Chapter 6
TMG Framework for Mining Unordered Subtrees

6.1 Introduction

This chapter describes the extension of the TMG framework for the mining of unordered induced/embedded subtrees. While in online tree-structured documents such as XML the information is presented in a particular order, in many applications the order among the sibling-nodes is considered unimportant or irrelevant to the task and is often not available. If one is interested in comparing different document structures, or the document is composed of data from several heterogeneous sources, it is very common for the order of sibling nodes to differ, although the information contained in the structure is essentially the same. In these cases, mining of unordered subtrees is much more suitable as a user can pose queries and does not have to worry about the order. All matching sub-structures will be returned with the difference being that the order of sibling nodes is not used as an additional candidate grouping criterion. Hence, the main difference when it comes to the mining of unordered subtrees is that the order of sibling nodes of a subtree can be exchanged and the resulting tree is still considered the same.

To briefly illustrate this aspect, consider the simple example given in Fig. 6.1.

When the order among siblings is taken into consideration, some subtrees that are actually automorphic would be considered to have different semantics. Sometimes this becomes too restrictive. For example, the following XML subtree fragment `<Book><Title:A/><Author:A/>` from Fig. 6.1 will be considered as a different subtree from subtree `<Book><Author:A/><Title:A/>`. Also, if we specify that the minimum support threshold is equal to 2, none of the above subtrees discovered from Fig. 6.1 would be frequent.

Often, we would like to relax the order among siblings such that the XML tree fragment `<Book><Author:A/><Title:A/>` (assume this is the representative form of automorphism group) would be considered equivalent with the XML tree fragment `<Book><Title:A/><Author:A/>`. With such an assumption, we would now find that the XML tree `<Book><Author:A/><Title:A/>` is frequent because its support count is ≥ minimum support specified, 2. Furthermore, sometimes we want to allow a certain degree of embedding to be present on the discovered subtrees. For example, we might want to consider the following XML subtree fragment `<Book><FirstName:X/><LastName:Y/>`, which is an embedded subtree not an induced subtree. If we consider order among siblings, a similar case will occur; that is, it becomes too restrictive such that the subtree `<Book><FirstName:X/><LastName:Y/>` becomes infrequent. Again, it is more logical to relax the order notion among the siblings so that the XML subtree fragment `<Book><FirstName:X/><LastName:Y/>` is considered equivalent to the other XML subtree fragment `<Book><LastName:Y/><FirstName:X/>`.

In the previous chapter, when we described the technique for ordered subtree mining, the order of sibling nodes is automatically preserved. This is because the candidates are enumerated as they are found in the tree database, and every variation of a subtree with respect to the order of its sibling nodes is considered as a separate candidate subtree. In unordered subtree mining, on the other hand, a subtree that occurs with different order among its sibling nodes needs to be considered as the same candidate. As mentioned in Chapter 2 and 3, this produces the problem of determining whether two trees are isomorphic to one another. Two trees $T_1$ and $T_2$ are isomorphic, denoted as $T_1 \cong T_2$, if there is a bijective correspondence between their node sets which preserves and reflects the structure of the trees. The group of possible trees obtained by permuting the sibling nodes in all possible ways is referred to as the automorphism group of a tree (Zaki 2005b). An automorphism group of a tree $T$, denoted as $Auto(T)$, is a complete set of label preserving automorphisms of $T$, i.e. $Auto(T) = \{T_1, \ldots, T_n\}$ where $T_i \cong T_j$, for any $i = (1, \ldots, n)$ and $j = (1, \ldots, n)$. During the pre-order traversal of a database, as is the approach in the TMG framework, ordered subtrees are generated by default. Hence, to mine unordered subtrees, it is necessary to identify which of these ordered subtrees form an automorphism group of an unordered subtree. One tree needs to be selected to uniquely represent the unordered trees in the automorphism group of unordered trees. This unique tree is referred to as the canonical form (CF) of an unordered tree. The problem of candidate generation is then to enumerate subtrees according to the canonical form so that the frequency of a candidate unordered subtree $T$, is correctly determined as the frequency of the members of $Auto(T)$. Hence, additional