Speaker Verification Using Support Vector Machines and High-Level Features

William M. Campbell, *Member, IEEE*, Joseph P. Campbell, *Fellow, IEEE*, Terry P. Gleason, Douglas A. Reynolds, *Senior Member, IEEE*, and Wade Shen

***Abstract*—High-level characteristics such as word usage, pronunciation, phonotactics, prosody, etc., have seen a resurgence for automatic speaker recognition over the last several years. With the availability of many conversation sides per speaker in current corpora, high-level systems now have the amount of data needed to sufficiently characterize a speaker. Although a significant amount of work has been done in finding novel high-level features, less work has been done on modeling these features. We describe a method of speaker modeling based upon support vector machines. Current high-level feature extraction produces sequences or lattices of tokens for a given conversation side. These sequences can be converted to counts and then frequencies of n-gram for a given conversation side. We use support vector machine modeling of these n-gram frequencies for speaker verification. We derive a new kernel based upon linearizing a log likelihood ratio scoring system. Generalizations of this method are shown to produce excellent results on a variety of high-level features. We demonstrate that our methods produce results significantly better than standard log-likelihood ratio modeling. We also demonstrate that our system can perform well in conjunction with standard cesptral speaker recognition systems.**

***Index Terms*—Speaker recognition, support vector machines (SVMs), high-level features.**

I. Introduction

We consider the problem of text-independent speaker verification. The problem is, given a claim of identity and an utterance from a speaker, determine if the claim is true or false. This problem has traditionally been solved by directly modeling the spectral content (i.e., cepstral coefficients) of the speech with Gaussian mixture models [1].

An area of considerable development over the last six years has been the use of *high-level* features for speaker verification. One of the initial investigations in this area was Doddington's idiolect work [2]—word n-grams from conversation sides were used to characterize a particular speaker. More recent systems have used a variety of approaches involving phone sequences [3], pronunciation modeling [4] and prosody [5]–[7]. Standard methods for modeling the n-gram sequence include log-likelihood ratios [2] and tree methods [8]. An overview of features and methods is given in the Johns Hopkins Workshop summary paper [9].

In this paper, we provide an in-depth description of an alternate language modeling approach for high-level speaker recognition—support vector machines (SVMs). Work on this novel method originally appeared in the short conference papers [10], [11]. Our goal in this paper is to expand the discussion of this original work providing more details and examples. The focus of this paper is on the derivation and application of SVMs to model speaker information in token sequences. As examples, we use phone and word sequences. The utility of high-level feature analysis has been justified by its use in many applications—e.g., forensic phonetics [12], [13] and automatic fusion systems [14].

SVMs [15] are two-class classifiers that are trained using methods that attempt to improve generalization performance. Since speaker verification is also a two-class problem—a verification is a decision whether an utterance is or is not the target speaker—the SVM framework is a natural fit to the problem. A key feature of SVMs is that they map input features to a high-dimensional space (SVM feature space) and then perform classification. Although this high-dimensional expansion seems problematic due to the *curse of dimensionality* [16], the SVM training criterion deals effectively with the problem.

A key problem addressed in this paper is how to model high-level features with SVMs. In earlier work with sequence kernels [17], [18], we argued that a conversation side should be represented as a single vector in SVM feature space. Using this analogy for the current work, we assume that the n-gram statistics within a given conversation side are relatively stable, i.e., most of the variation is seen in different conversation sides at different times and is due to speaker session variability and channel effects. Therefore, it makes sense to view a conversation side as a single "document" of tokens and create one vector in SVM feature space for this document.

This paper is organized as follows. In Section II, we discuss the basic technique of high-level feature extraction and classification. In Section III, we discuss SVMs and their application to high-level speaker recognition. Section IV discusses a key problem: kernel construction for the SVM speaker recognition system. We discuss a kernel for SVM speaker recognition based upon a linearization of a likelihood ratio classifier. In Section V, we discuss extensions in the literature to our methods. In Section VI, we apply the techniques to phone and word high-level features. We show that significant improvements in performance over the standard log-likelihood ratio technique can be achieved. Finally, in Section VII, we fuse high-level systems and a cepstral SVM system in various configurations to illustrate tradeoffs.

Manuscript received February 15, 2007; revised May 11, 2007. This work was supported by the Department of Homeland Security under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the U.S. Government. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mari Ostendorf.

The authors are with the Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA 02420 USA (e-mail: wcampbell@ll.mit.edu).

Digital Object Identifier 10.1109/TASL.2007.902874

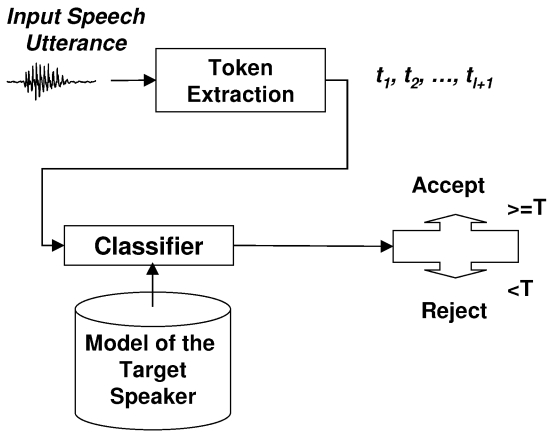

Fig. 1. High-level speaker recognition—basic setup.

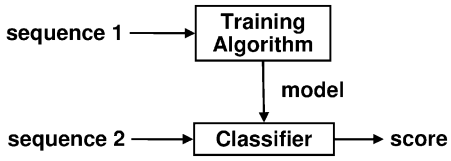

Fig. 2. Potential sequence kernel for speaker recognition.

II. High-Level Feature Extraction and Classification

The basic setup for our methods is shown in Fig. 1. From the input speech, we extract a series of tokens, $t_1, \ldots, t_{l+1}$. These tokens could be words, phones, pitch gestures, etc. The tokens are a discrete representation of the input speech; the representation is typically meaningful in a high-level linguistic sense. After the tokens have been extracted, we perform classification with the target speaker model. The classification process produces a score which can then be used for an accept (score above the threshold, T) or reject decision (score below the threshold, T).

Several issues arise from Fig. 1. First, generation of high-level tokens is a difficult problem. There are many decisions that must be considered: which token type best distinguishes speakers, which token type best complements standard cepstral systems in a fusion sense, is accuracy or consistency of the token stream the most important, is interpretability of the features necessary, etc. Although we do not address this question in detail since this is not the main focus of this work, we do provide some insight into the utility of high-level features in Section VII. For word sequences, we use the BBN Byblos large-vocabulary recognizer; see, for example, [19]. For phone recognition, we use the phone tokenizers derived from the parallel phone recocognition language modeling (PPRLM) system [20]. These token types will give a good idea of the generality of our approach and also highlight potential issues.

A second important question is—how should we model the token stream with a classifier? The standard method of modeling the token stream presented in [2] is to model the probabilities of n-grams using a likelihood ratio. The likelihood ratio is the ratio of the probability that the token sequence was generated by the target speaker to the probability that the sequence was generated by an impostor using n-gram distributions. Our proposed method of modeling uses SVMs instead of likelihood ratios to model the target speaker. We examine this approach in detail in the next section.

III. SVMs for High-Level Speaker Recognition

A. Support Vector Machines

An SVM is a two-class classifier constructed from sums of a kernel function $K(\cdot,\cdot)$

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{1}$$

where the y_i are target values, $\sum_{i=1}^{N} \alpha_i y_i = 0$, and $\alpha_i > 0$. The vectors $\mathbf{x}_i$ are support vectors and obtained from the training set by an optimization process [21]. The target values are either 1 or -1, depending upon whether the corresponding support vector is in class 1 or class 2. For classification, a class decision is based upon whether the value $f(\mathbf{x})$ is above or below a threshold.

The kernel $K(\cdot,\cdot)$ is constrained to have certain properties (the Mercer condition), so that $K(\cdot,\cdot)$ can be expressed as

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{b}(\mathbf{x})^t \mathbf{b}(\mathbf{y}) \tag{2}$$

where $\mathbf{b}(\mathbf{x})$ is a mapping from the input space (where $\mathbf{x}$ lives) to a possibly infinite dimensional space. The output space of $\mathbf{b}(\mathbf{x})$ is typically called (SVM) feature space.

Training of the SVM in (1) is based upon the maximum margin criterion. This criterion results in a quadratic programming problem. Many SVM optimization tools are now available; e.g., SVMTorch [21] or LIBSVM [22].

B. Sequence Kernels

From Fig. 1, the high-level feature generation produces a sequence of data. A challenge in applying SVMs in this setting is that kernels have traditionally been designed for single instance data. That is, the kernel usually compares two vectors of data. A solution to this problem is to use sequence kernels.

Sequences kernels compare entire sequences of data; i.e., we compute $K(X, Y)$, where X and Y are sequences of data. Since SVM training and scoring involves only kernel computations, single instance methods generalize to this setting in a straightforward manner. The main challenge in extending SVMs to a sequence setting is to design appropriate sequence kernels.

Several sequence kernels have been proposed in the literature for continuous and discrete data [11], [17], [23], [24]. We present in detail the method proposed in [10] and [11]. Our basic strategy is shown in Fig. 2. The idea is to form a comparison between two sequences based upon training and testing with a classifier [17]. We use sequence 1 to train a classifier and obtain a model. The model is then used in a classifier to score sequence 2. The output score is then nominally the kernel output.

Several issues arise with using the strategy in Fig. 2 as a kernel. First, the comparison may not generate a symmetric function, $K(x, y) \neq K(y, x)$. Second, the output score may not satisfy the Mercer condition. Practically, this means that the kernel does not correspond to high-dimensional expansion, $\mathbf{b}(\mathbf{x})$, so the standard theory of SVMs does not apply. Finally,

we must choose a classifier in Fig. 2. We will deal with all of these issues in Section IV where we construct a kernel.

One ambiguous aspect of the comparison strategy in Fig. 2 is the source of the sequences. In many speaker recognition scenarios (e.g., NIST speaker recognition evaluations [25]), speech from a speaker is divided into different conversation sides or utterances. One strategy is to generate a single sequence for a speaker by combining token streams from all conversation sides for the given speaker. This method is problematic since it does not capture session variability due to speaker effects, channel type, environment, etc. An alternate strategy is to consider each conversation side a separate sequence; so, for instance, if the speaker has six utterances, we have six sequences describing that speaker. We view each sequence as a *document* of tokens. This analogy allows us to explore connections with related work in information retrieval [26].

IV. Kernel Construction

In order to implement the kernel proposed in Fig. 2, we use a classifier based upon the likelihood ratio of target probability to impostor probability [2]. This approach is quite similar to a naive Bayes classifier [27] to model the sequences, since we are using an independence assumption.

Our first step of kernel construction is the selection of probabilities to describe the token sequence. We follow the work of [2] and [3] and use a *bag of* n-grams approach. For a token sequence, we produce n-grams by the standard transformation of the stream. For bigrams, for example, the sequence of tokens from a conversation side, $t_1, t_2, \ldots, t_{l+1}$, is transformed into the sequence of bigrams of tokens $x_1 = t_1\_t_2, x_2 = t_2\_t_3, \ldots, x_l = t_l\_t_{l+1}$. We then find probabilities of n-grams. That is, suppose we are considering bigrams of tokens and the unique bigrams are designated $d_1, \ldots, d_M$, then we calculate probabilities

$$p(d_j|X) = \frac{\#(x_i = d_j|X)}{\sum_k \#(x_i = d_k|X)} \tag{3}$$

where $\#(x_i = d_j|X)$ indicates the number of tokens in the sequence X equal to d_j.

A. Log-Likelihood Ratio Weighting

Suppose that we have two conversation sides from two speakers. Further suppose that the sequence of n-grams (for fixed n) in each conversation side is $X = x_1, x_2, \ldots, x_l$ and $Y = y_1, y_2, \ldots, y_m$, respectively. Also, assume that we have a large *background* corpus of tokens B that represents many speakers. We denote the unique set of n-grams in the background as $d_1, \ldots, d_M$. We can build a model based upon the conversation sides for each speaker consisting of the probability of n-grams, $p(d_i|X)$ and $p(d_i|Y)$. We then compute the likelihood ratio of the first conversation side as in standard verification systems [1]; a linearization of the likelihood ratio computation will serve as the kernel. Proceeding

$$\frac{p(X|Y)}{p(X|B)} = \frac{p(x_1, \ldots, x_l|Y)}{p(x_1, \ldots, x_l|B)} = \prod_{i=1}^{l} \frac{p(x_i|Y)}{p(x_i|B)} \tag{4}$$

where we have made the assumption that the probabilities are independent. We then consider the log of the likelihood ratio normalized by the number of observations

$$\begin{aligned} \text{score} &= \frac{1}{l} \sum_{i=1}^{l} \log \left(\frac{p(x_i|Y)}{p(x_i|B)} \right) \\ &= \sum_{j=1}^{M} \frac{\#(x_i = d_j|X)}{l} \log \left(\frac{p(d_j|Y)}{p(d_j \,|\, B)} \right) \\ &= \sum_{j=1}^{M} p(d_j|X) \log \left(\frac{p(d_j|Y)}{p(d_j|B)} \right). \end{aligned} \tag{5}$$

The score (5) is not a kernel, since it is not symmetric with respect to the role of the two sequences. We could symmetrize by reversing the roles of the two sequences and averaging the scores, but this still does not create a kernel—the resulting kernel matrix is not necessarily positive definite.

Many solutions are available for deriving a kernel from the pseudokernel in (5). One possibility is to form a linear approximation of the pseudokernel function. This process can be justified as follows. Any kernel induces a distance by the following identity

$$D(x,y)^2 = K(x,x) + K(y,y) - 2K(x,y). \tag{6}$$

The distance $D(x,y)$ is used to determine the SVM hyperplane via the maximum margin criterion. If we have an approximation to $D(x,y)$ that is accurate in a local region around the margin, then we can correctly pick support vectors and have appropriate values of α_i in (1); i.e., the main role of the kernel is to provide a method for picking support vectors and determining the maximum margin—a globally accurate distance metric is not needed.

Proceeding with this strategy, we linearize the log function in (5) by using $\log(x) \approx x - 1$ (Taylor series expansion around $x = 1$). We get

$$\begin{aligned} \text{score} &\approx \sum_{j=1}^{M} p(d_j|X) \frac{p(d_j|Y)}{p(d_j|B)} - \sum_{j=1}^{M} p(d_j|X) \\ &= \sum_{j=1}^{M} p(d_j|X) \frac{p(d_j|Y)}{p(d_j|B)} - 1 \\ &= \sum_{j=1}^{M} \frac{p(d_j|X)}{\sqrt{p(d_j|B)}} \frac{p(d_j|Y)}{\sqrt{p(d_j|B)}} - 1. \end{aligned} \tag{7}$$

Analogous to the information retrieval literature [26], we can express the result in (7) in vector form. First, we construct a vector $\mathbf{v}$ describing the conversation side

$$\mathbf{v}_X = [\, p(d_1|X) \quad \ldots \quad p(d_M|X) \,]^t. \tag{8}$$

A similar vector can be constructed from Y. In general, the vector $\mathbf{v}$ will be sparse since the conversation side will not contain all potential unigrams, bigrams, etc. The entries of $\mathbf{v}$ are then weighted with a diagonal matrix D with entries $D_{j,j} =$

$1/\sqrt{p(d_j|B)}$. The final kernel, rewritten from (7) and ignoring the constant, is an inner product

$$K(X,Y) = (D\mathbf{v}_X)^t(D\mathbf{v}_Y). \tag{9}$$

Analogous to the information retrieval literature (discussed below), we call (9) a term frequency LLR (TFLLR) kernel.

B. Relations to Information Retrieval and TFIDF

In standard information retrieval methods, frequencies of words in a document are represented in vector form as a product of a term frequency and a collection component [26]. The term frequency $\mathrm{TF}(d_i)$ is the number of occurrences of the word d_i in a document (a count, not a probability). The collection component is some measure of the commonality of the word in all the documents in the collection. Finally, after representing a document as a vector, the vector is scaled to a Euclidean (2-norm) of one.

A common collection component is the *inverse document frequency* (IDF). If we let $\mathrm{DF}(d_i)$ be the number of conversation sides where a particular n-gram, d_i, is observed, then a TFIDF representation has entries

$$\mathrm{TF}(d_i)\log\left(\frac{\#\text{ of conversations in background}}{\mathrm{DF}(d_i)}\right). \tag{10}$$

The kernel in (7) is very similar to methods used in the information retrieval literature with some subtle differences. First, we are using probabilities rather than counts for the vector entries in (8). This is basically an alternate normalization (1-norm) to using the Euclidean norm. Second, the weighting term for (7) is based upon the probability of an n-gram in the background. Standard IDF only measures if a word does or does not occur in a particular document.

Our intent in discussing information retrieval is twofold. First, we want to illustrate that our linearization of the likelihood ratio leads to a similar approach used in information retrieval. Second, we can borrow ideas from information retrieval to improve our approach.

C. Generalizations and Discussion

The kernel in (7) can be generalized in a straightforward way. There are several motivations for this generalization.

First, in (7), the background weighting term can become quite large if a given n-gram occurs infrequently in a document. In this case, the probability estimate of $p(d_j|B)$ may be inaccurate or underestimated. We would like to limit the influence of any one entry in the vector $\mathbf{v}$ in (9).

Second, we can rewrite the diagonal terms in (9) as

$$D_{j,j} = \sqrt{\frac{1}{p(d_j|B)}}. \tag{11}$$

The role of the square root function is to squash the dynamic range; i.e., it is monotone increasing and maps $[1,\infty]$ to itself.

Combining the previous two motivations, it is natural to use a diagonal weight of

$$D_{j,j} = \min\left(C_j, g_j\left(\frac{1}{p(d_j|B)}\right)\right) \tag{12}$$

where $g_j(\cdot)$ is a function that squashes the dynamic range.

Several points can be made about this generalization. First, for $g_j(x) = \sqrt{x}$, we obtain our TFLLR kernel. Second, similar to TFIDF, we can use $g_j(x) = \log(x) + 1$; we refer to this weighting as a TFLOG kernel. Third, an extreme case for (12) is to set $C_j = 1$. This results in weighting all terms equally, $D_{j,j} = 1$, for all j. Fourth, we note that both C_j and the function g_j are dependent on j. This variable weighting may make sense if we have some idea of the confidence of our estimates in the probabilities of n-grams in the background data set.

D. Implementation

Several implementation tricks can simplify the use of an SVM classifier in training and scoring. First, the SVM model of a speaker can be simplified if we combine all support vectors in (1). If we assume we have the SVM expansions $\mathbf{v}_i$ of the support vectors (i.e., the weighted versions of (9)), then

$$\begin{aligned} f(\mathbf{v}) &= \sum_{i=1}^{N} \alpha_i y_i \mathbf{v}_i^t \mathbf{v} + b \\ &= \mathbf{v}^t \left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{v}_i\right) + b \\ &= \mathbf{v}^t \mathbf{w} + b \end{aligned} \tag{13}$$

where $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{v}_i$. Depending on the sparsity structure of the individual support vectors, $\mathbf{w}$ may require less memory than storage of all of the support vectors. Also, using a single model $\mathbf{w}$ may reduce computation in scoring.

For SVM training, we typically use a fixed set of utterances that represent the -1 class in training; we call this a *background* set. The background should represent a typical set of impostors that the system would see in practice. For our experiments with telephone data for the NIST speaker recognition evaluations, several thousand utterances from a corpus such as Fisher [28] are used. The tokens, n-gram probabilities, and corresponding vectors (8) are precomputed and stored. A dictionary of possible n-grams is constructed from this background and determines the ordering of entries in the vector (8). The diagonal weighting in (9) is also determined from the background.

Training a speaker model then is performed in the following way. We extract tokens for each conversation side from a speaker, find n-gram probabilities, vectorize (8), and then weight with a function of the background probabilities. The speaker vectors are appended to the background set of vectors and training is performed with a linear kernel. A model is then formed using the model compaction in (13).

The SVM approach does not attempt to smooth the probabilities for a conversation side (3) as is common in language modeling. There are at least three reasons for this approach. First, SVMs perform a form of smoothing by selecting support vectors from the background and selectively combining these with the enrollment data—this forms a "smoothed" model. Second, smoothing tends to destroy the sparsity structure of the SVM kernel, increasing computation for training and scoring. Third, smoothing may destroy speaker-specific information critical for discrimination.

V. Extensions to SVM Methods

Several extensions to the SVM method of high-level speaker recognition have been proposed and implemented. These methods can improve performance, provide alternate implementation strategies, or provide additional information to a user. We briefly summarize this work. We only incorporate the lattice methods in our experiments in Section VI, since a full exploration of these techniques is beyond the scope of this paper.

A first extension is the use of *lattices* in place of 1-best token sequences [29]. Many recognition systems (e.g., phone or word) can produce a lattice $\mathcal{L}$ of possible alternative token hypotheses. From this lattice, expected counts of n-grams can be calculated. These expected counts can be used in place of 1-best counts; i.e., in (3), instead of using counts, expected counts from a lattice can be used to estimate probabilities. One can then use the kernel in (9) with an appropriate weighting.

A second extension to our approach is to use nuisance attribute projection (NAP) to reduce the variability of the token extraction under session and channel effects [30], [31]. The SVM NAP method [31] works by removing subspaces that cause variability in the kernel. This variability reduction, in turn, makes the corresponding distance (6) more accurately reflect between-speaker distances. NAP constructs a new kernel

$$\begin{aligned} K(X,Y) &= [\mathbf{P}D\mathbf{v}_x]^t\,[\mathbf{P}D\mathbf{v}_y] \\ &= \mathbf{v}_x^t D\mathbf{P}D\mathbf{v}_y \\ &= \mathbf{v}_x^t D(\mathbf{I}-\mathbf{u}\mathbf{u}^t)D\mathbf{v}_y \end{aligned} \tag{14}$$

where $\mathbf{P}$ is a projection ($\mathbf{P}^2=\mathbf{P}$), $\mathbf{u}$ is the direction being removed from the SVM expansion space, $\mathbf{v}_x$ and $\mathbf{v}_y$ are the SVM expansions, and $|\mathbf{u}|_2=1$. NAP has been shown to compensate some feature types (e.g., errorful phone transcripts) with dramatic results [30].

Another extension of our methods is in the area of interpretation of the significance of high-level features for speaker characterization. In [32], methods are discussed for determining which n-grams are significant for identifying the speaker. For instance, is the speaker characterized by low usage of the bigram "I know?" In this context, the weighting (12) can be seen as a normalization for typicality of a feature in a population. This interpretation framework is one attempt at finding relations between work in automatic speaker recognition and forensic phonetics [12], [13].

VI. Experiments

In this section, we demonstrate applications of the SVM methods to typical high-level features. We perform experiments on standard corpora and contrast our approach to a simple language modeling approach. Our experimental goal is to demonstrate the considerations in applying an SVM to high-level features; we are not trying to produce fully optimized high-level feature systems.

We explore two distinct types of features-open loop phone recognition and words from a speech-to-text (STT) system. These feature types contrast well and show different aspects of the SVM system. Phones are a higher rate token sequence with a lower number of distinct tokens (around 40). This results in dense expansion space vectors that have stable statistics. In contrast, STT word sequences are lower rate, have a large number of tokens, and result in very large sparse expansions.

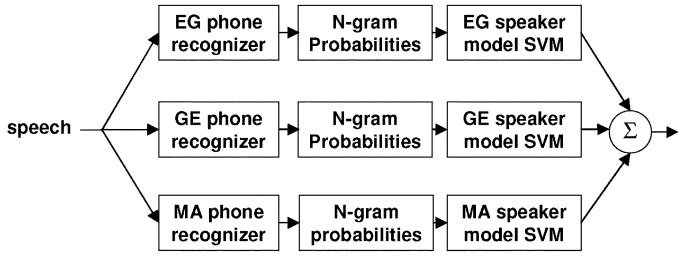

Fig. 3. Phonetic speaker verification using SVMs.

A. SVM Phone System

Our system for speaker verification using phone sequences is shown in Fig. 3. A test utterance is input to the system, and phones are found. The phone sequences are found using multiple recognizers in English (EG), German (GE), and Mandarin (MA). For each language, n-gram probabilities (3) are found for each output stream. Then, the probabilities are vectorized as in (8) using a sparse representation, i.e., we store the indices and probabilities of nonzero entries only. Next, probabilities are weighted using TFLLR as in (9) and (11). Finally, a score is computed per language using a linear kernel, and then the scores are fused by a weighted linear combination.

An interesting aspect of the system in Fig. 3 is that it uses multiple streams of phones in different languages. There are several reasons for this strategy. First, the system can be used without modification for speakers in multiple languages. Second, although not obvious, from experimentation we show that phone streams different from the language being spoken provide complementary information for speaker verification. That is, accuracy improves with these additional systems. A third point is that the system may also work in other languages not represented in the phone streams. It is known that in the case of language identification, language characterization can be performed even if a phone recognizer is not available in that particular language [20].

B. Phone Sequence Extraction

Phone sequence extraction for the speaker verification process is performed using the phone recognition system from the PPRLM language identification system [20]. PPRLM uses a Mel frequency cepstral coefficient (MFCC) front end with appended delta coefficients. Each phone is modeled in a gender-independent context-independent (monophone) manner using a three-state hidden Markov model (HMM). Phone recognition is performed with a Viterbi search using a fully connected null-grammar network of monophones.

The phone recognizers were trained using the OGI multilanguage corpus that was annotated by native speakers [33]. The phone error rate for the English tokenizer was reported by Zissman as approximately 58%. Although exact error rates for the Mandarin and German recognizers are not available, Zissman reports error rates on this type of phone recognizer in the 45%–58% range [20].

For a given input utterance, speech activity detection was performed with an energy-based detector. The speech was then split

into 5–10 s segments, and lattices were extracted using HTK [34]. Both the 1-best output and expected counts from the lattice were calculated using the SRI language modeling toolkit [35].

C. Phone Experiments Corpus

We applied the SVM phone system to the NIST 2005 Speaker Recognition Evaluation SRE corpus which was drawn from the Mixer corpora [36]. We experimented with the NIST 2005 SRE eight conversation side train, one conversation side test task using only the core English and handset data. This resulted in 1672 true trials and 14 406 false trials.

The background for the experiments was constructed from the Fisher corpus. Note that the corpus contained some non-English data (Chinese and Arabic), since the NIST SRE 2005 evaluation included a multilingual component. The resulting background had 4171 conversation sides.

Each conversation side in the NIST SRE dataset is nominally 5 min in length and is recorded over a telephone channel. Each conversation side consisted of a speaker having a conversation on a topic selected by an automatic operator; conversations were typically between unfamiliar individuals.

D. SVM Phone Training

Training for the system in Fig. 3 is based upon the the NIST SRE task definition. We treat each conversation side in the corpus as a document. From each of these conversation sides, we derive a single (sparse) vector of weighted probabilities. To train a model for a given speaker, we use a one-versus-all strategy. The speaker's conversation sides are trained to an SVM target value of $+1$. The conversation sides not in the current split are used as a background and trained to an SVM target value of -1. Note that this strategy ensures speakers used as impostors are unseen in the training data.

E. Results for the Phone Experiments

Scoring was performed using the SVM system shown in Fig. 3. Equal weighting of the speaker model scores from the three language tokenizers was used. Note that for a given n-gram order, we include all n-grams less than or equal to that order in the SVM expansion vector. Also, note that when an n-gram did not appear in the background, it was ignored in training and scoring. Training was performed using the SVMTorch [21] with a margin and error tradeoff of $c = 1$ [15].

Results were compared via DET curves [37] and equal error rates (EERs). We initially explored the use of different configurations for decoding and n-gram order for the English tokenizer. Results are shown in Fig. 4. In the figure, 1-best indicates that n-gram counts were derived from the 1-best decode; lattice indicates that expected counts from a lattice were used. From the figure, we see that the lattice configuration outperforms the 1-best configuration for bigrams. Also, the trigram configuration outperforms the bigram configuration. Higher order n-grams were not found to be beneficial.

We next considered the effect on performance of the language of the phone stream for the eight conversation side training case. Fig. 5 shows a DET plot with results corresponding to three language phone streams using the TFLLR kernel, lattice expected counts, and all n-grams up to trigram. A linear fusion of the SVM outputs was performed with equal weighting of the three scores, and this result is also shown in the figure. As expected, the best performing system is the fusion system. The best performing single language system is English; presumably, this is due to the match between the SRE task language (English) and the phone recognizer language.

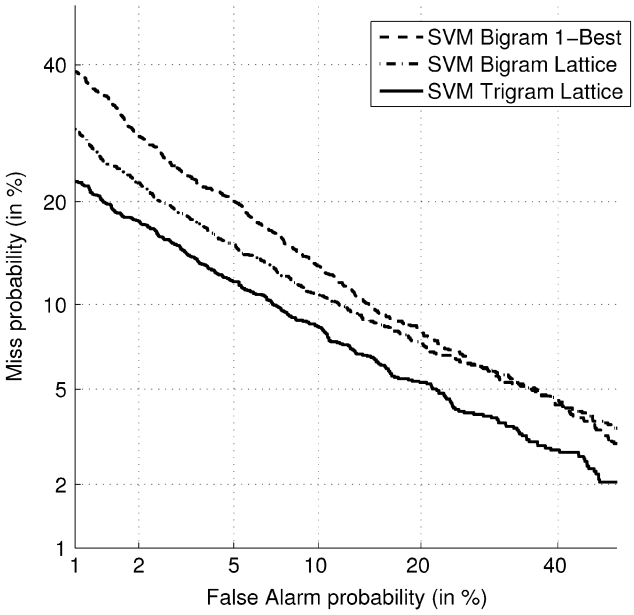

Fig. 4. Results for different configurations of the English recognizer on the NIST SRE 2005 using the SVM phone system.

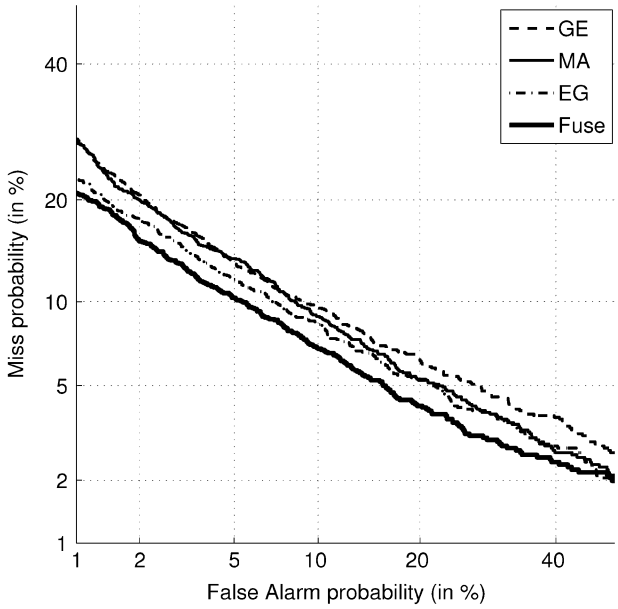

Fig. 5. Results for languages and a fusion on the NIST SRE 2005 dataset with an SVM phone system, TFLLR weighting, all n-grams to order 3, and lattice expected counts.

Fig. 6 shows a comparison of the fusion of the scores from the non-English branches of the phone system in Fig. 5 with the fusion of all scores from all languages. This figure gives an example of how the SVM system can generalize to unseen languages, as discussed in Section VI-A. Even though we use

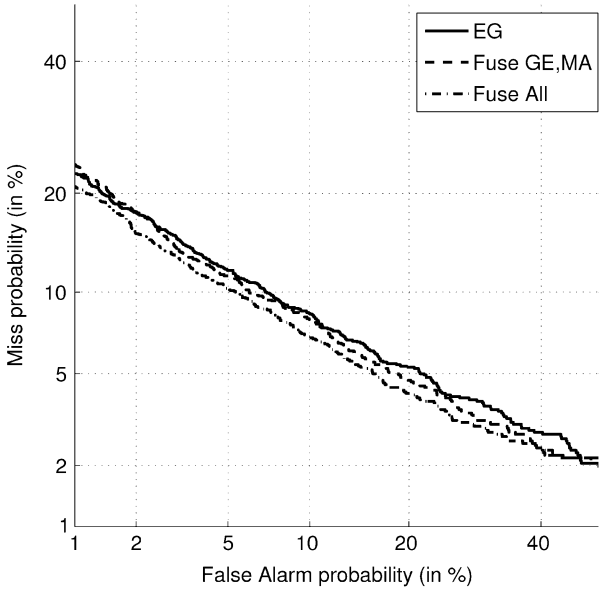

Fig. 6. Comparison of results for the SVM phone system with non-English language scores fused versus English-only and all languages fused.

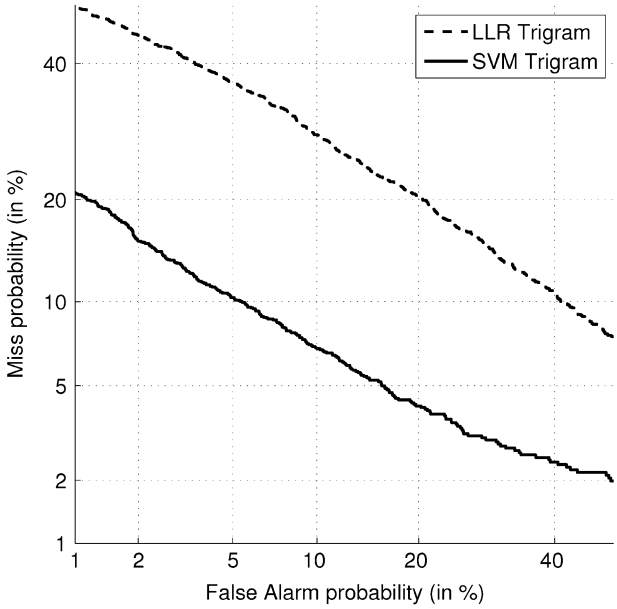

Fig. 7. DET plot for eight conversation side training showing a comparison of the SVM approach (solid line) to the standard log-likelihood ratio approach using trigrams (dashed line).

non-English tokenizers on English data, we obtain similar performance to a system having an English tokenizer. Note that similar results were obtained in [3].

Fig. 7 shows DET plots comparing the performance of the standard n-gram log-likelihood ratio (LLR) method [3] to our SVM method using the TFLLR weighting. The n-gram LLR system computes the LLR between a smoothed target speaker model and a background model as in (5). The target speaker model is smoothed with the background as in [29]; i.e., for a given n-gram, d_i

$$p_s(d_i|\text{spk}) = (1-\alpha)p(d_i|\text{spk}) + \alpha p(d_i|\text{bkg}) \qquad (15)$$

where p_s is the smoothed model. The LLR system was optimized for best performance. A smoothing constant of $\alpha = 0.05$ was selected based upon prior work [29]. T-norm was applied to the n-gram system using speakers from the Switchboard 2 corpora. Linear equally weighted fusion of T-norm scores from the three separate language tokenizers was used to improve performance (as for the SVM in Fig. 5). Also, fusion of lower order n-grams with the trigram LLR system was examined, but no added improvement was obtained. Fig. 7 shows that the SVM system significantly outperforms the LLR system with minimal tuning.

Table I summarizes the various phone experiments reported in this section. In the table, fusion in the language column indicates a linear equally weighted fusion of multiple language speaker recognition systems. The best performing configuration is an SVM with trigrams, three-language fusion, and lattice-based expected counts.

TABLE I
COMPARISON OF EERS FOR DIFFERENT MODELS AND CONFIGURATIONS FOR PHONETIC SPEAKER RECOGNITION ON THE NIST SRE 2005 EIGHT CONVERSATION TASK

Model	Decoding	n-gram order	Language	EER (%)
SVM	1-best	2	EG	11.75
SVM	Lattice	2	EG	10.54
SVM	Lattice	3	EG	8.86
SVM	Lattice	3	Fuse	7.95
LLR	Lattice	3	Fuse	20.21

F. SVM Word System Experiments

The word-based SVM speaker recognition system uses the same structure as one branch of the phone system shown in Fig. 3. The input acoustic test utterance is converted to an English word sequence. Probabilities of n-grams for the word sequence are then found (1-best decoding). These probabilities are vectorized, weighted, and then scored with an SVM. The resulting output score is compared to a threshold, and an accept or reject decision is made.

G. Word Sequence Extraction

Transcripts for conversation sides were extracted using BBN's Byblos 1xRT recognizer trained with 2000+ hours of telephone speech [38]. The word error rate of this system on the 2004 Rich Transcription evaluation English conversational telephone speech data is 19.8% [38]. Audio files were first segmented into chunks of 15 s or less using a two-class HMM (speech/nonspeech) trained on a small selection (approximately 4 h) of Switchboard II and Fisher data. One-best decode was performed for each segment using Byblos with SCTM + VTLN and HLDA adaptation.

H. Experiments

We applied the SVM word system to the NIST 2005 SRE corpus with the same test set as the SVM phone system; see Section VI-A. The SVM background for the experiments was constructed from the Fisher corpus.

We first considered the optimization of the SVM word kernel. Because word frequency is very low in a given conversation

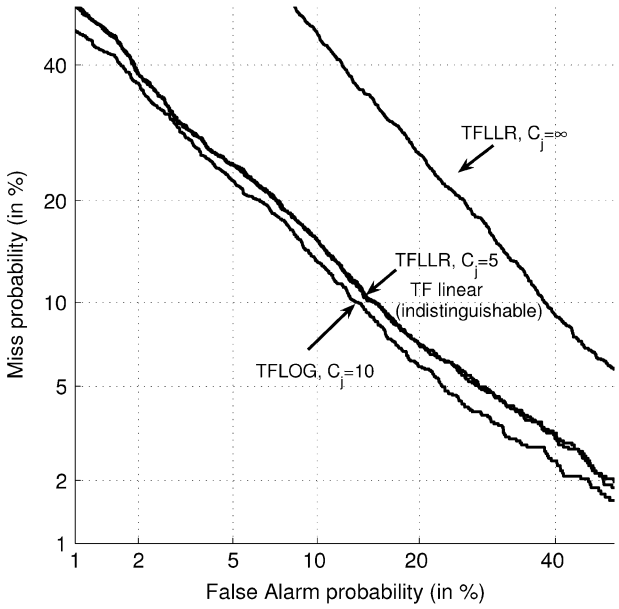

Fig. 8. Comparison of different weighting types for an SVM Word system.

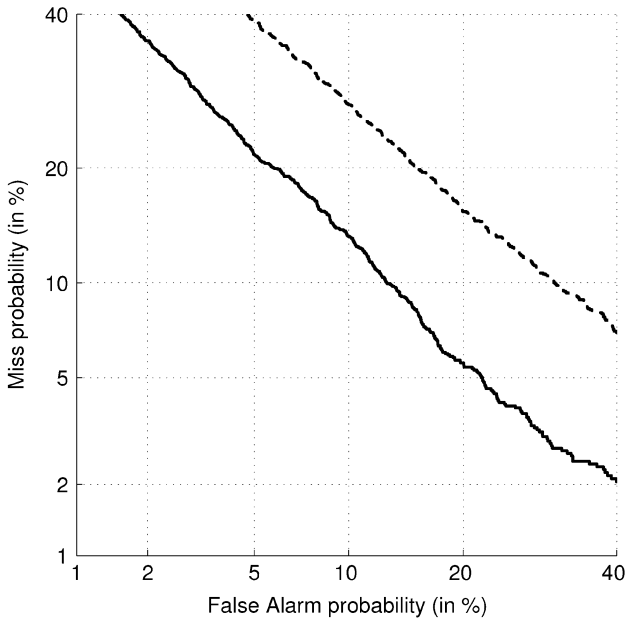

Fig. 9. DET plot for eight conversation side training showing a comparison of the SVM approach (solid line) to the standard n-gram log likelihood ratio approach (dashed line) using word bigrams.

TABLE II
COMPARISON OF DIFFERENT WEIGHTING STRATEGIES FOR THE NIST SRE 2005 TASK WITH AN SVM WORD SYSTEM

Weighting Method	C_j	EER
Linear	1	12.38%
TFLLR	5	12.33%
TFLLR	10	12.07%
TFLLR	∞	22.67%
TFLOG	10	11.55%
TFLOG	∞	11.65%

side, using the weighting proposed in (9), where the diagonal matrix has square root entries, was problematic. Low-frequency words in the background resulted in large weights. Therefore, we experimented with the generalization given in (12). The results are shown in Fig. 8.

Fig. 8 shows that choice of weighting is significant. If $C_j = \infty$ and the squashing function is an inverse square root (the TFLLR weighting), then the system performs poorly. Probabilities of infrequent words dominate the kernel function. For the other extreme case of $C_j = 1$ in (12), a linear kernel results (labeled as TF linear). This method shows considerable improvement over no weighting, but is not the best strategy. Other strategies of combinations C_j and using $g_j(x) = \log(x) + 1$ are also shown. Table II shows a comparison of the EERs for the different strategies. Overall, the best method we found was to use a log squashing function, TFLOG, and a $C_j = 10$.

The choice of TFLOG as the best performing weighting is reasonable since the dynamic range of $p(d_i|B)$, where B is the background, will be large because of infrequent bigrams and common unigrams. TFLOG will signficantly squash large values of $1/p(d_i|B)$, while approximately preserving smaller values of $1/p(d_i|B)$. This property will produce a kernel inner product that is not too sensitive to any single entry.

For comparison with the SVM system, we constructed a word log-likelihood ratio system similar to the one in [2] using unigrams and bigrams. We trained a background model using the Fisher data. Target models were smoothed to their background counterparts with an empirically derived weighting (0.8 optimized from dev-set trials). Bigrams with background model counts less than a fixed threshold (1.0) were rejected for scoring purposes. Using counts of bigrams extracted from 1-best word recognition, we scored each trial message against a target model and a background model and then computed the log-likelihood ratio.

Fig. 9 shows a comparison of the SVM system with the log-likelihood ratio system. Again, our best SVM system significantly outperforms a log-likelihood ratio system.

As with the SVM phone system, lattices can be used instead of 1-best decoding for estimating probabilities for the SVM Word system. Using lattices with TFLOG and $C_j = 10$ results in an EER of 11.06%. This gain is less dramatic than the SVM phone system, probably due to the higher accuracy of the STT system and the more careful pruning of lattices performed by this system.

VII. FUSION

In this section, we consider fusion of various combinations of the high-level systems and a cepstral SVM system. The cepstral SVM system is based on the generalized linear discriminant sequence (GLDS) kernel [18]. The front-end processing for this system uses the following parameters—20-ms frame size, 100 frames per second, preemphasis, Hamming windowing, Mel-frequency filterbanks, and an adaptive energy-based speech detector. From the filterbanks, 19 MFCC coefficients and the corresponding delta features are extracted, resulting in a 38-dimensional representation. Both RASTA and cepstral variance normalization are applied. The features are expanded in a GLDS kernel using all monomials up to degree 3. This results in an

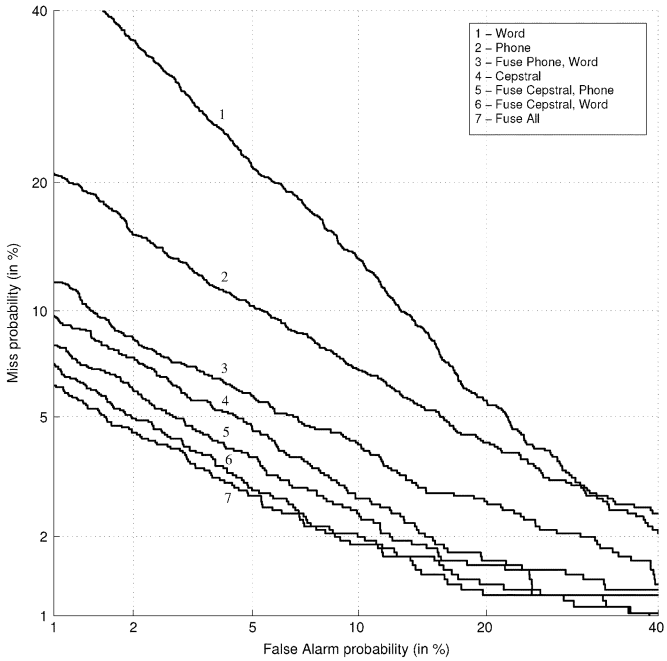

Fig. 10. DET plot of various fusions of high-level and cepstral systems on the NIST SRE 2005 task.

TABLE III
EERs FOR VARIOUS FUSION COMBINATIONS

Fused Systems	EER (%)
Word	11.51
Phone	7.95
Word, Phone	5.54
Cepstral	4.79
Cepstral, Phone	4.09
Cepstral, Word	3.73
All	3.43

SVM expansion space of dimension 10 660. A diagonal approximation is used in the kernel inner product. More details on this SVM cepstral system can be found in [18].

We fused SVM systems from Section VI with the cepstral SVM system using linear equal weighting of scores. This fusion strategy was used since it has proved robust and does not require a cross-validation training set. The SVM phone system was based on lattice decoding and expected counts, a TFLLR weighted kernel, trigram modeling, and a linear equal fusion of the three scores from the separate language streams. The SVM word system was based on 1-best decoding and TFLOG weighting.

Results for the various fusions are given in Fig. 10 and Table III. Several interesting results can be observed. First, the best system is a fusion of all high-level systems and the cepstral system. Second, we can see that the word system is more complementary to the cepstral system than the phone system. This feature is interesting since the phone system has a lower error rate than the word system, i.e., the best combination is not always achieved by picking the lowest error rate systems. Third, the word and the phone systems fuse well. In a certain sense, this reinforces the impression that the phone system acts as a "low resolution" cepstral system.

VIII. CONCLUSION

We have described a method of modeling high-level features in speaker recognition using an SVM. The method uses frequencies of n-grams in speaker conversation sides in a vector representation. Kernel inner product weighting was introduced based on a linearization of the log-likelihood ratio. Generalizations of this weighting were introduced to deal with infrequent n-grams. Application of these methods to phone and word high-level features illustrated the effectiveness of the methods. Fusion with a cepstral SVM system illustrated the potential of system combination.

REFERENCES

[1] D. A. Reynolds, T. F. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Process.*, vol. 10, no. 1–3, pp. 19–41, 2000.
[2] G. Doddington, "Speaker recognition based on idiolectal differences between speakers," in *Proc. Eurospeech*, 2001, pp. 2521–2524.
[3] W. D. Andrews, M. A. Kohler, J. P. Campbell, J. J. Godfrey, and J. Hernandez-Cordero, "Gender-dependent phonetic refraction for speaker recognition," in *Proc. ICASSP*, 2002, pp. I149–I153.
[4] D. Klusáček, J. Navarátil, D. A. Reynolds, and J. P. Campbell, "Conditional pronunciation modeling in speaker detection," in *Proc. ICASSP*, 2003, pp. IV-804–IV-807.
[5] K. Somnez, E. Shriberg, L. Heck, and M. Weintraub, "Modeling dynamic prosodic variation for speaker verification," in *Proc. ICSLP*, 1998, pp. 3189–3192.
[6] A. Adami, R. Mihaescu, D. A. Reynolds, and J. J. Godfrey, "Modeling prosodic dynamics for speaker recognition," in *Proc. ICASSP*, 2003, pp. IV-788–IV-791.
[7] S. Kajarekar, L. Ferrer, K. Sonmez, J. Zheng, E. Shriberg, and A. Stolcke, "Modeling NERFs for speaker recognition," in *Proc. Odyssey 2004: Speaker Lang. Recognition Workshop*, 2004, pp. 51–56.
[8] J. Navrátil, Q. Jin, W. D. Andrews, and J. P. Campbell, "Phonetic speaker recognition using maximum-likelihood binary-decision tree models," in *Proc. ICASSP*, 2003, pp. IV-796–IV-799.
[9] D. Reynolds, W. Andrews, J. Campbell, J. Navratil, B. Peskin, A. Adami, Q. Jin, D. Klusacek, J. Abramson, R. Mihaescu, J. Godfrey, D. Jones, and B. Xiang, "The SuperSID project: Exploiting high-level information for high-accuracy speaker recognition," in *Proc. ICASSP*, 2003, pp. IV-784–IV-787.
[10] W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek, "Phonetic speaker recognition with support vector machines," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.
[11] W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek, "High-level speaker verification with support vector machines," in *Proc. ICASSP*, 2004, pp. I-73–76.
[12] H. Hollien, *Forensic Voice Identification*. New York: Academic, 2001.
[13] P. Rose, *Forensic Speaker Identification*. New York: Taylor & Francis, 2002.
[14] J. P. Campbell, D. A. Reynolds, and R. B. Dunn, "Fusing high- and low-level features for speaker recognition," in *Proc. Eurospeech*, 2003, pp. 2665–2668.
[15] N. Cristianini and J. Shawe-Taylor, *Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: IEEE Press, 1994.
[17] W. M. Campbell, "Generalized linear discriminant sequence kernels for speaker recognition," in *Proc. ICASSP*, 2002, pp. 161–164.
[18] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Comput. Speech Lang.*, vol. 20, pp. 210–229, 2006.
[19] L. Nguyen, S. Matsoukas, J. Davenport, F. Kubala, R. Schwartz, and J. Makhoul, "Progress in transcription of broadcast news using byblos," *Speech Commun.*, vol. 38, no. 1–2, pp. 213–230, 2002.
[20] M. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech Audio Process.*, vol. 4, no. 1, pp. 31–44, Jan. 1996.
[21] R. Collobert and S. Bengio, "SVMTorch: Support vector machines for large-scale regression problems," *J. Mach. Learn. Res.*, vol. 1, pp. 143–160, 2001.

[22] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," National Taiwan University, Tech. Rep., 2005 [Online]. Available: http://www.csie.ntu.edu.tw/cjlin/papers/libsvm.pdf
[23] C. Cortes, P. Haffner, and M. Mohri, "Rational kernels," in *Advances in Neural Information Processing Systems 15*, S. T. S. Becker and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003, pp. 601–608.
[24] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. Cambridge, MA: MIT Press, 1998, pp. 487–493.
[25] M. Przybocki and A. Martin, "The NIST Year 2003 Speaker Recognition Evaluation Plan," 2003 [Online]. Available: http://www.nist.gov/speech/tests/spk/2003/index.htm.
[26] T. Joachims, *Learning to Classify Text Using Support Vector Machines*. Norwell, MA: Kluwer, 2002.
[27] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[28] C. Cieri, D. Miller, and K. Walker, "The Fisher corpus: A resource for the next generations of speech-to-text," in *Proc. 4th Int. Conf. Lang. Resources Evaluation*, 2004, pp. 69–71.
[29] A. Hatch, B. Peskin, and A. Stolcke, "Improved phonetic speaker recognition using lattice decoding," in *Proc. ICASSP*, 2005, pp. 169–172.
[30] W. M. Campbell, "Compensating for mismatch in high-level speaker recognition," in *Proc. IEEE Odyssey: Speaker Lang. Recognition Workshop*, 2006.
[31] A. Solomonoff, W. M. Campbell, and I. Boardman, "Advances in channel compensation for SVM speaker recognition," in *Proc. ICASSP*, 2005, pp. 629–632.
[32] W. M. Campbell, K. J. Brady, J. P. Campbell, R. Granville, and D. A. Reynolds, "Understanding scores in forensic speaker recognition," in *Proc. IEEE Odyssey: Speaker Lang. Recognition Workshop*, 2006.
[33] T. Lander, R. A. Cole, B. T. Oshika, and M. Noel, "The OGI 22 language telephone speech corpus," in *Proc. 4th Eur. Conf. Speech Commun. Technol.*, 1995, pp. 817–820.
[34] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge, U.K.: Entropic, Ltd., 2002.
[35] A. Stolcke, "SRILM—An extensible language modeling toolkit," in *Proc. ICSLP*, 2002, pp. 901–904.
[36] C. Cieri, W. D. Andrews, J. P. Campbell, G. Doddington, J. J. Godfrey, S. Huang, M. Liberman, A. F. Martin, H. Nakasone, M. A. Przybocki, and K. Walker, "The mixer and transcript reading corpora: Resources for multilingual, crosschannel speaker recognition research," in *Proc. Int. Conf. Lang. Resources Evaluation (LREC)*, 2006, pp. 117–120.
[37] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," in *Proc. Eurospeech*, 1997, pp. 1895–1898.
[38] S. Matsoukas, R. Prasad, S. Laxminarayan, B. Xiang, L. Nguyen, and R. Schwartz, "The 2004 BBN 1xRT recognition systems for English broadcast news and conversational telephone speech," in *Proc. ICSLP*, 2004, pp. 1641–1644.

William M. Campbell (M'95) received the B.S. degrees in electrical engineering, computer science, and mathematics from the South Dakota School of Mines and Technology, Rapid City, in 1990, the M.S. and Ph.D. degrees in applied mathematics from Cornell University, Ithaca, NY, in 1993 and 1995, respectively.

From 1995 to 1999, he was a Senior Research Scientist at the Motorola Space and Systems Technology Group (SSTG), Speech and Signal Processing Lab. While at Motorola SSTG, he did research on biometrics, speech interfaces for wearable computing, and communications for the battlefield. From 1999 to 2002, he was a Principal Research Scientist in Motorola Labs, where he worked on machine learning, biometrics, and telematics. Since 2002, he has been a Staff Member of the Information Systems Technology Group, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, where he is involved in digital signal processing and machine learning for speech applications. He holds 12 patents.

Dr. Campbell received the Motorola Distinguished Innovator Award. He is a member of Tau Beta Pi and Eta Kappa Nu.

Joeseph P. Campbell (S'90–M'92–SM'97–F'05) received the B.S. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1979, the M.S. degree from The Johns Hopkins University, Baltimore, MD, in 1986, and the Ph.D. degree from Oklahoma State University, Stillwater, in 1992, all in electrical engineering.

He joined the MIT Lincoln Laboratory, Lexington, MA, in 2001 as a Senior Staff member specializing in speaker recognition and biometrics. He served 22 years at the National Security Agency (NSA) where he developed the first DSP-chip software modem and the LPC-10e and FS-1016 CELP voice coders that form the basis of government secure voice systems. As a senior scientist in the NSA's Biometric Technology research group, he led voice verification research and chaired the Biometric Consortium. Later as the Acoustics Section leader of the NSA's Speech Research branch, he conducted speech and language processing algorithm research and evaluation and introduced phone-based automatic speaker recognition methods. He also taught at The Johns Hopkins University.

Dr. Campbell was an Associate Editor of the IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, an IEEE Signal Processing Society Distinguished Lecturer, a member of the IEEE Signal Processing Society's Board of Governors, and a Coeditor of *Digital Signal Processing* journal. He is currently a Cochair of the International Speech Communication Association's Speaker and Language Characterization Special Interest Group and a member of the National Academy of Sciences' *Whither Biometrics?* Committee.

Terry P. Gleason, photograph and biography not available at the time of publication

Douglas A. Reynolds (SM'98) received the B.E.E. and Ph.D. degrees in electrical engineering, both from the Georgia Institute of Technology, Atlanta.

He joined the Speech Systems Technology Group (now the Information Systems Technology Group) at the Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, in 1992. Currently, he is a Senior Member of Technical Staff, and his research interests include robust speaker and language identification and verification, speech recognition, and general problems in signal classification and clustering.

Dr. Reynolds is a senior member of IEEE Signal Processing Society and a cofounder and member of the steering committee of the Odyssey Speaker Recognition Workshop.

Wade Shen received the B.S. degree in electrical engineering and computer science from the University of California, Berkeley, in 1994 and the M.S. degree in computer science from the University of Maryland, College Park, in 1997.

His current areas of research involve machine translation and machine translation evaluation, speech, speaker, and language recognition for small-scale and embedded applications, named-entity extraction, and prosodic modeling. Prior to joining the Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, in 2003, he helped found and served as Chief Technology Officer for Vocentric Corporation, a company specializing in speech technologies for small devices.