

Predictive Self-Organizing Networks for Text Categorization

Ah-Hwee Tan

Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613
ahhwee@krdl.org.sg

Abstract. This paper introduces a class of predictive self-organizing neural networks known as Adaptive Resonance Associative Map (ARAM) for classification of free-text documents. Whereas most statistical approaches to text categorization derive classification knowledge based on training examples alone, ARAM performs supervised learning and integrates user-defined classification knowledge in the form of IF-THEN rules. Through our experiments on the Reuters-21578 news database, we showed that ARAM performed reasonably well in mining categorization knowledge from sparse and high dimensional document feature space. In addition, ARAM predictive accuracy and learning efficiency can be improved by incorporating a set of rules derived from the Reuters category description. The impact of rule insertion is most significant for categories with a small number of relevant documents.

1 Introduction

Text categorization refers to the task of automatically assigning documents into one or more predefined classes or categories. It can be considered as the simplest form of text mining in the sense that it abstracts the key content of a free-text documents into a single class label. In recent years, there has been an increasing number of statistical and machine learning techniques that automatically generate text categorization knowledge based on training examples. Such techniques include decision trees [2], K-nearest-neighbor system (KNN) [7, 19], rule induction [8], gradient descent neural networks [9, 16], regression models [18], Linear Least Square Fit (LLSF) [17], and support vector machines (SVM) [6, 7]. All these statistical methods adopt a supervised learning paradigm. During the learning phase, a classifier derives categorization knowledge from a set of pre-labeled or tagged documents. During the testing phase, the classifier makes prediction or classification on a separate set of unseen test cases. Supervised learning paradigm assumes the availability of a large pre-labeled or tagged training corpus. In specific domains, such corpora may not be readily available. In a personalized information filtering application, for example, few users would have the patience to provide feedback to a large number of documents for training the classifier. On the other hand, most users are willing to specify what they want explicitly. In such cases, it is desirable to have the flexibility of building a text classifier from examples as well as obtaining categorization knowledge directly from the users.

In machine learning literatures, hybrid models have been studied to integrate multiple knowledge sources for pattern classification. For example, Knowledge Based Artificial Neural Network (KBANN) refines imperfect domain knowledge using backpropagation neural networks [15]; Predictive self-organizing neural networks [4, 12] allows rule insertion at any point of the incremental learning process. Benchmark studies on several databases have shown that initializing such hybrid learning systems with prior knowledge not only improves predictive accuracy, but also produces better learning efficiency, in terms of the learning time as well as the final size of the classifiers [13]. In addition, promising results have been obtained by applying KBANN to build intelligent agents for web page classification [11].

This paper reports our evaluation of a class of predictive self-organizing neural networks known as Adaptive Resonance Associative Map (ARAM) [12] for text classification based on a popular public domain document database, namely Reuters-21578. The objective of our experiments is two-folded. First, we study ARAM's capability in mining categorization rules from sparse and high dimensional document feature vectors. Second, we investigate if ARAM's predictive accuracy and learning efficiency can be enhanced by incorporating a set of rules derived from the Reuters category description.

The rest of this article is organized as follows. Section 2 provides a brief introduction of ARAM. Section 3 describes our choice of feature selection and extraction methods. Section 4 presents the ARAM learning and classification algorithms. Section 5 presents the procedure for rule insertion. Section 6 reports the experimental results. The final section summarizes and concludes.

2 Adaptive Resonance Associative Map

ARAM belongs to a family of predictive self-organizing neural networks known as predictive Adaptive Resonance Theory (predictive ART) that performs incremental supervised learning of recognition categories (pattern classes) and multidimensional maps of patterns. An ARAM system can be visualized as two overlapping Adaptive Resonance Theory (ART) [3] modules consisting of two input fields F_1^a and F_1^b with an F_2 category field. For classification problems, the F_1^a field serves as the input field containing the input activity vector and the F_1^b field servers as the output field containing the output class vector. The F_2 field contains the activities of the recognition categories that are used to encode the patterns. During learning, given an input pattern presented at the F_1^a input layer and an output pattern presented at the F_1^b output field, a F_2 category node is selected to encode the pattern pair.

When performing classification tasks, ARAM formulates recognition categories of input patterns, and associates each category with its respective prediction. The knowledge that ARAM discovers during learning, is compatible with IF-THEN rule-based representation. Specifically, each node in the F_2 field represents a recognition category associating the F_1^a input patterns with the F_1^b output vectors. Learned weight vectors, one for each F_2 node, constitute a set of

rules that link antecedents to consequents. At any point during the incremental learning process, the system architecture can be translated into a compact set of rules. Similarly, domain knowledge in the form of IF-THEN rules can be inserted into ARAM architecture.

3 Features Selection/Extraction

As in statistical text categorization systems, ARAM adopts a bag-of-words approach to representing documents in the sense that each document is represented by a set of keyword features. ARAM keyword features can be obtained from two sources. Through rule insertion, a keyword feature can be specified explicitly by a user as an antecedent in a rule. Features can also be selected from the words in the training documents based on certain feature ranking metric. Some popularly used measures for feature ranking include keyword frequency, χ^2 statistics, and information gain. In our experiments, we only use χ^2 statistics which has been reported to be one of the most effective measures [20].

During rule insertion and keyword selection, we use an in-house morphological analyzer to identify the part-of-speech and the root form of each word. To reduce complexity, only the root forms of the noun and verb terms are extracted for further processing.

During keyword extraction, the document is first segmented and converted into a keyword feature vector

$$\mathbf{v} = (v_1, v_2, \dots, v_M). \quad (1)$$

where M is the number of keyword features selected. We experiment with three different document representation schemes described below.

tf scheme: This is the simplest and the first method that we used in earlier experiments [14]. The feature vector \mathbf{v} simply equals the term frequency vector tf such that the value of feature j

$$v_j = tf_j \quad (2)$$

where tf_j is the in-document frequency of the keyword w_j .

tf*idf scheme: A term weighting method based on *inverse document frequency* [10] is combined with the term frequency to produce the feature vector \mathbf{v} such that

$$v_j = tf_j \log_2 \frac{N}{df_j} \quad (3)$$

where N is the total number of documents in the collection and df_j is the number of documents that contains the keyword w_j .

log-tf*idf scheme: This is a variant of the *tf*idf* scheme. The feature vector \mathbf{v} is computed by

$$v_j = (1 + \log_2 tf_j) \log_2 \frac{N}{df_j}. \quad (4)$$

After encoding using one of the three feature representation schemes, the feature vector \mathbf{v} is normalized to produce the final feature vector

$$\mathbf{a} = \mathbf{v}/v_m \quad \text{where} \quad v_m \geq v_i \quad \forall i \neq m. \quad (5)$$

before presentation to the neural network classifier.

4 The Classifier

4.1 Learning

In an ARAM network (Figure 1), the unit for recruiting an F_2 category node is a complete pattern pair. Given a pair of patterns, the category field F_2 selects a winner that receives the largest overall input from the feature fields F_1^a and F_1^b . The winning node selected in F_2 then triggers a top-down priming on F_1^a and F_1^b , monitored by separate reset mechanisms. Code stabilization is ensured by restricting encoding to states where resonances are reached in both modules. By synchronizing the unsupervised categorization of two pattern sets, ARAM learns supervised mapping between the pattern sets. Due to the code stabilization mechanism, fast learning in a real-time environment is feasible.

The ART modules used in ARAM can be ART 1 [3], which categorizes binary patterns, or analog ART modules such as ART 2, ART 2-A, and fuzzy ART [5], which categorize both binary and analog patterns. The fuzzy ARAM model, that is composed of two overlapping fuzzy ART modules (Figure 1), is described below.

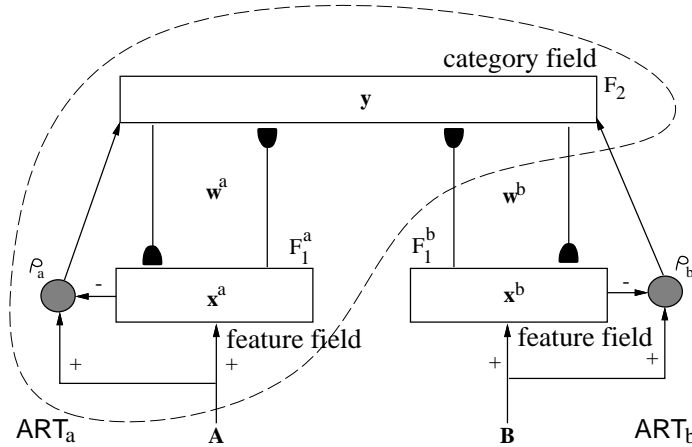


Fig. 1. The Adaptive Resonance Associative Map architecture.

Input vectors: Normalization of fuzzy ART inputs prevents category proliferation. The F_1^a and F_1^b input vectors are normalized by complement coding

that preserves amplitude information. Complement coding represents both the on-response and the off-response to an input vector \mathbf{a} . The complement coded F_1^a input vector \mathbf{A} is a $2M$ -dimensional vector

$$\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, \dots, a_M, a_1^c, \dots, a_M^c) \quad (6)$$

where $a_i^c \equiv 1 - a_i$.

Similarly, the complement coded F_1^b input vector \mathbf{B} is a $2N$ -dimensional vector

$$\mathbf{B} = (\mathbf{b}, \mathbf{b}^c) \equiv (b_1, \dots, b_N, b_1^c, \dots, b_N^c) \quad (7)$$

where $b_i^c \equiv 1 - b_i$.

Activity vectors: Let \mathbf{x}^a and \mathbf{x}^b denote the F_1^a and F_1^b activity vectors respectively. Let \mathbf{y} denote the F_2 activity vector. Upon input presentation, $\mathbf{x}^a = \mathbf{A}$ and $\mathbf{x}^b = \mathbf{B}$.

Weight vectors: Each F_2 category node j is associated with two adaptive weight templates \mathbf{w}_j^a and \mathbf{w}_j^b . Initially, all category nodes are uncommitted and all weights equal ones. After a category node is selected for encoding, it becomes *committed*.

Parameters: Fuzzy ARAM dynamics are determined by the choice parameters $\alpha_a > 0$ and $\alpha_b > 0$; the learning rates $\beta_a \in [0, 1]$ and $\beta_b \in [0, 1]$; the vigilance parameters $\rho_a \in [0, 1]$ and $\rho_b \in [0, 1]$; and a contribution parameter $\gamma \in [0, 1]$.

Category choice: Given the F_1^a and F_1^b input vectors \mathbf{A} and \mathbf{B} , for each F_2 node j , the choice function T_j is defined by

$$T_j = \gamma \frac{|\mathbf{A} \wedge \mathbf{w}_j^a|}{\alpha_a + |\mathbf{w}_j^a|} + (1 - \gamma) \frac{|\mathbf{B} \wedge \mathbf{w}_j^b|}{\alpha_b + |\mathbf{w}_j^b|}, \quad (8)$$

where the fuzzy AND operation \wedge is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i), \quad (9)$$

and where the norm $|\cdot|$ is defined by

$$|\mathbf{p}| \equiv \sum_i p_i \quad (10)$$

for vectors \mathbf{p} and \mathbf{q} .

The system is said to make a choice when at most one F_2 node can become active. The choice is indexed at J where

$$T_J = \max\{T_j : \text{for all } F_2 \text{ node } j\}. \quad (11)$$

When a category choice is made at node J , $y_J = 1$; and $y_j = 0$ for all $j \neq J$.

Resonance or reset: Resonance occurs if the *match functions*, m_J^a and m_J^b , meet the vigilance criteria in their respective modules:

$$m_J^a = \frac{|\mathbf{A} \wedge \mathbf{w}_J^a|}{|\mathbf{A}|} \geq \rho_a \quad \text{and} \quad m_J^b = \frac{|\mathbf{B} \wedge \mathbf{w}_J^b|}{|\mathbf{B}|} \geq \rho_b. \quad (12)$$

Learning then ensues, as defined below. If any of the vigilance constraints is violated, mismatch reset occurs in which the value of the choice function T_J is set to 0 for the duration of the input presentation. The search process repeats to select another new index J until resonance is achieved.

Learning: Once the search ends, the weight vectors \mathbf{w}_J^a and \mathbf{w}_J^b are updated according to the equations

$$\mathbf{w}_J^{a(\text{new})} = (1 - \beta_a)\mathbf{w}_J^{a(\text{old})} + \beta_a(\mathbf{A} \wedge \mathbf{w}_J^{a(\text{old})}) \quad (13)$$

and

$$\mathbf{w}_J^{b(\text{new})} = (1 - \beta_b)\mathbf{w}_J^{b(\text{old})} + \beta_b(\mathbf{B} \wedge \mathbf{w}_J^{b(\text{old})}) \quad (14)$$

respectively. For efficient coding of noisy input sets, it is useful to set $\beta_a = \beta_b = 1$ when J is an uncommitted node, and then take $\beta_a < 1$ and $\beta_b < 1$ after the category node is committed. *Fast learning* corresponds to setting $\beta_a = \beta_b = 1$ for committed nodes.

Match tracking: Match tracking rule as used in the ARTMAP search and prediction process [4] is useful in maximizing code compression. At the start of each input presentation, the vigilance parameter ρ_a equals a baseline vigilance $\overline{\rho}_a$. If a reset occurs in the category field F_2 , ρ_a is increased until it is slightly larger than the match function m_J^a . The search process then selects another F_2 node J under the revised vigilance criterion. With the match tracking rule and setting the contribution parameter $\gamma = 1$, ARAM emulates the search and test dynamics of ARTMAP.

4.2 Classification

In ARAM systems with category choice, only the F_2 node J that receives maximal $F_1^a \rightarrow F_2$ input T_j predicts ART_b output. In simulations,

$$y_j = \begin{cases} 1 & \text{if } j = J \text{ where } T_J > T_k \text{ for all } k \neq J \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

The F_1^b activity vector \mathbf{x}^b is given by

$$\mathbf{x}^b = \sum_j \mathbf{w}_j^b y_j = \mathbf{w}_J^b. \quad (16)$$

The output prediction vector \mathbf{B} is then given by

$$\mathbf{B} = (b_1, b_2, \dots, b_{2N}) = \mathbf{x}^b \quad (17)$$

where b_i indicates the confidence of assigning a pattern to category i .

5 Rule Insertion

ARAM incorporates a class of if-then rules that map a set of input attributes (antecedents) to a disjoint set of output attributes (consequents). The rules are conjunctive in the sense that the attributes in the IF clause and in the THEN clause have an *AND* relationship. Conjunctive rules has limited expressive power but is intuitive and reasonably comprehensive for representing simple heuristic for categorization.

ARAM rule insertion proceeds in two phases. The first phase parses the rules for keyword features. When a new keyword is encountered, it is added to a keyword feature table containing keywords obtained through automatic feature selection from training documents. Based on the keyword feature table, the second phase of rule insertion translates each rule into a $2M$ -dimensional vector \mathbf{A} and a $2N$ -dimensional vector \mathbf{B} , where M is the total number of features in the keyword feature table and N is the number of categories. Given a rule of the following format,

$$\text{IF } x_1, x_2, \dots, x_m, \neg y_1, \neg y_2, \dots, \neg y_n \quad \text{THEN } z_1, z_2, \dots, z_p$$

where x_1, \dots, x_m are the positive antecedents, y_1, \dots, y_n are the negative antecedents, and z_1, \dots, z_p are the consequents, the algorithm derives a pair of vectors \mathbf{A} and \mathbf{B} such that for each index $i = 1, \dots, M$,

$$(a_i, a_i^c) = \begin{cases} (1, 0) & \text{if } w_i = x_j \text{ for some } j \in \{1, \dots, m\} \\ (0, 1) & \text{if } w_i = y_j \text{ for some } j \in \{1, \dots, n\} \\ (1, 1) & \text{otherwise} \end{cases} \quad (18)$$

where w_i is the i^{th} entry in the keyword feature table; and for each index $i = 1, \dots, N$,

$$(b_i, b_i^c) = \begin{cases} (1, 0) & \text{if } w_i = z_j \text{ for some } j \in \{1, \dots, p\} \\ (0, 0) & \text{otherwise} \end{cases} \quad (19)$$

where w_i is the class label of the category i .

The vector pairs derived from the rules are then used as training patterns to initialize a ARAM network. During rule insertion, the vigilance parameters ρ_a and ρ_b are each set to 1 to ensure that only identical attribute vectors are grouped into one recognition category. Contradictory symbolic rules are detected during rule insertion when identical input attribute vectors are associated with distinct output attribute vectors.

6 Experiments: Reuters-21578

Reuters-21578 is chosen as the benchmark domain for a number of reasons. First, it is reasonably large, consisting of tens of thousands of pre-classified documents. Second, there is a good mix of large and small categories, in terms of the number of documents in the category. It enables us to compare ARAM's learning capability and the effect of rule insertion using different data characteristics. The

last but not the least, Reuters-21578 has been studied extensively in statistical text categorization literatures, allowing us to compare ARAM performance with prior arts.

To facilitate comparison, we used the recommended *ModApte* split (Reuters version 3) [1, 19] to partition the database into training and testing data. By selecting the 90 (out of a total of 135) categories that contain at least one training and one testing documents, there were 7770 training documents and 3019 testing documents.

6.1 Performance Measures

ARAM experiments adopt the most commonly used performance measures, including *recall*, *precision*, and the F_1 measure. *Recall* (r) is the percentage of the documents for a given category (i.e. topic) that are classified correctly. *Precision* (p) is the percentage of the predicted documents for a given category that are classified correctly. It is a normal practice to combine *recall* and *precision* in some way, so that classifiers can be compared in terms of a single rating. Two common ratings are the *break-even point* and the F_1 measure. *Break-even point* is the value at which *recall* equals *precision*. F_1 measure is defined as

$$F_1(r, p) = \frac{2rp}{r + p}. \quad (20)$$

These scores can be calculated for a series of binary classification experiments, one for each category, and then averaged across the experiments. Two types of averaging methods are commonly used: (1) *micro-averaging* technique that gives equal weights to each document; and (2) *macro-averaging* technique that gives equal weight to each category [19]. As micro-averaging F_1 scores are computed on a per-document basis, they tend to be dominated by the classifier’s performance on large categories. Macro-averaging F_1 scores, computed on a per-category basis, are more likely to be influenced by the classifier’s performance on small categories.

6.2 Learning and Classification

ARAM experiments used the following parameter values: choice parameters $\alpha_a = 0.1$, $\alpha_b = 0.1$; learning rates $\beta_a = \beta_b = 1.0$ for fast learning; contribution parameter $\gamma = 1.0$, and vigilance parameters $\rho_a = 0.8$, $\rho_b = 1.0$. Using a voting strategy, 10 voting ARAM produced a probabilistic score between 0 and 1. The score was then thresholded at a specific cut off point to produce a binary class prediction.

We fixed the number of keyword features at 100 determined empirically through earlier experiments. Null feature vectors and contradictory feature vectors were first removed from the training set before training. We experimented with all three feature representation schemes, namely *tf*, *tf*idf*, and *log-tf*idf*. Table 1 summarizes the performance of ARAM averaged across 90 categories of

Reuters-21578. Among the three feature representations, *log-tf*idf* produced the best performance in terms of micro-averaged F_1 . The *tf*idf* however performed better in terms of macro-averaged F_1 .

Table 1. Performance of ARAM using the three feature representation schemes in terms of micro-averaged recall, micro-averaged precision, micro-averaged F_1 and macro-averaged F_1 across all 90 categories of Reuters-21578.

Feature Representation	<i>miR</i>	<i>miP</i>	<i>miF₁</i>	<i>maF₁</i>
tf	0.8251	0.8376	0.8313	0.5497
tf*idf	0.8368	0.8381	0.8375	0.5691
log-tf*idf	0.8387	0.8439	0.8413	0.5423

6.3 Rule Insertion

A set of IF-THEN rules was crafted by the authors based on a description of the Reuters categories provided in the Reuters-21578 documentation (*cat-descriptions_120396.txt*). The rules simply linked the keywords mentioned in the description to their respective category labels. Creation of such rules was rather straight-forward. A total of 150 rules was created without any help from domain experts. They are generally short rules containing one to two keywords extracted from the category description. A partial set of rules is provided in Table 2 for illustration.

Table 2. An illustrative set of rules generated based on the category description.

acq	:- acquire acquisition
acq	:- merge merger
crude	:- crude oil
grain	:- grain
interest	:- interest
interest	:- rate
money-fx	:- foreign exchange
money-fx	:- money exchange

In the rule insertion experiments, rules were parsed and inserted into the ARAM networks before learning and classification. Table 3 compares the results

obtained by ARAM with and without rule insertion on the 10 most populated Reuters categories. The micro-averaged F_1 and the macro-averaged F_1 scores across the top 10 and all the 90 categories are also given. Eight out of the top 10 categories, namely *acq*, *money-fx*, *grain*, *crude*, *interest*, *ship*, *wheat*, and *corn*, showed noticeable improvement in F_1 measures by incorporating rules. Interestingly, one category, namely *trade*, produced worse results. No improvement is obtained for *earn*, the largest category. Overall improvement on the micro-averaged F_1 scores across the top 10 and all the 90 categories were 0.004 and 0.011 respectively. The improvement obtained on the macro-averaged F_1 scores, 0.006 for the top 10 and 0.055 for the 90 categories, were much more significant. This showed that rule insertion was most effective for categories with a smaller number of documents. The results were encouraging as even a simple set of rules was able to produce a noticeable improvement.

Table 3. Predictive performance of ARAM with and without rule insertion on Reuters-21578. N refers to the number of learning iterations for ARAM to achieve 100% accuracy on the training set. C refers to the number of ARAM recognition categories created. The last two rows show the *micro-averaged* F_1 and the *macro-averaged* F_1 across the top 10 and the 90 categories respectively. Boldfaced figures highlighted improvement obtained by rule insertion.

Category	Test Docs	No. of ARAM			ARAM w/rules		
		N	C	F_1	N	C	F_1
<i>earn</i>	1087	4.5	717.3	0.984	5.3	722.4	0.984
<i>acq</i>	719	5.1	732.0	0.930	6.1	732.9	0.938
<i>money-fx</i>	179	4.2	334.4	0.750	5.5	336.0	0.763
<i>grain</i>	149	4.7	104.5	0.895	5.0	108.3	0.906
<i>crude</i>	189	6.7	86.4	0.802	6.9	86.7	0.813
<i>trade</i>	117	6.0	257.2	0.689	6.8	260.9	0.661
<i>interest</i>	131	4.7	281.3	0.727	5.6	287.4	0.740
<i>ship</i>	89	4.6	46.8	0.793	4.8	48.7	0.796
<i>wheat</i>	71	4.2	87.7	0.789	8.1	89.0	0.803
<i>corn</i>	56	3.6	89.6	0.748	8.0	90.2	0.765
Top 10 (miF_1, maF_1)		(0.897,0.811)			(0.901,0.817)		
All 90 (miF_1, maF_1)		(0.841,0.542)			(0.852,0.597)		

Table 4 compares ARAM results with top performing classification systems on Reuters-21578 [19]. ARAM performed noticeably better than the gradient descent neural networks and the Native Bayes classifiers. Its miF_1 scores were comparable with those of SVM, KNN, and LLSF, but the maF_1 scores were significantly higher. As miF_1 scores are predominantly determined by the largest categories and maF_1 scores are dominated by the large number of small categories, the results indicated that ARAM performed fairly well for large categories and outperformed in small categories.

Table 4. Performance of ARAM compared with other top performing text classification systems across all 90 categories of Reuters-21578.

Classifiers	miR	miP	miF_1	maF_1
ARAM w/rules	0.8909	0.8155	0.8515	0.5967
ARAM	0.8961	0.7922	0.8409	0.5422
SVM	0.8120	0.9147	0.8599	0.5251
KNN	0.8339	0.8807	0.8567	0.5242
LLSF	0.8507	0.8489	0.8498	0.5008
Gradient descent NNet	0.7842	0.8785	0.8287	0.3765
Native Bayes	0.7688	0.8245	0.7956	0.3886

7 Conclusion

We have presented a novel approach to incorporate domain knowledge into a learning text categorization system. ARAM can be considered as a scaled down version of KNN. Increasing the ART_a vigilance parameter to 1.0 would cause ARAM's performance to converge to that of KNN with the price of storing all unique training examples. ARAM, therefore, is more scalable than KNN and is useful in situations when it is not practical to store all the training examples in the memory. Comparing with SVM, ARAM has the advantage of on-line incremental learning in the sense that learning of new examples does not require re-computation of recognition nodes using previously learned examples.

The most distinctive feature of ARAM, however, is its rule-based domain knowledge integration capability. The performance of ARAM is expected to improve further as good rules are added. The rule insertion capability is especially important when few training examples are available. This suggests that ARAM would be suitable for on-line text classification applications such as document filtering and personalization.

References

1. C. Apte, F. Damerau, and S.M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.
2. C. Apte, F. Damerau, and S.M. Weiss. Text mining with decision rules and decision trees. In *Proceedings of the Conference on Automated Learning and Discovery, Workshop 6: Learning from Text and the Web*, 1998.
3. G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.
4. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.

5. G. A. Carpenter, S. Grossberg, and D. B. Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.
6. S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representation for text categorization. In *Proceedings, ACM 7th International Conference on Information and Knowledge Management*, pages 148–155, 1998.
7. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings, 10th European Conference on Machine Learning (ECML'98)*, pages –, 1998.
8. D.D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings, Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94), Las Vegas*, pages 81–93, 1994.
9. H.T. Ng, W.B. Goh, and K.L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings, 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 67–73, 1997.
10. G. Salton and C Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
11. J.W. Shavlik and T. Eliassi-Rad. Building intelligent agents for web-based tasks: A theory-refinement approach. In *Working Notes of the Conf on Automated Learning and Discovery Workshop on Learning from Text and the Web, Pittsburgh, PA*, 1998.
12. A.-H. Tan. Adaptive Resonance Associative Map. *Neural Networks*, 8(3):437–446, 1995.
13. A.-H. Tan. Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing. *IEEE Transactions on Neural Networks*, 8(2):237–250, 1997.
14. A-H. Tan and Lai F-L. Text categorization, supervised learning, and domain knowledge integration. In *Proceedings, KDD-2000 International Workshop on Text Mining, Boston*, pages 113–114, 2000.
15. G. G. Towell, J. W. Shavlik, and M. O. Noordewier. Refinement of approximately correct domain theories by knowledge-based neural networks. In *Proceedings, 8th National Conference on AI, Boston, MA*, pages 861–866. AAAI Press/The MIT Press, 1990.
16. E. Wiener, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)*, 1995.
17. Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings, 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, 1994.
18. Y. Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.
19. Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings, 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49, 1999.
20. Y. Yang and J. P. Pedersen. Feature selection in statistical learning for text categorization. In *Proceedings, Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.

This article was processed using the L^AT_EX macro package with LLNCS style