



New Path Filling Method on Data Preprocessing in Web Mining

Chungsheng Zhang & Liyan Zhuang

School of Mathematics and Computer Science

Inner Mongolia University for the Nationalities

536 Huo Lin He Street, Tongliao, NeiMonggol 028043, China

E-mail: zhangcs_817@www.yahoo.com.cn

Fund Project: Inner Mongolia Educational Research Projects Subsidize(8)

Inner Mongolia talents Fund Projects Granted

Abstract

The article discusses the importance of data preprocessing in web mining and gives the topology structure for the website in the view of actual condition, analyzes the limitation of reference [3] and proposes a data structure based on adjacency list. The proposed method satisfies the actual condition of topology structure for the existed website. The special data structure and path filling algorithm based on adjacency list are given. The data structure satisfies the commonness of topology structure for the existed website and the time complexity is lower.

Keywords: Web mining, Data preprocessing, Path filling

1. Introduction

With the rapid development of Internet, and the gradual increase the amount of information, it is estimated that there has 350 million web pages in 1999, and is increasing the speed of one million per day. Google has recently declared it has indexed 3000 million web pages. The World Wide Web is the largest database at present, and it is a challenging task how to access effectively these data ^[1].

An effective approach to solving these problems is web mining. Web mining is that data mining technique is applied to web data to the discovery of the interesting usage patterns and implicit information ^[2].

However in fact, data mining has the strict quality requirements to these data that are deal with. A key step of data mining is the establishment of appropriate data sets ^[7], so it seems very important to carry out data preprocessing before data mining. According to statistics, two-thirds data mining analysts consider a complete data preprocessing spends about sixty percent of the whole mining time ^[8].

Web mining is classified into three categories: web content mining, web log mining, and web structure mining ^[6]. To web content mining and web structure mining, it seems be not critical of users identity, but when users are browsing web pages, because of existing the local cache and the proxy server cache, the web page got by users pressing "backward" button on browser, hasn't corresponding records in server log to web log mining ^[2], so we must carry out path filling, otherwise, it will seriously affect mining results.

The article [3] proposed an algorithm of STT, which a topology structure for web transforms into a binary tree, the method made some innovation indeed, but I think that there still exists some limitations in algorithm.

2. STT principle and limitation

The topology for website described in article [3] shows as figure 2-1, which is a tree structure, and in comparison with the actual condition exists in the following three problems:

- (1) The current real topology for website should be graph structure, shows as figure 2-2, and can completely exist path $E \rightarrow F$;
- (2) Generally, the depth is much larger than the width in the topology of website, tree transforms into binary tree in the article [3], which certainly will cause its search depth increase and algorithm efficiency decrease.
- (3) When general references solving path, all start from root node, while the actual condition is not true, solving path can start from any node.

3. Data structure based on adjacency list and path filling algorithm

3.1 Data structure of topology of web

After analysis and selection, data structure of topology for website uses the adjacent list data structure, constructed as follows.

Definition 3.1 Website nodes set can be described as a sequence L , L equals to $\{i \mid 1 \leq i \leq n, \text{ in which } n \text{ is the total node numbers}\}$, numbering of i starts from root node, the first is the internal layers from left to right, and the last is external layer from up to down, that is the width priority method sorts nodes.

For example, in figure2-2 symbolic node sequence is $\{A, B, C, D, E, F, G, H, I, J, K, L, M\}$, its corresponding digital sequence is L and L equals to $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$.

Theorem 3.1 Any filling path must be a node in searched path.

For example, suppose that there exists user path ABEIF in figure2-2, and filling path is ABEIBF, B is obviously a node in searched path.

Prove: The reasons of causing discontinuous point is that users come true by “backward” button on browser, and the backward node is certain in searched path, so theorem holds.

Theorem 3.2 If the sub-node numbers of a node in a searched path are less than or equal to 1, the node can not be filling node.

Prove: If the sub-node numbers of a node in a searched path equal to 0, the node is leaf node and can not be accessed by “backward” button, so the node can not be taken as filling node.

If the sub-node numbers of a node in a searched path equal to 1, and if the node can be arrived by “backward” button, the node is the direct parent node of its only sub-node, because it do not exist other sub-nodes, the node can not be taken as filling node.

Definition 3.2 Data structure of the topology for website is defined in language C as follows:

```
// defining adjacency sub-node
typedef struct CTNode{
    int child;                // sub-node number
    struct CTNode *next      // pointer of next node
}*ChildPtr;
// defining webpage node list
typedef struct{
    int child_count;         // the number of sub-node
    TElem Type data;        // node marking
    ChildPtr firstchild;    // pointer of the first node
}CTBox;
// defining website structure
typedef struct{
    CTBox nodes[MAX_TREE_SIZE]; // node list
int n, r;                   // the total number of node and the root of node
    }Ctree;
    int s_path[ ];           // original path
    int d_path[ ];          // path after filling
```

3.2 Path filling algorithm

Suppose i is the original path scan variable, j is the object path scan variable, and pre_i is the original path backtracking scan variable.

```
Void main ( )                // the main function of algorithm
{
int i =0;                    // initialize variables
```

```

int j=0;
int pre_i=0;
d_path[0]=s_path[0];           // deal with the first node of the original sequence
i=1;
for (i=1; len(s_path); i++)    // start from the second node and scan the original
    // sequence
{
    if (nodes[i-1].first=s_path[i]) { // is the first sub-node of next node, and assign value
// directly
        d_path[j]=s_path[i];
    else // otherwise, scan other sub-nodes
        { if (!find (i, j, nodes[i-1].first. next))
            for (pre_i = i-1; 0 ; i -- ) { // is not sub-node, and start backtracking
                if (nodes[pre_i].child_count <= 1)
                    continue; // if the sub-node numbers of the current node are less
else //than or equal to 1, by theorem 3.2, then continues
// backtracking, because the node is not filling node
                { if (!find ( pre_i, j, nodes[pre_i].first )
                    continue; // otherwise, scan sub-chain of the current node
                else
                    break; }; // found then return
                };
            };
        };
    i++;
    j++;
}

int function find ( int i, j, CTNode q ) // scan sub-chain function
{
int tag=0; // setting found mark: tag equals to 0, and means no
// found; tag equals to, and means found.

while (q) {
    if (q. child= s_path[i]) // found, then ends circulation
        { d_path[j]= s_path[i];
            tag =1;
            break; };
    else // otherwise, continues to scan
        { q=q. next;
            tag=0; };
    };
return tag;
}

```

4. Conclusion

Data structure based on adjacency list is proposed in the article, the proposed method satisfies the actual condition of topology structure for the existed website. Meanwhile, path filling algorithm based on adjacency list data structure is given. The algorithm overcomes the shortcomings that existed website transforms into binary tree to cause larger search depth, generally the width is more than the depth in existed website. And the algorithm uses only array, structure and simple circulation, so it is simple and low complexity.

References

Doro Tanasa, Brigitte Trousse. Advanced data preprocessing for intersites web usage mining[J]. IEEE Intelligent Systems, 2004, (3/4): 59-65.

Han, Jiawei, Kamber M. (2001). Conception and Technique of Data Mining. Beijing: Mechanical Industry Publishing House.

Jaideep Srivastava, Robert Cooley, Mukund Deshpande, et al. Web usage mining: Discovery and applications of usage patterns from web data [J]. Proc ACM SIGKDD, 2000, 1(2): 12-23.

Li, Chaofeng. (2007). Design and realization of data preprocessing algorithm in web usage mining. *Journal of South-Central University for Nationalities*, 26, 56-60.

Li, Liebiao, Zhang, Haipeng, Zhou, Yafeng. (2007). Research on data preprocessing methods in web log mining. *Computer Technology and Development*, 17,45-52.

Liu, Lijun, Zhou, Jun, Mei, Hongyan. (2007). Data preprocessing of web usage mining. *Computer Science*, 34, 200-204.

Ma, Ruiming, Li, Xiangyun. (2007). Research on data preprocessing technology in web log mining. *Computer Engineering and Design*, 28, 2358-2360.

Margaret H.Dunham. (2005). Data Mining Course. Beijing: Tsinghua University Press.

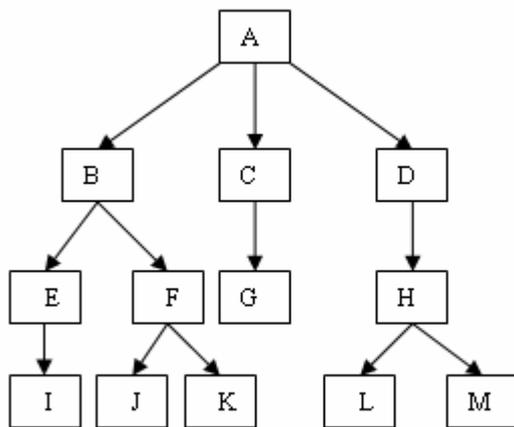


Figure 2-1 topology of web in article [3]

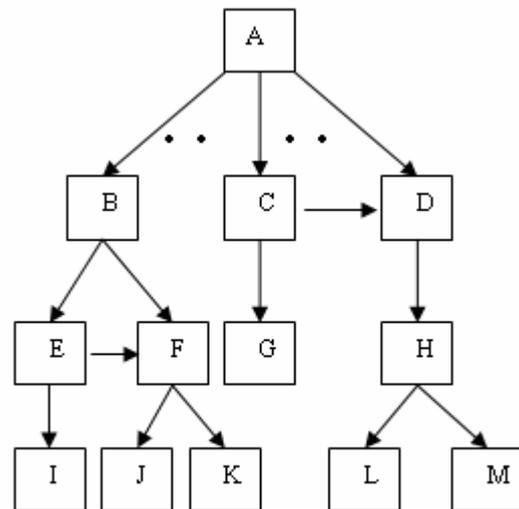


Figure 2-2 the actual topology for existed web