42. Vector-Based Spoken Language Classification

H. Li, B. Ma, C.-H. Lee

This chapter presents a vector space characterization (VSC) approach to automatic spoken language classification. It is assumed that the space of all spoken utterances can be represented by a universal set of fundamental acoustic units common to all languages. We address research issues related to defining the set of fundamental acoustic units, modeling these units, transcribing speech utterances with these unit models and designing vector-based decision rules for spoken language classification. The proposed VSC approach is evaluated on the 1996 and 2003 National Institute of Standards and Technology (NIST) language recognition evaluation tasks. It is shown that the VSC framework is capable of incorporating any combination of existing vector-based feature representations and classifier designs. We will demonstrate that the VSC-based classification systems achieve competitively low error rates for both spoken language identification and verification.

The chapter is organized as follows. In Sect. 42.1, we introduce the concept of vector space characterization of spoken utterance and establish the notion of acoustic letter, acoustic word and spoken document. In Sect. 42.2 we discuss acoustic segment modeling in relation to augmented phoneme inventory. In Sect. 42.3, we discuss voice tokenization and spoken document vectorization. In Sect. 42.4, we discuss vector-based classifier design strategies. In Sect. 42.5, we report several experiments as the case study of classifier design, and the analytic study of front- and back-end. Finally in Sect. 42.6, we summarize the discussions.

Spoken language classification is the process of determining the language identity of a given spoken utterance. Like many frontiers in pattern recognition, spoken language classification has been formulated in the framework of statistical modeling, where a model is built for each spoken language by fitting the model parameters to maximize the observed data likelihood. A spoken language classification system typically consists of a front- and a back-end. The front-end extracts features in order to characterize spoken languages, while the back-end makes classification decision based on the feature pattern of a test utterance.

When human beings are constantly exposed to a language without being given any linguistic knowledge, they learn to determine the language identity by perceiving some of the language cues. This motivates us to explore useful language cues from phonotactic statistics of spoken languages. Suppose that the world's languages can be characterized by a universal set of acoustic units. Any speech segment can now be considered as a realization of a concatenation of such units analogous to representing a text passage as a sequence of basic symbols, such as English letters or Chinese characters. Using a collection

of models, one for each unit, a given speech utterance can be automatically transcribed into a sequence of sound alphabets or *acoustic letters* By grouping adjacent *acoustic letters* to form *acoustic words* we can convert any collection of speech segments into a *spoken document*. Similar to representing a text document as a term vector – as is done in information retrieval and text categorization – a spoken document can now be represented by a vector with key terms defined as the set of all acoustic words. This process is also referred to as vectorization, which leads us into several interesting topics, such as selection of acoustic letters and acoustic words, and dimensionality reduction of term vector.

With recent advances in machine learning techniques, vector space modeling has emerged as a promising alternative solution to multiclass speech classification problem. The vector space approach inherits several attractive properties that we discover in text-based information retrieval. For example, it handles the fusion of different types of language cues in a high-dimensional vector seamlessly. As opposed to similarity-based likelihood measurement, the vector space approach is motivated by discriminative training, which is aimed at minimizing misclassification error. In this chapter, we are interested in practical issues related to vectorization of spoken document and vector-based classifier design.

42.1 Vector Space Characterization

In a typical setting, spoken language classification is usually formulated as a probabilistic pattern recognition problem [42.1] in which the a posterior probability, $P(X|\lambda_l)$, of the l-th language to be considered with a model λ_l given an unknown utterance X, is computed. To make a decision, the language that gives the maximum $P(X|\lambda_l)$ is usually identified as the target language. Many algorithms developed in automatic speech and speaker recognition [42.2] have been adopted and extended to language recognition. Recent advances in acoustic and language modeling [42.3] have also contributed to the technological progress of pattern recognition approaches to spoken language classification.

On another front vector space characterization has become a prevailing paradigm in the information retrieval (IR) community since its introduction in the early 1970s [42.4]. Inspired by recent advances in machine learning, a wide variety of VSC approaches have been adopted successfully to text categorization (TC) [42.5], which essentially refers to the assigning of a category or a topic to a given text document based on the frequencies of co-occurrences of certain key terms [42.6] in the particular category or topic of interest. A similar VSC treatment of spoken documents has also been attempted [42.7]. Since the feature vectors to be considered here are often very high dimensional, sometimes in tens or even hundreds of thousands in some cases, a probabilistic characterization of these vectors is usually not an easy task due to several factors: the lack of training data, the difficulty to specify a good model, and the curse of dimensionality in the estimation of so many parameters with so little data. Nonetheless, vector-based classifier design has some of its own attractive properties that does deserve some attention even without using conventional statistical modeling techniques.

In this chapter, we explore VSC characterization of spoken utterances and the VSC-based spoken language classification system design. We consider that a particular spoken language will always contain a set of high-frequency function words, prefixes, and suffixes, which are realized as acoustic substrings in spoken utterances. Individually, those substrings may be shared across languages. Collectively, the pattern of their co-occurrences can facilitate the discrimination of one language from another.

Suppose that an utterance X, represented by a sequence of speech feature vectors $\boldsymbol{O}$, is decoded or tokenized, into a *spoken document*, $T(X)$, consisting of a series of I acoustic units, $T(X) = \{t_1 \ldots, t_i \ldots, t_I\}$, each unit is drawn from a universal inventory, $U = \{u_1, \ldots, u_j, \ldots u_J\}$, of J acoustic letters shared by all the spoken languages to be considered, such that $t_i \in U$. In the following, we will refer to the process of decoding speech into spoken documents as *tokenization*. We are then able to establish a collection of acoustic words by grouping units occurring in consecutive orders to obtain a vocabulary of M distinct words, $W = \{w_1, \ldots, w_m, \ldots w_M\}$, such that each w_m can be a single-letter word like (u_j), a double-letter word like $(u_j u_k)$, a triple-letter word like $(u_j u_k u_l)$, and so on. Usually the vocabulary size, M, is equal to the total number of n-gram patterns needed to form words, e.g., $M = J + J \times J + J \times J \times J$ if only up to three tokens are considered as a valid acoustic word. Next, we can use

some form of function $f(w_m)$, such as latent semantic indexing (LSI) [42.8], to evaluate the significance of having the word w_m in the document, $T(X)$. We are now ready to establish an M-dimension feature vector, $\boldsymbol{v} = [f(w_1), \ldots, f(w_m), \ldots f(w_M)]^{\mathrm{T}}$ with $\boldsymbol{x}^{\mathrm{T}}$ denoting the transpose of vector $\boldsymbol{x}$, for each spoken document.

It is clear that we need a not-too-small number of fundamental acoustic units to cover the acoustic variation in the sound space. However a large J will result in a feature vector with a very high dimension if we would like to cover as many unit combinations when forming acoustic words. For example, with a moderate value of $J = 256$, we have $M = 65\,792$ even when we only consider words less than or equal to two letters. This is already a very large dimensionality not commonly utilized in speech and language processing algorithms. Finally, a VSC-based classifier evaluates a goodness of fit, or score function $S_l(\boldsymbol{v}) = S(\boldsymbol{v}|\lambda_l)$, between a given vector, $\boldsymbol{v}$, and a model of the l-th spoken language, λ_l, to make a decision. Any vector-based classifier can be used to design spoken language identification and verification systems.

In language identification, we assume that each language to be recognized by the system has been registered and known to the system in advance. Therefore, it is a closed-set test. The objective is to identify the language l, among a pool of L candidates, that has the closest match or the highest score to a given test sample. System performance is often measured by the recognition error rate. On the other hand, in language verification, also known as language detection, the test language may or may not be known in advance to the system. This is therefore an open-set test. The objective of a verification test is to confirm whether a test sample belongs to the language that it is claimed to be. In this case, the test sample is compared against the model of the claimed language to produce a verification score. A decision is then made based upon whether the score is above or below a threshold. Two types of errors, *false alarms* (or false positives) and *misdetections* (or false negatives), are thus resulted. A wide range of error pairs can be obtained depending on the operating verification threshold of the system. To facilitate comparisons between different systems, an equal error rate (EER), indicating the error when the two rates are the same, is usually reported as the system performance. The NIST language recognition evaluation (LRE) is a series of open evaluations for benchmarking the progress of ongoing technology (http://www.nist.gov/speech/tests/index.htm).

Three sets of issues to be addressed in designing a VSC-based language classification system include: (i) fundamental unit selection and modeling, (ii) extraction of a spoken document vector, which can be considered as a front-end design, and (iii) vector-based classifier learning, which is labeled as a back-end design. These will be discussed in detail in the following three sections, respectively. Issues related to the accuracy of tokenization, the discrimination of feature vectors, and the performance of language classifiers will also be discussed in the remainder of the chapter. By having various combinations of the front- and back-ends, we have the flexibility to design a collection of systems that cover a wide spectrum of system performances and computation complexities.

42.2 Unit Selection and Modeling

The first issue to be considered is selection and modeling of the universal set of fundamental acoustic units. Spoken languages, despite sounding different from one another, do share some basic similarities in their acoustic characteristics. In the following, we list the 10 most common words in English and Chinese:

- *the* ... *of* ... *to* ... *a* ... *and* ... *in* ... *that* ... *for* ... *one* ... *is* ... (English)
- *de* (的) ... *yi* (一) ... *he* (和) ... *zai* (在) ... *shi* (是) ... *le* (了) ... *bu* (不) ... *you* (有) ... *zhe* (这) ... *ge* (个) ... (Mandarin)

By coincidence, the top-ranked English word, ("*the*"), and the most common Chinese Mandarin word, ("*de*"), share a similar pronunciation, resulting in a similar unit transcription if a common collection of sound unit models are used to decode both English and Mandarin speech segments containing this pair of words. Nonetheless, we can rely on the co-occurrence statistics of those words to discriminate one language from another. As discussed in Sect. 42.1, a decoder or tokenizer is needed to convert spoken utterances into sequences of fundamental acoustic units specified in an acoustic inventory. We believe that units that are not linked to a particular phonetic definition can be more universal, and are therefore conceptually easier to adopt. Such acoustic units are thus highly desirable for universal language characterization, especially for rarely observed languages or languages

without an orthographic system or a well-documented phonetic dictionary.

A number of variants have been developed along these lines, which were referred to as language-independent acoustic phone models. Hazen reported using 87 phones from the Oregon Graduate Institute multilanguage telephone speech corpus (OGI-TS) corpus [42.9]. *Berkling* [42.10] explored the possibility of finding and using only those phones that best discriminate between language pairs. *Berkling* [42.11] and *Corredor-Ardoy* [42.12] used phone clustering algorithms to find a common set of phones for languages. However, these systems are constrained to operate only when a phonetically transcribed database is available. On a separate front, a general effort to circumvent the need for phonetic transcription can be traced back to *Lee*'s work [42.13] in acoustic segment models (ASMs) in which a collection of ASMs were used to characterize the fundamental set of speech units, and constructed in an unsupervised manner without any linguistic definition. Acoustic words were thus formed by decoding word examples into sequences of such acoustic units, and then utilized for medium-vocabulary isolated-word recognition. A similar ASM approach has recently been adopted for language identification [42.14]. We first discuss the grouping of phone sets from multiple phone inventories of different languages to form a universal collection of units and corresponding phone models.

42.2.1 Augmented Phoneme Inventory (API)

Attempts have been made to derive a universal collection of phones to cover all sounds described in an international phonetic inventory, e.g., the international phonetic alphabet or Worldbet [42.15]. This is a challenging endeavor in practice. Note that these sounds overlap considerably across languages. One possible approximation is to form a superset of phonemes from several languages. We call this set the augmented phoneme inventory (API). This idea has been explored in one way or another in many studies [42.9–11]. A good inventory needs to cover phonetically as many targeted languages as possible. This method can be effective when phonemes from all the spoken languages form a closed set as studied by *Hazen* [42.9]. Results from human perceptual experiments have also shown a similar effect whereby listeners' language identification performance improved by increasing their exposure to multiple languages [42.16].

The API-based tokenization was recently studied [42.17] by using a set of all 124 phones from English, Korean, and Mandarin, plus four noise units, and extrapolating them to the other nine languages in the NIST LRE tasks. This set of 128 units is referred to as API I, which is a proprietary phone set defined for an internal database called the Institute for Infocomm Research language identification (IIR-LID) database. Many preliminary experiments were conducted using the IIR-LID database and the API I phone set. For example, we explored an API-based approach to universal language characterization [42.17], and a text categorization approach [42.7], which formed the basis for the vector-based feature extraction to be discussed in the next section. To expand the acoustic and phonetic coverage, we used another larger set of APIs with 258 phones, from the six languages defined by the OGI-TS database. These six languages all appear in the NIST LRE tasks. This set will be referred to as API II. A detailed breakdown of how the two phone sets were formed with phone set counts for each language is listed in Table 42.1.

Table 42.1 The languages and phone sets of API I and II

API I	Count	API II	Count
English	44	English	48
Mandarin	43	Mandarin	39
Korean	37	German	52
General	4	Hindi	51
		Japanese	32
		Spanish	36
Total	128	Total	258

Once the set of units are defined, models for each individual language can be obtained using conventional hidden Markov model (HMM) [42.18] tools trained on the speech examples specifically collected for the particular language. The collection of these phone models can be used to decode a given utterance by performing parallel phone recognition (PPR) [42.1]. The resulting multiple sequences of phone units can be used to form the spoken document vector corresponding to the given utterance. We will discuss both PPR and spoken document vectorization in later sections.

42.2.2 Acoustic Segment Model (ASM)

The conventional phone-based language characterization commonly used in automatic speech recognition and spoken language identification suffers from two major shortcomings. First, a combined phone set, such as API I mentioned above, from a limited set of multiple languages cannot be easily extended to cover new and rarely observed languages. Second, to train the acoustic mod-

els for each language, a collection of transcribed speech data is needed, but is often difficult to come by for all languages of interest. Furthermore, acoustic mismatches may exist among speech collected in different countries and under various acoustic conditions so that the collection of phone models may not work well for some intended languages under diverse test environments. To alleviate these difficulties, a data-driven method that does not rely on phonetically transcribed data is often preferred. This can be accomplished by constructing consistent acoustic segment models [42.13] intended to cover the entire sound space of all spoken languages in an unsupervised manner using the entire collection of speech training examples for all languages. The ASM framework takes advantage of the concept of language-independent acoustic phone models, and benefits from the unsupervised acoustic modeling technique.

Next, we present an ASM method to establish a universal representation of acoustic units for multiple languages [42.17]. As in any other hidden Markov modeling approaches, the initialization of ASMs is a critical factor to the success of ASM-based systems. Note that the unsupervised, data-driven procedure to obtain ASMs may result in many unnecessary small segments because of the lack of phonetic or prosodic constraints (e.g., the number of segments in a word and the duration of an ASM) during segmentation. This is especially severe in the case of segmenting a huge collection of speech utterances given by a large population of speakers from different language backgrounds. The API approach uses phonetically defined units in the sound inventory. It has the advantage of having phonetic constraints in the segmentation process. By using the API to bootstrap ASMs, we effectively incorporate some phonetic knowledge relating to a few languages in the initialization to guide the ASM training process, which is described as follows.

Step 1
Carefully select a few languages, typically with large amounts of labeled data, and train language-specific phone models. Choose a set of J models for bootstrapping.

Step 2
Use these J models to decode all training utterances in the training corpora. Assume the recognized sequences are *true* labels.

Step 3
Force-align and segment all utterances in the training corpora, using the available set of labels and HMMs.

Step 4
Group all segments corresponding to a specific label into a class. Use these segments to retrain an HMM.

Step 5
Repeat steps 2–4 several times until convergence.

In this procedure, we jointly optimize the J models as well as the segmentation of all utterances. This is equivalent to the commonly adopted segmental maximum likelihood (ML) and k-means HMM training algorithm [42.18] through iterative optimization of segmentation and maximization. We found that API-bootstrapped ASMs are more stable than the randomly initialized ASMs. There are also other ways of initializing ASMs, by imposing language-independent phonetic or prosodic constraints. The API-bootstrapped ASMs outperforms the API by a big margin in our 1996 NIST LRE task. Readers are referred to [42.17] for details of these ASM experiments. Using this single set of ASMs, we can now transcribe all spoken utterances from any language into sequences of units coming from a common alphabet. We call this process universal voice tokenization (UVT). Instead of using a single set of universal ASMs, we can also utilize multiple sets of fundamental units for parallel voice tokenization (PVT). For example, starting with the three-language API I units defined above, we can train three sets of language-dependent ASMs, bootstrapped from the phone models for each of the three languages. In this way, we can generate multiple sets of spoken documents, one from each language-specific tokenizer.

42.2.3 Comparison of Unit Selection

With an established acoustic inventory obtained from the API or the ASM method, we are now able to tokenize any given speech utterance into a token sequence $T(X)$, in a form similar to a text-like document. Note that ASMs are trained in a self-organized manner. We may not be able to establish a phonetic lexicon using ASMs and translate an ASM sequence into words. However, as far as language classification is concerned, we are more interested in a consistent tokenization than the underlying lexical characterization of a spoken utterance. The self-organizing ASM modeling approach offers a key advantage: we no longer require the training speech data to be phonetically transcribed.

In summary, the main difference between the API and ASM methods is the relaxation of phone transcription for segmentation. In the API approach, we train the phone models according to manually transcribed phone

labels while via ASMs, the segmentation is done in iterations using automatic recognition results. In this way, ASM gives us two advantages: (i) it allows us to adjust a set of API phones from a small number of selected languages towards a larger set of targeted languages; (ii) the ASM can be trained on the similar acoustic data as are used for the intended task, thus potentially minimizing mismatch between the test data and the API that is trained on a prior set of phonetically transcribed speech.

42.3 Front-End: Voice Tokenization and Spoken Document Vectorization

We are now ready to represent a spoken document or a spoken query by a vector whose dimensionality is equal to the size of the total number of useful features, including the statistics of the units and their co-occurrences. It is precisely with the usage of such high-dimensional vectors that we expect the discrimination capability of the feature vector to improve the performance even when only phonotactic features are used. The voice tokenization (VT) discussed here can typically be accomplished in a way similar to continuous phone recognition to decode a spoken utterance into a sequence of phone units. Readers are referred to [42.3] for an in-depth discussion of the modeling and decoding techniques.

Figure 42.1a,b illustrates a vectorization process that converts a spoken utterance into a document vector with the PVT and UVT front-ends. An unknown testing utterance is represented as a query vector by the front-end. For the PVT front-end with an F-language superset of APIs as initialization, the VSC vectorization (Fig. 42.1a) forms a large composite document vector, or supervector, $\boldsymbol{v} = [\boldsymbol{v}_1^{\mathrm{T}}, \ldots, \boldsymbol{v}_f^{\mathrm{T}}, \ldots, \boldsymbol{v}_F^{\mathrm{T}}]^{\mathrm{T}}$ by stacking F vectors, with each vector, $\boldsymbol{v}_f$, representing the document vector of the f-th language, obtained from the individual phone recognizers. On the other hand, for the UVT front-end, it constructs a single document vector $\boldsymbol{v}$ from the single phone recognizer. Figure 42.2a,b illustrate the language identification and verification processes using a four-language API superset example.

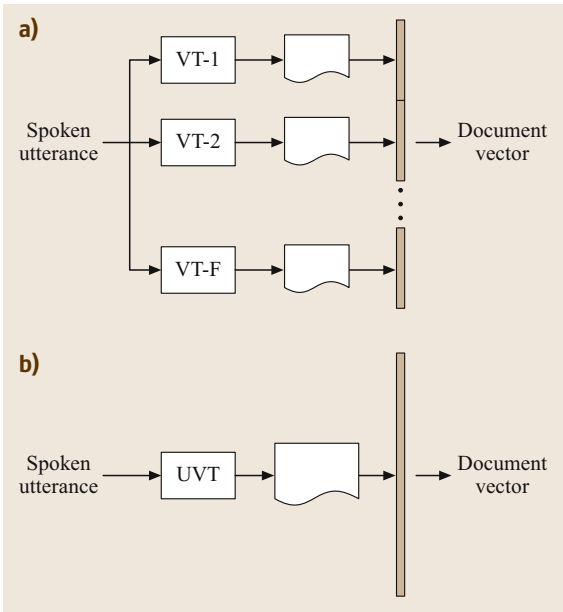

Fig. 42.1a,b Front-end: spoken utterance tokenization and vectorization **(a)** Spoken utterance tokenization and vectorization with PVT. **(b)** Spoken utterance tokenization and vectorization with UVT.

Since high-performance language-dependent acoustic and language models of phones are likely to be unnecessary for tokenization, we no longer need large amounts of training speech samples from each of the spoken languages. Instead, we only require a moderate-sized language-dependent training set of spoken documents in order to obtain a collection of spoken document vectors to be used to train a language-specific classifier for each language. When trigrams with 128 ASM units are incorporated, the vector dimensionality increases to over two million, which is well beyond the capability of current technology. It would be useful to find a balance between the *acoustic resolution*, i. e., the number of units J, needed to model the universal sound space for all languages, and the *language resolution*, i. e., the dimensionality of the spoken document vector M, needed to provide an adequate discriminative power for language identification. Readers are referred to [42.19] for a detailed study.

Many studies on vector-based document representation are available in information retrieval and text categorization literature [42.4, 5, 7, 20, 21]. In this chapter, we focus on language feature characterization that is used to discriminate between languages. Intuitively, sounds are heavily shared across different spoken languages because of the same human speech production mechanism that is used to produce them. The acoustic unit as proposed in Sect. 42.2 allows us to move away

from the conventional lexical descriptions of spoken languages. To account for the sequential, acoustic–phonotactic constraints, such as lexical constraints, we introduce the concept of an acoustic word (AW). An AW is typically smaller than a lexical word, and is composed of acoustic letters, such as the set of acoustic segment units, in an n-gram form. By exploiting the co-occurrence statistics of AWs, we can improve the discrimination power of the document vectors by incorporating acoustic words of different orders.

Suppose that we have a token sequence, $\{t_1, t_2, t_3, t_4\}$. We first derive the unigram statistics from the token sequence itself. We then compute the bigram statistics from $(\#t_1)$, (t_1t_2), (t_2t_3), (t_3t_4) and $(t_4\#)$ where the acoustic vocabulary is expanded to the token's right context. Similarly, we can extend this to the trigram statistics using $(\#t_1t_2)$, $(t_1t_2t_3)$, $(t_2t_3t_4)$ and $(t_3t_4\#)$ to account for both left and right contexts. The # sign is a place holder for free (or *do not care*) context. In the interest of manageability, we use only up to token trigrams. In this way, for an acoustic vocabulary of J tokens, we have potentially $J \times J$ bigram and $J \times J \times J$ trigram AWs to form a vocabulary of $M = J + J \times J + J \times J \times J$ AWs. Let us use the API I set defined in Sect. 42.2.1 to illustrate the concept of PVT-based vectorization. As shown in the left half of Table 42.1 that there are three subsets of units of sizes 44, 43, and 37 for English, Mandarin, and Korean, respectively. If we form AWs of up to two letters, this results in a lexicon of size $44^2 + 43^2 + 37^2 + 44 + 43 + 37 = 5278$, which is also the dimensionality of the document vectors.

This *bag-of-sounds* concept [42.22] is analogous to the *bag-of-words* paradigm originally formulated in the context of information retrieval and text categorization. In human languages, some words invariably occur more frequently than others. One of the most common ways to express this notion is known as the Zipf's law [42.23,24], which states that there is always a set of words that dominates most of the other words of a language in terms of their frequency of use. From some preliminary data, the theory can be extended to spoken words as well. This motivates the VSC-based approach.

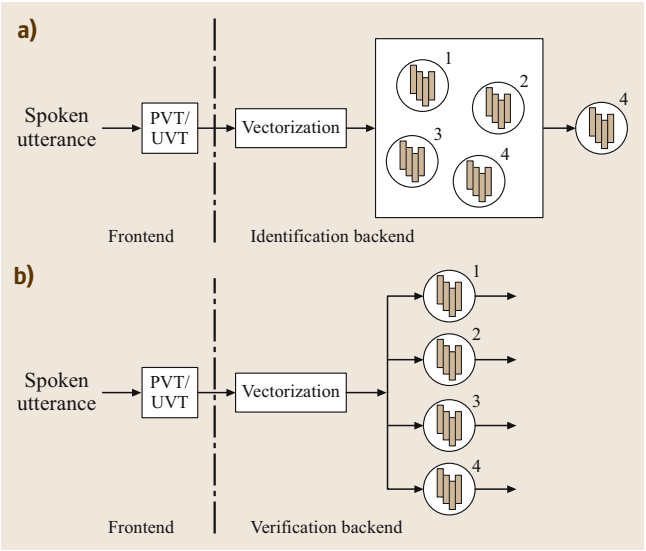

Fig. 42.2a,b Back-end: spoken language classification **(a)** Language identification using multiclass classification (four languages). **(b)** Language verification with independent Bayes decisions (four languages).

42.4 Back-End: Vector-Based Classifier Design

After a spoken utterance is tokenized and vectorized into a high-dimensional vector, language classification can be cast as a vector-based classification problem. There are many ways to construct the VSC back-end. The support vector machine (SVM) framework is a popular choice in many applications [42.5, 25]. Since the training of SVMs is usually optimized on a structural risk-minimization principle [42.26], SVMs have been shown to achieve good performances in several real-world tasks. In the rest of the chapter, we will use the SVM framework to illustrate the design of language classification systems. Other vector-based classifiers can also be utilized. Spoken language identification and verification system block diagrams for a four-language ($L = 4$) case are illustrated in Figs. 42.2a and b, respectively.

To train a two-class SVM, also known as a binary SVM, any input document vector $\boldsymbol{v}$ is labeled as Y_+ or Y_-, depending on whether the input belongs to the desired language category or not. Given a training database of D spoken document vectors, the binary SVM is a classifier of the form $g(\boldsymbol{v}) = \Omega^{\mathrm{T}}\psi(\boldsymbol{v}) + \omega_0$, characterized by a weight vector Ω, an offset ω_0, and a kernel function $\psi(\cdot)$. We can augment Ω with ω_0 and $\psi(\boldsymbol{v})$ with a unity, thus eliminating ω_0 in the rest of our discussion. SVM learning is usually posed as

an optimization problem with the goal of maximizing a margin, i. e., the distance between the separating hyperplane, $\Omega^{\mathrm{T}} \cdot \psi(\boldsymbol{v}) = 0$, and the nearest training vectors, such that if $g(\boldsymbol{v}) > 0$, then $\boldsymbol{v} \in Y_+$, and if $g(\boldsymbol{v}) \leq 0$, then $\boldsymbol{v} \in Y_-$. An extension of this formulation also allows for a wider margin at the cost of misclassifying some of the training examples. We used the SVM$^{\mathrm{light}}$ version 6.01 program to train all SVM models discussed in this chapter (http://svmlight.joachims.org/). This package allows us to explore both linear and nonlinear SVM kernels. A linear kernel SVM has $\psi(\boldsymbol{v}) = \boldsymbol{v}$. Other forms of kernels can also be used. Without loss of generality, we limit our discussion to linear-kernel SVM in the rest of this chapter.

42.4.1 Ensemble Classifier Design

When the number of classes L is greater than two, maximum margin classifier designs do not extend easily. Various approaches have been suggested. Some follow the concept of ensemble classifiers by having a collection of binary SVMs, such as *one-versus-rest*, *one-versus-one*, and their variants [42.27]; others build decision trees using top-down greedy algorithms or bottom-up clustering algorithms. We discuss these in detail in the present section.

Given L competing classes with all training vectors labeled as one of the L classes, the training set, T, is divided into a collection of L subsets, $T = \{T_1 \ldots, T_l \ldots, T_L\}$, with all samples that are labeled as class l belong to the set T_l.

One-Versus-Rest (OVR) SVM

This is conceptually the simplest multiclass ensemble classifier. We build L binary SVMs, each with the target l-th class as the positive category, with positive training set T_l, and the rest belonging to the negative category, with training set $\bar{T}_l = \{\boldsymbol{v}_i | i \neq l\}$ of negative examples. This approach is usually computationally expensive since we need to solve L quadratic program (QP) optimization problems, each involving D vectors with a large dimensionality, M. Given a test sample, the OVR decision function chooses the class that corresponds to the maximum value specified by the furthest hyperplane. As illustrated in Fig. 42.3a, the QP solutions result in the dotted line partitions while the decisions are made based on the solid line partitions. This gives errors as shown in the shaded area of Fig. 42.3a. Since there are usually more negative samples than positive ones, the OVR SVMs are usually hard to train because of the imbalance in the data distribution.

One-Versus-One (OVO) SVM

This method constructs binary SVMs for all $L \times (L-1)/2$ pairs of classes, one SVM for each pair of competing classes. When compared with OVR SVM, the OVO SVM method significantly reduces the computational needs in training because there are less training examples to train each pairwise SVM. In total, we need to solve $L \times (L-1)/2$ QP optimization problems. For each SVM, only data from the two competing classes are involved. Hence, we deal with a much smaller QP problem than that in OVR SVM. However, OVO SVM still leaves some undecided regions as shown in the shaded area of Fig. 42.3b.

Generalized SVM

This method is similar to OVR except that it modifies the support vector loss function to deal directly with multiple classes [42.28, 29]. Instead of solving a collection of independent SVMs, this method seeks to solve a single, monolithic QP problem of size $(L-1) \times D$, that

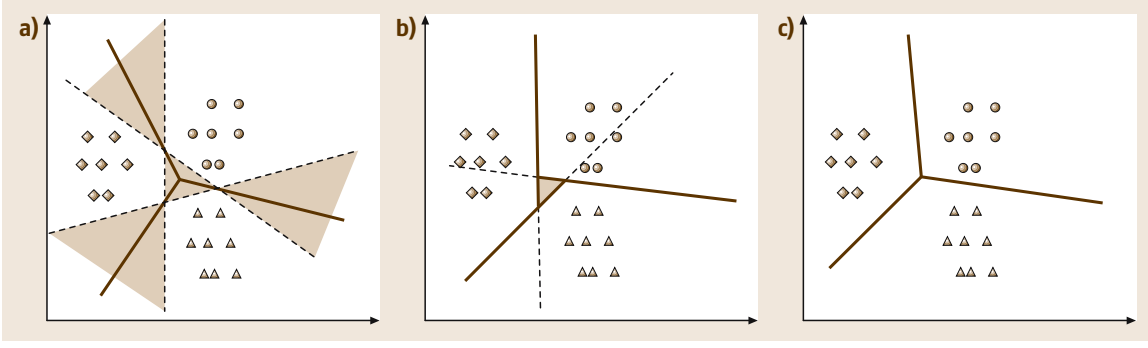

Fig. 42.3a–c SVM-based ensemble classifier design **(a)** One-versus-rest SVM. **(b)** One-versus-one SVM. **(c)** Generalized SVM.

maximizes the margins between all class pairs simultaneously. In general, it is computationally more expensive to solve a monolithic problem than multiple binary subproblems, such as is done by OVO SVMs, with the same amount of training data [42.25]. Generalized SVM can lead to L decision regions as illustrated in Fig. 42.3c. Although generalized SVMs are difficult to learn, other multiclass classifiers can serve this purpose. For example a maximum figure-of-merit (MFoM) learning algorithm has been shown to be effective in discriminative multiclass text categorization [42.6] and spoken language identification [42.7].

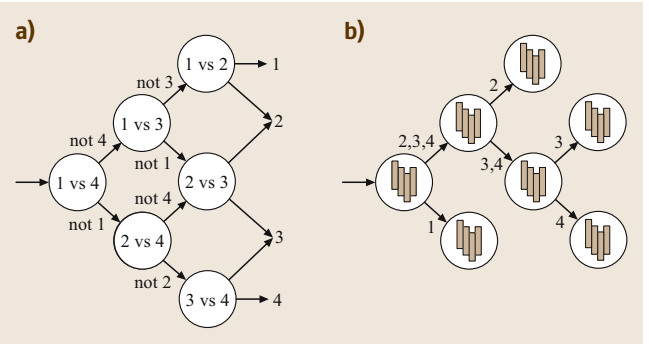

Fig. 42.4a,b Decision strategy **(a)** Direct acyclic graph. **(b)** Binary decision tree.

42.4.2 Ensemble Decision Strategy

The idea of an ensemble decision is to consult a large number of independently constructed classifiers to make a decision based on consensus. It provides a solution to the multiclass classification problem with independently trained binary classifiers. First, we construct many subordinate classifiers, each responsible for removing some uncertainty regarding the class assignment of the test sample; and secondly, we apply classification schemes to make collective decisions. To make a final decision for a multiclass classification problem with an ensemble of binary classifiers, two strategies have been well studied, the maximum wins (max. wins) (MW) and directed acyclic graph (DAG) strategies. The MW strategy makes a collective decision based on a number of individual decisions. Each decision is represented by an individual binary SVM. The collection of individual SVMs is arranged in a flat structure. In practice, we evaluate a test sample against all the $L\times(L-1)/2$ binary SVMs. The class that gains most of the winning votes is then chosen as the identified class.

A DAG is a graph whose edges have an orientation but with no cycles. The DAG strategy [42.30] uses a rooted binary directed acyclic graph which has $L\times(L-1)/2$ internal nodes with L leaves. Figure 42.4a gives an example of a four-class DAG classifier, with six internal nodes and four leaves. Each node represents one of the $L\times(L-1)/2$ binary SVM. To evaluate a test sample $\boldsymbol{v}$, starting at the root node, a binary decision is made at each node to eliminate one candidate class every time a node is visited. The node that survives all the eliminations is eventually chosen as the identified class. Readers can walk through the example in Fig. 42.4a. Thus, for a problem of L-class classification, there are $L-1$ steps of decisions before we arrive at an answer. Furthermore, a node or leaf in DAG is reachable by more than one possible path through the system. Therefore, the decision graph that the path traverses is a DAG, and not simply a tree. In this way, DAG allows for a more efficient representation of redundancies and repetitions that can occur in different branches of the tree, by allowing possible merging of different decision paths. The difference between a DAG and a decision tree can be observed by comparing the tree structures in Fig. 42.4b and the DAG in Fig. 42.4a.

Although both the MW and DAG strategies make decisions based on the same OVO SVMs, the DAG approach demonstrates two clear advantages: (i) with a decision tree architecture, DAG is a much more efficient strategy than the MW during testing. It only involves $(L-1)$ SVM operations, as opposed to $L\times(L-1)/2$ comparisons in MW; and (ii) DAG is amenable to a VC-style bound on the generalization error while bounds on the generalization errors have yet to be establish for the MW method [42.30].

While the MW and DAG strategies can produce accurate class predictions, they lack a structural representation for class similarities. With a large number of classes, we may want to have a meaningful organization of these classes. As shown in Fig. 42.4b, a decision tree presents the classes in a hierarchical structure of clusters and subclusters according to their proximity. It makes a final decision through steps of subdecisions for a test sample. At each node in the tree, a decision is made to reduce the uncertainty of the test samples. Starting from the root node through the successor nodes, we make the final decision when a leaf node is reached.

42.4.3 Generalized VSC-Based Classification

We have discussed designing ensemble classifiers and decision strategies based on high-dimension feature vectors and binary SVMs. In some cases, it is beneficial to

reduce feature dimensions in order to compensate for the lack of training vectors and potential measurement noise in feature extraction. Once we have a vector of lower dimension many well-known probabilistic modeling and machine learning techniques can be used to design classifiers and formulate decision rules.

We also consider an indirect vectorization mechanism based on the scores obtained from the ensemble classifiers discussed earlier. Since these scores characterize the distribution of the outputs of a variety of classifiers, they can be grouped together to form a score *supervector* describing an overall behavior of a score distribution over different classifiers on all competing classes. One example of such a supervector is to concatenate HMM state scores from all competing models to form an overall score vector. This approach has been shown to have good discrimination power in isolated-word recognition [42.31, 32]. Since these score supervectors are usually obtained from a finite set of ensemble classifiers, their dimensionality is usually much smaller than that of the spoken document vectors discussed in Sect. 42.3. These score supervectors are more amendable to using the conventional probabilistic modeling approaches. We call this combination framework generalized VSC-based classification; it uses high-dimensional feature vectors to generate scores from a collection of ensemble classifiers, and forms low-dimensional score supervectors to perform final classifier design and decisions. All the language identification and verification experiments to be discussed in Sect. 42.5 are based on this generalized VSC classification approach.

Feature Reduction

In many cases, it is desirable to reduce the dimensionality of the document vectors to a manageable size so that probabilistic models, such as a Gaussian mixture model (GMM), can be easily utilized. Many dimensionality reduction approaches, such as truncated singular value decomposition (SVD) [42.8] or principal component analysis (PCA) [42.33], have been studied. Suppose that we are given a database with D documents and M distinguished attributes, also known as terms. Let $\boldsymbol{A}$ denote the corresponding $M \times D$ document-term matrix with entries $a_{m,d}$ that represent the importance of the m-th term in the d-th document. SVD effectively reduces the dimensionality by finding the closest rank-R approximation to $\boldsymbol{A}$ in the Frobenius norm, while PCA finds the R-dimensional subspace that best represents the full data with respect to a minimum squared error. Although the SVD or the PCA method finds subspaces that are useful for representing the original high-dimensional vector space, there is no reason to assume that the resulting projections must be useful for discriminating between data in different classes [42.34].

Linear discriminant analysis finds a decision surface, also known as a hyperplane, that minimizes the *sample risk* or *misclassification error* for linearly separable classes. In the two-class case, the linear discriminant function [42.34] is expressed as $g(\boldsymbol{v}) = \Omega^{\mathrm{T}}\boldsymbol{v}$, with $g(\boldsymbol{v})$ representing the signed distance between $\boldsymbol{v}$ and the decision surface $\Omega^{\mathrm{T}}\boldsymbol{v} = 0$. In this way, from the perspective of dimensionality reduction, a multidimensional feature vector $\boldsymbol{v}$ is projected to a one-dimensional space by $g(\boldsymbol{v})$.

Vectorization with Ensemble Scores

If we employ a linear SVM for each of the subordinate classifiers, the outputs $g(\boldsymbol{v})$ from the SVMs then form a vector of signed distance. In this way, a high-dimensional document vector can be effectively reduced to a much lower dimensionality. From an OVR SVM ensemble classifier, we obtain a vector of $Q = L$ dimensionality; from an OVO SVM ensemble classifier, we arrive at a vector of $Q = L \times (L-1)/2$ dimensionality. Clearly, not only do we effectively reduce the dimensionality from a large M to a small Q, but we also represent the spoken document vector in a discriminative space of language pairs.

Studies show that an ensemble classifier performs well when the number of subordinate classifiers Q is set to around $10 \log_2 L$. In general, more subordinate classifiers improve performance, but at a higher computational cost [42.35]. Hence, from the perspective of expressiveness, OVO SVM ensemble classifier seems to be a better choice than OVR SVM when we are dealing with 15 languages as in *CallFriend*, where we have $105 = 15 \times (15-1)/2$ OVO SVM outputs versus 15 OVR SVM outputs (http://www.ldc.upenn.edu/Catalog/project_index.jsp).

Generalized Vector-Based Classifiers and Decision Rules

We mentioned above that generalized SVMs attempt to design ensemble classifiers by considering all possible partitions of the vector space as shown in Fig. 42.3c, which is usually a difficult problem. On the other hand, if we can define a class discriminant function, $S(\boldsymbol{v}|\lambda_l)$ for the l-th class with λ_l as the model for the target l-th class, and be able to consistently rank a goodness-of-fit score between the unknown vector $\boldsymbol{v}$ and the model λ_l, a decision rule for identification can easily be formulated as:

$$\hat{l} = \underset{l}{\operatorname{argmax}}\, S(\boldsymbol{v}|\lambda_l)\,. \tag{42.1}$$

Similarly, a verification decision can be made to accept the claimed class identity l if:

$$S(\boldsymbol{v}|\lambda_l) - S(\boldsymbol{v}|\bar{\lambda}_l) > \tau_l\,, \tag{42.2}$$

with $S(\boldsymbol{v}|\bar{\lambda}_l)$ denoting an antidiscriminant function score that is typically a score representing the collective contribution from all competing classes other than the l-th class, and τ_l denoting the verification threshold for the l-th class.

One good example is a linear discriminant function (LDF)-based score function $S(\boldsymbol{v}|\Omega_l) = \Omega_l{}^{\mathrm{T}}\boldsymbol{v}$ mentioned early in the SVM design, which is simply an inner product between the weight vector Ω_l of the l-th class and the unknown vector $\boldsymbol{v}$. Another commonly used example is to model each class by a GMM, so that we have $S(\boldsymbol{v}|\lambda_l)$, evaluating the likelihood of observing $\boldsymbol{v}$ under GMM λ_l. Artificial neural networks (ANNs) [42.36] are another popular approach that can be used to approximately map the input vectors to the output decisions. Results of using both LDF- and ANN-based generalized classifier design approaches have been recently reported in [42.37] for spoken language classification in the 2005 NIST LRE task. Classifier learning is based on the minimum classification error (MCE) and maximal figure-of-merit (MFoM) training principles which have been extensively studied in and successfully applied to speech recognition [42.38], and text categorization [42.6] tasks.

42.5 Language Classification Experiments and Discussion

We illustrate the VSC framework for the language identification results on the primary subset of 30 s test segments of the 1996 NIST LRE task. On the other hand, language verification benchmarks are established based on the 1996 and 2003 NIST LRE tasks.

42.5.1 Experimental Setup

The training sets for building models came from three sets of corpora, namely: (i) the three-language IIR-LID database [42.7] with English, Mandarin, and Korean; (ii) the six-language OGI-TS (multilanguage telephone speech, http://cslu.cse.ogi.edu/corpora/corpCurrent.html) database with English, German, Hindi, Japanese, Mandarin, and Spanish; and (iii) the 12-language Linguistic Data Consortium (LDC) *CallFriend* database. The overlap between the *CallFriend* database and the 1996 LRE data was removed from the training data as suggested http://www.nist.gov/speech/tests/index.htm for 2003 LRE. As English, Mandarin, and Spanish each have two accented versions in the *CallFriend* database, in all, we were given a 15-language set, which includes the three additional accents. The IIR-LID and OGI-TS databases were only used for bootstrapping the acoustic models as an initial set of phones. Both IIR-LID and OGI-TS databases are telephone speech with phonetic transcriptions. In addition, the *CallFriend* database was used for fully fledged ASM acoustic modeling, vectorization, and classifier design. It contains telephone conversations of the same 12 languages that are in the 1996 and 2003 NIST LRE tasks, but without phonetic transcriptions. The three databases were recorded independently of each other.

In the IIR-LID database, each language contains more than 150 hours of speech, while in the OGI-TS database, each language amounts to less than one hour of speech. Furthermore in the *CallFriend* database, each of the 15 language databases consists of 40 telephone conversations with each lasting approximately 30 min, giving a total of about 20 hours per language. For vectorization and classifier design, each conversation in the training set was segmented into overlapping sessions, resulting in about 12 000 sessions for each duration per language. For testing, there were three different duration settings: 3, 10, and 30 s. The 1996 NIST LRE evaluation data consist of 1503, 1501, and 1492 sessions of speech segments with 3, 10, and 30 s in length, respectively. The 2003 NIST LRE evaluation data consist of 1280 sessions per duration. The test data were labeled with the 12 main languages only, without considering their accent type.

We configured each ASM with a three-state left-to-right hidden Markov model. It was found that a 32-mixture Gaussian state density provides adequate results as compared with a 16- or 8-mixture Gaussian state [42.17]. It was also reported that the ASM set derived from API II (258-ASM) slightly outperforms that from API I (128-ASM) by having a broader acoustic coverage [42.19]. We will adopt the 128-ASM for language identification and the 258-ASM for language verification experiments.

Table 42.2 Error rates (ER%) and number of support vectorson 30 s NIST 1996 LRE

No. sessions per language	1000	2000	6000	12 000	Testing cost (no. SVM decisions)
DAG (ER%)	18.2	16.2	14.4	13.9	$L-1$
Max. wins (ER%)	19.0	16.7	15.1	14.5	$L\times(L-1)/2$
No. support vectors	1048	1457	1951	2142	

With 15 languages including accents in the *CallFriend* database, we trained $105 = 15\times(15-1)/2$ OVO SVMs to model the languages and accents. In language identification, we implemented both MW and DAG decision strategies based on OVO SVMs. The conventional language model (LM) scoring techniques used in the parallel phone recognition followed by language modeling (P-PRLM) approaches [42.1] can also be used as a back-end to contrast the VSC back-end discussed earlier. For language verification, to appreciate the contributions of the different front- and back-ends, we construct and test four combination systems using the PVT and UVT front-ends, and the VSC and LM back-ends. In the 1996 NIST LRE task, we built a system to identify 12 languages.

42.5.2 Language Identification

For language identification, the objective is to identify the most likely language given a test sample, as illustrated in Fig. 42.2a. This is usually considered as a closed-set problem assuming that all the languages to be decided are known to the system in advance. We follow the experiment setup in the NIST LRE tasks, and the evaluation is carried out on recorded telephony speech of 12 languages for the 1996 and 2003 NIST LRE: Arabic, English, Farsi, French, German, Hindi, Japanese, Korean, Mandarin, Spanish, Tamil, and Vietnamese.

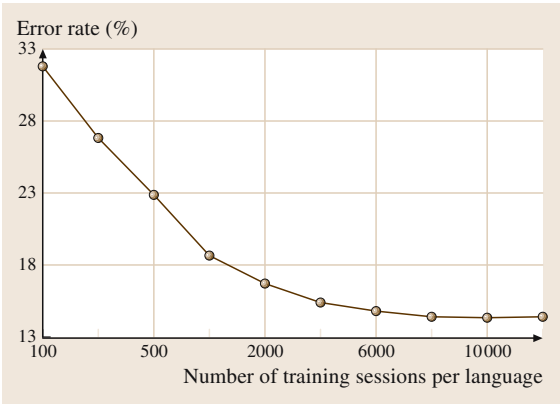

Fig. 42.5 Error rates (ER%) as a function of training set size evaluated on 30 s NIST 1996 LRE

It is well known that the size of the training set often affects the performance of a pattern classification system significantly. Let us study the effects of training set size on system performance using the DAG strategy. We proceed with the document vectors based on bigram AWs derived from the 128-ASM set. In Fig. 42.5, we reported the language identification error rate as a function of the number of training sessions (vectors). As the number of training sessions increase, the training cost of SVMs increase as well. The full corpus includes 12 000 spoken documents, or sessions, for each language/accent. The subset was randomly selected from the full corpus with an equal amount of data from each language. We observe that, when subset size grows beyond 8000 sessions per language, performance begins to saturate.

In Table 42.2, we compare the error rates using different training corpus sizes evaluated on the 30 s NIST 1996 LRE test set. We also report the number of resulting support vectors in the set of binary SVMs. When the number of sessions per language was 1000, there were 2000 training vectors for each language pair. The two-class SVMs gave about 1048 support vectors per binary SVM on average. Note that the number of support vectors increases as training corpus grows, but at a much slower rate than that of the training corpus size. When the training corpus size grows by 12 times, the number of support vectors only doubles. This explains the fact that, beyond 8000 sessions, increasing the training corpus size does not translate into substantial accuracy improvements.

As discussed in Sect. 42.4.1, both the MW and DAG SVM strategies make decisions based on the same OVO SVMs. We further compare the performance of the two strategies in Table 42.2. It is worth pointing out that it is much less expensive to train a set of OVO SVMs than a set of OVR SVMs [42.30]. From the perspective of computation cost in testing, the DAG strategy involves $L/2$ times less computation than MW does. The DAG approach also demonstrates superior performance to that obtained from the MW strategy in terms of accuracy consistency on all tests.

42.5.3 Language Verification

In the case of language verification, the goal is to verify whether a language identity claim is true or false. To this end, a typical VSC back-end consists of multiple independent Bayes decisions, each making a decision for a candidate language, as in Fig. 42.2b. Language verification is an open-set hypothesis testing problem. The language verification experiments discussed next follow the procedure in Sect. 42.4.3. We have described the two different front-ends, PVT and UVT, with both the VSC and LM back-ends. To gain further insight into the behavior of each of the front- and back-ends, it is desirable to investigate the performance of each of the four system combinations, namely PVT–LM, PVT–VSC, UVT–LM, and UVT–VSC, where the PVT/UVT front-ends are built on a set of universal ASMs.

Without loss of generality, we deployed the same 258-ASM in two different settings. First, the 258 ASMs were arranged in a six-language PVT front-end. They were redistributed according to their API II definitions into three languages. Second, they were lumped together in a single UVT front-end. The training of 258-ASM was discussed in Sect. 42.2.2.

The PVT/UVT front-end converts spoken documents into high-dimensional vectors as illustrated in Fig. 42.2a,b. The six-language PVT front-end generates vectors of 11 708 ($=48^2+39^2+52^2+51^2+32^2+36^2+48+39+52+51+32+36$) dimensions (PVT vectors), while the UVT front-end generates those of 66 822 $\left(=258^2+258\right)$ dimensions (UVT vectors).

We can use the vectors generated by the PVT and UVT front-ends directly. We can also use the scores generated by the binary classifiers. With the OVO SVMs, we further convert the document vectors into $105=15\times(15-1)/2$ attributes, with each attribute representing a signed distance between a document vector and the decision hyperplane of an OVO SVM. This score supervector represents 99.1% and 99.8% dimensionality reductions from the original PVT and UVT vector dimensions, respectively. We further train 512-mixture GMMs, λ^{x+} and λ^{x-}, for each of the languages, and report the equal error rates (EER in percentage) between misdetections and false-alarms.

The UVT–LM system follows the block diagram of the language-independent acoustic phone recognition approach [42.12]. The PVT–LM is implemented as in [42.1]. The LM back-end uses trigrams to derive phonotactic scores. The results on the 1996 and 2003 NIST LRE tasks are reported in Tables 42.3 and 42.4, respectively.

Table 42.3 Equal error rates (ERR%) comparison of four systems on NIST 1996 LRE

System	30 s	10 s	3 s
PVT–VSC	2.75	8.23	21.16
PVT–LM	2.92	8.39	18.61
UVT–VSC	4.87	11.18	22.38
UVT–LM	6.78	15.90	27.20

Table 42.4 Equal error rates (ERR%) comparison of four systems on NIST 2003 LRE

System	30 s	10 s	3 s
PVT–VSC	4.02	10.97	21.66
PVT–LM	4.62	11.30	21.18
UVT–VSC	6.81	13.75	24.44
UVT–LM	10.81	19.95	30.48

Before looking into the results, let us examine the combinative effect of the front- and back-ends. In these combinative systems, there are two unique front-end settings, PVT and UVT. The PVT converts an input spoken utterance into six spoken documents using the parallel front-end, while the UVT converts an input into a single document. The LM in the PVT–LM and that in the UVT–LM are different; the former has 6×12 n-gram language models while the latter has only 12 language models. That is to say, the former LM classifier is more complex, with a larger number of parameters. On the other hand, the VSC in the PVT–VSC and the VSC in the UVT–VSC are of different complexities as well. Although the dimensionality of the supervector from PVT is lower than that of UVT, the supervector is several times as dense as that of UVT because there are many low-occurrence AWs in the case of UVT, resulting in more-complex SVMs [42.26]. In other words, the VSC classifier in the PVT–VSC is more complex than that in UVT–VSC. In terms of the overall classifier back-end complexity, we rank the four systems from high to low as follows: PVT–VSC, PVT–LM or UVT–VSC, and UVT–LM.

We now summarize what we have discussed: (i) the VSC back-end demonstrates a clear advantage over the LM back-end for the 30 s trials, while LM works better for the 3 s trials in general. This can easily be explained by the fact that the VSC models are designed to capture higher-order phonotactics. As a result, VSC favors long utterances which provide a richer set of long span phonotactic information over short utterances; (ii) the system performance correlates highly with the complexity of system architectures. This can be found

consistently in Tables 42.3 and 42.4, where the PVT–VSC presents the best result with an EER of 2.75% and 4.02% for the 30 s 1996 and 2003 NIST LRE tasks, respectively. It is then followed by PVT–LM, UVT–VSC, and UVT–LM. Note that we can increase the system complexity by including more phone recognizers in PVT. We expect more phone recognizers to further improve the PVT–VSC system performance. As a general remark, the OVO SVM followed by the DAG decision strategy is shown to be computationally effective in both training and testing in language identification. The OVO SVM also delivers outstanding system performances in both identification and verification tasks.

42.5.4 Overall Performance Comparison

We summarize the overall performance of the SVM approach and compare its results with other state-of-the-art systems recently reported in the literature. Only the results of the 30 s segment testing, which represent the primary interest in the LRE tasks, are listed in Table 42.5. Here systems 1–3 are trained and tested on the same database configuration. Therefore, the results can be directly compared. We extract the corresponding results from Tables 42.3 and 42.4. Two sets of recently reported results are worth mentioning. They are listed under systems 4 and 5, and extracted from [42.39] and [42.40], respectively. It is clear from the results in Table 42.5 that the performance of the PVT–VSC system represents one of the best reported results on the 1996 and 2003 NIST LRE tasks.

Table 42.5 Equal error rates (ERR%) benchmark on 30 s NIST 1996/2003 LRE (in [42.40], Russian data are excluded)

	System	1996 LRE	2003 LRE
1	PVT–VSC	2.75	4.02
2	PVT–LM	2.92	4.62
3	UVT–VSC	4.87	6.81
4	Phone lattice [42.39]	3.20	4.00
5	Parallel PRLM [42.40]	5.60	6.60

The proposed VSC-based language classifier only compares phonotactic statistics from spoken documents. We have not explored the use of the acoustic scores resulting from the tokenization process. It was reported that there is a clear win in combining information about the acoustic scores along with the phonotactic statistics [42.12, 40, 41]. Furthermore, the fusion of phonotactic statistics at different levels of resolutions also improves overall performance [42.42]. We have good reasons to expect that fusion among our four combinative systems, or between our systems and other existing methods including the acoustic score classifier [42.41] and GMM tokenizer [42.43], will produce further improvements. This has been demonstrated by some recently reported results, such as the 2.70% EER on the 2003 NIST LRE in [42.39]. However, it is beyond the scope of this chapter to discuss classifier fusions.

42.6 Summary

We have presented a vector-based spoken language classification framework that addresses a range of questions from the theoretical issue of the nature of the mind to more-practical aspects such as design considerations. As a case study, this chapter covers three areas: (i) a vector space characterization strategy for representing spoken documents; (ii) a systematic discussion of language identification and verification formulation cast in the discriminative classifier design strategy; (iii) a number of practical issues for system implementation in applications such as the NIST LRE. Using a conventional SVM classifier design with the PVT front-end and VSC back-end combination, we achieved an EER of 2.75% and 4.02% in the 30 s 1996 and 2003 NIST LRE tasks, respectively. This represents one of the best reported results using a single classifier.

We have studied a number of practical issues in ensemble classifier design, such as the computational needs in the training and testing of different strategies. In language identification, we have successfully demonstrated through a case study that DAG is a better solution than MW in terms of computational efficiency and system accuracy. Multiclass SVM by itself is still an ongoing research issue. Moving forward, recent progress in discriminative classifier designs, such as MFoM [42.6] and decomposition method [42.25] for monolithic solutions to multiclass problems, will result in further improvements. In language verification, we introduced the concept of using SVM decision hyperplanes as the projection directions for dimensionality reduction. This allows us to carry out language verification under the classical GMM framework. The use of

OVO SVM outputs can also be seen as an implementation of the concept of error-correcting output coding (ECOC) [42.44, 45], which makes classification decision by a consensus among subordinate classifiers. One of the central issues in ECOC is choosing a good code. It will be interesting to study other ECOC codes beyond OVO SVMs.

One of the important properties of vectorization is that it handles the fusion of different types of language cues in a high dimensional vector seamlessly. We have successfully implemented the *bag-of-sounds* spoken document vectors using a mix of n-gram statistics of *acoustic letters*. We believe that this framework can be extended to accommodate heterogeneous attributes such as acoustic and prosodic features as well. Whereas fusion of classifier outputs has been the focus of recent research [42.40, 46], vector space modeling explores a new horizon where fusion can take place at the feature level. The same classifier design theory is applicable to other vector-based representation such as GMM *supervector* [42.47], maximum-likelihood linear regression (MLLR) vector [42.48], and generalized linear discriminant sequence (GLDS) vector representation [42.49].

In summary, vector-based spoken language classification benefits from progress in the areas of acoustic modeling, machine learning, and text-based information retrieval. We have successfully treated spoken language classification as a text categorization problem with the topic category being the language identity itself. The same theory also applies to other spoken document classification tasks, such as topic detection and tracking, and *query-by-example* spoken document retrieval. This is another highly anticipated research direction.

References

42.1 M.A. Zissman: Comparison of four approaches to automatic language identification of telephone speech, IEEE Trans. Speech Audio Process. **4**(1), 31–44 (1996), 1

42.2 C.-H. Lee, F.K. Soong, K.K. Paliwal (Eds.): *Automatic Speech and Speaker Recognition: Advanced Topics* (Kluwer Academic, Dordrecht 1996)

42.3 J.L. Gauvain, L. Lamel: Large-vocabulary continuous speech recognition: advances and applications, Proc. IEEE **88**(8), 1181–1200 (2000)

42.4 G. Salton: *The SMART Retrieval System* (Prentice-Hall, Englewood Cliffs 1971)

42.5 F. Sebastiani: Machine learning in automated text categorization, ACM Comput. Surv. **34**(1), 1–47 (2002)

42.6 S. Gao, W. Wu, C.-H. Lee, T.-S. Chua: A MFoM learning approach to robust multiclass multi-label text categorization, Proc. ICML (2004) pp. 42–49

42.7 S. Gao, B. Ma, H. Li, C.-H. Lee: A text-categorization approach to spoken language identification, Proc. Interspeech (2005) pp. 2837–2840

42.8 J.R. Bellegarda: Exploiting latent semantic information in statistical language modeling, Proc. IEEE **88**(8), 1279–1296 (2000)

42.9 T.J. Hazen: *Automatic Language Identification Using a Segment-based Approach* (MIT, Cambridge 1993), MS Thesis

42.10 K.M. Berkling, E. Barnard: Analysis of phoneme-based features for language identification, Proc. ICASSP (1994) pp. 289–292

42.11 K.M. Berkling, E. Barnard: Language identification of six languages based on a common set of broad phonemes, Proc. ICSLP (1994) pp. 1891–1894

42.12 C. Corredor-Ardoy, J.L. Gauvain, M. Adda-Decker, L. Lamel: Language identification with language-independent acoustic models, Proc. Eurospeech, Vol. 1 (1997) pp. 55–58

42.13 C.-H. Lee, F.K. Soong, B.-H. Juang: A segment model based approach to speech recognition, Proc. ICASSP (1988) pp. 501–504

42.14 A.K.V.S. Jayram, V. Ramasubramanian, T.V. Sreenivas: Language identification using parallel subword recognition, Proc. ICASSP (2003) pp. 32–35

42.15 J.L. Hieronymus: ASCII phonetic symbols for the world's languages: Worldbet, Technical Report AT&T Bell Labs (1994)

42.16 Y.K. Muthusamy, N. Jain, R.A. Cole: Perceptual benchmarks for automatic language identification, Proc. ICASSP (1994) pp. 333–336

42.17 B. Ma, H. Li, C.-H. Lee: An acoustic segment modeling approach to automatic language identification, Proc. Interspeech (2005) pp. 2829–2832

42.18 L.R. Rabiner: A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE **77**(2), 257–286 (1989)

42.19 H. Li, B. Ma, C.-H. Lee: A vector space modeling approach to spoken language identification, IEEE Trans. Audio Speech Language Process. **15**(1), 271–284 (2007)

42.20 H.K.J. Kuo, C.-H. Lee: Discriminative training of natural language call routers, IEEE Trans. Speech Audio Process. **11**(1), 24–35 (2003)

42.21 J. Chu-Carroll, B. Carpenter: Vector-based natural languagecall routing, Comput. Linguist. **25**(3), 361–388 (1999)

42.22 H. Li, B. Ma: A phonotactic language model for spoken language identification, Proc. ACL (2005) pp. 515–522

42.23 G.K. Zipf: *Human Behavior and the Principal of Least Effort, An Introduction to Human Ecology* (Addison-Wesley, Reading 1949)

42.24 K.S. Jones: A statistical interpretation of term specificity and its application in retrieval, J. Doc. **28**, 11–20 (1972)

42.25 C.-W. Hsu, C.-J. Lin: A comparison of methods for multiclass support vector machines, IEEE T. Neural Netw. **13**(2), 415–425 (2002)

42.26 V. Vapnik: *The Nature of Statistical Learning Theory* (Springer, Berlin, Heidelberg 1995)

42.27 A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardin, S. Levy: A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis, Bioinformatics **21**(5), 631–643 (2005)

42.28 J. Weston, C. Watkins: Multi-class support vector machines, Tech. Rep. CSD-TR-98-04 (University of London, London 1998)

42.29 Y. Lee, Y. Lin, G. Wahba: Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data, J. Am. Stat. Assoc. **99**(465), 67–81 (2004)

42.30 J.C. Platt, N. Cristianini, J. Shawe-Taylor: Large margin DAG's for multiclass classification, Advances in Neural Information Processing Systems, Vol. 12 (Cambridge, MIT Press 2000) pp. 547–553

42.31 S. Katagiri, C.-H. Lee: A New Hybrid Algorithm for Speech Recognition Based on HMM Segmentation and Discriminative Classification, IEEE Trans. Speech Audio Process. **1**(4), 421–430 (1993)

42.32 K.-Y. Su, C.-H. Lee: Speech Recognition using Weighted HMM and Subspace Projection Approaches, IEEE Trans. Speech Audio Process. **2**(1), 69–79 (1994)

42.33 M. Kobayashi, M. Aono: Vector space models for search and cluster mining. In: *Survey of Text Mining*, ed. by M.W. Berry (Springer, Berlin, Heidelberg 2003)

42.34 R.O. Duda, P.E. Hart, D.G. Stork: *Pattern Classification* (Wiley, New York 2001)

42.35 K. Crammer, Y. Singer: Improved Output Coding for Classification Using Continuous Relaxation, Proc. NIPS (2000) pp. 437–443

42.36 S. Haykin: *Neural Networks: A Comprehensive Foundation* (McMillan, London 1994)

42.37 J. Li, S. Yaman, C.-H. Lee, B. Ma, R. Tong, D. Zhu, H. Li: Language recognition based on score distribution feature vectors and discriminative classfier fusion, Proc IEEE Odyssey Speaker and Language Reognition Workshop (2006)

42.38 S. Katagiri, B.-H. Juang, C.-H. Lee: Pattern Recognition Using A Generalized Probabilistic Descent Method, Proc. IEEE **86**(11), 2345–2373 (1998)

42.39 J.L. Gauvain, A. Messaoudi, H. Schwenk: Language recognition using phone lattices, Proc. ICSLP (2004) pp. 1215–1218

42.40 E. Singer, P.A. Torres-Carrasquillo, T.P. Gleason, W.M. Campbell, D.A. Reynolds: Acoustic, phonetic and discriminative approaches to automatic language recognition, Proc. Eurospeech (2003) pp. 1345–1348

42.41 P.A. Torres-Carassquilo, E. Singer, M.A. Kohler, R.J. Greene, D.A. Reynolds, J.R. Deller Jr.: Approaches to language identification using Gaussian mixture models and shifted delta cepstral features, Proc. ICSLP (2002) pp. 89–92

42.42 B.P. Lim, H. Li, B. Ma: Using local and global phonotactic features in Chinese dialect identification, Proc. ICASSP (2005) pp. 577–580

42.43 P.A. Torres-Carrasquillo, D.A. Reynolds, R.J. Deller Jr.: Language identification using Gaussian mixture model tokenization, Proc. ICASSP (2002) pp. 757–760

42.44 T.G. Dietterich, G. Bakiri: Solving multiclass learning problems via error-correcting output codes, J Artif. Intell. Res. **2**, 263–286 (1995)

42.45 H. Li, B. Ma, R. Tong: Vector-Based Spoken Language Recognition using Output Coding, Proc. Interspeech (2006)

42.46 R. Tong, B. Ma, D. Zhu, H. Li, E.S. Chng: Integrating acoustic, prosodic and phonotactic features for spoken language identification, Proc. ICASSP **1**, 205–208 (2006)

42.47 W.M. Campbell, D.E. Sturim, D.A. Reynolds: Support vector machines using GMM Supervectors for speaker recognition, IEEE Signal Process. Lett. **13**(5), 308–311 (2006)

42.48 A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, A. Venkataraman: MLLR transforms as features in speaker recognition, Proc. Interspeech (2005) pp. 2425–2428

42.49 W.M. Campbell, J.P. Campbell, D.A. Reynolds, E. Singer, P.A. Torres-Carrasquillo: Support vector machines for speaker and language recognition, Comput. Speech. Lang. **20**(2-3), 210–229 (2005)