基于闭合频繁 Induced子树的 GML文档结构聚类

苗建新, 吉根林, 朱颖雯

(南京师范大学 计算机科学与技术学院, 江苏 南京 210097)

[摘要] 提出了一种 GML文档结构聚类新算法 MCF_CLU. 与其它相关算法不同, 该算法基于闭合频繁 Induced子树进行聚类, 聚类过程中不需树之间的两两相似度比较, 而是挖掘 GML文档数据库的闭合频繁 Induced子树, 为每个文档求一个闭合频繁 Induced子树作为该文档的代表树, 将具有相同代表树的文档聚为一类. 聚类过程中自动生成簇的个数, 为每个簇形成聚类描述, 而且能够发现孤立点. 实验结果表明算法 MCF_CLU是有效的, 且性能优于其它同类算法.

[关键词] 闭合频繁 Induced子树, GML结构聚类, 聚类

[中图分类号] TP 311 [文献标识码] A [文章编号] 1672-1292(2009)02-0061-04

Clustering GML Documents by Structure Based on Closed Frequent Induced Subtrees

Miao Jianxin, Ji Genlin, Zhu Yingwen

(School of Computer Sciences, Nanjing Normal University, Nanjing 210097, China)

Abstract This paper presents an algorithm MCF_CLU for clustering GML documents by structure. Different from other algorithms, it goes on clustering based on the closed frequent induced subtrees, and doesn't need comparing the similarity between trees. The closed frequent induced subtrees of all the GML documents are computed. The representative closed frequent induced subtree of every document is obtained. The documents which have the same representative tree are regarded as a cluster. During the clustering process, not only the number of clusters can be obtained automatically, but the description of the clusters can be achieved. By the way, the isolated points of the documents can be found. The experimental results show that MCF_CLU is effective, and that its performance is superior to those of other GML clustering algorithms.

Key words closed frequent induced subtree; clustering GML by structure; clustering

GML(Geography Markup Language)是一种树型结构的地理信息编码方式, GML文档结构聚类本质上是树的聚类. 近年来对树聚类的研究侧重在相似度度量上, 可以分为两大类, 一类是用编辑距离[1,2]来度量, 编辑距离是指由一棵树转变成另一棵树的最少操作步骤; 另一类是用 Jaccard系数度量相似度[3]. 以编辑距离度量相似度, 精度高, 但是计算复杂; 基于 Jaccard系数度量相似度, 计算简单, 但精度低. 目前树聚类方法通常使用 HAC(Hierarchical Agglomerative Clustering)或 ROCK(Robust Clustering Algorithm for Categorical Attributes)[4]聚类. 这些聚类方法都需要事先确定簇的个数, 而且聚类过程中, 需要大量地对整个数据集进行两两相似度计算, 使得聚类效率很差.

为此, 本文提出了一种基于闭合频繁 Induced子树的 GML文档结构聚类算法 MCF_CLU. 该算法的最大特点是不需要树之间的大量的相似度比较, 而且不用给定聚类的簇的个数. 首先对 GML文档预处理, 得出 GML文档对应的最小树结构. 然后进行闭合频繁 Induced子树的挖掘, 同时记录每个闭合频繁 Induced子树对应的文档集合. 为每个文档求一个闭合频繁 Induced子树作为该文档的代表树, 将具有相同代表树的文档聚为一类. 聚类过程中自动生成簇的个数, 为每个簇形成聚类描述, 而且能够发现孤立点. 实验结果

收稿日期: 2008-04-10

基金项目: 国家自然科学基金(40771163)资助项目.

通讯联系人: 吉根林, 教授, 博士生导师, 研究方向: 数据挖掘及应用技术、XML技术. E-mail: glji@njnu.edu.cn

表明算法 MCF_CLU 是有效的,且性能优于其它同类算法.

1　相关概念

定义 **1**　Induced子树. 给定根有序标号树 $T = (V, E, L, v_0)$, $T' = (V', E', L', v_0')$. 若满足以下条件,则称树 T'是树 T 的 Induced子树.

① $V' \subseteq V$, $E' \subseteq E$.

② L'保持 T'中 V'的标号.

③若节点 v_1, $v_2 \in V$, v_1, $v_2 \in V'$, 且先序遍历 T时 v_1在 v_2之前,则先序遍历 T'时, v_1 也在 v_2之前.

④若节点 v_1, $v_2 \in V$, v_1, $v_2 \in V'$, 且在 T'中 v_1是 v_2的双亲,则在 T中 v_1 是 v_2的双亲.

定义 **2**　频繁子树. 给定根有序标签树的数据库 TDB以及子树 T, T的支持度定义为 $s(T) = |p(T)/N|$, 其中$p(T)$是 TDB中包含 T的树的棵数, N是 TDB中树的棵数. 所以, T是频繁子树当且仅当 $s(T) >= minsup$. minsup是用户指定的最小支持度阈值.

定义 **3**　闭合频繁子树. 如果子树 T是频繁子树,且它的任何超树的支持度均小于 T的支持度,则称 T为闭合频繁子树.

2　聚类算法 MCF_CLU

本文提出的 MCF_CLU 算法与传统的聚类算法不同,没有涉及到相似度的比较,是一种基于闭合频繁 Induced子树的映射聚类,在聚类过程中自动确定簇的个数.

在聚类过程中,首先对 GML文档预处理,将 GML文档解析成一棵 DOM树,然后将 GML文档最小化,去除它的重复节点.重复节点包括 nestedrepeated节点与 repeated节点. nestedrepeated节点是指标签和祖先节点标签相同的非叶子节点, repeated节点是指从根节点到该节点的路径重复出现的节点. Nesting Reduction是指将 nestedrepeated节点的后继节点上移,如图 1中 Step1所示; Repeating Reduction是指删除 repeated节点并把它的后继节点合并到与该节点路径相同的节点下面,如图 1中 Step2所示.

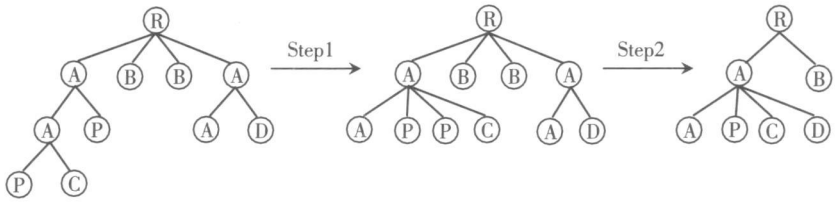

图 1　GML 文档最小化处理

Fig.1　The minimized process of GML document

在对 GML文档预处理后,进行闭合频繁 Induced子树的挖掘,同时记录每棵闭合频繁 Induced子树出现的文档集合.闭合频繁 Induced子树的挖掘参见 CMTreeMiner[6]算法.每个文档不只有一个闭合频繁 Induced子树,节点个数最多的子树作为文档的代表树,具有相同代表树的文档聚为一类.下面给出 GML文档结构聚类算法 MCF_CLU 的算法描述.

算法 **1**　MCF_CLU.

输入: GML文档集合 D,阈值参数 θ,最小支持度 minsupp;

输出: GML文档类别;

(1) TDB= PreProcess(D);　　//GML文档预处理,得到代表 GML文档的最小结构

(2) GetClosedFrequentSet(TDB, CL, LT, minsupp);

　　//求取 TDB在 minsupp下的闭合频繁 Induced子树集合 CL以及对应的文档集合 LT

(3) ProcessSubTrees(CL, LT, θ);　　//对求得的闭合频繁 Induced子树处理,减少子树数目

(4) GetRepresentSubtree(CL, LT, D);　　//计算 GML文档的代表树,具有相同代表树的文档聚为一类,没有代表树的作为孤立点

2.1　闭合频繁 Induced子树的选择

本文提出的 MCF_CLU 本质上是一种基于闭合频繁 Induced子树的映射聚类,为避免聚类时划分过

细, 对挖掘出的闭合频繁 Induced子树进行预处理, 从中选择最能表征数据库特征的子树参加下面的聚类. 对闭合频繁 Induced子树的预处理基于这样的思路: 若 CL中两棵子树含有的相同节点个数所占比例超过阈值 θ, 则保留 CL中节点个数多的子树, 并且将这两棵闭合频繁 Induced子树对应的 GML文档集合进行合并. 闭合频繁 Induced子树的选择算法见算法 2.

算法 **2** ProcessSubTrees.

输入: 闭合频繁 Induced子树集合 CL, CL对应的 GML文档集合 LT和阈值参数 θ;

输出: 处理后的 CL和 LT;

```
(1) for each t_i∈ CL do
(2) for each t_j∈ CL and t_i! = t_j do
(3) { snode= SameNode(t_i, t_j);          //计算 t_i, t_j的相同节点个数
(4) if (NodeNum(t_i) > NodeNum(t_j))      //t_i的节点个数比 t_j的节点个数多
(5)     { if (snode/NodeNum(t_i) > θ)
(6)       {LT[t_i] = LT[t_i]∪ LT[t_j];    //t_i和 t_j对应的文档合并
(7)        Delete(t_j)}                   //删除 t_j
(8) }
(9) else
(10) if (snode/NodeNum(t_j) > θ)
(11)    { LT[t_i] = LT[t_i]∪ LT[t_j];     //t_i和 t_j对应的文档合并
(12)      Delete(t_i)}                    //删除 t_i
(13) }
```

2.2 文档代表树的选择

在闭合频繁 Induced子树求出以后, 接下来需要为每个文档求取代表树, 具体算法见算法 3.

算法 **3** GetRepresentSubtree.

输入: GML文档集 D, 闭合频繁 Induced子树集 CL及对应的文档集 LT;

输出: GML文档代表树;

```
(1) for each d∈ D do
(2) repsent[d] = NULL;                    //开始 d的代表树为空, 最后代表树为空的文档为孤立点
(3) pos= SortTree(CL);                    //对 CL中每棵子树按其节点数降序排列,
                                          //并将其对应文档位置保存在数组 pos中
(4) for each d∈ LT[pos[0]] do             //LT[pos[0]]为子树 CL[0]对应的文档集
(5)     repsent[d] = CL[0];               //LT[pos[0]]中每个文档的代表树为 CL[0]
(6) for (i= 1; i< CL.size; i++) do        //扫描闭合频繁 Induced子树集 CL中的每个子树
(7) for each d∈ LT[pos[i]] and repsent[t] = = NULL do
(8)     repsent[d] = CL[i]}               //LT[pos[i]]中没有代表树的文档, 其代表树为 CL[i]
```

3 实验结果与分析

为验证本文提出的 MCF_CLU算法的有效性, 在 Intel Pentium Ⅳ 2.93 GHz, 内存 512M的微机上, 将 MCF_CLU算法利用 VC++加以实现, 并与 S-Grace算法[3]以及采用边集比较的 HAC算法进行了比较实验.

HAC是用来进行树聚类最多的聚类算法, 而边集比较的相似度度量方法在精度和复杂度方面都适中. S-Grace算法是 Wang Lian等 2004年提出的, 算法的大体思路是: 对 XML文档预处理, 得到代表 XML文档的 s-graph, s-graph之间的相似度比较也是采用边集比较法进行, 在聚类过程中采用 ROCK聚类算法.

层次聚类或 S-Grace算法在聚类过程中, 都要涉及到大量的树与树之间的相似度的计算, 而相似度计算是个很复杂的过程, 因此聚类的时间性能很差. 图 2是 3个算法的时间性能比较, GML数据集是根据预先定义的 DTD自动生成的文档集合, 在预处理后平均每个文档有 20个节点, 每个数据集有 4类分布均匀的文档. 从图 2可以看出, 本文的算法在时间性能上有绝对的优势.

本文提出的 MCF_CLU 算法是基于闭合频繁 Induced子树的,因此支持度的选取对于算法的时间性能以及聚类结果都有着不可忽视的影响.支持度取得太小,会导致产生过多的闭合频繁 Induced子树,增加聚类时间,会导致本来应分到一起的文档集合分为更小的簇.当支持度增加到一定程度后,会导致本应产生的代表一个簇的闭合频繁 Induced子树不再频繁,会产生过多的孤立点,用户在没有先验知识的前提下,可以通过反复的试验取得满足自己要求的最小支持度,得到满意的聚类结果.表 1给出的是 3个算法的聚类精度比较.

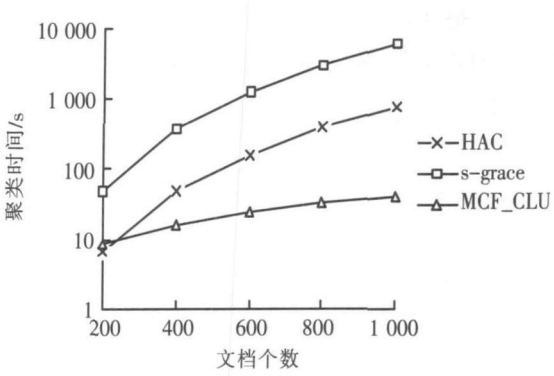

图 2　聚类时间性能比较

Fig.2　The time comparison of clustering algorithms

4　结　语

本文针对 GML文档结构聚类提出了 MCF_CLU 算法.该算法通过闭合频繁 Induced子树的挖掘为每个文档求得最大闭合频繁 Induced子树,根据最大闭合频繁 Induced子树的分布进行聚类,不需要事先确定聚类个数,在聚类过程中能够发现孤立点,而且能为每个簇形成聚类描述.实验表明该算法有很好的性能.

表 1　算法的平均查准率比较表

Table 1　The comparison of average precision ratio of clustering algorithms

算法	平均聚类精度
HAC	100%
s-grace	100%
MCF_CLU	98%

[参考文献](References)

[1] Chawathe S S. Comparing hierarchical data in external memory[C]//Proceedings of the VLDB Conference. San Francisco: Morgan Kaufmann Publishers Inc, 1999: 90-101.

[2] De Francesca F, Gordano G, Ortale R, et al. A general framework for XML document clustering[R]. ICAR-CNR (Consiglio Nazionale delle Ricerche Istituto di Calcolo e Reti ad Alte Prestazioni), 2003.

[3] Lian W, Cheung D W, Mamoulis, et al. An efficient and scalable algorithm for clustering XML documents by structure[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(1): 82-96.

[4] Guha S, Rastogi R, Shim K. ROCK: a robust clustering algorithm for categorical attributes[C]//Proceedings of ICDE99 (International Conference on Data Engineering). Los Alamitos: IEEE Computer Society, 1999: 512-521.

[5] Dalamagas T, Cheng T, Winkel K, et al. Clustering XML documents using structural summaries[C]//Current Trends in Database Technology-EDBT 2004 Workshops. Berlin: Springer, 2004: 547-556.

[6] Chi Y, Xia Y, Yang Y, et al. Mining closed and maximal frequent subtrees from databases of labeled rooted trees[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(2): 190-202.

[责任编辑: 严海琳]