

Spoken Language Recognition Using Ensemble Classifiers

Bin Ma, *Senior Member, IEEE*, Haizhou Li, *Senior Member, IEEE*, and Rong Tong

Abstract— In this paper, we study a novel approach to spoken language recognition using an ensemble of binary classifiers. In this framework, we begin by representing a speech utterance with a high dimensional feature vector such as the phonotactic characteristics or the polynomial expansion of cepstral features. A binary classifier can be built based on such feature vectors. We adopt distributed output coding strategy in ensemble classifier design, where we decompose a multi-class language recognition problem into many binary classification tasks, each of which addresses a language recognition subtask by using a component classifier. Then we combine the results of the component classifiers to form an output code as a hypothesized solution to the overall language recognition problem. In this way, we effectively project high dimensional feature vectors into a tractable low dimensional space, yet maintaining language discriminative characteristics of the spoken utterances. By fusing the output codes from both phonotactic features and cepstral features, we achieve equal-error-rate of 1.38% and 3.20% for 30-second trials on the 2003 and 2005 NIST language recognition evaluation databases.

Index Terms - Ensemble Classifiers, Component Classifier Selection, Output Codes, Spoken Language Recognition

I. INTRODUCTION

AUTOMATIC spoken language recognition is a process of determining the language spoken in an utterance. As multilingual applications are becoming increasingly popular due to globalization and the growth of international business, spoken language recognition has become an essential technology in areas such as multilingual conversational systems, spoken language translation, multilingual speech recognition, and spoken document retrieval.

One of the fundamental issues in automatic spoken language recognition is to explore the discriminative cues for spoken languages. These cues, such as acoustic features and phonotactic representations, reflect different aspects of spoken language characteristics. Another issue is how to effectively organize and exploit the language cues in the classifier design for the best performance.

The state-of-the-art systems rely on two types of features: the acoustic features and the phonotactic features. The acoustic features reflect low-level spectral characteristics, while the phonotactic features represent the phonological constraints that govern a spoken language. Both features have

been shown to be effective in spoken language recognition [1][2].

Gaussian mixture model (GMM) is commonly used to exploit the acoustic features [1]. Recently, Campbell et al. proposed representing a spoken utterance in a high-dimensional feature vector using the generalized linear discriminant sequence (GLDS) kernel. The GLDS kernel expands acoustic features using a monomial basis for support vector machines (SVM) classification, leading to successful applications in language and speaker recognition tasks [3]. The kernel allows us to compare spoken utterances by measuring the similarity between the spectral polynomial expansion vectors. To capture temporal information across multiple frames, shifted delta cepstral (SDC) coefficients [4] were used in addition to the frame-based spectral feature.

Note that each spoken language is governed by a set of phonological constraints, which are encoded in a phonetic lexicon. This observation inspires a phonotactic solution to language recognition. Although common sounds are shared considerably across spoken languages, the statistics of the sounds, such as phone n -grams, differ very much from one language to another. The phonotactic statistics may be derived from a single universal phone recognizer [5]-[10], or a set of parallel phone recognizers (PPR) [1][2][11][12]. The phone recognizer is often referred to as a sound tokenizer because the definition of the phone has been generalized [9][10] in some applications. Studies have shown that PPR is effective, which benefits from its multiple phonetic token sequences. In human perceptual experiments, listeners with a multilingual background often perform better than monolingual listeners in identifying unfamiliar languages. Similarly, we expect the phonotactic statistics from different token sequences to provide complementary information for language recognition.

The PPR front-end has been investigated in combination with both the phone n -gram language models (LM) [2] and the vector space modeling (VSM) backend [13]. The LM backend evaluates each token sequence using multiple language models, each of which describes a token sequence from the perspective of a target language. With VSM backend, the n -gram statistics from each token sequence form a high-dimensional feature vector, also known as a *bag-of-sounds* (BOS) vector. Then, a composite vector is constructed by stacking multiple *bag-of-sounds* vectors derived from multiple token sequences.

Both the spectral polynomial expansion and *bag-of-sounds* approach are considered vector space modeling techniques because they represent a spoken utterance as a high dimensional vector for pattern classification. In many cases, it

is desirable to reduce the dimensionality of the aforementioned high-dimensional vectors to a manageable size so that probabilistic models, such as Gaussian mixture model (GMM) and artificial neural network (ANN), can be easily employed for pattern classification. The challenge is to reduce the vector size dramatically while maintaining its discriminative ability at the same time. Many dimensionality reduction approaches, such as truncated singular value decomposition (SVD) [14] and principal component analysis (PCA) [15], have been studied. However, these approaches are not directly aimed at enhancing language discriminative characteristics.

Language recognition is a typical multi-class classification problem. The distributed output coding method [16] solves a multi-class problem by reducing it to multiple binary classification problems. A set of binary classifiers, each trained to distinguish between two disjoint subsets of the labeled data, is constructed to create a distributed output representation for a test instance, and referred to as output code. Such a set of binary classifiers can be seen as a classifier ensemble. Each class is assigned a unique binary vector, which represents the output code for instances in the class, and referred to as the *codeword*. In this way, multi-class data are represented by multiple *codewords*. For a test instance, the classification can be achieved by finding the nearest *codeword* in Hamming distance. It was shown that the distributed output representation improved the generalization capability on a wide range of multi-class learning tasks [16]. An improved output coding method with continuous relaxation on the output scores was also proposed to improve the classification performance [17]. The continuous relaxation also allows us to represent a class with a statistical model instead of a binary *codeword*.

Ensemble classifiers are shown to be particularly useful if each of its component classifiers is well trained in a different region of the feature space. They are also called *mixture-of-expert* models, modular classifiers, or occasionally pooled classifiers [18]. The basic idea underlying ensemble classifiers is the resampling technique, that suggests building multiple component classifiers, each of which is trained on a different sampling of a training set.

In this paper, we adopt binary SVM classifier as the component classifier. We would like to investigate the use of distributed output coding technique as the ensemble classifier design strategy for spoken language recognition. We are interested in the language discriminative ability presented by different classifier design strategies. Hereafter, distributed output coding is also referred to as output coding for short.

Specifically, we will apply the output coding technique to dimensionality reduction of both the *bag-of-sounds* vectors and the spectral polynomial expansion vectors, and study: (i) the SVM partitioning strategy that describes the properties of spoken languages and generates output codes; (ii) the effects of SVM component classifier selection on spoken language recognition performance; (iii) the comparison among different component classifier selection strategies, such as one-versus-rest, one-versus-one and discriminative ability ranking. The language recognition experiments are conducted on 1996, 2003 and 2005 NIST Language Recognition Evaluation

(LRE) corpora. The systems in this paper contributed to the IIR¹'s submission to the 2005 NIST LRE.

This paper is organized as follows. In Section II, we describe the system architecture with PPR & SDC front-end and VSM backend, which uses an ensemble of binary SVM component classifiers to derive output codes. In Section III, we study different classifier selection strategies and their effectiveness. In Section IV, we report experimental results. Finally we conclude in Section V.

II. LANGUAGE RECOGNITION SYSTEM

A. System Architecture

We propose a language recognition system architecture that comprises a feature extraction front-end, a vectorization back-end and a language recognition decision classifier. Fig. 1 illustrates a PPR-VSM system with a set of parallel phone recognizers (PPR) in the voice tokenization front-end. It employs a vector space modeling (VSM) backend that encodes the composite vectors into output codes for dimensionality reduction. Fig. 2 illustrates a SDC-VSM system with shifted-delta-cepstral (SDC) feature extraction front-end. It employs a VSM backend, which first represents a speech utterance in a single polynomial expansion vector [3] and then encodes the feature vectors into output codes for language detection.

We begin by describing the PPR-VSM architecture. Suppose that we have F phone recognizers with a phone inventory of $v = \{v_1, \dots, v_f, \dots, v_F\}$ and the number of phones in v_f is n_f . An utterance is decoded by these phone recognizers into F independent sequences of phone tokens. Each of these token sequences can be expressed by a high dimensional phonotactic feature vector with the n -gram counts. The dimension of the feature vector is equal to the total number of n -gram patterns needed to highlight the overall behavior of the utterance. If unigram and bigram are the only concerns, we will have a vector of $n_f + n_f^2$ phonotactic features, denoted as V_f , to represent the utterance by the f -th phone recognizer.

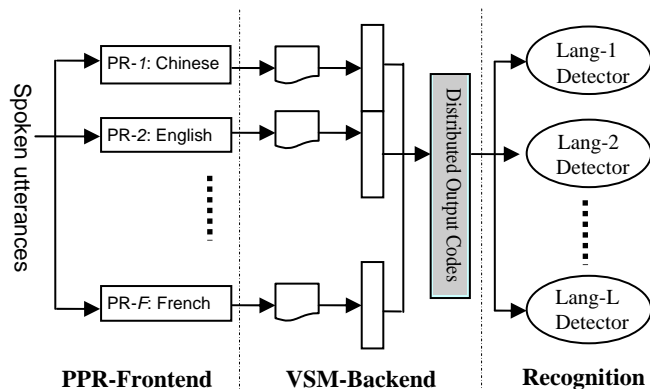


Fig 1. Block diagram of PPR-VSM language recognition system.

¹ Institute for Infocomm Research, Singapore

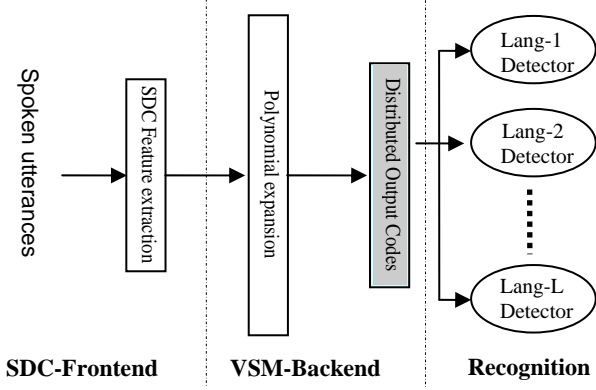


Fig 2. Block diagram of SDC-VSM language recognition system.

One of the advantages of VSM [13] is that it allows the discriminative training over high dimensional feature vectors [19]. After a spoken utterance is vectorized, language recognition can be cast as a vector-based classification problem. To benefit from its the distribution-free property, we adopt the support vector machine (SVM) [20] to construct the VSM backend. As shown in Fig. 1, we concatenate all the F phonotactic feature vectors, into a large composite *bag-of-sounds* vector

$$V = [V_1^t, \dots, V_f^t, \dots, V_F^t]^t, \quad (1)$$

with a dimension of

$$K = \sum_f (n_f + n_f^2), \quad (2)$$

if only unigram and bigram features are included. The SVM is then trained on these composite *bag-of-sounds* vectors. By using a single composite feature vector, we effectively fuse phonotactic features resulting from multiple phone recognizers and make classification decision using a single SVM decision hyperplane. The PPR-VSM architecture in this paper is similar to that in [13] except for the distributed output representation, which is the focus of this paper.

The SDC-VSM system is motivated by the prior work by Campbell et al. [3], where frame-based shifted-delta-cepstral (SDC) features are expanded to a high dimensional space. A generalized linear discriminant sequence (GLDS) kernel is used to measure similarity between spoken utterances. The SDC-VSM architecture is similar to the GLDS approach in [3] except for the distributed output representation. Our idea is to reduce the high dimensional feature vector into a lower dimensional space which is discriminatively-motivated. SDC-VSM therefore produces output codes in the similar way as PPR-VSM does. We can build language detectors for both systems in the same way.

B. Dimensionality Reduction of Feature Vectors

Ideally we would like to reduce the dimensionality of the aforementioned high-dimensional feature vectors to less than a few hundreds while maintaining their discriminative characteristics at the same time. Let's first discuss some traditional techniques to do so. Suppose that we are given a

training set of D such vectors, each having K attributes. Let A denote the corresponding $K \times D$ vector-attribute matrix. SVD [14] can effectively reduce the dimensionality by finding the closest rank- R approximation to A in the Frobenius norm.

Given a high-dimensional vector \mathbf{x} , we can use the following transformation to construct a new vector \mathbf{y} in the R -dimensional space,

$$\mathbf{y} = (US^{-1})^t \mathbf{x} = \mathbf{W}_{SVD}^t \mathbf{x}, \quad (3)$$

where U is a $K \times R$ left singular matrix with rows u_k , $1 \leq k \leq K$, S is a $R \times R$ diagonal matrix of singular values, with $s_1 \geq s_2 \geq \dots \geq s_R > 0$, and $R \ll K$ [14].

This approach is known as latent semantic indexing (LSI) [14], which is based on the assumption that there is some underlying latent semantic structure in the vector-attribute matrix that is corrupted by the wide variety of attributes used in the high dimensional vectors. The LSI dimensionality reduction allows us to retain semantic structure in the low dimensional space.

Similar to SVD method, PCA [15] finds R -dimensional subspace that best represents the full data with respect to a minimum squared error. Although SVD and PCA methods find subspaces that are useful for representing the original high-dimensional vector space, there is no reason to assume that the resulting projections will be useful for discrimination between data in different classes [18].

C. Linear Discriminant

Another established algorithm for dimensionality reduction is linear discriminant analysis (LDA). It estimates a linear projection matrix that transforms the original feature vectors onto a subspace by maximizing the class discrimination [18], [21].

We project a vector \mathbf{x} from a K -dimensional space to a Q -dimensional space by Q discriminant functions [18] in the directions that are efficient for class discrimination.

$$y_i = \mathbf{w}_i^t \mathbf{x} \quad i = 1, \dots, Q. \quad (4)$$

If we consider y_i as components of a vector \mathbf{y} and the weight vectors \mathbf{w}_i as the columns of a $K \times Q$ matrix \mathbf{W}_{LDA} , then the projection can be written as a single matrix equation

$$\mathbf{y} = \mathbf{W}_{LDA}^t \mathbf{x}. \quad (5)$$

\mathbf{W}_{LDA} is similar to \mathbf{W}_{SVD} in (3) except that it is achieved by solving Q Fisher linear discriminants $J(\mathbf{w}_i)$ [18].

In LDA, the problem of finding a linear discriminant function is formulated as the problem of minimizing a criterion function. The common criterion function for classification is the classification error, the average loss incurred in classifying the set of training samples. The minimum squared-error (MSE) is one of the solutions to the problem, by solving a set of linear equations. The procedure involves all the training samples and assumes that within-cluster scatter matrix is nonsingular.

As a variant of LDA, support vector machine (SVM) provides an alternative solution for discriminative functions.

First, MSE approaches devote equal attention to all data items, whereas we are interested in making the correct classification decision. In contrast, an SVM system places a hyperplane in a high dimensional space so that the hyperplane has the maximum margin. Therefore, SVM is more interested in the decision boundaries than the sample distributions themselves. Second, SVM relies on preprocessing of the samples to represent patterns in dimension, typically much higher than that in the original feature space. The idea is that, with an appropriate nonlinear mapping to a sufficiently high dimension, samples from two classes can always be separated by a hyperplane. SVM has shown to be an effective machine learning method for pattern classification in high dimensional spaces. Many attempts, for instance, in text categorization [22] have been rather successful.

SVM has been shown to be effective in separating high dimensional vectors in two-class problems, in which SVM effectively projects the high dimensional vector \mathbf{x} into a scalar value $y = f(\mathbf{x})$. Suppose that we have a collection of N support vectors, an SVM is constructed from the sum of kernel functions $k(\cdot, \cdot)$:

$$f(\mathbf{x}) = \sum_{n=1}^N \alpha_n t_n k(\mathbf{x}, \mathbf{x}_n) + d \quad (6)$$

where t_n are the ideal outputs, $\sum_{n=1}^N \alpha_n t_n = 0$ and $\alpha_n > 0$. For a linear kernel $k(\mathbf{x}, \mathbf{x}_n) = \mathbf{x}^t \mathbf{x}_n$, (6) can be easily rewritten as

$$y = f(\mathbf{x}) = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n^t \mathbf{x} + d = \mathbf{w}_c^t \bar{\mathbf{x}} \quad (7)$$

where $\mathbf{w}_c = [\sum_{n=1}^N \alpha_n t_n \mathbf{x}_n, d]$ and $\bar{\mathbf{x}} = [\mathbf{x}, 1]$. Interestingly, once we have trained an SVM, we can collapse all the support vectors \mathbf{x}_n into a single weight vector \mathbf{w}_c similar to the weight vectors \mathbf{w}_i in (4). If we see \mathbf{w}_c as the columns of a $K \times Q$ matrix \mathbf{W}_{SVM} , then the projection can be written as

$$\mathbf{y} = \mathbf{W}_{SVM}^t \bar{\mathbf{x}} \quad (8)$$

\mathbf{W}_{SVM} is similar to \mathbf{W}_{LDA} or \mathbf{W}_{SVD} except that it is constructed by Q SVM decision hyperplanes. Each SVM is represented by a weight vector \mathbf{w}_c . In this way, the Q SVM outputs form a dimension-reduced space, which characterizes the language space of interest with Q discriminative hyperplanes.

D. Output Codes as Discriminative Vectors

The aforementioned dimensionality reduction can be considered as a linear transformation, in which a multidimensional feature vector \mathbf{x} is projected to a 1-dimensional space $f(\mathbf{x}) = \mathbf{w}^t \mathbf{x}$. From a two-class classification point of view, the linear function represents a direction in which one language is separated from another with minimum sample risk. If we employ a linear SVM as the linear discriminant function for each of the component classifiers, then the outputs $f(\mathbf{x})$ from a collection of Q such SVMs form a vector of signed distance. In this way, a high dimensional feature vector \mathbf{x} can be reduced to one of much

lower dimension $\{f^1(\mathbf{x}), f^2(\mathbf{x}), \dots, f^Q(\mathbf{x})\}$.

The collection of SVMs is expected to produce effective output codes as discriminative vectors. There are many different ways to do so as will be discussed in Section III. Two classical space partitioning methods were studied in pattern recognition literature [18].

1) *One-Versus-Rest (OVR) SVMs*: The high-dimensional vectors in one target language are considered as the positive set and those from all other languages as the negative set. We can arrive at the vectors with the dimension,

$$Q = L, \quad (9)$$

where L is the number of target languages;

2) *One-Versus-One (OVO) SVMs*: We build a SVM for each language pair [13]. As we have $L \times (L-1)/2$ distinct language pairs, therefore, this strategy results in vectors with the dimensions,

$$Q = L \times (L-1)/2. \quad (10)$$

In this way, not only do we effectively reduce the dimensionality from a large K to a small Q , but also represent the features in a discriminative space of competing language partitions. We also refer to the Q -dimension output code as a discriminative vector.

E. Language Recognition as a Hypothesis Test

We formulate the language recognition as a hypothesis test. For each target language, we build a language detector which consists of two Gaussian mixture models (GMM) $\{m^+, m^-\}$. m^+ is trained on the output codes of target language, called positive model, while m^- is trained on those of its competing languages, called negative model. We define the confidence of a test sample O belonging to language m^+ as the posterior odds in a hypothesis test under the Bayesian interpretation. We have H_0 , which hypothesizes that O is language m^+ , and H_1 , which hypothesizes otherwise. The posterior odds is approximated by the likelihood ratio:

$$\lambda = \log \left(\frac{p(O|m^+)}{p(O|m^-)} \right) \quad (11)$$

The higher the λ is, the more probable H_0 overtakes H_1 . The likelihood ratio is used for the final language recognition decision.

III. CLASSIFIER DESIGN STRATEGY

A. Output Coding with Ensemble of Binary Classifiers

Output coding [16] is a general method for solving multi-class problems by decomposing them to multiple binary classification problems with an ensemble of binary classifiers. Typically, the output from each binary classifier is defined as discrete codes of 0 and 1, contributing one bit in the bit-vector representation of output collection. Using binary output coding, a class is encoded by a centroid code, otherwise known as *codeword*.

Using an SVM output as the binary output coding bit b , we have $b=1$ if $f(\mathbf{x}) > 0$, and $b=0$ otherwise. Let's describe the technique of output coding through a simple example: the task of classifying a vector into the $L=4$ classes. Each language is assigned with a Q -bit vector with $Q > \log_2 L$, for example, Chinese - 1000, English - 0100, French - 0010 and Korean - 0001.

Each individual SVM is seen as a component classifier. A bit in the bit-vector represents the output of an individual SVM that describes one discriminative attribute about the languages of interest. The bit-vector is an output collection of Q individual SVMs. An ensemble classifier makes collective decision based on a number of individual decisions represented in the bit-vector. The greater the distance between the *codewords* is, the better discriminability that a multi-class classifier has. In general, a larger code size leads to a better performance.

Some recent work improves the performance of output coding by relaxing the output codes from binary coding to continuous coding [17]. By adopting continuous output code, a class can be modeled by a Gaussian mixture model (GMM), as formulated in Section II-E, which allows the collective decision to be made using pooled results from component classifiers. In practice, (8) is implemented like this. First, we train a set of Q independent SVM classifiers $\{f^1(\cdot), f^2(\cdot), \dots, f^Q(\cdot)\}$, then we represent each of the training or test utterance \mathbf{x} as a vector of Q real-valued SVM outputs $\mathbf{y} = \{f^1(\mathbf{x}), f^2(\mathbf{x}), \dots, f^Q(\mathbf{x})\}$, known as continuous output code. In this paper, we adopt continuous output coding for its improved performance in a probabilistic formulation. For detailed discussions on continuous relaxation of output codes, readers are referred to [17].

B. OVR and OVO SVM Classifiers

Now the question is how to construct the Q SVMs. One intuitive idea is to construct the Q independent SVMs, each deciding a target language vs. the rest. In doing so, we denote the samples labeled with the target category as the positive samples, and the rest as the negative samples. The code size of the so defined output code vector equals the number of classes L as in the one-versus-rest (OVR) partitioning strategy. In binary coding, for an ensemble of binary classifiers, we can choose the code size Q in the range $[\log_2 L, (2^L - 1)/2]$ [16]. A code size of $Q < \log_2 L$ cannot even assign a distinct bit-vector to each label. At the other extreme, a code of size $Q > (2^L - 1)/2$ must contain duplicate columns, which means two individual classifiers learning the same task. Note that, with the same code size, continuous output coding consistently outperforms binary coding in multi-class classification tasks.

Another idea, besides one-versus-rest, is to construct the Q independent SVMs by having pairwise SVM classifiers as the component classifiers. Suppose that we have L language categories, we build $L \times (L-1)/2$ such classifiers, with each discriminating a language pair, resulting in an output code of $Q = L \times (L-1)/2$ dimensions as in the one-versus-one (OVO)

partitioning strategy. To understand why one might expect the pairwise output code to work well, we can consider the problem of learning to classify three languages, Chinese, Japanese and English. Each language pair has its distinctive distinguishing characteristics shown in Table I:

Table I
Pairwise SVM example of three target languages

	Tonal	Syllabic
Chinese	Yes	Yes
Japanese	No	Yes
English	No	No

If we build $Q = 3 \times (3-1)/2 = 3$ component SVMs, each discriminating a language pair, one can expect Chinese-Japanese classifier to learn the strong association between tonal characteristics and Chinese, in order to differentiate it from Japanese. Likewise, Japanese-English classifier will learn the strong association between a syllabic structure and Japanese. During the training of a Chinese-Japanese SVM classifier, we take Chinese data as the positive samples and Japanese data as the negative samples, leaving the English data out. The same principle applies to the training of all SVMs. To the Chinese-Japanese SVM, when a Chinese feature vector \mathbf{x} is presented, the SVM is trained to derive a positive value $f(\mathbf{x}) > 0$. In contrast, the SVM will respond to a Japanese vector with a negative value $f(\mathbf{x}) < 0$. The SVM output value for an English vector is to be decided by the SVM depending on how the SVM sees English from the point of view of the Chinese-Japanese decision hyperplane.

C. Discriminative Ability Ranking (DAR)

Actually, a component classifier can be obtained by placing an SVM hyperplane arbitrarily in the space of training database of L target languages. One-versus-rest is just one way of placing the SVM hyperplane. We would like to study how to conduct the component classifier selection to select those SVM classifiers that represent the highest discriminative characteristics between languages.

It is logical, in practice, to place a hyperplane that separate languages, with the training data of one or more languages on one side as the positive set and the rest on another side as the negative set. In this way, for L languages, we have $2^{L-1} - 1$ different ways of hyperplane placement, hereafter referred to as partition, each of which represents certain discriminative characteristics between languages. One-versus-rest partitioning strategy is just one of them. It is desirable for a small but effective subset of those hyperplanes to form an ensemble of classifiers to generate output codes that represent the high dimensional feature vector.

To study the strategies of component classifier selection, we construct an SVM classifier on each hyperplane placement and exhaust all the $2^{L-1} - 1$ possibilities. We design a discriminative ability indicator, based on the margin width of the resulting hyperplane, to measure the discriminative ability of SVM classifiers among the target languages. It is known in SVM training that the non-linearly separable situations are

typically solved by imposing a penalty to misclassified vectors, establishing a trade-off between margin width and the error rate in training data. Two factors constitute the discriminative ability indicator. One is the margin width of the resulting hyperplane which is inversely proportional to $\|a\|$ where

$$a = \sum_{n=1}^N \alpha_n t_n \mathbf{x}_n \quad (12)$$

is the weight vector seen as the normal to the hyperplane. α_n and t_n have been defined in (6). Another is the accuracy performance (Acc) that a candidate SVM classifier offers to the training data. We use the product of these two factors as the discriminative ability indicator:

$$DA = \frac{Acc}{\|a\|} \quad (13)$$

We use DA to rank all the candidate SVM classifiers, and expect to shortlist the top Q SVM classifiers that are of the highest DA values and to use their output codes to form discriminative vectors. We found in the experiments that the top $Q=L$ SVM classifiers perfectly coincide with those one-versus-rest (OVR) hyperplanes, that is, having one language in the positive set and the rest in the negative set. This finding further validates the effectiveness of OVR strategy.

IV. EXPERIMENTS

We followed the experiment setup in the NIST Language Recognition Evaluation (LRE) tasks². The tasks were intended to establish a baseline of performance for language recognition of conversational telephone speech. The evaluation was carried out on recorded telephony speech in 12 languages, Arabic, English, Farsi, French, German, Hindi, Japanese, Korean, Mandarin, Spanish, Tamil, and Vietnamese, for the 1996, 2003 NIST LRE tasks, and in 7 languages, English, Hindi, Japanese, Korean, Mandarin, Spanish, and Tamil for the 2005 NIST LRE task. The 1996 NIST LRE evaluation data consist of 1,492, 1,501, and 1,503 sessions of 30, 10 and 3 seconds respectively. The 2003 NIST LRE evaluation data consist of 1,280 sessions for each of the durations, while the 2005 data consist of 3,662 sessions per duration. Without loss of generality, we use 30-second sessions in the 1996, 2003 and 2005³ LRE evaluation data for the following experiments. We will report language verification results in terms of equal error rate (EER). EER is the point where the probabilities of false acceptances and false rejections are equal. We suppose that the priors for target and non-target languages are equal.

In the experiments, we are interested in the effects of distributed output coding approach in spoken language recognition by using both parallel phone recognizers (PPR) and shifted delta cepstral (SDC) front-ends. We would like to compare different component classifier selection strategies such as one-versus-rest, one-versus-one, and discriminative

ability ranking. Each component classifier selection results in a set of partitions that defines the output code.

A. PPR-based and Spectral-based Front-ends

We employ two independent PPRs (see Fig. 1) using two different phone recognizer sets to derive phonotactic features. The objective of using two sets of PPRs is to study whether a selected ensemble classifier based on one PPR can be readily used for another PPR, such that exhaustive computation for component classifier selection can be avoided for new PPR. We also employ a spectral-based front-end with SDC coefficients [4] to derive acoustic features (see Fig. 2).

The first PPR, referred to as PPR1, includes phone recognizers of seven languages, namely English, Korean, Mandarin, Japanese, Hindi, Spanish and German. A total of 300 phones, 44 for English, 37 for Korean, 43 for Mandarin, 32 for Japanese, 56 for Hindi, 36 for Spanish and 52 for German, were adopted in our experiments to construct the parallel phone recognizers for creating phone token sequences. English phone models were trained on IIR-LID⁴ database. Korean phone models were trained on LDC Korean corpus (LDC2003S03). Mandarin phonemes were trained on MAT corpus [23]. Other phone models were trained on OGI-TS⁵ (Multi-language Telephone Speech) database. 39-dimensional features consisting of 12 MFCCs and normalized energy, plus their first and second order time derivatives, were extracted for each frame. Utterance-based cepstral mean subtraction was applied to the features to remove channel distortion. HTK toolkit was used to train a 3-state HMM with mixture Gaussians for each of 300 phones. Phone recognition was carried out using a fully connected null-grammar network of phones without language models.

The second PPR, referred to as PPR2, consists of three phone recognizers based on long temporal context developed by Brno University of Technology⁶ (BUT). The three phone recognizers of Czech, Hungarian and Russian, were trained on the SpeechDat-E⁷ speech databases. A total of 158 phones, 45 for Czech, 61 for Hungarian and 52 for Russian, are adopted to construct the parallel phone recognizers for creating phone token sequences in PPR front-end.

We also derived shifted delta cepstral (SDC) coefficients by stacking delta cepstral parameters computed across multiple speech frames to capture temporal spectral information. As formulated in [4], SDC computations are controlled by four parameters (N, d, P, k). In our experiments, we used (7, 1, 3, 7) SDC parameter configuration, resulting 49-dimensional feature vector, as in [1].

B. Training and Development Corpus

We used the 12-language LDC CallFriend⁸ speech database as the training and development corpus for language recognition experiments. The 12 languages are the same as

⁴ Language identification corpus in Institute for Infocomm Research

⁵ <http://cslu.cse.ogi.edu/corpora/corpCurrent.html>

⁶ http://www.fit.vutbr.cz/research/groups/speech/index_e.php?id=phnrec

⁷ <http://www.fee.vutbr.cz/speechdat-e/>

⁸ See <http://www ldc.upenn.edu/>. The overlap between the *CallFriend* database and the 1996 LRE data were not used as the training and development data, as suggested in <http://www.nist.gov/speech/tests/index.htm> for 2003 LRE.

² <http://www.nist.gov/speech/tests/index.htm>

³ The evaluation sessions in India English are removed due to insufficient training and development data.

those in the 1996 and 2003 NIST LRE tasks. In the 12-language CallFriend database, there are two accents for each of the three languages, English, Mandarin and Spanish. We treated these accents as different languages. As a result, we have a total of 15 target languages. Each of the 15 languages in CallFriend database has three sets of speech data, training set, development set and evaluation set. Each of the three data sets consists of 20 telephone conversations with each lasting approximately 30 minutes.

Each conversation was segmented into small speech chunks using voice activity detection (VAD) algorithm. A roughly 30-second long session consists of multiple speech chunks. The sessions extracted from the CallFriend training sets were used to create high dimensional *bag-of-sounds* vectors and polynomial expansion of shifted delta cepstral (SDC) features. For the 7-language PPR1, we derived a *bag-of-sounds* vector with 13,314 ($= 44^2 + 37^2 + 43^2 + 32^2 + 56^2 + 36^2 + 52^2 + 44 + 37 + 43 + 32 + 56 + 36 + 52$) elements for each session, using only the unigram and bigram statistics. For the 3-language PPR2, a *bag-of-sounds* vector with 8,608 ($= 45^2 + 61^2 + 52^2 + 45 + 61 + 52$) elements was derived for each session. In the spectral-based front-end, the 49-dimension SDC coefficients were expanded to a high dimensional feature space by calculating all the monomials up to order 3, resulting in $\binom{(49+1)+3-1}{3} = 22,100$ elements for each session. The aforementioned high dimensional feature vectors generated from the front-ends were used to train the SVMs⁹. Each of these SVMs generates one output code value, thus one element in the output code.

The CallFriend training sets of 15 languages were used in the component classifier selection process. A number of SVMs that are of the highest discriminative ability were selected according to (13) to construct an ensemble of SVM classifiers. The CallFriend development sets were used for GMM modeling of the output codes. Each speech session in the CallFriend development data is evaluated by the selected ensemble of SVM classifiers to generate an output code. As a result, a bunch of output codes were generated for each target language. To conduct the language recognition as a hypothesis test, we constructed two GMM models, one was trained by the output codes of the target language itself, another was trained by those of its competing languages. The likelihood ratio shown in (11) was used to make the final decision.

C. Discriminative Ability Ranking

With the high dimensional feature vectors from CallFriend training data, we now look into the choice of component classifiers for dimensionality reduction. As mentioned in section III-C, we have $2^{15-1} - 1 = 16,383$ different ways of hyperplane placement for 15 languages in the training data. We design a SVM for each hyperplane. From these 16,383 SVMs, we will select those with the highest discriminative ability to generate output codes. To exhaust all the possible hyperplanes, much computing is involved. We select the output codes using PPR1, PPR2 and SDC front-ends on the

CallFriend training data and examine the performances of the resulting output codes.

All the 16,383 SVMs were ranked according to the discriminative ability indicator shown in (13). We found that, with all the three front-ends, the top 15 ($L=15$) output codes coincide with the 15 one-versus-rest SVMs. The next high ranked SVMs are almost always those in two-versus-rest setting, with two languages on one side and the rest on another side, followed by those in three-versus-rest setting and so on.

To examine the consistency of the resulting SVMs from different front-ends, we compare the top M SVMs generated from the three front-ends. Fig. 3 illustrates the similarity between the top M SVMs selected from the PPR1 and PPR2 front-ends. The similarity is defined as the percentage of overlap between the PPR1 and PPR2 SVM shortlists. There are three peaks in the curve. The first peak presents the SVM shortlists of all the one-versus-rest SVMs. The second peak presents the SVM shortlists including both the one-versus-rest and two-versus-rest SVMs. The third peak presents the SVM shortlists by further adding the three-versus-rest SVMs.

The curve in Fig. 3 shows that the top SVM classifiers from different front-ends, PPR1 and PPR2, demonstrate consistency in describing the language characteristics among the target languages. Therefore, we can select an ensemble of SVM classifiers from one front-end, and apply them to the other front-ends directly. Fig. 4 shows the similarity between the top SVMs selected from PPR1 front-end and SDC front-end. It reveals the same trend as that in Fig. 3.

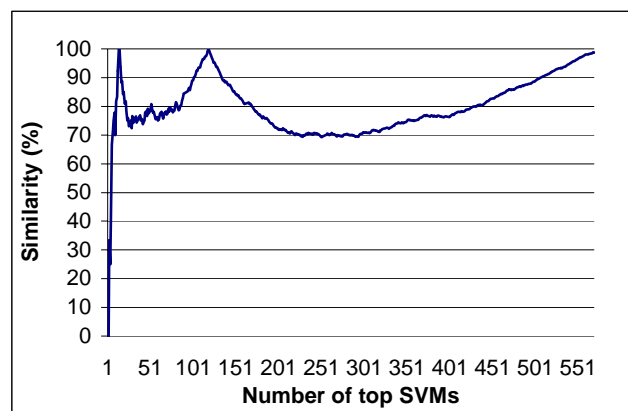


Fig. 3. The overlap of top SVM component classifiers selected from PPR1 and PPR2 front-ends.

⁹ <http://svmlight.joachims.org/>

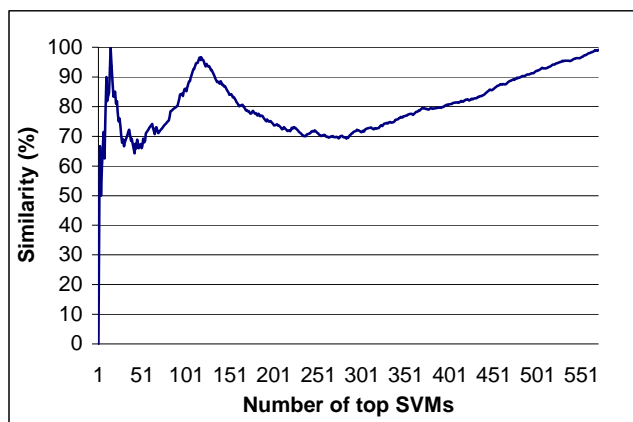


Fig. 4. The overlap of top SVM component classifiers selected from PPR1 and SDC front-ends.

D. Language Recognition Experiments

We devise three experiments to validate the effectiveness of the component classifier selection strategies. Summarizing the theory in Section II and III, we build the language recognition systems in three steps:

- Step-1: Use CallFriend training data to select the component classifiers;
- Step-2: Use CallFriend development data to derive the output codes using the selected component classifiers from Step 1;
- Step-3: Use the resulting output codes to build the GMM-based language detectors;

We use the Expectation-Maximization algorithm to train the GMM models. Each language detector consists of two GMM models $\{m^+, m^-\}$. m^+ has 4 Gaussian mixture components while m^- has 32 Gaussian mixture components.

In the first experiment, we are interested in examining the effects of different component classifier selection strategies using the PPR1 front-end. We carried out language recognition on the 1996, 2003, and 2005 NIST LRE speech corpora using the PPR-VSM language recognition system (see Fig. 1). Fig. 5 shows the EER performance as a function of the output code size.

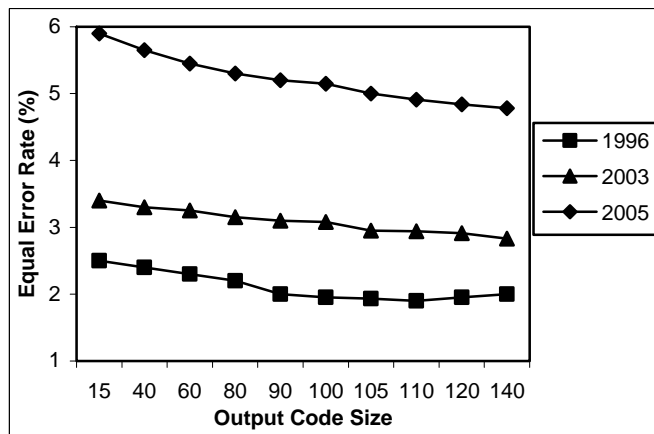


Fig. 5. Performance of PPR-VSM system on 1996, 2003, and 2005 NIST LRE corpora using PPR1-selected SVM component classifiers and PPR1-derived output codes.

The three curves start with top 15 SVM component classifiers that coincide with the one-versus-rest partitions. As expected, the performance of language recognition is improved as the output code size increases. We expect the trend to continue so long as sufficient training data for GMM modeling are available. Of course, larger code size also incurs a higher computational cost.

Now we compare the three strategies: one-versus-rest (OVR) SVMs as in (9), one-versus-one (OVO) SVMs as in (10), and discriminative ability ranking (DAR) as in (13). As discussed, the top 15 selected SVM component classifiers are common between OVR and DAR. To establish a fair comparison, we compare the component classifier selection strategies and the truncated singular value decomposition (SVD) dimensionality reduction algorithm¹⁰ and use the same code size of 105 for OVO and DAR. The results in Table II show that all the three component classifier selection approaches consistently outperform the SVD dimensionality reduction algorithm.

Table II. Performance (EER%) comparison of different dimensionality reduction techniques using PPR-VSM language recognition system

EER (%)	SVD 105	OVR $Q_1 = 15$	OVO $Q_2 = 105$	DAR $Q_3 = 105$
1996	3.63	2.50	2.75	1.93
2003	4.83	3.40	4.02	2.95
2005	7.35	5.90	5.78	4.96

Note that one can use different front-ends in Step-1 and Step-2. In the second experiment, we would like to study selecting component classifiers with one front-end then applying them in another front-end. We used PPR1 and PPR2 to derive two sets of component classifiers in Step-1, and apply both of them in PPR2 front-end to derive output codes in Step-2. As summarized in Fig. 6, it is shown that the two ensemble classifiers provide comparable results. Although PPR1 and PPR2 front-ends are constructed using phones from two non-overlapping language sets, they partition the language space in a similar way. This suggests that we can select ensemble classifiers using one front-end and apply to another when deriving output codes. In this way, exhaustive computation for the component classifier selection in a new front-end may be avoided.

¹⁰ <http://tedlab.mit.edu/~dr/SVDLIBC/>

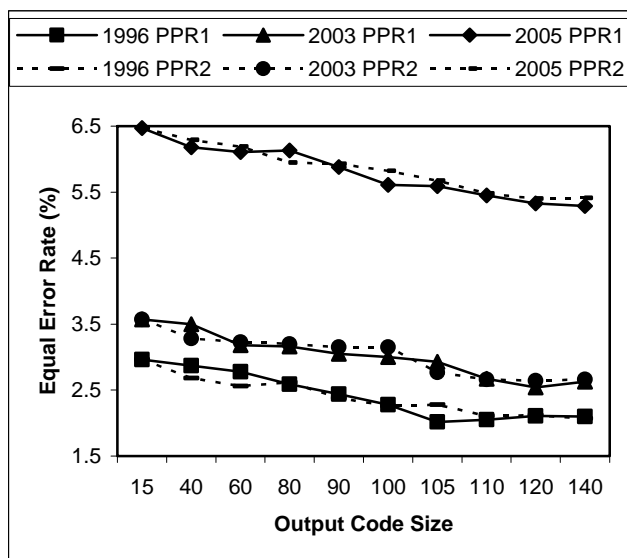


Fig. 6. Performance of PPR-VSM system (see Fig. 1) on 1996, 2003, and 2005 NIST LRE corpora. In discriminative ability ranking, SVM classifiers are selected using PPR1 and PPR2 front-ends while output codes are always derived through PPR2 front-end.

After having studied the component classifier selection strategies, in the third experiment, we would like to look into the SDC features. For ease of comparison, we continued to use the ensemble classifier selected using PPR1 front-end. We derived output codes from SDC front-end for the CallFriend development data. As summarized in Fig. 7, it is shown that the equal error rate (EER) decreases as a function of the output code size, in a similar trend as that in Fig. 5. Compared with the first experiment, SDC features are less effective than PPR1 features.

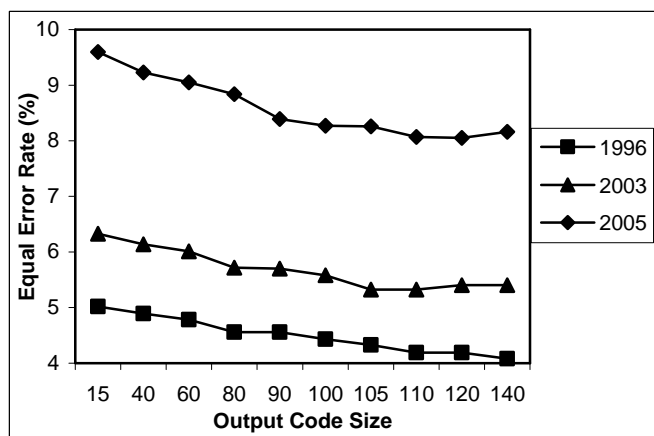


Fig. 7. Performance of SDC-VSM system (see Fig. 2) on 1996, 2003, and 2005 NIST LRE corpora. SVM classifiers are selected using PPR1.

In summary, the three experiments show that: (i) By carefully selecting ensemble classifiers based on the three component classifier selection strategies, we can find a set of SVM classifiers for output codes that work well for the dimensionality reduction in language recognition; (ii) Although the selection of SVM ensemble classifier based on the discriminative ability ranking (DAR) strategy requires exhaustive computation and is time-consuming, it is possible

to find a set of component classifiers based on a single front-end, and readily apply them to other front-ends.

E. Overall System with Score Fusion

Besides exploring the informative language cues from multiple sources, another important issue in spoken language recognition is how to efficiently fuse these cues to achieve better performance. We adopted the SVM combiner of three SVMs with polynomial kernels of order 1, 2, and 3 [24], shown in Fig. 8. The input scores are the likelihood ratios in (11) from multiple language detectors. The final score for language recognition is produced by averaging the output scores of three SVM classifiers.

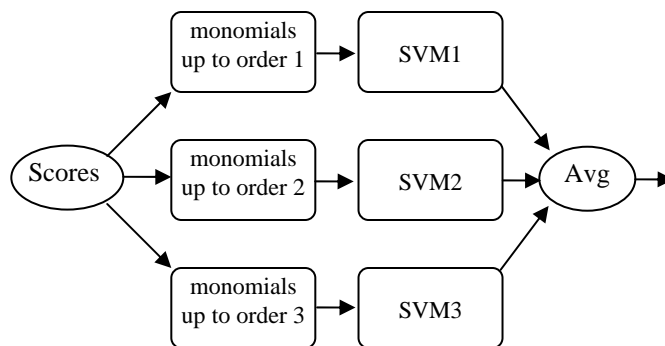


Fig. 8. SVM combiner for score fusion of multiple sources.

Table III reports the language recognition results on the 2003 and 2005 NIST LRE corpora. The 1996 NIST LRE data were used as the development data for the SVM combiner.

Table III. Performance (EER%) of score fusion across PPR-VSM and SDC-VSM systems on 2003 and 2005 NIST LRE corpora

Score Fusion	NIST 2003	NIST 2005
PPR1	2.95	4.96
+ PPR2	1.51	3.70
+ SDC	1.38	3.20

In the score fusion experiment, we used 105 PPR1-selected component classifiers based on DAR strategy to derive output codes through PPR1, PPR2 and SDC front-ends. We incrementally fused scores from three language detectors. EER is reduced by 48.8% on the 2003 NIST LRE corpus by fusing PPR2 with PPR1 score. By further adding SDC, we achieve the best performance of EER=1.38%. It clearly shows that PPR1, PPR2 and SDC features provide complementary information to each other. The score fusion consistently improves overall performance.

We also report our overall result in comparison with other state-of-the-art language recognition systems in the literature. Table IV shows the performance of three other systems on the 2003 NIST LRE corpus. It is found that our result represents one of the best performances on the same task.

Table IV. Performance (EER%) benchmark on 30-second 2003 NIST LRE corpus

Systems	EER %
MITLL [1]	2.8

LIMS1 [11]	2.7
BUT [25]	0.8
PPR1+PPR2+SDC	1.38

V. DISCUSSION

In this paper, we proposed a new approach to ensemble classifier design for vector space spoken language recognition. We have studied different ensemble classifier design strategies motivated by the distributed output coding technique.

We would like to highlight that the output coding strategy is different from the resampling technique – another ensemble classifier designing principle – such as bagging and boosting. The basic insight underlying resampling techniques is that multiple data sets are drawn from a given data set to enable the estimation of values and ranges of arbitrary statistics [18]. With resampling techniques, each data subset is used to construct a component classifier. Therefore, the resulting component classifiers are of the same general form each other. They are trained to achieve the same decision task as that of the classifier ensemble. However, the output coding strategy solves a multi-class classification problem by reducing it into multiple binary classification problems, each responsible for removing some uncertainty about the correct class of the input [26]. Therefore, each binary classification problem is unique and different from that the classifier ensemble is to achieve. For example, each one-versus-rest SVM serves as an expert in discriminating one language from others, while the classifier ensemble is to discriminate all languages. The ensemble classifier makes the final decision given the output codes pooled from the component classifiers as formulated in Section II-E.

It is found that the system performance improves as the output code size increases. Note that larger output code size needs more computation. It is a challenge to select component classifiers that are small in size yet achieve good performance. The empirical results in this paper suggest that one-versus-rest, two-versus-rest, three-versus-rest, and so on, are those of high discriminative ability.

It is reported that score fusion from multiple output codes achieves equal-error-rate of 1.38% and 3.20% on the 2003 and 2005 NIST LRE databases for 30-second trials, which benefits from both phonotactic and acoustic language cues. The system in this paper contributed to the IIR's submission to the 2005 NIST Language Recognition Evaluation (LRE).

We have discussed the applications of the output coding techniques in two vector space modeling applications. Without loss of generality, the idea of output coding and the system architecture in Fig. 1 and Fig. 2 can be extended to applications based on GMM supervector [27] or MLLR supervector [28]. Furthermore, dimensionality reduction is one of the important subjects in pattern classification in general. We believe that the proposed output coding approach is easily applicable in other pattern classification applications as well.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Khe Chai Sim and Dr. Hanwu Sun of the Institute of Infocomm Research, Singapore for their assistance in the experiments. The authors also benefited greatly from many valuable discussions with them.

REFERENCES

- [1] E. Singer, P. A. Torres-Carrasquillo, T. P. Gleason, W. M. Campbell and D. A. Reynolds, "Acoustic, phonetic and discriminative approaches to automatic language recognition," in *Proc. Eurospeech*, pp. 1345-1348, 2003.
- [2] M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech and Audio Processing*, Vol. 4, No. 1, pp. 31-44, 1996.
- [3] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer and P. A. Torres-Carrasquillo "Support vector machines for speaker and language recognition," *Computer Speech and Language*, Vol. 20, pp. 210-229, 2006.
- [4] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. Deller, Jr., "Approaches to language identification using Gaussian mixture models and shifted delta cepstral features," in *Proc. ICSLP*, pp. 89-92, 2002.
- [5] T. J. Hazen and V. W. Zue, "Recent improvements in an approach to segment-based automatic language identification," in *Proc. ICSLP*, pp. 1883-1886, 1994.
- [6] K. M. Berkling, and E. Barnard, "Analysis of phoneme-based features for language identification," in *Proc. ICASSP*, pp. 289-292, 1994.
- [7] K. M. Berkling, and E. Barnard, "Language identification of six languages based on a common set of broad phonemes," in *Proc. ICSLP*, pp. 1891-1894, 1994.
- [8] C. Corredor-Ardoy, J. L. Gauvain, M. Adda-Decker, L. Lamel, "Language identification with language-independent acoustic models," in *Proc. Eurospeech*, pp. 55-58, 1997.
- [9] H. Li and B. Ma, "A phonotactic language model for spoken language identification," in *Proc. ACL*, pp. 512-522, 2005.
- [10] B. Ma, H. Li and C.-H. Lee, "An acoustic segment modeling approach to automatic language identification," in *Proc. Interspeech*, pp. 2829-2832, 2005.
- [11] J. L. Gauvain, A. Messaoudi, and H. Schwenk. "Language recognition using phone lattices," in *Proc. ICSLP*, pp. 1215-1218, 2004.
- [12] R. Tong, B. Ma, D. Zhu, H. Li and E. S. Chng, "Integrating acoustic, prosodic and phonotactic features for spoken language identification," in *Proc. ICASSP*, pp. 205-208, 2006.
- [13] H. Li, B. Ma and C.-H. Lee, "A vector space modeling approach to spoken language identification," *IEEE Trans. Audio, Speech and Language Processing*, Vol. 15, No. 1, pp. 271-284, 2007.
- [14] J. R. Bellegarda, "Exploiting latent semantic information in statistical language modeling," in *Proc. IEEE*, Vol. 88, No. 8, pp. 1279-1296, 2000.
- [15] M. Kobayashi and M. Aono, "Vector space models for search and cluster mining," *Survey of Text Mining*, M. W. Berry (ed), Springer, 2003.
- [16] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp.263-286, 1995.
- [17] K. Crammer, and Y. Singer, "Improved output coding for classification using continuous relaxation," in *Proc. NIPS*, pp. 437-443, 2000.
- [18] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, John Wiley & Sons, Inc., 2001.
- [19] S. Gao, B. Ma, H. Li and C.-H. Lee, "A text-categorization approach to spoken language identification," in *Proc. Interspeech*, pp. 2837-2840, 2005.
- [20] V. Vapnik, *The nature of statistical learning theory*. Springer-Verlag, 1995.
- [21] H. Kim, P. Howland and H. Park, "Dimension reduction in text classification with support vector machines," *Journal of Machine Learning Research*, 6, pp.37-53, 2005.
- [22] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc of European Conference on Machine Learning*, Berlin, 1998.

- [23] H.-C. Wang, "MAT - A project to collect Mandarin speech data through telephone networks in Taiwan," *Int. J. Comput. Linguistics Chinese Language Process*, Vol. 2, No. 1, pp. 73-89, 1997.
- [24] L. Ferrer, E. Shriberg, S. S. Kajarekar, A. Stolcke, K. Sonmez, A. Venkatarman and H. Bratt, "The contribution of cepstral and stylistic features to SRI's NIST speaker recognition evaluation system," in *Proc. ICASSP*, pp. 101-104, 2006.
- [25] L. Burget, P. Matejka and J. Cernocky, "Discriminative training techniques for acoustic language identification," in *Proc. ICASSP*, pp. 209-112, 2006.
- [26] A. Berger, "Error-correcting output coding for text classification," in *Proc. Workshop on machine learning for information filtering - IJCAI*, 1999.
- [27] W. M. Campbell, D. E. Sturim and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," in *Proc. IEEE Signal Processing Letters*, Vol. 13, No. 5, pp. 308-311, 2006.
- [28] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg and A. Venkataraman, "MLLR transforms as features in speaker recognition," in *Proc Interspeech*, pp. 2425-2428, 2005.