INFORMATICA, 2004, Vol. 15, No. 4, 551–564

© 2004 *Institute of Mathematics and Informatics, Vilnius*

Text Categorization Using Neural Networks Initialized with Decision Trees

Nerijus REMEIKIS, Ignas SKUČAS, Vida MELNINKAITĖ
Faculty of Informatics, Vytautas Magnus University
Vileikos 8, 3035 Kaunas, Lithuania
e-mail: nerijus_remeikis@fc.vdu.lt, ignas_skucas@fc.vdu.lt, vida_melninkaite@fc.vdu.lt

Received: September 2004

Abstract. Text categorization – the assignment of natural language documents to one or more predefined categories based on their semantic content – is an important component in many information organization and management tasks. Performance of neural networks learning is known to be sensitive to the initial weights and architecture. This paper discusses the use multilayer neural network initialization with decision tree classifier for improving text categorization accuracy. Decision tree from root node until a final leave is used for initialization of each single unit. Growing decision trees with increasingly larger amounts of training data will result in larger decision tree sizes. As a result, the neural networks constructed from these decision trees are often larger and more complex than necessary. Appropriate choice of certainty factor is able to produce trees that are essentially constant in size in the face of increasingly larger training sets. Experimental results support the conclusion that error based pruning can be used to produce appropriately sized trees, which are directly mapped to optimal neural network architecture with good accuracy. The experimental evaluation demonstrates this approach provides better classification accuracy with Reuters-21578 corpus, one of the standard benchmarks for text categorization tasks. We present results comparing the accuracy of this approach with multilayer neural network initialized with traditional random method and decision tree classifiers.

Key words: text classification, decision trees, neural networks.

Introduction

As the volume of information continues to increase, there is growing interest in helping people better find, filter, and manage these resources. Text categorization – the assignment of natural language documents to one or more predefined categories based on their semantic content – is an important component in many information organization and management tasks. Automatic text categorization task can play an important role in a wide variety of more flexible, dynamic and personalized tasks as well: real-time sorting of email or files, document management systems, search engines, digital libraries, decision support systems for design. Text categorization is now being applied in many contexts, ranging from document indexing based on a controlled vocabulary, to document filtering, automated metadata generation, word sense disambiguation, population of hierarchical catalogues of Web resources, and in general any application requiring document organization or selective and adaptive document dispatching.

A number of statistical classification methods and machine learning techniques have been applied to text categorization, including techniques based on decision trees (Lewis and Ringuette, 1994), neural networks (Wiener *et al*., 1995), Bayes probabilistic approaches (Lewis and Ringuette, 1994). However there is still need more accurate text classifiers based on new learning approaches.

The purpose of the current work is to describe ways in which hybrid approach can be applied to the problem of text categorization, and to test its performance relative to a number of other text categorization algorithms. In this paper, we introduce the use of a hybrid decision tree and neural network technique to the problem of text categorization, because hybrid approaches can simulate human reasoning in a way that a decision tree learning is used to do qualitative analysis and neural learning is used to do subsequent quantitative analysis. Our approach is based on hybrid algorithm, which was described in paper (Banerji, 1997) and has been shown to perform well on standard machine learning tasks. The main idea of this method is to transform decision trees to neural networks. Our proposed hybrid approach for text categorization task constructs the networks by directly mapping decision nodes or rules to the neural units and compresses the network by removing unimportant and redundant units and connections.

Problem Definition

Automatic text categorization has always been an important application and research topic since the inception of digital documents. The text classification task can be defined as assigning category labels to new documents based on the knowledge gained in a classification system at the training stage. A wide range of supervised machine learning algorithms has been applied to this area using a training data set of categorized documents. Although text classification performance results has been quite encouraging, but there is still need more accurate text classifies based on new learning approaches (Sebastiani, 2002).

Performance of neural networks learning is known to be sensitive to the initial weights and architecture – number of hidden layers and neurons in these layers. Traditionally, the initial values of weights are determined randomly in the backpropagation neural network (BPN) and modified by the generalized delta rule. Although the BPN has been implemented in many applications, it still has some major drawbacks; namely, its convergence tends to be very slow, the optimal number of hidden nodes is difficult to determine, and usually it does not yield optimal solutions. Usefulness of good initialization, effect of initial values, prior weights are discussed in (Raudys, 2001). Recently, pattern recognition techniques have been used to initialize weights. Raudys and Skurichina (1992) used a piece-wise linear classiffier to initialize hidden layer weights of the three-layer neural network. In (Sethi, 1991; Banerji, 1997) decision tree was applied to initialize the neural network. These methods typically construct the networks by directly mapping decision nodes or rules to the neural units. Text categorization datasets are large and increases very rapidly. Growing decision trees with increasingly larger amounts of training data will result in larger decision tree sizes. As a result, the neural networks constructed from these decision trees are often larger and more complex than necessary.

Error based pruning can be used to prune a decision tree. It uses a parameter, the certainty factor, which affects the size of the pruned tree. Here, we show that varying the certainty factor allows significantly smaller trees to be obtained with minimal or no accuracy loss. Also, appropriate choice of certainty factor is able to produce trees that are essentially constant in size in the face of increasingly larger training sets. Experimental results support the conclusion that error based pruning can be used to produce appropriately sized trees, which are directly mapped to optimal neural network architecture with good accuracy.

This paper presents our attempt to improve text classification accuracy by neural network initialized with decision tree classifier and to compare classification performance to previous results.

Decision Tree Mapping to Neural Network

Decision Tree Construction Algorithm

Algorithm constructs the decision tree with a *divide and conquer* strategy. Each node in a tree is *associated* with a set of cases. At the beginning, only the root is present, with associated the whole training set and with all case weights equal to 1. At each node the following *divide and conquer* algorithm is executed, trying to exploit the locally best choice, with no backtracking allowed.

Let T be the set of cases associated at the node. The weighted frequency $freq(C_i, T)$ is computed of cases in T whose class is C_i, $i \in [1, N]$. If all cases in T belong to a same class C_j (or the number of cases in T is less than a certain value) then the node is a leaf, with associated class C_j.

If $T_1, \ldots, T_s$ are the subsets of T and T contains cases belonging to two or more classes, then the *information gain* of each attribute is calculated:

$$I = H(T) - \sum_{i=1}^{s} \frac{|T_i|}{|T|} \times H(T_i), \tag{1}$$

where

$$H(T) = -\sum_{j=1}^{n} \frac{freq(C_j, T)}{|T|} \times \log_2 \Big(\frac{freq(C_j, T)}{|T|} \Big) \tag{2}$$

is the entropy function.

While having an option to select information gain, by default, however, C4.5 (Quinlan, 1993) considers the *information gain ratio* of the splitting $T_1, \ldots, T_s$ which is the ratio of information gain to its split information:

$$S(T) = -\sum_{i=1}^{s} \frac{|T_i|}{|T|} \times \log_2 \Big(\frac{|T_i|}{|T|} \Big). \tag{3}$$

Training Multilayer Neural Network

Neurons in the input layer only act as buffers for distributing the input signals x_i to neurons in the hidden layer. Each neuron j in the hidden layer sums up its input signals x_i after weighting them with the strengths of the respective connections w_{ji} from the input layer and computes its output y_j as a function f of the sum as follows

$$y_j = f\left(\sum w_{ji}x_i\right), \tag{4}$$

where f can be a simple threshold function, a linear or a sigmoid function.

There are many available learning algorithms in the literature (Rumelhart and Mcclelland, 1986; Haykin, 1994). Backpropagation with momentum is the most commonly adopted neural network training algorithm (Rumelhart and Mcclelland, 1986). The backpropagation algorithm employs a gradient descent technique to adopt the neural network weights to minimise the mean squared difference between the ANN output and the desired output. Mean square error of one output neuron over all n examples is difference between the target t and actual a network output:

$$mse = \frac{1}{n}\sum_{i}^{n}\big(t(i) - a(i)\big)^2, \tag{5}$$

The change in weight $\Delta w_{ji}(k)$ between neurons i and j is as follows

$$\Delta w_{ji}(k) = \eta\delta_j x_i + \alpha\Delta w_{ji}(k-1), \tag{6}$$

where η is a parameter called the learning coefficient, $\alpha$is the momentum coefficient, and δ_j is a factor depending on output neuron or a hidden neuron. For output neurons

$$\delta_j = \frac{\partial f}{\partial net_j}(y_j - y_{net_j}), \tag{7}$$

where $net_j = \sum_i x_i w_{ji}, y_j, y_{net_j}$ are the target and the neural outputs for neuron j, respectively. For hidden neurons

$$\delta_j = \frac{\partial f}{\partial net_j}\sum_q w_{qj}\delta_q. \tag{8}$$

As there are no target outputs for hidden neurons in (4), the difference between the target and the actual neural output of a hidden neuron j is replaced by the weighted sum of the δ_q terms already obtained for neurons q connected to the output of j. Thus, iteratively, beginning with the output layer, the δ term is computed for neurons in all layers and weight updates determined for all connections according to (5).

Training a neural network by backpropagation with momentum training algorithm to compute y involves presenting it sequentially with different training sets. Differences

between the target output and the actual output of the neural network are backpropagated through the network to adapt its weights using (5)–(7). The adaptation is carried out after the presentation of each set. Each training epoch is completed after all patterns in the training set have been applied to the networks.

Decision Tree Error-Based Pruning

In general, a decision tree can be grown so as to have zero error on the training set. Also, in general, over-fitting occurs and the tree needs to be pruned in order to generalize well on the test set.

Error-based pruning considers the E errors among the N training examples at a leaf of the tree to give an estimate of the error probability for that node. The assumption is that these are E events in N independent trials which is, of course, not perfectly true. We want to know what the observed result tells us about the probability of an error over the entire population of examples that will end up at the leaf. Using the binomial theorem, confidence limits can be calculated for the probability of error for a given confidence level. The confidence level is the certainty factor parameter *CF*. The upper limit of the probability is found. Given this value the predicted number of errors for each leaf of a test node being considered for pruning can be calculated by multiplying the number of examples at the leaf by the upper limit of the probability confidence limit. The predicted number of errors if a node was a leaf can be calculated from the observed number of errors after its leaves are collapsed. The leaves are pruned if the number of predicted errors after pruning is less than the sum of predicted errors across the leaves. The smaller the *CF* becomes the more certain we are that the confidence interval contains the true probability of error. That is, the confidence interval is wider, and the upper limit on the probability that a particular example is in error is higher, making an example more likely to be incorrect and hence more pruning will be done. With a $CF = 100$ we have no confidence that the true error is in the interval and would simply take the observed error rate at the leaf.

The first thing to recognize is that a tree pruned at CF_2 can be obtained by pruning the tree pruned at CF_1 when $CF_1 > CF_2$. So, the search for the appropriate certainty factor would consist of choosing an initial certainty factor, CF_i, for pruning and then evaluating the resultant tree on the validation set to determine its accuracy. Next, choose a new certainty factor lower than the last and prune the pruned tree. Evaluate the resultant tree on the validation set. Continue creating new pruned trees until the stopping criterion is met.

Neural Network Initialization with Decision Tree Classifier

If we compare decision trees and neural networks we can see that their advantages and drawbacks are almost complementary. For instance humans easily understand knowledge representation of decision trees, which is not the case for neural networks. Decision trees have trouble dealing with noise in training data, which is again not the case for neural

networks, decision trees learn very fast and neural networks learn relatively slow, etc. Our idea was to combine decision trees and neural networks in order to combine their advantages. That is why we developed a combined approach for building decision trees.

First we build a decision tree that is then used to initialize a neural network. Such a network is then trained using the same training objects.

The source decision tree is converted to a disjunctive normal form, which is a set of normalized rules. Then the disjunctive normal form serves as source for determining the neural network's topology and weights. The neural network has two hidden layers. The number of neurons on each hidden layer depends on rules in the disjunctive normal form. The number of neurons in the output layer depends on how many outcomes are possible in the training set. The conversion is described in the next steps (Fig. 1):

1. Build a decision tree using (1)–(3).
2. Every path from the root of the tree to every single leaf is presented as a rule.
3. The set of rules if transformed into the disjunctive normal form, which is minimal representation of original set of rules.
4. In the input layer create as many neurons as there are attributes in the training set.
5. For each literal in the disjunctive normal form there is a neuron created in the first hidden layer (literal layer) of a neural network.
6. Set weights for each neuron in the literal layer, that represents a literal in the form (attribute > value) to $w_0 = -\sigma * value$ for each literal in the form (attribute $\leqslant$ value) to $w_0 = \sigma * value$. Set all the remaining weights to $+\beta$ or $-\beta$ with equal probability. Constant $\sigma = 5$ and constant $\beta = 0.025$. These values were determined by cross validating on an artificially created dataset.
7. For every conjunction of literals create a neuron in the second hidden layer (conjunctive layer).
8. Set weights that link each neuron in the conjunctive layer with the appropriate neuron in the literal layer to $w_0 = \sigma * (2n - 1)/2$, where n is a number of literals in the conjunct. Set all the remaining weights to $+\beta$ or $-\beta$ with equal probability.
9. For every possible class create a neuron in the output layer (disjunctive layer).
10. Set weights that link each neuron in the disjunctive layer with the appropriate neuron in the conjunctive layer to $w_0 = -\sigma * (1/2)$ Set all remaining weights to $+\beta$ or $-\beta$ with equal probability.
11. Train the neural network with the same training objects as were used for training the decision tree using (4)–(8).

Such network is then trained using backpropagation. Mean square error of such network converges toward 0 much faster than it would in the case of randomly set weights in the network. Even if we would use neural network before the backpropagation stage, it would already give good results.

Fig. 1 depicts a typical decision tree that might be generated by decision tree algorithm using (1)–(3). This decision tree could be expressed equivalently, as following rules in disjunctive normal form:

$$(\text{Attribute1} < 5) \vee ((\text{Attribute1} \geqslant 5) \wedge (\text{Attribute2} < 8)) \Rightarrow \text{Category1}.$$
$$(\text{Attribute1} \geqslant 5) \wedge (\text{Attribute2} \geqslant 8) \Rightarrow \text{Category2}.$$

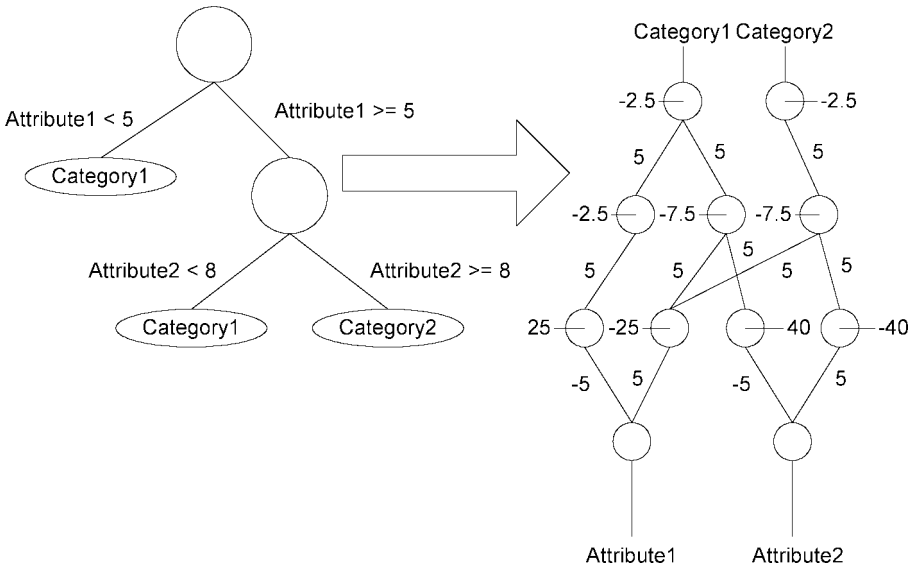

Fig. 1. Transformation of decision tree to neural network.

The technique creates four nodes in the literal layer that corresponds to $(\mathrm{Attribute1} < 5)$, $(\mathrm{Attribute1} \geqslant 5)$, $(\mathrm{Attribute2} < 8)$, $(\mathrm{Attribute2} \geqslant 8)$. The literal $(\mathrm{Attribute1} < 5)$ is characterized by the hidden node that has a bias of 25 and a weight of -5 on the input edge from $\mathrm{Attribute1}$. The other nodes in the literal layer represent, in order $(\mathrm{Attribute1} \geqslant 5)$, $(\mathrm{Attribute2} < 8)$, $(\mathrm{Attribute2} \geqslant 8)$. Three nodes are generated in the conjunction layer for $(\mathrm{Attribute1} < 5)$, $(\mathrm{Attribute1} \geqslant 5) \wedge (\mathrm{Attribute2} < 8)$, $(\mathrm{Attribute1} \geqslant 5) \wedge (\mathrm{Attribute2} \geqslant 8)$. For each such node, the weights on the edges from relevant nodes in the literal layer are set to 5, and the bias of the node is set to $-(5n - 2.5)$, where n denotes the number of literals in the disjunct. Two nodes are created in the output (disjunction) layer, one each for the two classes. The execution of this step is identical to the previous step. The bias of the nodes is set to -2.5. A final step connects each node in a layer to nodes in the previous and the next layer that remains disconnected from it after the mentioned procedure. The weights on these edges are set to 0.025 and -0.025 with equal probabilities. Fig. 1 depicts the skeletal network generated prior to the final step.

Experiment and Results

Text Corpora

It is hard to find standard benchmark sets for text classification, where each method can be tested and its performance compared reliably with other methods. The Reuters sets are a notable exception. This collection consists a set of newswire stories classified under categories related to economics. Although different versions are available, many researchers

use it for benchmarking. We will use the ApteMod version of Reuters-21578 (Yang and Liu, 1999). The ApteMod set has 7769 documents for training and 3019 for testing, after stemming and stop word removal 24240 unique terms remain. The Aptemod version has an average of 1.3 categories per document, with a total of 90 categories that occur in both sets.

Feature Selection and Extraction

In text categorization, features are often measures of frequencies of words appearing in a document. Feature selection chooses which features to be used in classification. It is preferable to use less features than the raw measurements (say, frequency of each word), so that classification will be performed in a feature space of a lower dimensionality. By reducing the dimensions of the feature space, it not only increases the efficiency of the training and test processes, but also reduces the risk of overfitting the model to data. Feature extraction computes the chosen features from an input document. In statistical classification, features are represented in a numerical vector, which is subsequently used by the classifiers. Feature selection involves stop word removal, stemming, and term selection:

- **Stop Word Removal.** Words used in text indexing and retrieval are called terms. According to the term discrimination model, moderate frequency terms discriminate the best. High frequency words, which are called *stop words*, have low information content, and therefore have weak discriminating power. They are removed according to a list of common stop words.
- **Stemming.** Stemming reduces morphological variants to the root word. For example, “asks”, “asked”, and “asking” are all reduced to “ask” after stemming. This relates the same word in different morphological forms and reduces the number of distinctive words. The *Porter stemmer* is a commonly used stemmer (Frakes and Baeza–Yate, 1992).
- **Term Selection.** Even after the removal of stop words and stemming, the number of distinct words in a document set may still be too large, and most of them appear only occasionally. In addition to removing high frequency words, the term discrimination model suggests that low frequency words are hard to learn about and therefore do not help much. They should be removed to reduce the dimensions of the vector space as well. We used information gain selection method (Yang and Pedersen, 1997).
- **Feature Extraction.** After the terms are selected, for each document a feature vector is generated whose elements are the feature values of each term. A commonly used feature value is the *term frequency* (number of occurrence of a term in a document).

Classification

A number of classifiers have been tried on text categorization. In our experiment, we focused on the evaluation of the neural network initialized with decision tree classifier

on text categorization. We compare its accuracy to those of classical decision tree C4.5 (Quinlan, 1993) and neural network initialized with random initial weights. Other the error back-propagation neural networks parameters that have been used in the experiments:

- learning rate – 0.3;
- momentum 0.1;
- number of iteration – 100.

Evaluation

The performance of category ranking can be evaluated in terms of *precision* and *recall*, computed at any threshold on the ranked list of categories of each document. The category assignment of a binary classifier can be evaluated using a two-way contingency table (Table 1).

The relationship between the system classification and the expert judgment is expressed using four values as shown in Table 1. Precision is defined as $a/(a+b)$, and recall is defined as $a/(a+c)$.

For evaluating performance average across categories, there are two conventional methods, namely macro-averaging and micro-averaging. Macro-averaged performance scores are computed by first computing the scores for the per-category contingency tables and then averaging these per-category scores to compute the global means. Micro-averaged performance scores are computed by first creating a global contingency table whose cell values are the sums of the corresponding cells in the per-category contingency tables, and then use this global contingency table to compute the micro-averaged performance scores. There is an important distinction between macro-averaging and micro-averaging. Micro-averaging performance scores give equal weight to every document, and is therefore considered a per-document average. Likewise, macro-average performance scores give equal weight to every category, regardless of its frequency, and is therefore a per-category average.

Results

Decision tree size grows approximately linearly with increasingly larger amounts training set size. Neural networks constructed from these trees are often larger than necessary. Large networks take a long time to learn, and tend to give accurate classification results for training data, but not for unknown test data. There are various approaches to

Table 1
The decision matrix for calculating the classification accuracy

	Expert YES	Expert NO
System YES	a	b
System NO	c	d

pruning decision trees, including error-based pruning, reduced error pruning, minimum description length pruning, and others (Quinlan, 1993). One well-known element of machine learning folklore is that decision tree pruning methods generally do not prune hard enough. Here we will show that, in general, an appropriate setting of the certainty factor for error-based pruning will cause decision tree size to plateau.

Fig. 2 shows results obtained with the Reuters-21578 data set. The training set size is varied from 1000 to 9584. Data is plotted as the error-based pruning certainty parameter is varied across values of 25 (the default), 10, 1, 0.1, 0.01, 0.001, 0.0001, 0.00001. The curve for the default certainty factor grows linearly. However, the family of curves clearly shows that the behavior depends on the value of the certainty factor. If the certainty factor is set as low as 0.001, then the average tree size varies between 61 and 373 over all training set sizes. That is, the tree size is minimal and optimal, just as desired.

The proposed technique was compared to backpropagation neural network with weights initialized randomly. Mean square error convergence is presented graphically in Fig. 3. Mean square error of neural network initialized with decision tree converges toward 0 much faster than in the case of randomly set weights.

For evaluating the effectiveness of our system, we used the breakeven points. The breakeven point is the point at which precision equals recall. Table 2 summarizes performance for our proposed neural network initialized with pruned decision tree, Banerjee classifier and to previous results achieved by neural networks and decision trees classifiers (Sebastiani, 2002). The measures used are precision/recall-breakeven point, micro-average and macro-average on ten most populated Reuters categories and over all categories.

On the Reuters data our hybrid approach performs best among other given methods. Slightly worse perform Banerjee neural network and decision tree classifers. Neural network initialized with traditional random method shows the worst results.

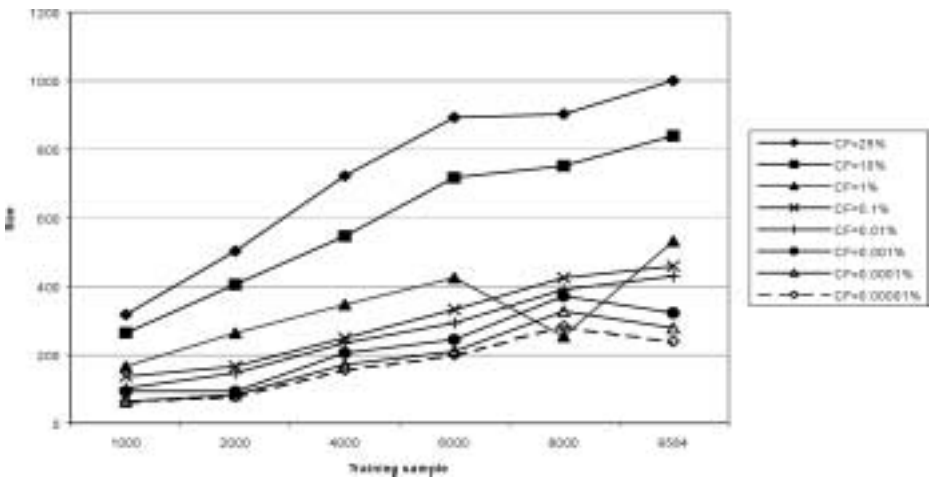

Fig. 2. Low certainty factor can give constant tree size with added training data.

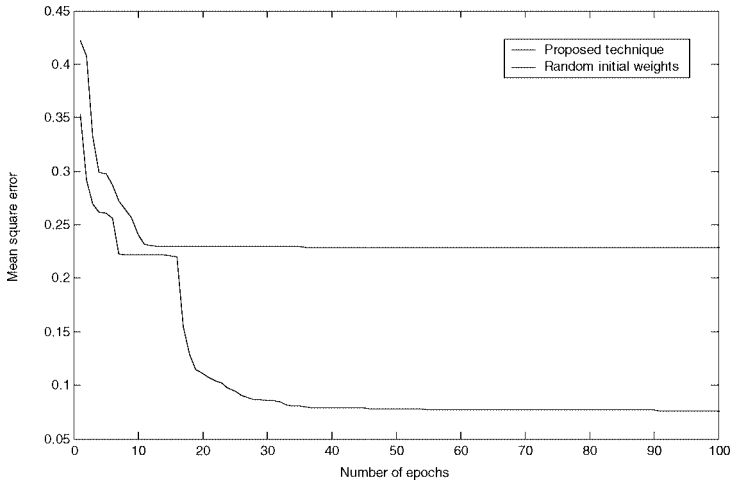

Fig. 3. Mean square error convergence.

Table 2

Precision/recall-breakeven point for ten most populated Reuters categories and micro averaged, macro averaged performance over all categories

Category	Proposed approach (%)	Hybrid approach (Banerjee) (%)	Neural networks (%)	Decision trees (%)
Acq	91.9	89.9	84.2	85.3
Corn	88.3	82.3	62.7	87.7
Crude	78.1	77.0	58.4	75.5
Earn	97.2	89.2	85.6	96.1
Grain	89.3	72.1	56.4	89.1
Interest	73.1	70.1	60.8	49.1
money-fx	70.4	72.4	62.3	69.4
Ship	81.2	73.2	67.6	80.9
Trade	76.7	69.7	59.2	59.2
Wheat	88.5	86.5	46.9	85.5
Micro-avg	83.8	81.8	68.2	79.4
Macro- avg	81.24	78.24	64.40	77.78

Conclusions

This paper discussed a neural network initialization technique for text categorization. It employs the use of neural networks initialized with decision tree classifier. Our study provides evidence that hybrid approach can be used for the construction of effective classifiers for automatic text categorization. We have presented a hybrid decision tree and neural network algorithm for building the classifier.

The method presented in this has some advantages over (Sethi, 1991; Banerji, 1997)

method. Decision tree size grows approximately linearly with increasingly larger amounts training set size. Neural networks constructed from these trees are often larger and complex than necessary. When the certainty factor value is appropriately tuned for the data set, error-based pruning can give trees that are essentially constant in size regardless of the increasingly larger amount of training data. The solution for choosing the certainty factor is given in this paper. This generally requires values of the certainty factor much smaller than the default value. Appropriate choice of certainty factor is able to produce trees that are essentially constant in size in the face of increasingly larger training sets. Experimental results support the conclusion that error based pruning can be used to produce appropriately sized trees, which are directly mapped to optimal neural network architecture with good accuracy.

This paper showed that hybrid decision tree and neural network approach improved accuracy in text classification task and are substantially better than single decision tree or neural network initialized randomly text classifiers performance comparable to previous results.

Although encouraging results have been obtained using hybrid approach based classifier, there is still much work remaining to be investigated. They include to create new decision tree construction algorithm for textual data and to determine how much it has an effect on hybrid classifier accuracy. These issues are left for our future works.

References

Banerjee, A. (1997). Initializing neural networks using decision trees. *Computational Learning Theory and Natural Learning Systems*, **IV**, 3–15.

Frakes, W., and R. Baeza–Yates (1992). *Information Retrieval*: *Data Structures & Algorithms*. Prentice Hall.

Haykin, S. (1994). *Neural Networks*: *A Comprehensive Foundation.* Macmillan College Publishing Comp., New York.

Yang, Y., and J. Pedersen (1997). A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, Nashville, US. pp. 412–420.

Yang, Y., and X. Liu (1999). A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, Berkeley, US. pp. 42–49.

Lewis, D.D., and M. Ringuette (1994). A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas. pp. 81–93.

Quinlan, J.R. (1993). *C4-5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

Raudys, S. (2001). *Statistical and Neural Classifiers*: *an Integrated Approach to Design.* Springer-Verlag, NY.

Raudys, S., and M. Skurichina (1992). The role of the number of training samples on weight initialization of artificial neural net classifier. In *Neuroinformatics and Neurocomputers. Proc. RNNS/IEEE Symposium*. Rostov-on-Don, Russia. pp. 343–353.

Rumelhart, D.E., and J.L. Mcclelland (1986). *Parallel Distributed Processing* 1. MIT Press, Cambridge, MA.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, **34**(1), 1–47.

Sethi, I.K. (1991). Decision tree performance enhancement using an artificial neural network implementation. In I.K. Sethi and A.K. Jain (Eds.), *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*. Elsevier, Amsterdam. pp. 71–88.

Wiener, E.D., J.O. Pedersen and A.S. Weigend (1995). A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas. pp. 317–332.

N. Remeikis is a doctoral student at Faculty of Informatics, Vytautas Magnus University. His research interest include intelligent information retrieval, classification and hybrid machine learning.

I. Skučas is a professor at Faculty of Informatics, Vytautas Magnus University. His research interests include mathematical modeling of the systems and adapting for design.

V. Melninkaitė is a docent at Faculty of Informatics, Vytautas Magnus University. Her research interests include data engineering and design of the systems.

Tekstinių dokumentų klasifikavimas naudojant neuroninius tinklus inicializuotus sprendimų medžiais

Nerijus REMEIKIS, Ignas SKUČAS, Vida MELNINKAITĖ

Vystantis informacinėms technologijoms vis didesnę reikšmę įgyja informacijos personalizavimas, organizavimas, valdymas ir sprendimų priėmimas projektavime. Dokumentų semantikos įvertinimas yra sudėtingas ir daugiareikšmis procesas. Straipsnyje nagrinejama daugiasluoksnio neuroninio tinklo, kurio architektūra ir pradiniai svoriai nustatomi naudojant sprendimo medžio klasifikatorių, taikymas tekstinių dokumentų klasifikavimo tikslumui padidinti. Didėjant apmokymui skirtų duomenų kiekiui sprendimų medžių dydis auga tiesiškai. Todėl iš šių sprendimų medžių sukonstruoti neuroniniai tinklai yra per daug dideli ir sudėtingi. Straipsnyje šią problemą siūloma spręsti naudojant klaidos įvertinimu paremtą sprendimų medžių mažinimą. Tinkamas tikrumo faktoriaus parinkimas leidžia stabilizuoti sprendimų medžio dydį didinant apmokymui skirtų duomenų aibę. Pateikiami pasiūlymai kaip tinkamai parinkti tikrumo faktorių sprendimų medžių mažinimui. Atlikto eksperimento rezultatai rodo, kad daugiasluoksnio neuroninio tinklo, inicializuoto sprendimų medžio klasifikatoriumi, pagalba galima pasiekti didesnį klasifikavimo tikslumą su standartiniu dokumentų rinkiniu Reuters-21578, kuris naudojamas informacijos klasifikavimo uždavinių eksperimentų įvertinimui. Gauti rezultatai palyginti su kitų autorių tyrimų rezultatais, gautais naudojant sprendimų medžių ir neuroninių tinklų klasifikatoriais. Numatomi tolimesni darbai susiję su kitų kriterijų informacijos semantikos įvertinimui tyrimu, bei pateiktų algoritmų modifikavimu.