Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Improved Classification Based on Predictive Association Rules

Zhixin Hao, Xuan Wang, Lin Yao, Yaoyun Zhang
Intelligence Computing Research Center, Shenzhen Graduate School
Harbin Institute of Technology
Shenzhen, China
hzx1984@gmail.com, wangxuan@insun.hit.edu.cn, yaolin@hit.edu.cn, xiaoni5122@gmail.com

***Abstract*—Classification based on predictive association rules (CPAR) is a kind of association classification methods which combines the advantages of both associative classification and traditional rule-based classification. For rule generation, CPAR is more efficient than traditional rule-based classification because much repeated calculation is avoided and multiple literals can be selected to generate multiple rules simultaneously. Despite these advantages above in rule generation, the prediction processes have the weaknesses of class rule distribution imbalance and interruption of incorrect class rules. Further, it is useless to instances satisfying no rules. To tackle these problems, this paper presents Class Weighting Adjustment, Center Vector-based Pre-classification and Post-processing with Support Vector Machine. Experiments on Chinese text classification corpus TanCorp show that our algorithm achieves an average improvement of 5.91% on F1 score compared with CPAR.**

Keywords*—CPAR, Class Weighting Adjustment, Center Vector-based Pre-classification, Support Vector Machine

I. INTRODUCTION

Classification methods based on association rules have grasped much more attention of researchers nowadays. According to several reports, higher classification accuracy has been achieved than traditional classification approaches e.g. C4.5, FOIL and RIPPER [1]. It can make prediction from example with unknown labels by mining association rules in training corpus. In general, three steps are contained in association rules classification [2]: (1).Rule Generation: extract candidate rules in the training corpus which satisfy minimum support predefined with some data mining algorithm. (2). Rule Selection: evaluate all candidate rules, only the rules satisfying confidence predefined can be retained. (3). Classification: choose the best rule from classifier for prediction.

Most traditional association classification algorithms only have concerned about selecting literals through frequently items. The attributes of a rule only depends on the minimum support predefined and their classification abilities are ignored. The threshold of minimum support plays a very important role. However, these approaches are time-consuming and it is difficult to select high quality rules. CPAR has combined the advantages of both associative classification and traditional rule-based classification. It employs Predictive Rule Mining algorithm to generate rules directly from training corpus. In this way, CPAR generates much smaller set of high-quality predictive rules and generates all the rules according to the rules generated before to avoid redundant rule generation. For selecting literals, instead of selecting only the best literal for the rule, CPAR selects all the literals close to the best one thus generates more useful rules [3].

The rule generation of CPAR makes it a more efficient associative classification algorithm. But, there are mainly three flaws in the processes of rule evaluation and classification. For each of them a solution is proposed as follows: (1) The number of rules of each class has imbalanced distribution. It may range from several dozen to several thousand. Therefore, an example is predicted more easily to the class with more rules than the fewer ones. So Class Weighting Adjustment is proposed which will balance the intensity of classification rules by adjusting weight factor iteratively, the effect of strong classes and weak ones will be more balanced. (2) In the stage of classification, each class is treated uniformly for examples which will increase the probability of mis-classification. Before classification, similarity computation between example and the center vector of each class is proposed. We only load the rules of a class whose similarity of center vector to the example is greater than average similarity. Similarity computation is based on vector space model. (3) CPAR is useless for the instances satisfying no rules. Experimental data of TanCorp shows that there are about 4% testing instances satisfy no rules. Post-processing with Support Vector Machine (SVM) is proposed to predict these examples [4].

The paper is organized as follows. Section 2 describes the general ideas of classification based on predictive association rules. In Section 3, we discuss our Improved CPAR (ICPAR) algorithm in detail. Section 4 presents the experimental result of ICPAR and make comparisons with CPAR, KNN, and SVM. The concluding remarks are presented in Section 5.

II. CLASSIFICATION BASED ON PREDICTIVE ASSOCIATION RULES

CPAR is an association classification algorithm which combines the advantages of both associative classification and traditional rule-based classification. A greedy algorithm Predictive Rule Mining is used to generate rules directly from training data. In order to avoid over fitting, CPAR employs expected accuracy to evaluate rule and best k rules of each class are used for prediction. There are three main steps in CPAR which are given as follows [3]:

A. Rule Generation

Some definitions are first given as follows:

Literal: A literal p is an attribute-value pair in the form of (A_i, v), where A_i is an attribute and v is a value. If and only if $t_i = v$, tuple t will satisfy literal p, t_i is the value of i^{th} attribute of t. In rule generation algorithm of CPAR, every document is represented by a set of terms and there is no weight for each term. So we make no difference in literal and term in this paper.

Rule: A rule r is in the form of $p_1 \wedge p_2 \wedge ... \wedge p_l \rightarrow c$, p_i is a literal and c is a class label. When a tuple t satisfies all the literals of rule r, we say that tuple t satisfies rule r. The tuple t can be predicted to class c.

Given a rule r:$P_r \rightarrow C_r$. Positive examples are examples satisfy P_r and belonging to class C_r. Negative examples are examples satisfy P_r and not belonging to class C_r.

The gain of literal p is defined as (1). P and N are sets of positive and negative examples respectively. |P| and |N| are number of examples in respective sets P and N. After literal p is added to current rule r, there will be $|P^*|$ positive and $|N^*|$ negative examples satisfying the new rule's body.

$$gain(p) = |P^*| (\log\frac{|P^*|}{|P^*|+|N^*|} - \log\frac{|P|}{|P|+|N|}) \quad (1)$$

For an iteration of rule generation of CPAR, literal with best gain is selected to current r. A rule is generated until the gain value of the literal is smaller than predefined threshold. The most time-consuming part of traditional FOIL algorithm lies in computing gain of every literal when selecting the best literal [5]. CPAR uses PNArray to enhance the efficiency of the computations.

If an example is correctly covered by a rule, CPAR decreases its weight instead of removing it from Foil and CPAR gets more rules. Moreover, CPAR selects not only the best literal but also the literals getting gain closer to the best one. It can get multiple rules simultaneously and concrete rule generation algorithm can be found in [3].

B. Rule Evaluation

Laplace expected error estimation is used in CPAR to estimate the excepted accuracy of rules which belonging to class c .

$$LaplaceAccuracy = (n_c + 1)/(n_{tot} + k) \quad (2)$$

In equation (2), k is the total class number, n_{tot} is the number of examples which satisfy the rule's body and in which there are n_c examples belonging to class c .

C. Classification

For each example, CPAR select the best k rules of each class which satisfying it. Then compute the average accuracy of the k rules and predict the example to the class with the highest average accuracy.

III. Improved CPAR

To overcome the weaknesses of CPAR described in Section 1, Class Weighting Adjustment, Center Vector-based Pre-classification and Post-processing with SVM are proposed. The rest of this section will discuss each approach in detail.

A. Class Weighting Adjustment

The processing of rule evaluation in CPAR only considers about the expected accuracy of each rule without taking in account the weight of every class. Table I lists the number of rules in each class generated by rule generation algorithm of CPAR on training corpus of TanCorp. Further, Table I depicts that the class rule numbers are unbalanced. Therefore, the example will be easily classified into class with more rule number.

TABLE I. Rule number of each class

Recruitment	Sports	Health	Region	Entertainment	Estate
4216	385	3270	29	27	936
Education	**Automobile**	**Computer**	**Technique**	**Art**	**Economic**
442	224	2140	605	205	1163

In order to avoid this, a Class Weighting Adjustment algorithm is used which has been presented in [6]. Some definitions are given as follows.

Class Rule Intensity: intensity of class c_i is defined as (3).

$$\rho_t(c_i) = \varepsilon_1^t(c_i) + \varepsilon_2^t(c_i) \quad (0 \le \rho \le 2) \quad (3)$$

ε_1 is the probability of examples wrongly classified into c_i . ε_2 is the probability of examples correctly classified into c_i . The stronger class rule intensity is, the larger ρ value of the class will be.

Class Weight Vector: $\overline{W_t} = [w_1^t, w_2^t, ..., w_m^t]$ where $\sum_{i=1}^{m}(w_i^t \times n_i) = n$. $\overline{W_t}$ is a weight vector before the first t times weighting adjustment. w_i^t is the weight of class c_i and n_i is the number of rules of class c_i . The total number of rules is n.

Initially, weight vector is set to $[1,1,...,1]$. For single iteration, the weighting factor for every class is defined as (4). New class weight w_i^{t+1} of class c_i will be computed as (5).

$$\alpha_t(c_i) = 2 - \rho_t(c_i) \quad (4)$$

$$w_i^{t+1} = w_i^t \times a_t(c_i) \quad (5)$$

Because new class weight must satisfy the condition $\sum_{i=1}^{m} (w_i^{t+1} \times n_i) = n$, so the normalize expression of w_i^{t+1} is computed as (6).

$$w_i^{t+1} = \frac{w_i^t \times \alpha_t(c_i) \times n}{Z^t} \tag{6}$$

$$Z^t = \sum_{i=1}^{m} w_i^t \times \alpha_t(c_i) \times n_i \tag{7}$$

The algorithm repeats iterations until all the classes rule intensity are equally same or reach the iteration time predetermined. Further more, the Class Weighting Adjustment algorithm is presented in Fig.1.

```
Input: the rules of each class and the train set T
Output: class weight of each class
Procedure Class Weighting Adjustment Algorithm
1   set the weight of each class to 1 in classweight
2   n← total rule number of each class
3   iteration time t←0
4   while t++ < iteration time predefined
5       make prediction with classification algorithm of
6       CPAR for T and record e1, e2 of each class
7       Z←0
8       for each class c in all classes
9           p(c) ← e1(c) + e2(c)
10          a(c) ← 2 - p(c)
11          classweight(c) *= a(c)
12          Z += classweight(c) * class_rule_number(c)
13      end
14      for each class c in all classes
15          classweight(c) = classweight(c) * n / Z
16      end
17  end
18  return classweight
```

Figure 1. Class Weight Algorithm

B. Center Vector-Based Pre-classification

For prediction, the classification in CPAR directly loads the rules of each class. Classification result will be easily affected by incorrect class. We can use a statistic for an example belonging to a certain class to test the inference i.e. the similarity between the example and the class is larger than average similarity of the example with all classes. There are 97.43% documents in TanCorp satisfying this inference. Before loading class rules, we can compute the similarity between example with each class and average similarity through all classes. Only the rules of class whose similarity with example is larger than average similarity are used. This method can reduce the probability of incorrect class interruption.

During similarity computation, Vector Space Model [7] is used for document representation. A document d_i is made up by a set of feature items $(t_{i1}, t_{i2}, ..., t_{im})$ and the weights correspond to them $(w_{i1}, w_{i2}, ..., w_{im})$. There are several approaches for weight computation; TF-IDF is one of them. In equation (8), TF_{ik} is the frequency of t_{ik} turning up in document d_i. There are N documents in training corpus. The number of documents which contain feature item t_{ik} is denoted by N_k.

$$w_{ik} = TFIDF(t_{ik}) = TF_{ik} * \log \frac{N}{N_k} \tag{8}$$

Feature selection is a technology for feature dimensional reduction. It can simplify computation and avoid over fitting [8]. Chi-square statistics is used here. It can measure correlation degree of term t and class c. Chi-square statistics assumes that t and c have a known distribution. This distribution is denoted by χ^2 -distribution. The correlation degree of t and c will be higher when they have a bigger χ^2 value. The formula of χ^2 is shown in (9).

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C)(B+D)(A+B)(C+D)} \tag{9}$$

N is the total document number in training corpus, A is the number of documents which belong to class c and contain t. B is the number of documents not belonging to c but containing t. C is the number of documents belonging to c without t. D is the number of documents not belonging to c and without word t. The Chi-square value for term t can be computed as described in (10). M is the total class number. After feature selection, the terms will be removed whose Chi-square values are smaller than threshold predefined. Moreover, the terms whose Chi-square values are larger than threshold will be retained as document feature.

$$\chi^2_{\max}(t) = \max_{i=1}^{M} \chi^2(t, c_i) \tag{10}$$

Up till now, the representation can be given to any document. For a class, we can sum all the documents VSM belonging to it and compute the average center vector of the class. Towards document d_i and center vector c_j of class j, similarity between them can be computed as explained in (11).

$$sim(d_i, c_j) = \frac{\sum_{k=1}^{M} w_{ik} \times w_{jk}}{\sqrt{\sum_{k=1}^{M} w_{ik}^2} \times \sqrt{\sum_{k=1}^{M} w_{jk}^2}} \tag{11}$$

C. Post-Processing with SVM

After classification with the class rules, there may still have some test examples satisfying no rule. Table II shows the probability of this kind of documents in test corpus of TanCorp. CPAR is not applicable for these documents. Under this condition, Post-processing with SVM is proposed for the purpose of classification.

TABLE II. PROBABILITY OF EXAMPLES BELONGING TO EACH CLASS WITH NO RULE SATISFYING

Recruitment	Sports	Health	Region	Entertainment	Estate
2.5%	2.2%	2.9%	0	2.7%	7.5%
Education	**Automobile**	**Computer**	**Technique**	**Art**	**Economic**
6.8%	1.7%	3.1%	8.7%	0.9%	2.5%

SVM is a pattern recognition approach based on statistical learning theory firstly proposed by Vapnik et al in 1995[9]. It's a universal learning approach developed from theory of VC-Dimension and idea of structural risk minimization. In order to get greatest generalization ability, SVM finds an optimal tradeoff between the complexity of the model and learning ability according to limited sample information. The basic thought of SVM is described as follows.

Suppose some given data points belong to one of the two classes and it is to decide that to which class a new data point might belong. In SVM, a data point is viewed as a p-dimensional vector and what needs to be found is that whether there is a (p-1)-dimensional hyperplane which can separate them. Moreover, the nearest distance between a point in one separating hyperplane and a point in the other separating hyperplane is maximized. It's known as maximum-margin hyperplane.

Suppose there are N data points, every data point can be written as (x_i, y_i) ($i = 1,2,...,N$ $x \in R^n$ $y \in \{-1,1\}$). Any hyperplane can be written as $wx - b = 0$. In model training of SVM, parameter w and b is chosen to maximize the margin following the constraint for each i , $y_i(wx_i + b) \geq 1 (y_i = 1)$ and $y_i(wx_i + b) \leq -1 (y_i = -1)$ .The two equations can be generalize as (12).

$$y_i(wx_i + b) \geq 1, i = 1,2,...,N \tag{12}$$

According to the rule of geometry, the distance between these two separating hyperplanes is $2/\|w\|$. Maximizing it is equally to minimize $\|w\|^2/2$. The separating hyperplane satisfying (12) and minimizing $\|w\|^2/2$ is called optimal separating hyperplane. Using Lagrange Multiplier can get an equivalent problem to maximize (13). In optimal separating hyperplane, appropriate kernel function $K(x_i, x_j)$ can be selected to complete linear classification after some non-linear transformation. Corresponding classification function is given below in (14). b^* is the threshold of classification. If $f(x) > 0$, x will be predicted to the class, otherwise will not.

$$L_D = \sum_i a_i = \frac{1}{2}\sum_{i,j} a_i a_j y_i y_j (x_i, x_j) \tag{13}$$

$$f(x) = \mathrm{sgn}(\sum_{i=1}^{n} a_i^* y_i K(x_i, x) + b^*) \tag{14}$$

It has been recognized that SVM worked well for text classification. There are mainly four reasons of using SVM for post-processing of CPAR [10]. (1).SVM doesn't depend on the number of features because of its application of over fitting protection. Large feature spaces can be handled in SVM. (2). There are only few irrelevant features in text classification. Feature selection may result in a loss of information and SVM can combine many features which are used for classification. (3). The document vector of text classification only contains few items of value not equal to zero. SVM is well suited for problems with sparse document vector. (4). Most text classification problems are linearly separable and SVM is suitable for finding such linear separator.

In addition, ICPAR algorithm is defined in Fig.2.

```
Input: rule set generated by rule generation algorithm of CPAR
       Center Vector of each class
       SVM model trained by training corpus
       Testing corpus
Output: class labels predicted for each example in testing corpus
Procedure ICPAR Algorithm
1   compute the weight of each class using class weighting adjustment
2   for each example t in testing corpus
3       v←vsm of t
4       avg_sim←average similarity of v with all the
                classes′ center vector
5       max_except←0 max_except_label←-1
6       for each class c in all classes
7           if the similarity of c′ s center vector with v
.           is larger than avg_sim
8               rulec←the rules of c
9               satisfy_rulec←the rules of rulec satisfying example t
10              sumc←sum up the best k rules′ expected accuracy
                in satisfy_rulec
11              expectc←sumc*class weight of c
12              if expectc>max_expect
13                  max_expect←expectc
14                  max_expect_label←c
15      end
16      if max_expect>0
17          predict example t into class max_expect_label
18      else
19          predict example t with post-processing svm model
20  end
```

Figure 2. Algorithm of ICPAR

IV. EXPERIMENTAL RESULTS

Our experiments are conducted on PC with Windows XP operating system, primary frequency 3.0G cpu, 2.0G primary memory. Chinese text classification corpus TanCorp V1.0 supplied by Institute of Computing Technology, Chinese Academy of Science is used for experiment [11]. There are 12 classes in TanCorp and the document number of each class is listed in Table III. In order to avoid over fitting, 5-fold cross validation is employed.

TABLE III. DOCUMENT NUMBER OF EACH CLASS

Recruitment	Sports	Health	Region	Entertainment	Estate
605	2305	1405	150	1500	935
Education	**Automobile**	**Computer**	**Technique**	**Art**	**Economic**
805	590	2940	1040	545	815

In rule generation of CPAR, we set threshold of positive weight factor δ to 0.04, threshold of literal gain to 0.6, weight decay factor α to 2/3 and min_gain_ratio (the threshold ratio of a literal's gain to the best literal's gain, the literal will be selected when the ratio of its gain to the best literal's gain is larger than it) to 0.8. CPAR select the best k rules of each class for classification, Fig.3 shows accuracy of different k in range from 1 to 10. When k equals to 5, we got a highest accuracy. So the best 5 rules are used in CPAR. In Class Weighting Adjustment algorithm, the iteration time is set to 5. In addition, our SVM implementation is based on LibSVM [12].

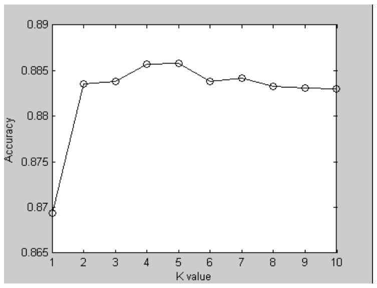

Figure 3. Different accuracy of different k value

Precision and Recall are widely used in statistical classifications. To a certain class, A denotes the number of examples classified into the class correctly. B denotes the example number misclassified to the class. C denotes the number of examples which belong to the class but classified to another class. The precision and recall are defined as follow.

Precision= $A/(A+B)$ if $(A+B)>0$; otherwise Precision=1

Recall= $A/(A+C)$ if $(A+C)>0$; otherwise Recall=1

A popular measure that combines Precision and Recall is F-score which is given in (15). β is a parameter which can adjust weights for precision and recall. When β =1, it is known as F1 measure. The overall performance on the whole data set is evaluated by macro average which is the arithmetic average of each class' performance index.

$$F_\beta = (1+\beta^2)\cdot(precision\cdot recall)/(\beta^2 precision + recall) \tag{15}$$

$$F1 = 2\cdot(precision\cdot recall)/(precision + recall) \tag{16}$$

Table IV illuminates that the F1-Meassue of macro average is increased by adding Class Weighting Adjustment (CWA), Center Vector-based Pre-classification (CVP) and Post-processing with SVM (PSVM) to CPAR in sequence. Further, Table IV depicts our improvements towards CPAR.

TABLE IV. MACRO_AVERAGE PERFORMANCE INDEX OF DIFFERENT IMPROVEMENTS

	CPAR	**CWA added**	**CVP added**	**PSVM added**
Macro_precision	86.53%	86.93%	86.79%	91.85%
Macro_recall	85.30%	87.51%	89.13%	91.26%
Macro_F1	85.22%	86.58%	87.26%	91.13%

From Fig.4 it can be seen that all of CWA, CVP and PSVM have increased the F1 values of most classes. Through data analysis, during each improvement for every class among them, the classification accuracy of examples belonging to the class is increased. Moreover, the probability of examples belonging to other classes misclassified to the class is decreased. Therefore, both recall and precision of the class are increased. So the F1 value of the class is increased. But exceptions do exist.

Class label 0 represents the class "Recruitment", which has the largest number of rules. Therefore, examples belonging to other classes can be easily misclassified to it. That is why the precision of "Recruitment" is lowest of all classes. Even though CWA can decline this influence, it can also reduce the recall of the class because the larger number of rules a class have, the lower weight of the class will be. So the classification intensity of the class is weakened by CWA. This causes some examples belonging to class "Recruitment" are misclassified to other classes. Therefore, from Fig.4 we can see that the F1 value of the class "Recruitment" is reduced by adding CWA.

Moreover, the F1 value of class "Recruitment" is lowest in all classes. From error analysis we find that there are about 10.41% examples of "Recruitment" misclassified to the class "Computer" and also some others misclassified to the class "Education" or "Economic". This is the flaw of the classification algorithm based on rules. For instance, it is reasonable that an example belonging to "Recruitment" contains the words of "internet", "electronic" and "company". But the three word can just make a rule of class "Computer" i.e. internet $\wedge$ electronic $\wedge$ company $\rightarrow$ Computer. When there are enough rules of this kind, the example is misclassified to class "Computer". Moreover, it reduces the precision of Class "Computer". Clearly, the example of Recruitment has correlation with example of "Computer", "Education" or "Economic". Therefore, mis-classification is unavoidable in this situation. In order to improve classification performance, a modified rule generation algorithm is needed which is our future research direction.

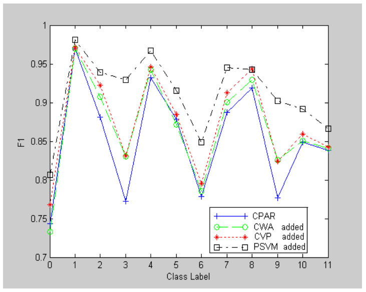

Figure 4. F1 comparison on each class adding each imprvement to CPAR

Text classification approaches KNN [13], SVM [12] and CPAR [3] are used to compare with ICPAR. Figs. 5, 6 and 7 given below show the comparisons of Precision, Recall and F1 in four different approaches on each class respectively. We can see that ICPAR gives the highest average performance in each competition.

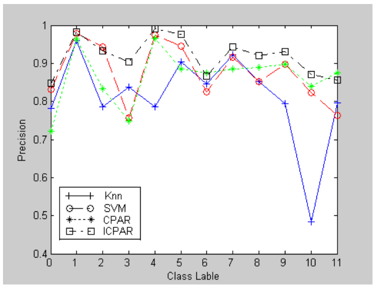

Figure 5. Precision comparison in different approach on each class

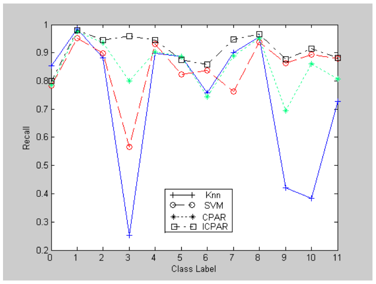

Figure 6. Recall comparison in different approach on each class

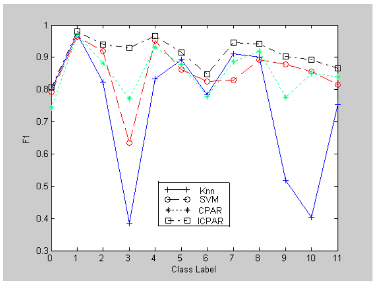

Figure 7. F1 comparison in different approaches on each class

Table V shows the results of KNN, SVM, CPAR and ICPAR. Further, Table V depicts that ICPAR has the highest classification performance in four classification approaches. We can also conclude that the classification performance of CPAR with SVM is more efficient than CPAR or SVM alone. Because of the weakness of rule based classification, CPAR is inadequate to examples satisfying no rule. Based on statistical learning theory, SVM can weight each single feature term in the process of model training. In this regard, SVM is superior to CPAR in generalization ability for new instances. The integration of CPAR and SVM in different ways to get better classification performance will be our future research direction.

TABLE V. COMPARISON OF DIFFERE NT MACRO_AVERAGE PERFORMANCE INDEX

	Knn	**SVM**	**CPAR**	**ICPAR**
Macro_precision	81.25%	87.56%	86.53%	91.85%
Macro_recall	74.22%	84.41%	85.30%	91.26%
Macro_F1	74.80%	85.17%	85.22%	91.13%

V. CONCLUSIONS

In this paper, we have presented that ICPAR achieved a higher accuracy than CPAR. Further, the proposed ICPAR has the following distinguished features: (1) It adjusts the weight of each class using Class Weighting Adjustment algorithm which balances the classifying ability of each class. (2) Instead of loading the rules of all classes, Center Vector-based Pre-classification selected and loaded only those rules of classes having high level of concern with the example. Therefore, the probability of incorrect classifying for the example has been reduced. (3) Post-processing with SVM has been used for examples satisfying no rules. There is no way to classify this kind of examples in CPAR.

The proposed algorithm has been illustrated that it is efficient. The improvement in extracted rules' quality by combining the characteristics of Chinese text, as well as the elimination of the rules which may interrupt the correct classification result, will be our future research direction.

ACKNOWLEDGMENT

This work is supported by the National High-tech R&D Program of China (863 Program, No. 2007AA01Z194). And appreciate to Institute of Computing Technology, Chinese Academy of Science for the Chinese text classification corpus.

REFERENCES

[1] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In ICDM'01, pp. 369{376, San Jose, CA, Nov. 2001.

[2] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In KDD'98, pp. 80-86, New York, NY, Aug. 1998.

[3] Yin, X. and Han, J.: CPAR: Classification Based on Predictive Association Rules. Proc SIAM Int Conf on Data Mining (SDM'03), 2003, 331-335

[4] Rennie, J.D.M.[Jason D. M.], Rifkin, R.[Ryan], Improving Multiclass Text Classification with the Support Vector Machine, MIT AIM-2001-026, October 2001.

[5] J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In Proc. 1993 European Conf. Machine Learning, pp. 3{20, Vienna, Austria, 1993.

[6] Chen and Hu, Text Association Categorization Based on Self-Adaptive Weighting. Journal of Chinese Computer Systems. Vol.28, No.1.

[7] Liu Shao-hui et al. An Approach of Multi-hierarchy Text Classification Based on Vector Space Model. Journal of Chinese Information Processing. Vol.16, No.3.

[8] Yang Y M, Pedersen J. A comparative study on feature selection in text categorization. In: Proceedings of the 14th International Conference on Machine Learning (ICML97). San Francisco, USA: Morgan Kaufmann Publishers, 1997.412-420.

[9] C. Cortes and V. Vapnik, Support-Vector Networks, Machine Learning, 20(3):273-297.

[10] Joachims T. Text categorization with support vector machines[C]. Proc of European Conference on Machine Learning(ECML) . [S. l.] : [s. n.] ,1998.

[11] Songbo Tan et al. A Novel Refinement Approach for Text Categorization. ACM CIKM 2005.

[12] Chang C-C ; C-J Lin. C.-C. Chang and C.-J. Lin, LIBSVM: A library for support vector machines 2001. Software available at <u> http://www.csie.ntu.edu.tw/~cjlin/libsvm</u>.

[13] Oh-Woog Kwon, Jong-Hyoek Lee, Web page classification based on k-Nearest Neighbor approach.