Proposing a Classification Algorithm for User Identification According To User Web Log Analysis

Soodeh Peyvandi, Hossein M. Shirazi, Ahmad Faraahi

Student at the Department of Computer Science, Tehran, Iran.
Faculty of Information Technology, Malek-Ashtar University of Technology, Tehran, Iran.
Faculty of Information Technology, Payamenur University, Tehran, Iran.

Abstract: Request classification is one of important strategies of Web. User behavior analysis can make Web service more intelligent and secure. Recently, tree structures have become a popular way for storing and manipulating huge amount of data. The classification of these data can facilitate storage, retrieval, indexing, query answering and different processing operations. In this paper, we propose User-Classifier algorithm for rule based classification of tree structured data that to classify web log user. This algorithm is based on extracting special tree pattern from training dataset. Our experiments show that User-Classifier reduces running time. In the case of complete classification, User-Classifier shows the best classification quality.

Key words: Tree-structured data, structural classification, rule based classification, induce closed tree pattern.

INTRODUCTION

With the continuous growth of data in web, different knowledge discovery tasks have become increasingly important. In this paper we intend to focus on the classification task. This task can be defined as follow: Given a set of training data classified into some predefined categories, learn to automatically categorize new data. Classifying based on using only the content of trees ignores a significant amount of structural information hidden in the structure of trees. Therefore, recently (mainly from 2003) a growing interest has been emerged to develop new approaches which use the structural information of trees. (Zaki and Aggarwal, 2006) discussed the idea of constructing structural rules and proposed XRule algorithm for classification of XML documents. During the training phase, this algorithm finds the structures which are most related to the class variable. During the testing phase these structures are used to perform the structural classification. (Zaki and Aggarwal, 2006) showed that this classifier is significantly more effective than text classifiers due to its ability in use of distinguishing structures. For real datasets, the accuracy of XRule is 2–4% better than CBA (Li et al., 1998) and SVM (Joachims, 2002). Based on our best knowledge and to the time of writing this paper, XRule is the most effective algorithm for classification of tree structured data. In rule based classifiers (such as XRule), pattern extraction phase plays an important role on the accuracy and efficiency of the algorithm. Different types of tree patterns can be extracted from a forest of trees. In this paper, we improve XRule by using different types of patterns. We propose User-Classifier a structural classifier based on closed induced tree patterns. We show that in the complete classification User-Classifier gives the best classification quality compared to XRule. Furthermore, its running time and complexity are always less than XRule.

Related Workes:

When classifying tree structured data, based on type of the dataset, availability of information included in the dataset and behavior of the classifier, content information or information hidden in the structure can be used. Content-based classifiers: Several algorithms have been proposed to address the problem of text classification. In general, traditional text classification algorithms can be divided into following categories: rule based classifiers (Apte et al., 1994), decision trees (Fuhr et al., 1991), inductive learning algorithms (Cohen and Y. Singer, 1996), regression models (Yang and Chute, 1992), nearest neighbor classifiers (Creecy et al., 1992), Bayesian belief network based classifiers (Tzeras and Hartmann, 1993) and neural network-based classifiers (Wiener et al., 2005). Content-based algorithms consider each data point as a bag of words and as a result they lose a lot of information hidden in the structure of the tree. Structure-based classifiers: These algorithms consider the structure of the trees and take advantage of the information hidden in the structure. In (Fuhr and Weikum, 2002) a classification technique using hierarchical taxonomies was extended. Their algorithm follows a two-step process, called focused crawling, that interleave crawling and classification. A belief network-based generative Bayesian model was presented in (Denoyer and Gallinari, 2003) to classify XML documents. Their solution considers each structured document as a Directed Acyclic Graph. The algorithm of (Candillier et al., 2005) considered each XML tree as a set of attributes-values. The attributes-values sets are constructed by relations like parent-child and next-sibling between the nodes of the input tree. Classification and clustering of these attributes-values are done by means of text classification algorithms. Among the structure-based classifiers, rule based approaches show better results. In these algorithms a set of structural rules are constructed during the training phase. These rules are based on extracting frequent tree patterns from the training data. In the test step, for each input tree, rule(s) whose antecedents are sub-structures of the input tree are selected and after a rule combination step, label of the input tree is determined. The most efficient rule based structural classifier is XRule (Zaki and Aggarwal, 2005). It is based on finding frequent embedded tree patterns from training dataset and assuming that the presence

Corresponding Author: Soodeh Peyvandi, Student at the Department of Computer Science, Tehran, Iran
E-mail: soodehpeyvandi@gmail.com
of a particular kind of structural patterns in an XML document is related to its probable belonging to a particular category. An important and challenging part of rule based structural classifiers which mainly affect the efficiency's extracting tree patterns from the training dataset. There are different algorithms for the problem of finding frequent tree patterns from a forest of trees. Zaki presented Tree Miner (2005) to mine embedded ordered frequent tree patterns. He used an efficient data structure called scope-list for frequency counting and proposed rightmost path extension to generate non-redundant candidates. Asai et al. (2002) independently proposed the rightmost candidate generation. They developed FreqT for mining frequent induced ordered tree patterns. Chi et al. (2003) proposed Free Tree Miner for mining induced unordered free trees. Other algorithms for mining induced unordered tree patterns are Path Join (Xiao et al., 2003). Chi et al. (2005) proposed CMTreeMiner for mining both closed and maximal frequent sub trees in a database of rooted unordered trees.

I. Structural Classification Algorithms Base On Closed Tree Patterns:

Figure 1 shows an overall view of our proposed systems User-Classifier for classification of tree-structured data. During the training step, User-Classifier at first mine tree patterns from the training dataset, then they build structural classification rule set. After constructing rules, they are sorted and undesirable rules are eliminated. The training phase of User-Classifier is done. In the testing phase, for each data point, User-Classifier choose a subset of rule set, called covering rule set, which are most related rules to the data point. Then based on the similarity between these rules and the data point as well as the confidence degree of the selected rules, label of the data point is predicted. In the rest of this section, we explain different steps of User-Classifier in details.

A. Training phase:

In the training phase, characteristic properties of the training dataset are extracted and based on them a unique description of each classification class is created. Following, we describe in details different operations performed by User-Classifier Identifier during the training step.

- Mining tree pattern:

In order to extract the features of the training dataset, we consider frequent tree patterns contained in the training dataset. Here, we propose use of closed induced tree patterns for User-Classifier to express the features of the training dataset. Closed tree patterns are compact and complete; their number is fewer than the number of all the frequent tree patterns and they do not miss any important information. Reducing number of rules without jeopardizing the accuracy of the classifier which is called rule pruning has a very important effect on the efficiency and effectiveness of the classification (Bayardo, 1997) and (Li et al., 2001). It is also very important to allow domain experts to adjust a classifier by editing its rule-set. Previous experiments show that manual alteration of rule-set can lead to significant improvement in the classification (Antonie, et al., 2003). Using closed tree patterns instead of all the frequent tree patterns implies examination of some techniques proposed for the rule pruning. For example, one rule pruning technique is applied when there are two or more contradictory rules. In these situations if two rules have the same confidence, the more specific rule (one that its precedence is super tree of another’s precedence) is preferred. For finding closed tree patterns, we use CMTreeMiner algorithm (Cohen and Singer, 1996).

CMTreeMiner is a computationally efficient algorithm that discovers only maximal or only closed tree patterns in a database of labeled rooted trees without prior finding of all the frequent tree patterns. This algorithm has two versions: CMOrderedTreeMiner for rooted ordered trees and CMUnorderedTreeMiner for rooted unordered trees. Here, we use the ordered version. The algorithm mines closed frequent sub-trees by traversing an enumeration tree that enumerates all the frequent sub-trees (enumeration tree first introduced in (Asai et al., 2003) and is a lattice improved for tree candidate generation). CMTreeMiner uses some techniques to prune the branches of the enumeration tree that do not correspond to closed frequent sub-trees. These techniques are as follow: Left-blanket pruning and right-blanket pruning. In this algorithm, several heuristic techniques are proposed to arrange the order of computation so that expensive computation is avoided as much as possible. We have done some alterations in CMTreeMiner to make it more suitable for classification rule generation. We consider a candidate as a frequent pattern if it is frequent for at least one class. Pattern p is said to be frequent in class C if its frequency in training trees belonging to C is greater than C’s minimum-support. It is possible that a pattern is frequent in more than one class. Furthermore, we keep a class index which shows the class label of each tree in the training dataset. This index is used for fast update of the frequency of a generated tree candidate per each class and to test whether or not it is frequent in each class. Each pattern P found by revised CMTreeMiner has three parts: string representation of P, frequency of P in the whole of the training dataset and an array in which cell i shows the frequency of P in trees belonging to class i. Figure 2 shows the format of found patterns from the training dataset.

Fig. 1: Components of the proposed structural classifiers.
where Con is the training dataset its structural rules are constructed. After mining closed tree patterns (in User-Classifier), for each class \( C_i \) of the training dataset its structural rules are constructed. We assign two values to each structural rule: support, confidence. For a structural rule in the form of frequent tree pattern \( \Rightarrow C_i \), the support is defined as the percentage of the trees in the training dataset which contain \( T \) and have class label \( C_i \). Confidence is defined as the ratio of the number of trees containing \( T \) and having class label \( C_i \) to the number of trees containing \( T \) in the entire of the training dataset.

Rule elimination:

For the rule frequent tree pattern \( \Rightarrow C_i \) if its confidence is equal to 0.5, this rule cannot distinguish between class \( C_i \) and \( C_i \)'s negative class. Therefore, this rule should be eliminated. If confidence of a rule is less than 0.5, it gives more evidence for not belonging the tree to the class. Therefore, we append a rule to classifier’s rule-set (and say it is acceptable) if its confidence is greater than 0.5.

Rule ordering:

An ordering among the classification rules combines rule-sets of classes based on a precedence relation and generates a totally ordered rule set for the classifier. Consider two structural rules \( R_1: T_1 \Rightarrow C_1 \) and \( R_2: T_2 \Rightarrow C_2 \). Assume that the support and the confidence for \( R_1 \) are \( s_1 \) and \( c_1 \), respectively; and those for \( R_2 \) are \( s_2 \) and \( c_2 \), respectively. In User-Classifier \( R_1 \) precedes \( R_2 \), if:

1. Confidence: The confidence of \( R_1 \) is greater than the confidence of \( R_2 \) \( (c_1 > c_2) \).
2. Support: two rules have equal confidences, but the support of \( R_1 \) is greater than the support of \( R_2 \) \( (s_1 > s_2) \).
3. Size of antecedent: two rules have equal confidences and equal supports, but \( T_1 \) is greater than \( T_2 \) \( (|T_1| > |T_2|) \), and
4. Lexicographical order of antecedent: two rules have equal confidences and equal supports and their antecedents have equal sizes, but \( T_1 \) is lexicographically before than \( T_2 \).

We use string’s lexicographical order to define a total lexicographical order of all labeled, ordered, and rooted trees. In the other words, we say that tree \( T_1 \) is lexicographically before than tree \( T_2 \) if and only if the \( T_1 \)'s string representation is lexicographically greater than or equal to the \( T_2 \)'s string representation. Several string representations of trees have been introduced by Zaki (2005), Chi et al. (2003;2004), Asai et al. (2003) and Nijssen et al. (2003). User-Classifier uses Zaki’s string representation. As described in [25], tree \( T \) can be represented uniquely by its string representation which is generated as follows: Add vertex labels of \( T \) to its string encoding \( S \) in a depth-first preorder traversal of \( T \) and add a unique symbol (e.g. “$”) when backtracking from a child to its direct parent. When comparing two string representations, we assume that all the labels of the trees are greater than the special symbol (“$”).

B. Testing phase:

The testing phase of our classification systems consists of two main operations: rule selection and rule combination.

Rule selection:

For each testing tree, User-Classifier scans the rule-set in order and performs a tree matching operation between the left hand of the rule (tree pattern) and the testing tree. For testing tree \( T \), if there is no acceptable covering rule and we want to determine \( T \)'s class, \( T \) is assigned to the class with the greatest number of members. Selected rules for a test tree are divided into classes. We note that during the rule generation step, for each rule, its confidence in each of the classes is determined. It is possible that a rule belongs to more than one class (convey a testing tree to more than one class).

---

**Fig. 2:** Format of patterns generated by revised CMTreeMiner.
**Rule combination:**

In this step, selected rules are combined and according to result of combination, label of the test trees determined. There are several possible methods for combining rules:

1. **Best Rule:** We select the best rule that covers the test tree. Since the rule-set is totally ordered, the best rule is the first one which covers the test tree.

2. **Best K Rules:** We select the best (first, if the rule-set is ordered) K rules that cover the test tree and then compute the average confidence of each class. For class i, its average confidence is defined as the average of confidences of rules belonging to it. If there is a class that its average confidence is greater than 0.5, the test tree is assigned to this class. Otherwise, the test tree is assigned to the class with the greatest number of members.

3. **All Rules:** We contribute all the selected rules in the averaging process. There is no overall best combination for all the data sets and all the algorithms. For unstructured data the “Best Rule” mechanism works better than the “Best K Rules” (Coenen and P. Leng, 2004). For User-Classifier, we use a revised form of “All Rules” mechanism because Zaki (Joachims, 2002) shows that “All Rules” is best way for tree-structured data.

**II. Experimental result:**

We performed extensive experiments to evaluate the accuracy and efficiency of the proposed algorithm using data from real applications. We did our experiments on a 2.8 GHz Intel Pentium IV PC with a 512 MB main memory, running UNIX operating system. All the algorithms were implemented in C++. Based on our best knowledge and to the time of writing this paper, XRule (Zaki and Aggarwal, 2006) is the most efficient algorithm for classification of tree-structured data. Therefore, we selected this algorithm for our comparisons.

**Dataset:**

We have used real datasets for our comparisons. Used dataset, which is called CSLOG, contains the web access trees of the CS department of the Rensselaer Polytechnic Institute during three weeks (Zaki, 2005). To convert this dataset to a classification dataset (Zaki and Aggarwal, 2006), chose to categorize each user-session into one of two class labels: “edu” corresponds to users coming from either an “edu” or an “ac” domain, while “other” class corresponds to users coming from other domains. The goal of a classification algorithm for this dataset can be to distinguish users who come from “edu” or “ac” versus other domains according to their browsing behavior within the CS web site. In this dataset, each week’s log-file is separated into a different data set. Therefore, there are three different dataset: CSLOG1 for the first week, CSLOG2 for the second week and CSLOG3 for the third week. Furthermore, combining CSLOG1 and CSLOG2 generates a new dataset called CSLOG12.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of trees</th>
<th>Number of trees in ‘edu’ class</th>
<th>Number of trees in ‘other’ class</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSLOG1</td>
<td>8074</td>
<td>1962</td>
<td>6112</td>
</tr>
<tr>
<td>CSLOG2</td>
<td>7407</td>
<td>1656</td>
<td>5751</td>
</tr>
<tr>
<td>CSLOG3</td>
<td>7628</td>
<td>1798</td>
<td>5830</td>
</tr>
<tr>
<td>CSLOG12</td>
<td>13934</td>
<td>2969</td>
<td>10965</td>
</tr>
</tbody>
</table>

**Evaluation criteria:**

Let class “edu” is the positive class and class “other” is the negative class. At the end of the testing phase four metrics will be obtained:

1. TP: number of positive cases that were correctly classified as positive.
2. TN: number of negative cases that were correctly classified as negative.
3. FP: number of negative cases that were wrongly classified as positive.
4. FN: number of positive cases that were wrongly classified as negative. There are two types of misclassification: classify a positive case as negative, or classify a negative case as positive, respectively represented as FN and FP. The accuracy of a classification algorithm on test dataset D is defined as the ratio of the number of correctly classified cases to the total number of predictions made, i.e.

$$ A = \frac{TP + TN}{TP + FP + TN + FN} $$

(1)

**Empirical evaluation results:**

In this section we empirically study the effect of the improvements applied on XRule. In our experiments, we have considered all the possible combinations of the datasets as training dataset or testing dataset. We use the notation CSLOG x−y to denote that the classifier is trained on CSLOG x and is tested on CSLOG y. For example, CSLOG12-3 means that the classification algorithm learns from CSLOG12 and is tested on CSLOG3.

**Evaluation of the partial classification:**

An important factor on the number of the generated rules and as a result on the accuracy, complexity and running time of the classification is minimum-support. Determining best value for minimum-support needs some experiments. For each
training-testing datasets and for all the algorithms, we performed classification for different values of minimum-support in the range of \([0.01\%, 0.02\%, 0.03\%... 0.10\%]\) and then selected one resulting to the highest accuracy. If two values of minimum-support bring us the same accuracy, the greater minimum-support is selected. TABLE II summarizes values of the best minimum-support for the classifiers. We note that a greater value for the best minimum-support may have more interest owing to it directly affects runtime and complexity of classification. Values of the best minimum-support for User-Classifier and XRule are always 0.01\%. In this paper, we evaluate 2-class classification problem and consider equal values for the minimum-supports of two classes.

**Table 2:** Values of the best minimum-support for user-classifier and xrule for different training-testing datasets (in the partial classification)

<table>
<thead>
<tr>
<th>Training-testing dataset</th>
<th>User Classifier</th>
<th>XRule</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSLOG1-2</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG3-2</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG1-12</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG2-1</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG2-3</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG2-12</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG3-1</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG3-2</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG3-12</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG12-1</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG12-2</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
<tr>
<td>CSLOG12-3</td>
<td>0.01%</td>
<td>0.01%</td>
</tr>
</tbody>
</table>

Figure 3 shows the accuracy of User-Classifier and XRule on the different training-testing datasets. Values of accuracy for User-Classifier and XRule are between 0.82\% and 0.84\%.

Figure 4 shows the time of extracting frequent tree patterns from the training dataset. During this period, User-Classifier mines induced closed tree patterns and XRule extracts embedded frequent tree patterns. User-Classifier is faster than XRule since it only extracts closed tree patterns. We note that for all classifiers the pattern extraction time is tiny and it can be ignored compared to the training step time and the test step time. Figure 5 depicts the running time of the training steps of the classifiers for the different training-testing datasets. During this time, operations such as rule induction, rule elimination and rule ordering. For User-Classifier and XRule this time is considerable and User-Classifier learns faster than XRule. In Figure 6, running times of the testing steps of the algorithms for the different training-testing datasets are given. During this time, for each testing tree a subset of rules is selected and combined and the label of the input tree is determined. Similar to the training step, User-Classifier faster than XRule. Figure 7 shows the total running time of the classifiers including the pattern extraction time, the training time and the testing time. User-Classifier and XRule is the most time consuming one. Total running times of User-Classifier are between 59.9393 seconds (for CSLOG1-2) and 83.596 seconds (for CSLOG3-12) and total running times of XRule are between 59.857 seconds (for CSLOG2-3) and 112.4614 seconds (for CSLOG12-1).

Complexity can be measured in the terms of numbers of rules, where smaller rule sets that are reasonably close to the best accuracy are sometimes preferred to more complex rules sets with greater accuracy (Apte et al., 1994). In relational databases large item-sets are approximately 3–4 times more than maximal item-sets. In tree structured data it is possible that a set of nodes form different trees while all the trees are the induced sub tree of a specific super tree.

**Table III** summarizes complexities of User-Classifier and XRule for the different training-testing datasets. Difference among the complexities of User-Classifier and XRule is less than the difference among the number of induced and frequent tree patterns. The reason is that frequent tree patterns (which are used by XRule) generate more unacceptable rules which are ignored during the rule elimination step.

Figure 8 shows the complexities of User-Classifier and XRule for the different training-testing datasets.
Fig. 4: Comparison of running time of the two pattern extraction phase for the different classification algorithms.

Fig. 5: Comparison of running time of the training step of User-Classifier and XRule over the different training-testing datasets.

Fig. 6: Comparison of running time of the testing step of User-Classifier and XRule over the different training-testing datasets.

Fig. 7: Comparison of total running time of User-Classifier and XRule over the different training-testing datasets.
Conclusion:

In this paper, we proposed efficient and accurate algorithm for classification of tree-structured data, called User-Classifier. In order to create structural classification rules, User-Classifier mines induced closed tree patterns from the training dataset. Values of accuracy for User-Classifier and XRule are between 0.82% and 0.84% However User-Classifier is faster than XRule since it only extracts closed tree pattern. Difference among the complexities of User-Classifier and XRule is less than the difference among the number of induced and frequent tree patterns. The reason is that frequent tree patterns (which are used by XRule) generate more unacceptable rules which are ignored during the rule elimination step.

REFERENCES


Liu, B., W. Hsu and Y. Ma, 1998. Integrating classification and association rule mining, ACM International Conference on Knowledge Discovery and Data Mining.


