



On Machine Learning Methods for Chinese Document Categorization

JI HE

School of Computing, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260

heji@comp.nus.edu.sg

AH-HWEE TAN

Nanyang Technological University, School of Computer Engineering, Blk N4, 2A-13 Nanyang Avenue, Singapore 639798

asahtan@ntu.edu.sg

CHEW-LIM TAN

School of Computing, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260

tancl@comp.nus.edu.sg

Abstract. This paper reports our comparative evaluation of three machine learning methods, namely k Nearest Neighbor (kNN), Support Vector Machines (SVM), and Adaptive Resonance Associative Map (ARAM) for Chinese document categorization. Based on two Chinese corpora, a series of controlled experiments evaluated their learning capabilities and efficiency in mining text classification knowledge. Benchmark experiments showed that their predictive performance were roughly comparable, especially on clean and well organized data sets. While kNN and ARAM yield better performances than SVM on small and clean data sets, SVM and ARAM significantly outperformed kNN on noisy data. Comparing efficiency, kNN was notably more costly in terms of time and memory than the other two methods. SVM is highly efficient in learning from well organized samples of moderate size, although on relatively large and noisy data the efficiency of SVM and ARAM are comparable.

Keywords: text categorization, machine learning, comparative experiments

1. Introduction

Text categorization refers to the task of automatically assigning one or multiple predefined category labels to free text documents. Whereas an extensive range of methods have been applied to English text categorization, relatively few benchmarks have been done for Chinese text. Typical approaches to Chinese text categorization, such as Naive Bayes (NB) [1], Vector Space Model (VSM) [2, 3] and Linear List Square Fit (LLSF) [4, 5], have well studied theoretical basis derived from the information retrieval research, but are not known to be the best classifiers [6, 7]. In addition, there is a lack of publicly available Chinese corpus for evaluating Chinese text categorization systems.

This paper reports our comparative evaluation of three machine learning methods, namely k Nearest Neighbor (kNN) [8], Support Vector Machines (SVM) [9], and Associative Resonance Associative Map (ARAM) [10] for Chinese text categorization. kNN and SVM have been reported as the top performing methods for English text categorization [7]. ARAM belongs to a popularly known family of predictive self-organizing neural networks [11] but until recently, has not been used for text categorization. Our comparative experiments employed two Chinese corpora, namely the TREC People's Daily news corpus (TREC) and the Chinese web corpus (WEB). Based on the benchmark experiments on these two corpora, we examined and compared the predictive accuracy as well as the efficiency of the three classifiers.

The rest of this paper is organized as follows. Section 2 describes our choice of the feature selection and extraction methods. Section 3 gives a summary of kNN and SVM, and reviews the less familiar ARAM algorithm in more details. Section 4 presents our evaluation paradigm and reports the experimental results. Section 5 discusses the results and compares the three classifiers in terms of predictive accuracy and efficiency. The last section summarizes our findings of the comparative experiments.

2. Feature Selection and Extraction

A pre-requisite of text categorization is to extract a suitable feature representation of the documents. Typically, word stems are suggested as the representation units by information retrieval research. However, unlike English and other Indo-European languages, Chinese text does not have a natural delimiter between words. As a consequence, word segmentation is a major issue in Chinese text processing. Chinese word segmentation methods have been extensively discussed in the literature. Unfortunately, perfect precision and disambiguation cannot be reached. As a result, the inherent errors caused by word segmentation always remain as a problem in Chinese information processing.

In our experiments, a word-class bi-gram model is adopted to segment each training document into a set of tokens. The lexicon used by the segmentation model contains over 64,000 words in 1,006 classes. High precision segmentation is not the focus of our work. Instead we aim to compare the various classifiers as long as the noises in the documents sets caused by word segmentation are reasonably low.

To select keyword features for classification, *CHI* (χ) statistics is adopted as the ranking metric in our experiments. A prior study on several well-known corpora including Reuters-21578 and OHSUMED has showed that CHI statistics generally outperforms other feature ranking measures, such as term strength (TS), document frequency (DF), mutual information (MI), and information gain (IG) [12]. For a token t , its CHI measure is defined by

$$\chi(t) = \sqrt{\frac{(n_{ct} + n_{\bar{c}t} + n_{c\bar{t}} + n_{\bar{c}\bar{t}})(n_{ct}n_{\bar{c}\bar{t}} - n_{\bar{c}t}n_{c\bar{t}})^2}{(n_{ct} + n_{\bar{c}\bar{t}})(n_{\bar{c}t} + n_{c\bar{t}})(n_{ct} + n_{\bar{c}t})(n_{c\bar{t}} + n_{\bar{c}\bar{t}})}} \quad (1)$$

where n_{ct} and $n_{\bar{c}t}$ are the number of documents in the positive category and the negative category

respectively¹ in which the token t appears at least once; and $n_{c\bar{t}}$ and $n_{\bar{c}\bar{t}}$ are the number of documents in the positive category and the negative category respectively in which the token t doesn't appear. A set of tokens with the highest CHI measures are then selected as keyword features $\{w_i \mid i = 1, 2, \dots, M\}$, where M is the number of keywords in the feature set.

During feature extraction, the document is first segmented and converted into a keyword frequency vector $(tf_1, tf_2, \dots, tf_M)$, where tf_i is the in-document term frequency of keyword w_i . A term weighting method based on *inverse document frequency* (IDF) [13] and L1-normalization are then applied on the frequency vector to produce the keyword feature vector

$$\mathbf{x} = \frac{(x_1, x_2, \dots, x_M)}{\max\{x_i\}}, \quad (2)$$

in which x_i is computed by

$$x_i = (1 + \log_2 tf_i) \log_2 \frac{n}{n_i}, \quad (3)$$

where n is the number of documents in the whole training set and n_i is the number of training documents in which the keyword w_i appears at least once.

3. Classifiers

3.1. *K* Nearest Neighbor

k Nearest Neighbor (kNN) is a traditional statistical pattern recognition algorithm [8]. It has been studied extensively for text categorization [7]. In essence, kNN makes prediction based on the k training patterns that are closest to the unseen (testing) pattern, according to a distance metric. The commonly used distance metrics that measure the similarity between two normalized patterns include the Euclidean distance

$$D(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_i (p_i - q_i)^2}, \quad (4)$$

the inner product

$$D(\mathbf{p}, \mathbf{q}) = \sum_i p_i q_i, \quad (5)$$

and the cosine similarity

$$D(\mathbf{p}, \mathbf{q}) = \frac{\sum_i p_i q_i}{\sqrt{\sum_i p_i^2} \sqrt{\sum_i q_i^2}}. \quad (6)$$

The class assignment to the test pattern is based on the class(es) of the closest k training patterns. A commonly used method is to label the test pattern with the class that has the most instances among the k nearest neighbors. Specifically, the class index $y(x)$ assigned to the test pattern \mathbf{x} is given by

$$y(x) = \operatorname{argmax}_i \{n(\mathbf{x}_j, c_i) \mid \mathbf{x}_j \in kNN\}, \quad (7)$$

where $n(\mathbf{x}_j, c_i)$ is the number of training patterns \mathbf{x}_j in the k nearest neighbor set that are associated with class c_i .

3.2. Support Vector Machines

Support Vector Machine (SVM) is a relatively new class of machine learning techniques first introduced by Vapnik [9]. Based on the *structural risk minimization* principle of the computational learning theory, SVM seeks a decision surface to separate the training data points into two classes and makes decisions based on the *support vectors* that are selected as the only effective elements from the training set.

Given a set of linearly separable points $S = \{\mathbf{x}_i \mid i = 1, 2, \dots, N\}$, each point \mathbf{x}_i belonging to one of the two classes labelled as $y_i \in \{-1, +1\}$, a *separating hyper-plane* divides S into two sides, each containing points with the same class label only. The separating hyper-plane can be identified by the pair (\mathbf{w}, b) that satisfies

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \quad (8)$$

for any data point \mathbf{x} on the hyper-plane and

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad (9)$$

for any training sample $\mathbf{x}_i \in S$. The goal of the SVM learning is to find the *optimal separating hyper-plane (OSH)* that has the maximal margin to both sides. This can be formulated as:

$$\begin{aligned} &\text{minimize} && \frac{1}{2} \|\mathbf{w}\|^2 \\ &\text{subject to} && y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \end{aligned} \quad (10)$$

The points that are closest to the OSH are termed *support vectors* (Fig. 1).

During classification, SVM makes decision based on the globally optimized separating hyper-plane. It simply finds out on which side of the OSH the test

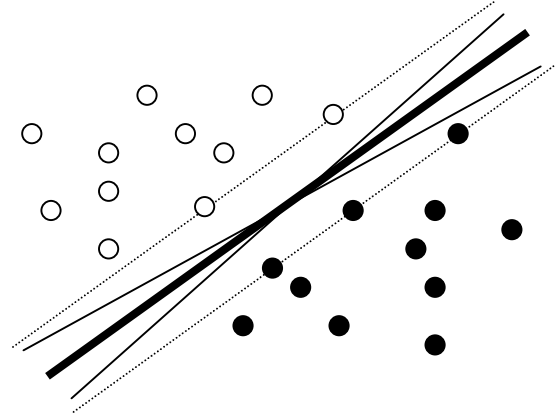


Figure 1. Separating hyper-planes (the set of solid lines), optimal separating hyper-plane (the bold solid line), and support vectors (data points on the dashed lines). The dashed lines identify the maximum margin.

pattern is located. This property makes SVM highly competitive, compared with other traditional pattern recognition methods, in terms of predictive accuracy and efficiency [7].

The SVM problem can be extended to non-linear case using non-linear hyper-plane based on *convolution function*, such as the polynomial function

$$K(\mathbf{p}, \mathbf{q}) = (\gamma(\mathbf{p} \cdot \mathbf{q}) + c)^d \quad (11)$$

and the radial basis function (RBF)

$$K(\mathbf{p}, \mathbf{q}) = e^{-\gamma \|\mathbf{p} - \mathbf{q}\|^2} \quad (12)$$

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

Various quadratic programming algorithms have been proposed and extensively studied to solve the SVM problem. In recent years, Joachims has done much research on the application of SVM to text categorization. His *SVM^{light}* system² based on the decomposition idea of Osuna et al. [14] has been proven to be practical in learning from relatively high dimensional and large-scale training set [15].

3.3. Adaptive Resonance Associative Map

Adaptive Resonance Associative Map (ARAM) is a class of predictive self-organizing neural networks [11] 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

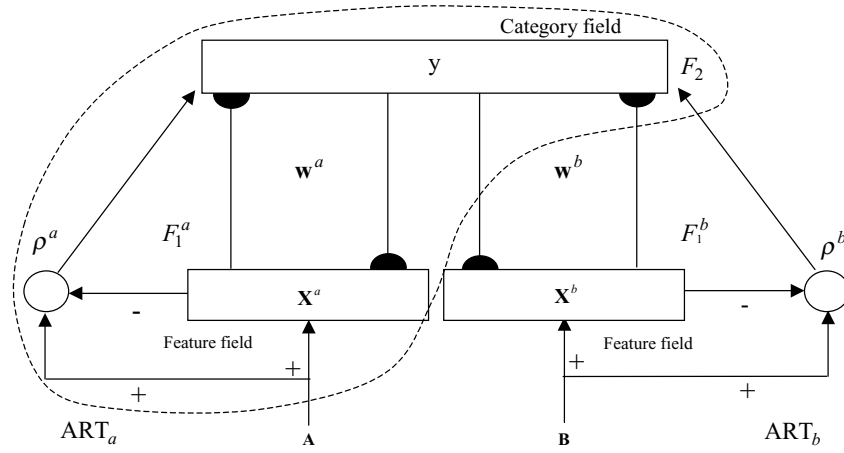


Figure 2. The Adaptive Resonance Associative Map architecture.

Theory (ART) modules consisting of two input fields F_1^a and F_1^b with an F_2 category field [10, 16] (Fig. 2). For classification problems, the F_1^a field serves as the input field containing the feature vector and the F_1^b field serves as the output field containing the class prediction vector. The F_2 field contains the activities of the recognition categories that are used to encode the patterns.

When performing classification tasks, ARAM formulates recognition categories of input patterns and associates each category with its respective prediction. During learning, given an input pattern (document features) presented at the F_1^a input layer and an output pattern (class label) presented at the F_1^b output field, the category field F_2 selects a winner that receives the largest overall input. 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 resonance 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, which categorizes binary patterns, or analog ART modules such as ART 2, ART 2-A, and fuzzy ART, which categorize both binary and analog patterns. The fuzzy ARAM [10] algorithm based on fuzzy ART [11] is summarized below.

Parameters: The 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 $\rho_b \in [0, 1]$; the contribution parameter $\gamma \in [0, 1]$; and the k-max decision parameter k .

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

$$\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) \equiv (a_1, \dots, a_M, a_1^c, \dots, a_M^c) \quad (13)$$

where $a_i^c \equiv 1 - a_i$. Similarly, the complement coded F_1^b input vector \mathbf{B} is a $2N$ -dimensional vector

$$\mathbf{B} = (\mathbf{b}, \mathbf{b}^c) \equiv (b_1, \dots, b_N, b_1^c, \dots, b_N^c) \quad (14)$$

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

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*.

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

$$T_j = \gamma \frac{|\mathbf{A} \wedge \mathbf{w}_j^a|}{\alpha_a + |\mathbf{w}_j^a|} + (1 - \gamma) \frac{|\mathbf{B} \wedge \mathbf{w}_j^b|}{\alpha_b + |\mathbf{w}_j^b|}, \quad (15)$$

where the fuzzy AND operation \wedge is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv \min(p_i, q_i), \quad (16)$$

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

$$|\mathbf{p}| \equiv \sum_i p_i \quad (17)$$

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

$$T_J = \max\{T_j : \text{for all } F_2 \text{ node } j\}. \quad (18)$$

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:

$$m_J^a = \frac{|\mathbf{A} \wedge \mathbf{w}_J^a|}{|\mathbf{A}|} \geq \rho_a \quad (19)$$

and

$$m_J^b = \frac{|\mathbf{B} \wedge \mathbf{w}_J^b|}{|\mathbf{B}|} \geq \rho_b. \quad (20)$$

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

$$\mathbf{w}_J^{a(\text{new})} = (1 - \beta_a)\mathbf{w}_J^{a(\text{old})} + \beta_a(\mathbf{A} \wedge \mathbf{w}_J^{a(\text{old})}) \quad (21)$$

and

$$\mathbf{w}_J^{b(\text{new})} = (1 - \beta_b)\mathbf{w}_J^{b(\text{old})} + \beta_b(\mathbf{B} \wedge \mathbf{w}_J^{b(\text{old})}) \quad (22)$$

respectively. *Fast learning* corresponds to setting $\beta_a = \beta_b = 1$ for committed nodes.

K-max decision rule: During classification, ARAM works in the spirit of kNN system. Using a k-max rule,

the output is predicted by a set of k F_2 nodes with the largest $F_1^a \rightarrow F_2$ input T_j . The F_2 activity values y_j first normalized by

$$y_j = \begin{cases} T_j / \sum_{k \in \pi} T_k & \text{if } j \text{ in } \pi \\ 0 & \text{otherwise,} \end{cases} \quad (23)$$

where π is the set of k category nodes with the largest input T_j . The F_1^b activity vector \mathbf{x}^b is computed by

$$\mathbf{x}^b = \sum_j \mathbf{w}_j^b y_j. \quad (24)$$

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

$$\mathbf{B} \equiv (b_1, b_2, \dots, b_N) = \mathbf{x}^b, \quad (25)$$

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

Voting strategy: As a drawback inherited from ART, ARAM is sensitive to the input order of the training samples. To tackle this problem, multiple ARAM classifiers can be trained using the same set of patterns in different orders of presentation. During classification, the output vectors of multiple ARAM are combined to yield a final output vector

$$\mathbf{B} = \frac{\sum \mathbf{B}_v}{V}, \quad (26)$$

where V is the number of voting ARAMs and \mathbf{B}_v is the output vector produced by voter v .

4. Evaluation Experiments

4.1. Performance Measures

Our experiments adopted the commonly used performance measures, including *recall*, *precision*, and F_1 measures. Given a testing set \mathbf{A} containing documents pre-labelled with category c and a prediction set \mathbf{B} labelled with category c by the classifier, the recall (R) and precision (P) measures are defined by

$$R = \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A}|} \quad (27)$$

and

$$P = \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{B}|} \quad (28)$$

respectively, where the norm $|\cdot|$ denotes the size of the document set [17]. It is a common practice to combine recall and precision in some way so that classifiers can be compared in terms of a single rating. Our experiments used F_1 rating, a measure that gave equal weights to R and P . The F_1 measure, first introduced by Rijsbergen [17] and subsequently used in Yang's relative comparison experiments [7], is defined by

$$F_1 = \frac{2|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A}| + |\mathbf{B}|}. \quad (29)$$

In terms of P and R , the F_1 value is

$$F_1 = \frac{2RP}{R + P}. \quad (30)$$

We notice in the information retrieval literatures, *break-even point* [18], defined as the value where $R = P$, was more widely used than F_1 measure. However, our experiments showed that for a given classifier, its break-even point was relatively difficult to reach in a limited number of experiments using a fixed set of parameters. Our empirical experiments also showed that if R was reasonably close to P , F_1 had approximately the same distribution as break-even point.

The benchmark on each corpus was simplified into multiple binary classification experiments. In each experiment, a chosen category was tagged as the positive category and the other categories in the same corpus were combined as the negative category. Micro-averaged scores and macro-averaged on the whole corpus were then produced across the experiments. With micro-averaging, the performance measures were computed across the documents by adding all the document counts across the different tests and calculating using these summed values. With macro-averaging, each category was assigned with the same weight and the performance measures were calculated across the categories. It is understandable that micro-averaged scores and macro-averaged scores reflect a classifier's performance on large categories and small categories respectively [7].

4.2. Significance Test

To compare the performance between two systems, we employed the combined 5x2cv F test [19] as the significance test measure based on the micro-averaged and macro-averaged F_1 values. The combined 5x2cv F test is a variance of the 5x2cv paired t test introduced by

Ditterich [20], which in turn is a slight improvement to the k -fold cross-validated paired t test. The combined 5x2cv F test can be summarized as follow.

The null hypothesis for the significance test states that on a randomly drawn training set, two learning algorithms \mathbf{A} and \mathbf{B} will have the same performance measure (in this case the micro-averaged and macro-averaged F_1 scores) on the testing set. A 5x2cv test contains five replications of two-fold cross-validation. In each replication, samples are randomly partitioned into two disjoint sets S_1 and S_2 . Both sets contain equal-sized positive samples as well as negative samples. Algorithms \mathbf{A} and \mathbf{B} are then trained on each set and tested on the other set. Each replication produces four predictive scores, namely $p_A^{(1)}$ and $p_B^{(1)}$ by training on S_1 and testing on S_2 ; and $p_A^{(2)}$ and $p_B^{(2)}$ by training on S_2 and testing on S_1 . Based on these four scores, two estimated differences $p^{(1)} = p_A^{(1)} - p_B^{(1)}$ and $p^{(2)} = p_A^{(2)} - p_B^{(2)}$ are calculated. They further lead to the estimated variance $s^2 = (p^{(1)} - \bar{p})^2 + (p^{(2)} - \bar{p})^2$, where $\bar{p} = (p^{(1)} + p^{(2)})/2$. The combined 5x2cv f score is then defined by

$$f = \frac{\sum_{i=1}^5 \sum_{j=1}^2 (p_i^{(j)})^2}{2 \sum_{i=1}^5 s_i^2}, \quad (31)$$

where s_i^2 is the variance computed from the i -th replication, and $p_i^{(j)}$ is the $p^{(j)}$ value from the i -th replication for $j = 1, 2$.

Alpaydm showed that under the null hypothesis, f approximated F distribution with 10 and 5 degrees of freedom [19], where the significance levels 0.05 and 0.01 corresponded to the two thresholds $f_0 = 4.74$ and $f_1 = 10.05$ respectively. Given a combined 5x2cv f score computed based on the performance of a pair of class A and B, we compared f with threshold values f_0 and f_1 to determine if A is superior to B at significance levels of 0.05 and 0.01 respectively.

Alpaydm's report showed that the combined 5x2cv F test had lower Type I error and higher power than the 5x2cv t test. In Ditterich's comparison experiments, the 5x2cv t test slightly outperformed other widely used statistical tests, such as t test based on random train/test splits and t test based on 10-fold cross-validation [20].

4.3. TREC-5 People's Daily News Corpus

The TREC-5 People's Daily news corpus is a subset of the Mandarin News Corpus announced by the Linguistic Data Consortium (LDC) in 1995. 77,733 documents

in the corpus cover a variety of topics, including international and domestic news, sports, and culture. Since the corpus was originally intended for evaluating information retrieval systems, each document was assigned a class label in our experiments for the purpose of text categorization. The labelling process was based on the topic information contained in the header field of each document, which were manually labelled in the original newspaper and retained in the corpus as fields *cat* and *headline*. Documents in the corpus were first automatically clustered into 101 groups under a simple rule that documents in each group contained the same *cat* label. With manual review of each group, groups with too general contents or containing too few articles were discarded; small groups with similar contents were merged. 6 top-level categories were then generated based on the contents of the remaining 33,047 documents, namely Politics, Law and Society (*Poli*), Literature and Arts (*Arts*), Education, Science and Culture (*Edu*), Sports (*Spts*), Theory and Academy (*Acad*), and Economics (*Eco*). To obtain a clean data set for the purpose of our experiments, the corpus was further filtered and reduced to an even smaller size that each category contained 600 documents only.³

Our experiments compared the classifiers' performance based on training/testing sets of different sizes. On a chosen category, a varying number of positive samples (ranged from 100 to 600) and double number of negative samples evenly distributed in the other five categories were randomly selected. They were then

evenly split into training and testing folds as in the 5x2cv tests. The document features used in each test were fixed at the 1,500 features selected from the 600 positive samples and 1,200 negative samples for each category.

kNN experiments used cosine distance as the similarity measure. Optimal k values for training sets of different sizes were determined by cross-validations using varying k values ranging from 1 to 39 and the best results were recorded. Preliminary experiments showed that SVM with RBF kernel significantly outperformed linear kernel on all categories. Hence further experiments used RBF kernel by setting $\gamma = 0.1$. Other SVM parameters used the default built-in values in *SVM^{light}*. ARAM experiments employed the default parameter values: learning rates $\beta_a = \beta_b = 1.0$; contribution parameter $\gamma = 1.0$; vigilance parameter $\rho_b = 1.0$; and $k = 1$ in category prediction. The vigilance parameter ρ_a was set to 0.9 for high precision. In addition, using a voting strategy, five ARAM systems trained using the same set of patterns in different orders of presentation were combined to yield a final prediction vector.

Figure 3 depicts the predictive performance of the three classifiers in terms of micro-averaged F_1 measures. Since each batch of the controlled experiments on the TREC corpus used the same number of training/testing samples on all categories, it was understandable that the macro-averages scores equaled to the micro-averaged scores.

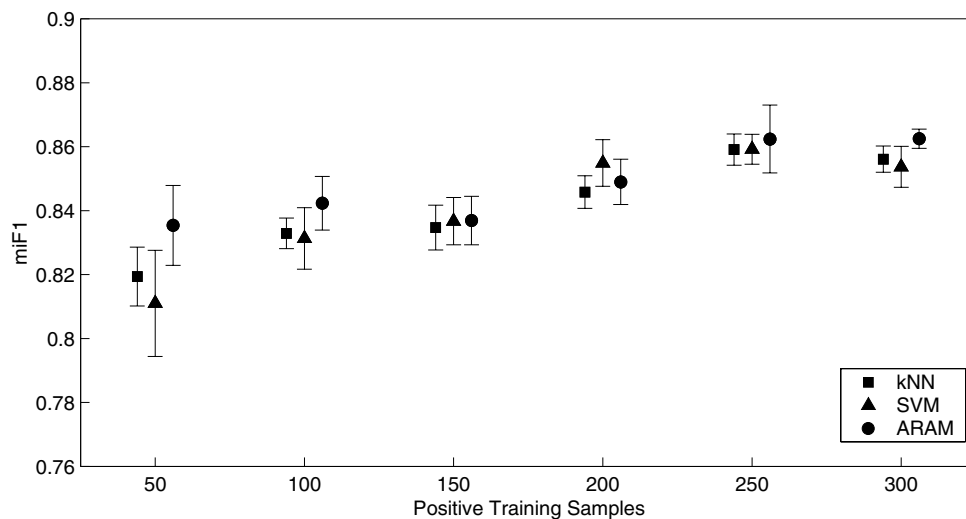


Figure 3. Micro-averaged F_1 measures of kNN, SVM, and ARAM on the TREC corpus, using 50 to 300 positive training samples for each category. Error bars donate the standard deviation across ten tests.

Table 1. Significance of cross-classifier comparisons. P denotes the number of positive training samples. “>” denotes better than at significance level 0.05; “~” denotes no significant difference.

| P | Statistical significance |
|---------|--------------------------|
| 50 | ARAM > kNN > SVM |
| 100 | ARAM > {kNN, SVM} |
| 150–300 | ARAM ~ kNN ~ SVM |

Regardless of the size of the training sets, the performance produced by the three classifiers were rather good (with F_1 scores of 0.80 and above). Whereas ARAM and kNN outperformed SVM on small training sets (100 or less positive samples), the three classifiers demonstrated very similar performance with training sets of moderate size (150 and above positive samples). Our observations were supported by the significance test results based on the one-to-one cross-classifier comparisons as summarized in Table 1.

4.4. Chinese Web Corpus

The Chinese web corpus (WEB) (Table 2) collected in-house consists of over 8,436 web pages downloaded from various Chinese web sites covering a wide variety of topics. Compared with the TREC corpus, the documents in this corpus are much noisier due to the great variety among web pages in terms of document length, style, and content. In addition, a number of documents are assigned with multiple (two or three) category labels. This makes categorization task on this corpus very challenging.

Table 2. The eight top-level categories of the WEB corpus sorted by the category size. P and N indicate the number of the positive and negative samples respectively.

| Category | Description | P | N |
|-------------|----------------------------|-------|-------|
| <i>Biz</i> | Business | 4,102 | 4,334 |
| <i>IT</i> | Computer and internet info | 1,719 | 6,717 |
| <i>Joy</i> | Online entertainment info | 1,231 | 7,205 |
| <i>Arts</i> | Literature, arts | 587 | 7,849 |
| <i>Edu</i> | Education | 422 | 8,014 |
| <i>Med</i> | Medical care related info | 305 | 8,121 |
| <i>Blf</i> | Philosophy and religion | 228 | 8,208 |
| <i>Sci</i> | Various kinds of science | 211 | 8,325 |

Table 3. Cross-classifier performance comparisons based on micro-averaged F_1 and macro-averaged F_1 scores. “>>” denotes better than at significance level 0.01; “>” denotes better than at significance level 0.05.

| | |
|------------------|--------------------|
| Micro-ave. F_1 | {ARAM, SVM} > kNN |
| Macro-ave. F_1 | {ARAM, SVM} >> kNN |

For experiments on the WEB corpus, the number of document features was fixed at 500 as we tended to get very low CHI values beyond the first 500 features. Cross-validations suggested an optimal k value of 3 for kNN. The size of cache for SVM^{light} kernel evaluations was doubled from 40 mega bytes to 80 mega bytes in order to improve the training speed on large training sets. In addition, ARAM’s vigilance parameter ρ_a was decreased to 0.7 to keep the number of recognition categories manageable.

Figure 4 shows the three classifiers’ performance on the eight categories in terms of F_1 measures. Figure 5 reports their micro-averaged and macro-averaged F_1 scores across all categories. The scores produced by ARAM and SVM were roughly comparable. Their micro-averaged F_1 scores predominantly determined by the performance on the large categories such as *Biz* and *IT*, were significantly higher than that of kNN. The performance difference in terms of macro-averaged F_1 scores were even more significant. These observations harmonized with the significance test results show in Table 3.

5. Discussions

5.1. Predictive Accuracy

On relatively large and well-organized categories (such as those from the TREC corpus), all the three classifiers demonstrated rather similar performance. This suggests that, as long as we have a sufficient number of clean training patterns, the three learning methods under evaluation can produce reasonably good generalization performance for Chinese text classification. The different approaches adopted by the trio in learning categorization knowledge are best seen in the light of the distinct learning peculiarities they exhibit on the small and noisy training sets.

kNN is a lazy learning method in the sense that it does not carry out any off-line learning to generate a particular category knowledge representation. Instead, kNN performs on-line scoring to find the training patterns

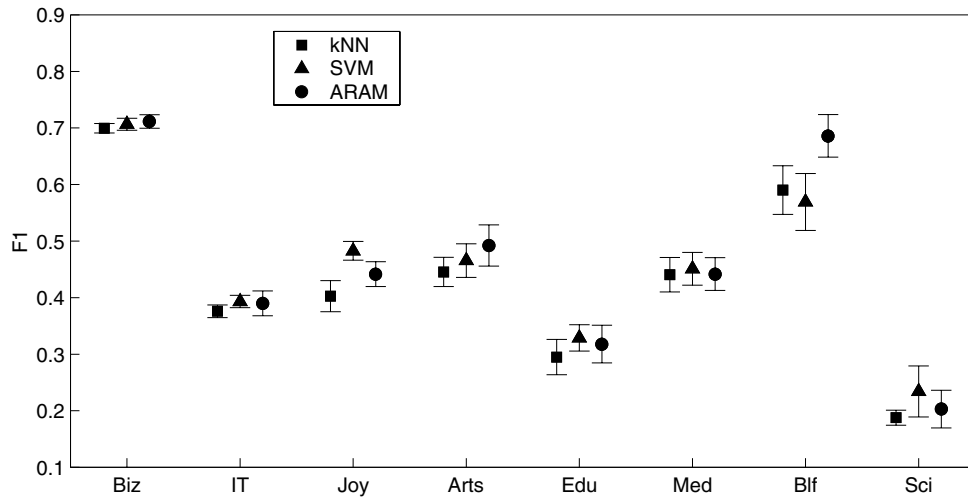


Figure 4. F_1 measures of kNN, SVM, and ARAM on the eight categories of the WEB corpus. Error bars donate the standard deviation across ten tests.

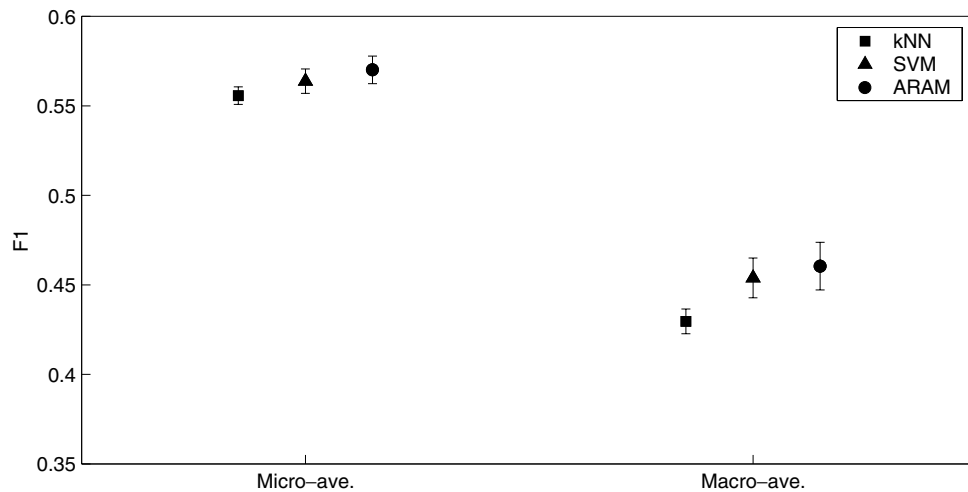


Figure 5. Micro-averaged and macro-averaged F_1 measures of kNN, SVM, and ARAM on the WEB corpus. Error bars donate the standard deviation across ten tests.

that are nearest to a test pattern and makes the decision based on the statistical presumption that patterns in the same category have similar feature representation. The presumption is generally true for most well-organized pattern sets. Thus kNN exhibits a relatively satisfactory predictive accuracy across small and large categories from the TREC corpus. However, due to the inherent noises of the individual nearest neighbors, kNN performed significantly worse than SVM and ARAM on noisy training sets from the WEB corpus. In addition, since kNN prediction is strictly localized to the testing pattern's neighbors, it is difficult to decide an optimal k

value without empirical experiments. Our experiments showed that optimal k was affected by the size and quality of the training samples as well as the ratio of the positive sample size over the negative sample size. Typically optimal k values have to be determined via a number of cross-validations.

SVM identifies global optimal separating hyperplanes (*OSH*) across the training data points and makes classification decisions based on representative data instances (known as *support vectors*). Because the decision boundary is globally optimized, SVM shows very accurate prediction with training sets of reasonable

size. However, in the absence of sufficient and clean training instances, the *OSH* generated may not be very representative. Thus on small training sets of the TREC corpus, SVM's performance was slightly lower than that of kNN. In addition, the use of different SVM kernel could give great impact to SVM's predictive accuracy. Our experiments showed that SVM with linear kernel performed significantly worse than RBF kernel on the two Chinese corpora. With preliminary knowledge to decide a kernel that best fits the training sample distribution, SVM can produce rather good performance.

ARAM follows the spirits of nearest-rectangle based algorithms by generating recognition categories from the training patterns. The incrementally learned rules abstract the key attributes of the training patterns and eliminate minor inconsistencies in the data patterns. During classifying, it works in a fashion similar to that of kNN. The major difference is that ARAM uses the learned recognition categories as the similarity scoring unit whereas kNN uses the raw in-processed training patterns. It follows that ARAM exhibits similar predictive accuracy as kNN but is more robust in learning from relatively noisy data set.

Our experimental results on kNN and SVM are slightly different from those reported in the English categorization literatures. Joachims showed that RBF SVM significantly outperformed kNN [21]. Our experiments however did not produce noticeable difference between these two classifiers' performance on the TREC corpus. This may be that we optimized k values for kNN according to the size of the training set while Joachims used a single k value across all categories. For WEB corpus that utilized a fixed k value, our comparative results harmonized with Joachims' reports. Yang and Liu mentioned that in their experiments, linear SVM produced slightly better results than the non-linear models [7]. Our experiments, however, showed that RBF SVM consistently performed better than linear SVM on the two Chinese corpora.

5.2. Efficiency

Besides predictive accuracy, we are interested in the efficiency of the classifiers. An analysis of the memory and time complexity of the classifiers used in our experiments is given below.

Let P and Q be the numbers of training and testing instances respectively and M be the number of the document features. Using a naive implementation, the time

complexity of kNN learning is $O(PM)$, in the sense that it simply stores the features of all the input training samples. During testing, it is $O(PQM)$. The memory complexity of kNN's learning and prediction can be estimated to be $O(PM)$. When P is large, kNN prediction can be rather slow and costly in terms of time as well as memory.

The time complexity of SVM learning is dominated by the constraint solving algorithm. SVM learning is generally rather costly. Algorithms applied in *SVM^{light}* to save memory resource and increase learning speed includes sparse vector representation (storing non-zero document features only), learning task shrinking (reducing a high-dimensional learning task into several low-dimensional sub tasks), and kernel evaluation buffering. Hence *SVM^{light}* turned out to be very efficient in our experiments. Let D be the dimension of the sub learning tasks, F be the maximum number of non-zero features in any of the training examples, E be the number of kernel evaluations, and V be the number of support vectors. The time complexity of *SVM^{light}* learning is estimated to be $O(EDPF)$ [15]. During testing it is $O(VQM)$. The memory complexity of *SVM^{light}*, during learning and testing, can be estimated to be $O(PD + D^2)$ and $O(VM)$ respectively. Compared with kNN, SVM shows a significantly better predictive efficiency as V is generally much smaller than P . However, shrinking a P -dimensional learning task into several D -dimensional sub tasks increases the number of kernel evaluations E . On large and noisy training sets, the time cost of SVM learning may be noticeably higher.

The time complexity of ARAM learning is affected by the number of recognition categories C and the number of learning iterations I . It can be estimated as $O(IPCM)$. During testing, the time complexity is $O(CQM)$. The memory cost during learning or prediction can be estimated as $O(CM)$. Compared with SVM, ARAM learning is significantly less resource intensive due to its incremental learning property. In addition, it is more efficient than kNN during prediction since C is much smaller than P .

Table 4 compares the efficiency of the three classifiers on several selected categories in terms of the CPU times used during training and testing. Figures for the TREC corpus were based on experiments using 300 positive training samples. Training time for kNN was not reported as the time cost in storing training features was effectively zero after the I/O operation time was excluded.

Table 4. Efficiency of kNN, SVM, and ARAM on selected categories of the TREC and WEB corpora. Categories are presented in increasing order of size and decreasing order of quality. T_{trn} and T_{tst} indicate the training and testing time (in seconds) respectively. E indicates the number of kernel evaluations. SV indicates the number of support vectors. I indicates the number of learning iterations. RC indicates the number of recognition categories.

| Category | kNN | | SVM | | | ARAM | | | |
|-----------|-----------|-----------|-------|-----------|-----------|------|------|-----------|-----------|
| | T_{tst} | E | SV | T_{trn} | T_{tst} | I | RC | T_{trn} | T_{tst} |
| TREC-Poli | 80.0 | 360,066 | 519 | 1.19 | 0.80 | 2 | 62 | 13.5 | 7.3 |
| TREC-Edu | 80.0 | 398,380 | 573 | 1.70 | 0.96 | 2 | 65 | 14.3 | 8.6 |
| WEB-Arts | 308.0 | 2,077,688 | 540 | 10.53 | 1.66 | 3 | 27 | 6.3 | 2.7 |
| WEB-IT | 314.0 | 5,046,739 | 1,894 | 41.79 | 19.96 | 3 | 137 | 53.9 | 21.1 |

The time cost of kNN was fairly consistent across categories containing approximately the same number of training and testing samples, despite of the varying k values and the characteristics of the document sets. As the number of training or testing samples increased, the time cost of kNN increased linearly. On clean categories of moderate scale such as those in the TREC corpus, SVM demonstrated outstanding efficiency over ARAM and kNN. ARAM in turn was noticeably faster than kNN. This suggests that as long as the document set is clean and the category is well defined, SVM is the clear winner. On relatively large and noisy categories such as those in the WEB corpus, however, the efficiency of SVM and ARAM were roughly comparable and significantly higher than that of kNN. This shows that SVM and ARAM would be better choices for large document set.

6. Conclusions

We have presented extensive experimental results on the evaluation of the three machine learning methods, namely kNN, SVM, and ARAM, based on the two Chinese corpora. The key findings of our empirical experiments are summarized as follows.

- Given sufficient number of training patterns of good quality, all three methods can produce satisfactory generalization performance on unseen test documents.
- On small and well-organized training sets, kNN and ARAM produce similar performance superior to that of SVM. In other words, kNN and ARAM seem to be more capable than SVM in extracting categorization knowledge from relatively small and clean training sets. ARAM and SVM however are more robust than kNN on large and noisy training sets.

- Comparing efficiency, kNN is perhaps the most inefficient classifier among the trio. On clean training sets of moderate scale, SVM shows an efficiency unmatched by kNN and ARAM. On large scale or noisy training sets, however, the efficiency of ARAM and SVM are roughly comparable.

Acknowledgments

We would like to thank our colleagues, J. Su and G.-D. Zhou, for providing the Chinese segmentation software and F.-L. Lai for valuable suggestions in designing the experiments. We thank T. Joachims for making *SVM^{light}* available. In addition, we are very grateful to the anonymous referees for the useful comments to a previous version of the manuscript.

Notes

1. Without loss of generalization, our experiments refer to binary classification tasks.
2. *SVM^{light}* is available via <http://ais.gmd.de/thorsten/smv.light/>
3. The TREC corpus mentioned in the following refers to the reconstructed document subset associated with category labels.

References

1. L. Zhu, "The theory and experiments on automatic chinese documents classification," *Journal of the China Society for Scientific and Technical Information (in Chinese)*, vol. 6, no. 6, 1987.
2. T. Zou, Y. Huang, and F. Zhang, "Technology of information mining on WWW," *Journal of the China Society for Scientific and Technical Information (in Chinese)*, vol. 18, no. 4, pp. 291–295, 1999.
3. T. Zou, J.-C. Wang, Y. Huang, and F.-Y. Zhang, "The design and implementation of an automatic chinese documents classification system," *Journal for Chinese Information Processing (in Chinese)*, vol. 12, no. 2, 1998.

4. S.-Q. Cao, F.-H. Zeng, and H.-G. Cao, "A mathematical model for automatic chinese text categorization," *Journal of the China Society for Scientific and Technical Information (in Chinese)*, vol. 18, no. 1, pp. 27–32, 1999.
5. Y. Yang, "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval," in *17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)*, 1994.
6. Y. Yang, "An evaluation of statistical approaches to text categorization," *Journal of Information Retrieval*, vol. 1, nos. 1/2, pp. 67–88, 1999.
7. Y. Yang and X. Liu, "A re-examination of text categorization methods," in *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 42–49, 1999.
8. B.V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*, IEEE Computer Society Press: Las Alamitos, California, 1991.
9. C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.
10. A.-H. Tan, "Adaptive resonance associative map," *Neural Networks*, vol. 8, no. 3, pp. 437–446, 1995.
11. 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*, vol. 4, pp. 759–771, 1991.
12. Y. Yang and J.P. Pedersen, "A comparative study on feature selection in text categorization," in *Fourteenth International Conference on Machine Learning (ICML'97)*, pp. 412–420, 1997.
13. G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.
14. E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Neural Networks for Signal Processing VII—Proceeding of the 1997 IEEE Workshop*, New York, pp. 276–285, 1997.
15. T. Joachims, "Making large-scales SVM learning practical," in *Advances in Kernel Methods—Support Vector Learning*, edited by B. Scholkopf, C. Burges and A. Smola, MIT Press, 1999.
16. A.-H. Tan, "Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 237–235, 1997.
17. C.J. van Rijsbergen, *Information Retrieval*, Butterworths: London, 1979.
18. D.D. Lewis, "Representation and learning in information retrieval," PhD thesis, Graduate School of the University of Massachusetts, 1992.
19. E. Alpaydin, "Combined 5x2cv F test for comparing supervised classification learning algorithms," *Neural Computation*, vol. 11, no. 8, pp. 1885–1892, 1999.
20. T.G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation*, vol. 10, no. 7, pp. 1895–1924, 1998.
21. T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of the European Conference on Machine Learning*, Springer, 1998.



Ji He is a Ph.D. candidate in the Department of Computer Science, School of Computing, National University of Singapore. His research interests include knowledge discovery, domain knowledge integration, and text mining. He received his Bachelor's degree in Electronic Engineering and Master's degree in Information Management from Shanghai Jiaotong University in 1997 and 2000 respectively.



Ah-Hwee Tan is an Associate Professor in the School of Computer Engineering at Nanyang Technological University. He was a Research Manager and Senior Member of Research Staff at the Kent Ridge Digital Labs, Laboratories for Information Technology, and Institute for Infocomm Research, where he led R&D projects in knowledge discovery, document analysis, and information mining. He received his Ph.D. in Cognitive and Neural Systems from Boston University in 1994. Prior to that, he obtained his Bachelor of Science (First Class Honors) (1989) and Master of Science (1991) in Computer and Information Science from the National University of Singapore. He is an editorial board member of *Applied Intelligence* and a member of Singapore Computer Society, ACM, and ACM SIGKDD.



Chew-Lim Tan is an Associate Professor in the Department of Computer Science, School of Computing, National University of Singapore. His research interests are expert systems, document image and text processing, neural networks and genetic programming. He obtained a B.Sc. (Hons) degree in Physics in 1971 from the University of Singapore, an M.Sc. degree in Radiation Studies in 1973 from the University of Surrey, U.K., and a Ph.D. degree in Computer Science in 1986 from the University of Virginia, U.S.A.