Fundamenta Informaticae 76 (2007) 171–187
IOS Press

Rough set Based Ensemble Classifier for Web Page Classification

Suman Saha, C.A. Murthy* and Sankar K. Pal
Center for Soft Computing Research
Indian Statistical Institute, India
{*ssaha_r, murthy, sankar*}*@isical.ac.in*

Abstract. Combining the results of a number of individually trained classification systems to obtain a more accurate classifier is a widely used technique in pattern recognition. In this article, we have introduced a rough set based meta classifier to classify web pages. The proposed method consists of two parts. In the first part, the output of every individual classifier is considered for constructing a decision table. In the second part, rough set attribute reduction and rule generation processes are used on the decision table to construct a meta classifier. It has been shown that (1) the performance of the meta classifier is better than the performance of every constituent classifier and, (2) the meta classifier is optimal with respect to a quality measure defined in the article. Experimental studies show that the meta classifier improves accuracy of classification uniformly over some benchmark corpora and beats other ensemble approaches in accuracy by a decisive margin, thus demonstrating the theoretical results. Apart from this, it reduces the CPU load compared to other ensemble classification techniques by removing redundant classifiers from the combination.

Keywords: Text classification, Rough set, Meta classifier.

1. Introduction

1.1. The problem of web page classification

The World Wide Web contains an estimate of 11.5 billion indexable pages as reported in January 2005 by Google (http://www.google.com) and an estimate of 11 million or more pages being added daily. Describing and organizing this vast amount of content is essential for realizing the web as an effective information resource. Text classification has become an important process for helping web search engines to organize this vast amount of data. For instance, web directories, such as Dmoz (http://dmoz.org), Yahoo (http://www.yahoo.com) and Looksmart (http://www.looksmart.com), divide the indexed web documents into a number of categories for the users to limit the search scope. Moreover, text classification

*Address for correspondence: Center for Soft Computing Research, Indian Statistical Institute, Kolkata 700108, India

makes the results easier to browse. If the results returned by the search engine have been classified into a specified category, the users can choose the interesting category to continue browsing. Traditionally, text classification is performed manually by domain experts. However, human classification is unlikely to keep pace with the rate of growth of the web. Hence, as the web continues to increase, the importance of automatic web page classification becomes necessary. In addition, automatic classification is much cheaper and faster than human classification.

To make the text classification process automatic, machine learning techniques can be applied to generate classification models from a set of text documents with prelabeled categories. The classification model can then be used to automatically assign natural language texts to the predefined categories based on their contents. In order to apply a machine learning technique to web page classification, the following problems need to be solved. First, to build a web page classifier, we need to collect a set of web pages as training examples to train the machine learning system. These training examples should have predefined class labels. Second, the content of a web page in the training set should be analyzed and the page should be represented using a formalism that the learning system requires for representing training examples. A learner is first presented with training documents, each labeled as containing or not containing material relevant to a given topic; the label is denoted by c which can take values 1 or -1 (we can turn multi-topic problems into an ensemble of two-topic problems by building a yes/no classifier for each topic, this is standard). The learner processes the training documents, generally collecting term statistics and estimating various model parameters. Later, test instances are presented without the label, and the learner has to guess if each test document is or is not relevant to the given topic. Scalability and memory footprint can become critical issues as enormous training sets become increasingly available. Web directories contain millions of training instances which occupy tens of gigabytes, whereas even high-end servers are mostly limited to 5-8GB of RAM. Sampling down the training set hurts accuracy in such high dimensional regimes: every additional training document helps, and most features reveal some useful class information. Naive Bayes (NB), rule induction, decision trees and support vector machines (SVMs) are some of the best-known classifiers and bagging, Boosting, Stacking and ECOC are some well known classifier ensemble approaches employed to date [16, 4, 6, 20].

1.2. Popular ensemble approaches for text classification

Generally speaking, an ensemble approach involves two stages, namely model generation and model combination. In this subsection, we examine the model generation and model combination strategies in the popular ensemble approaches for the text classification.

Bagging involves a ”bootstrap” procedure for model generation: each model is generated over a subset of the training examples using random sampling with replacement (the sample size is equal to the size of the original training set). The model combination strategy for bagging is majority vote. Simple as it is, this strategy can reduce variance when combined with model generation strategies. Several studies on bagging have shown that it is effective in reducing classification errors [3].

Boosting is a general approach to improving the effectiveness of learning. Boosting has been the subject of both theoretical analysis and practical applications [16]. Unlike bagging, in which each model is generated independently, boosting forces the base classifier to focus on the misclassified examples in previous iterations. In this way, each new model can compensate for the weakness of previous models and thus correct the inductive bias gradually. Applying boosting to text categorization tasks, Schapire

and Singer evaluated AdaBoost on the benchmark corpus of Reuters news stories and obtained results comparable to Support Vector Machines and k-NN methods, which are among the top classifiers for text classification evaluation. Empirical studies on boosting and bagging show that while both approaches can substantially improve accuracy, boosting exhibits greater benefits. Therefore, we provide only the results of boosting in our comparative experiments.

Stacked generalization is a way of combining multiple models that have been learned for a classification task. Typically, different learning algorithms learn different models for the task at hand, and in the most common form of stacking the first step is to collect the output of each model into a new set of data. For each instance in the original training set, this data set represents every model's prediction of that instance's class, along with its true classification. During this step, care is taken to ensure that the models are formed from a batch of training data that does not include the instance in question, in just the same way as ordinary cross validation. The new data are treated as the data for another learning problem, and in the second step a learning algorithm is employed to solve this problem [7, 24, 20].

ECOC is an ensemble approach for solving multi-class categorization problems originally introduced by Dietterich and Bakiri [6]. It reduces a k-class classification problem into ensemble of binary classification problems and combines the predictions of those L classifiers using the nearest codeword (for example, by Hamming distance). The code matrix R defines how each sub-model is generated. There have been many code matrices proposed, such as Dense matrix and BCH codes. Recent work has demonstrated that ECOC offers improvement over the standard one-against-all method in text classification and provided theoretical evidence for the use of random codes [8].

1.3. Related work

Combining the results of a number of individually trained classifiers to obtain a better classifier is a technique that has been extensively researched for text mining, and shows considerable promise on many test sets, [5, 9, 10]. For many methods, such as Bagging, a large number of classifiers are combined. These are typically produced by an ensemble of identical classifiers, trained on different randomly chosen sets of instances [22]. Alternatively, the predictions of a smaller number of different types of classifiers trained on the same data may be combined. Research on combining text categorizers has mainly taken the latter route. This may be because the relatively large numbers of features and data sets used for text prohibit the training of many classifiers. Some approaches considered simple probability averaging strategies and more complex ways of combining the results of four text filtering techniques with different optimization and document representation schemes [17]. It was found that the simple strategies could improve on the best categorizer only to label documents. They were unable to estimate probabilities accurately and were consistently outperformed by the best single algorithm for a filtering application. The more complex strategies were less successful than the simple ones. Larkey and Croft report a consistent improvement in precision for linear combination of scores from pairs of classifiers in a medical domain, and a greater improvement for a three-way combination [9]. Recall only exceeds that of the better classifier for half the cases, but this is not unreasonable as it would generally be expected that gains in precision would come at the expense of recall. Li and Jain experimented with three different methods for combining the results of four typical classifiers: simple voting, and two methods for selecting the classifier with the highest local accuracy for a problem [10]. They found that "Combinations of multiple classifiers did not always improve the classification accuracy compared to the best individual classifier".

Scott reports selected breakeven results from a simple voting system made up of rule based classifiers that used different text representations, words, stemmed words, noun phrases etc. [18].

These suggest that performance can be improved over the best single categorizer. Finally Craven et al. tried combining votes from several variants of naive Bayes classifiers in a Web based application. They report that the combined classifiers were not uniformly better than their constituents [10].

1.4. Observations leading to our approach

Previous approaches to classifier combination have typically restricted the decision making process at the granular meta data label to different variants of voting approaches [21]. The information considered at the granular meta data level is not used extensively by the popular meta classifier systems to verify the combination or taking the decision with intelligent approaches [19]. Since a classifier rarely is the best choice across a whole domain, an intuitive alternative is to identify the document-specific context that differentiates between regions where a base classifier has higher or lower non-redundancy [12]. While ensembles provide very accurate classifiers, there are problems that may limit their practical application. One problem is the need for a large number of classifiers for achieving good performance. So an important line of research, therefore, is to find ways of converting these ensembles into less redundant representations. Second difficulty with ensemble classifiers is that an ensemble provides little insight into the correctness of the decision making process for the classification task.

In this paper, an approach named RSM(Rough Set Meta classifier) is proposed, which is designed to extract decision rules from trained classifiers ensemble that perform classification tasks. RSM utilizes trained classifier ensembles to generate a number of instances containing prediction made by individual classifier as condition attribute values and actual class as decision attribute value. Then RSM constructs a decision table with one instance in each row. Once the decision table is constructed rough set attribute reduction is performed to determine core and minimal reduct. The combination of classifiers corresponding to the features of minimal reduct are then taken to form classifier ensemble for RSM classifier system. Now from the minimal reduct obtained in the previous step we compute decision rules by finding mapping between decision attribute and condition attributes. These decision rules obtained by rough set technique are then used to perform classification task. Our approach tries to solve the problem of representing less redundant ensemble of classifies and the problem of making reasonable decision from the predictions of ensemble classifiers, by using rough set attribute reduction and rough set decision rule generation on a granular meta data generated by base classifiers from input data. In order to realize the specified objectives, the paper introduces rough set preliminaries in section 2. Section 3 presents mathematical framework to represent classification in rough set paradigm. Section 4 covers a new meta classifier termed as RSM. Finally, the performance of the RSM is reported in Section 5.

2. Preliminaries

2.1. Rough set

Rough set theory was developed by Zdzislaw Pawlak in the early 1980's [13]. It deals with the classificatory analysis of data tables. The data can be acquired from measurements or from human experts. The main goal of the rough set analysis is to synthesize approximation of concepts from the acquired data. We initially describe how synthesis takes place in an information system. In some instances, the aim

may be to gain insight into the problem at hand by analyzing the constructed model, i.e. the structure of the model is itself of interest. In other applications, the transparent and explainable features of the model may be of secondary importance and the main objective is to construct a classifier that classifies unseen objects well. An important feature of rough sets is that the theory is followed by practical implementations of toolkits that support interactive model development [23].

2.2. Information systems and indiscernibility

A complete information system expresses all the knowledge available about the objects being studied. More formally, an information system is a pair, $S = (U, A)$, where U is a non-empty finite set of objects called the universe and $A = \{a_1, a_2, ..., a_j\}$ is a non-empty finite set of attributes on U. With every attribute $a \in A$ we associate a set V_a such that $a : U \rightarrow V_a$. The set V_a is called the value set of a [13, 14]. This value set equates to the range of values associated with a specific variable. The data set U contained in the information system is used as the basis for the development of subsets of it that are "coarser" than U. As with any data analysis technique, details are lost, but the removal of details are controlled to uncover the underlying characteristics of the data. The technique works by, lowering the degree of precision in data, based on a rigorous mathematical theory. A core concept of rough sets theory is that of equivalence between objects (called indiscernibility). Objects in the information system about which we have the same knowledge form an equivalence relation. If $B \subset A$ there is an associated equivalence relation, $IND_A(B)$, called the B-indiscernibility relation. It is defined as: $IND_A(B) = \{(x, \acute{x}) \in U^2 \mid \forall a \in B, a(x) = a(\acute{x})\}$. $If (x, \acute{x}) \in IND_A(B)$, then the objects x and $\acute{x}$ are indiscernible from each other when considering the subset B of attributes. Equivalence relations lead to the universe being divided into partitions, which can then be used to build new subsets of the universe [15, 23].

2.3. Lower and upper approximations

Let $S = (U, A)$ be an information system, and let $B \subset A$ and $X \subset U$. We can describe the subset X using only the information contained in the attribute values from the subset B by constructing two subsets, referred to as the B-lower and B-upper approximations of X, and denoted as $B_*(X)$ and $B^*(X)$ respectively, where: $B_*(X) = \{x | [x]_B \in X\}$, where $[x]_B$ is an equivalence class corresponding to B and $B^*(X) = \{x | [x]_B \cap X \neq \phi\}$, where $[x]_B$ is an equivalence class corresponding to B. The lower approximation contains objects that are definitely in the subset X and the upper approximation contains objects that may or may not be in X. A third subset is also useful in analysis, the boundary region, which is the difference between the upper and lower approximations. This definition of a rough (approximate) set in terms of two other sets is contributed by Pawlak [13, 14, 15]. Any partition P of universe U defines an indiscernibility relation $IND(P) : xIND(P)y$ iff $(x, y \in X)$ for some $X \in P$. Let $P = \{P_1, P_2, ..., P_n\}$, $Q = \{Q_1, Q_2, ..., Q_m\}$ are partitions of U. We define the P-lower approximation of Q and the P-upper approximation of Q, respectively by $P_*Q = \{P_*Q_1, P_*Q_2, ..., P_*Q_m\}$ where $P_*Q_i = \{x \in U : x \in P_j \subseteq Q_i$ for some $P_j \in P\}$ for $i = 1, 2, ..., m$ $P^*Q = \{P^*Q_1, P^*Q_2, ..., P^*Q_m\}$ where $P^*Q_i = \{x \in U : x \in P_j$ and $P_j \cap Q_i \neq \phi$ for some $P_j \in P\}$ for $i = 1, 2, ..., m$.

2.4. Decision rules

To date, most of the published literature in rough sets has concentrated on a specific type of information system, referred to as a decision system. In a decision system, at least one of the attributes is a decision attribute. This decision attribute partitions the information system into concepts. The rule generation problem is expressed in rough set theory as finding mappings from the partitions induced by the equivalence relations in the condition attributes to the partitions induced by the equivalence relations in the decision attribute(s). These mappings are usually expressed in terms of decision rules. More formally we can associate a formal language L(S) with an information system S = (U, A). Expressions in this language are logical formulas built up from attributes and attribute-value pairs and standard logical connectives (Pawlak 1999). A decision rule in L is an expression $P \rightarrow Q$ (read if P then Q), where P and Q are respectively the conditions and decisions of the rule. Each rule can be assigned a confidence factor, which is the number of objects in the attribute subset that also satisfy the decision subset (concept), divided by the total number of objects in the attribute subset [2, 1].

3. Rough set view of classification results

In the problem of classification we train a learning algorithm and validate the trained algorithm. This task is performed, using some test-train split on a given categorized dataset. In the notion of rough set, let U be the given categorized dataset and $P = \{C_1, C_2, ..., C_k\}$ where $C_i \neq \phi$ for $i = 1, 2, 3..., k$, $\cup_{i=1}^{k} C_i = U$ and $C_i \cap C_j = \phi$ for $i \neq j$ and $i, j = 1, 2, 3..., k$ be a partition on U which determines given categories of U. Output of a classifier determines a new partition on U. This new partition is close to the given one with respect to some measure. In rough set terminology each class of the given partition is a given concept about dataset and output of classifiers determines new concepts about same dataset. Now given concepts can be expressed approximately by upper approximation and lower approximation constructed by generated concepts.

Example: Let $S = \{1, 2, 3, ..., 100\}$ be a set with a given partition $P = \{P_{01} = \{1, 2, ..., 20\}$,$P_{02} = \{21, 22, ..., 40\}$, $P_{03} = \{41, 42, ..., 60\}$, $P_{04} = \{61, 62, ..., 80\}$, $P_{05} = \{81, 82, ..., 100\}\}$

Let classifier C_1 generate a partition $P_1 = \{P_{11} = \{1, 2, ..., 10\}$, $P_{12} = \{21, 22, ..., 40\}$, $P_{13} = \{41, 42, ..., 60\}$, $P_{14} = \{61, 62, ..., 80\}$, $P_{15} = \{11, 12, ..., 20, 81, 82, ..., 100\}\}$, classifier C_2 generate a partition $P_2 = \{P_{21} = \{1, 2, ..., 20\}$, $P_{22} = \{21, 22, ..., 40\}$, $P_{23} = \{41, 42, ..., 60\}$, $P_{24} = \{61, 62, ..., 70\}$, $P_{25} = \{71, 72, ..., 80, 81, 82, ..., 100\}\}$, and classifier C_3 generate a partition $P_3 = \{P_{31} = \{1, 2, ..., 19\}$, $P_{32} = \{21, 22, ..., 39\}$, $P_{33} = \{41, 42, ..., 59\}$, $P_{34} = \{61, 62, ..., 79\}$, $P_{35} = \{20, 40, 60, 80, 81, 82, ..., 100\}\}$.

Misclassification rates of classifiers C_1, C_2 and C_3 are 10%, 10% and 5% respectively. The concepts of P has been represented in terms of lower approximation and upper approximation by other partitions, P_1, P_2, P_3, $P_1 \cup P_2$, $P_1 \cup P_3$ and $P_2 \cup P_3$ in Table 1.

Since combination of P_1 and P_2 can express the given partition accurately, we don't need to use any other partition with P_1 and P_2. If we have a case such as in the example, i.e., each set of P is defined by some partition P_i, where $i = 1, 2, 3$ then we can use this fact in object classification.

Let $P_1 = \{P_{11}, P_{12}, ..., P_{1k}\}, P_2 = \{P_{21}, P_{22}, ..., P_{2k}\}, ..., P_n = \{P_{n1}, P_{n2}, ..., P_{nk}\}$ be the Partitions generated by classifiers $c_1, c_2, ..., c_n$ on the given data set A. Let $SP = \{X_1, X_2, ...X_t\}$ be the super partition of $P_1, P_2, ..., P_n$.

Table 1. Expressing P by lower approximation and upper approximation of other partitions

	P_{01}	P_{02}	P_{03}	P_{04}	P_{05}
P_{1*}	P_{11}	P_{12}	P_{13}	P_{14}	ϕ
P_1^*	$P_{11} \cup P_{15}$	P_{12}	P_{13}	P_{14}	P_{15}
P_{2*}	P_{21}	P_{22}	P_{23}	P_{24}	ϕ
P_2^*	P_{21}	P_{22}	P_{23}	$P_{24} \cup P_{25}$	P_{25}
P_{3*}	P_{31}	P_{32}	P_{33}	P_{34}	ϕ
P_3^*	$P_{31} \cup P_{35}$	$P_{32} \cup P_{35}$	$P_{33} \cup P_{35}$	$P_{34} \cup P_{35}$	P_{35}
$(P_1 \cap P_2)_*$	P_{21}	P_{12}	P_{13}	P_{14}	$P_{15} \cap P_{25}$
$(P_1 \cap P_2)^*$	P_{21}	P_{12}	P_{13}	P_{14}	$P_{15} \cap P_{25}$
$(P_1 \cap P_3)_*$	P_{31}	P_{12}	P_{13}	P_{14}	ϕ
$(P_1 \cap P_3)^*$	$P_{11} \cup P_{15}$	P_{12}	P_{13}	P_{14}	$P_{15} \cap P_{35}$
$(P_2 \cap P_3)_*$	P_{21}	P_{22}	P_{23}	P_{34}	ϕ
$(P_2 \cap P_3)^*$	P_{21}	P_{22}	P_{23}	$P_{24} \cup P_{25}$	$P_{25} \cap P_{35}$

Now an ensemble classifier f is a function from SP to $P = \{C_1, C_2, ..., C_k\}$. It can be written as $f : SP \rightarrow P$ where $|SP| \leq n^k$ and $|P| = k$. We denote rough set based ensemble classifier as f_{RSM} and defined as:

$$f_{RSM}(x) = C_i \text{ if } |x \cap C_i| \geq |x \cap C_j| \ \forall \ j = 1, 2, 3..., k \text{ where } x \in SP \text{ and } C_i \in P \quad (1)$$

First we define a quality measure for the considered set of classifiers. Let us consider two partitions P, Q of U and a class of classifiers, i.e., functions $f : Q \rightarrow P$ (we assume $f(Q) = P$, i.e., classifiers are from Q onto P).

We define the error of f relative to P as:

$$ErP(f) = \sum_{x \in Q} \frac{|x|}{|U|} \Big(1 - \frac{|x \cap f(x)|}{|x|}\Big) = \frac{1}{|U|} \sum_{x \in Q} \big(|x| - |x \cap f(x)|\big) \quad (2)$$

We define optimality of f relative to P as:

$$ErP(f) \leq ErP(g) \ \forall \ g : Q \rightarrow P \quad (3)$$

Theorem 1: Rough set based ensemble classifier is an optimal classifier combination technique.

Proof: Let $u \in A$ and u corresponds $x \in SP$ Then error of f_{RSM} corresponding to u is $1 - \frac{|x \cap f_{RSM}(x)|}{|x|}$ To show that f_{RSM} is optimal, let $g : SP \rightarrow P$ be any other function. Let $x \in SP$ be arbitrary and

$f_{RSM}(x) = C_a$ and $g(x) = C_b$. By defination of f_{RSM} $|x \cap C_a| \geq |x \cap C_b|$. Therefore $\sum_{x \in SP} \frac{|x|}{|U|}(1 - \frac{|x \cap C_a|}{|x|}) \leq \sum_{x \in SP} \frac{|x|}{|U|}(1 - \frac{|x \cap C_b|}{|x|})$. Therefore $ErP(f_{RSM}) \leq ErP(f)$ i.e. f_{RSM} is optimal.

Theorem 2: The performance of the rough set based ensemble classifier is at least same as every one of its constituent single classifiers.

Proof: Let $P_r = \{P_{r1}, P_{r2}, ..., P_{rk}\}$ be a partition corresponding to a constituent classifier c_r. If c_r performs better than f_{RSM} then there exists a one one correspondence of P_r, partition corresponding to classifier c_r, and P, partition corresponding to the given categories. Let $h : P_r \rightarrow P$ be this correspondence. Since SP is a refinement of P_r, $H : SP \rightarrow P$ can be defined such that, for any $x \in SP$ x is a proper subset of only one P_{ri} and $H(x) = h(P_{ri})$. Now $ErP(c_r)$ is same as $ErP(H)$, (by definetion of H). But $ErP(H)$ can't be less than $ErP(f_{RSM})$. Therefore no constituent classifier perform better than f_{RSM}.

Remarks

Rough sets are used to select classifiers based on their ability to form a good combination independent of their individual accuracy. The prediction made by a constituent classifier about the category of data instances is not considered to make the decision but the way a constituent classifier makes the partition on dataset is the main consideration of the method. Usually, we deal with two kinds of partitions defined by any classifier: the partition defined on a given sample and the partition on the whole universe of objects (including unseen objects). The condition in the definition of f_{RSM} is expressed using the partition on a sample but in our inductive reasoning we assume that the condition is preserved on the partition of the whole universe too.

4. Rough set meta classifier (RSM)

Our approach named RSM is designed to extract decision rules from trained classifier ensembles that perform classification tasks. RSM utilizes trained ensembles to generate a number of instances consists of prediction of individual classifier as condition attribute value and actual class as decision attribute value. Then construct a decision table with one instance in each row. Once the decision table is constructed rough set attribute reduction is performed to determine core and minimal reduct. The classifiers corresponding to minimal reduct are then taken to form classifier ensemble for RSM classifier system. Now from the minimal reduct, we compute decision rules by finding mapping between decision attribute and condition attributes. These decision rules obtained by rough set technique are then used to perform classification task. Our approach tries to solve the problems, of representing less redundant ensemble of classifiers and making reasonable decision from the predictions of ensemble classifiers, by using rough set attribute reduction and rough set decision rule generation on a granular meta data generated by ensemble classifiers from input data.

Key idea of our algorithm is:

1. Redundancy removal from the generated model.

2. Decision rule generation from reduced model for classification of web documents.

4.1. Description of method

Model generation: We divide the data set U into three parts, namely train set, validate set and test set. To generate the initial classifier model for RSM we assumes a pool of base level classifiers and train them with train set of word vector representation of the documents. This trained classifiers ensemble is used by RSM to generate meta data for analysis.

Meta data generation: We require the outputs of the base classifiers, meta data, to train the meta classifier. To generate meta data from the given web documents we validate trained classifier ensembles on validation set. We call the predictions made by the classifier ensembles meta data because they are generated by trained classifier ensembles from input data. This meta data represented in the form of decision table is the input of rough set data analysis algorithm. Unlike word vector representation of web documents meta data has a simple brief format, where classifier in the ensemble contribute the existence of an attribute, values of this attribute can be any class level that is determined by the classifier at the time of validation. So the number of attributes is the same as number of classifiers and the number of objects is equal to the number of document validated by the base level classifiers.

Formation of decision table: Decision table U_1 = (C, D) consists of one instance in each row and columns contain document ID, value of condition attributes and value of decision attribute. For each instance validated in the previous step, we put instance number as document ID, predictions of base level classifies as values of condition attributes and actual class of the document as the value of decision attribute. That is, we are adding one more column, decision attribute, in the meta data. Values of this new column are the actual class of the corresponding object.

Analyzing meta data: Rough set based attribute reduction techniques eliminate superfluous attributes and create a minimal sufficient subset of attributes for a decision table. Such minimal sufficient subset of attributes, called a reduct, is an essential part of the decision table which can discern all examples discernible by the original table and cannot be reduced any more. This reduced set of classifiers provide the same classification ability as the decision table. Given set of classifiers may have more than one reduced set, all of which perform same as original, in that case minimal reduct is selected for final classification task.

Removing redundancy: Once the reduct is computed we remove redundant classifiers from the ensemble and construct new reduced ensemble of classifiers with the remaining base level classifiers. Note that we don't need to train this new combination because they are already trained.

Extracting decision rules from meta data: In this step we perform rough set decision rule learning algorithm to take decision at meta data level.

Classification: For classification of the remaining documents we validate the test data and decision rules are used to classify the documents. Examples of a decision table and a rule set are shown in tables 2 and 3 respectively.

Remark: It may be noted that unique set of classifiers may not be obtained as a reduct in the proposed method, since a rough set theoretic formulation is used.

Table 2. A slice of decision table for WebKb data set with only three category

object	maxent	nb	svm	Actual class
1.	course	course	course	course
2.	faculty	course	course	course
3.	course	course	course	course
4.	department	department	department	department
5.	department	department	faculty	department
6.	faculty	faculty	faculty	faculty
7.	course	course	faculty	faculty
8.	course	faculty	faculty	faculty

Table 3. Decision rules for WebKb data set with only three category

(svm=course)&(maxent=course)&(nb=course)=>(class=course)
(nb=faculty)&(maxent=faculty)&(svm=faculty)=>(class=faculty)
(maxent=department)&(nb=department)=>(class=department)
(maxent=faculty)&(nb=faculty)&(svm=course)=>(class=course)
(nb=faculty)&(maxent=department)=>(class=course)
(maxent=faculty)&(nb=department)=>(class=faculty)
(nb=faculty)&(svm=course)&(maxent=course)=>(class=faculty)

4.2. Proposed algorithm

Algorithm 1.

Input:

a set of labeled data $U = \{(x_i, y_i), i = 1, 2, ..., n\}$, where labels y_i is one of $1, 2, 3, ..., K$

a pull of base classifiers $H = \{h_1, h_2, ..., h_t\}$

Algorithm:

Step 1. split U into U_1, U_2, U_3

Step 2. for each h_j in H

Step 3. Train h_j by U_1

Step 4. end

Step 5. for each s in U_2

Step 6. Test $h_1, h_2, ...h_t$

Step 7. Add prediction in Decision Table as value of condition attributes $c_1, c_2, ...c_t$

Step 8. Add actual class in Decision Table as decision attribute d

Step 9. end

Step 10. compute Rough set reduct H_1, a subset of H

Step 11. compute Decision rules

Step 12. Add rules in R, a rule base

Step 13. for each s in U_3

Step 14. Test H_1 and Get prediction of reduced base classifiers

Step 15. Apply rule

Output: category of s

4.3. Evaluation of method

Here learning is performed twice to solve the problem, that is, a classifier ensembles is trained and then rules are learned from their predictions. The reason is that the goal of RSM is to improve the non-redundancy of trained classifier ensembles and generate very accurate rules for decision making, which means that the ensembles have already been trained and the ”real” cost of RSM is the second phase learning. Moreover, the cost of twice learning is worthwhile even without the consideration of the goal, which makes it a competitive alternative to present more accurate meta classification approach. We don’t need to test hundreds of classifiers, which is common in other ensemble methods. Redundancy removal process reduces the CPU load in later steps.

5. Experimental results

We performed a large number of experiments to test the output of RSM. We now describe the corpora, methodology, and results.

5.1. Data collection

We crawled the Looksmart and Dmoz web directories to collect examples data for our learning problem [Table 4]. These directories are well known for maintaining a categorised web documents. The web directories are multi-level tree-structured hierarchy. The top level of the tree, which is the first level below the root of the tree, contains 13 and 16 categories respectively. Each of these categories contains sub-categories that are placed in the second level below the root. We use the top-level categories to label the web pages in our experiments. We processed the data set to remove brief greeting sentence, image, Java script, and other non-textual information, stop-words and stem with Porter's stemming algorithm. We use the standard "TFIDF" document representation from IR. In keeping with some of the best systems at TREC (http://trec.nist.gov/), our IDF for term t is $\ln(\frac{|D|}{|D_t|})$ where D is the document collection and $D_t \subset D$ is the set of documents containing t . The term frequency $TF(d,t) = 1 + \ln(1 + \ln(n(d,t)))$, where $n(d,t) > 0$ is the raw frequency of t in document d (TF is zero if $n(d,t) = 0$). d is represented as a sparse vector with the t^{th} component being $IDF(t)TF(d,t)$. The L_2 norm of each document vector is scaled to 1 before submitting it to the classifier [11, 4].

We used the following publicly available data sets. The first three are well-known in recent information retrieval literature, small in size and suitable for controlled experiments on accuracy and CPU scaling. The last two data sets are large; they were mainly used to test memory scaling (but we verified that they show similar patterns of accuracy as the smaller data sets). In our experiment we divided the training data set in two parts for first phase and second phase training.

Reuters: 7700 training and 3000 test documents ("MOD-APTE" split), 30000 terms, 135 categories. The raw text takes about 21 MB.

20NG: Here 18800 total documents organized in a directory structure with 20 topics. For each topic the files are listed alphabetically and the first 25documents. There are 94000 terms. The raw concatenated text takes up 25 MB.

WebKB: Here there are 8300 documents in 7 categories. About 4300 pages on 7 categories (faculty, project, etc.) were collected from 4 universities and about 4000 miscellaneous pages were collected from other universities. For each classification task, any one of the four university pages are selected as test documents and rest as training documents. The raw text is about 26 MB.

Dmoz: A cut was taken across the Dmoz (http://dmoz.org/) topic tree yielding 16 topics covering most areas of Web content. The raw text occupied 271 MB.

Looksmart: We crawled a part of Looksmart web directory (http://www.looksmart.com) containing 26000 web pages divided into 13 top level directories with 2000 documents each directory. The crawled data takes 126 MB

5.2. Results

Table 6 presents the performance results on the above mentioned benchmark corpora. We have described below several issues in detail the selection of base classifiers, training requirements in first phase and training requirements in second phase. We compared RSM with other ensemble classifiers like Bagging, Adaboost and Stacking considering different types of configurations for our method and corresponding configurations for competing methods.

5.2.1. Base classifiers selection

We compared our method with some well known ensemble classifiers, AdaBoost, Bagging and Stacking. Comparison results are shown in [Table 6]. For Stacking and RSM we used **J4.8** [a Java re-implementation of the decision tree learning algorithm C4.5 (Quinlan, 1993)], **NB** [the naive Bayes algorithm of John and Langley (1995)], **IB**k [the k-nearest neighbor algorithm of Aha, Kibler, and Albert (1991)], and **MaxEnt** [classifier based on the principle of maximum entropy] as base level classifiers. Among these classifiers NB performs the best and hence we used NB as base classifier for AdaBoost and Bagging.

5.2.2. Effect of varying training percentages

Training Percentage coupled with performance is an important issue in two step learning methods. We tested our algorithm on 20NG and WebKB dataset taking different percentage of training in first step and second step represented in [Table 5].

5.2.3. Comparison of RSM with other ensemble classifiers

[Table 8 & 6] shows the comparison of RSM with other ensemble classifiers. Same base level classifiers are chosen for RSM and Stacking and best among the base level classifiers is chosen for AdaBoost and Bagging. Results of [Table 6] show that RSM perform better than all other methods like Bagging, AdaBoost and Stacking for every data set considered here. Results of [Table 8] show that RSM performs better than all other methods like Bagging, AdaBoost and Stacking for every category of the Dmoz data set. These results are a demonstration of the mathematical proof of Theorem 1.

5.2.4. Comparison of RSM with single classifiers

To compare RSM with single classifier we need to consider two cases: (1) comparison with a single classifier which has already been considered as a base classifier of RSM, (2) comparison with a single classifier which was not considered as a base classifier of RSM. In the first case we have shown mathematically that RSM will perform better than its constituent classifiers. Results of [Table 7] follow the mathematical demonstration. In the second case, comparison with SVM is given, because it is known as best for text classification [4]. We considered NB, IBk and J4.8 as base classifier of RSM. Accuracy of these three classifiers is less than accuracy of SVM for text classification. Results of [Table 7] show that RSM with this configuration perform slightly better than SVM

Table 4. A part of DMoz web directory

Category	web pages	Category	web pages
Arts	5000	News	4747
Business	4497	Recreation	4506
Computers	4745	Reference	4501
Games	4485	Regional	5253
Health	3957	Science	4230
Home	3729	Shopping	3510
Sports	4141	World	3141
Kids and Teens	3772	Society	4615

Table 5. Training Percentage and performance

Number	Training		Performance	
	1^{st} Phase	2^{nd} Phase	WebKB	20NG
1	5%	5%	98.73%	94.58%
2	5%	10%	99.03%	94.65%
3	5%	15%	99.12%	94.93%
4	10%	5%	99.79%	96.16%
5	10%	10%	99.79%	96.17%

Table 6. Accuracy comparison of RSM with other ensemble classifiers on some benchmark corpora.

Dataset	AdaBoost	Bagging	Stacking	RSM
Dmoz	94.94%	91.35%	92.64%	99.44%
20NG	92.82%	87.80%	93.29%	96.16%
WebKB	97.85%	95.77%	96.77%	99.79%
Reuters	89.91%	86.22%	87.63%	94.31%
Looksmart	99.74%	99.55%	99.64%	99.97%

Table 7. Accuracy comparison of RSM with single classifier on some benchmark corpora.

Dataset	SVM	RSM using SVM	RSM not using SVM
Dmoz	98.40%	99.64%	99.44%
20NG	92.96%	96.29%	96.16%
WebKB	96.11%	99.77%	99.79%
Reuters	93.48%	94.63%	94.31%
Looksmart	99.49%	99.97%	99.97%

Table 8. Accuracy comparison of classifiers for Dmoz categories.

Category	Linear SVM	AdaBoost	Bagging	Stacking	RSM
Arts	99.00%	97.70%	98.52%	97.70%	99.94%
Business	93.29%	95.60%	62.68%	95.60%	99.02%
Computers	97.71%	97.10%	70.27%	97.10%	99.70%
Games	99.85%	99.11%	99.67%	99.11%	99.94%
Health	99.06%	99.35%	97.46%	99.35%	100.00%
Home	97.85%	78.62%	81.99%	78.62%	98.65%
Kids and Teens	99.79%	98.56%	99.03%	98.56%	99.91%
News	98.17%	79.55%	59.24%	79.55%	99.21%
Recreation	96.90%	95.28%	95.21%	95.28%	98.85%
Reference	98.04%	97.26%	92.04%	97.26%	99.59%
Regional	97.59%	93.42%	93.37%	93.42%	98.25%
Science	96.57%	95.56%	79.80%	95.56%	98.97%
Shopping	92.46%	89.71%	63.33%	89.71%	95.63%
Society	96.75%	93.51%	87.90%	93.51%	98.34%
Sports	98.98%	96.82%	98.95%	96.82%	99.81%
World	100.00%	97.50%	89.23%	97.50%	100.00%
Average	98.40%	94.94%	91.35%	92.64%	99.44%

5.2.5. Removal of redundant classifiers

We used WebKb data set and J4.8, IBk, NB and SVM as base classifiers of RSM. The reduct obtained in this setup consists only of two classifiers, IBk and SVM. In another setup we used 20NG data set and J4.8, IBk, NB and SVM as base classifiers of RSM. The two reducts obtained here are $\{$J4.8, IBk, NB$\}$ and $\{$J4.8, IBk, SVM$\}$. These demonstrate the use of less number of classifiers i.e. less CPU load at the testing time of RSM.

6. Conclusion

We proposed a methodology for building a meta classifier for text documents that centers on combining multiple distinct classifiers with rough set paradigm. It views a classifier output as a partition on the dataset. It tries to find the effectiveness of classifiers to build the combination classifier. Our method uses decision rules to make final prediction about the category of text documents. Experimental studies show that it improves accuracy uniformly over some benchmark corpora. Apart from this, by removing redundant classifiers from the combination it reduces the CPU load compared to other ensemble classification techniques. It is possible to search for further improvement of the results, e.g., by applying methods for selecting the "best" reduct.

References

[1] Rough Set Exprloration System (RSES) available at http://logic.mimuw.edu.pl/ rses/.

[2] Bazan, J. G., Nguyen, H. S., Nguyen, S. H., Synak, P., Wroblewski, J.: Rough set algorithms in classification problem, *Rough set methods and applications: new developments in knowledge discovery in information systems, Physica-Verlag GmbH, Heidelberg, Germany*, 2000, 49–88.

[3] Breiman, L.: Bagging predictors, *Machine Learning*, **24**, 1996, 123–140.

[4] Chakrabarti, S., Roy, S., Mahesh, V., Soundalgekar: Fast and accurate text classification via multiple linear discriminant projections, *The International Journal on Very Large Data Bases*, **12**(2), 2003, 170–185.

[5] Dietterich, T. G.: An experimental comparison of three methods for constructing ensembles of decision tree, *Machine Learning*, **40**, 2000, 139–158.

[6] Dietterich, T. G., Bakiri, G.: Solving multiclass learning problems via errorcorrecting output codes, *Journal of Artificial Intelligence Research*, **2**, January 1995, 263–286.

[7] Dzeroski, S., Zenko, B.: Is Combining Classifiers with Stacking Better than Selecting the Best One?, *Machine Learning, Kluwer Academic Publishers, Hingham, MA, USA*, **54**(3), 2004, 255–273, ISSN 0885-6125.

[8] Gama, J.: Combining Classiers by Constructive Induction, *Ninth European Conference on Machine Learning, Springer 1997, Prague, Czech Republic*, April 1997.

[9] Larkey, L. S., Croft, W. B.: Combining classifiers in text categorization, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval, ACM Press, New York, US*, 1996, 289–297.

[10] Li, Y. H., Jain, A. K.: Classification of text documents, *Computer Journal*, **41**(8), 1998, 537–546.

[11] McCallum, A., Nigam, K.: A comparison of event models for naive Bayes text classication, *International Conference on Machine Learning, Morgan Kaufmann Publishers, Madison, Wisconsin USA*, 1998.

[12] Merz, C. J.: Using Correspondence Analysis to Combine Classiers, *Machine Learning. Kluwer Academic Publishers*, **36(1/2)**, 1999, 33–58.

[13] Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data, *Kluwer Academic Publishers, Dordrecht; Boston*, 1991.

[14] Pawlak, Z.: Rough sets and decision analysis, *INFOR: Information system and operational research*, **38**(3), 2000, 132–144.

[15] Pawlak, Z.: Rough sets and decision algorithms, in Rough Sets and Current Trends in Computing (Second International Conference, RSCTC 2000), Springer, Berlin, *RSCTC*, 2001, 30–45.

[16] Quinlan, J. R.: Bagging, Boosting and C4.5, *AAAI96: In Proc. of the 13 AAAI(American Association for Artificial Intelligence) Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA*, 1996, 725–730.

[17] Schutze, H., Hull, D. A., Pederson, J. O.: comparison of classiers and document representations for the routing problem, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM Press, Seattle, Washington, USA*, 1995, 229–237.

[18] Scott, S., Matwin, S.: Feature engineering for text classification, *ICML: International Conference on Machine Learning', Morgan Kaufmann Publishers, San Francisco, US, Bled, Slovenia*, 1999, 379–388.

[19] Senator, T. E.: Multi-Stage Classification, *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA*, 2005, 386–393.

[20] Ting, K. M., Witten, I. H.: Issues in stacked generalization, *Journal of Articial Intelligence Research*, **10**, 1999, 271–289.

[21] Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective Voting of Heterogeneous Classifiers, *15th European Conference on Machine Learning, Springer, Berlin / Heidelberg*, **3201**, 2004, 20–24, ISSN 0302-9743.

[22] Tumer, K., Ghosh, J.: Robust combining of disparate classifiers through order statistics, *Pattern Analysis and Applications, 5 (2002)*, 2002.

[23] Wang, G., Liu, Q., Yao, Y., Skowron, A.: Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, *RSFDGrC, Springer 2003, Chongqing, China*, 2003, 41–48.

[24] Zenko, B., Todorovski, L., Dzeroski, S.: A Comparison of Stacking with Meta Decision Trees to Bagging, Boosting, and Stacking with other Methods, *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA*, 2001, 669–670.