q_2,S_2\rangle\rangle$是浏览状态,若$S_1=S_2$,则称$\langle Q_1,S_1\rangle\rangle$和$\langle Q_2,S_2\rangle\rangle$等价.

因为分面导航采用关键字匹配(类别匹配)的方式进行精确检索,所以不同查询对应的查询结果可能相同.例如,在图1中,查询$\{c,d\}$和$\{e\}$等价,因为它们的查询结果均为$\{2,4,5,6\}$.表1给出了查询结果包含至少两个资源的所有浏览状态的划分,记为$\mathcal{S}_{[2]}$.其中每一行为对应$\mathcal{S}_{[2]}$中的一个等价类.

① 图中每个节点表示集合S中的一个元素,节点的位置按它们在偏序中的次序从底向上排列:$\forall a,b\in S$,若$a<b(a\leq b\wedge a\neq b)$,则$a$排在$b$的下边;如果$a<b$,且不存在$c\in S$满足$a<c<b$,则在$a$和$b$之间连一条线.这样画出的图称为偏序集$(S,\leq)$的哈斯图.

表 1　划分浏览状态集 $\mathcal{S}$[2]

查询结果	查询
{1,2,3,4,5,6}	{},{a},{c},{f},{a,f},{c,f}
{1,2,3,4,5}	{h},{c,h},{a,h}
{2,4,5,6}	{e},{d},{c,d},{e,f},{d,f},{c,d,f}
{1,3,5,6}	{g},{c,g},{a,g}
{1,3,4,5}	{b},{b,c},{b,h},{b,f},{b,c,h},{b,c,f}
{2,4,5}	{e,h},{d,h},{c,d,h}
{1,3,5}	{g,h},{b,g},{a,g,h},{b,c,g},{b,g,h},{c,g,h},{b,c,g,h}
{4,5}	{b,e},{b,d},{b,c,d},{b,e,h},{b,d,h},{b,c,d,h}
{5,6}	{e,g},{d,g},{c,d,g}

3.2　层次概念格模型

在分面导航过程中，用户从某浏览状态出发，经过多步细化和泛化后，有可能又得到同样的查询结果，尽管其查询不同. 在现有的分面导航中，这种“原地转圈”难以被察觉，影响数据探索效果和搜索效率. 因此，我们希望描绘一张关于浏览状态的全局地图，防止用户“迷路”.

由表 1 可知，当数据库的分面目录较大且资源数目较多时，呈现给用户的无论是所有可能的查询结果还是它们对应的所有查询，都会导致庞大臃肿的“地图”，难以计算和浏览. 因此，我们希望提供一个压缩的、信息无损的、简洁的“地图”.

定义 4.　令 X 和 Y 均为查询，若 $\forall u\in Y,\exists v\in X$，满足 $v\leq_c u$，则称 X 蕴含 Y，记作 $X\leq_c Y$. 若 $X\leq_c Y$ 且 $X\neq Y$，则 $X<_c Y$.

不同查询之间可能存在分类语义蕴含关系. 给定浏览状态等价类，我们根据蕴含关系得到其对应的所有查询之间的偏序集. 该偏序集中的极大元素（称为极泛化查询）描述了对应查询结果的极简分类语义，而极小元素（称为极细化查询）描述了对应查询结果的极繁分类语义.

定义 5.　令 X 是查询. 若对任意查询 $Y(Y>_c X\rightarrow resources(Y)\supset resources(X))$，则称 X 是极泛化的. 若对任意查询 $Y(Y<_c X\rightarrow resources(Y)\subset resources(X))$，则称 X 是极细化的.

进一步，我们可以证明给定浏览状态等价类，尽管其极泛化查询可能有多个，但其极细化查询是唯一的. 也就是说，只存在唯一的极繁分类语义描述.

引理 1.　令 X 是极细化查询. 对任意查询 Y，若 $resources(X)\subseteq resources(Y)$，则 $X\leq_c Y$.

证明.　假设 $X\leq_c Y$ 不成立 $\forall u\in Y$，必有：(1) $\forall v\in X$，u 与 v 相互独立；或(2) $\exists v\in X$，$u<_c v$.

对情况(1)，$resources(X)\supseteq resources(X\cup\{u\})=resources(X)\cap resources(\{u\})\supseteq resources(X)\cap resources(Y)=resources(X)$. 因为 $resources(X)\subseteq resources(Y)$，所以 $resources(X)=resources(X\cup\{u\})$. 这与 X 是极细化类别集相矛盾，故(1)不成立.

对情况(2)，令 C 是 X 中所有大于 u 的类别构成的集合，则 $Z=X-C\cup\{u\}$ 是不可约类别集. 易知 $resources(C)\supseteq resources(\{u\})$. 故 $resources(X)=resources((X-C)\cup C)=resources(X-C)\cap resources(C)\supseteq resources(X-C)\cap resources(\{u\})=resources(Z)$，即 $resources(X)\supseteq resources(Z)$ 成立. 因为 $u\in Y$，故 $resources(\{u\})\supseteq resources(Y)$. 又 $resources(X)\subseteq resources(Y)$，故 $resources(\{u\})\supseteq resources(X)$. 显然，$resources(X-C)\supseteq resources(X)$. 故 $resources(Z)=resources(X-C\cup\{u\})=resources(X-C)\cap resources(\{u\})\supseteq resources(X)\cap resources(X)=resources(X)$，即 $resources(Z)\supseteq resources(X)$ 成立. 因此，$resources(Z)=resources(X)$. 因为 $Z<_c X$，这与 X 是极细化查询相矛盾，故(2)不成立.

综上所述，假设不成立，必有 $X\leq_c Y$.　　证毕.

定理 1.　查询结果和极细化查询一一对应. 查询结果对应一个或多个极泛化查询.

证明.　给定任意查询结果 S，令 $\mathcal{Q}$ 是由所有查询结果为 S 的查询构成的集合，即 $\mathcal{Q}=\{X:X$ 是查询 $\wedge resources(X)=S\}$. 显然 $\mathcal{Q}$ 非空. 根据定义，$\mathcal{Q}$ 中的极大元素为极泛化查询，极小元素为极细化查询，二者均存在. 令 X 和 Y 是 $\mathcal{Q}$ 中的极小元素，根据引理 1，因为 $resources(X)=resources(Y)$，所以 $X\leq_c Y$ 且 $Y\leq_c X$. 故 $X=Y$. 由此可知，$\mathcal{Q}$ 中极小元素仅有一个. 因此，查询结果和极细化查询一一对应，查询结果对应一个或多个极泛化查询.　　证毕.

因此，浏览状态等价类和极细化查询一一对应. 但是，如果仅呈现极细化查询，用户尚无法确定还有哪些查询对应同一查询结果. 在此，我们证明浏览状态等价类之间的格结构可以帮助实现该信息的重构.

引理 2.　令 Q_1 和 Q_2 分别是对应查询结果 S_1 和 S_2 的极细化查询. $Q_1<_c Q_2$ 当且仅当 $S_1\subset S_2$.

证明.　(⇒)因为 $Q_1<_c Q_2$，根据定义，$\forall u\in Q_2$，$\exists v\in C_1$，有 $v\leq_c u$ $\forall r\in resources(Q_1)$，令 $X=concise(categories(r))$. 根据定义，有 $X\leq_c\{v\}$，所以 $X\leq_c\{u\}$. 因此，$r\in resources(Q_2)$. 故 $S_1\subseteq S_2$. 根据定理 1，因为 $Q_1\neq Q_2$，所以 $S_1\neq S_2$. 因此，$S_1\subset S_2$ 成立.

(⇐)根据引理 1，因为 $S_1\subset S_2$，所以 $Q_1\leq_c Q_2$. 根据定理 1，$Q_1\neq Q_2$. 因此，$Q_1<_c Q_2$ 成立.　　证毕.

定理 2. 令$\mathcal{Q}$是查询结果包含至少δ个资源的所有极细化查询$\forall X_1, X_2, \cdots, X_k \in \mathcal{Q} \cup concise(\mathcal{C})\}$，$\exists Y, Z \in \mathcal{Q} \cup concise(\mathcal{C})$，其分别为$X_1, X_2, \cdots, X_k$的最小公共上界和最大公共下界.$\langle \mathcal{Q} \cup concise(\mathcal{C}), <_c \rangle$是完全有界格.

证明. 根据引理 2，$\mathcal{T}$的极细化查询是$X_1, X_2, \cdots, X_k$的公共上界.令$\{Y_1, Y_2, \cdots, Y_m\}$为$X_1, X_2, \cdots, X_k$的所有公共上界.令查询结果$S = resources(\bigcup_{i=1,\cdots,m} Y_i)$的极细化查询为$Y$，令$U = \bigcup_{j=1,\cdots,k} resources(X_j)$，则$S = \bigcap_{i=1,\cdots,m} resources(Y_i) \supseteq \bigcap_{i=1,\cdots,m} U = U$.故$Y \in \{Y_1, Y_2, \cdots, Y_m\}\ \forall i \in [1,m], S \subseteq resources(Y_i)$.根据引理 2，有$Y \leq_c Y_i$.所以$Y$是$X_1, X_2, \cdots, X_k$的最小公共上界.

令$S = resources(\bigcup_{i=1,\cdots,k} X_i)$.当$|S| \geq \delta$时，令$Z$是其极细化查询$\forall i \in [1,k], S \subseteq resources(X_i)$.根据引理 2 易知，$Z$是$X_1, X_2, \cdots, X_k$的公共下界.令$L$是$X_1, X_2, \cdots, X_k$的任意公共下界，即$\forall i \in [1,k]$，有$L \leq_c X_i$.故$resources(L) \subseteq resources(X_i)$.因此，$resources(L) \subseteq \bigcap_{i=1,\cdots,k} resources(X_i) = S$.由引理 2，$L \leq_c Z$.所以$Z$是$X_1, X_2, \cdots, X_k$的最大公共下界.当$|S| < \delta$时，易知$concise(\mathcal{C})$是$X_1, X_2, \cdots, X_k$的最大公共下界.综上所述，$\exists Z \in \mathcal{Q} \cup concise(\mathcal{C})$，$Z$为$X_1, X_2, \cdots, X_k$的最大公共下界.

因此，偏序集$\langle \mathcal{Q} \cup concise(\mathcal{C}), <_c \rangle$是完全格.令极细化查询$X$满足$resources(X) = \mathcal{T}$，显然$X \in \mathcal{Q}$.$X$和$concise(\mathcal{C})$分别是该格的最大上界和最小下界，故$\langle \mathcal{Q} \cup concise(\mathcal{C}), <_c \rangle$是完全有界格. 证毕.

根据定理 2，给定多个查询，存在最小的极细化查询，其查询结果同时包含每个给定查询的查询结果；若所有给定查询的查询结果的交集至少包含δ个资源，则存在最大的极细化查询，其查询结果同时被每个给定查询的查询结果包含.

定义 6. 我们称偏序集$\langle \mathcal{Q}, <_c \rangle$的哈斯图为层次概念格.

图 2 是对应表 1 的层次概念格.为便于理解，图中同时给出了每个极细化查询的查询结果.

现有的分面导航可利用层次概念格高效实现.给定查询(初始查询或经过细化或泛化操作形成的新查询)，首先定位该查询对应的格中节点(该节点唯一).然后，根据该节点的后继，可以知道哪些类别的细化结果非空及其细化结果的大小，从而生成对应当前浏览状态的动态分面目录.特别的，我们还能知道动态分面目录中所有类别的非空细化结果之间的包含/相等关系，从而进一步优化动态分面目录的

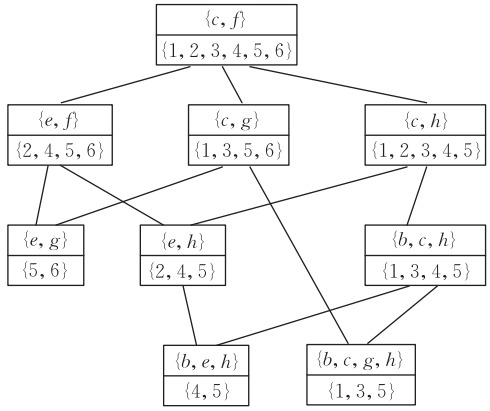


图 2 层次概念格($\delta=2$)

结构.在 3.3 节中，我们将阐述基于层次概念格的新的具有丰富语义的导航操作.

根据定理 3，基于层次概念格的边可以确定每个极细化查询对应的所有极泛化查询.因为查询结果对应的所有查询均介于对应的极细化查询和某个极泛化查询之间，所以基于层次概念格的边可以确定每个查询结果对应的所有查询.

定理 3. 令极细化查询X的所有前驱节点为$P_1, P_2, \cdots, P_k$.对任意查询$G \geq_c X$，G是$resources(X)$的极泛化查询当且仅当：(1) $\forall i \in [1,k], P_i \leq_c G$不成立；且，(2)对任意查询$Y >_c G$，$\exists i \in [1,k]$，使得$P_i \leq_c Y$.

证明. 令$S = resources(X)$.

($\Rightarrow$)假设$\exists i \in [1,k]$，$P_i \leq_c G$成立.根据$resources(P_i) \subseteq resources(G)$，再根据格定义$S \subset resources(P_i)$，故$S \subset resources(G)$.因为$G$是$S$的极泛化查询，所以$S = resources(G)$.上述结论互相矛盾，故假设不成立，条件(1)成立.对任意查询$Y >_c G$，根据极泛化查询的定义可知：$resources(G) \subset resources(Y)$.因为$S = resources(G)$，所以$S \subset resources(Y)$.根据格定义知，$\exists i \in [1,k]$，使得$P_i \leq_c Y$.

($\Leftarrow$)因为$G \geq_c X$，所以$S \subseteq resources(G)$.根据条件(1)，$\forall i \in [1,k], resources(P_i) \subseteq resources(G)$不成立.根据格定义知：$S = resources(G)$，所以$G$是$S$的查询.根据引理 2，对任意查询$Y >_c G$，$resources(Y) \supseteq resources(P_i) \supset S$.因此，$G$是$S$的极泛化查询. 证毕.

综上所述，层次概念格(即所有极细化查询及其格结构)是对所有浏览状态及其跳转关系的无损压缩.

3.3 基于层次概念格的分面导航操作及算法

层次概念格完整并简洁地表达了查询结果间的

包含关系.相应地,用户的浏览过程就是在格中不同节点之间进行跳转.分面导航通过动态分面目录来引导用户在不同浏览状态之间有效地跳转.现有的分面导航仅支持基于单关键字的细化或泛化操作,即用户选择动态目录中某类别细化当前查询结果,或通过删除一个查询关键字泛化当前查询结果.这种导航操作方式单一,只具备基本的知识发现能力.

基于层次概念格,本文设计了以下新的导航操作以支持用户在不同浏览状态之间更灵活地跳转,从而更有效地进行知识发现:

(1)显示分类语义范围 $semantics(Q)$:给定查询 Q,定位格中相应节点,显示其极细化查询和所有极泛化查询;

(2)指定比例进行细化 $refine(X,\delta)$ 或泛化 $expand(X,\delta)$:分别显示与当前节点 X 距离不超过 δ 的极远后代节点和极远祖先节点;

(3)历史查询分析 $join(X_1,X_2,\cdots,X_k)$ 和 $meet(X_1,X_2,\cdots,X_k)$:分别显示节点 $X_1,X_2,\cdots,X_k$ 的最近公共祖先和最近公共后代.

定义 7. 节点 X_1 和 X_2 的距离为 $dist(X_1,X_2)=1-|S_1\cap S_2|/|S_1\cup S_2|$,其中 $S_1=resources(X_1)$,$S_2=resources(X_2)$.

semantics 操作帮助用户定位给定查询对应的节点,该节点(即极细化查询)给出了对应查询结果的分类语义下界;根据定理 3,其分类语义的上界可通过该节点在层次概念格中的直接前驱节点得到.

expand 和 *refine* 操作按比例细化和泛化查询结果,使得用户更易控制细化和泛化的粒度,逐步逼近潜在查询目的.另一方面,在传统分面导航中需要多步操作才能达到的浏览状态,通过上述操作可能只需一步,从而提高了信息查找效率.

join 和 *meet* 操作可以用来分析多个历史查询之间的关系.*meet* 操作帮助用户了解多个查询结果的交集的分类语义,而 *join* 操作分析它们同属什么类别.

基于上述操作,用户的查询日志可以形式化地表示为格上的节点跳转轨迹,并可通过适当的可视化技术呈现.接下来,我们基于层次概念格给出上述导航算法的实现.

3.3.1　*semantics* 操作算法

任意查询(例如查询 Q)对应层次概念格中唯一节点(例如节点 X),该节点满足 $X\leq_c Q$ 且 $resources(X)=resources(Q)$.要定位查询 Q 对应的层次概念格中节点,较为直接的做法是,对层次概念格进行自顶向下或自底向上的洪泛式搜索.在搜索过程中,当前节点 X 满足要求,当且仅当 $X\leq_c Q$,且其任意前驱节点和 Q 相互独立.考虑到层次概念格节点规模可能很大(如包含几十万个节点),上述直接定位方法的效率较低,难以满足导航实时性的需求.因此,本文提出基于极小极细化查询的倒排索引来提高定位操作的效率.

极小极细化查询对应层次概念格中最底层节点.图 2 中所有极小极细化查询为 $\{\{e,g\},\{b,e,h\},\{b,c,g,h\}\}$.

定义 8.　令 $\langle\mathcal{Q},<_c\rangle$ 是层次概念格且 $X\in\mathcal{Q}$,若 $\neg\exists Y\in\mathcal{Q}(Y<_c X)$,则称 X 是极小极细化查询.

表 2　层次概念格的倒排索引表(令节点 $\{e,g\}$,$\{b,e,h\}$,$\{b,c,g,h\}$ 的编号分别为 n_1,n_2 和 n_3)

类别	节点编号
a	n_1,n_2,n_3
b	n_2,n_3
c	n_1,n_2,n_3
d	n_1,n_2
e	n_1,n_2
f	n_1,n_2,n_3
g	n_1,n_3
h	n_2,n_3

我们按如下方式构建层次概念格的倒排索引表.每个列表对应分面目录中一个类别.对每个极小极细化查询 X,将 X 所在节点的标识符添加到 $verbose(X)$ 中每个类别对应的列表中.表 2 给出了对应图 2 中层次概念格的倒排索引结构.

算法 1.　*locate*(Q).

输入:查询 Q

输出:若 Q 频繁,返回层次概念格中节点 $close(Q)$;否则,返回空值.

```
Set minnodes←{n: ∀u∈verbose(Q)(节点 n 在 u 的索引列表中)};//所有分类语义蕴含 Q 的节点
if minnodes is empty
  return null;
令 X 为 minnodes 中任意节点;
while preds(X) is not empty
  bool bUp←false;//是否继续向上访问
  foreach Y∈preds(X)
    if Y≤cQ
      X←Y;
      bUp←true;
      break;
  if not bUp
    break;
return X;
```

基于上述倒排索引,算法 1 可以快速定位对应

查询 Q 的格节点，从而得到其查询结果的最细分类语义描述. 首先，找出对应查询结果被 $resources(Q)$ 包含的所有极小极细化查询 $minnodes$. 当 $minnodes$ 为空时，查询 Q 对应的资源数目小于阈值 δ. 否则，从 $minnodes$ 中任意节点 X 出发沿格中边向上攀爬，每次从 X 的前驱节点集 $preds(X)$ 中选取任意不小于 Q 的节点作为当前节点，直至不存在满足该条件的节点. 为减少攀爬步骤，我们可以在每一步中选取对应查询结果最大的那个节点.

根据定理 3，当前节点的极泛化查询可通过该节点及其前驱节点的极细化查询得到. 依次考察 $verbose(Q)$ 的大小为 $|verbose(Q)|-1$ 到 1 的每个子集 G. 如果 $concise(G)$ 不大于等于任意 $P_i(i\in[1,k])$，将其放入候选集；否则，G 的任意子集都不再考察. 最后，候选集中极大元素即为当前节点的极泛化查询.

3.3.2 *expand* 和 *refine* 操作算法

定义 7 给出的节点距离反映了其查询结果之间的重合度. 易知该距离与 Jaccard 距离一致，均满足三角不等式，即任意给定节点 X_1, X_2, X_3，有 $|dist(X_1,X_2)-dist(X_2,X_3)|\leq dist(X_1,X_3)$. 因此，我们可以采用如下方式进行 *expand* 和 *refine* 操作.

对于 $expand(X,\delta)$ 操作，从 X 出发以洪泛方式访问其祖先节点，直至所达节点的所有前驱节点与 X 的距离均超过 δ. 对于 $refine(X,\delta)$ 操作，从 X 出发以洪泛方式访问其后代节点，直至所达节点的所有后代节点与 X 的距离均超过 δ.

3.3.3 *join* 和 *meet* 操作算法

join 和 *meet* 是格特有的操作，分别返回给定多个节点 $X_1, X_2, \cdots, X_k$ 的最小公共上界和最大公共下界.

如图 3 所示，假定已经得到节点 N_{ub}，满足 $\forall i\in[1,k]$，有 $X_i\leq_c N_{ub}$. 令 J 是按下述方法得到的节点：

1. 令 N_{ub} 为当前节点.

2. 判断是否存在当前节点的直接后继节点 Z，$\forall i\in[1,k]$，有 $X_i\leq_c Z$. 如果不存在，则返回当前节点并结束算法；如果存在，则将当前节点置为 Z，并返回步 2.

如图 3 所示，假定已经得到节点 N_{lb}，满足 $\forall i\in[1,k]$，有 $N_{lb}\leq_c X_i$. 令 M 是按下述方法得到的节点：

1. 令 N_{lb} 为当前节点.

2. 判断是否存在当前节点的直接前驱节点 Z，$\forall i\in[1,k]$，有 $Z\leq_c X_i$. 如果不存在，则返回当前节点并结束算法；如果存在，则将当前节点置为 Z，并返回步 2.

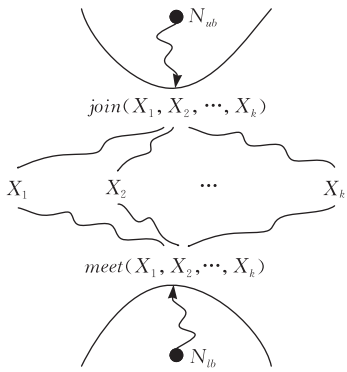


图 3 *join* 和 *meet* 的计算

易知，J 和 M 分别是给定节点 $X_1, X_2, \cdots, X_k$ 的最小公共上界和最大公共下界，即为 *join* 和 *meet* 操作的返回结果. 格的根节点是满足要求的 N_{ub}，N_{lb} 可以通过扫描所有极小极细化查询得到.

根据下述定理，*meet* 操作还可以通过算法 1 间接实现.

定理 4. $meet(X_1, X_2, \cdots, X_k)=locate(\bigcup_{i=1,\cdots,k} X_i)$.

证明. 令 $S_i=resources(X_i)$，$\forall i\in[1,k]$. 令 $U=\bigcup_{i=1,\cdots,k}X_i$，$S'=S_1\cap S_2\cap\cdots\cap S_k$. 易知 $S'=resources(U)$. 根据定理 1，S' 对应的极细化查询 Y 是 $X_1, X_2, \cdots, X_k$ 的公共下界. 假设 Y 不是 $X_1, X_2, \cdots, X_k$ 的最大公共下界，则存在查询结果 S''，$S'\subset S''$ 且 $\forall i\in[1,k]$，有 $S''\subseteq S_i$. 因此，$S'\subset S_1\cap S_2\cap\cdots\cap S_k=S'$. 故假设不成立，$Y$ 是 $X_1, X_2, \cdots, X_k$ 的最大公共下界. 因为 $resources(locate(\bigcup_{i=1,\cdots,k}X_i))=S'$，所以结论成立. 证毕.

4 L-Miner：层次概念格挖掘算法

当数据库较大时，在用户浏览过程中动态生成层次概念格的代价太高，如计算当前查询结果的极细化查询就需要扫描一遍数据库. 为满足实时性，需要预先生成格. 本文提出一种层次概念格挖掘算法 L-Miner，它采用一棵类别集枚举树以自顶向下且深度优先方式生成格节点，同时进行边更新.

4.1 基于深度优先的格增量挖掘框架

层次概念格的挖掘任务分为两部分，节点挖掘和边挖掘. 我们采用基于深度优先的格增量挖掘框架. 该框架通过深度优先遍历一棵类别集枚举树生成所有节点，该类别集枚举树中节点和格节点一一对应. 每当生成一个新节点，就更新已有节点之间的边. 因此，当所有节点都生成时，格中所有边也同时

得到.该框架下的两个关键操作,冗余检测和节点后继维护,将在 4.2 节和 4.3 节中分别叙述.

4.1.1　类别集枚举树

类别集枚举树通过枚举类别组合来生成查询结果至少包含 δ 个资源的所有极细化查询.树节点和格节点一一对应,因此可将类别集枚举树看成是层次概念格的自顶向下的深度优先遍历树.

每个树节点表示为三元组$\langle fci,supp,ref_seq\rangle$,$fci$ 是极细化查询,$supp$ 是支持度,(等于$|resources(fci)|$),ref_seq 是收缩对序列.每个收缩对表示为二元组$\langle ref_item,trans\rangle$,其中 $trans=resources(fci\cup\{ref\_item\})$ 且 $\delta\leq|trans|<resources(fci)$.

在类别集枚举树中,根节点对应整个资源集.对每个树节点,依次访问其所有收缩对,对非冗余的收缩对生成该树节点的孩子节点.在此过程中,后代树节点是通过不断收缩祖先节点对应的资源集得到的.

图 4 为对应图 2 中层次概念格的类别集枚举树.节点$\mathcal{N}_1\sim\mathcal{N}_9$的生成顺序是深度优先的,与其编号一致.根节点$\mathcal{N}_1$对应整个资源集$\{1,2,3,4,5,6\}$.因为所有类别中只有 a,c,f 对应所有资源,而 a 是 c 的超类,所以$\mathcal{N}_1.fci=concise(\{a,c,f\})=\{c,f\}$.其余类别 b,d,e,g,h 的查询结果分别为$\{1,3,4,5\}$,$\{2,4,5,6\}$,$\{2,4,5,6\}$,$\{1,3,5,6\}$,$\{1,2,3,4,5\}$,所包含的资源数均小于 6,故均为根节点的收缩对.收缩对列表中类别 b,d,g,h 分别对应$\mathcal{N}_1$的孩子$\mathcal{N}_2,\mathcal{N}_5,\mathcal{N}_8,\mathcal{N}_9$.由于类别 e 对应的细化结果和 d 的相同,所以是冗余的、不生成$\mathcal{N}_1$的孩子.

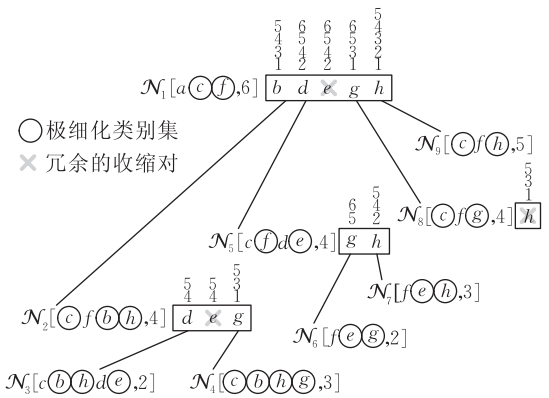


图 4　类别集枚举树

令当前节点为$\mathcal{N}_i$,以其为根的子树按如下方式生成.依次对$\mathcal{N}_i$的收缩对序列中每个收缩对 R_j 进行冗余检测,即是否存在已生成的树节点$\mathcal{N}_t(t<i)$满足 $resources(\mathcal{N}_t.fci)=R_j.trans$.若否,则生成$\mathcal{N}_i$的新孩子节点$\mathcal{N}_k$,其对应资源集为 $R_j.trans$,然后对其深度优先遍历;若是,则跳过该收缩对并检测下一个收缩对.由于枚举树是深度优先生成的,故只有$\mathcal{N}_k$所在的子树均已生成后才会考察其父节点$\mathcal{N}_i$中排在R_j后的收缩对.

对新生成的孩子节点$\mathcal{N}_k$,其支持度等于$|R_j.trans|$,极细化查询为 $concise(\mathcal{N}_i.fci\cup R_j.ref_item\cup\{R_m.ref_item:R_m$是$\mathcal{N}_i$的收缩对$\wedge R_m$排在$R_j$之后$\wedge R_j.trans\subseteq R_m.trans\})$.$\mathcal{N}_k$的收缩对序列为$\{\langle R_m.ref_item,R_m.trans\cap R_j.trans\rangle:R_m$是$\mathcal{N}_i$的收缩对$\wedge R_m$排在$R_j$之后$\wedge\delta\leq|R_j.trans\cap R_m.trans|<|R_j.trans|\}$的任意排列.

新节点的初始化过程举例如下.$\mathcal{N}_1$的第 1 个孩子$\mathcal{N}_2$对应其第 1 个收缩对$\langle b,\{1,3,4,5\}\rangle$,其 $trans$ 和后面的收缩对的交分别为$\{4,5\}$,$\{4,5\}$,$\{1,3,5\}$和$\{1,3,4,5\}$.由此可见,d,e,g 可将$\mathcal{N}_2$细化为不小于阈值的更小的资源集,而 h 无法将其细化得更小.因此,$\mathcal{N}_2.fci=concise(\mathcal{N}_1.fci\cup\{b\}\cup\{h\})=\{c,b,h\}$(图中以圆圈标识).同理,$\mathcal{N}_2$的第 1 个收缩对对应其第 1 个孩子$\mathcal{N}_3$.该收缩对与其后收缩对的交分别为$\{4,5\}$和$\{5\}$.故$\mathcal{N}_3.fci=concise(\mathcal{N}_2.fci\cup\{d\}\cup\{e\})=\{b,h,e\}$,且$\mathcal{N}_3$的收缩对序列为空(因为$\delta=2$).然后,考察$\mathcal{N}_2$的第 2 个收缩对$\langle e,\{4,5\}\rangle$,它的 $trans$ 和$\mathcal{N}_2$相同,故冗余,不生成新节点.不断执行上述过程直至遍历生成整棵枚举树.

值得注意的是,在上述过程中:(1)收缩对序列的初始顺序可以任意;(2)令当前正在考察$\mathcal{N}_i$的收缩对R_j,若其 $trans$ 包含或等于其后某收缩对R_t的 $trans$,则可以删除R_t(因为它必冗余).

关于类别集枚举树的正确性,可参照频繁闭项集挖掘算法中的相关证明[31].

4.1.2　边更新

L-Miner 实时维护所有已生成节点之间基于偏序关系的哈斯图.每生成一个新节点,就更新此哈斯图的边.因为层次概念格就是所有节点之间基于偏序关系的哈斯图,所以当所有节点生成时,同时得到格中所有边.

虽然树节点和格节点一一对应,但是树中边和格中边不存在这种关系,具体而言:

(1)若节点 Y 是节点 X 在枚举树上的孩子,则 Y 不一定是 X 在格中的直接后继;

(2)若节点 Y 是节点 X 在枚举树上的后代但非孩子,则 Y 一定不是 X 在格中的直接后继;

(3)若节点Y是节点X在格中的直接后继,则Y可能在枚举树深度优先遍历生成过程中先于 X 生成.

性质 1. 当节点所在子树均已生成时，该节点在格中的所有后继节点均已生成.

基于上述性质，我们设计如下边增量维护方法. 首先，类别集枚举树的节点生成顺序和前序遍历一致，但输出顺序和后序遍历一致. 其次，树节点输出时，除了输出其极细化查询和支持度外，同时输出该节点在格中到其直接后继的边.

我们通过如下方法确定树节点输出时其在格中的直接后继. 对枚举树中当前路径上每个节点$\mathcal{N}$，我们动态维护其在层次概念格中的直接后继列表，记为 $succs(\mathcal{N})$. 根据性质 1，当$\mathcal{N}$输出后，$succs(\mathcal{N})$不再发生变化，即为$\mathcal{N}$在最后得到的格中的所有直接后继.

传统方法一般采用分阶段法挖掘格，先生成所有节点，然后确定边. 由于节点数目通常较为庞大，在确定每个节点的后继节点时，需要比较大量的节点. 我们使用的边增量计算方法可以减小需要比较的节点，从而提高边的计算效率.

4.2 冗余检测

冗余检测是确定是否存在已生成的树节点，其查询结果与给定收缩对的资源集相同. 冗余检测是类别集枚举树生成过程中的关键操作，节点生成效率很大程度上取决于冗余检测的效率. 显然，扫描所有已生成的节点并进行比较的效率太低. 在此，我们基于 3.1 节中提出的关于极小极细化查询的倒排索引来完成高效的冗余检测.

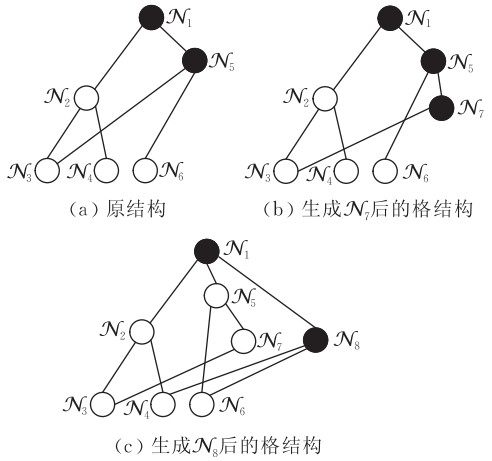
性质 2. 令当前树节点为$\mathcal{X}$，$\mathcal{X}$中待检测的收缩对为$\langle u,S\rangle$. $\langle u,S\rangle$是冗余的当且仅当$\exists\mathcal{Y}\in succs(\mathcal{X})$，$locate(\mathcal{Y}.fci\cup\{u\})$返回非空节点，且其支持度等于$|S|$.

例如，对图 4 中节点$\mathcal{N}_8$，其收缩对序列仅含一个收缩对 $R=\langle h,\{1,3,5\}\rangle$. 当$\mathcal{N}_8$新生成时，其后继节点为$\mathcal{N}_4$和$\mathcal{N}_6$. 对 R 进行冗余检测时，只需查验$\mathcal{N}_4$和$\mathcal{N}_6$被 h 细化后能否得到$\{1,3,5\}$. 因为 $h\in\mathcal{N}_4.fci$，所以 $locate(\mathcal{N}_4.fci\cup\{h\})$返回节点$\mathcal{N}_4$. 又$\mathcal{N}_4.supp=3$，故该收缩对是冗余的.

因此，冗余检测中的关键操作就是 3.1 节中讨论的定位操作，基于极小极细化查询的倒排索引可以高效地完成.

4.3 节点后继维护

图 5(a)显示了$\mathcal{N}_6$输出时的格结构. 与树结构相比，多了一条边$\langle\mathcal{N}_5,\mathcal{N}_3\rangle$. 图 5(b)显示了$\mathcal{N}_7$生成后已挖掘格结构的变化情况. 除新节点$\mathcal{N}_7$和其父节点$\mathcal{N}_5$外，其余节点的后继不发生变化. 边$\langle\mathcal{N}_5,\mathcal{N}_3\rangle$被$\langle\mathcal{N}_7,\mathcal{N}_3\rangle$取代；另外，多了一条新节点到其父节点的边$\langle\mathcal{N}_5,\mathcal{N}_7\rangle$. 图 5(b)显示了$\mathcal{N}_8$生成后已挖掘格结构的变化情况. 除新增新节点到其父节点的边$\langle\mathcal{N}_1,\mathcal{N}_8\rangle$外，还多了两条新节点到其初始后继节点的边$\langle\mathcal{N}_8,\mathcal{N}_4\rangle$和$\langle\mathcal{N}_8,\mathcal{N}_6\rangle$.



(a) 原结构 (b) 生成$\mathcal{N}_7$后的格结构

(c) 生成$\mathcal{N}_8$后的格结构

图 5 结点生成后格结构的变化(当前路径上节点为黑色，已输出节点为白色)

根据上述例子，易证新节点生成时，除新节点和其父节点的后继发生变化外，其余节点的后继保持不变. 因此，我们在枚举树生成过程中同时生成所有节点到其后继节点的边，在格生成后根据后继关系很方便就得到前驱关系，由此完成格的挖掘.

令$\mathcal{Y}$是新生成的节点，对应其树上父节点$\mathcal{X}$的收缩对$\langle u,S\rangle$. 我们只需初始化$\mathcal{Y}$的直接后继并更新$\mathcal{X}$的后继，其余节点的直接后继不发生变化. 具体而言：

(1) $succs(\mathcal{Y})\leftarrow\max\{\mathcal{V}:\mathcal{Z}\in succs(\mathcal{X})\wedge\mathcal{V}=locate(\mathcal{Z}.fci\cup\{u\})\wedge\mathcal{V}$非空$\}$，其中节点之间的偏序关系与其极细化查询之间的偏序关系一致；

(2) $succs(\mathcal{X})\leftarrow succs(\mathcal{X})-succs(\mathcal{Y})\cup\{\mathcal{Y}\}$.

因此，关键操作仍然是 3.1 节中讨论的定位操作，可高效地完成.

5 实验结果与分析

本节通过实验分析我们提出的格挖掘算法 L-Miner的运行速度和索引大小. 运行环境为 Lenovo 启天 M7000，CPU 为双核 2.93GHz，2GB 内存，Ubuntu9.04 操作系统. 代码用 C++编写.

我们采用 UCIKDDArchive① 中的公开数据集

① http://kdd.ics.uci.edu/

chess，connect，mushroom 和 pumsb，其中每条记录对应一个资源，每个项集对应一个类别. 频繁项集挖掘算法中广泛采用这些数据集进行性能分析[20-31]. 表 3 列举了每个数据集的总资源个数(＃*Trans*)、总类别个数(＃*Items*)、每个资源对应的平均类别数(*AL*)和最多类别数(*ML*). mushroom 数据集描述了不同种类蘑菇的特征，pumsb 数据集是人口普查信息，它们代表了实际应用中可能遇到的分面数据库. 因此，格挖掘算法在 mushroom 和 pumsb 数据集上的性能可以有效指导基于层次概念格的分面导航系统的构建.

表 3　数据集特征

数据集	＃*Trans*	＃*Items*	*AL*	*ML*
chess	3196	76	37	37
connect	67557	130	43	43
mushroom	8124	119	23	23
pumsb	49046	2113	74	74

尽管大量的频繁闭项集挖掘可用来计算格节点，但只有 CHARM-L[31] 同时高效地计算边. 因此，对比算法选择 CHARM-L. 以下各图的 *min_sup* 指查询结果所包含的最少资源个数 δ 占总资源数的比例.

5.1　索引规模

L-Miner 用倒排表索引目前已生成的所有极小极细化查询，而 CHARM-L 用倒排表索引所有已生成的极细化查询. 因此，二者的索引大小之比约等于格中极小节点数目除以所有节点数目.

图 6 表明 L-Miner 的索引远小于 CHARM-L 的索引. 例如，在 chess 数据集上，当 *min_sup* 为 60% 时，极小节点数为 3323，而所有节点数为 98 393，故 L-Miner 的索引大小约占 CHARM-L 的索引大小的 3.4%(≈3323/98 393). 在 connect(*min_sup*＝65%)、mushroom(*min_sup*＝1%)和 pumsb(*min_sup*＝65%)数据集上，该比例分别为 3.2%(≈1587/49 705)、13.1%(≈6768/51 640)和 3.4%(≈18 182/496 070).

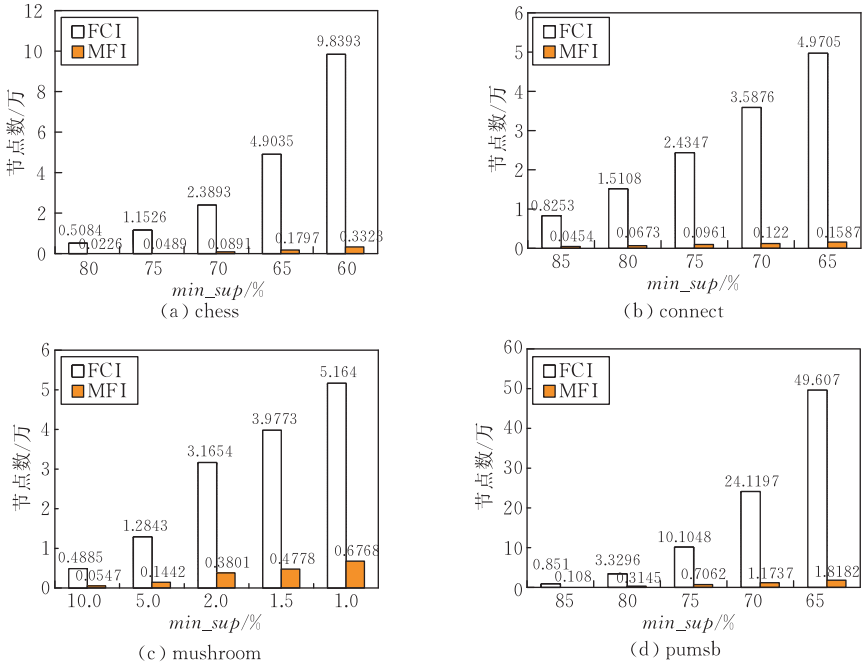


图 6　表 3 中数据集在不同 *min_sup* 下的格中节点数(FCI)和极小节点数(MFI)

5.2　格挖掘速度

L-Miner 和 CHARM-L 的节点生成方式是一致的，即对同样的收缩对排序方式，所有节点的生成顺序相同. 它们的主要区别在于冗余检测和边增量生成采用了不同的数据结构和算法. 我们考察了 3 种收缩对排序方式：① 表示按其细化类别的标识符大小排序；② 表示按其细化结果的大小升序排序；③ 表示按其细化结果的大小降序排序. 对于每个格，我们测试了 L-Miner 和 CHARM-L 在这 3 种序下的挖掘时间，结果如图 7 所示.

可以看出：(1) L-Miner 对序的变化不敏感，而 CHARM-L 在序 ② 下的速度明显慢于其它序；(2) L-Miner 在任意序下的速度均快于或接近于 CHARM-L 在特定序下达到的最快速度，且随着格的增加，前者远快于后者；(3) 随着格大小增加(即 *min_sup* 减小)，L-Miner 的挖掘时间缓慢增长，而

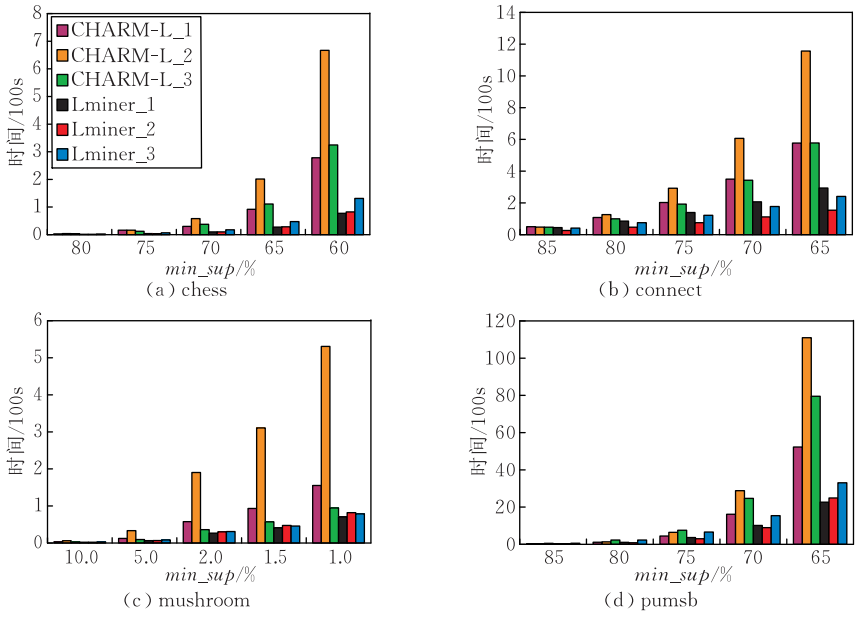


图 7 表 3 中数据集在不同 *min_sup* 下的格挖掘速度

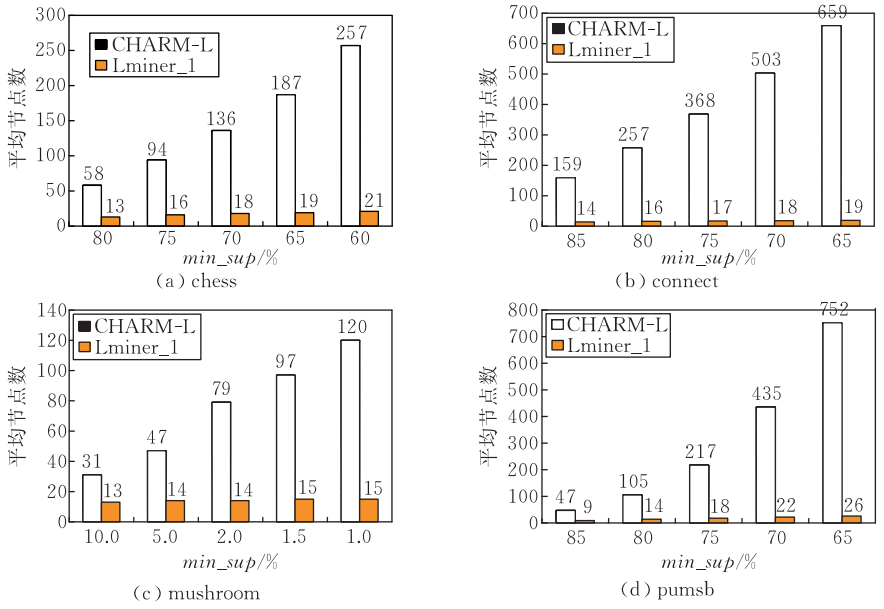


图 8 考察每个收缩对时需访问的平均节点数

CHARM-L 在各种序下均快速增长. 例如,在 chess 数据集上,当 *min_sup* 为 60%时,L-Miner 在不同收缩对排序方式下的平均速度约为 CHAMR-L 的 4.4 倍. 在 connect(*min_sup* = 65%)、mushroom(*min_sup*=1%)和 pumsb(*min_sup*=65%)数据集上,该速度比值分别为 3.4 倍、3.4 倍和 3.0 倍.

图 8 描述了序②下 L-Miner 和 CHARM-L 在考察每个收缩对时所需访问的平均节点数. 在 chess 数据集上,随着格节点数从 5084 增加到 98 393,L-Miner 所需访问的平均节点数从 13 缓慢增加至 21,而 CHAMR-L 则从 58 快速增加到 257. 在数据集 connect 上,随着格节点数从 8253 增加到 49 705,L-Miner 所需访问的平均节点数从 14 缓慢增加至 19,而 CHAMR-L 则从 159 快速增加到 659. 在 mushroom 数据集上,随着格节点数从 4885 增加到 51640,L-Miner 所需访问的平均节点数从 13 缓慢增加至 15,而 CHAMR-L 则从 31 快速增加到 120. 在 pumsb 数据集上,随着格节点数从 8510 增加到 496 070,L-Miner 所需访问的平均节点数从 9 缓慢增加至 26,而 CHAMR-L 则从 47 快速增加到 752. 因此,L-Miner 随着格的增大具有良好的可扩展性.

6　结论及下一步研究

本文提出了面向分面导航的层次概念格模型和挖掘算法.其主要贡献包括：(1)提出了描述所有浏览状态及其关系的本体——层次概念格，用来建模分面导航过程中用户在不同浏览状态之间的跳转；(2)提出了基于层次概念格的导航操作，以支持用户在不同浏览状态之间更灵活地跳转，从而更有效地进行知识发现；(3)提出了格挖掘算法 L-Miner，并基于公开数据集进行了深入的性能分析；实验结果表明 L-Miner 的格挖掘速度远快于现有算法，而其构建的索引结构的存储代价更低.进一步的工作包括对基于层次概念格的分面导航原型系统实现，以及将该技术应用于个人信息管理、电子商务和数字图书馆.

参　考　文　献

[1] Bergman M K. The deep web: Surfacing hidden value. Journal of Electronic Publishing, 2001, 7(1): 113-153

[2] Kashyap A, Hristidis V, Petropoulos M. FACeTOR: Cost-driven exploration of faceted query results//Proceedings of the 17th International Conference Information and Knowl-Edge Management (CIKM'10). Toronto, Ontario, Canada, 2010: 719-728

[3] Hearst M. Clustering versus faceted categories for information exploration. Communications of the ACM, 2006, 49(4): 59-61

[4] Roy S B et al. Minimum-effort driven dynamic faceted search in structured databases//Proceedings of the 17th International Conference on Information and Knowl-Edge Management (CIKM'08). Napa Valley, California, USA, 2008: 13-22

[5] Roy S B et al. DynaCet: Building dynamic faceted search systems over databases//Proceedings of the 25th International Conference on Data Mining (ICDE'09). Shanghai, China, 2009: 1463-1466

[6] Kashyap A et al. Exploring biomedical databases with BioNav //Proceedings of the 28th ACM SIGMOD International Conference on Management of Data (SIGMOD'09). Providence, Rhode Island, USA, 2009: 1079-1082

[7] Kashyap A, Hristidis V, Petropoulos M, Tavoulari S. Effective navigation of query results based on concept hierarchies. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(4): 540-553

[8] Koren J, Zhang Y, Liu X. Personalized interactive faceted search//Proceedings of the 17th International Conference on World Wide Web (WWW'08). Beijing, China, 2008: 477-486

[9] Ben-Yitzhak O et al. Beyond basic faceted search//Proceedings of the International Conference on Web Search and Web Data Mining (WSDM'08). Palo Alto, California, USA, 2008: 33-44

[10] Dash D et al. Dynamic faceted search for discovery-driven analysis//Proceedings of the 17th International Conference on Information and Knowledge Management (CIKM'08). Napa Valley, California, USA, 2008: 3-12

[11] Wu P, Sismanis Y, Reinwald B. Towards keyword-driven analytical processing//Proceedings of the 26th ACM SIGMOD International Conference on Management of Data (SIGMOD'07). Beijing, China, 2007: 617-628

[12] Tunkelang D. Dynamic category sets: An approach for faceted search//Proceedings of the ACM SIGIR'06 Workshop on Faceted Search. Seattle, Washington, USA, 2006: 12-16

[13] Dakka W, Ipeirotis P, Wood K. Automatic construction of multifaceted browsing interfaces//Proceedings of the 14th International Conference on Information and Knowledge Management (CIKM'05). Bremen, Germany, 2005: 768-775

[14] Dakka W, Ipeirotis P G. Automatic extraction of useful facet hierarchies from text databases//Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE'08). Cancun, Mexico, 2008: 466-475

[15] Ling X et al. Mining multi-faceted overviews of arbitrary topics in a text collection//Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08). Las Vegas, Nevada, USA, 2008: 497-505

[16] Oren E, Delbru R, Decker S. Extending faceted navigation for RDF data//Proceedings of the 5th International Semantic Web Conference (ISWC'06). Athens, Georgia, 2006: 559-572

[17] Sinha V, Karger D R. Magnet: Supporting navigation in semis-tructured data environments//Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'05). Baltimore, Maryland, USA, 2005: 97-106

[18] Dachselt R et al. FacetZoom: A continuous multi-scale widget for navigating hierarchical metadata//Proceedings of the 26th SIGCHI Conference on Human Factors in Computing Systems (SIGCHI'08). Florence, Italy, 2008: 1353-1356

[19] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations. Heidelberg: Springer, 1999

[20] Burdick D, Calimlim M, Gehrke J. MAFIA: A maximal frequent itemset algorithm for transactional databases//Proceedings of the 17th International Conference on Data Mining (ICDE'01). Heidelberg, Germany, 2001: 443-452

[21] Grahne G, Zhu J. Efficiently using prefix-trees in mining frequent itemsets//Proceedings of the IEEE ICDM'03 Workshop on Frequent Itemset Mining Implementations. Melbourne, Florida, USA, 2003: 123-132

[22] Han J, Wang J, Lu Y, Tzvetkov P. Mining top-k frequent closed patterns without minimum support//Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02). Maebashi City, Japan, 2002: 211-218

[23] Lucchese C, Orlando S, Perego R. Fast and memory efficient mining of frequent closed itemsets. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(1): 21-36

[24] Moonestinghe H, Fodeh S, Tan P N. Frequent closed itemset mining using prefix graphs with an efficient flow-based pruning strategy//Proceedings of the 6th IEEE International Conference on Data Mining(ICDM'06). Hong Kong, China, 2006: 426-435

[25] Pasquier N et al. Discovering frequent closed itemsets for association rules//Proceedings of the 7th International Conference on Database Theory (ICDT'99). Jerusalem, Israel, 1999: 398-416

[26] Pasquier N, Bastide Y, Taouil R, Lakhal L. Efficient mining of association rules using closed itemset lattices. Information Systems, 1999, 24(1): 25-46

[27] Pei J, Han J, Mao R. CLOSET: An efficient algorithm for mining frequent closed itemsets//Proceedings of the SIGMOD'00 Workshop on Data Mining and Knowledge Discovery. Dallas, Texas, USA, 2000: 21-30

[28] Singh N, Singh S, Mahanta A. Closeminer: Discovering frequent closed itemsets using frequent closed tidsets//Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05). Houston, Texas, USA, 2005: 633-636

[29] Uno T, Kiyomi M, Arimura H. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets//Proceedings of the IEEE ICDM'04 Workshop on Frequent Itemset Mining Implementations. Brighton, UK, 2004

[30] Wang J et al. CLOSET+: Searching for the best strategies for mining frequent closed itemsets//Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03). Washington, DC, USA, 2003: 236-245

[31] Zaki M J, Hsiao C. Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(4): 462-478

HE Chao, born in 1982, Ph. D. candidate. His research interests include data mining, faceted search and exploratory search.

CHENG Xue-Qi, born in 1971, professor, Ph. D. supervisor. His research interests include network science, web search & data mining, and P2P & distributed computing.

GUO Jia-Feng, born in 1980, Ph. D., assistant professor. His research interests include Web search and mining, machine learning, social network.

Background

Recently, there is a growing interest on exploratory search. Traditional information retrieval often assumes that users can pose specific queries and such queries can be answered by a relevant document. In real life, however, users often pose exploratory queries due to vague information need or exploratory search behavior. Traditional information retrieval adopts a query- and-response paradigm, while in exploratory search, information need is satisfied by a series of consecutive queries. Users need to investigate and learn from the query results, in order to form a more specific query at the next step.

Faceted search is an important kind of exploratory search, which emerges in the fields of enterprise search and vertical search. Its goal is organizing query results into a dynamic taxonomy according to the faceted metadata associated with resources. Faceted navigation is now becoming the dominant search user interface in e-Commerce websites (like eBay and Amazon) and digital libraries (like ACM digital library). The main focus of faceted navigation lies on three aspects: facet selection, facet ranking and facet mining.

Facet selection is an essential task in faceted navigation. Given a query, it decides what facet and their values appear in the query results, which compose a dynamic taxonomy. Hence users can avoid dead-end in the process of navigation. Such a dynamic taxonomy is also effective in analyzing and understanding query results, by previewing the next-step refinement results. Most research efforts are devoted to a further selection in case that dynamic taxonomy is prohibitively large. However, current faceted navigation can only preview one-step refinement. This paper aims to build up a data model describing the relations among different navigation states. The model not only guides more effective faceted selection, but also serves as a global map in the process of faceted navigation.

Authors' recent studies mainly focus on facet selection. Authors have made some in-depth research on the data model of faceted navigation and efficient algorithms of pre-computing the model for the sake of real-time response in faceted navigation.

The research is part of the work in the 863 project "New Theories and Methods on Web Search and Data Mining" (No. 60933005). Our group has already made a significant progress, including the work on learning to rank, user query understanding, faceted navigation and recommendation, which have been published in top conferences such as SIGIR, WWW, IJCAI and CIKM.