International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:11, 2007

Text Mining Technique for Data Mining Application

M. Govindarajan

***Abstract*—**Text Mining is around applying knowledge discovery techniques to unstructured text is termed knowledge discovery in text (KDT), or Text data mining or Text Mining. In decision tree approach is most useful in classification problem. With this technique, tree is constructed to model the classification process. There are two basic steps in the technique: building the tree and applying the tree to the database. This paper describes a proposed C5.0 classifier that performs rulesets, cross validation and boosting for original C5.0 in order to reduce the optimization of error ratio. The feasibility and the benefits of the proposed approach are demonstrated by means of medial data set like hypothyroid. It is shown that, the performance of a classifier on the training cases from which it was constructed gives a poor estimate by sampling or using a separate test file, either way, the classifier is evaluated on cases that were not used to build and evaluate the classifier are both are large. If the cases in hypothyroid.data and hypothyroid.test were to be shuffled and divided into a new 2772 case training set and a 1000 case test set, C5.0 might construct a different classifier with a lower or higher error rate on the test cases. An important feature of see5 is its ability to classifiers called rulesets. The ruleset has an error rate 0.5 % on the test cases. The standard errors of the means provide an estimate of the variability of results. One way to get a more reliable estimate of predictive is by f-fold –cross- validation. The error rate of a classifier produced from all the cases is estimated as the ratio of the total number of errors on the hold-out cases to the total number of cases. The Boost option with x trials instructs See5 to construct up to x classifiers in this manner. Trials over numerous datasets, large and small, show that on average 10-classifier boosting reduces the error rate for test cases by about 25%.

***Keywords*—**C5.0, Error Ratio, text mining, training data, test data.

I. Introduction

DECISION tree approach is most useful in classification problems. With this technique, a tree is constructed to model the classification process. Once the tree is built, it is applied to each tuple in the database and results in a classification for that tuple. There are two basic steps in the technique: building the tree and applying the tree to the database.

1) C 5.0 algorithm: [3] C5.0 (called See5 on windows) is a commercial version of C4.5 now widely used in many data mining packages such as Clementine and RuleQuest. It is targeted toward use with large datasets. The DT induction is close to that C4.5, but the rule generation is different. Unlike C4.5, the precise algorithms used for C4.5 have not been divulged. C5.0 does include improvements to generate rules. Results show that C5.0 improves on memory usage by about 90 percent, runs between 5.7 and 240 times faster than C4.5 and produces more accurate rules.

One major improvement to the accuracy of C5.0 is based on boosting. Boosting is an approach to combining different classifiers. While boosting normally increase the time that it takes to run a specific classifier, it does improve the accuracy. Boosting does not always help when the training data contains a lot of noise. Boosting works by creating multiple training sets from one training set. Each item in the training set is assigned a weight. The weight indicates the importance of this item to the classification. A classifier is constructed for each combination of weights used. Thus, multiple classifiers are actually constructed. When C5.0 performs a classification, each classifier is assigned a vote, voting is performed, and the target tuple is assigned to the class with the most number of votes.

2) Proposed C5.0 algorithm: The feasibility the benefits of the proposed approach are demonstrated by means of medial data set like hypothyroid. It is shown that, the performance of a classifier on the training cases from which it was constructed gives a poor estimate by sampling or using a separate test file, either way, the classifier is evaluated on cases that were not used to build and evaluate the classifier are both are large. If the cases in hypothyroid data and hypothyroid test were to be shuffled and divided into a new 2772 case training set and a 1000 case test set, C5.0 might construct a different classifier with a lower or higher error rate on the test cases. One way to get a more reliable estimate of predictive is by f-fold –cross-validation. The error rate of a classifier produced from all the cases is estimated as the ratio of the total number of errors on the hold-out cases to the total number of cases.

Another innovation incorporated in See5 is adaptive boosting, based on the work of Rob Schapire and Yoav Freund. The idea is to generate several classifiers (either decision trees or rulesets) rather than just one. When a new case is to be classified, each classifier votes for its predicted class and the votes are counted to determine the final class.

A. Performances of Decision Trees

Decision trees [3] are powerful and popular tools for classification and prediction. The attractiveness of decision trees is due to the fact that, in contrast to neural networks, decision trees represent *rules*. Rules can readily be expressed so that humans can understand them or even directly used in a database access language like SQL so that records falling into a particular category may be retrieved. In some applications, the accuracy of a classification or prediction is the only thing that matters. In such situations we do not necessarily care how or why the model works. In other situations, the ability to

Manuscript received November 15, 2007. This work was supported by Career Award for Young Teachers (CAYT) grant from All India Council for Technical Education, New Delhi.

M. Govindarajan is with the Annamalai University, Annamalai Nagar, Tamil Nadu, India (phone: 91-4144-221946; e-mail: govind_aucse@yahoo.com).

explain the reason for a decision is crucial. In marketing one has describe the customer segments to marketing professionals, so that they can utilize this knowledge in launching a successful marketing campaign. This domain expert must recognize and approve this discovered knowledge, and for this we need good descriptions. There are a variety of algorithms for building decision trees that share the desirable quality of interpretability. A well known and frequently used over the years is C4.5 (or improved, but commercial version See5/C5.0). A decision tree can be used to classify an example by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance. Decision tree induction is a typical inductive approach to learn knowledge on classification. The key requirements to do mining with decision trees are:

- *Attribute-value description*: object or case must be expressible in terms of a fixed collection of properties or attributes. This means that we need to discrete continuous attributes, or this must have been provided in the algorithm.
- *Predefined classes (target attribute values)*: The categories to which examples are to be assigned must have been established beforehand (supervised data).
- *Discrete classes*: A case does or does not belong to a particular class, and there must be more cases than classes.
- *Sufficient data*: Usually hundreds or even thousands of training cases.

1) Which attribute is the best classifier?

The estimation criterion in the decision tree algorithm [3] is the selection of an attribute to test at each decision node in the tree. The goal is to select the attribute that is most useful for classifying examples. A good quantitative measure of the worth of an attribute is a statistical property called *information gain* that measures how well a given attribute separates the training examples according to their target classification. This measure is used to select among the candidate attributes at each step while growing the tree.

2) Issues in data mining with decision trees

Practical issues in learning decision trees include determining how deeply to grow the decision tree, handling continuous attributes, choosing an appropriate attribute selection measure, handling training data with missing attribute values, handing attributes with differing costs, and improving computational efficiency. Below we discuss each of these issues and extensions to the basic ID3 algorithm that address them.

3) Avoiding over-fitting the data

In principle decision tree algorithm described in Fig. 2 can grow each branch of the tree just deeply enough to perfectly classify the training examples. While this is sometimes a reasonable strategy, in fact it can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that *over-fit* the training examples. Over-fitting [3] is a significant practical difficulty for decision tree learning and many other learning methods. There are several approaches to avoiding over-fitting in decision tree learning. These can be grouped into two classes:

- approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,
- approaches that allow the tree to over-fit the data, and then post prune the tree.

B. Strengths and weakness of Decision Tree Methods

The strengths of decision tree methods are:

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

The weaknesses of decision tree methods:

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
- Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.
- Decision trees do not treat well non-rectangular regions. Most decision-tree algorithms only examine a single field at a time. This leads to rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.

The remainder of this article is structured as follows. First, the state of the art is analyzed to motivate our work (Section II) and the C5.0 algorithm was described (Section III). Then, the model evaluation of C5.0 algorithm is introduced (Section IV). After that, preprocessing of dataset was described.

(Section V). Finally, the main findings are summarized and an outlook on future work is given (Section VI).

II. State of the Art

In this section, the state of the art concerning rule sets, cross validation, boosting of C5.0 algorithm is investigated. The results of this survey will motivate a new approach.

A. Related Work

These articles focus on error ratio using rulesets, cross validation, boosting of C5.0 algorithm. Rule sets, Cross validation, boosting methods are described in [1]. Here, we discuss examples of the combination of C5.0 and Proposed C5.0 algorithm. The feasibility the benefits of the proposed approach are demonstrated by means of medial data set like hypothyroid. The following steps are carrying out to classify decision tree methods like C5.0 algorithm [4].

1. Decision tree induction: Construct a DT using training data
2. For each $t_i \in D$, apply the DT to determine its class

Since the application of a given tuple to a DT is relatively straightforward.

B. Motivation for a New Approach

C5.0 [4] offers a number of improvements on C4.5. Some of these are:

- Speed - C5.0 is significantly faster than C4.5 (several orders of magnitude)
- Memory Usage - C5.0 is more memory efficient than C4.5
- Smaller Decision Trees - C5.0 gets similar results to C4.5 with considerably smaller decision trees.
- Support For Boosting - Boosting improves the trees and gives them more accuracy.
- Weighting - C5.0 allows you to weight different attributes and misclassification types.

Winnowing - C5.0 automatically winnows the data to help reduce noise.

III. Classification with C5.0 Algorithm

C5.0 (called See5 on windows) [4] is a commercial version of C4.5 now widely used in many data mining packages such as Clementine and RuleQuest. It is targeted toward use with large datasets. The DT induction is close to that C4.5, but the rule generation is different. Unlike C4.5, the precise algorithms used for C4.5 have not been divulged. C5.0 does include improvements to generate rules. Results show that C5.0 improves on memory usage by about 90 percent, runs between 5.7 and 240 times faster than C4.5 and produces more accurate rules. One major improvement to the accuracy of C5.0 is based on boosting. Boosting is an approach to combining different classifiers. While boosting normally increase the time that it takes to run a specific classifier, it does improve the accuracy. Boosting does not always help when the training data contains a lot of noise. Boosting works by creating multiple training sets from one training set. Each item in the training set is assigned a weight. The weight indicates the importance of this item to the classification. A classifier is constructed for each combination of weights used. Thus, multiple classifiers are actually constructed. When C5.0 performs a classification, each classifier is assigned a vote, voting is performed, and the target tuple is assigned to the class with the most number of votes.

In pseudocode the algorithm looks like this:

- Check for base cases
- For each attribute a
- Find the normalized information gain from splitting on a
- Let a_best be the attribute with the highest normalized
- information gain

- Create a decision node node that splits on a_best
- recurse on the sublists obtained by splitting on a_best and add those nodes as children of node

IV. Model Evaluation

In this section, a schematic overview of rule sets, cross validation, boosting used for C5.0 algorithm. Then, the standard techniques are sketched and our innovative extensions are described in detail.

A. Rulesets

Decision trees can sometimes be quite difficulty to understand. An important feature of C5.0 is its ability to generate classifiers called rulesets that consists of unordered collections of (relatively) simple if-then rules. Rulesets [1] are generally easier to understand than trees since each rule describes a specific context associated with a class. Furthermore, a ruleset generated from a tree usually has fewer rules than the tree has leaves, another plus for comprehensibility. (In this example, the first decision tree with 14 leaves is reduced to seven rules.) Finally, rules are often more accurate predictors than decision trees -- a point not illustrated here, since the ruleset has an error rate of 0.5% on the test cases. For very large datasets, however, generating rules with the ruleset option can require considerably more computer time.

B. Cross-Validation Method

The performance of a classifier on the training cases from which it was constructed gives a poor estimate of its accuracy on new cases. The true predictive accuracy of the classifier can be estimated by sampling, as above, or by using a separate test file; either way, the classifier is evaluated on cases that were not used to build it. However, this estimate can be unreliable unless the numbers of cases used to build and evaluate the classifier are both large. If the cases in hypothyroid.data and hypothyroid.test were to be shuffled and divided into a new 2772-case training set and a 1000-case test

set, See5 might construct a different classifier with a lower or higher error rate on the test cases. One way to get a more reliable estimate of predictive accuracy is by f-fold cross-validation. The cases in the data file are divided into f blocks of roughly the same size and class distribution. For each block in turn, a classifier is constructed from the cases in the remaining blocks and tested on the cases in the hold-out block. In this way, each case is used just once as a test case. The error rate of a classifier produced from all the cases is estimated as the ratio of the total number of errors on the hold-out [6] cases to the total number of cases.

C. Boosting

Another innovation incorporated in See5 is adaptive boosting, based on the work of Rob Schapire and Yoav Freund. The idea is to generate several classifiers (either decision trees or rulesets) rather than just one. When a new case is to be classified, each classifier votes for its predicted class and the votes are counted to determine the final class. But how can we generate several classifiers from a single dataset? As the first step, a single decision tree or ruleset is constructed as before from the training data (e.g. hypothyroid.data). This classifier will usually make mistakes on some cases in the data; the first decision tree, for instance, gives the wrong class for 7 cases in hypothyroid.data. When the second classifier is constructed, more attention is paid to these cases in an attempt to get them right. As a consequence, the second classifier will generally be different from the first. It also will make errors on some cases, and these become the the focus of attention during construction of the third classifier. This process continues for a pre-determined number of iterations or trials, but stops if the most recent classifiers is either extremely accurate or inaccurate. The Boost option with x trials instructs See5 to construct up to x classifiers in this manner. Naturally, constructing multiple classifiers requires more computation that building a single classifier -- but the effort can pay dividends! Trials over numerous datasets, large and small, show that on average 10-classifier boosting reduces the error rate for test cases by about 25%.

V. Data Pre-Processing

Even though See5 is relatively fast, building classifiers from large numbers of cases can take an inconveniently long time, especially when options such as boosting are employed. See5 incorporates a facility to extract a random sample from a dataset, construct a classifier from the sample, and then test the classifier on a disjoint collection of cases. By using a smaller set of training cases in this way, the process of generating a classifier is expedited, but at the cost of a possible reduction in the classifier's predictive performance. The Sample option with x% has two consequences. Firstly, a random sample containing x% of the cases in the application's data file is used to construct the classifier. Secondly, the classifier is evaluated on a non-overlapping set of test cases consisting of another (disjoint) sample of the same size as the training set (if x is less than 50%), or all cases that were not used in the training set (if x is greater than or equal to 50%). In the hypothyroid example, using a sample of 60% would cause a classifier to be constructed from a randomly-selected 1663 of the 2772 cases in hypothyroid.data [6] then tested on the remaining 1109 cases. By default, the random sample changes every time that a classifier is constructed, so that successive runs of See5 with sampling will usually produce different results. This re-sampling can be avoided by selecting the Lock sample option that uses the current sample for constructing subsequent classifiers. If this option is selected, the sample will change only when another application is loaded, the sample percentage is altered, the option is unselected, or See5 is restarted.

VI. Empirical Results

In this section we demonstrated the properties and advantages of our approach by means of data set like hypothyroid. The performance of classification algorithms is usually examined by evaluating the accuracy of the classification. However, since classification is often a fuzzy problem [1] [7], the correct answer may depend on the user. Traditional algorithm [2] evaluation approaches such as determining the space and time overhead can be used, but these approaches are usually secondary. Classification accuracy [4] is usually calculated determining the percentage of tuples placed in the correct class. This ignores the fact that there also may be a cost associated with an incorrect assignment to the wrong class. This perhaps should also be determined [12]. We examine the Performance of rulsets, cross validation, boosting for original C5.0 algorithm depending on error rate. The standard errors of the means provide an estimate of the variability of results.

TABLE I
Properties of Data Set

Dataset Factor of	Proposed C5.0 algorithm (PC5.0)	Original C5.0 algorithm (OC5.0)	Faster by
Hypothyroid	0.5 %	7.2 %	6.7 %

TABLE II
Error Ratio (Using Rulesets)

Dataset Factor of	Proposed C5.0 algorithm (PC5.0)	Original C5.0 algorithm (OC5.0)	Faster by
Hypothyroid	0.5 %	7.2 %	6.7 %

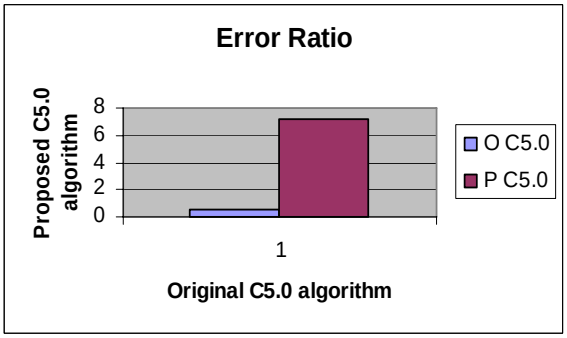

Fig. 1 Error ratio for Rulesets

TABLE III
ERROR RATIO (USING 10-CROSS VALIDATION)

Dataset Factor of	Proposed C5.0 algorithm (PC5.0)	Original C5.0 algorithm (OC5.0)	Faster by
Hypothyroid	0.3 %	7.2 %	6.9 %

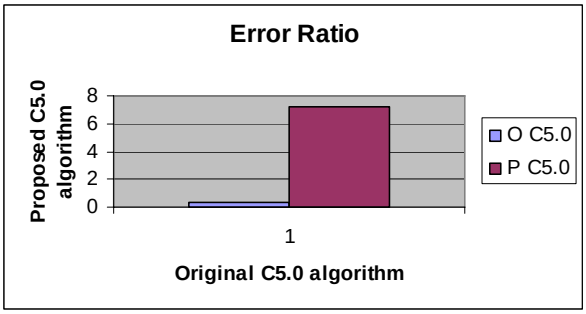

Fig. 2 Error ratio for 10-fold Cross Validation

TABLE IV
ERROR RATIO (USING BOOSTING)

Dataset Factor of	Proposed C5.0 algorithm (PC5.0)	Original C5.0 algorithm (OC5.0)	Faster by
Hypothyroid	0.2 %	7.2 %	7.0 %

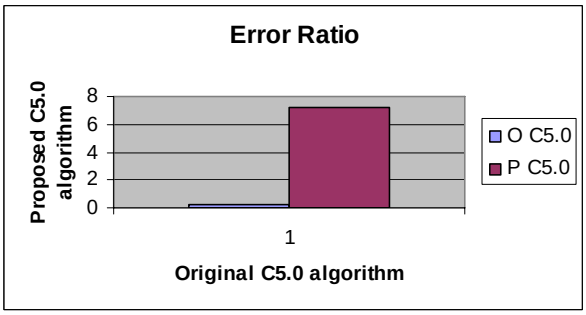

Fig. 3 Error ratio for Boosting

VII. CONCLUSION

In this work we developed one text mining classifier using C5.0 methods to measure the classification accuracy for hypothyroid data set. First, we utilized our developed text mining algorithms, including text mining techniques based on classification of medical data like hypothyroid After that, we employ exiting C5.0 to deal with measure the classification accuracy. It is shown that, the performance of a classifier on the training cases from which it was constructed gives a poor estimate by sampling or using a separate test file, either way, the classifier is evaluated on cases that were not used to build and evaluate the classifier are both are large. If the cases in hypothyroid.data and hypothyroid.test were to be shuffled and divided into a new 2772 case training set and a 1000 case test set, C5.0 might construct a different classifier with a lower or higher error rate on the test cases. An important feature of see5 [11] is its ability to classifiers called rulesets. The ruleset has an error rate 0.5 % on the test cases. The standard errors of the means provide an estimate of the variability of results. One way to get a more reliable estimate of predictive is by f-fold – cross- validation. The error rate of a classifier produced from all the cases is estimated as the ratio of the total number of errors on the hold-out cases to the total number of cases. The Boost option with x trials instructs See5 to construct up to x classifiers in this manner. Naturally, constructing multiple classifiers requires more computation that building a single classifier -- but the effort can pay dividends! Trials over numerous datasets, large and small, show that on average 10-classifier boosting reduces the error rate for test cases by about 25%.

ACKNOWLEDGEMENT

Authors gratefully acknowledge the authorities of Annamalai University for the facilities offered and encouragement to carry out this work. This part of work is supported in part by the first author got Career Award for Young Teachers (CAYT) grant from All India Council for Technical Education, New Delhi. They would also like to thank the reviewer's for their valuable remarks.

REFERENCES

[1] Themis P.Exarchos, Markos G. Tsipouras, Costas P. Exarchos, Costas Papaloukas, Dimitrios I. Fotiadis, Lampros K. Michalis, "A methodology for the automated creation of fuzzy expert systems for ischaemic and arrhymic beat classification based on a set of rules obtained by a decision tree" Artificial Intelligence in medicine (2007) 40, 187-200.

[2] M.Govindarajan, Dr.RM.Chandrasekaran, "Classifier Based Text Mining for Neural Network" Proceeding of XII international conference on computer, electrical and system science and engineering, may 24-26, Vienna , Austria, waste.org,2007. pp. 200-205.

[3] Jiawei Han , Micheline Kamber " Data Mining – Concepts and Techniques" Elsevier, 2007 pages 291- 310.

[4] Margaret H.Dunham, "Data Mining- Introductory and Advanced Topics" Pearson Education, 2007 pages 92-101.

[5] Marion Verduijn, Lucia Sacchi, Niels Peek, Riccardo Bellazzi, Evert de Jonge, Bas A.J.M. de Mol."Temporal abstraction for feature extraction: A comparative case study in prediction from intensive care monitorinf data" Artificial Intelligence in Medicine (2007) 41, 1-12.

[6] Kemal Polat, Salih Gunes, Sulayman Tosun "Diagnosis disease using artificial immune recognition system and fuzzy weighted pre-processing" Pattern Recognition 39 (2006) 2186-2193.

[7] Tim W.Nattemper, Bert Arnrich, Oliver Lichte, Wiebke Timm, Andreas Degenhard, Linda Pointon, Carmel Hayes, Martin O. Leach. "Evaluation of radiological features for breast tumour classification in clinical screening with machine learning methods" Artificial Intelligence in Medicine (2005) 34, 129-139.
[8] Sanchis, A., GIL, J.A. and Heras, A. (2003): "El análisis discriminante en la previsión de la insolvencia en las empresas de seguros no vida", Revista Española de Financiación y Contabilidad,116, enero-marzo, 183-233.
[9] Segovia, M.J., Gil, J.A., Heras, A. and Vilar, J.L. (2003): "Lametodología Rough Set frente al Análisis Discriminante en los problemas de clasificación multiatributo", XI Jornadas ASEPUMA, Oviedo, Spain.
[10] Venables, W.N. and Ripley, B.D. (2002): Modern Applied Statistics with S, Springer-Verlag, New York.
[11] De Anders, J. (2001): "Statistical Techniques vs. SEE5 Algorithm. An Application to a Small Business Environment", International Journal of Digital Accounting Research, 1 (2), 153-179.
[12] Duda, R.O., Hart, P.E. and STORK, D.G. (2001): Pattern Classification, John Wiley & Sons, Inc., New York.

M. Govindarajan received the B.E and M.E and Pursuing Doctoral Degree in Computer Science and Engineering from Annamalai University, Tamil Nadu, India in 2001 and 2005 and 2006 respectively. He is currently a lecturer (Senior Scale) at the Department of Computer Science and Engineering, Annamalai University, Tamil Nadu, India. He has presented and published more than 25 papers in Conferences and Journals. His current Research Interests include Data Mining and its applications, Algorithms, Text Mining, Neural Networks, genetic Algorithms, support vector machine, Radial Basis Function, ontology based Reasoning, Case Based Reasoning.

He has conducted National Conference on Recent Trends in Data Mining and its Applications (March 11-12, 2006) as well as proposed to conduct National Conference on "Research Prospects on knowledge Mining will be March 22nd & 23rd , 2008. He was the recipient of the Achievement Award for the field and to the Conference Bio-Engineering, Computer science, Knowledge Mining (2006), Prague, Czech Republic and All India Council for Technical Education "Career Award for Young Teachers (2006), New Delhi, India. He is Life Member of Computer Society of India, Indian Society for Technical Education and Session Member of Indian Science Congress Association.