MCE Training Techniques for Topic Identification of Spoken Audio Documents

Timothy J. Hazen, *Senior Member, IEEE*

***Abstract*—In this paper, we discuss the use of minimum classification error (MCE) training as a means for improving traditional approaches to topic identification such as naive Bayes classifiers and support vector machines. A key element of our new MCE training techniques is their ability to efficiently apply jackknifing or leave-one-out training to yield improved models which generalize better to unseen data. Experiments were conducted using recorded human–human telephone conversations from the Fisher Corpus using feature vector representations from word-based automatic speech recognition lattices. Sizeable improvements in topic identification accuracy using the new MCE training techniques were observed.**

***Index Terms*—Discriminative training, machine learning, speech recognition, topic identification.**

I. Introduction

The problem of topic identification (topic ID) has long been studied in the text processing community where research has produced a variety of techniques and enabled applications such as e-mail spam filtering and inappropriate web material detection [16], [22], [26]. The speech processing community has largely based its topic ID research on techniques originally developed for text processing. In particular, both Bayesian classifiers [21], [25] and support vector machines (SVMs) [10], as applied to *bag-of-words* document representations have been successfully ported to speech-based applications. Minimum classification error (MCE) training has also been used in the call routing field to further improve the performance of linear classifiers [18], [20], [30]. Although these techniques have proven effective, they still possess known deficiencies which remain open research topics. In this paper, we introduce novel methods for applying MCE training to classifiers which improve upon these existing approaches.

One challenging aspect of the topic ID problem is that the common words most prevalent in a document (i.e., articles, conjunctions, auxiliary verbs, etc.) are also the least relevant for determining the underlying topic. Instead it is the sporadic use of a few topic-specific content words which typically define the topic. Thus, it is important for topic ID systems to reduce the effect of function words on classification decisions while enhancing the contribution of content bearing words. Various techniques have been developed to address this problem including stop listing, feature selection, and feature weighting (e.g., inverse document frequency weighting). Though these techniques have proven effective, they generally provide only an indirect means for determining which words are most important for topic ID. A better approach is to determine the relative importance of words based directly on their contribution to the correct classification of the topic. We have previously shown that MCE training can yield sizeable improvements in topic ID accuracy when applied to feature weights in both naive Bayes classifiers [13] and SVMs [14].

Another issue of concern arises from the traditional approach of training multi-class SVMs. For a classification problem with M classes, SVM training would typically optimize M different two-class *one-versus-rest* classifiers independently instead of globally optimizing a single M-class classifier. As a result, the scores of the different two-class SVMs may not be well calibrated for comparison within the M-class problem. Platt [23] has addressed this calibration issue by fitting sigmoid functions to the outputs of the SVM classifiers to generate scores approximating probabilities, while Haffner *et al.* [10] and Brummer and van Leeuwan [2] have used logistic regression to calibrate the output scores of various types of classifiers. We have previously presented a mechanism for calibrating multi-class SVMs using the MCE algorithm that operates using an efficient leave-one-out approximation which allows all training tokens to be treated as unseen data during the optimization process [15].

Although methods for training multi-class SVMs using soft-margin objective functions exist [7], some researchers have expressed concerns about the soft-margin optimization approach and its sensitivity to outlier training tokens. Arenas-García and Pérez-Cruz [1] have argued instead for optimization using empirical risk minimization. Similarly, Gao *et al.* [8] have introduced a maximum figure of merit approach for optimizing the detection performance of SVM-based topic detection systems, while Hatch *et al.* [11] have proposed a minimum equal error rate optimization process for training the SVM parameters in speaker recognition systems. In these cases, soft margin optimization was replaced by an objective function mirroring the system's evaluation criterion. In our previous work, we have similarly developed an MCE approach for training the parameters of a multi-class SVM classifier [15].

In this paper, we summarize the MCE training techniques we have developed and the experimental results we have obtained for topic identification of spoken audio documents. This

Manuscript received November 12, 2010; revised March 04, 2011; accepted March 21, 2011. Date of publication April 07, 2011; date of current version September 16, 2011. This work was supported by the Air Force Research Laboratory under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Haizhou Li.

The author is with the MIT Lincoln Laboratory, Lexington, MA 02420 USA (e-mail: hazen@ll.mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2011.2139207

paper extends our previous work by providing additional algorithmic details and empirical analyses not provided in our previous work. We demonstrate the effectiveness of our techniques on topic ID experiments conducted on the Fisher corpus of recorded human–human conversations.

II. Spoken Document Representations for Topic ID

A. Overview

For the topic ID problem, assume we have a collection of training data $D = \{d_1, \ldots, d_{N_D}\}$ containing N_D individual audio documents. Also assume that we have a set $T = \{t_1, \ldots, t_{N_T}\}$ of N_T different topics of interest. In the closed-set topic classification problem that we will address in this paper we assume that each document d is associated with one and only one topic t.

In the commonly used *bag-of-words* approach, each document is represented by the occurrence counts of the individual words present in the document, independent of their ordering. Thus, a document d can be represented as

$$d = \{c_1, c_2, \ldots, c_{N_V}\}. \quad (1)$$

Here, each c_w is the occurrence count of a specific word w, where w is a specific word from a vocabulary V containing N_V unique words. This bag-of-words representation has proven highly effective despite its simplicity [16], [19].

B. Extracting Features From ASR Lattices

When performing topic ID on spoken audio documents, the spoken words must be ascertained using automatic speech recognition (ASR). In many domains, highly accurate and robust ASR is not yet possible thereby requiring topic ID systems to be capable of handling errorful ASR outputs. In this situation, it has become standard practice to produce a network, or *lattice*, of alternate ASR hypotheses instead of the single top-choice (or 1-best) hypothesis. From an ASR lattice, a posterior probability of occurrence for each word hypothesis can be computed. The posterior probabilities for an individual word w can be summed across all lattice arcs containing that word from all lattices associated with an audio document to produce an estimated occurrence count for that word in the audio document. For consistency, we also represent the estimated count for word w as c_w when this count is derived from the ASR output lattices.

C. Feature Selection

Topic ID systems often preselect a set of features (i.e., words) which carry the most topic relevant information for further processing while ignoring words that carry little to no value for identifying topics. A variety of feature selection methods have been previously explored for both text processing [29] and speech processing [12]. In our work, we have had the most success selecting features based on the estimated topic posterior probability $P(t \mid w)$, where t is a specific topic and w is a specific word. The maximum *a posteriori* probability (MAP) estimate of $P(t \mid w)$ is expressed as

$$P(t \mid w) = \frac{N_{w \mid t} + N_T P(t)}{N_w + N_T}. \quad (2)$$

Here, $N_{w \mid t}$ is the number of times word w appears in documents about topic t, N_w is the total number of times w appears over all documents, N_T is the total number of topics, and $P(t)$ is the prior likelihood of topic t as estimated from the training corpus. Feature selection in this work is performed by selecting the top N words w which maximize $P(t \mid w)$ for each t.

D. Feature Vector Normalization

Each document d is typically represented by a feature vector $\vec{x}$ that represents the contents of d. Because different documents can have different lengths, it can be useful to apply L1 normalization to convert a collection of raw word counts into a normalized feature vector $\vec{x}$. The value for element x_w in an L1 normalized vector $\vec{x}$ for document d is given as

$$x_w = P(w \mid d) = \frac{c_w}{\sum_{\forall i \in V} c_i}. \quad (3)$$

Here, V can either represent the full vocabulary or a subset of selected features.

E. Feature Weighting

In practice, classifiers such as SVMs often do not directly use the vector of raw word counts or relative frequencies, but instead employ some form of feature weighting in order to de-emphasize the common function words (i.e., articles, conjunctions, prepositions, etc.) while enhancing the important content words. One weighting function for de-emphasizing common words that can be applied to an L1 normalized feature vector is expressed as

$$x_w = \frac{P(w \mid d)}{\sqrt{P(w)}} \quad (4)$$

Here, $P(w)$ is estimated from the full collection of training documents using MAP estimation as follows:

$$P(w) = \frac{N_w + 1}{N_W + N_V}. \quad (5)$$

In this expression, N_w is the number of occurrences of the specific word w in the training corpus and N_W is the total count of all words from the N_V word vocabulary in the training corpus. Vector normalization and weighting as expressed in (4) has been referred to as the *term frequency-log likelihood ratio* (TF-LLR) representation [3].

Another commonly used weighting is inverse document frequency (IDF), as represented as

$$\mathrm{idf}(w) = \log \frac{N_D}{N_{D \cap w}}. \quad (6)$$

Here, N_D is the total number of training documents while $N_{D \cap w}$ is the number of training documents containing the word w. Because the words in audio documents are generally not known, $N_{D \cap w}$ is also not known and must be estimated from ASR lattices [28]. In our work, $N_{D \cap w}$ is estimated from the training corpus as

$$N_{D \cap w} = \max\left(\kappa, \sum_{\forall d \in D} \min(c_w, 1)\right). \tag{7}$$

Here, c_w is an estimated count of word w in document d as computed from ASR posterior lattices. Thus, c_w can have a positive value less than one. The parameter κ provides a floor on $N_{D \cap w}$ thus setting an upper bound on $\mathrm{idf}(w)$. In our work, we set $\kappa = 0.01$. Using IDF weighting in conjunction with the raw counts c_w yields the traditional *term frequency-inverse document frequency* (TF-IDF) representation:

$$x_w = c_w \,\mathrm{idf}(w). \tag{8}$$

III. Traditional Classifiers

A. Naive Bayes Classification

In our naive Bayes approach to topic ID, a hypothesis testing likelihood ratio approach is used. A document, represented by a relative frequency feature vector $\vec{x}$, is scored against a specific topic t using this expression

$$S(\vec{x}, t) = -b_t + \sum_{\forall w \in V} x_w \log \frac{P(w \,|\, t)}{P(w \,|\, \bar{t})}. \tag{9}$$

Here, $P(w \,|\, t)$ is the probability of word w within documents on topic t while $P(w \,|\, \bar{t})$ is the probability of w in documents that are not on topic t. The features x_w in $\vec{x}$ are determined as in (3). The offset value b_t can be used to incorporate the prior distribution $P(t)$, though our experiments all assume that topics are equally likely and $b_t = 0$.

The probability function $P(w \,|\, t)$ is learned from labeled training data using MAP estimation as follows:

$$P(w \,|\, t) = \frac{N_{w\,|\,t} + N_V P(w)}{N_{W\,|\,t} + N_V}. \tag{10}$$

Here, N_V is the number of unique words in the vocabulary, $N_{w\,|\,t}$ is the count of w over documents on topic t, and $N_{W\,|\,t}$ is the total count of all words in documents on topic t. $P(w \,|\, \bar{t})$ is estimated in the same manner using documents not about topic t. The term $P(w)$ represents the prior likelihood of w occurring as represented by (5).

B. Support Vectors Machines

Traditional SVM training finds a hyperplane which maximally separates positive and negative training tokens in a vector space [27]. SVMs are commonly trained with soft margin optimization which allows training tokens to violate the SVM's separation margin constraint with some penalty [6].

In its standard form, an SVM is a two-class classifier. To create a multi-class SVM for a problem with N_T classes, a one-versus-rest SVM classifier is typically learned for each topic class t yielding the following scoring function:

$$S(\vec{x}, t) = -b_t + \sum_{\forall i} \alpha_{i,t} K(\vec{v}_i, \vec{x}). \tag{11}$$

Here, each vector $\vec{v}_i$ is a unique training vector or *support vector* from the set of training documents. Each $\alpha_{i,t}$ value represents the learned support vector weight for training vector i for topic class t. For notational simplicity, the $\alpha_{i,t}$ values here absorb the ± 1 valued class labels $y_{i,t}$ which are often included in the SVM expression. The b_t value is an offset which is typically set such that vectors lying on the decision hyperplane receive a neutral score of zero.

The function $K(\vec{v}, \vec{x})$ is a *kernel function* for comparing the vectors $\vec{v}$ and $\vec{x}$. While many kernel functions may be used in a support vector machine, a linear kernel function is generally adequate for typical topic ID problems and is used in this work. The linear kernel function is simply the dot product between the two component vectors

$$K(\vec{v}, \vec{x}) = \vec{v} \cdot \vec{x}. \tag{12}$$

The vectors in this SVM kernel function would typically be represented using either the TF-IDF or TF-LLR form.

C. Linear Classifier Representation

Both the naive Bayes and linear SVM classifiers are linear classifiers, i.e., classifiers that define one linear separating hyperplane per class in the feature vector space. For a single topic t, a linear classifier can be expressed as

$$S(\vec{x}, t) = \vec{r}_t \cdot \vec{x} - b_t \tag{13}$$

where $\vec{r}_t$ is a linear projection vector for topic t and b_t is a score offset which is typically set to center the topic decision boundary around the score of 0. From here, the set of projection vectors $\vec{r}_t$ can be concatenated to form a matrix R and the score offsets b_t can be concatenated to form a vector $\vec{b}$, thus creating a multi-class linear classifier defined as

$$\vec{s} = \begin{bmatrix} S(\vec{x}, t_1) \\ \vdots \\ S(\vec{x}, t_{N_T}) \end{bmatrix} = R\vec{x} - \vec{b}. \tag{14}$$

Within this interpretation, R is sometimes referred to as the *routing matrix* within the call routing literature [18], [20].

The naive Bayes expression in (9) can be written in the linear classifier form by expressing each element $r_{t,w}$ within R is as follows:

$$r_{t,w} = \log \frac{P(w \,|\, t)}{P(w \,|\, \bar{t})}. \tag{15}$$

Similarly, the linear kernel SVM expressed by (11) and (12) can be written in the linear classifier form by expressing each element $r_{t,w}$ within R as follows:

$$r_{t,w} = \sum_{\forall i} \alpha_{i,t} v_{i,w}. \tag{16}$$

Here, $v_{i,w}$ represents element w of support vector $\vec{v}_i$.

IV. MCE Training

A. MCE Training Overview

The primary focus of this paper is the use of minimum classification error (MCE) training on topic ID classifiers. MCE training adjusts the parameters of a classifier in order to minimize its classification error rate [17]. It applies an iterative gradient descent approach to minimize a smooth optimization function approximating the error rate of the classifier. This process begins with the computation of a misclassification measure $M(\vec{x})$ for a feature vector $\vec{x}$ being scored by a classifier. A simple misclassification measure for the classifier on vector $\vec{x}$ is expressed as follows:

$$M(\vec{x}) = S(\vec{x}, t_I) - S(\vec{x}, t_C). \tag{17}$$

Here t_C represents the correct topic for vector $\vec{x}$ (assuming $\vec{x}$ is assigned one and only one topic label) and t_I is the best scoring incorrect topic. In (17), documents with $M(\vec{x}) < 0$ are correctly classified, while $M(\vec{x}) > 0$ indicates a classification error. A step function applied to $M(\vec{x})$ could be used to indicate whether an error has been made. Because gradient descent training requires a smooth function that is differentiable, an error step function can instead be approximated with a sigmoid loss function as follows:

$$\ell(\vec{x}) = \frac{1}{1 + e^{-\beta M(\vec{x})}}. \tag{18}$$

Here, β is the slope factor of the sigmoid function. Larger values of β yield closer approximations to the step function. MCE seeks to minimize the average of value of $\ell(\vec{x})$ over all $\vec{x}$. For a specific $\vec{x}, \ell(\vec{x})$ can be differentiated with respect to any model parameter θ as follows:

$$\frac{\partial \ell(\vec{x})}{\partial \theta} = \beta \ell(\vec{x})(1 - \ell(\vec{x})) \frac{\partial M(\vec{x})}{\partial \theta}. \tag{19}$$

From the partial derivatives of the loss function over all parameters, gradient descent optimization is performed on these parameters. This can proceed either in sequential mode (where the parameters are updated after each document is observed) or in batch mode (where partial derivatives are accumulated over the whole document collection and a single update is performed at the end of iteration). The MCE batch update for a general parameter θ is as follows:

$$\theta^{[i+1]} = \theta^{[i]} - \frac{\epsilon}{N_D} \sum_{\forall d \in D} \frac{\partial \ell(\vec{x})}{\partial \theta}. \tag{20}$$

Here, i represents the ith iteration of MCE training and ϵ is a learning rate controlling the gradient descent step size. For sequential learning, the summation is removed and the update is performed after each individual document is observed. Training progresses until convergence of the average value of the loss function is achieved.

B. MCE Training From Multiple Incorrect Hypotheses

In (17), the misclassification measure is computed using only the single top scoring incorrect hypothesis. This expression can be generalized to incorporate information from all incorrect hypotheses as follows:

$$M(\vec{x}) = F(\vec{x}, \bar{t}_C) - S(\vec{x}, t_C) \tag{21}$$

where

$$F(\vec{x}, \bar{t}_C) = \frac{1}{\eta} \log \left[\frac{1}{N_T - 1} \sum_{\forall t \neq t_C} e^{\eta S(\vec{x}, t)} \right]. \tag{22}$$

Here, the score for the single best incorrect topic is replaced with a weighted combination of the scores from all incorrect topics with the worst scoring topics contributing the most to the score. The weighting of the individual topics is controlled by the scaling factor η. When $\eta \rightarrow \infty$, then (21) reduces to the form of (17).

C. MCE Training of Feature Weights

In practice the process of feature selection is generally found to be beneficial to topic ID systems. By ignoring words that carry little topic discrimination ability, the noise created by the presence of these words in the decision process is removed and accuracy is improved. However, the feature selection process is a blunt instrument; words are deemed either useful and kept, or useless and ignored. A more nuanced approached would be to assign continuous valued weights to the features. For example, the basic linear classifier expression in (14) could be rewritten as

$$\vec{s} = R(\vec{\lambda} * \vec{x}) - \vec{b}. \tag{23}$$

Here, the $*$ operator performs a term-wise multiplication of vectors $\vec{\lambda}$ and $\vec{x}$, where each element λ_w of $\vec{\lambda}$ serves as a continuous valued feature weight applied to the corresponding feature x_w. If each $\vec{x}$ consists of the full set of word features in the vocabulary V, then a feature selection process could be viewed as setting $\lambda_w = 1$ for all selected words and $\lambda_w = 0$ for all ignored words. In our previous work, we have explored learning the feature weights in $\vec{\lambda}$ automatically using the MCE training algorithm for either the naive Bayes classifier [13] or the SVM classifier [14]. Because both classifiers can be reduced to the linear classifier form of (23), the MCE training process is largely identical for both. Also note that MCE training of feature weights can be applied to either the full vocabulary of all words, or to only a preselected set of words retained by the feature selection process.

To perform MCE training of the weights in $\vec{\lambda}$, the MCE algorithm computes the partial derivative of the misclassification measure $M(\vec{x})$ [as required by (19)] with respect to each feature weight λ_w within $\vec{\lambda}$ as follows:

$$\frac{\partial M(\vec{x})}{\partial \lambda_w} = x_w \left(-r_{t_C, w} + \sum_{\forall t \neq t_C} \gamma_t r_{t,w} \right). \tag{24}$$

Here, γ_t resembles a posterior probability over the incorrect topics as defined by

$$\gamma_t = \frac{e^{\eta S(\vec{x}, t)}}{\sum_{\forall t_i \neq t_C} e^{\eta S(\vec{x}, t_i)}}. \tag{25}$$

Here, the variable η can be viewed as a posterior scaling factor. As $\eta \to \infty$, then $\gamma_{t_I} \to 1$ for the best scoring incorrect topic t_I, and $\gamma_t \to 0$ for all other incorrect topics $t \neq t_I$.

When updating the feature vector weights, the average loss function can often be improved simply by jointly increasing the magnitude of all feature weights, i.e., whenever there are sufficiently more correctly classified training vectors than incorrectly classified vectors the average loss is reduced simply by pushing all training tokens further away from the decision boundary. To avoid this trap, normalization is applied to force the L1 distance of the feature weights to remain fixed during iterative updating. Thus, the following normalization is applied after every training iteration i:

$$\sum_w \lambda_w^{[i]} = \sum_w \lambda_w^{[0]}. \tag{26}$$

D. Jackknife Training

When training the feature weighting vector $\vec{\lambda}$, we would ideally like to learn which features are most useful for topic identification using previously unseen training data (i.e., training data not used to train R). This would help alleviate the potential problem of over-tuning the feature weighting vector to the training data. On the other hand, we would also like to use all available data to train the routing matrix R if at all possible. One solution to this problem is to perform data jackknifing when training $\vec{\lambda}$. In this approach, we subdivide the training data D into N_P separate non-overlapping partitions $P_1, \ldots, P_{N_P}$. For each partition P_i, we can train a routing matrix $R_{\bar{i}}$ using all of the data except the documents in partition P_i. We can then perform MCE updating on each training vector $\vec{v} \in P_i$ using $R_{\bar{i}}$ instead of R. In this fashion every training vector $\vec{v}_1, \ldots, \vec{v}_{N_D}$ can be treated as an unseen vector $\vec{x}$ during MCE training of $\vec{\lambda}$, while the final routing matrix R can still be trained using all available training data. In the extreme case, we can set $N_P = N_D$ such that the jackknife training is equivalent to *leave-one-out* training.

E. MCE Optimization of SVM Parameters

Recall from Section III-B, that the basic form of an SVM classifier for topic t is

$$S(\vec{x}, t) = -b_t + \sum_{\forall i} \alpha_{i,t} K(\vec{v}_i, \vec{x}). \tag{27}$$

Recall also that support vector weights $\alpha_{i,t}$ absorb the ± 1 valued class labels $y_{i,t}$ in our notation. Because of the geometric constraints imposed in SVM training, the positive training tokens for class t have positive valued weights $\alpha_{i,t}$, while the negative training tokens have negative weights. Furthermore, if the set of training vectors is divided into the set of positive document exemplars for class t, D_t^+, and the set of negative document exemplars for class t, D_t^-, the following equality holds:

$$\left(\sum_{\forall i : \vec{v}_i \in D_t^+} \alpha_{i,t} \right) = \left(\sum_{\forall i : \vec{v}_i \in D_t^-} -\alpha_{i,t} \right) = a_t. \tag{28}$$

Here, we have introduced the term a_t, which we will refer to as the SVM scaling factor for class t. We can further define a set of prescaled support vector weights as

$$\omega_{i,t} = \alpha_{i,t} / a_t. \tag{29}$$

We should also note that factoring the scale factor a_t out of the summation in (28) yields the following constraint:

$$\left(\sum_{\forall i : \vec{v}_i \in D_t^+} \omega_{i,t} \right) = \left(\sum_{\forall i : \vec{v}_i \in D_t^-} -\omega_{i,t} \right) = 1. \tag{30}$$

Next, we can define the weighted sum of kernel scores for the positive and negative support vectors independently as

$$U^+(\vec{x}, t) = \sum_{\forall i : \vec{v}_i \in D_t^+} \omega_{i,t} K(\vec{v}_i, \vec{x}) \tag{31}$$

$$U^-(\vec{x}, t) = \sum_{\forall i : \vec{v}_i \in D_t^-} \omega_{i,t} K(\vec{v}_i, \vec{x}). \tag{32}$$

We next define the full unscaled SVM score $U(\vec{x}, t)$ to be

$$U(\vec{x}, t) = U^+(\vec{x}, t) + U^-(\vec{x}, t). \tag{33}$$

Using these definitions we can rewrite (27) as

$$S(\vec{x}, t) = -b_t + a_t U(\vec{x}, t). \tag{34}$$

Using this interpretation of the SVM expression, the discriminative capabilities of the individual two-class SVMs created for each class t are captured by the set of class specific support vector weights $\omega_{i,t}$ contained within $U(\vec{x}, t)$, while the calibration of the full multi-class SVM is captured by the settings of the scale factors a_t and decision thresholds b_t.

To optimize our multi-class SVM system, we use MCE training applied to the individual parameters of the SVM classifier. We will refer to a process that optimizes only the SVM scale and threshold parameters, a_t and b_t, as *calibration*. If we additionally train the prescaled support vector weights $\omega_{i,t}$, we will refer to this as *full optimization*.

While SVM training often selects a sparse subset of training vectors to be support vectors, our derivation uses all training vectors within the SVM expression in (27), with the non-support vectors carrying a weight of zero. MCE training thus allows training vectors that start with an initial weight of zero to obtain a nonzero weight during optimization.

For MCE training we must first compute the partial derivatives of the misclassification measure $M(\vec{x})$ with respect to each parameter. The expressions of these derivatives for each a_t, b_t and $\omega_{i,t}$ parameter are

$$\frac{\partial M(\vec{x})}{\partial a_t} = \begin{cases} -U(\vec{x}, t), & \text{if } t = t_C \\ \gamma_t U(\vec{x}, t), & \text{if } t \neq t_C \end{cases} \tag{35}$$

$$\frac{\partial M(\vec{x})}{\partial b_t} = \begin{cases} 1, & \text{if } t = t_C \\ -\gamma_t, & \text{if } t \neq t_C \end{cases} \tag{36}$$

$$\frac{\partial M(\vec{x})}{\partial \omega_{i,t}} = \begin{cases} -a_t K(\vec{v}_i, \vec{x}), & \text{if } t = t_C \\ \gamma_t a_t K(\vec{v}_i, \vec{x}), & \text{if } t \neq t_C \end{cases}. \tag{37}$$

Here, the γ_t parameters are posterior-like weights over the incorrect classes as defined in (25).

To maintain scoring within the same dynamic range as the original classifier, the collection of a_t and b_t terms are normalized after each training iteration i to adhere to these L1 distance constraints

$$\sum_{\forall t} a_t^{[i]} = \sum_{\forall t} a_t^{[0]} \quad \text{and} \quad \sum_{\forall t} b_t^{[i]} = \sum_{\forall t} b_t^{[0]}. \qquad (38)$$

We also require the support vector weights to adhere to the constraints given in (30) and we impose the constraint that the weights for the positive training tokens of a class must be non-negative while the weights for the negative training tokens must be non-positive.

During full optimization, the a_t and b_t are first optimized using batch training, while the $\omega_{i,t}$ weights are held fixed, and then the $\omega_{i,t}$ weights are optimized using batch training while the a_t and b_t parameters are held fixed. The training procedure iteratively alternates between these two optimization steps until convergence is reached.

F. Leave-One-Out Scoring for SVMs

While jackknife style training can be used with SVMs, it would require separate SVM training runs for each partition. This can become computationally cumbersome, if not prohibitively expensive, as the number of partitions grows. To address this issue, we have developed an efficient approximation to leave-one-out scoring for SVMs.

In leave-one-out scoring, a training vector $\vec{v}_j$ to be scored must be removed from the training set, the trained models must be adjusted to account for the removal of this vector, and the adjusted model must produce a score for the held out vector. From this score the loss $\ell(\vec{v}_j)$ can be determined and the appropriate MCE updates can be computed for the held out vector. Our approach approximates the score of an SVM model by removing the training vector $\vec{v}_j$ as a support vector and renormalizing the weights of the remaining support vectors to account for the lost weight of the held-out vector. In the case that the held-out vector $\vec{v}_j$ is a positive support vector for class t, the computation of $U^+(\vec{v}_j, t)$ is approximated as follows:

$$U^+(\vec{v}_j, t) = \sum_{\forall i: i \neq j, \vec{v}_i \in D_t^+} \frac{\omega_{i,t}}{1-\omega_{j,t}} K(\vec{v}_i, \vec{v}_j). \qquad (39)$$

This expression is not valid for the degenerate case where $\omega_{j,t} = 1$, but in our experiments we have yet to encounter this case. Should this occur one could approximate $U^+(\vec{v}_j, t)$ by assigning uniform weights to the remaining collection of positive training vectors for class t. For the case where the held-out vector $\vec{v}_j$ is a negative support vector for class t, we similarly adjust the computation of $U^-(\vec{v}_j, t)$ as follows:

$$U^-(\vec{v}_j, t) = \sum_{\forall i: i \neq j, \vec{v}_i \in D_t^-} \frac{\omega_{i,t}}{1+\omega_{j,t}} K(\vec{v}_i, \vec{v}_j). \qquad (40)$$

Using this leave-one-out scoring approximation, every training vector $\vec{v}$ in the training set can be used as an unseen observation during MCE training.

G. Comparison to Previous Work

At this point, it important to contrast the MCE training presented in this paper with the previous use of MCE training in the call routing field [18], [20], [30]. All of these previous efforts used MCE to optimize a routing matrix R. Various classifiers were used to initialize R including classifiers based on cosine similarity metrics [18], naive Bayes classifiers [20] and Beta classifiers [30]. In each of the previous efforts the routing matrix R is optimized with MCE training directly on the training data. Despite the successes of these previous efforts, these approaches have some drawbacks.

First, special care was required in these efforts to avoid over-training to the training set. This is likely because the full routing matrix required a large number of parameters ($N_V \times N_T$ parameters for a system with N_V features and N_T topics). In our work, over-training is at least partially ameliorated through the training of smaller sets of tied parameters, such as the N_V different feature weights in $\vec{\lambda}$, or the set of $N_D \times N_T$ sparse support vector weights present in the SVM.

Next, by optimizing the routing matrix directly on the training data, these previous MCE approaches cannot improve performance if the initial classification error rate on the training data is already very small (or zero) because little further improvement can be gained from additional MCE training. The jackknife and leave-one-out training methods used in our work allow all training tokens to be treated as unseen data during MCE training thus improving generalization to actual unseen data. Additionally, when directly optimizing the full routing matrix, the underlying structure of the original classifier is inherently lost during the MCE training process. As such, there is not an obvious avenue for employing similar jackknife or leave-one-out training with these previous methods.

V. Experimental Details

A. Data Set

Our experiments used the English Phase 1 portion of the Fisher Corpus containing 10-minute-long telephone conversations between two people [5]. Each conversation discussed one topic chosen from a set of 40 prespecified topics. For this work, the corpus was subdivided into three subsets: 1) a 553 hour recognizer training set containing 3104 calls; 2) a 244 hour topic ID training set with 1374 calls; and 3) a 226 hour topic ID test set with 1372 calls.

B. Word-Based Speech Recognition

For word-based ASR we use the MIT SUMMIT speech recognizer [9]. The system's acoustic models were trained using a standard maximum-likelihood approach on the full 553 hour recognition training set without any form of speaker normalization or adaptation. For language modeling, the system uses a basic trigram language model with a 31.5 K word vocabulary trained using the transcripts of the recognizer training set. This system performs recognition faster than real time (on a current PC) but its error rate is high (over 40%).

When the recognizer was run on the topic ID training set, only 30 364 words from the full recognizer vocabulary were observed in the resulting collection of lattice outputs. Thus, these 30 364 words represent the full vocabulary set actually used in the topic ID experiments.

C. Training and Testing Details

In our experiments, every call in the topic ID train and test sets is represented by a single feature vector containing the expected counts of the words observed during the call as hypothesized by the ASR system. The topic ID models are trained and optimized using only the topic ID training set, and topic ID performance is measured on the topic ID test set. Performance in this paper is measured using the classification error rate for the closed set topic ID scenario.

In our SVM experiments, the initial SVM is trained using the open source LIBSVM Package [4] with a linear kernel and a margin cost of 10. The training data in our experiments is linearly separable for most feature set sizes, so we did not explore any other kernel functions. Also, all settings of the SVM margin cost greater than 1 generally yield identical results while performance degrades for costs less than 1 (i.e., the SVM margin term dominates the soft margin training).

D. MCE Parameter Settings

When using the MCE training algorithms described in this work, appropriate values must be set for the sigmoid slope factor β, the posterior scale factor η of the scoring function for incorrect topics, and the MCE learning rate parameter ϵ. In this work, we set $\beta = 10$ and $\eta = 100$ in all experiments. We should note that our investigations have shown that performance is not very sensitive to the setting of η though performance does degrade severely when $\eta < 10$. Across the wide range of values $\eta > 10$, little difference in performance is observed, though very large values of η can result in slower convergence during training. For $\eta = 100$, it is typically the case that only the top 1 to 5 incorrect topic hypotheses receive non-negligible weights during MCE training updates. The system is more sensitive to the setting of β though values ranging between $5 < \beta < 20$ all give very similar results.

It is important for the learning rate ϵ to be appropriately set in order for efficient convergence of MCE training to occur. Our system adaptively tunes ϵ during MCE training using a modified version of the RPROP algorithm [24]. RPROP increases or decreases ϵ in response to the observed rate of convergence of the average loss function over the previous three training iterations. In our experiments ϵ is initialized to a value of 1, but it is not uncommon for it to vary by numerous orders in magnitude during training. Using this approach, MCE training in our experiments typically requires between 50 and 500 iterations to fully converge. We must also note that the MCE algorithm is also only guaranteed to converge to a local minimum in the average loss function and not necessarily to the globally optimal solution.

In our experiments, sequential MCE training was used when learning the feature weights in the naive Bayes system. In practice, batch training yielded similar results in the naive Bayes system but its convergence during training was slower. Batch MCE training was used in all SVM experiments, as sequential training (for reasons we have not yet determined) behaved erratically in the SVM system.

VI. Experimental Results

A. ASR Lattice Versus ASR 1-Best Result

Fig. 1 shows the baseline performance for the naive Bayes classifier, and for the SVM classifier using TF-LLR feature weighting, as the number of preselected features is varied from the full vocabulary of 30 364 words down to only 384 words. Feature selection is performed as discussed in Section II-C. The N most discriminative words per topic were selected using (2) for different values of N ranging from 10 to 1000. Performance using the 1-best ASR hypotheses is compared against using ASR lattices. The full lattices are particularly beneficial to the SVM system but also yield improvements for the naive Bayes system. All other experiments in this paper are conducted using ASR lattices.

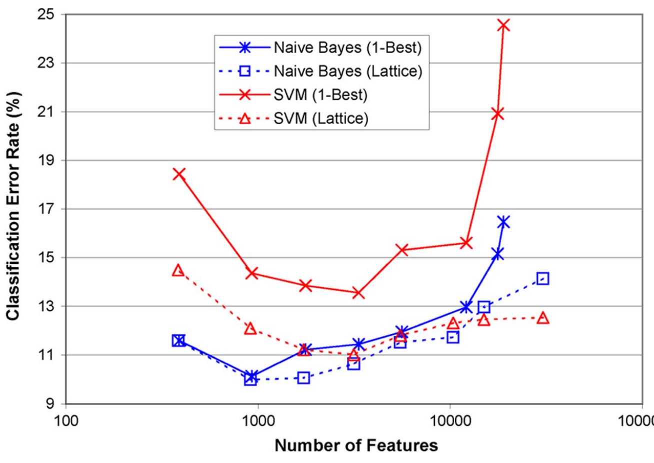

Fig. 1. Closed set topic classification results for naive Bayes and SVM classifiers using 1-best ASR output and full ASR lattices, as the number of preselected features is varied from the full vocabulary size of over 30 K words down to 384 words.

B. Convergence of MCE Training of Feature Weights

Fig. 2 shows the performance of the full vocabulary naive Bayes system during MCE training of the feature weights. The figure shows both the classification error rate and the average value of the loss function (expressed as a percentage) for both the training set and the (unseen) test set. For the training set, these metrics are computed over all held-out training partitions used during the jackknife process. On the training data, the loss function decreases monotonically until convergence is reached. On the test data, some evidence of training overshoot is observed when the MCE learning rate reaches a peak value of $\epsilon = 626$ at iteration 73. Starting at iteration 73 the RPROP algorithm begins a series of learning rate reductions which restore the learning process towards smooth and continuous improvement on both the training data and the test set. Similar MCE learning patterns are observed in our other experiments.

C. Results for MCE Training of Feature Weights

Fig. 3 shows the closed-set classification results for both the naive Bayes (NB) classifier and the SVM classifier using TF-LLR feature weighting. Results are shown both with and without the application of MCE feature weight (FW) training as the number of preselected features is varied from the full vocabulary of 30 364 words down to 384 words. MCE feature weight training on the NB system is performed using the jackknifing approach with ten training partitions. MCE feature weight training on the SVM system is performed using the SVM leave-one-out approximation.

As seen in the figure, the standard NB and SVM systems both benefit from the application of feature selection. The best NB system achieves an error rate of 10.0% using 912 features (down

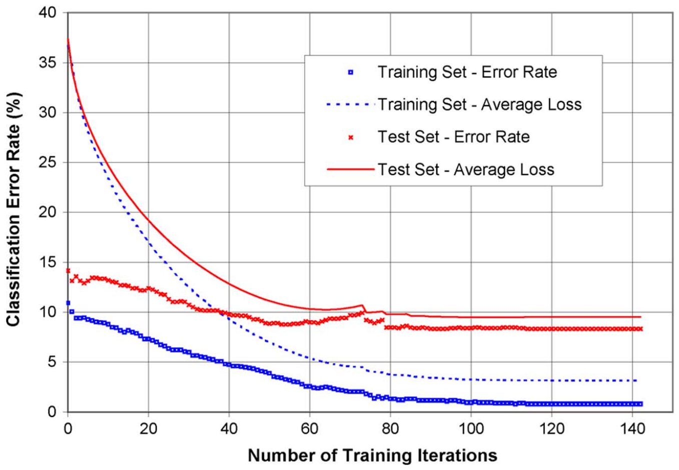

Fig. 2. Closed set topic classification error rate and average loss on the train and test sets for the models produced during each iteration of MCE training of feature weights for the naive Bayes system using the full vocabulary.

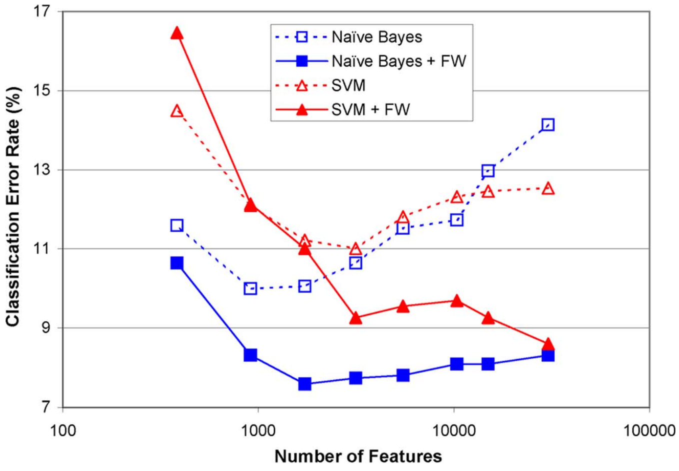

Fig. 3. Closed set topic classification results for naive Bayes and SVM classifiers, both with and without MCE feature weight (FW) training, as the number of preselected features is varied from the full vocabulary size of 30 K words down to 384 words.

from 14.1% using the full vocabulary). The standard SVM performs better than the naive Bayes system when using the full vocabulary (12.5% versus 14.1%) but it also sees less benefit from feature selection with its best performance of 11.0% achieved with a feature set of 3154 preselected words.

Both the NB and SVM systems see substantial improvements in accuracy when MCE feature weight training is applied. The error rate of the NB system using the full vocabulary is reduced from 14.1% to 8.3% after applying the MCE training. This full vocabulary result is close to the best NB result using MCE training of 7.7% achieved with a feature set size of 3151 words. With MCE feature weight training, the SVM system achieves its best accuracy of 8.6% when using the full vocabulary.

D. Results for MCE Training of SVM Parameters

Fig. 4 shows the classification results when applying MCE training to parameters of the TF-LLR SVM classifier. The results compare the basic SVM system and the SVM system with MCE FW training against the basic SVM trained with the MCE calibration and MCE optimization processes described in Section IV.E. The results show that MCE calibration improves the basic SVM when the feature set size is large but the calibration process is not as effective as MCE FW training. On

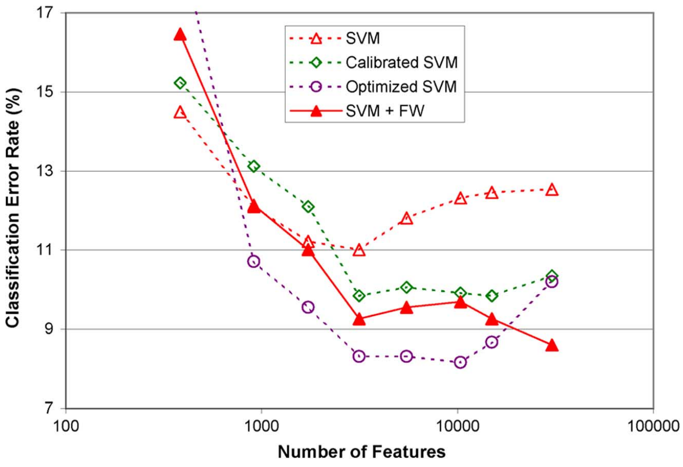

Fig. 4. Closed set topic classification results for the basic SVM, the MCE calibrated SVM, the MCE optimized SVM, and the basic SVM with MCE feature weight (FW) training, as the number of preselected features is varied from the full vocabulary of 30 K words down to 384 words.

the other hand, the full MCE optimization of the SVM yields better results than MCE FW training at all but the smallest and largest feature set sizes. The MCE optimization of the SVM parameters also provides sizeable improvements over the baseline SVM at all but the smallest feature set sizes, thereby showing the efficacy of this training procedure.

E. Results for Full MCE Training

Fig. 5 shows the results when combining MCE training of feature weights with MCE training of SVM parameters within the TF-LLR SVM classifier. For comparison, the figure first shows the naive Bayes system with MCE FW training from Fig. 3 and the SVM system with MCE FW training from Fig. 4. The figure then shows the SVM system with FW training when it is subsequently trained using MCE calibration and full MCE optimization. In these cases, MCE calibration has little additional effect on performance, perhaps because the MCE FW training is implicitly creating a system with better calibrated scores. On the other hand, full MCE optimization does provide some additional benefit beyond MCE FW training for the larger feature set sizes.

It is important to note that MCE optimization in our experiments was performed once after MCE FW training was applied to the initial SVM models. Alternatively, the MCE FW training and MCE optimization stages could be iteratively reapplied multiple times. We performed this experiment as well, but found that this approach resulted in over-tuning to the training set and degraded performance on the test set.

F. Importance of Jackknife Training

Our MCE training techniques rely on the use of either jackknife training (for the naive Bayes systems) or leave-one-out training (for the SVM systems). This process is particularly important for success on our data set. When processing high-dimensionality data, it is very easy to create a classifier that has little to no error on the training data. In our final experiments (presented in the next subsection), the baseline SVM systems are able to perfectly separate the training data achieving an error rate of 0% on the training data. As a result, any further MCE training of the SVM applied directly to the training data without using leave-one-out

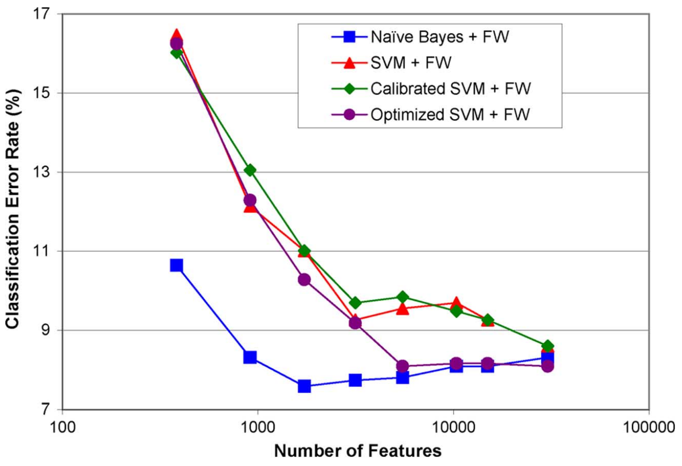

Fig. 5. Closed set topic classification results for the NB classifier with MCE feature weight (FW) training, the basic SVM with MCE FW training, the MCE calibrated SVM with MCE FW training, and the MCE optimized SVM with MCE FW training as the number of preselected features is varied from the full vocabulary of 30 K words down to 384 words.

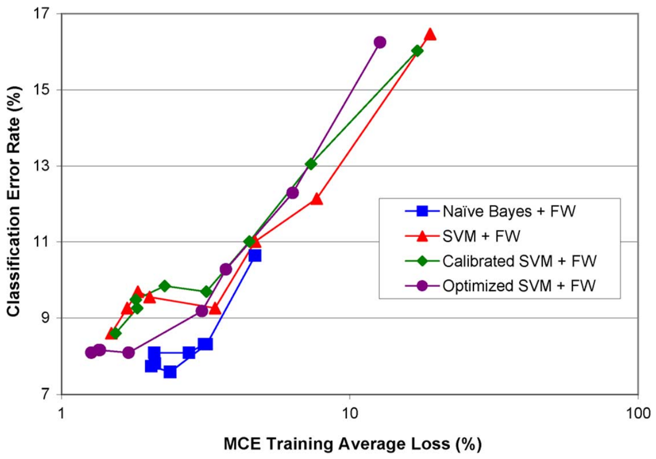

Fig. 6. Closed set topic classification error rate versus MCE training average loss for the NB classifier with MCE feature weight (FW) training, the basic SVM with MCE FW training, the MCE calibrated SVM with MCE FW training, and the MCE optimized SVM with MCE FW training as the number of preselected features is varied from the full vocabulary of 30 K words down to 384 words.

training yields no discernible effect. Our baseline naive Bayes system also has a very low training error of 2.3%. Thus, MCE training without jackknifing performs corrective training on only a small number of errors, and fails to yield any improvements that generalize to the test data. We also attempted to directly optimize the full routing matrices initialized using naive Bayes and SVM classifiers on the training data using the MCE approach of Kuo and Lee [18] and Liu *et al.* [20]. These attempts resulted in no improvement to the SVM system and a statistically insignificant degradation of the naive Bayes system.

G. Correlation of Training Loss and CER

In examining Figs. 3–5, it is easy to recognize that optimal performance on the test set can be achieved only if the system can predetermine the best feature set size to use. Towards this end, Fig. 6 shows the relationship between the classification error rate on the test set and the average loss of the system on the training set for the four systems presented in Fig. 5. These plots show a very clear correlation between the training loss and the classification performance on the final test set when comparing different feature set sizes used within a particular classifier configuration. This demonstrates that the selection of the best feature set size to use for a particular classifier can be reasonably determined by choosing the feature set size that produces the smallest average loss on the (held-out) training data.

H. Summary of Final Results

Table I summarizes our results using various forms of MCE training on the naive Bayes classifier and on the SVM classifier using either TF-LLR or TF-IDF normalization. For each specific classifier and MCE training condition, eight different feature sets ranging in size from only 384 words up to the full vocabulary were used during training. The feature set that achieved the lowest average loss over the training data was selected for use on the test set. The results in terms of total classification errors and classification error rate are displayed for the automatically selected feature sets in Table I.

In Table I, the best baseline system on this task is the naive Bayes system using only 912 preselected words as features. This

TABLE I
TOPIC ID PERFORMANCE FOR THE VARIOUS CLASSIFIERS DESCRIBED IN THIS PAPER USING THE SET OF PRESELECTED FEATURES FOR EACH CLASSIFIER THAT YIELDS THE MINIMUM AVERAGE LOSS OVER JACKKNIFED OR LEAVE-ONE-OUT TRAINING DATA

Classifier	MCE Training	# of Features	# of Errors	Error Rate (%)
Naive Bayes	None	912	137	10.0
	FW	3151	106	7.7
TF-LLR SVM	None	14696	171	12.5
	Calibration	10354	136	9.9
	Optimization	3151	114	8.3
	FW	30364	118	8.6
	FW + Calibration	30364	118	8.6
	FW + Optimization	30364	111	8.1
TF-IDF SVM	None	30364	156	11.4
	Calibration	30364	147	10.7
	Optimization	14946	142	10.3
	FW	30364	136	9.9
	FW + Calibration	10354	134	9.8
	FW + Optimization	30364	125	9.1

system achieves an error rate of 10.0% and outperforms both baseline SVM systems. When MCE feature weight training is applied to the naive Bayes system, we find that the selected feature set increases in size to 3151 words, and the error rate is reduced by a relative 23% down to 7.7%. This improvement is statistically significant to a level of $p = 0.005$ using a McNemar significance test. This is also the best result obtained in these experiments.

In examining the effect of MCE training on the SVM systems, we observe that all forms of MCE training introduced in this paper improve the performance of the SVM models. When the combination of MCE feature weight training and full MCE optimization of the SVM parameters are used in sequence, the TF-LLR SVM sees a 35% reduction in error rate from 12.5% down to 8.1% while the TF-IDF SVM sees a 20% reduction in error rate from 11.4% down to 9.1%. Though the baseline TF-IDF SVM system achieves a better result than the baseline TF-LLR SVM system, it is interesting to note that the TF-LLR system is better after MCE training is applied.

In the TF-LLR SVM results, all five of the different combinations of MCE training yielded statistically significant improvements over the baseline SVM system to a level of $p = 0.05$ or better using a McNemar significance test. However, almost all of the performance differences between these five combinations of MCE training applied to the TF-LLR SVM are not statistically significant. In the TF-IDF SVM results, only the differences between the three systems using MCE training of the feature weights (FW) and the baseline TF-IDF SVM system were statistically significant. Thus, further experiments on a larger data set will be needed to firmly determine if the gains observed from the combination of training both feature weights and SVM parameters represent a convincingly useful combination of training techniques.

Although the results also show the MCE trained naive Bayes system outperforming the best SVM systems, we should note that this result is unusual as a majority of the text categorization literature points to the SVM approach being superior to the naive Bayes approach. However, a closer examination of Figs. 3–5 reveals that the naive Bayes system reaches its best performance with a relatively small number of features while the SVM system achieves better performance as the dimensionality grows. Thus, when using the MCE techniques presented in this paper, one might expect better relative performance from SVM systems on problems with a larger feature space than the one used in these experiments.

VII. Conclusion

In this paper, we have discussed the use of minimum classification error (MCE) training as a means for improving naive Bayes and SVM classifiers for the topic ID problem. We have presented novel MCE training techniques which can efficiently apply jackknifing or leave-one-out training to yield improved models which generalize better to unseen data. Experiments were conducted on data from the Fisher Corpus using feature vector representations from word-based automatic speech recognition lattices. Our experiments demonstrated sizeable improvements in topic identification accuracy using the new MCE training techniques. In future work, we plan to apply these techniques on other topic ID corpora and to other problems such as automatic language identification.

Acknowledgment

The author would like to thank F. Richardson and A. Margolis for their contributions to the early development of the techniques presented in this paper.

References

[1] J. Arenas-García and F. Pérez-Cruz, “Multi-class support vector machines: A new approach,” in *Proc. ICASSP*, Hong Kong, China, Apr. 2003, pp. 781–784.

[2] N. Brummer and D. van Leeuwan, “On calibration of language recognition scores,” in *Proc. Speaker Lang. Recognition Workshop*, San Juan, Puerto Rico, Jun. 2006.

[3] W. Campbell *et al.*, “Phonetic speaker recognition with support vector machines,” in *Adv. Neural Inf. Process. Syst. 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[4] C.-C. Chang and C.-J. Lin, “LIBSVM—A Library for Support Vector Machines,” [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[5] C. Cieri, D. Miller, and K. Walker, “The Fisher corpus: A resource for the next generation of speech-to-text,” in *Proc. Int. Conf. Lang. Resources Eval.*, Lisbon, Portugal, May 2004, pp. 69–71.

[6] R. Collobert and S. Bengio, “SVMTorch: Support vector machines for large-scale regression problems,” *J. Mach. Learn. Res.*, vol. 1, pp. 143–160, 2001.

[7] K. Crammer and Y. Singer, “On the algorithmic implementation of multi-class kernel-based vector machines,” *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, 2001.

[8] S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua, “A MFoM learning approach to robust multiclass multi-label text categorization,” in *Proc. ICML*, Banff, AB, Canada, Jul. 2004, pp. 42–49.

[9] J. Glass, “A probabilistic framework for segment-based speech recognition,” *Comput. Speech Lang.*, vol. 17, no. 2–3, pp. 137–152, 2003.

[10] P. Haffner, G. Tur, and J. Wright, “Optimizing SVMs for complex call classification,” in *Proc. ICASSP*, Hong Kong, China, Apr. 2003, pp. 632–635.

[11] A. Hatch, A. Stolcke, and B. Peskin, “Combining feature sets with support vector machines: Application to speaker recognition,” in *Proc. IEEE Workshop Autom. Speech Recognition Understand.*, San Juan, Puerto Rico, Nov. 2005, pp. 75–79.

[12] T. Hazen, F. Richardson, and A. Margolis, “Topic identification from audio recordings using word and phone recognition lattices,” in *Proc. IEEE Workshop Autom. Speech Recognition Understand.*, Kyoto, Japan, Dec. 2007, pp. 659–664.

[13] T. Hazen and A. Margolis, “Discriminative feature weighting using MCE training for topic identification of spoken audio recordings,” in *Proc. IEEE ICASSP*, Las Vegas, NV, Apr. 2008, pp. 4965–4968.

[14] T. Hazen and F. Richardson, “A hybrid SVM/MCE training approach for vector space topic identification of spoken audio recordings,” in *Proc. Interspeech*, Brisbane, Australia, Sep. 2008, pp. 2542–2545.

[15] T. Hazen, “Multi-class SVM optimization using MCE training with application to topic identification,” in *Proc. IEEE ICASSP*, Dallas, TX, Mar. 2010, pp. 5350–5353.

[16] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *Proc. Eur. Conf. Mach. Learn.*, Chemnitz, Germany, Apr. 1998, pp. 137–142.

[17] B.-H. Juang and S. Katagiri, “Discriminative learning for minimum error classification,” *IEEE Trans. Signal Process.*, vol. 40, no. 12, pp. 3043–3054, Dec. 1992.

[18] H.-K. J. Kuo and C.-H. Lee, “Discriminative training of natural language call routers,” *IEEE Trans. Speech Audio Process.*, vol. 11, no. 1, pp. 24–35, Jan. 2003.

[19] D. Lewis, “Naive (Bayes) at forty: The independence assumption in information retrieval,” in *Proc. Eur. Conf. Mach. Learn.*, Chemnitz, Germany, Apr. 1998, pp. 4–15.

[20] P. Liu, H. Jiang, and I. Zitouni, “Discriminative training of naive Bayes classifiers for natural language call routing,” in *Proc. Interspeech*, Jeju Island, Korea, Oct. 2004, pp. 1589–1592.

[21] J. McDonough *et al.*, “Approaches to topic identification on the switchboard corpus,” in *Proc. ICASSP*, Adelaide, Australia, Apr. 1994, pp. 385–388.

[22] C. Manning and H. Schütze, “Text categorization,” in *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999, ch. 16, pp. 575–608.

[23] J. Platt, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” in *Advances in Large Margin Classifiers*. Cambridge, MA: MIT Press, 1999, pp. 61–74.

[24] M. Riedmiller and H. Braun, “RPROP—A fast adaptive learning algorithm,” in *Proc. Int. Symp. Comput. Inf.*, Antalya, Turkey, 1992, pp. 279–285.

[25] R. Schwartz, T. Imai, L. Nguyen, and J. Makhoul, “A maximum likelihood model for topic classification of broadcast news,” in *Proc. Eurospeech*, Rhodes, Greece, Sep. 1997, pp. 1455–1458.

[26] F. Sebastiani, “Machine learning in automated text categorization,” *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.

[27] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[28] J. Wintrode and S. Kulp, “Techniques for rapid and robust topic identification of conversational telephone speech,” in *Proc. Interspeech*, Brighton, U.K., Sept. 2009, pp. 1471–1474.

[29] Y. Yang and J. Pedersen, “A comparative study on feature selection in text categorization,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Nashville, TN, Jul. 1997, pp. 412–420.

[30] I. Zitouni, “Constrained minimization and discriminative training for natural language call routing,” *IEEE Trans. Speech Audio Process.*, vol. 16, no. 1, pp. 208–215, Jan. 2008.

Timothy J. Hazen (M’04–SM’10) received the S.B., S.M., and Ph.D. degrees from the Massachusetts Institute of Technology (MIT), Cambridge, in 1991, 1993, and 1998, respectively.

From 1998 until 2007, he was a Research Scientist at the MIT Computer Science and Artificial Intelligence Laboratory. Since 2007, he has been a Member of the Technical Staff at MIT Lincoln Laboratory, Lexington, MA.

Dr. Hazen has previously served as an Associate Editor of the IEEE Transactions on Audio, Speech, and Language Processing and as a member of the IEEE Signal Processing Society’s Speech and Language Processing Technical Committee.