

A Maximal Figure-of-Merit (MFoM) Learning Approach to Robust Classifier Design for Text Categorization

Sheng Gao

Institute for Infocomm Research, Singapore

Wen Wu

Language Technologies Institute, Carnegie Mellon University

Chin-Hui Lee

School of Electrical and Computer Engineering, Georgia Institute of Technology

and

Tat-Seng Chua

School of Computing, National University of Singapore

ABSTRACT

We propose a maximal figure-of-merit learning (MFoM) approach for robust classifier design, which directly optimizes performance metrics of interest for different target classifiers. The proposed approach, embedding the decision functions of classifiers and performance metrics into the overall training objective, learns the parameters of classifiers in a decision-feedback manner to effectively take into account of both positive and negative training samples, and therefore reduce the required size of the positive training data. It has three desirable properties: (a) it is a performance metric oriented learning; (b) the optimized metric is consistent in both training and evaluation sets; and, (c) it is more robust and less sensitive to data variation, and can handle insufficient training data scenario. We evaluate it on the text categorization task using the Reuters-21578 dataset. Training a F_1 -based binary tree classifier using MFoM, we observed significantly improved performance and enhanced robustness compared to the baseline and SVM, especially on categories with insufficient training samples. The generality for designing other metric-based classifiers is also demonstrated by comparing the precision, recall, and F_1 -based classifiers. The results clearly show the consistency in performance between the training and evaluation stages for each classifier, and MFoM optimizes the chosen metric.

Authors' present addresses: Sheng Gao, Institute for Infocomm Research, 21 Heng Mui Keng Terrace, 119613, Singapore, email: gaosheng@i2r.a-star.edu.sg; Wen Wu, Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, email: wenwu@cs.cmu.edu; Chin-Hui Lee, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA, email: chl@ece.gatech.edu; Tat-Seng Chua, School of Computing, National University of Singapore, 3 Science Drive 2, 117543, Singapore, email: chuats@comp.nus.edu.sg.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; I.5.2 [Pattern Recognition]: Design Methodology

General Terms: Algorithms, Performance, Experimentation, Theory

Additional Key Words and Phrases: Text categorization, information retrieval, maximal figure-of-merit, generalized probabilistic descent method, latent semantic indexing, decision tree

1. INTRODUCTION

Text categorization (TC) is a process of classifying a text document into some pre-defined categories. It is an important research problem in information retrieval (IR), information extraction and filtering, and natural language processing. In the past two decades TC has received much attention (Sebastiani 2002). A number of machine learning approaches to TC have been proposed. They are *Bayesian method* (Lewis, *et al.* 1994, Makoto, *et al.* 1995, McCallum, *et al.* 1998), *k-nearest neighbors (kNN)* (Masand, *et al.* 1992, Yang 1994, Yang, *et al.* 1999), *Rocchio algorithm* (Buckley, *et al.* 1994, Joachims 1997), *artificial neural networks (ANN)* (Guo, *et al.* 1992, Liu, *et al.* 2001, Ng, *et al.* 1997, Ruiz, *et al.* 1999, Schutze, *et al.* 1995), *support vector machines (SVM)* (Cortes, *et al.* 1995, Joachims, 1998, Joachims, 2002), *boosting* (Schapire, *et al.* 2000), *decision tree (DT)* (Breiman, *et al.* 1984, Brodley, *et al.* 1995, Guo, *et al.* 1992, Lewis, *et al.* 1994, Quinlan 1993, Utgoff, *et al.* 1991), and *hidden Markov model (HMM)* (Denoyer, *et al.* 2001, Miller, *et al.* 1999). Some software packages, such as C4.5¹, CART², and SVM³, are also widely available to the researchers.

For a given classification problem, some metrics are often chosen to evaluate the performance of the classifiers. These measures may be different from one application to another. For instance, precision, recall and F_1 measures are widely used in the TC and IR communities. For evaluating speaker verification performance, a combination of false alarms and false detection errors, called detection cost function, is commonly adopted. However, the above metrics only measure the performance of classifiers at one chosen operating point of the ROC (Receiver operating characteristic) curve. To measure the performance of classifier on the overall operating points, the use of full ROC curve is a good candidate. Recently, there is a growing interest in machine learning community to study the ROC related metrics such as the AUC (Area under the ROC curve) for evaluation and classifier design (Fawcett, 2003; Flach, 2003; Yan, *et al.*, 2003). One possible empirical estimation of AUC is the mean average precision (MAP), which is a measure adopted in TREC Video Retrieval Evaluation⁴ to evaluate the ranking performance of classifiers for semantic visual concept detection. In order to cater to different application requirements and the need to measure the performance of classifiers based on

¹ <http://www.cse.unsw.edu.au/~quinlan/>

² <http://www.salford-systems.com/>

³ <http://www.ece.umn.edu/groups/ece8591/software/svm.html>

⁴ <http://www-nlpir.nist.gov/projects/tv2005/tv2005.html>

different performance metrics, it is desirable for classifier designers to have a flexible system training tool that is geared to optimizing a given performance measure based on the application requirements. However, most conventional techniques train the classifiers using the same algorithms for all real-world applications without concerning the required evaluation criteria. Instead, they train the classifier by tuning a set of parameters empirically.

In this paper we proposed a *Maximal Figure-of-Merit* (MFoM) approach to overcoming the above problems by optimizing different performance metrics of interest directly (Gao, *et al.* 2003). MFoM works by: (a) optimizing a given classifier for a chosen metric; (b) providing a unified framework to facilitate the formulation of the metric oriented objective function; and (c) maximizing the margin between the decision boundaries for robustness. It has a consistent learning objective for both training and testing with different performance evaluation metric of interest for a given application. Some smoothing functions are defined to directly integrate the desired discrete quantities (e.g. recall, precision and F_1) into a continuous and differentiable objective function of the classifier parameters. The key lies in formulating a one-dimensional quantity, called class misclassification function, which measures the degree of separation between the desired class and the competing classes. The formulation is flexible and applicable to both two-class and multi-class classification problems. By embedding the overall training objective into the decision functions used by the classifiers, MFoM is capable of learning the parameters of the classifiers in a decision-feedback manner to effectively take into account of both positive and negative training samples, and therefore reduce the required size of the training set. A generalized probabilistic descent algorithm is applied to solve the highly nonlinear optimization problem. Compared with other learning approaches, MFoM has the following advantages: 1) it can be tailored to different performance metrics for text categorization given the interest of users; 2) it has consistency of the optimized metric on both training set and evaluation set; and 3) it can deal with insufficient training data because it is less sensitive to data variation. We evaluate MFoM on the Reuters-21578 text categorization task. In all the experiments, we adopt a binary decision tree classifier with a linear discriminant function trained by MFoM at each internal node. Moreover, to demonstrate the generality of MFoM for designing classifiers for different metrics, we train the MFoM classifiers based on precision, recall, and F_1 respectively. The results clearly show that the performance of each classifier is consistent between the training stage and evaluation stage, and that the classifier is able to obtain the best chosen metric on the testing data.

The idea of optimizing performance metrics of interest directly is not only important for researchers working in the area of text categorization, but also quite useful for a wider audience who are working on other problems, such as spam filtering, automated workflow, document retrieval, insider threat detection etc. Given a task, an appropriate metric can be chosen for it and an optimal system can be designed accordingly using MFoM. In order to better study and discuss the properties of MFoM, we particularly focus on the application of MFoM to text categorization on the Reuters dataset in the paper. We expect readers are triggered by the methodology of the proposed approach and apply it to their particular problems.

To learn a classifier, most of the approaches aim at learning the parameters of a joint distribution, $P(X, C)$, between the observed feature vector, X , of a document and its corresponding category, C . It is well known that if $P(X, C)$ is specified exactly, an optimal classifier can be designed to minimize the Bayes risk (e.g. Duda, *et al.*, 2001; Lee, *et al.* 2000). Unfortunately, for most real-world problems, the exact form is usually unavailable. Even we are lucky to be given the distribution form, its parameters would still need to

be estimated from a labeled training set. ANN and SVM are two popular discriminative learning approaches to training TC classifiers. They do not directly estimate the joint distributions. Instead, the SVM attempts to learn the hyper-planes that maximally separate the different categories and its estimation is usually based on a structural risk minimization principle, while the ANN framework attempts to map the input feature vectors provided by the training samples to some desired category labels by minimizing a mean-squared-error between the designated labels and the network outputs. In contrast to ANN and SVM, the Bayesian learning principle tries to learn the joint distribution, $P(X, C)$. To reduce the complexity of the design, $P(X, C)$ is usually decomposed into two learning components, $P(X/C)$ and $P(C)$, known as the conditional class and prior class distributions, respectively. In most pattern recognition scenarios, the latter is assumed to be equally probable and ignored (Lewis, *et al.* 1994, Makoto, *et al.* 1995, McCallum, *et al.* 1998). The former is generally assumed to be a distribution with a known form, for example multinomial densities in naïve Bayes approaches for text applications (Lewis, *et al.* 1994, Makoto, *et al.* 1995, McCallum, *et al.* 1995), and mixture Gaussian densities in HMM for automatic speech and speaker recognition (Lee, *et al.* 1996). *Maximum likelihood* (ML) techniques are then used to estimate the corresponding distribution parameters. When the actual testing data do not support this assumed conditional class distribution, the system performance is often degraded. To overcome this deficiency, a new *minimum classification error* (MCE) framework (Juang, *et al.* 1997, Katagiri, *et al.* 1998, Kuo *et al.* 2003, Lee, *et al.* 2000) was proposed to directly minimize the empirical error of the training set. Nonlinear optimization techniques, such as *generalized probabilistic descent* (GPD) algorithms (Katagiri *et al.* 1998), were used to solve the classifier parameters in an iterative manner. The MCE techniques attempt to estimate the decision boundaries between the competing classes without the need to estimate the distribution forms and their parameters. It is therefore applicable to any classifier and any class conditional distributions. This is similar to the ANN and SVM approaches discussed earlier. Furthermore, a measure of the expected separation between the competing classes is also explicitly maximized in the optimization process, and therefore the classifier performance and robustness are both enhanced. MCE and the related techniques have been successfully applied to many speech and speaker recognition problems (Juang, *et al.* 1997, Katagiri, *et al.* 1998). Recently it was extended to the vector-based call routing and document retrieval (Kuo *et al.* 2003).

There are plenty of TC studies available in the literature. The reader is referred to a recent tutorial (Sebastiani, 2002) for an in-depth look at the state-of-the-art TC techniques. The remainder of the paper is organized as follows. The next section introduces the theory of MFoM. Section 3 lays out the iterative GPD algorithm for training MFoM tree classifiers. In section 4 shows the experimental results based on the Reuters-21578 task and their detailed analyses. Brief discussions on non-linear classifier and convexity are presented in section 5. Finally we summarize our findings in section 6.

2. MAXIMAL FIGURE-OF-MERIT LEARNING APPROACH

Most evaluation metrics of interest in classification are the discrete functions of the error counts, and they are often not differentiable with respect to the classifier parameters, so some smooth approximation is needed to embed the given classifier decision rules into the overall objective functions for optimization. Motivated by the above observation, we propose the maximal figure-of-merit (MFoM) learning approach that can optimize different performance metrics of interest given different target classifiers.

2.1 Performance Metrics

A set of metrics is needed to evaluate the performance of the classification systems. To achieve the best performance for the different applications with different requirements, it is important to take into account the effect of these metrics on the design of the classifier. Here we only discuss three measures (precision, recall, and F_1 measure) widely used in TC. They are denoted by P_j , R_j , and F_j for a class C_j , respectively (here the subscript j is the category index), and defined as

$$P_j = \frac{TP_j}{TP_j + FP_j} \quad (2.1)$$

$$R_j = \frac{TP_j}{TP_j + FN_j} \quad (2.2)$$

$$F_j = \frac{2P_jR_j}{R_j + P_j} = \frac{2TP_j}{FP_j + FN_j + 2TP_j} \quad (2.3)$$

TP_j (true positive): no. of documents correctly assigned;

FP_j (false positive): no. of documents falsely accepted;

FN_j (false negative): no. of documents falsely rejected.

From these definitions it is clear that these metrics are discrete quantities for counting errors. It is well known that the above performance measures are closely related to the classifiers. However, this relationship is implicit without any explicit continuous and differentiable function. Therefore, it is impossible to estimate the parameter of the classifier by optimizing these measures directly. Here we will discuss some approximation functions to fix the above problems.

2.2 Approximating Overall Performance Metrics

To formulate a continuous and differentiable performance metric, a class loss function, $l_j(X;W)$, is defined for each class C_j to approximate the error counts. Its value should be close to 0 for correct, and 1 for incorrect classification, respectively. Clearly this loss is a function of the document feature vector, X , and the classifier parameters, W . We can then estimate the true positive, false positive and false negative functions for C_j by summing over all samples in the training set T as follows:

$$TP_j \approx \sum_{X \in T} (1 - l_j(X;W)) \cdot 1(X \in C_j) \quad (2.4)$$

$$FP_j \approx \sum_{X \in T} (1 - l_j(X;W)) \cdot 1(X \notin C_j) \quad (2.5)$$

$$FN_j \approx \sum_{X \in T} l_j(X;W) \cdot 1(X \in C_j) \quad (2.6)$$

where $I(A)$ is an indicator function of any logical expression, A . As for the choice of an appropriate loss function, any smooth 0-1 function of a one-dimensional variable approximating a step function at the origin will be sufficient. A sigmoid function shown below is often adopted (Juang, *et al.* 1997, Katagiri, *et al.* 1998, Kuo, *et al.* 2003, Lee, *et al.* 2000),

$$l_j(X;W) = \frac{1}{1 + e^{-[\alpha d_j(X;W) + \beta]}} \quad (2.7)$$

where $d_j(X;W)$ is a class misclassification function as defined in Eq.(2.8), α is a positive constant that controls the size of the learning window and the learning rate, and β is a constant measuring the offset of $d_j(X;W)$ from 0. For simplicity, we name the value of $d_j(X;W)$ as d -value in the next.

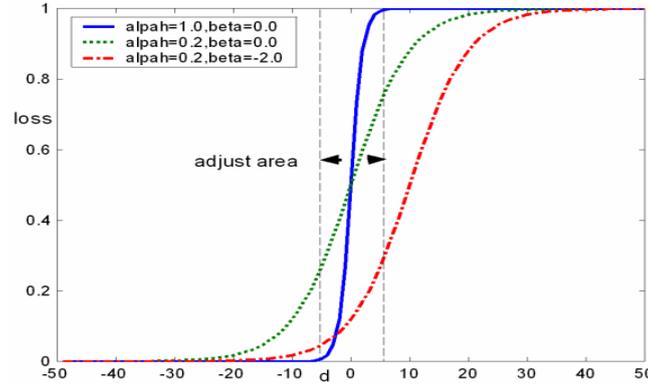


Figure 1 The setting of sigmoid function parameters

Figure 1 illustrates three sigmoid function curves in Eq. (2.7) with three different settings for α and β . Also shown is the adjusted area (known as the learning window near the decision boundary) for the parameter setting, $\alpha = 1$ and $\beta = 0$. Only the training samples with the d -value falling in this area can be effectively used to update the classifier parameters in MFoM. For others, the gradient of Eq. (2.7) is close to 0 (see Section 3 for further discussion). This property is similar to SVM, where only support vectors having effects on the decision boundary are adjusted. Since the range of the variation for the d -value is different for the various real-world classification problems, it is necessary to select a suitable α empirically so that $\alpha d_j(X;W) + \beta$ falls in the adjusted area for most training samples. If α is set very high, the curve in the learning window will become very steep, thus the adjusted area becomes too narrow. The setting for β is related to finding the location of the decision boundary between the two competing classes. Their values can be set according to the distribution of $d_j(X;W)$ in the training set. With more samples located in the adjusted area, better learning is achieved.

To complete the definitions in Eq.(2.4)-(2.7), the final step is to define the class misclassification function, $d_j(X;W)$, that is negative for a correct decision and positive otherwise. In the binary tree classifiers, and with the linear discriminant function (LDF) decision rules at every internal node (See Section 3), we have the following natural choice (Note: $j \in \Theta$, the set of all classes, with $\Theta = \{C+, C-\}$) for the binary case:

$$\begin{cases} d_j(X;W) = -f(X,W) & \text{for } X \in C^+ \\ d_j(X;W) = f(X,W) & \text{for } X \in C^- \end{cases} \quad (2.8a)$$

$$f(W,X) = \sum_{k=1}^R w_k x_k - w_0 = W^T \cdot X \quad (2.8b)$$

where R is the feature dimension, $W = [w_0, w_1, \dots, w_R]^T$ the augmented weight vector which should be learned from the training data, and $X = [-1, x_1, x_2, \dots, x_R]^T$ the augmented feature vector for a document.

For a single-level binary tree classifier, the measure in Eq.(2.8a) serves as a simple way to embed the classifier into the performance metrics of interest. However in the multi-level binary tree classifiers we are designing, there is no simple way of defining an overall misclassification measure in a global manner. Instead we repetitively use the function in Eq. (2.8a) in every internal node of the tree. This results in a need to optimize the node level performance metrics in a recursive manner. Although a local optimization at each individual node does not necessarily lead to a global optimization of the entire tree, such a node-by-node optimization process of the performance metrics have produced satisfactory results, just like in the case of designing of the multi-level binary tree classifiers.

Now we are ready to formulate an overall objective function to serve as a figure-of-merit for a given application. For example, using the class F_1 measure defined in Eq. (2.3), we can easily define an overall approximate F_1 measure as follows ($|\Theta|$: the total number of categories):

$$F(T;W) = \frac{1}{|\Theta|} \sum_{j \in \Theta} F_j = \frac{1}{|\Theta|} \sum_{j \in \Theta} \frac{2TP_j}{FP_j + FN_j + 2TP_j} \quad (2.9)$$

The optimal classifier can be obtained by optimizing Eq. (2.9). In this paper, we study the binary tree classifiers for multiple category classification, where N binary classifiers are trained independently. However, it is difficult to simultaneously learn a set of LDF classifiers. A feasible solution used in our experiments is to train each LDF classifier in each node independently for maximizing its F_1 measure only using the partial training samples entered into it.

Other metrics, such as recall and precision of both positive and negative samples, can also be used when training the LDF classifier. One example is to define the following approximated error rate at each node using the MCE objective function (L : the total number of samples in the training corpus, T),

$$L(T;W) = \frac{1}{L} \sum_{X \in T} \sum_{j \in \Theta} l_j(X;W) \cdot 1(X \in C_j) \quad (2.10)$$

Minimizing the above equation will generate a classifier that optimizes the minimal classification error rate.

Similarly, interested readers can design their objective functions with respect to other metrics such as AUC, MAP, etc.. Furthermore, MFoM also can be exploited to solve the regression problem. Since the loss function can be used to explain how much probability of the target is mis-predicted by the model, it is feasible to apply it to the regression problem. For example, when the regression error is mapped using the loss function such as Eq. (2.7), an objective function for regression errors can be defined as Eq. (2.10), and is solved using the MFoM algorithm. Similarly, in case of the KL-divergence as a distance measure between the correct

distribution and predicted one, this distance can also be mapped using the loss function and then the MFoM can be used to solve the model estimation. However, further study should be done for efficiency and effectiveness of the MFoM learning in other domains. It is out of our discussion in the paper.

2.3 From Binary to Multiple-Category Classifiers

The MFoM formulation for training binary classifiers can be extended to more complex cases, e.g. designing discriminative multi-category classifiers. For a decision rule that classifies a given document X into one of N categories, a popular choice is to maximize over all class scores, i.e.

$$C(X) = \arg \max_{j \in \Theta} [g_j(X; W)], \quad 1 \leq j \leq N \quad (2.11)$$

where $C(X)$ is the class label assigned by the decision rule, $g_j(X; W)$ is a class discriminant function to compute the score for class j , and W is the entire set of parameters of all classifiers. Such rules have been used extensively in other applications, such as automatic speech recognition in which one out of many possible sentences are chosen as the most likely recognized string. This of course cannot be easily accomplished by conventional binary classifiers. In multi-category classification, a correct classification decision is made for a document X coming from the class C_j , if $C(X) = C_j$, i.e. $\forall i$

$$g_j(X; W) > g_{i \neq j}(X; W) \quad X \in C_j \quad (2.12)$$

One way to embed the discrete decision rule in Eq. (2.11) into an optimization objective function is to define a one-dimensional class misclassification function, $d_j(X; W)$, such that Eq. (2.12) is equivalent to $d_j(X; W) < 0$ in the case of correct decision. This can be accomplished as follows by (Juang, *et al.* 1997, Katagiri, *et al.* 1998, Kuo, *et al.* 2003, Lee, *et al.* 2000):

$$d_j(X; W) = -g_j(X; W) + \left[\frac{1}{|\Theta| - 1} \left\{ \sum_{i, i \neq j} g_i^\eta(X; W) \right\} \right]^{1/\eta} \quad (2.13)$$

where η is a positive constant. The second term of the RHS above is the geometric average of all the scores from other competing classes. It is noted that when η approaches ∞ , the term converges to the highest score among the competing ones. In general, Eq. (2.13) < 0 if Eq. (2.12) is true, but not vice versa. Intuitively the absolute value of the quantity in Eq. (2.13) characterizes the separation between the correct and competing classes. We can maximize this separation by minimizing the expected value of $d_j(X; W)$ or any non-decreasing function of it. Again the same loss function, as defined in Eq. (2.7), can be used to approximate the error. As an illustration for the MCE case, the total error could be approximated similarly as shown in Eq. (2.10). This is also the overall objective function to be optimized. GPD algorithms can then be used to find the solution. There are no other known formulations capable of directly solving this type of discriminative multi-category classification problems. We have developed such an MFoM learning approach to multi-category TC. This new framework and some experimental results have been reported in Gao, *et al.*, 2004.

3. ALGORITHMS FOR LEARNING MFOM CLASSIFIERS

We select the binary tree classifier as the baseline classifier in the paper and focus on training the binary tree classifiers by optimizing each internal node of the tree using MFoM. Readers can choose other classifiers of interest to implement the corresponding MFoM classifiers using the same methodology discussed above. We start from the binary tree generation algorithm.

3.1 Learning the Baseline Binary Tree Classifier

In the community of TC, the multi-category classification problem is often solved by independently designing a set of N *binary classifiers*, each only determines if a document is relevant to a specific category. Each binary classifier is trained from a category-based training set, which is obtained by re-labeling each document in the training corpus T with either the positive class, $C+$, if it is relevant to the category, or the negative class, $C-$, if irrelevant. Figure 2 shows an example of the binary tree classifier. Each leaf node is given a relevance label (positive or negative). All training samples that enter into it are classified according to the node label. Recursive splitting is then performed to build the tree node-by-node.

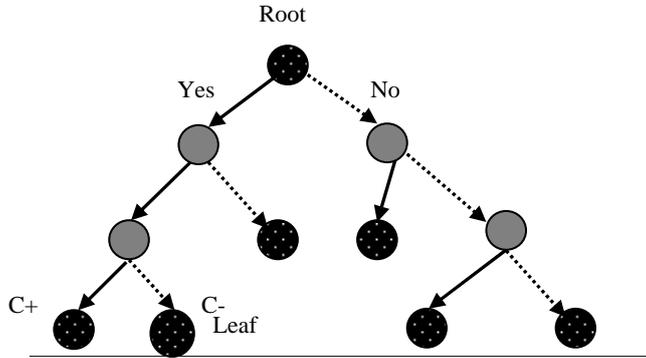


Figure 2 A multi-level binary tree classifier

For each node a decision rule is defined. In this study we use LDF (see Eq. (2.8b)) that can be learned from these samples corresponding to the node. The tree generation algorithm is briefly outlined as follows. First, the decision rule is applied to split the samples at the root node into two child nodes, the *Yes* node (a node pointed by a solid arrow line, the left branch of a node in Figure 2) and the *No* node (pointed by a dashed arrow line, the right branch of a node in Figure 2). For any document with the feature, X , if $f(W, X) > 0$, it is assigned into the *Yes* node. Otherwise, it is assigned into the *No* node. The similar decision rule is also applied at each internal node. If a decision would be made at the internal node, then the *Yes* node corresponds to the positive class ($C+$) while the *No* node refers to the negative ($C-$). The optimal parameters W are learned using the MFoM algorithm discussed above. This procedure runs recursively until there are no more nodes left to be split. At this point a binary tree classifier is finally established. In testing for a given document with feature X , the learned tree classifier assigns the document as either the positive class or the negative. The process starts from the root. If $f(W, X) > 0$, it enters the *Yes* node (left branch). Otherwise, it enters the *No* node (right branch). It continues until a leaf is reached, and the label at the leaf is then assigned to the document.

Many algorithms are proposed to find the weights of LDF (Breiman, et al. 1984, Brodley, et al. 1995, Quinlan 1993, Utgoff, et al. 1991). In this paper, we use the conventional gradient descent algorithm based on the perceptron criterion function $L(T; W)$ defined below (Duda, et al. 2001) for training the linear classifier at each internal node to obtain the baseline tree classifier,

$$L(T; W) = \sum_{X \in T} -f(W, X) \quad (3.1)$$

By minimizing the perceptron criterion, the parameters can be updated iteratively using the batch perceptron algorithm (Duda, et al. 2001),

$$W_{t+1} = W_t + \varepsilon_t \cdot \sum_{X \in T_t} X \quad (3.2)$$

where T_t is the set of the incorrectly classified samples and ε_t is the learning constant at the t -th iteration.

3.2 Learning MFoM Binary Tree Classifiers

In Section 2 we have discussed how to approximate the performance measures. Using the definitions in Eqs.(2.4) – (2.8) for a class j , an objective function can be formulated. For example, the loss function is defined as $L(T; W) = -F(T; W)$ in Eq.(2.9) in the case of using F_1 measure as the performance metric, while it is set to $L(T; W) = F(T; W)$ in Eq.(2.10) in the case of employing the classification error as the metric. From the above definitions, the objective function is a continuous and differentiable function of the parameter vector W . As discussed in Section 2.2, we do not try to jointly train N binary tree classifiers for optimizing the overall performance metric in our experiments. Instead, the binary tree classifier for each category is independently built by splitting a node using the learned LDF for optimizing the chosen metric (e.g. F_1 or classification error). Therefore for each non-leaf node, n , a metric oriented objective function, $L^n(T; W)$, is designed and a classifier is trained using the training samples assigned to it. In this paper, we are interested in designing the optimal classifier for each node using the MFoM approach rather than training a global tree classifier. So without any confusion, we will ignore this node superscript and use the notations, $L(T; W)$, for this objective function of each non-leaf node in the following discussions.

Generally speaking, the objective function is a highly nonlinear function of the classifier parameters W . It is difficult, if not impossible, to find its closed form solution. To solve the optimization problem numerically, a GPD algorithm is employed to seek an optimal W^* that can minimize $L(T; W)$. Denote its gradient as $\nabla L(T; W)$, W^* can then be found by the following iterative method, which has been proven to converge to a local minimum (Katagiri, et al. 1998),

$$W_{t+1} = W_t - \kappa_t \nabla L(T; W)|_{W=W_t} \quad (3.3)$$

where $\kappa_t = \kappa_0(1 - t/mCycles)$, W_t is the parameter vector at the t -th iteration, κ_0 is the initial learning rate, $mCycles$ is the pre-defined maximum iteration cycles, and κ_t is the updated step size or learning rate, which is a linearly decreasing function of the iteration index in our experiments. The GPD algorithm assures that, with more iterations, the value of the objective function, $L(T; W)$, decreases in a probabilistic sense until it reaches a local optimum. Therefore the F_1 measure increases and the classification error decreases with the iteration index. We will demonstrate this property empirically in the next Section. It is interesting to note the

similarity between Eqs. (3.3) and (3.2). The components of the gradient vector, $\partial L(X;W)/\partial W_i$, of the overall objective function in Eq. (3.3) can be computed using the chain rule.

In the binary classification for TC, we are more concerned with the performance of the positive classes (C_+). For the F_1 measure, we take into account of the effects of both positive and negative training samples; and the overall loss for a class is defined as:

$$L(T;W) = -\frac{2TP_+}{FP_+ + FN_+ + 2TP_+} = -\frac{2TP_+}{FP_+ + TP_+ + L_+} \quad (3.4)$$

where the constant L_+ is the size of the positive training set. The gradient of its objective function in Eq. (3.4) can be derived as

$$\nabla L(T,W)|_{w=W_i} = \frac{2}{(FP_+ + TP_+ + L_+)} \left[\frac{(FP_+ + L_+)}{(FP_+ + TP_+ + L_+)} \frac{\partial FN_+}{\partial W} \Big|_{w=W_i} + \frac{TP_+}{(FP_+ + TP_+ + L_+)} \frac{\partial FP_+}{\partial W} \Big|_{w=W_i} \right] \quad (3.5a)$$

with the two gradient terms on the positive and negative samples evaluated as follows,

$$\frac{\partial FN_+}{\partial W} \Big|_{w=W_i} = \sum_{X \in T} \alpha l_+ (1 - l_+) \cdot (-X) \cdot 1(X \in C_+) \quad (3.5b)$$

$$\frac{\partial FP_+}{\partial W} \Big|_{w=W_i} = \sum_{X \in T} \alpha l_- (1 - l_-) \cdot (X) \cdot 1(X \in C_-) \quad (3.5c)$$

with l_+ and l_- being the values of the positive and negative loss functions at the current value of W_i (in Eq. (2.7)), and X is the augmented vector in Eq. (2.8b). The term $l_+(1-l_+)$ or $l_-(1-l_-)$ indicates the significance of a sample in the learning update, which has a maximal value at the point $\alpha \cdot d_j(\cdot) + \beta = 0$. It is clear that only the training samples located within the learning window have much effect (See Eq. (2.7) and Figure 1). The final updated value in Eq. (3.5a) is a linear combination of the contributions from both the positive and negative samples.

For measures concerning classification accuracy, such as recall, only the positive samples contribute information to the evaluation metric. In the following we define the classification error as an average error over all positive and negative training samples instead:

$$L(T;W) = \frac{1}{L_+} \sum_{X \in T} l_+(X;W) 1(X \in C_+) + \frac{1}{L_-} \sum_{X \in T} l_-(X;W) 1(X \in C_-), \quad (3.6)$$

where L_- is the size of the negative training sets. The gradient of the objective function in Eq. (3.6) can similarly be derived as

$$\nabla L(T;W)|_{w=W_i} = \frac{1}{L_+} \sum_{X \in T} \alpha l_+ (1 - l_+) \cdot (-X) \cdot 1(X \in C_+) + \frac{1}{L_-} \sum_{X \in T} \alpha l_- (1 - l_-) \cdot (X) \cdot 1(X \in C_-). \quad (3.7)$$

3.3 Parameter Initialization

The classifier parameters should be initialized properly for any iterative algorithms in order to reduce the training time and speed up convergence. In Section 3.2 above, we have introduced the GPD-based MFoM learning algorithm. Now we discuss two ways for parameter initialization. One is to choose a value randomly or base on some expert knowledge. Another is to set the initial parameters equal to the model parameters estimated by the traditional learning methods (e.g. maximum likelihood based estimation of the classifiers, Juang, *et al.* 1997).

In this study the initial parameters of LDF, W , are empirically set equal to the mean of the training samples of the class whose number of training documents is less, while the offset term w_0 is initialized at the minimum value of the function $\sum w_k x_k$. In practice, this initialization strategy works reasonably well for both batch perceptron algorithm (BPA) and GPD-based MFoM learning. For GPD initialization it is also fine to use the parameter set of the baseline classifiers obtained with BPA.

3.4 Latent Semantic Indexing (LSI) for Feature Extraction

We have not discussed the document representation till now. In most information retrieval tasks, a document is often represented by a feature vector with the dimension equal to the size of the lexicon. Each component of the feature vector corresponds to the contribution of a term occurred in the document. In a typical application the lexicon usually has more than ten thousand entries. Many techniques, such as feature selection (Sebastiani 2002), have been proposed to reduce the dimension. We have adopted the latent semantic indexing (Deerwester 1990, Bellegarda 2000) for document representation in the paper, which is an efficient way to achieve both feature extraction and reduction. Hofmann (1999) proposed the probabilistic LSI. One good property of LSI or PLSI is that the word clustering has some latent semantic meaning. In this study, singular value decomposition (SVD) based LSI is used to get a lower dimension than the original one by decomposing the term-document matrix H into a multiplication of three matrices:

$$H = USV^T \quad (3.8)$$

where: $U : M \times R$ left singular matrix with rows $u_i, 1 \leq i \leq M$

$S : R \times R$ diagonal matrix of singular values $s_1 \geq s_2 \geq \dots \geq s_R > 0$,

$V : N \times R$ right singular matrix with rows $v_j, 1 \leq j \leq N$.

M : the size of the lexicon.

N : the number of the documents in the corpus.

Both left and right singular matrices are column-orthogonal. If we retain only the top Q singular values in matrix S and zero out other $(R-Q)$ components, the LSI feature dimension could be effectively reduced to Q that could be much smaller than R . By doing so, the three matrices are much smaller in size than those in Eq. (3.8) and it greatly reduces the computation requirements. We will explore some possibilities in Section 4.

The (i,j) -th element, $H(i,j)$, of matrix H describes the association between the i -th term and the j -th document, which is defined as

$$H(i,j) = (1 - \varepsilon_i) \cdot c_{i,j} / n_j \quad (3.9)$$

where $c_{i,j}$ is the number of times of the i -th term occurred in the j -th document, n_j the total number of words which appear in the j -th document, and ε_i the normalized entropy of the i -th term in the corpus which is further defined as

$$\varepsilon_i = -\frac{1}{\log N} \sum_{j=1}^N \frac{c_{i,j}}{t_i} \log \frac{c_{i,j}}{t_i} \quad (3.10)$$

where $t_i = \sum_j c_{i,j}$ denotes the occurrence times of the i -th term in the corpus.

With the above definitions, any document represented by a vector \tilde{d}^T in the original M dimensional space, can be described with an R dimensional vector, \tilde{v} .

$$\tilde{v} = \tilde{d}^T U S^{-1} \quad (3.11)$$

4. RESULTS AND ANALYSIS

4.1 Experimental Setup

In the following, we report our experiments on the ModApte version of the Reuters-21578 TC task⁵. Many evaluations have been documented based on this corpus (Denoyer, *et al.* 2001, Joachims 1998, Lewis, *et al.* 1994, Liu, *et al.* 2001, Wu, *et al.* 2002, Yang, *et al.* 1999). The original format of the text documents in this corpus is in SGML. We first remove all unlabeled documents, and perform some preprocessing on the remaining documents to filter out the unused parts of the documents by preserving only the title, the alphabetic part of dates excluding the digits, and the body text. We then retain only the categories that have at least one document in both the training and testing sets. This process gives rise to a collection of 90 categories. Altogether, 7,770 training and 3,019 testing documents are left. A set of 319 stop words and terms that occur less than 4 times are then removed. This gives a lexicon with 10,118 unique entries. The distribution of documents over the 90 categories is very unbalanced (Yang, *et al.* 1999). For example, the most frequent category ‘earn’ has 2,877 training documents. On the other hand some categories, such as ‘sun-meal’, ‘castor-oil’, and ‘lin-oil’, have only one training document. This inconsistency has to be considered in evaluating the robustness of the classifiers. Figure 3 shows the document frequencies of the 90 categories in the training (Fig. 3a) and testing sets (Fig. 3b), respectively.

To evaluate the classification performance for each category, precision, recall, and F_1 measure in Eqs. (2.1) - (2.3) are used. To measure the average performance for the 90 categories, the macro-averaging F_1 and the micro-averaging F_1 are used (Sebastiani 2002, Yang, *et al.* 1999). They are respectively defined as follows:

$$F_1^M = \frac{2 \sum_{i=1}^N R_i \sum_{i=1}^N P_i}{N (\sum_{i=1}^N P_i + \sum_{i=1}^N R_i)} \quad (4.1)$$

$$F_1^\mu = \frac{2 \sum_{i=1}^N TP_i}{\sum_{i=1}^N FP_i + \sum_{i=1}^N FN_i + 2 \sum_{i=1}^N TP_i} \quad (4.2)$$

The micro-averaging method calculates the global measures by giving the category’s local performance measures different weights based on their numbers of positive documents. Macro-averaging method treats every category equally, and calculates the global measures as the mean of all categories’ local measures. It is clear that the overall objective function defined in Eq. (2.9) resembles the micro-averaging F_1 defined in Eq. (4.2). It will be shown later that the F_1 -based MFoM approach formulated accordingly also works better when using micro-averaging F_1 for the comparison.

⁵ <http://www.daviddlewis.com/resources/testcollections/reuters21578/>

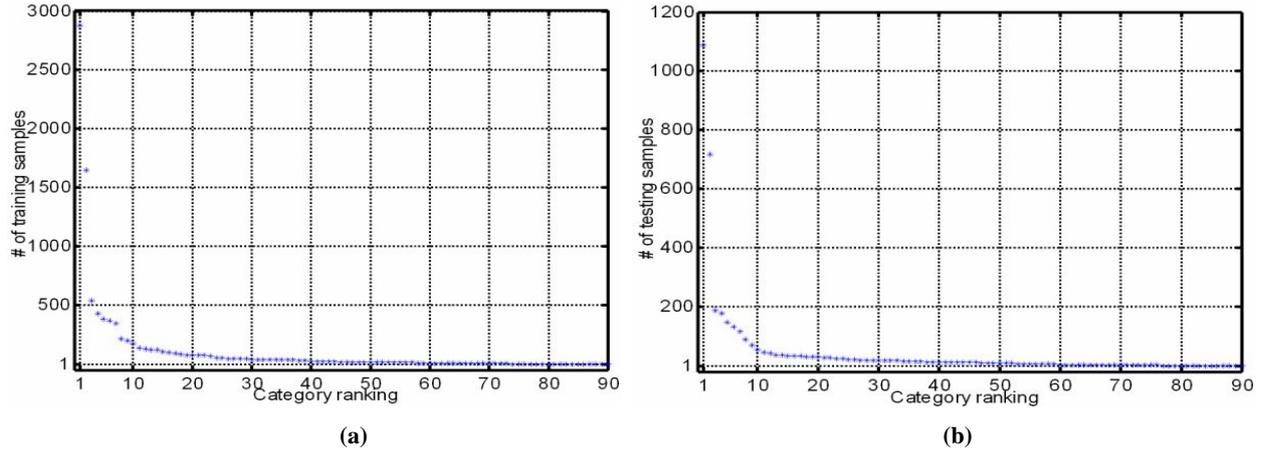


Figure 3 Document frequencies of 90 categories in the training (Left) and testing sets (Right)

4.2 Setting of Parameters for MFoM Learning

A 10,118 by 7,770 term-document matrix is first built from the training set using the LSI feature extraction method as discussed in Section 3. After performing singular value decomposition (SVD⁶, Berry, *et al.* 1993), the full rank of the matrix is found to be 1,613, the independent dimension of the feature vector in the latent semantic space. As compared to the original dimension of 10,118, there is already a substantial feature reduction. Further feature reduction may be carried out by retaining only the top few feature components.

In this study, the F_1 -based MFoM learning algorithm as defined in Eq. (3.4) is applied to train the linear classifiers. We carried out a series of initial studies to select the parameters for MFoM learning. The initial learning rate κ_0 in Eq. (3.3) is set by observing the changes in values of the objective functions based on the first k (k is set to 50) iterations on a few chosen categories. A good setting for κ_0 is one such that the value of the objective function changes more drastically for the initial few iterations and stabilizes with monotonically increasing or decreasing value after that. Based on our initial studies, we set κ_0 to 0.05 empirically.

The value for α for the Sigmoid function in Eq.(2.7) will influence the convergence rate as well as the smoothing of MFoM learning. As can be deduced from Eq.(2.7), a larger α value implies less smoothing in learning, while a smaller α value will slow down the rate of convergence. We carried out an initial set of experiments with k iterations to set a suitable value for α . The value is adjusted by comparing the value of the objective function (the approximated metric) with the correct metric for the training set. If they are very close, it means that the value for α is too big. If their difference is too large and the convergence is slow, it means that the value for α is too small. In this paper, we empirically set the value for α to 20.

The value of β will affect the decision boundary. It is also observed that adjusting its value may influence the precision and recall. In this paper, we dynamically adjust the value of β and set it to the average of the class misclassification function values of all training samples using Eq.(2.13). However, it was found that a constant value for β also works well.

⁶ <http://www.netlib.org/svdpack/>

Table 1 Micro-averaging F_1 (for all 90 categories) as a function of LSI feature dimensions and GPD iterations

Dimension /Iteration	500ite	1000ite	2000ite	3000ite
100	0.833	0.826	0.827	0.825
200	0.854	0.856	0.854	0.854
400	0.870	0.872	0.870	0.870
800	0.878	0.879	0.882	0.880
1200	0.880	0.882	0.882	0.883
1613	0.881	0.882	0.884	0.883

Table 2 Macro-averaging F_1 (for all 90 categories) as a function of LSI feature dimension and GPD iterations

Dimension /Iteration	500ite	1000ite	2000ite	3000ite
100	0.472	0.464	0.464	0.461
200	0.504	0.506	0.504	0.505
400	0.532	0.532	0.530	0.528
800	0.541	0.540	0.540	0.538
1200	0.556	0.566	0.565	0.565
1613	0.556	0.555	0.556	0.554

4.3 Properties of MFoM Learning

4.3.1 Performance as a Function of LSI Feature Dimension

First we will study the effects of varying the dimensions of LSI feature space and the iteration cycles in the GPD algorithm on the performance of F_1 -based MFoM learning. We vary the dimension from 100 to its full rank of 1,613, and the maximum iteration number of the GPD algorithm from 500 to 3,000. The overall performance measures for all 90 categories in these experiments are shown in Tables 1 and 2 for the micro-averaging and macro-averaging F_1 , respectively.

In both Tables, the first column is the selected dimension of the LSI feature space and the first row gives the maximum iteration number for the GPD training. From the Tables, we can see that both the micro-averaging and macro-averaging F_1 values increase in most cases as the dimension varies from 100 until its full rank of 1,613. For example, in the case of 500 GPD iterations, the micro-averaging F_1 value was improved from 0.833 to 0.881 and the macro-averaging F_1 value reaches 0.556 from 0.472. It is also observed that when the dimension is beyond 800, only a slight improvement in the respective F_1 measure is observed. The values shown in bold in each column in Table 1 indicate the best micro-averaging F_1 achievable for a given dimension with a fixed number of iterations. Since GPD only attains a local optimum in a probabilistic sense, we can only set the maximum iteration number empirically. In principle, we want to keep this number small so as to speed up training. However, it should not be too small in order to ensure convergence. Since the full rank dimensional feature with 2000 iterations gives the best overall performance, we will use these settings for the rest of experiments.

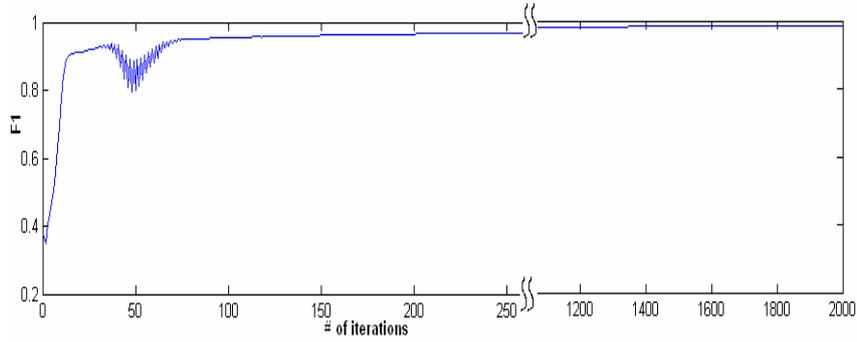


Figure 4: GPD convergence for category ‘Acq’ (feature dimension: 1613)

4.3.2 Convergence Behavior of the GPD Algorithm

The convergence property of GPD algorithm is illustrated in Figure 4 based on the category ‘Acq’ using the “equal weight” parameter initialization strategy as discussed in Section 3.3. From Figure 4, we observe some fluctuations in the F_1 values at the beginning of the algorithm (with about less than 80 iterations). This is because the GPD learning step size was set too high initially. The step size is then reduced linearly after the initial iterations, and the F_1 value increases smoothly from 0.372 until it reaches a stable value of 0.985. It is interesting to note that this convergence value for the training set is similar to the F_1 value of 0.968 for the testing set (as shown in Table 5), indicating a close match of the actual performance. This is an important feature for the MFoM learning to predict a desired performance measure in training without the need of running testing experiments, or collecting a large amount of testing data that can be very expensive in some situations.

4.3.3 Margin Analysis of Decision Boundaries

The decision margin measures the degree of separation between two classes. Now we will analyze how the MFoM algorithm influences the decision boundary between the positive and the negative classes. This analysis will reveal a nice property for MFoM that it is a robust discriminative learning algorithm. To illustrate this property, we plot the distributions of the class misclassification value defined in Eq. (2.8) for the positive and negative classes, respectively.

Figures 5-8 illustrate how MFoM maximally separates the decision margin between the positive and negative classes. Here the categories “Acq” with 1,650 positive samples and “Oat” with only 8 positive samples are chosen as the examples. Figures 5-7 are for “Acq” and Figure 8 for “Oat”. Due to insufficient number of positive samples to plot the curves for the positive class for “Oat” category, only the two curves for the negative class are illustrated in Figure 8. Figure 5 shows the distributions for the positive and negative classes at the beginning of the F_1 -based MFoM learning, while Figure 6 gives their distributions after 2000-iteration of MFoM learning. Figure 7 combines Figure 5 and Figure 6 to clearly show the changes of the distribution curves and the effects of MFoM learning on the decision margin. The arrows in Figure 7 clearly indicate the meaning of each curve. Intuitively an improved separation between the positive and negative classes over the original classifier is observed after the MFoM training.

We have pointed out in Section 2 that the value of the class misclassification function is a good indicator whether a correct classification decision has been made and how far it is from the decision boundary (a good measure of robustness), and the area of overlap is proportional to the classification error. At the beginning of MFoM learning, there is a large overlap between the distributions of C^+ and C^- samples (see Figure 5). This implies that there will be higher false positive and false negative rates, and a lower F_1 measure of the positive class as indicated in the beginning of GPD iterations in Figure 4. After 2000-iteration of MFoM learning, the curve of the positive class moves left while that of the negative class moves right. This results in a significant reduction of their overlap areas (see Figure 6) and an obvious reduction of classification errors because of the lower false positives and false negatives. It explains the increase of F_1 measure of the positive class as indicated at the end of GPD-iteration in Figure 4. Figure 6 also shows that the curves of the distribution become ‘flat’ after MFoM training, which implies that the resulting classifier is able to accommodate samples with more variance. The similar observation is also seen in Figure 8 for the topic “Oat” with a few positive samples. This is a clear indication that the MFoM-trained classifiers are more robust and less sensitive to data variation.

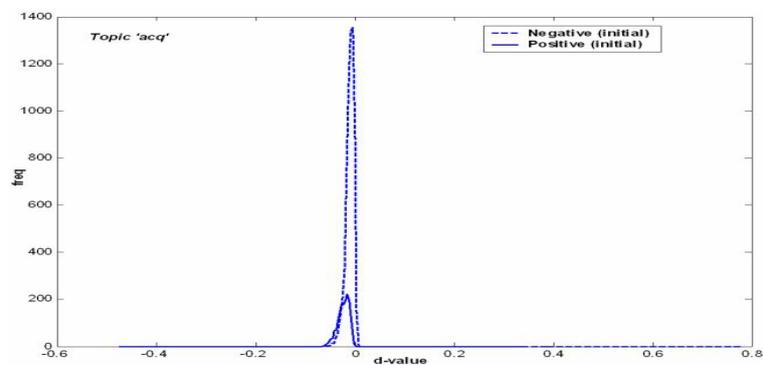


Figure 5: The d-value distributions of the positive and negative classes at the beginning of F_1 -based MFoM training for category ‘Acq’

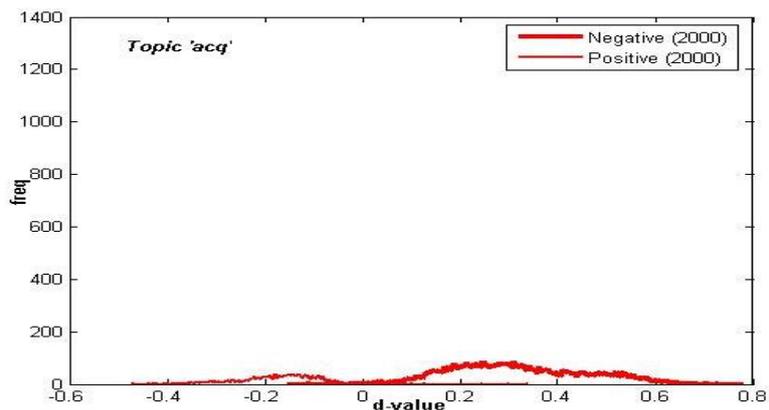


Figure 6: The d-value distributions of the positive and negative classes after 2000-iteration of F_1 -based MFoM training for category ‘Acq’

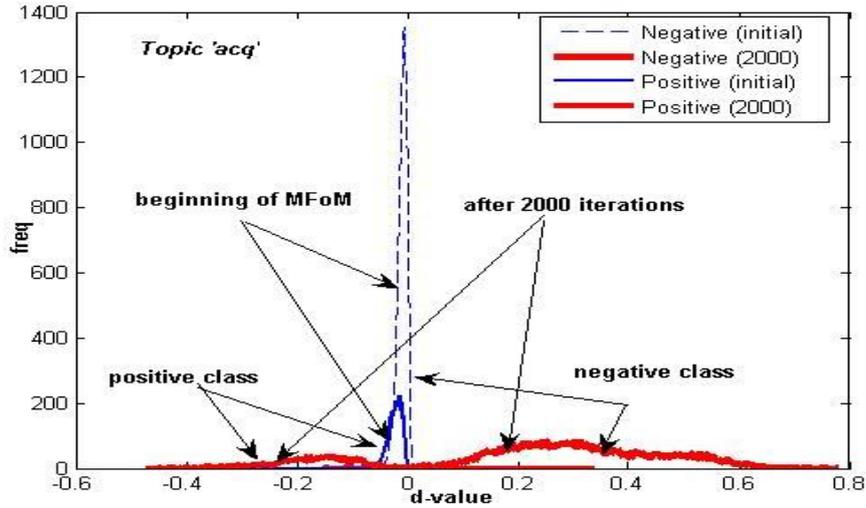


Figure 7: The d-value distributions before and after F_1 -based MFoM training for category ‘acq’ (merging Figure 5 and 6)

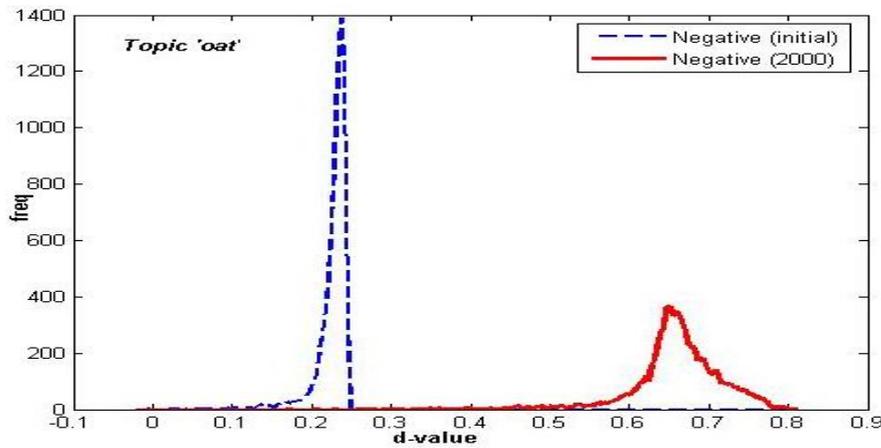


Figure 8: The d-value distributions before and after F_1 -based MFoM training for category ‘oat’ with small positive samples (Note: We plot only the curves for the negative class due to insufficient number of samples to plot curves for the positive class)

4.4 Performance and Robustness Analysis

From our analysis in Section 4.3.3, we have shown that MFoM can maximally separate the decision margin between the competing classes. This property makes MFoM more robust and less sensitive to data variation than the conventional learning approaches. We expect the MFoM-learned classifiers to give a better performance, especially for learning classifiers for the categories with only a few positive training samples. In the following sub-Sections, we address this important issue and compare the performance and robustness of MFoM with the SVM approach.

4.4.1 Robust Text Categorization with MFoM Learning

Learning classifier parameters with a set of sparse training data is always a challenging problem in many real-world applications. Most of the conventional learning methods (e.g. SVM, k -NN) can work well when a sufficient amount of training samples is available.

But they usually fail in the sparse training cases. Here we will show that MFoM works much better than the most popular SVM algorithms, especially when there is an insufficient amount of training data to learn the complex classifiers.

Table 3 Comparison of performance of classifiers on the various category collections with different training sample sizes

Category split (threshold of training sizes)	Micro-averaging F_1				Macro- averaging F_1			
	LSI-SVM- DEF	LSI-SVM- OPT	F-MFoM	s-test	LSI-SVM- DEF	LSI-SVM- OPT	F-MFoM	S-test
All 90	0.866	0.883	0.884	~	0.433	0.492	0.556	>>
Top 10 (≥ 150)	0.926	0.937	0.933	~	0.861	0.883	0.883	~
Other 80 (< 150)	0.645	0.691	0.720	~	0.377	0.439	0.512	>>
Top 16 (≥ 100)	0.916	0.928	0.925	~	0.841	0.862	0.869	~
Other 74 (< 100)	0.589	0.646	0.679	~	0.340	0.406	0.483	>>
Top 39 (≥ 30)	0.890	0.904	0.902	~	0.757	0.799	0.800	~
Other 51 (< 30)	0.340	0.429	0.537	>>	0.181	0.251	0.361	>>
Top 60 (≥ 10)	0.875	0.891	0.891	~	0.650	0.700	0.734	>
Other 30 (< 10)	0.000	0.108	0.310	>	0.000	0.075	0.198	>

Note: “>>”: means MFoM is much better than linear SVM with P-value < 0.01 . “>”: means MFoM is better than linear SVM with $0.01 \leq P\text{-value} \leq 0.05$. “~”: means MFoM performs equally well with linear SVM with P-value > 0.05 .

Table 4 F_1 values of MFoM and SVM as a function of the size of the positive samples for 4 selected categories (Note: the total available training samples for the categories are shown in parentheses)

Positive training sample sizes	Acq (1,650)			Wheat (212)			Grain (433)			Money-fx (550)		
	F- MFoM	LSI- SVM- OPT	LSI- SVM- DEF	F- MFoM	LSI- SVM- OPT	LSI- SVM- DEF	F- MFoM	LSI- SVM- OPT	LSI- SVM- DEF	F- MFoM	LSI- SVM- OPT	LSI- SVM- DEF
10	0.008	0.000	0.00	0.368	0.132	0.000	0.326	0.013	0.000	0.151	0.000	0.000
50	0.347	0.128	0.006	0.781	0.765	0.566	0.716	0.732	0.138	0.541	0.393	0.074
100	0.663	0.565	0.218	0.824	0.797	0.660	0.824	0.827	0.608	0.693	0.641	0.434
200	0.842	0.840	0.732	-	-	-	0.886	0.876	0.794	0.760	0.710	0.671
400	0.914	0.915	0.890	-	-	-	-	-	-	0.821	0.825	0.785
800	0.953	0.959	0.938	-	-	-	-	-	-	-	-	-
All	0.968	0.971	0.953	0.870	0.861	0.797	0.906	0.919	0.885	0.826	0.838	0.830

The first set of experiments analyzes the effects on the performance of classifiers on different sub-collections of categories with varying number of positive training samples. Starting with all 90 categories in one collection, we successively split the categories into two collections according to the size of positive training sample sizes of less than 150, 100, 30 and 10, respectively. Table 3 summarizes the micro- and macro-averaging results of these cases, together with the micro sign test (s-test) and macro sign test (S-test) (Yang & Liu, 1999) for MFoM and linear SVM classifiers. For example, in the last row of Table 3, we have two collections, one containing categories with greater than or equal to 10 training samples, and the other for categories with less than 10 training samples. Here we only compare the linear SVM (labeled as LSI-SVM) with the F_1 -based MFoM classifier (labeled as F-MFoM) using the full

rank (i.e. 1613) LSI features. The linear SVM was trained based on the SVM-light ⁷ package. The optimal configuration of parameter C (trade-off between training error and margin) for SVM-light is found empirically by seeking an optimal C (= 1.0) based on some randomly selected categories (referred to as LSI-SVM-OPT). For comparison, we also include the performance of SVM-light with the default configuration (LSI-SVM-DEF) in Table 3. It is seen that its performance of LFI-SVM-DEF is worse than the optimal configuration in all cases. Thus we evaluate the significant test by comparing only F-MFoM to LSI-SVM-OPT.

Here we discuss the robustness of MFoM and SVM learning with these different selections of positive training samples. It is clear that the overall performance measures are the function of training sample size. Although only a slight improvement from LSI-SVM-OPT to F_1 -based MFoM was observed while evaluating on categories with large training sample sizes, a significant gain was found for those categories with a small number of training samples. For example, the MFoM-learned classifier gave a micro-averaging F_1 value of 0.310, for all the categories with less than 10 training samples, while the LSI-based linear SVM classifiers could only achieve 0.108. The above performance difference is more pronounced when comparing the macro-averaging F_1 values. For instance, we see that such F_1 value improved from 0.251 for LSI-SVM-OPT to 0.361 for F-MFoM classifiers for the collection of 51 categories with less than 30 training samples. In contrast, the improvement in macro-averaging F_1 value was only from 0.799 to 0.800 for the collection of 39 categories with more than 30 training samples. In terms of macro-averaging F_1 measure for the overall 90 categories, MFoM significantly outperforms the SVM. For those categories with a small number of training samples (e.g. < 30), MFoM clearly outperforms SVM in terms of the micro-averaging and macro-averaging F_1 .

To further investigate the relationship between the classifier performance and the training set size, we perform a second set of experiments to study the variation of F_1 performance for MFoM and SVM based classifiers on a fixed category with varying number of positive training samples. Here four categories, “Acq”, “Wheat”, “Grain”, and “Money-fx”, are chosen. For each category, the size of the training set is increased from 10 to its maximum by randomly picking a subset of positive samples to train the classifiers. In all experiments, all available negative training samples are used. Table 4 compares the performance difference between the F_1 -based MFoM and the LSI-based linear SVM classifiers (optimal configuration, LSI-SVM-OPT, and default configuration, LSI-SVM-DEF) for each category. It can be seen that the F_1 value is improved with increasing size of positive training samples in all cases for the two learning algorithms. Again SVM with the optimal configuration performs consistently better than that with the default. The advantage of MFoM is clearly demonstrated when only a small number of positive samples (e.g. less than 100) are available. In cases with very sparse training set (e.g. less than 10), MFoM still works while the SVM classifier fails to produce any correct results for two categories of “Acq” and “Monet-fx”. For the other categories, for example, the MFoM achieves an F_1 value of 0.368 for the “Wheat” category as compared to a value of 0.132 achievable by SVM (optimal configuration) when only 10 positive samples were used for training the classifiers.

⁷ <http://svmlight.joachims.org/>

Table 5 Performance comparison in F_1 among the baseline, LSI-SVM-DEF, LSI-SVM-OPT, J-SVM and F-MFoM

Category	Binary Tree		SVM		
	Baseline	F-MFoM	LSI-SVM-DEF	LSI-SVM-OPT	J-SVM
Earn	0.979	0.979	0.982	0.984	0.982
Acq	0.953	0.968	0.953	0.971	0.956
Money-fx	0.784	0.826	0.830	0.838	0.785
Grain	0.889	0.906	0.885	0.919	0.931
Crude	0.887	0.897	0.899	0.898	0.894
Trade	0.730	0.807	0.802	0.802	0.792
Interest	0.743	0.792	0.771	0.788	0.748
Ship	0.853	0.878	0.850	0.886	0.865
Wheat	0.829	0.870	0.797	0.861	0.868
Corn	0.821	0.891	0.800	0.863	0.878
Micro-avg (all 90)	0.854	0.884	0.866	0.883	0.875
Macro-avg (all 90)	0.519	0.556	0.433	0.492	NA*

*The result for J-SVM on macro-averaging F_1 is not available.

Table 6 Overall performance measure vs. the optimized metrics

		Recall	Precision	F_1
Micro-avg (all 90)	MaxRe	0.876	0.848	0.862
	MaxPr	0.641	0.973	0.773
	MaxF1	0.857	0.914	0.884
Macro-avg (all 90)	MaxRe	0.613	0.619	0.616
	MaxPr	0.133	0.367	0.195
	MaxF1	0.483	0.655	0.556

Table 7 Significant test at the micro level for three types of metric oriented classifiers

	Recall	Precision	F_1
Significant test	MaxRe >> MaxF1 >> MaxPr	MaxPr >> MaxF1 >> MaxRe	MaxF1 >> MaxRe >> MaxPr

Note: “>>”: P-value <0.01.

4.4.2 Overall Performance Comparison

For TC, the top performance in most data sets was often achieved with the SVM-based classifiers. In Table 5, we compare the performance of our binary tree classifiers with F_1 -based MFoM learning and without (labeled as “Baseline”). We also report the new results using the linear SVM based on the LSI features (SVM-light with optimal, LSI-SVM-OPT, and default configurations, LSI-SVM-DEF, respectively). Here, the experimental setting was the same as that described in Section 4.3.1. As a reference, we also include the state-of-the-art results obtained with the linear SVM from Joachims, 2002 (labeled “J-SVM”, with $C=1.0$, full 9,600 features, no feature selection).

With F_1 -based MFoM learning we obtained the micro-averaging and macro-averaging F_1 values of 0.884 and 0.556, respectively. When compared to the baseline and LSI-SVM-OPT, we observe a significant improvement in the results in terms of macro-averaging

F_1 (significance test is shown in the row labeled with “All 90” in Table 3). When compared with J-SVM, only a slight improvement in the results is observed. However, a closer look at the top 10 categories shows that the F_1 -based MFoM classifier produces better F_1 values than those obtained with the J-SVM classifiers on most categories (except for two categories “Earn” and “Grain”).

It is worth noting that in the past we only reported results for collections with top 10 categories and with all 90 categories, like the ones listed in Table 5. We need to also pay attention to robustness issues and performance variation with the training set sizes, especially for categories with sparse training data, and report the results in the style similar to Tables 3 and 4, or other more meaningful indicators.

4.5 Flexibility in Designing Metric-oriented Classifier

As discussed in Section 2, one advantage of MFoM is its ability to integrate different performance metric of interest into an overall objection function and to learn the parameters of the classifier by optimizing this function. This makes it a feasible approach to design an optimal classifier in terms of the preferred metric. For example, following the principle of MFoM learning for objective function design, it is simple to formulate the classifier learning algorithm for optimizing the precision, recall, or even the AUC metric discussed in (Fawcett, 2003; Flach, 2003; Yan, et al, 2003). The performance of MFoM classifier for optimizing the F_1 measure has been investigated in the previous Sections, here we will compare its performance with other MFoM classifiers based on recall and precision measures for the overall 90 categories.

The classifier is still the binary decision tree discussed in Section 3.1. The only difference is that the parameters of LDF function at each non-leaf node are estimated by maximizing the precision or recall. From the definitions of precision, recall and F_1 in Eqs. (2.1-2.3), maximizing recall will cause the classifier to highly bias towards the positive class since no negative training sample is used in the GPD estimation. The classifier by maximizing the precision has less bias than that by maximizing the recall because there are some negative samples (i.e. false accepted) that are used. We use the experimental settings as that used in Section 4.3. The resulting MFoM classifiers are denoted by MaxPr, MaxRe and MaxF1 designed to maximizing the average precision, average recall and F_1 , respectively. The results for the overall 90 categories for the three classifiers are listed in Table 6. For each type of metric, the best classifier is highlighted. To compare the systems using the precision and recall, the p-test is preferred as discussed in Yang & Liu (1999), which is an evaluation at the micro level. Similarly, we also use the micro level evaluation when comparing the F_1 score for the three types of classifiers. The results for the significant test at the micro level are shown in Table 7.

From Tables 6 and 7, we could draw the following observations:

- [1] The MFoM approach is flexible to design the classifier by optimizing different preferred metric. By following the principle of MFoM learning, the objective function for different metric and different classifier can be easily formulated and solved using the GPD algorithm.
- [2] The classifiers learned by optimizing different metrics have distinctive properties. Each classifier is able to produce optimal result in the evaluation set for the chosen metric that it is designed for. For example, the classifier trained by maximizing the F_1 metric in the training set also has the best F_1 performance for the test set. Similarly, the classifier trained by maximizing the

recall (or precision) produces the best results in recall (or precision) metric. This desirable property is expected for many real-world applications, in which the user designates a performance metric.

- [3] Among the three classifiers, the MaxF1 classifier, besides being the best for F_1 measure, also performs well in terms of recall and precision. This is because the F_1 measure balances the positive samples and negative samples better than the other two.

5. DISCUSSION

5.1 Learning Non-linear Classifier Using MFoM

We have discussed the MFoM learning approach for designing the linear classifier, experimentally analyzed its good properties, and compared with the state-of-art results. However, the proposed approach is not limited to linear classifiers. It can also be used to optimize other kinds of classifiers with the preferred metric. It is easy to derive the corresponding training algorithm given a specific classifier, such as hidden Markov model, Gaussian mixture model, etc., and a metric, as long as the gradient with respect to the classifier parameters can be computed. For example, if we define the classifier as a weighted summary of some kernel functions (similar to the SVM based classifier) for the training samples, the same non-linear transformation can also be incorporated into MFoM. In this paper we only focus on the discussion of MFoM formulation and the GPD algorithm. Its extension to non-linear transformation using these kernel functions can easily be derived. Interested readers can design their specific classifiers using this methodology.

5.2 The Convexity of the Objective Function

The proposed approach is a generic method for designing the classifiers. It is independent on the selected classifier. Because of the non-linearity of the loss function discussed in the paper and the characteristics of the chosen classifier, the objective function is often non-linear and there is not an analytic solution for it. Although in some cases the objective function is convex and an analytic solution exists, it is out of the scope of this paper to derive the possible analytical solution. Here we only provide a way to design the metric-oriented objective function and a possible method to find its solution. If the readers know their objective function is convex, they can apply other algorithms for convex optimization to find its global optimum, not necessarily to use the gradient algorithm introduced in the paper.

5.3 Regularized Linear Classification and MFoM Learning

Robustness is one of the central issues when designing classifiers for the real-world classification problems. In many cases collecting enough training data is too costly, if not impossible. To alleviate the data sparseness problem, some penalty terms such as minimum description length (MDL) (Duda, *et al.* 2001), which measures the complexity of the model, is often appended to the objective algorithm. This family of techniques includes linear SVM and regularized logistic regression linear classifiers (Zhang & Yang, 2003). The objective function of the family of the regularized linear classification algorithms is defined as

$$L(W) = \frac{1}{N} \sum_{k=1}^N f(z_k W^T X_k) + \lambda \|W\| \quad (5.1)$$

where N is the size of the training set, and λ is a constant. The first term in the right hand side is a measure of the regression error in the training set, and the second is a penalty weighed by a constant. For ridge regression, regularized logistic regression and linear SVM, the embedding function, $f(z_k W^T X_k)$, is defined in the following accordingly:

$$f(z_k W^T X_k) = (1 - z_k W^T X_k)^2 \quad (5.2)$$

$$f(z_k W^T X_k) = \log(1 + \exp(-z_k W^T X_k)) \quad (5.3)$$

$$f(z_k W^T X_k) = \max\{0, 1 - z_k (W^T X_k + b)\} \quad (5.4)$$

Linear SVM can be considered as a special case of the general regularized linear classification. In both approaches, the objective functions are closely related to the empirical risk in the training set. Their differences are in the definition of their loss functions. To get a bounded solution of the objective function in Eq. (5.1), a penalty defined by the 2-norm regularization term is appended. Finding the decision boundary is formulated as a constrained linear optimization problem. The classifier can be trained by minimizing the objective function, which also implies minimizing an approximation of the empirical classification errors.

In contrast, MFoM formulates the learning of classifiers using a consistent objective function for training and testing, and embeds the classification decision rules into the performance measure of interest. Optimizing this function directly optimizes the performance metric and class separation simultaneously. The performance improvement can be attributed to a joint and explicit optimization of the classifier parameters and their performance, making MFoM different from other existing learning frameworks, e.g. SVM, ANN, and regularized linear classification methods.

6. CONCLUSION

In this paper we proposed a maximal figure-of-merit (MFoM) learning approach to classifier design and experimentally verified its effectiveness. This training strategy explicitly relates different classifiers of interest to different preferred performance metric by formulating a consistent overall objective function. To embed the discrete decision rules into the objective function, a smooth approximation of the discrete error counts was used. This decision-feedback learning framework is attractive because it offers a novel way to directly optimizing the desired performance figure-of-merit. The measures obtained in the training stage can also be used to predict the same level of performance of the chosen metric on a similar testing data set. This makes it easier for a designer to estimate the classifier performance without the need to run an extensive set of experiments or collect a large set of evaluation data, which can be very expensive. To further highlight the characteristics of MFoM, we showed that the MFoM learning approach maximizes the class separation of training data. All these indicate that the MFoM-learned classifiers are robust, and insensitive to the training data variation and unseen testing conditions.

We applied MFoM to text categorization and reported the experimental results using the ModApte version of Reuters-21578 corpus. We compared the MFoM-learned classifiers with the baseline binary classifiers, the linear SVM classifiers trained on the same

set of 1,613-dimension LSI features used in the binary tree classifiers, and the state-of-the-art linear SVM classifiers with all 9,600 features without feature selection. Our results showed that the F_1 -based MFoM is able to achieve high performance for the TC task, and that it outperformed most competing methods on a wide variety of categories and experimental settings.

To demonstrate the robustness property, we split the 90 categories into two collections, and compared the performance differences on the categories with different training sizes. A significant improvement was observed if we only consider categories with less positive training samples, while no pronounced enhancements were observed in cases with more training samples. It can be concluded that MFoM can tolerate the insufficient training data. We also observed the same robustness improvement by splitting the training samples into smaller sets on some categories with a large number of training samples. We suggested for this type of experimental analysis to be included in the future experimental studies for TC and IR applications.

Moreover, we experimentally studied the relation between the performance measure and the optimized metric used in the MFoM learned classifier. Our experiments demonstrated that the optimized metric on the training set for MFoM is also best on the evaluation set. This is a good property expected in many real-world applications. Since the classifiers learned from different metrics have different properties, the decision fusion by combining them is a good topic for future work. We will also explore the new multi-category classification paradigms that go beyond the currently prevailing binary classification approaches to TC. Our preliminary results already indicated that the MFoM-trained multiclass classifiers give better and more robust performance than the MFoM-trained binary tree classifiers. We will also apply MFoM methodology to other interesting classification and verification problems in natural language processing, text categorization, information retrieval and data mining.

7. ACKNOWLEDGEMENT

Part of this work was carried out while the first author was with National University of Singapore (NUS), supported under the Research Grant RP3989903 provided by the National Science and Technology Board and the Ministry of Education of Singapore. The authors would also like to thank Dr. Hwee-Tou Ng and Dr. Xiao-Li Li of NUS for providing invaluable suggestions and Dr. Qin-bin Sun for his kindly support. The authors would also thank anonymous reviews and the associate editor for insightful and helpful comments that help to improve the quality of the final manuscript.

8. REFERENCE

- Bellegarda, J. R. (2000) Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*. Vol.88, No.8, pp.1279-1296.
- Berry, M. et al. (1993) SVDPACKC (Version 1.0) User's Guide. University of Tennessee Technical Report CS-93-194.
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C.J. (1984) *Classification and Regression Trees*. Wadsworth Int.
- Brodley, C. E. and Utgoff, P. E. (1995) Multivariate decision trees. *Machine Learning*. Vol.19, No.1, pp.45-77.

- Buckley, C., Salton, G. and Allan, J. (1994) The effect of adding relevance information in a relevance feedback environment. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.292-300.
- Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*. Vol.20, pp.273-297.
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. (1990) Indexing by latent semantic analysis. *Journal of the American Society for Information Science*. Vol.41, No. 6, pp.391-407.
- Denoyer, L., Zaragoza, H. and Gallinari, P. (2001) HMM-based passage models for document classification and ranking. In: Proceedings of the 23rd European Colloquium on Information Retrieval Research, pp.126-135.
- Duda, R. O., Hart, P. E. and Stork, D. G. (2001) *Pattern classification*. Second Edition, A Wiley-Interscience Publication.
- Fawcett, T. (2003). ROC graphs: notes and practical considerations for researchers. *Technical Report HPL-2003-4*, HP Labs.
- Flach, P. A. (2003). The geometry of ROC space: understanding machine learning metrics through ROC isometrics. *ICML'03*.
- Gao, S., Wu, W., Lee, C.H., and Chua, T.S. (2003) A maximal figure-of-merit learning approach to text categorization. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp.174-181, Toronto, Canada.
- Gao, S, Wu, W., Lee, C.H., and Chua, T.S. (2004) An MFoM learning approach to robust multiclass multi-label text categorization. In: Proceedings of the 21st Annual International Conference on Machine Learning, Banff, Canada.
- Guo, H. and Gelfand S. B. (1992) Classification trees with neural network feature extraction. *IEEE Transactions on Neural Networks*, Vol. 3, No. 6, pp.923-933.
- Hofmann, T. (1999) Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.50-57.
- Joachims, T. (1997) A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In: Proceedings of the 14th International Conference on Machine Learning, pp.143-151.
- Joachims, T. (1998) Text categorization with Support Vector Machines: learning with many relevant features. In: Proceedings of the 10th European Conference on Machine Learning, pp.137-142.
- Joachims, T. (2002) *Learning to classify text using Support Vector Machines*. Kluwer Academic Publishers.
- Juang, B.-H., Chou, W. and Lee, C.-H. (1997) Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, Vol.5, No. 2, pp.257-265.
- Katagiri, S., Juang, B.-H. and Lee, C.-H. (1998) Pattern recognition using a family of design algorithm based upon the generalized probabilistic descent method. *Proceedings of the IEEE*, Vol.86, No.11, pp.2345-2373.
- Kuo, H. K. J. and Lee, C.-H. (2003) Discriminative training of natural language call routers. *IEEE Transactions on Speech and Audio Processing*, Vol.11, No.1, pp.24-35.
- Lee, C.-H. and Huo, Q. (2000) On adaptive decision rules and decision parameter adaptation for automatic speech recognition. *Proceedings of the IEEE*, Vol.88, No.8, pp.1241-1269.

- Lee, C.-H., Soong, F. K. and Paliwal, K. K. (1996) *Automatic Speech and Speaker Recognition: Advanced Topics*. Kluwer Academic Publishers, Boston.
- Lewis, D. and Ringuette, M. (1994) A comparison of two learning algorithms for text categorization. In: *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*, pp.81-93.
- Li, Y. H. and Jain, A. K. (1998) Classification of text documents. *The Computer Journal*. Vol.41, No.8, pp.537-546.
- Liu, J. M. and Chua, T. S. (2001) Building semantic perceptron net for topic spotting. In: *Proceedings of the 37th Meeting of Association of Computational Linguistics*, pp.370-377.
- Makoto, I. and Takenobu, T. (1995) Cluster-based text categorization: a comparison of category search strategies. In: *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.273-280.
- Masand, B., Lino, G. and Waltz, D. (1992) Classifying news stories using memory based reasoning. In: *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.59-65.
- McCallum, A. and Nigam, K. (1998) A comparison of event models for naïve Bayes text classification. In: *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, pp.41-48.
- Miller, D. R. H., Leek, T. and Schwartz, R. M. (1999) A Hidden Markov model information retrieval system. In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.214-221.
- Ng, H. T., Goh, W. B. and Low, K. L. (1997) Feature selection, perceptron learning, and a usability case study for text categorization. In: *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.67-73.
- Quinlan, J. (1993) *C4.5: Programming for machine learning*. Morgan Kaufmann.
- Ruiz, M. E. and Srinivasan, P. (1999) Hierarchical neural networks for text categorization. In: *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.81-82.
- Schapire, R. and Singer, Y. (2000) BoosTexter: A boosting-based system for text categorization. *Machine Learning*, Vol.39, No.2-3, pp.135-168.
- Schutze, H., Hull, D. A. and Pedersen, J. O. (1995) A comparison of classifier and document representations for the routing problem. In: *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.229-237.
- Sebastiani, F. (2002) Machine learning in automated text categorization. *ACM Computing Surveys*, Vol.34, No.1, pp.1-47.
- Utgoff, P. E. and Brodley, C. E (1991) Linear machine decision trees. COINS Technical Report 91-10, Dept. of Computer Science, University of Massachusetts.
- Vapnik, V. (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag.
- Wu, H. R., Phang, T. H., Liu, B., and Li, X. L. (2002) A refinement approach to handling model misfit in text categorization. In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.207-216.

- Yan, L., Dodier R., Mozer, M.C. & Wolniewicz, R. (2003) Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In: In: Proceedings of the 20th Annual International Conference on Machine Learning.
- Yang, Y.M. (1994) Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.13-22.
- Yang, Y.M. and Liu, X. (1999) A re-examination of text categorization methods. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.42-49.
- Zhang, J. and Yang, Y.M. (2003) Robustness of Regularized Linear Classification Methods in Text Categorization. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.190-197.